Performing A* search for yard crane dispatching in container terminals

Guo, Xi; Huang, Shell Ying

2008

Guo, X., & Huang, S. Y. (2008). Performing A* search for yard crane dispatching in container terminals. 20th IEEE International Conference on Tools with Artificial Intelligence, 2008. ICTAI '08.

<https://hdl.handle.net/10356/79824>

<https://doi.org/10.1109/ICTAI.2008.77>

© 2008 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE. This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder. <http://www.ieee.org/portal/site> This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.

Performing A* Search for Yard Crane Dispatching in Container Terminals

Xi Guo and Shell Ying Huang
School of Computer Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798
guox0006@ntu.edu.sg, ASSYHUANG@ntu.edu.sg

Abstract

We address the problem of dispatching a yard crane in its appointed zone of a container terminal. The objective of yard crane dispatching is to determine a sequence for handling all coming jobs within the zone that minimizes the average vehicle waiting time. We propose 2 modified A* search algorithms with admissible heuristics to provide fast and optimal dispatching solution. Simulation results show that the algorithm using the improved heuristic is able to find the optimal solution over 2.4×10^{18} possible dispatching sequences in about 3 to 4 seconds under heavy work load.

1. Introduction

When a vessel berths at a terminal, a number of Quay Cranes first unload containers from the vessel onto vehicles. Vehicles then travel to specific locations in the yard to be served by Yard Crane for temporary storage of containers in the yard. Then the loading of the vessel is carried out in the reverse order.

One way to organize the yard operations is to partition a yard into a number of non-overlapping zones to be handled by individual yard cranes in a planning window. Each zone may be part of one yard block, one yard block, or more than one yard blocks in the same row. 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 and wait at specific slot location. Yard cranes may need different amount of time to gantry between different slot locations in serving vehicle jobs and crane gantry times contribute to the vehicle waiting times. When a yard crane is busy serving other vehicle(s), a vehicle also needs to wait for the crane.

Figure 1. Storage yard partitions into zones

Figure 2. A yard Block with Slots

In this paper we address the problem of dispatching a yard crane in its appointed zone to minimize vehicle waiting times and therefore to better support the continuous feeding of vehicles to the quay cranes and reduce vessel turnaround times. It is a complex problem with unique characteristics. The most important one is the each job's waiting time and thus its finish time is affected by both the previous yard crane location and crane available time. This makes techniques like dynamic programming not quite applicable. An factor is an idle yard crane's pre-gantry ability may shorten a vehicle waiting time. Unlike servers in job shop problems, the yard crane can not only be reactive to actual job arrivals but also proactive to near arrivals.

One difficulty of yard crane dispatching is the dynamics of job arrivals. 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 operation, a congestion encountered in vehicle travelling or human factors like individual driver behavior. This makes techniques like integer programming which requires job arrival information

completely known in advance not realistically feasible. However, with real-time tracking of terminal assets, it is possible to predict with high accuracy job arrivals in near future. We study the yard crane dispatching problem where the planning period is small such that job arrival information can be predicted accurately.

Even when the planning window is small, it is still too time consuming to find the optimal solution by conventional methods. The problem of yard crane dispatching has been proved to be NP-complete [3]. People have tried employing special machine like Integer Programming [2, 4] to explore all possibilities with smaller problem sizes than what we study here. Others proposed heuristic methods to reduce computation time by sacrificing optimism [5, 8], e.g. simulated annealing.

In this paper, we propose 2 algorithms which are guaranteed to find the optimal solution for a single crane in a planning window and are very fast to be used in real time environment. The proposed A* algorithms are admissible, complete and optimally effective [1].

The rest of the paper is structured as follows. Section 2 briefly reviews the related works. A formal problem description is given in Section 3. Our modified A* search algorithms for yard crane dispatching are presented in Section 4. Section 5 presents the experimental evaluation of the performance of our proposed algorithms. Conclusion is drawn in Section 6.

2. Related Work

Several focused studies on the yard crane dispatching problem in container terminals have been done. Kim and Kim [2] proposed a Mixed Integer Programming (MIP) method for the routing of a single yard crane to support the loading operations of a vessel. Based on the MIP formulation, an optimal algorithm is presented. Kim and Kim [5] 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 [4] 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 is not considered.

A* search has been widely adapted in various applications with domain specific heuristics. For examples, Hamdi and Mulvaney [6] proposed a

Prioritized A* algorithm in solving real-time elevator dispatching problem; Lee and Cho [7] presented an application of the A* algorithm in grating manufacturing layout design. To the best of our knowledge, our work is the first to apply the A* technique to solve the yard crane dispatching problem in container terminals.

3. Problem Formulation

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 2 hours

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 , completion time of job i is F_i and the crane gantry time from job x to job y is G_{xy} . We represent the search space by a tree where each job is a node except the start node. An edge (h, i) in the tree has a weight (cost) equal to the vehicle waiting time for job i if the crane is to do job i after finishing job h . This edge weight, W_i , is given by Equation (1)

$$W_i = F_i - S_i - A_i \quad (1)$$

The objective of minimizing average waiting time for vehicles is expressed as

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

When the arrival time and slot location of the next job is known beforehand, the yard crane could start moving towards the next job location in advance of the actual vehicle arrival. Job finishing time with this pre-gantry ability is shown in Equation (2).

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

$[i]$ and $[i+1]$ denote the i th and the $(i+1)$ th jobs in the crane's service sequence.

Figure 3. Search Space of the Problem

Search space of the tree representation of yard crane dispatching problem is shown in Figure 3. 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 height n and there are in total $n!$ such paths. The objective of the problem is now transformed into finding the path which has minimum total cost from start to a leaf node.

4. Optimal Search Algorithms

We propose a modified A* search algorithm to find the optimal solution with 2 admissible heuristic functions to estimate the cost from a current node to a leaf node.

4.1 Modified A* search

Pseudocode of the Modified A* for yard crane dispatching is shown in Figure 4. In the calculation for the cost function of nodes, let

$N = \text{set of all jobs, } |N| = n$

$P(x) = \text{set of jobs already in the (partial) path from the root of the tree to } x$

$U(x) = N - P(x)$

$h(x)$ is the estimated lower bound cost of the path from node x to a leaf node. $g(x)$ is now cumulated job waiting time from start to node x with W_i computed by Equation (3):

$$g(x) = \sum W_i, \text{ where } i \in P(x). \quad (3)$$

Modified A* Search

```
// the cost function  $f(x)$  determines the priority of node  $x$ 
Create a node containing start status as  $node\_start$ 
Create a node containing infinite cost as  $node\_best$ 
Put  $node\_start$  in the priority queue  $q\_open$ 
WHILE  $q\_open$  is not empty
     $node\_cur = \text{remove\_first}(q\_open)$ 
    IF  $f(node\_cur)$  is no better than  $f(node\_best)$ 
        Return  $node\_best$ 
    IF  $node\_cur$  is a leaf node
        Update  $node\_best$  as  $node\_cur$ 
    ELSE
        Generate each  $node\_successor$  of  $node\_cur$ 
        FOR each  $node\_successor$   $x$ 
            Get true cost from start to this node  $g(x)$ 
            Estimate lower bound cost from this node  $h(x)$ 
            Get cost function  $f(x) = g(x) + h(x)$ 
            Add  $x$  node to priority queue  $q\_open$ 
```

Figure 4. Pseudocode of Modified A* Search

Unlike the original A* search, we are not actually looking for a “goal” state in the yard crane dispatching problem. The objective of our tree searching is to find a path (dispatching sequence) with minimum cost (e.g. total job waiting time). The search stops when a path to a leaf node x that has an $f(x)$ value lower than that of any path in the open list or until the open list is empty.

4.2 The First Admissible Heuristic $h(x)$

In this problem, an admissible heuristic means $h(x)$ never overestimates the cost to handle the remaining jobs not in the path from root to x . Now, the problem is how to evaluate $h(x)$ where x is a successor node of $node_cur$ in the algorithm presented in Figure 4. We illustrate this situation in Figure 5. All successors of $node_cur$, that is, all jobs in $U(node_cur)$, are to be put in a list L and re-indexed, where $A_i \leq A_{i+1}$, for $i = 1, 2, \dots, M-1$, $M = |U(node_cur)|$. Let

Figure 5. Jobs in $U(node_cur)$ re-indexed and generated as children of $node_cur$.

Consider $x = J_i$. To compute $h(x)$, the cost from J_i to a leaf node, we need to estimate the total job waiting time for the remaining jobs, that is, jobs in $U(x)$ where $U(x) = U(node_cur) - \{J_i\}$. $U(x)$ can be partitioned into two groups:

$$U(x = J_i) = C_{earlier} + C_{later}$$

$$C_{earlier} = \{J_j\}, \text{ where } j = 1, 2, \dots, i-1$$

$$C_{later} = \{J_j\}, \text{ where } j = i+1, i+2, \dots, M$$

Jobs in $C_{earlier}$ are the ones that arrive earlier than J_i , therefore it is certain that these jobs have to wait for the service of J_i according to the current dispatching sequence. If we approximate the crane service time for all jobs by a constant value of T_s , then it is for sure that the first job to be handled in $C_{earlier}$ has to wait for at least T . The second job to be handled in $C_{earlier}$ will have to wait for at least two T_s , the sum of the job service times of J_i and the first job handled in $C_{earlier}$. Likewise, the last job to be handled in $C_{earlier}$ will have to wait for at least $(i-1) T_s$, the sum of the job service times for J_i and the previous $(i-2)$ jobs in $C_{earlier}$. If a job in C_{later} is to be served before a job in $C_{earlier}$, its waiting time will be even longer. Thus, the minimum total waiting time for jobs in $C_{earlier}$ due to the time the crane spent after the arrival of J_i on servicing other jobs is described in Expression (4).

$$[1 + 2 + \dots + (i-1)] * T_s = \sum_{j=1}^{i-1} j T_s \quad (4)$$

In addition, for each job in $C_{earlier}$, it definitely needs to wait for the time period from its arrival to the

arrival of J_i as J_i will be served first by the yard crane among all successors of $node_cur$. The minimum total waiting time for jobs in $C_{earlier}$ till the arrival of J_i is

$$(A_i - A_1) + \dots + (A_i - A_{i-1}) = \sum_{j=1}^{i-1} (A_i - A_j) \quad (5)$$

The heuristic $h(x = J_i)$ is then the sum of these two above components:

$$h(i) = \sum_{j=1}^{i-1} jT_s + \sum_{j=1}^{i-1} (A_i - A_j)$$

Estimated minimum cost from node J_i to a leaf node is now labelled as $h(i)$, where J_i is the i th job arrival among all the successors of $node_cur$. It is not difficult to prove that $h(i)$ is admissible.

4.3 An Improved Heuristic $h(x)$

The closer a heuristic estimation $h(x)$ is to the actual cost $h^*(x)$, the better the performance of A* search will be. The heuristic function presented in the last section estimates the job waiting times for those jobs which arrive earlier than $x = J_i$. We could improve the heuristic further.

Figure 6. Time Line of Successor Jobs of $node_cur$. A_i is the arrival time of J_i .

As discussed in the last section, the successor jobs of $node_cur$ excluding J_i are arranged by their arrival order and partitioned into two groups, $C_{earlier}$ containing the jobs that arrive earlier than J_i and C_{later} containing the jobs that arrive later than J_i . The waiting times for jobs in $C_{earlier}$ is estimated by expression (5) and (4) representing the waiting times incurred before A_i and after A_i respectively. We further tighten the difference between the heuristic $h(x)$ and the actual cost $h^*(x)$ by incorporating estimates for the waiting times of jobs in C_{later} . We note that some jobs in C_{later} may arrive earlier than the completion time of J_i . An example is J_{i+1} in Figure 6 with arrival time A_{i+1} earlier than F_i . This will certainly result in a waiting period $F_i - A_i$. For jobs in C_{later} , these waiting times are estimated in expression (6). This component could possibly result in zero when all the jobs in C_{later} arrive later than F_i .

$$\sum_{j=i+1}^m \text{Max} \{0, F_i - A_j\} \quad (6)$$

The improved heuristic combines this component and the previous two as follows.

$$h(i) = \sum_{j=1}^{i-1} jT_s + \sum_{j=1}^{i-1} (A_i - A_j) + \sum_{j=i+1}^m \text{Max} \{0, F_i - A_j\}$$

The heuristic function is expected to provide better estimates of the actual costs $h^*(x)$ which will reduce further the computational time for the optimal dispatching sequence. It is still admissible as the third component added can never overestimate the cost contributed by the jobs in C_{later} .

5. Performance Evaluation

5.1 Experimental Design

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. Each experiment computes 20 jobs corresponding to a planning period of 1 to 3 hours depending on job inter-arrival times. Intra-block speed of the crane is 7.8km/hour. Inter-block gantry between two adjacent blocks of the same row takes 200 seconds and the yard crane service time is taken as a constant of 180 seconds for all jobs. Two sets of experiments were conducted with zones of sizes of 1 and 2 yard blocks respectively. Three scenarios of different vehicle arriving 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 (crane service time + average inter job gantry time). The slot locations of the jobs are generated randomly within the zone.

Four algorithms evaluated are: A* with the first heuristic (H1), A* with improved heuristic (H2), pure greedy search and exhaustive search. Pure greedy is a search algorithm by expanding the node whose current cumulated waiting time is minimum using $f(x) = g(x)$. Exhaustive search is implemented by depth first search. As the problem tree is finite and all branches have same depth, the search is complete and optimal. For each setting, 15 independent runs were performed and results are the averages from these multiple runs.

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

	Heavy Load	Normal Load	Light Load
Zone = 1 BLK	240s	300s	420s
Zone = 2 BLKs	390s	480s	570s

5.2 Results Analysis

The quality of a yard crane dispatching algorithm is measured by two indicators: the time to compute the dispatching plan and the quality of the dispatching plan. For all algorithms used in the experiments, we are looking for the optimal dispatching solution with minimum average job waiting time.

Data in Table 2 show that the A* algorithm with heuristic H2 takes less computational time to find the optimal solution than H1 when the zone size is 1 block under all three work load cases. It means H2 expands fewer nodes than H1 for the same problems. This suggests that the heuristic used in H2 is able to generate an estimation of the total waiting time closer to the actual total waiting time than H1. These results show that A* search can produce optimal crane dispatching sequence with very low computational costs. Even in the heavy load case, H2 can find the optimal solution out of over 2.4×10^{18} possible dispatching sequences in about 3 seconds. This is good enough even to be used in real time environment.

Table 2. Computational Time of Zone = 1BLK/2BLKs

1BLK/2BLKs	Heavy Load	Normal Load	Light Load
A*_H1	6.36/4.72	2.54/3.30	0.47/0.77
A*_H2	3.12/2.72	1.30/1.07	0.31/0.54
Pure Greedy	Run Out of Memory		
Exhaustive Search	Run Out of Time (>7200 sec)		

Pure greedy search run out of memory in all cases tested. It is because the algorithm keeps too many partially evaluated crane dispatching sequences in the memory when employing the heuristic $f(x) = g(x)$. This shows that $g(x)$, the current accumulated waiting time from the partial dispatching sequence, is a very poor estimation to the actual cost $h^*(x)$. When a search algorithm cannot get the optimal results within two hours we interpret it as running out of time, because such amount of time is not acceptable in real time operational management. Experiments show that in all cases tested exhaustive search would run out of time.

When the work load of vehicle job arrivals is heavy, Modified A* search algorithms of both H1 and H2 need to take more time to work out the optimal solution. This can be explained using queuing theory: the closer job arrival rate to the service rate (yard crane handling rate) is, the higher probability is to see queuing jobs. When several vehicle jobs wait for the yard crane service, their relative job locations would be an important factor in determining the optimal service order. The algorithms need to expand and evaluate more partial job sequences in finding the optimal, resulting in longer computational times.

Similar results have been obtained in the zone with 2 blocks. Comparing results across two zone sizes, we find H2 always performs better than H1 under different work load cases. We could conclude that H2 is more effective than H1. A* search with Heuristic H2 is fast enough to be used in real time environments to find the optimal solution.

6. Conclusion

In this paper, we address the problem of yard crane dispatching in container terminals. Modified A* search algorithms with 2 admissible heuristics are proposed to compute optimal dispatching solutions. Experiments were carried out to evaluate the algorithms under three various work load cases in two yard zones of different sizes: 1BLK and 2BLKs. Results show that the proposed algorithms consistently perform very well over all tested cases. Our modified A* search with the improved heuristic is able to find the optimal solution over 2.4×10^{18} possible dispatching sequences in about 3 to 4 seconds under heavy work load. This actually suggest that in the situation where if there is some update/change in job arrival information in real time, the dispatching sequence may be re-computed without delaying the crane operations.

7. References

- [1] Dechter, R. and Pearl, J. Generalized best-first search strategies and the optimality of A*, JACM, 32(3): 505-536, 1985.
- [2] Kim, KH. and Kim, KY. An optimal routing algorithm for a transfer crane in port container terminals. Transportation Science 33 (1), 17-33. 1999
- [3] Narasimhan, A., Palekar, U.S. Analysis and algorithm for the transtainer routing problem in container port operation. Transportation Science 36 (1), 63-78. 2002
- [4] Ng, W.C. Crane scheduling in container yards with intercrane interference. European Journal of Operational Research 164, 64-78. 2005
- [5] Kim, KY. and Kim, KH. Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals. Naval Research Logistics 50, 498-514. 2003
- [6] Hamdi, M. and Mulvaney, D.J. Prioritized A* search in real-time elevator dispatching. Control Engineering Practice, 15(2), pp. 219-230. 2007
- [7] Lee, H.Y., and Cho, T.H. A*-based layout design for gratings allocation. CAD Computer Aided Design, 40(4), pp. 455-464. 2008
- [8] Lee, D.-H., Cao, Z., Meng, Q. Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm. Int J Product Econ 107:115-124. 2007