A Two Stage Yard Crane Workload Partitioning and Job Sequencing Algorithm for Container Terminals

Xi Guo  
School of Computer Engineering  
Nanyang Technological University  
Block N4, Nanyang Avenue, Singapore 639798  
guox0006@ntu.edu.sg

Shell Ying Huang  
School of Computer Engineering  
Nanyang Technological University  
Block N4, Nanyang Avenue, Singapore 639798  
assyhuang@ntu.edu.sg

ABSTRACT

We propose a new YC workload partitioning scheme which employs dynamic data driven simulations to conduct what-if experiments in container terminals. Both the optimal partition of the workload in a row of yard blocks and the optimal dispatching sequences for individual YCs are achieved. The practical consideration of the safety constraints is included. A dynamic programming (DP) approach is used to avoid re-computation. An efficient two stage workload partition algorithm (TSWP) is proposed which successfully reduces the number of full what-if simulations while maintaining solution optimality. An effective lower bound (LB) generator with adjustable LB accuracy is designed in supporting the TSWP algorithm. Experimental results show that the TSWP algorithm outperforms the pure DP approach in all tested scenarios and takes less than 1 part per thousand computational time of the DP approach.

Categories and Subject Descriptors
1.6.7 [Simulation and Modeling]: Simulation and Modeling – simulation support system.
1.2.8 [Artificial Intelligence]: Problem Solving, Control Methods and Search – scheduling.

General Terms
Algorithms, Performance, Experimentation

Keywords
Optimization, Yard Crane Dispatching, Decision-Making

1. INTRODUCTION

Today, about 80~90% of global trade by volume is shipped in containers. As of August 2009, there exist 4686 container vessels with the capacity equal to 12.8 million TEUs [1]. Container ports, which serve as hubs of container transshipment, are crucial nodes in the marine transportation network. The demand on high quality services from container terminals includes efficiency and reliability in container handling which in turn requires the port to utilize its resources efficiently.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Container terminals are open systems of material flow. Figure 1 shows a large part of a typical layout of a container terminal. Usually in conventional terminals, the storage yard is divided into several tens of yard blocks in a number of rows in parallel to the quay. Each yard block may have more than 30 slots of containers stored in length. Vehicles travel along lanes to transfer containers between quay side and yard side. When multiple vessels are loading and unloading, vehicles may arrive at different slot locations for storing and retrieving containers. Local trucks may also arrive through terminal gates to unload export containers or to load import containers. As a result, Yard Cranes (YCs) need to gantry across different slot locations in serving vehicle jobs with a mixture of operation types. Both the YC and the vehicle must be at the same slot position for loading or unloading of containers. When YCs are busy serving other vehicle(s), a vehicle might need to wait at the yard for YC services. YC gantry times also contribute to vehicle waiting times.

One most important performance target of container terminal operation is to minimize vessel turn-around time. It means the YCs need to serve vehicle jobs as efficiently as possible to reduce the delay of vehicles at the yard side in order to continuously feed the quay sides to support Quay Crane (QC) operations. The main objective at the yard side is then translated to minimizing the average vehicle job waiting time for YCs service.

YC deployment is a very complex problem because the workload distribution is uneven and changes dynamically over time. To cater for this, YCs need to move not only within a yard block, but also from one block to another. The movements of a Rubber Tired Gantry Crane include: intra-block linear gantry, inter-block linear gantry and inter-block cross gantry. Intra-block linear gantries are linear movements along the lane within a yard block. Inter-block linear gantries are movements from one block to another in the same row with a similar speed like intra-block gantry, e.g. from BLK5 to BLK4 in Figure 1. Inter-block cross gantries are movements from one block to another in a different row, e.g. from BLK 2 to BLK 6 in Figure 1. A YC doing cross gantry has to make two 90° turns which take much longer time than linear gantry and may also delay the vehicle movements by blocking the lanes. After equipment ordering at the beginning of a shift, a common practice is to initially assign YCs to various yard blocks to work. Then a re-distribution of YCs among the yard blocks is done from time to time to match the dynamically changing workload. However, the workload partitioning in units of yard blocks usually cannot balance the workload among YCs. Also, the number of YCs is usually less than the number of yard blocks. So some blocks with arriving jobs may not have any YC. These result in long vehicle waiting time at the yard side.
To reduce these long waiting times, we propose a hierarchical scheme for YC operation management which is organized in three levels. Level 1 distributes YCs among different rows at suitable times based on predicted future workload. Cross gantries only occur at these suitable times. Level 2 partitions yard blocks in the same row into a number of zones based on the workload distribution in the row with each zone assigned an individual YC. A zone may not be a whole yard block and the zone partition will change over time to cope with the dynamic change of workload distribution. To address safety concerns about possible crane clashes and to simplify dispatching decisions, non-overlapping zones are considered. Level 3 computes job serving sequences for YCs in respective zones. The hierarchical scheme aims at employing inter-block linear gantries to achieve flexible re-partitioning of workload in the same row of yard blocks in between the re-distributions of YCs among different rows to reduce the number of time-consuming cross gantries.

The successfulness of YC deployment highly depends on the accuracy of the future workload prediction. In real world applications, workload is commonly estimated as the total number of jobs expected in a future time interval. And in literature, workload is often roughly derived as proportional to the number of jobs [10, 11]. We propose to consider the average job waiting time as an indicator for YC workload. So long job waiting time indicates heavy workload. Then those estimations based on the number of jobs may deviate widely from true “workload” as 1) For the same number of jobs, a YC will incur different gantry time because of the different job locations or different job serving sequences; 2) For the same set of jobs, different clustering of arrival times will result in different job waiting times; 3) The QC subsystem, the vehicle subsystem and the YC subsystem are closely related and inter-dependent. Any unexpected situation in other subsystem may result in vehicles missing their estimated arrival times at the yard blocks. Our solution is to use a Dynamic Data Driven Simulation System (DDDAS) to obtain accurate workload prediction.

We propose to solve the problems at Level 2 and Level 3 together by employing a DDDAS. Each row is assigned a number of YCs by Level 1 and then our system generates candidate partition plans for the row by considering all possible ways of partitioning its yard blocks into the required number of zones at Level 2 and considering all possible jobs serving sequence of the YC for each zone at Level 3. The contributions of the paper are: (1) The DDDAS simulates what-if scenarios with great accuracy and fidelity of details based on current system states and candidate partition plans as realistic workloads. (2) Two algorithms are proposed to accelerate the computational time of what-if simulations. One approach is based on dynamic programming (DP) with consideration of crane interferences. The other approach is a new Two-Stage Workload Partitioning (TSWP) algorithm which helps to efficiently filter out potential bad partition plans to reduce the number of fully-generated what-if scenarios. Computational time is significantly reduced while maintaining optimality at the same time.

In the rest of the paper, we discuss the related work in Section 2 and present the DDDAS framework in Section 3. The DP approach and the TSWP algorithm are proposed in Section 4. These algorithms are evaluated through simulation experiments in Section 5. Finally, Section 6 concludes the paper.

2. RELATED WORK

In the field of YC dispatching, Kim and Kim [6] studied the loading problem using a single YC with a given load plan and a given bay plan. A Mixed Integer Programming (MIP) model is proposed for routing a single YC. Later, they [7] extended the study by comparing exact optimization, a beam search heuristic and a Genetic Algorithm (GA). Numerical experiments show that the proposed beam search heuristic outperforms the GA. However, the solution quality by the beam-search heuristic is not clear as optimal solutions are not known due to the excessive computational time required. Also, the assumption of having dedicated YCs just to support vessel loading operations may not be practical for terminals with larger number of berths. Lee et al. [9] addressed the problem of vessel loading with two YCs serving one QC and working in two zones. A simulated annealing algorithm was presented of which completion time of all jobs is on average 10.03% above a loosely estimated lower bound.

Published works focusing on inter-block level YCs deployment are not abundant. Zhang et al. [10] described the YCs deployment

![Figure 1. Layout & container flows in a typical terminal](image-url)
problem among blocks with forecasted workload per block per planning horizon (4hrs). The problem is formulated as a MIP model and solved by a modified Lagrangian relaxation method. Only one transfer of a YC in and out of a block is allowed and the forecasted workload is based on the total number of container moves per block. Cheung et al. [11] also studied the YC scheduling of inter-block movements. In the planning horizon, assumptions are made that YCs leave a block only at the beginning of a small time period and jobs only arrive at the beginning of a small time period. A MIP model was proposed with a Lagrangian decomposition solution. A successive piecewise-linear approximation approach is also proposed for large-sized problems. Ng [8] considered the problem of multiple YCs sharing a single bi-directional travel lane and modeled it as an integer program. A heuristic based on dynamic programming was proposed. However, safety constraints for adjacent YCs were not considered. The workload was roughly estimated by a simple greedy heuristic. Performance was described as on average 7.3% above a generated lower bound. Petering et al. [13] presented a simulation study to show that YCs should prioritize retrieval operations and should consider predicted arriving jobs in addition to arrived jobs. Only rule-based dispatching algorithms were considered.

3. SIMULATION-BASED PARTITIONING-DISPATCHING SYSTEM

To support the hierarchical YC operation management, an integrated YC deployment scheme using dynamic data driven simulation is proposed for 2 purposes: 1) to find an optimal plan that partitions the jobs in one row into a number of zones so that each yard crane will be in charge of one non-overlapping zone in the planning window; 2) to determine an optimal YC dispatching sequence (i.e. job serving sequence) in each zone with minimized average job waiting time. The entire partitioning process consists of two components (modules): the optimization module is responsible for (i) generating candidate partition plans; (ii) analyzing and evaluating the results of the what-if experiments and (iii) selecting the optimal partition plan together with the dispatching sequences for the cranes. The simulation module will (i) make predictions of vehicles arrivals by conducting simulation which is driven by the real time location information of the terminal vehicles; (ii) conduct what-if experiments according to the partition plans as requested by the optimization module to predict the performance of these plans.

Upon the request, real time vehicle and YC locations and other relevant information will be collected from the terminal yard and passed into the simulation module and the optimization module. In the simulation module, the model templates including the models for YCs and vehicles will be initiated to form a “base model” representing the terminal at the time of the deployment request. The base model simulates the terminal operations from this time and makes predictions of the vehicle arrivals to the row of yard blocks in the planning period. With the predicted vehicle arrivals, the optimization module will generate YC deployment strategies which are the possible partition plans and pass them to the simulation module for what-if experiments. The what-if simulations predict how YCs will perform using each of these partition plans. The results will be passed into the optimization module for analysis. The partition plan that returns the best performance will be identified and used to assign working zones to YCs. When using real time data driven simulation to derive the crane performance with a partition plan, we also compute the sequence in which a crane will serve the vehicle jobs to yield the best performance for each of the zones. So at the end of the workload partitioning process, we have both the partitioning plan for the YCs and the dispatching sequence for each crane.

4. PARTITION PLANS GENERATION AND SELECTION

The optimization module receives from the simulations module the accurate predictions of the vehicle arrivals to the row in the planning period. Let N be the total number of jobs to the row and Q be the max slot location index of the row. Li is the location of job i, where 1 ≤ Li ≤ Q. For safety concerns, YCs need to be separated by at least 8 slots at all times and obviously each zone for one YC has to be a continuous sequence of slot locations. If zone (a, b) with slots ranging from a to b is assigned to YC x and zone (b+1, c) is assigned to YC x+1, then |Li − Lj| ≥ 8 for all Li ∈ (a, b) and Lj ∈ (b+1, c). Suppose that there are R such potential boundaries like, P1, P2…PR separating the unbreakable areas as shown in Figure 3 with P1 and P5 at the 2 ends of the row.

As shown in Figure 2, a request for Row level YC deployment is generated at the beginning of every planning window, e.g. every 1 hour or as necessary to match the dynamically changing workload. We propose 2 algorithms to compute the optimal partition plan.

4.1 A Dynamic Programming Approach

We denote a part of a row starting from P1 to Pj by
The total job waiting time whose job slot locations are in \( Z(P_1,P_r) \) is denoted by \( T(Z(P_1,P_r),m) \), where \( m \) is the number of YCs allocated to serve in the area. The number of cranes \( k \) allocated to \( Z(P_1,P_r) \) must be smaller than or equal to the number of inseparable areas \((j-i)\). The objective of minimizing the total vehicle job waiting time for the whole row could be expressed as

\[
\begin{align*}
\text{Min } & T(Z(P_1,P_r),m) \\
\text{subject to } & i\leq j \leq R+1
\end{align*}
\]

For example, a problem of deploying 4 YCs for 40 jobs in a row can be formulated as follows:

\[
T(Z(P_1,P_r),m) = \min \left\{ T(Z(P_1,P_1),1) + T(Z(P_1,P_r),m-1) \right\}
\]

for \( m > 1 \), where \( i < k < j, j-k = m-1 \)

When \( m \) is equal to one, all jobs whose slot locations fall in \( Z(P_1,P_r) \) will be served by the same YC. In this study, we employ a modified A* search approach [14] for running what-if simulation and generating the dispatching sequences to predict accurate workload of the crane. \( A^*(P_1,P_r) \) returns the minimized total job waiting time for all jobs whose job locations are in \( Z(P_1,P_r) \). When \( m \) is greater than one, we break \( Z(P_1,P_r) \) into two parts, \( Z(P_1,P_k) \) and \( Z(P_k,P_r) \). One YC will be in charge of \( Z(P_1,P_k) \) and the remaining \((m-1)\) cranes will be in charge of \( Z(P_k,P_r) \). The \( k \) value that minimizes the total job waiting time is the correct choice of the partition point. The constraint of \( j-k=m-1 \) states that the number of unbreakable areas in a zone must be greater than or equal to the number of YCs to avoid assigning two or more YCs to one unbreakable area.

This DP approach restates the optimization problem of YC workload partitioning in recursive form as the optimal solutions of subproblems can be used to find the optimal solution of the overall problem. There are many duplicated subproblems in the solution process. We use a 3-dimensional dictionary DMAP to avoid the re-computation of subproblems. The total job waiting time for \( Z(P_1,P_r) \) with \( m \) YCs will be stored in the entry \( \text{DMAP}[j][i][m] \). The difference to the DP approach in [8] is that they did not consider safety constraints for adjacent YCs. Secondly, they roughly estimate the workload in a single YC zone using a simple greedy heuristics instead of finding the optimal dispatching sequence. Also, they considered every slot position as a potential boundary, which may result in an empty job zone \((a,b)\), where \( a,b \in P \) (i.e., there is no job between \( a \) and \( b \)), or repeated partitioning, e.g., zone\((a,b)\) and zone\((a,b+1)\), where \( b,b+1 \in P \) (i.e., no job at \( b \) and \( b+1 \)).

\[ Z(P_1,P_r) \] where \( i,j \) are integers and \( 0 \leq i < j < R+1 \)

### 4.2 A Two-Stage Workload Partitioning (TSWP) Algorithm

Employing the DP approach, re-computation of the same subproblem is avoided. However, the algorithm still needs to fully evaluate all possible partition plans including those extremely unbalanced ones which are highly unlikely to be the optimal plan. For example, a problem of deploying 4 YCs for 40 jobs in a row may have a possible partition 2-1-2-3-5, in terms of the number of jobs. Intuitively, the plan is unlikely to be optimal. YCs in the first three zones may result in long idle time while the YC in the last zone will be overloaded resulting in long vehicle waiting times. The computational time to find the optimal job sequence for a single YC increases exponentially as the number of jobs increases [2]. Computational time on those extremely unbalanced plans contributes a large proportion of the total simulation time.

A new TSWP algorithm is proposed to filter those bad plans quickly without full evaluation while still guarantee optimality at the same time. The approach uses an improved deployment strategy as shown in Figure 4. A request for a quick evaluation of the candidate plans will be sent from the optimization module to the simulation module for profiling the potentials of the plans with their performance Lower Bounds (LBs). Full what-if experiments are conducted only for those promising plans to save computational time while no optimal plan will be missed. Pseudocode of the TSWP algorithm is shown in Figure 5. Basically, there are two stages in the workload partition process:

1) All possible partition plans will be evaluated quickly to find their respective LB of total job waiting time. In the problem of workload partitioning, the LB of a partition plan is the summation of all LBs of the zones in the plan. The dictionary DMAP will keep the LB of all zones evaluated thus no re-computation for LB of the same zone occurs. Every time the evaluation of a possible partition plan is completed, it will be added to a priority queue planQ which keeps plans in ascending order of their LBs.

2) Conduct full what-if experiments for plans according to their potentials to be the optimal plan. The algorithm always gives priority to the plan that has the least LB and has not been explored, in other words, the top item in planQ. The search of the optimal plan will stop when either the true performance of the current plan is better than the least LB of all unexplored plans or planQ is empty and what-if experiments have been done for all plans. When the full what-if experiment for a zone is done, its true performance will be updated in the dictionary map DMAP to avoid any re-computation in the future.

The success of this algorithm depends on the effectiveness of stage 1 where the LBs of the plans are quickly assessed. If partition plans with long total job waiting times can be identified by large LBs, these plans will be at the end of the queue for which full what-if experiments are unlikely to be explored. We will present our LB generator in the next section and show that we could avoid full what-if experiments for many partition plans especially those unlikely to be the optimal plan.

### 4.3 LB Generator and Accuracy

The accuracy of the LB assessment affects the computational time distribution between Stage 1 and Stage 2. When the LBs are
rough, they cannot profile well the likelihood of plans to be the optimal one. Little time might be spent in Stage 1 while much longer time will be spent in Stage 2. In the extreme case, if LBs for all plans are zero (not computed), no knowledge of the plan potentials is available and all plans need to be fully explored in Stage 2. On the other hand, when more time is consumed in Stage 1 and the LBs are highly accurate, much time will be saved in Stage 2. As optimality likelihoods are well profiled by the priority queue, only a small proportion of the partition plans in the priority queue needs to be explored by the what-if experiments to confirm the optimal plan. In the extreme case, if LBs generated in Stage 1 are exactly the true performances of the plans, no time is needed in Stage 2. It could be seen easily that in the two extreme cases, one stage is exactly the full exploration of all partition plans as described in Section 4.1 and the other stage is doing nothing. The issue is how much Stage 1 should do to achieve substantial savings in computational time for the YC workload partitioning and job sequencing problem.

We propose a modified A* search LB generator where the accuracy of LB $\alpha$ is adjustable. We define $\alpha$ as

$$\alpha = \frac{LB \text{ estimation}}{Optimal \text{ solution value}}$$

(1)

A* search aims to find the optimal path from an initial node to one goal node. It keeps an open list of nodes to be expanded and always tries to select the most promising node first based on an evaluation function $f(x) = g(x) + h(x)$. $g(x)$ is the cost from the start node to $x$ and $h(x)$ is the estimated lowest cost from $x$ to the goal node. When $h(x)$ never overestimates the true cost $h^*(x)$, A* search was proved to be complete and optimally effective [3]. After a portion of the A* search graph is explored, the best $f(x)$ value among all unexplored nodes in the search graph is used as the LB for optimal job waiting time of a single YC zone in this problem. To control the LB estimation process, we use $\beta$ to denote the percentage of nodes explored in the search and it is defined as:

$$\beta = \frac{\text{No. of nodes explored in LB estimation}}{\text{No. of nodes explored to find optimal solution}} : N$$

The relationship between $\alpha$ and $\beta$ of the LB generator could be expressed by

$$\alpha = \beta^2 \quad \text{where} \quad \beta \in [0,1]$$

In general, the more nodes explored leads to a LB with a better estimation of the optimal solution.

One way to characterize the quality of a heuristic search is the effective branching factor (EBF). If the total number of nodes explored to find the optimal solution for a particular problem is $N$, and the solution depth is $d$, then $b^*$ is the EBF that a uniform tree of depth $d$ would have to contain $N+1$ nodes. Thus

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

(3)

Usually the EBF is fairly constant for sufficiently hard problems [4]. The EBF of a single YC dispatching problem for its assigned zone has been evaluated by [5]. It was confirmed that the EBF value does not change much for different problem sizes $x$. Thus, given a zone of $x$ jobs, the solution depth $d=x$. We use equations (3) and (2) to determine the number of nodes to be explored in the LB estimation.

5. PERFORMANCE EVALUATION

5.1 Experimental Design

To evaluate the performance of the proposed YC workload partition schemes, experiments were carried out. Parameter settings like terminal layout, YC gantry speed, vehicle arrival patterns and YC handling rate were obtained from real world terminal models. The gantry speed of the YC is 7.8km/hour within a block of 36 slots. Inter-block linear gantry uses the same speed as we consider YC movements having a higher priority than vehicle movements. The mean YC service time is taken as 120 seconds for all jobs.

<table>
<thead>
<tr>
<th>Table 1. Mean Inter-Arrival Time for various YC settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>seconds</td>
</tr>
<tr>
<td>Mean Inter-Arrival Time</td>
</tr>
</tbody>
</table>

We compare the performance of the TSWP algorithm using different $\beta$ values against the DP approach. There are 5 blocks in a row of yard blocks. We tested scenarios where 3, 4 or 5 YCs are allocated to the row respectively for the planning period of one
6. CONCLUSIONS AND FUTURE WORK

This paper proposes a new YC workload partition scheme which employs dynamic data driven simulation to conduct what-if experiments. Both the optimal workload partition plan for a row and the optimal dispatching sequences for individual YCs are achieved. A DP approach is described to avoid re-computation. An efficient TSWP algorithm is further proposed which filters out bad plan candidates quickly to reduce the number of full what-if simulations carried out while maintaining optimality. Experimental results show that the TSWP algorithm outperforms the DP approach in all tested scenarios such that it only takes around part per thousand computational time of the DP approach. Future work includes designing an algorithm to find the optimal β value and extending our approach to reduce what-if simulation scenarios in general dynamic data driven application systems.

7. REFERENCES