Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds.

A SIMULATION BASED HYBRID ALGORITHM FOR YARD CRANE DISPATCHING IN CONTAINER TERMINALS

Xi Guo
Shell Ying Huang
Wen Jing Hsu
Malcolm Yoke Hean Low

School of Computer Engineering
Nanyang Technological University
SINGAPORE, 639798

ABSTRACT

The problem of yard crane dispatching in container terminals is addressed in this paper. We proposed two new hybrid algorithms which combine the advantages of A* heuristic search and Recursive Backtracking with prioritized search order to accelerate the solution process. The algorithms proposed use real time data-driven simulation to accurately predict the time taken by the yard crane in performing its operations and this helps in getting an optimal dispatching sequence that can be followed by the yard crane. Experiments carried out show that the proposed algorithms consistently perform very well over all tested cases. The best performing algorithm is able to find the optimal solution over 2.4×10^{18} possible dispatching sequences in about 0.3 to 0.4 seconds under heavy workload. The characteristics of memory-saving and interruptibility enable the algorithm to be easily integrated into a complete yard crane management system in real world applications. In such real time yard crane management system, our proposed algorithms can be used as an effective and efficient tool to support complex and intelligent higher level planning in addition to managing the yard crane operations in its appointed zone.

1 INTRODUCTION

In today's business logistics, containerization has revolutionized cargo shipping and resulted in increasing global trade flow. Container terminals serve as crucial hubs in the transportation chain. In providing efficient service, minimizing vessel turnaround time is the common and key performance goal of terminal operations. When a vessel berths at a terminal, a number of Quay Cranes are allocated to serve the vessel in the process of unloading and then loading containers. Quay cranes first unload containers from vessel onto vehicles for transferring them to the storage yard area. A vehicle would arrive at a specific job location at a yard block and be served by a Yard Crane to pick up the container and temporarily store it in the yard block. The loading of containers onto a vessel is carried out in the reverse order.

One way to organize the yard operations is to partition a yard into a number of zones to be handled by individual yard cranes according to the predicted number of loading and unloading operations in the next planning window. For example, a planning window may be 1 hour or 2 hours. Each zone may be part of one yard block or more in the same row. Within one planning window, zones are static and non-overlapping. Figure 1 shows a possible partition of the storage yard into zones in a typical container terminal.

In a yard block, containers are arranged in a number of rows and slots, as shown in Figure 2. Vehicles travel along lanes as shown by the dotted lines for container jobs. A number of rows and slots next to each other will form a cluster. Containers unloaded from a vessel to be further loaded onto another vessel or to be transferred inland later will be temporarily stored in a few clusters distributed in the yard. So when a vessel is unloading, the vehicles will be going to different yard clusters. When multiple vessels are loading and unloading, vehicles will arrive at different slot locations. Local trucks carrying export containers may also arrive at any time to unload at some slot locations. As a result, yard cranes need to gantry between different slot locations of its appointed zone in serving vehicle jobs. Crane gantry times contribute to the vehicle waiting times. When a yard crane is busy serving other vehicle(s), a vehicle needs to wait for it to come and load/unload its container.

In this paper we address the problem of dispatching a yard crane in its appointed zone to minimize vehicle waiting times and therefore better support the continuous feeding of vehicles to the quay cranes and reduce vessel turnaround times. This improves the overall terminal throughput. The objective of yard crane dispatching is to determine a sequence for handling all

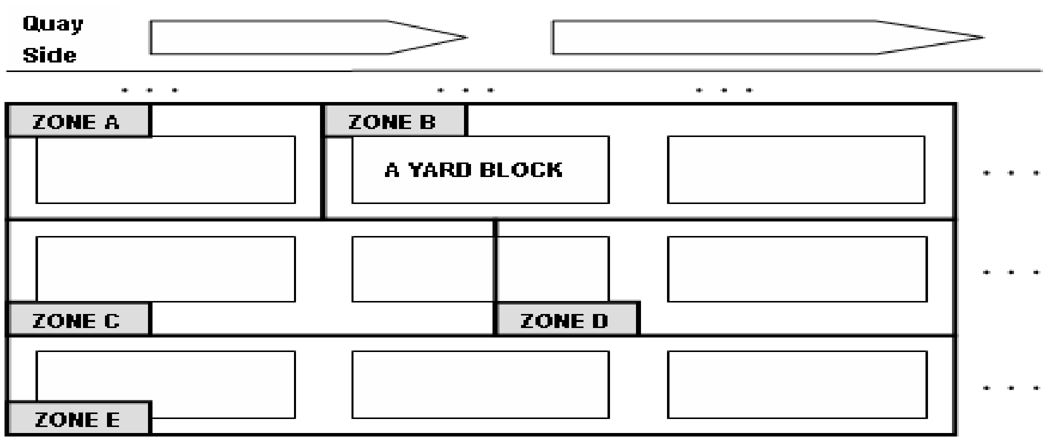

Figure 1: Storage yard partitioned into zones

incoming jobs within the zone that minimizes the average vehicle waiting time. It is a complex problem because both the job arrival times and the job locations may affect the dispatching sequence. For example, a job arriving later but nearer to the current job location could possibly be a better choice to be served next than a job arriving earlier but in a location further away because of the smaller gantry time. It means that the choice of the next job is affected by the previous yard crane location and crane available time. This makes techniques like dynamic programming not quite applicable. Secondly, an idle yard crane can pre-gantry to the chosen next job location before the actual job arrival to shorten vehicle waiting time if the next several job arrivals can be accurately predicted. Unlike servers in job shop problems, the yard crane can not only be reactive to actual job arrivals but also proactive to near arrivals.

Without the predictions of future vehicle job arrivals, two common policies used in container terminals for yard crane dispatching are First Come First Serve (FCFS) and Nearest Job First (NJF). FCFS simply serve the jobs according to their arrival sequence of the zone. It often incurs a lot of gantry movements and results in yard cranes spending significant amount of time on gantry. NJF could reduce the gantry time by always serving the job nearest to the yard crane's current position. However, it may lead to the starving of a job if it is at an isolated slot position. Furthermore, a common and serious shortcoming of the existing policies is that a yard crane would only start gantry towards a vehicle after it has arrived at a yard block. However, with technologies like DGPS, RFID and wireless communication, real time advanced information has great potential in helping to make optimal or near optimal decisions in job sequencing.

The sources of real time information include vehicle/container arrival information from terminal gates, container lists from planned berthing vessels, location information of traveling vehicles and yard crane status. Based on these, vehicle job arrivals in the near future could be predicted with high accuracy. This makes Pre-Gantry possible. Pre-Gantry is the ability of yard crane to start moving towards a next job location before the actual vehicle arrival. Once the next job location and arrival time is known, pre-gantry makes use of the crane idle time between jobs whenever possible so that the job waiting time for crane could be minimized. Greater potential in performance improvements would come from new dispatching algorithms using not only the information of next job but that of next several future jobs. These next several future jobs will form a planning window.

One difficulty of yard crane dispatching based on prediction information is the dynamics of job arrivals. Because terminal operations of quay cranes, yard cranes and transportation vehicles are inter-dependent and stochastic, actual vehicle job arrivals hardly follow a fixed schedule. For example, a vehicle job could be delayed due to possible delay of a quay crane oper-

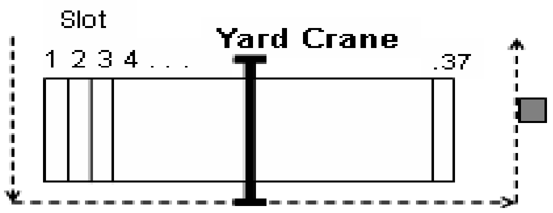

Figure 2: A yard Block with Slots

ation, a congestion encountered in vehicle travelling or human factors like individual driver behavior. This makes techniques like integer programming which requires job arrival information be completely known in advance not realistically feasible. However, with real-time tracking of terminal assets employing technologies including transponders, Radio Frequency ID gates (RFID), and Global Positioning System (GPS), it is possible to predict job arrivals in the near future with high accuracy. We study the yard crane dispatching problem where the planning period is small such that job arrival information can be predicted accurately.

Another difficulty of dispatching yard cranes based on prediction information is problem complexity. Even when the planning window is small, it could still be too time consuming to find the optimal solution by conventional methods including integer programming. The problem of yard crane dispatching has been proved to be NP-complete (Narasimhan and Palekar 2002), which means the time it takes to find the optimal dispatching solution increases exponentially with the problem size. A yard crane with 10 job requests would have over 3.6 million possible dispatching solutions. People have tried employing special method like Integer Programming (Kim and Kim 1999, Ng 2005) to explore all possibilities with smaller problem sizes than what we study here. Others proposed heuristic methods to reduce computation time by sacrificing optimism (Kim and Kim 2003; Lee, Cao, and Meng 2007), e.g. simulated annealing.

In our earlier study (Guo and Huang 2008), an A*search algorithm is proposed for yard crane dispatching problem which is guaranteed to find the optimal solution efficiently, e.g., be able to find the optimal over 2.4×10^{18} possible dispatching sequences in about 3 to 4 seconds under heavy work load. However this algorithm has two potential limitations in real world applications. First is the memory usage. It is well known that A*search usually runs out of memory space long before it runs out of time because it needs to keep all potential paths in memory in the search for the optimal. The number of nodes to be kept in memory will increase exponentially as the problem size increases. This characteristic limits its application in large size problems. The second concern is that real time dispatching often requires interruptible algorithms. Being interruptible means an algorithm could work out a solution very quickly and keeps improving it until the optimal is reached or interrupted by real time constraint to provide the current "best" solution. The previously proposed algorithm can find the optimal solution but is not able to provide a "current best" solution when interrupted. This characteristic could limit the ability of intelligent real time systems to trade deliberation time for quality of result.

Two new algorithms are proposed in this paper which overcomes the problem of massive memory usage and are interruptible with graceful degradation for integration to intelligent real time systems. At the same time, they guaranteed to find the optimal dispatching sequence efficiently. The new algorithms are hybrids of A* search and Recursive Back Tracking (RBT) with further improvement brought about by prioritized reordering of nodes to be searched. The best performing algorithm between the two guarantees to find the optimal solution for a single crane to handle the jobs in a planning window and are very fast to be used in real time environment. Experiment results show that it could find the optimal sequence out of 2.4×10^{18} possible dispatching sequences in about 0.2 to 0.4 seconds. It uses less computational time than the A* search method even though it explores a higher number of nodes and has a higher effective branching factor in the tree search.

The rest of the paper is structured as follows. In the next section we briefly review the related works. A formal problem description is given and the search space of the problem is transferred into a tree representation in Section 3. Our algorithm based on Recursive Back Tracking (RBT) with acceleration techniques for yard crane dispatching are presented in Section 4. Section 5 is dedicated to the experimental evaluation we have conducted to examine the performance of our proposed algorithms. Conclusion is drawn in Section 6.

2 OTHER RELATED WORK

The problems of scheduling and dispatching resources in logistics have been widely studied by researchers from different fields (Bramel and Simchi 1997). However, the results reported in the research literature are not directly applicable to a container terminal due to its unique characteristics (Ng and Mak 2005). In container terminals, inter-related operations and subsystem coupling make the scheduling problems very complex and dynamic.

Several focused studies on the yard crane dispatching or scheduling problem in container terminals have been carried out as it is of great importance to the overall terminal performance. Kim and Kim (1999) proposed a Mixed Integer Programming (MIP) method for the problem of routing single yard crane to support the loading operations of a vessel. Based on the MIP formulation, an optimal algorithm is presented. Kim and Kim (2003) presented heuristic algorithms for the same problem, which outperform a Genetic Algorithm (GA) shown by numerical experiments. However, the assumption in these works of having dedicated yard cranes just to support vessel loading operations is not always practical for terminals with many berths and more yard blocks than cranes.

Ng (2005) studied the problem of scheduling multiple yard cranes by modeling it as an integer program. A heuristic was proposed to minimize total loading time. However, loading sequence requirements not considered. Another heuristic approach was done by Lee, Cao, and Meng (2007). It addressed the problem of vessel loading with two yard cranes serving one quay crane. A simulated annealing algorithm is presented aimed at minimizing the total loading time. Computational experiments show that completion time of the proposed algorithm is on average 10.03% above a loosely estimated lower bound. Linn and Zhang (2002) built a MIP model to investigate dynamic yard crane deployment on a shift-to-shift basis to minimize the total unfinished work at the end of each shift. However the MIP model cannot produce the optimal solution in a timely manner. A heuristic was developed and their experiments show that the amount of overflow work is only 3~6% higher than the optimal solution found by MIP.

Due to the problem complexity, simulation has been one of the employed techniques to study the overall terminal systems. Shabayek and Yeung (2002) describe a simulation model to simulate the Kwai Chung container terminals in Hong Kong. They investigate the accuracy of predictions of actual terminal operations and conclude with good results.

3 PROBLEM FORMULATION

The objective of a yard crane is to serve every vehicle as soon as possible so that the vehicles will continuously feed the quay cranes so as to minimize vessel turnaround time.

3.1 General Description

In this work, we make the following assumptions in the yard crane dispatching model:

- Each vehicle job handles one container only.
- Future job information can be accurately predicted for a relative short planning period, e.g. 1 hour

The problem of dispatching a yard crane is stated as follows. There are N jobs in the current planning period with (predicted) vehicle arrival times A_i ($i = 1, 2, \dots, N$) and their respective job slot locations, the yard crane service time of job i is S_i and the crane gantry time from job slot x to job slot y is G_{xy}. What is the job sequence for the crane to serve the vehicle jobs such that the average waiting time for vehicles is minimized? Let F_i represents the completion time of job i ,the average waiting time for vehicles is given by

$$\frac{1}{N}\sum_{i=1}^{N}(F_i - S_i - A_i)$$

Generally when vehicle arrivals cannot be predicted reliably, the yard crane can only start to move towards a next job location after the actual job arrival. We define the job finishing time in this case by Equation (1). [*i*] and [*i+1*] denote the *i*th and the (*i+1*)th jobs in the crane's service sequence. Note that [*i+1*] may not be the job that arrives after [*i*] in the arrival sequence. If the arrival time and slot location of the next job is known beforehand, the yard crane could possibly start moving towards the next job location in advance of the actual vehicle arrival, which is referred to as the pre-gantry ability. We define the job finishing time with pre-gantry ability by Equation (2).

$$F_{[i+1]} = \max\{A_{[i+1]}, F_{[i]}\} + G_{[i][i+1]} + S_{[i+1]} \tag{1}$$

$$F_{[i+1]} = \max\{A_{[i+1]}, F_{[i]} + G_{[i][i+1]}\} + S_{[i+1]} \tag{2}$$

The advantage of the pre-gantry ability is the possibility of utilizing the yard crane idle time between jobs to transfer between different job locations.

3.2 Tree Representation

The yard crane dispatching problem is to find a dispatching sequence that minimizes the average waiting time of the vehicles. We represent the search space by a tree where each job is a node except the start node. An edge (a, b) in the tree has a weight (cost) equal to the vehicle waiting time for job b if the crane is to do job b after finishing job a. This edge weight, or the vehicle waiting time, W_b, is given by

$$W_b = F_b - S_b - A_b \tag{3}$$

where F_b, S_b and A_b are the job completion time, crane service time and vehicle arrival time respectively for job b. F_b is given by Equation (2) with [i] = a and [i+1] = b. We assume F_s=0, and the crane initial position at a randomly generated slot location. Figure 3 shows the search space of this tree representation.

For a dispatching problem of n jobs, each path from the start node to a leaf node in the tree represents a complete dispatching sequence of n jobs and there are in total n! such paths. So the complete search space is a tree of height n and n! leaf nodes. The objective of the problem is now transformed into finding the path which has minimum total cost from start to a leaf node.

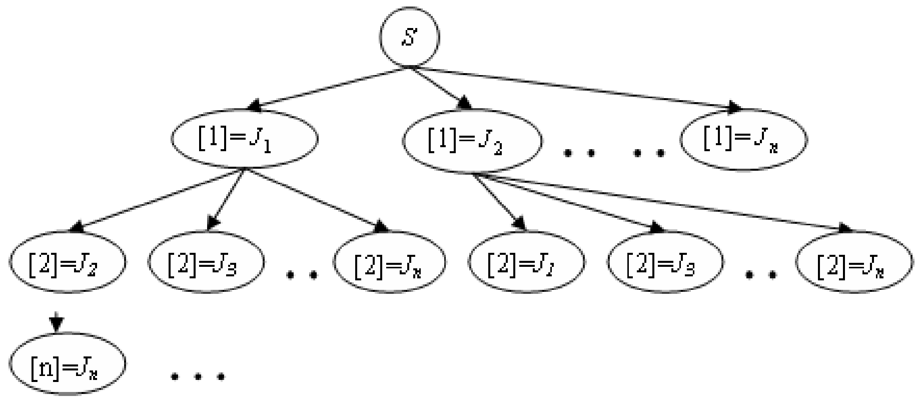

Figure 3: Search Space of the Problem

4 THE ALGORITHMS

Recursive Back Tracking (RBT) for tree search is complete and optimal if the depth of a search tree is finite and if there is no time constraint. A RBT based algorithm is interruptible as the current best solution will be provided even if the entire search is not finished due to real time constraint. However, it needs to traverse through the entire tree to find the optimal solution which would be time-consuming for such NP-Hard problems.

4.1 Hybrid of RBT and heuristics

We propose to employ heuristics to cut the search space. Every time we expand a new node along a search path, the job waiting time could be obtained and the total job waiting time from start to current node g(x) is updated. Let g(x) represent this cumulated cost of all planned jobs from start node till current node. The pruning of the search space could be more effective if the cheapest cost from the current node *x* to a goal node can be computed. Let h(x) represent the estimated minimum total job waiting time for all unplanned jobs from the current job node. If h(x) is admissible, in other words, h(x) never overestimates the cost from the current node to a goal node, the heuristic is by nature optimal because it thinks the cost of solving the problem is less than it actually is (Russell and Norvig 2003). Since g(x) is the exact cost from start node to the current node x, this implies that g(x)+h(x) never overestimates the true cost of a solution through node x. The decision to stop searching further the sub-trees of the current node could use the evaluation of g(x)+h(x), given by Equations (4) and (5). This evaluation function is proposed in our A* search paper (Guo and Huang 2008). By combining this heuristic into the RBT algorithm, we propose the RBTA* which will not miss the optimal solution and will greatly reduce the computation time of RBT. Pseudo-code is shown in Figure 4.

$$g(i) = \sum W_i, \qquad where\ i \in planned\_jobs \tag{4}$$

$$h(i) = \sum_{j=1}^{i-1} j\, S_j + \sum_{j=1}^{i-1} (A_i - A_j) + \sum_{j=i+i}^{M} Max\ \{0, F_i - A_j\} \tag{5}$$

where W_i is defined by (3).

```
// Unfinished Job List
$J = \{J_1, J_2, \dots, J_n\}$
YCDispatching (J) {
// This function is triggered when YC starts to serve the last job in the previous planning window
    Update J using real time predicted arrival information;
    newJ = $\phi$ ; optimalJ = $\phi$ ;
    CurSmallest = $\infty$;
    RBTA*(J, newJ);
}
RBTA*(J, newJ) {
    FOR each job $J_i \in J$
        Select Ji as the next job to serve after jobs in newJ;
        Append Ji to newJ;
        Remove this job from J;
        Simulate gantry time from last job in newJ to job Ji's and service time $S_i$ to obtain $W_i$ and compute
g(Ji) by (4);
        Estimate lower bound cost h(Ji) for jobs in J by (5) ;
        IF g(Ji)+h(Ji) is smaller than CurSmallest
            IF J is not empty
                RBTA*(J, newJ);
            ELSEIF TotalWaitingT is smaller than CurSmallest
                Update CurSmallest as TotalWaitingT;
                Update optimalJ = newJ; //Store Optimal List
    ENDFOR
}
```

Figure 4: Pseudocode of RBTA*

For dispatching a list J of jobs, the estimated arrival times for these jobs are updated using prediction algorithms which takes in the real time terminal information like the real time vehicle positions. In Figure 4, optimalJ is created to record the current best dispatching sequence. In each iteration of the FOR loop in RBTA*, an unplanned job Ji will be considered as the current job to be served by a yard crane. Dynamic data-driven simulation of the yard crane gantry and loading/unloading operation enables the computation of the cumulated job waiting time g(x) from the first job till the current job in the partially built dispatching sequence newJ. The minimum total job waiting time for jobs not planned yet in the partially built dispatching sequence is computed according to Equation (5). If the sum g(x)+h(x) which represents the estimated minimum total job waiting time for any dispatching sequence which has the partially built dispatching sequence as the prefix is smaller than that of the current best dispatching sequence, the function will be called to explore further the current path in the search tree. Otherwise, the current path is terminated and another unplanned job will be considered as the current job. If the current job is a leaf node, the algorithm has formed a dispatching sequence and the current best solution will be updated if necessary.

The estimated minimum cost h(x) from the current node to a leaf node needs to be as closer as possible to the real cost h*(x) so as to reduce unnecessary search. On the other hand, h(x) must never overestimate h*(x) to guarantee the finding of the optimal solution. In each iteration, picking job J_i as the current job results in a partial built dispatching sequence of i jobs and there are *n* - i jobs left unplanned. Therefore there are *factorial(n-i)* dispatching sequences with the partially built dispatching sequence as the prefix. Whenever g(x) + h(x) is found not to be better than the current best solution, a total of *factorial(n-i)* dispatching sequences are pruned from the search space. The h(x) function proposed involves simple computations and more details could be found in the paper (Guo and Huang 2008).

4.2 Acceleration with Prioritized Re-ordering

In the algorithm of RBTA*, unnecessary exploration to sub-trees is pruned according to the knowledge of the current best solution. In the worst case, the optimal route could be reached last and we could still be forced to explore the entire problem space. Therefore, discovery of a good path which has near optimal total job waiting time at the early stage of the tree search is crucial to the performance of the RBTA*.

In the pseudocode shown in Figure 4, the recursive algorithm takes as input a list J of jobs that have not been planned in the current dispatching sequence. For each job in list J, it will be considered as a next job J_i to do and followed by a function call for the remaining jobs *(J- {J_i})*. To accelerating the search process, we proposed a technique named prioritized re-ordering to produce a search order which is more likely to discover a good dispatching sequence early in the planning process, instead of choosing the next job randomly. Two re-ordering approaches are used and compared in this project:

1. The first approach is re-ordering the jobs according to their arrival time. The rational behind is that considering serving first the job which arrives earlier is more likely to minimize its waiting time and lead to an optimal or near-optimal path in minimizing the overall job waiting time. This improves the chances of finding an optimal or near-optimal path early and will help in pruning more sub-trees in the subsequent searches.
2. Another re-ordering approach is to sort the jobs by their reachable time of the yard crane in the zone as described in Equation (6).

$$\text{T\_Reachable}_{[i+1]} = \max\{A_{[i+1]}, F_{[i]} + G_{[i][i+1]}\} \tag{6}$$

For serving jobs at different slot locations of an appointed zone, the yard crane need to gantry between different job locations. If the next vehicle job [i+1] arrives earlier than the time the yard crane can gantry to its job location, the reachable time of the job will be yard crane's arrival time. If the yard crane can pre-gantry to the location of job [i+1] first, the reachable time of the job will be the vehicle's arrival time. Basically, the reachable time for [i+1] is its earliest possible start time for yard crane service after [i]. Note that the first reachable job from job [i]'s location may not be the one with the earliest arrival time among jobs in J as a job could be far away from job [i]. It should also be noted that among the jobs in J with their predicted arrival times, the first reachable job may not the one nearest to [i]. This is because the nearest job may arrive later than the time the yard crane could reach another job that arrives earlier. The rational behind this approach is that giving priority to serve the job that is first reachable by the yard crane improves the crane's throughput and therefore is likely to lead to an optimal or near-optimal solution.

These two approaches will be evaluated by our simulation experiments.

5 PERFORMANCE EVALUATION

5.1 Design of Experiments

To evaluate the performance of the proposed yard crane dispatching algorithms, simulation experiments were carried out. Parameter settings were obtained from real world terminal models. The linear gantry speed of a yard crane is 7.8km/hour. As reshuffling operation of containers in the yard storage is commonly done separately, we assume that containers to be retrieved are on the top of the stack at their specific slots and containers to be stored in yard will be placed on top of the stack of their slot locations. The yard crane service time is then assumed to be the same for all container jobs. Other recent studies using constant yard crane service time include Lee, Cao, and Meng (2007), Jung and Kim (2006) and Lee et al. (2006). The simulation model is programmed using C++ language under Visual C++ 6.0 complier on Pentium Core2 with 2.66GHz and 3GB of RAM.

Three sets of experiments were conducted where the yard crane is in charge of a zone of 1, 1.5 and 2 yard blocks respectively. Each block has 36 slots for container storage. For each set of experiments, three scenarios of different vehicle arrival rates were tested. The means of the exponential distributions of inter-arrival times are listed in Table 1 in unit of seconds. Heavy load means a job arrival rate slightly smaller than the yard crane's servicing rate (average crane service time + average inter job gantry time). Similar to Hoshino et al. (2005), Zeng and Yang (2009) and Kim, Park, Jin (2008), the slot locations of the vehicle jobs are generated randomly within the zone.

Table 1: Means of job inter arrival times in one zone

	Heavy Load	Normal Load	Light Load
Zone = 1 BLK	180s	240s	270s
Zone = 1.5BLKs	225s	270s	300s
Zone = 2 BLKs	270s	300s	330s

Planning windows of two different sizes are examined in the experiments, namely planning for 10 jobs and planning for 20 jobs respectively. For each experimental setting, 8 independent runs were performed and the results in the next section are the averages from these multiple runs. Six algorithms are implemented for studying and comparing the performance of RBTA* under different scenarios as listed in Table 2.

Table 2: Algorithms and their abbreviations

Abbreviation	Algorithms
RBTA*_Arl	RBTA* prioritized by arrival of jobs
RBTA*_Rcb	RBTA* prioritized by reachable of jobs
RBTA*_Ran	RBTA* choosing the next job randomly
A*	A* search(search in the order of g(x)+h(x))
PGS	Pure Greedy search(search in the order of g(x))
EXS	Exhaustive search

These include three variations of the RBTA* algorithms where prioritized re-ordering of jobs according to their arrival times, or according to their reachable times as defined by (6) or random ordering. These three variations are compared with A* search (Guo and Huang 2008), Pure Greedy search and Exhaustive search which is recursive depth first search. Theoretically, for a problem of finite number of jobs, all six algorithms are complete, that is, able to find the optimal dispatching solution given enough time and memory space. In addition, A* search is optimally effective.

5.2 Results Analysis

The quality of a yard crane dispatching algorithm is measured by two indicators: the computational cost of the dispatching plan and the quality of the dispatching plan. The quality of the dispatching plan is measured by the average job waiting time. For all algorithms compared and evaluated in the experiments, we are able to generate an optimal dispatching solution with minimum average job waiting time. The computational cost of a dispatching plan is measured by the time taken by an algorithm and by the effective branching factor in the tree search.

5.2.1 Computational Time for Planning Windows of 10 Jobs

The computational times for computing the optimal dispatching sequence for a planning window of 10 jobs are shown in Table 3. The Exhaustive Search (ExS) method is able to explore the entire search tree and find the optimal solution in about 85 seconds for the three zone sizes under the three workload cases. The Pure Greedy search (PGS), using the evaluation function *f(x) = g(x)*, performs better than ExS as it terminates the search along the branches of the search tree where the total job waiting time from the partially built dispatching sequence is already worse than the current best solution. . The A* search with the evaluation function *f(x) = g(x)+h(x)* is a significant improvement from Pure Greedy. The differences demonstrate the effectiveness of the admissible estimation of the minimum future cost h(x) from the current job to the end of the dispatching sequence.

Generally speaking, all three algorithms RBTA*_Arl, RBTA*_Rcb and RBTA*_Ran perform well. The improvements of these algorithms from those of the PGS are proofs of the effectiveness of applying good heuristics to further prune the search tree than what can be achieved by PGS. Among RBTA*_Arl, RBTA*_Rcb and RBTA*_Ran, the results confirm that the acceleration technique of prioritized re-ordering is very effective in improving the search time. For a planning window of 10 jobs, RBTA*_Arl and RBTA*_Rcb have similar timing and take comparable computational time as the A* which is optimally effective. Note that RBTA*_Arl and RBTA*_Rcb have significantly lower demand on memory space than A* and is interruptible to return the current best solution if the real time operational environment requires it.

Table 3: Computational time in seconds over various settings for length=10 jobs

Zone Size	Workload	RBTA*_Arl	RBTA*_Rcb	RBTA*_Ran	A*	PGS	ExS
1 BLK	HeavyL	0.01	0.01	1.34	0.01	0.63	85.13
	NormalL	0.02	0.02	1.26	0.01	0.24	85.05
	LightL	0.02	0.02	1.23	0.02	0.21	85.03
1.5BLKs	HeavyL	0.02	0.02	1.36	0.02	1.08	84.83
	NormalL	0.01	0.01	1.21	0.02	0.73	84.75
	LightL	0.01	0.01	1.18	0.02	0.61	84.57
2 BLKs	HeavyL	0.02	0.02	1.53	0.02	0.91	84.94
	NormalL	0.02	0.02	1.51	0.02	0.69	85.04
	LightL	0.02	0.02	1.30	0.02	0.59	84.97

5.2.2 Computational Time for Planning Windows of 20 Jobs

The performances of the algorithms are further evaluated for a longer planning window of 20 jobs to test their scalability.

When a search algorithm cannot get the optimal result within one hour we interpret it as running out of time. Experiments show that in all cases tested exhaustive search for a list of 20 jobs would run out of time. Exhaustive search is implemented by recursive depth first search. As the problem tree is finite and all branches have same depth, the search is complete and optimal. For the recursive depth first implementation, only current path information is kept in the system with the max of 20 jobs in the path. Therefore exhaustive search does not have the running out of memory problem. However, taking more than 1 hour to find the optimal solution is hardly acceptable for a planning window of around 1~2 hours. RBTA*_Ran is also considered as running out of time as in the worst case, it might need to explore the entire search space.

The pure greedy search runs out of memory in all cases tested for a list of 20 jobs. Employing the evaluation function f(x) = g(x), the pure greedy search runs out of memory because it is keeping too many partially evaluated dispatching sequences in the memory. As this algorithm could be equivalently considered as an A*search algorithm whose h(x) is always equal to zero, it also shows that g(x), the current accumulated waiting time from the partial dispatching sequence is a very poor estimate of the actual cost $h^*(x)$.

The remaining three best performing algorithms are compared in Table 4. It shows that the RBTA* with the two reordering methods performs well in all experimental settings when compared against the optimally effective A* search. Although they are likely to explore more nodes in the search tree than A* method, they do not need to keep the queue for deciding which node should be visited next as in the A*search. The overhead in maintaining this queue in every node expansion takes additional computational time. RBTA* prioritized by the arrival times of jobs (RBTA*_Arl) performs slightly better than RBTA* prioritized by reachable times of the yard crane (RBTA*_Rcb). This suggests that RBTA*_Rcb has the risk of favouring a partial dispatching sequence which may lead to starvation of a far away job. However this will only be detected when the current dispatching sequence is almost complete, wasting search time. These results show that for this larger planning window of 20 jobs, even in the heavy workload case, RBTA* can find the optimal solution out of over 2.4×10^{18} possible dispatching sequences in about 0.3 seconds. This is good enough even to be used in real time environment. Note that when the problem size increases further, A* search might also face the potential risk of running out of memory even though it has not happened for this window size.

Table 4: Computational time over various settings for length=20 jobs

Zone Size	**Workload**	**RBTA*_Arl**	**RBTA*_Rcb**	**A***
1 BLK	HeavyL	0.26	0.35	1.84
	NormalL	0.07	0.10	0.38
	LightL	0.04	0.05	0.18
1.5BLKs	HeavyL	0.36	0.52	2.39
	NormalL	0.24	0.40	1.98
	LightL	0.12	0.18	0.84
2 BLKs	HeavyL	0.43	0.54	13.46
	NormalL	0.41	0.64	6.44
	LightL	0.14	0.24	1.27

In the planning of real world container terminals, higher level planning need to be done ahead of time in determining the allocation of yard cranes to different rows of yard blocks and the partition of zones in a row that a yard crane will be in charge in a planning window. These planning activities involve the balancing and estimating of the workload of individual rows of yard blocks and zones over various possible cases which might need running many of such lower level simulation-planning scenarios for single yard cranes. Our proposed algorithms would be a solid tool to support the complex and advanced higher level planning for problems of large terminals.

5.2.3 Effective Branching Factor

One way to characterize the quality of a heuristic is the effective branching factor b*. If the total number of nodes generated for a particular problem is N, and the solution depth is d, then b* is the branching factor that a uniform tree of depth d would have to have in order to contain N+1 nodes. Thus

$$N + 1 = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d$$

The effective branching factor can vary across problem instances, but usually it is fairly constant for sufficiently hard problems. Therefore, experimental measurements of b* on a small set of problems can provide a good guide to the heuristic's overall usefulness (Russell and Norvig 2003). Comparison of the search cost and effective branching factors for RBTA* prioritized by random order, Pure Greedy Search, A* search, RBTA* prioritized by arrival of jobs and RBTA* prioritized by reachable of jobs are shown in Table 5.

Table 5: Comparison of Search Cost and Effective Branching Factor among various algorithms

	Search Cost (the number of nodes explored)					Effective Branching Factor				
d	RBTA* _Ran	PGS	A*	RBTA* _Arl	RBTA* _Rcb	RBTA* _Ran	PGS	A*	RBTA* _Arl	RBTA* _Rcb
10	96439	5215	55	97	101	3.02	2.22	1.30	1.40	1.41
12	787812	35574	119	227	232	2.99	2.28	1.33	1.42	1.43
14	---	238655	200	368	403	---	2.33	1.32	1.39	1.41
16	---	---	409	746	823	---	---	1.34	1.40	1.41
18	---	---	781	1500	1500	---	---	1.34	1.40	1.40
20	---	---	1687	3216	3284	---	---	1.36	1.41	1.42

Results show that RBTA*_Ran runs out of acceptable time for planning windows with the number of jobs greater than or equal to 14. Pure Greedy method run out of memory in the current experimental settings when the number of jobs in the planning window reaches 16. The closer the effective branching factor is to 1, the smaller the number of nodes are explored to find the optimal path. The two RBTA* methods with prioritized re-ordering do have results worse than the A* method in terms of the number of nodes explored in the search. This also confirms that A* is optimally effective. However, the much lower computational overhead of the RBTA* methods as discussed in the last section more than compensates for the higher effective branching factor.

6 CONCLUSION

In this paper, we address the problem of yard crane dispatching in container terminals. We proposed two new hybrid algorithms which combine the advantages of A* heuristic search and RBT with prioritized re-ordering to accelerate the search process. The algorithms proposed use real time data-driven simulation to accurately predict the time taken by the yard crane in performing its operations and this help in getting an optimal dispatching sequence can be followed by the yard crane. Experiments were carried out to evaluate the algorithms under three various workload cases in three different sized yard zones and two planning windows with different number of jobs. Results show that the proposed algorithms consistently perform very well over all tested cases. Two alternatives, pure greedy search fails to find the solution and the exhaustive search takes too long to be practical in dispatching jobs of length 20. Our RBTA* algorithm with the prioritized re-ordering acceleration is able to find the optimal solution over 2.4×10^{18} possible dispatching sequences in about 0.3 to 0.4 seconds under heavy work load. This also suggest that in the situation where if there is some unexpected update/change in job arrival information in real time, the dispatching sequence may be re-computed without delaying the crane operations. The characteristic of memory-saving and interruptibility enables the algorithm to be easily integrated into a complete yard crane management system in real world applications. In such real time yard crane management system, our proposed algorithms can be used as an effective and efficient tool to support complex and advanced higher level planning in addition to managing the yard crane operations in its appointed zone.

REFERENCE

Bramel, J., and D. Simchi-Levi. 1997. *The Logic of logistics: theory, algorithms, and applications for logistics management*. Springer-Verlag, New York.

Guo, X., and S.Y. Huang. 2008. Performing A* search for yard crane dispatching in container terminals. In *Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence*. 1:263-267.

Hoshino, S., J. Ota, A. Shinozaki, and H. Hashimoto. 2005. Highly efficient AGV transportation system management using agent cooperation and container storage planning. *IEEE/RSJ Int. Conf. Intell. Robots and Syst*. p.1588-1593.

Jung, S.H., and K.H. Kim. 2006. Loading scheduling for multiple quay cranes in port container terminals. *Journal of Intelligent Manufacturing* 17:479-492.

Kim, K.H., Y-M. Park, and M.-J. Jin. 2008. An optimal layout of container yards. *OR Spectrum* 30:675-695.

Kim, K.H., and K.Y. Kim. 1999. An optimal routing algorithm for a transfer crane in port container terminals. *Transportation Science* 33 (1): 17–33.

Kim, K.Y. and K.H. Kim. 2003. Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals. *Naval Research Logistics* 50: 498–514.

Lee, D-H., Z. Cao, Q. Meng. 2007. Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm. *Int J Product Econ* 107:115–124.

Lee, L.H., E.P. Chew, K.C. Tan, and Y.B. Han. 2006. An optimization model for storage yard management in transshipment hubs. *OR Spectrum* 28:539-561.

Linn, R.J. and C.Q. Zhang. 2003. A heuristic for dynamic yard crane deployment in a container terminal. *IIE Transactions* 35(2):161-174.

Narasimhan, A., and U.S. Palekar. 2002. Analysis and algorithm for the transtainer routing problem in container port operation. *Transportation Science* 36 (1), 63–78.

Ng, W.C. 2005. Crane scheduling in container yards with intercrane interference. *European Journal of Operational Research* 164: 64–78.

Ng, W.C., and K.L. Mak. 2005. Yard crane scheduling in port container terminals. *Applied Mathematical Modelling* 29: 263-275.

Russell, S., and P. Norvig. 2003. *Artificial Intelligence: A Modern Approach*. 2nd Edition. Pearson Education, Inc., Upper Saddle River, NJ.

Shabayek, A.A., and W.W. Yeung. 2002. A simulation for the Kwai Chung container terminal in Hong Kong. *European Journal of Operational Research* 40: 1–11.

Zeng, Q., and Z. Yang. 2009. Integrating simulation and optimization to schedule loading operations in container terminals. *Computers and Operation Research* 36:1935-1944.

AUTHOR BIOGRAPHIES

XI GUO is a Ph.D. student at Nanyang Technological University(NTU), School of Computer Engineering(SCE), Singapore. Her research interests include real time control, artificial intelligence, and efficiency enhancement techniques for simulation-based optimization. Her email address is <guox0006@ntu.edu.sg>.

SHELLYING HUANG is an associate professor of Computer Science in SCE at NTU (Singapore). Her research interests are in intelligent decision support systems, intelligent agents and agent-based simulations. Her email address is <assyhuang@ntu.edu.sg>.

WEN JING HSU is an associate professor at SCE, NTU. His research interests include parallel and distributed processing and Provable efficient algorithms. His email address is <hsu@ntu.edu.sg>.

MALCOLM LOW is an assistant professor at SCE, NTU. His research interests include parallel computing, modelling and simulation, planning and scheduling optimization. His email address is <yhlow@ntu.edu.sg>.