A bounded two-level dynamic programming algorithm for quay crane scheduling in container terminals

Huang, Shell Ying; Li, Ya

2018

Huang, S. Y., & Li, Y. (2018). A bounded two-level dynamic programming algorithm for quay crane scheduling in container terminals. *Computers & Industrial Engineering*, 123303-313. doi:10.1016/j.cie.2018.06.010

<https://hdl.handle.net/10356/89771>

<https://doi.org/10.1016/j.cie.2018.06.010> rims:208363

© 2018 Elsevier. This is the author created version of a work that has been peer reviewed and accepted for publication by *Computers & Industrial Engineering*, Elsevier. It incorporates referee's comments but changes resulting from the publishing process, such as copyediting, structural formatting, may not be reflected in this document. The published version is available at: [<http://doi.org/10.1016/j.cie.2018.06.010>]

Downloaded on 26 Apr 2023 21:13:53 SGT

A bounded two-level dynamic programming algorithm for quay crane scheduling in container terminals

Sheli Ying Huang¹ and Ya Li
School of Computer Science and Engineering
Nanyang Technological University, Singapore
{ASSYHUANG, LIYA}@NTU.EDU.SG

ABSTRACT

One of the optimization problems in terminal operations is the quay crane scheduling problem. The quay crane scheduling algorithm plays a critical role because it directly affects the length of the vessel loading and unloading process, which means vessel turnaround time. We propose a bounded two-level dynamic programming (DP) algorithm which keeps the simplicity of the bay-based approach but overcomes its shortcomings. We also propose a method to estimate the lower bound to quay crane scheduling given the lists of unloading and loading containers and the number of quay cranes assigned to the vessel. This lower bound is used both to reduce computational time of the 2-level DP algorithm and to evaluate our crane scheduling method. Our experiments with real vessel unloading and loading lists for 80 vessels show that the vessel makespan is close to the lower bound. The computational times for the 80 vessels with up to 6600 container moves by cranes in loading and unloading a vessel are all under two minutes.

Keywords: Container terminals, quay crane scheduling, dynamic programming, lower bound

1 Introduction

One of the optimization problems in container terminal operations is the quay crane scheduling problem. A quay crane scheduling solution potentially contributes in a critical way to the operation planning of a terminal. It directly affects the duration of the vessel loading and unloading process, which means vessel turnaround time. In the fiercely competitive container handling business, while bigger and bigger container vessels are built, minimizing the vessel turnaround time has become a top priority for container terminal operators. The capability of producing well-balanced and well-coordinated quay crane operation plans will benefit both a terminal operator and its customers tremendously.

When a vessel arrives at a terminal, quay cranes (QCs) will unload containers and then load containers for the vessel at the quayside. Before a vessel arrives, the list of containers to be unloaded and the list to be loaded, together with their locations on the vessel, are provided to the terminal operator. QCs are allocated to the vessel and a quay crane scheduling plan for the unloading and loading operations is decided. It may happen that not all QCs are able to start the operations for a vessel at the same time. A

¹ Corresponding Author

number of QCs may start on a vessel first. Then one or more QCs may join the unloading and loading operations of this vessel after completing operations for a neighbouring vessel at a later time.

Figure 1 shows the side view of a container vessel. Containers are stacked longitudinally in a number of 40-foot bays and latitudinally in a number of rows in a vessel. Two 20-foot containers will occupy one 40-foot slot in a bay. The container stacks above deck are supported by hatch covers. A QC will unload then load containers in one bay before moving to another bay. When a QC handles the containers in a certain bay, it unloads the containers above deck, then removes some hatch covers, then unloads the containers below deck (in the hold). After unloading, it loads containers below deck and then above deck. Two neighbouring QCs cannot cross each other. They also need to keep a safe distance from each other all the time. The quay crane scheduling algorithm decides the list of containers to be handled by each crane and when they should move to a bay to operate. The objective is that the cranes as a whole take the shortest time to complete all the operations of the vessel.

Figure 1: Side view of a container vessel.

Currently there are two main approaches to quay crane scheduling problem, namely, the bay-based scheduling algorithms (Zhu and Lim, 2006; Lee and Chen, 2010; Lee et al., 2008; Lim et al., 2004, 2007; Liu et al., 2006) and the cluster-based scheduling algorithms (Legato et al., 2012; Bierwirth and Meisel, 2009; Chen et al., 2014; Chung and Chan, 2013; Kim and Park, 2004; Meisel, 2011; Meisel and Bierwirth, 2011; Moccia et al., 2006; Monaco and Sammarra, 2011; Sammarra et al., 2007; Tavakkoli-Moghaddam et al., 2009). The existing bay-based algorithms are simpler than the cluster-based algorithms (Chung and Chan, 2013) but generally suffer from several common shortcomings (Chen et al., 2014): (i) the safety separation distance between QCs are not strictly enforced throughout the scheduled moves; (ii) the time required by the crane's gantry movement is not accounted for; (iii) cranes' initial positions and ready time are not considered. We propose a bounded two-level dynamic programming (DP) algorithm for the scheduling problem. Our work will keep the simplicity of the bay-based approach but overcome all three of its shortcomings. We observe the non-crossing constraints and maintain realistic separation distance between cranes. Crane gantry movement time between bays is accurately reflected, e.g. correct timing to move across the vessel bridge. To be more realistic, we also include the time to remove and put back hatch covers when needed. The 20-foot and 40-foot containers in a 40-foot bay are handled as cranes do with intra-bay crane movements. We also propose a method to estimate the lower bound of the m-crane scheduling problem. Our experiments with real vessel unloading and loading lists for 80 vessels show that the performance is close to lower bound.

In the rest of the paper, we review the related studies in Section 2. Our bounded 2-level DP algorithm for quay crane scheduling is presented in Section 3. The experimental evaluations are presented in Section 4. Conclusion is drawn in the last section.

2 Review of related work

There are mainly two categories of quay crane scheduling solutions (Chen et al., 2014), the bay-based or the cluster-based algorithms. The bay-based approach allocates whole bays of container unloading and loading operations to QCs. This approach also includes algorithms where the units of allocation to each QC are groups of bays. In the cluster-based approach, the work units to allocate to QCs are sets of containers in adjacent locations which have the same loading or destination ports.

For the bay-based approach, Lim et al. (2004) proposed a bipartite matching problem as a representation for the quay crane scheduling problem. They proposed a DP method to solve the QC scheduling problem. However, the solution prohibits two cranes to work in locations close-by even at different times. This means if every bay has jobs, there will be no solution. They also proposed a probabilistic tabu search and a squeaky wheel optimization with local search method. Zhu and Lim (2006) presented a branch-and-bound algorithm to schedule QCs with non-crossing constraints. Liu et al. (2006) studied the problem of allocating QCs to vessels in one linear quay and scheduling their operations. They proposed an MIP model to solve the problem. Lim et al. (2007) proposed an approximation algorithm which generates solutions at most 2 times the optimum. A simulated annealing heuristic that uses random neighborhood generation is used for large problems. Lee et al. (2008) presented an MIP model for allocating containers in units of holds to QCs. In their definition, a hold refers to the containers that are beneath a hatch cover and those above it. A genetic algorithm was proposed to solve it. Lee and Chen (2010) proposed two approximation schemes for QC scheduling with non-crossing constraints. One method is DP with embedded binary search and another is its enhanced version. However, crane separation distance is not considered.

The cluster-based approach (Legato et al., 2012; Bierwirth and Meisel, 2009; Chen et al., 2014; Chung and Chan, 2013; Kim and Park, 2004; Meisel, 2011; Meisel and Bierwirth, 2011; Moccia et al., 2006; Monaco and Sammarra, 2011; Sammarra et al., 2007; Tavakkoli-Moghaddam et al., 2009) does not have to assign whole bays or number of bays to each crane. This allows more flexibility in achieving balanced workload among different QCs than the bay-based approach because in real operations, workload in different bays can vary significantly. However, the number of work units to allocate is much higher than the number of bays. Kim and Park (2004) were the first to propose the cluster-based QC scheduling algorithm. They proposed a MIP model and a branch-and-bound heuristic. Moccia et al. (2006) formulate the QC scheduling problem as a vehicle routing problem and proposed a branch-and-cut algorithm. Tavakkoli-Moghaddam proposed a MIP model and a genetic algorithm (Tavakkoli-Moghaddam et al., 2009). A mathematical model and a branch-and-bound heuristic algorithm were proposed by Bierwirth and Meisel (2009) to solve the QC scheduling problem. Chung and Chen (2013) proposed a genetic algorithm where the mutation rate and the mutation mechanism is controlled during the execution of the genetic algorithm to find the assignments of tasks to QCs. Chen et al. (2014) proposed cluster-based scheduling models which are simpler than earlier models and can be solved by standard optimization solvers. In our proposed two-level DP algorithm, the first level allocates work as the bay-based methods. The second level will adjust workload by shifting part of the work in some bays to other QCs. Therefore, workload in some bays can be shared by neighbouring QCs.

3 The QC scheduling algorithm

3.1 The problem

A few hours before a vessel arrives at a terminal, the terminal operator will be given a list of containers to be unloaded and a list of containers to be loaded for the vessel. In these lists the location of each container is specified by a bay number, a row number and a tier number (in the container stack). In general, since the stowage plan of a vessel has many constraints to satisfy while trying to maximize the number of containers to be carried on a vessel, the number of containers to be handled among various bays may vary widely. Some bays may have zero unloading or loading containers while some other bays may have the capacity number or any number in between. The number of QCs assigned to a vessel is generally known. We assume that all QCs are identical so they work with the same speed. We first consider the problem where all QCs are ready from the start of the vessel operations. Later we show the extension to the algorithm to solve QC scheduling problems where some QCs will join the operation of a vessel at later times. The assumptions are

- (1) QCs are numbered starting from 1 from the left end of the vessel.
- (2) Vessel bays are numbered starting from 1 from the left end of the vessel.
- (3) When a QC is operating at a bay, it will unload containers above deck, remove hatch cover(s), unload containers below deck, load containers below deck, put back hatch covers, load containers above deck before moving to another bay.
- (4) QCs move from lower-numbered 40-ft bays to higher-numbered 40-ft bays (from left to right) but may move from a 20-ft bay to the 20-ft bay on the left within a 40-ft bay.
- (5) QCs are allocated consecutive bays which have container operations.
- (6) If a QC is required to give space to a neighbouring QC after completing its work, it can move beyond the end of a vessel.
- (7) QCs have to be separated by two 40-ft bays. This means if one QC is at a 40-ft bay b , the closest position another QC can be is 40-ft bay $b \pm 3$. To ensure this, the algorithm has to satisfy three requirements as follows. (i) The starting bays of the QCs must have enough separation distance. (ii) During the operations of a QC, the schedule of the QC has to maintain separation distance with the QC on the right. If the QC on the right has completed, forced movement of the QC on the right is done to give space to a QC on the left – this may trigger a chain of forced movements of multiple QCs. (iii) The vessel bridge is considered equivalent to one bay as far as separation distance is concerned. The gantry time taken across the bridge is assumed to be the same as the gantry time taken across one bay.

The objective of the scheduling algorithm is to minimize the vessel makespan, i.e. the time taken to complete all unloading and loading operations for a vessel. This means to minimize the longest working time among the QCs. The working time of a QC is the time from the start of its first container (unload/load) operation till the completion of its last container operation.

3.2 The basic algorithm

Let the number of QCs serving a vessel be m . The top level logic of the algorithm is presented in Figure 2. First, the container operations are sorted according to their bay numbers and a list of bays that have work is produced. Let the number of 40-ft bays that has work be n . For example, Figure 3 shows a work list w containing a list of bays that has container operations (unloading and/or loading) from one of the vessels we tested. The number of bays that has work, n , is 14.

The QC scheduling algorithm computes the best partition of work which is represented in a work list by the DP approach. In the first level DP, executed by the first call to the QCscheduling method in Figure 2, each element in the work list represents one whole bay of work. Thus the first level DP will find the best way to divide the list of consecutive bays into partitions so that the longest working time among QCs is minimized. To achieve a better balance of work among QCs, the fourth step in Figure 2 computes the direction of shifting part of work in some bays among the QCs. A new work list is prepared. In this list, different amounts of work which may be shifted between neighbouring QCs will appear as new elements. The second level DP, executed by the second call to the QCscheduling method in Figure 2, computes the best partition of the new work list to achieve workload balance among QCs.

1. Preprocessing the unloading list and the loading list according to their bay number to initialize work list **w1**;
2. Initialize dynamic program's table **matrix1** for first level DP;
3. QCscheduling(**m**, **w1**, **0**, **n1**, **matrix1**); // n1 is the size of w1
4. Compute work flow required in order to achieve better workload balance and initialize work list **w2**;
5. Initialize dynamic program's table **matrix2** for second level DP;
6. QCscheduling(**m**, **w2**, **0**, **n2**, **matrix2**); // n2 is the size of w2

Figure 2: The 2-level DP algorithm.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
w	1	2	3	6	8	9	11	12	13	14	15	16	18	20

Figure 3: An example work list.

In the formulation of the DP program, the terms used are

$A[k, b]$ = the longest QC time from the best partition from work list element $w[b]$ to $w[n-1]$ by k QCs, i.e. the longest QC time from the best partition by k QCs for the section from $w[b]$ to the end of the work list.

$T[s]$ = the total time for QC **m** (the rightmost QC) to handle from $w[s]$ to $w[n-1]$. QC **m** is the rightmost QC thus QC interference is not considered. The total time includes all necessary intra-bay and inter-bay gantry movement time, time to remove hatch covers, time to put back hatch covers and time to handle all containers.

$T[k, s, t]$ = the total time for QC k to handle from $w[s]$ to $w[t]$, $k < m$, $t < n-1$. In addition to all the necessary working time as in $T[s]$ (here it works till $w[t]$), it also includes the idle/waiting time of QC k due to interference with QC $k+1$ (the crane on the right). The waiting time is obtained from simulation of QC operations.

The DP program computes

$$A[1, i] = T[i] \quad (1)$$

$$A[k, i] = \min_{is < j \leq im} \max(A[k-1, j], T[m-k+1, i, j-1]) \quad \text{for } 2 \leq k \leq m \quad (2)$$

For (1), it gives the time taken by a single QC (the rightmost QC) to complete the operations from $w[i]$ to $w[n-1]$. For (2), it gives the shortest time taken by k QCs to complete the operations from $w[i]$ to $w[n-1]$, $2 \leq k \leq m$. QC $m-k+1$ is the leftmost QC among the k QCs. The leftmost QC starts from $w[i]$ and the second leftmost QC starts from $w[j]$. QC position for $w[i]$ and that for $w[j]$ must have at least 2 bays between them. To avoid unnecessary computations, the value of j is restricted to $is < j \leq im$. Indices is and im are decided as follows.

- 1) Index is is the smallest index of w which defines the smallest section from $w[i]$ to $w[is]$ that the leftmost QC may handle. Figure 4 presents the method to compute is .
- 2) Index $im - 1$ is the largest index of w which defines the largest section from $w[i]$ to $w[im-1]$ that the leftmost QC may handle. Thus from $w[im]$ to $w[n-1]$, the $k-1$ QCs must have enough space to work (enough separation distance). Figure 5 presents the method to compute im .

```

calculatels(i) { // calculate  $is$ 
    if ( $w[i+1] - w[i] > 2$ )    return  $i$ ;
    else if ( $w[i+2] - w[i] > 2$ )    return  $i+1$ ;
    else return  $i+2$ ;
}

```

Figure 4: Calculation of is .

```

calculatelm(k, n) { // calculate  $im$ 
     $L = n$ ;
     $im = L - 1$  //here initialise the index  $im = L-1$  as  $w$ 's index starts from 0
    For each QC  $q$  from 1 to  $k-1$  {
        If ( $im == n-1$  or  $w[im]-w[im-1] > 2$ ) minNumOfBay = 1;
        Else if ( $w[im]-w[im-2] > 2$ ) minNumOfBay = 2;
        Else minNumOfBay = 3;
         $L = L - \text{minNumOfBay}$ ;
         $im = L$ ; }
    return  $im$ ;
}

```

Figure 5: Calculation of im .

Figure 6 shows the DP program. $\text{RightmostQCtime}(i)$ computes the total working time by the rightmost QC that includes all necessary intra-bay and inter-bay gantry movement time, time to remove hatch covers, time to put back hatch covers and time to handle all containers from $w[i]$ to $w[n-1]$. An intra-bay gantry movement happens within a 40-ft bay when a QC needs to move between the 40-ft position to a 20-ft position (either on the left or on the right) or between two 20-ft positions (the two odd bays within the even bay). $\text{QCtime}(k, s, t)$ computes the total time for QC k to handle from $w[s]$ to $w[t]$ including the idle/waiting time of QC k due to interference with QC $k+1$ (the crane on the right). The waiting time is obtained from simulation of QC operations. The array $\text{startBay}[i]$ stores the index number of w which gives the first bay where QC i starts working. Each QC works from $\text{startBay}[i]$ till $w[\text{startBay}[i+1]-1]$.

After the first level DP, the fourth step in Figure 2 makes preparation for the second level DP by computing the direction and the amount of workload adjustment among the QCs. Figure 7 outlines the procedure. It produces a new array **w2** for the fifth step second level DP in Figure 2.

```

QCscheduling(NofQCs, w, startIndex, n, A)

```

```

{
If (NofQCs is 1) {
    If (A[1, startIndex] == -1) A[1, startIndex] = RightmostQCtime(startIndex);
    Return A[1, startIndex]; }
Else {
    smallest = maxint;
    is = calculatels(startIndex); // Figure 4
    im = calculatelm(NofQCs, n); // Figure 5
    For (j = is+1 to im) {
        If (A[NofQCs-1, j] ≠ -1)
            current = A[NofQCs-1, j];
        else current = max(QCscheduling(NofQCs-1, w, j, n, A), QCtime(m-NofQCs+1, startIndex, j-1));
        if (current < smallest) {
            smallest = current; startBay[m-NofQCs+2] = j; }
    }
    A[NofQCs, startIndex] = smallest;
Return smallest;
}

```

Figure 6: DP for the computation of A[NofQCs, startIndex].

1. Compute the working time for each QC based on the decisions of first call to QCscheduling;
2. Between each pair of neighbouring QCs, decide whether some work should be shifted from one QC to another or no shift of workload is needed. This depends on the difference between the average working time of all QCs on the left and the average working time of all QCs on the right;
3. If a shift of work between a pair of QCs is needed, estimate the amount of work to be shifted based on the average working time of all QCs and the respective current working time of the two QCs;
4. Aggregate all bays in the array **w1** that are allocated to one QC by the first level DP step into one single entry. Thus **m** QCs means **m** entries.
5. Between each pair of neighbouring QCs, if a shift of work is needed, split the single entry in **w1** for the QC which is sending some work to its neighbour into 4 entries, for the second call to QCscheduling to decide.

Figure 7: Compute work flow required in order to achieve better workload balance.

Figure 8. An example of QC working time after first level DP.

As an example, the working time (minutes) for the 4 QCs, after first level DP, is shown in Figure 8. The arrows are the decisions for workload shift from step 2 in Figure 7. The numbers below the arrows are the results of step 3. The entries of the array **w2** after step 5 are shown in Figure 9. After the procedure in Figure 7, the second level DP will work on the array **w2** to find the best division of work.

All the bays allocated to QC1 by first level DP minus 52 minutes of work	One third of the 52 minutes of work in the rightmost bay of QC1	One third of the 52 minutes of work in the rightmost bay of QC1	One third of the 52 minutes of work in the rightmost bay of QC1	All the bays allocated to QC2 by first level DP
0	1	2	3	4
One third of the 154 minutes of work in the leftmost bay of QC3	One third of the 154 minutes of work in the leftmost bay of QC3	One third of the 154 minutes of work in the leftmost bay of QC3	All the bays allocated to QC3 by first level DP minus 154 minutes of work in the leftmost bay and 78 minutes of work in the rightmost bay	
5	6	7	8	
One third of the 78 minutes of work in the rightmost bay of QC3	One third of the 78 minutes of work in the rightmost bay of QC3	One third of the 78 minutes of work in the rightmost bay of QC3	All the bays allocated to QC4 by first level DP	
9	10	11	12	

Figure 9. The resultant array w2.

3.3 The bounded dynamic programming algorithm

Dynamic programming algorithms can be computational intensive. In Equation (2), to compute $A[k-1, j]$, we need to recursively apply Equation (2). To compute $T[m-k+1, i, j-1]$, we need to simulate the operations of QC $m-k+1$ with the considerations of the operations of QC $m-k+2$ to obtain the total time required by this QC, including its waiting time due to interference for QC $m-k+2$ on the right. We use the concept of lower bound to reduce the computations required in Equation (2). The bounded dynamic program which is used in both Step 3 and Step 6 in Figure 2 is presented in Figure 10. The idea is, to compute $\min_{is < j \leq im} \max(A[k-1, j], T[m-k+1, i, j-1])$, we do not need to compute every $A[k-1, j]$ and every $T[m-k+1, i, j-1]$ for $is < j \leq im$. We first compute one pair of $A[k-1, j]$ and $T[m-k+1, i, j-1]$ for a promising value of j . Remember the value of $\max(A[k-1, j], T[m-k+1, i, j-1])$ as the current smallest value of $A[k, i]$. Subsequently, we only compute a pair of $A[k-1, j]$ and $T[m-k+1, i, j-1]$ for other values of j if the lower bound, $LB(\max(A[k-1, j], T[m-k+1, i, j-1]))$, is smaller than the current smallest.

The lower bound for $\max(A[k-1, j], T[m-k+1, i, j-1])$ is

$$LB(\max(A[k-1, j], T[m-k+1, i, j-1])) = \max(LB(A[k-1, j]), LB(T[m-k+1, i, j-1]))$$

The lower bound for $A[k, i]$ for any values of k and i is the minimum amount of time required by k QCs to complete the operations from $w[i]$ to $w[n-1]$ for a vessel. In Figure 10, it is represented by $LB_A[k, i]$. The computation of the lower bound is shown in Figure 11. LB1 in Figure 11 is based on the idea that the minimum amount of time may be achieved if all k QCs are able to work simultaneously for the whole period of time. So it is the total amount of time to complete all container operations by a single crane divided by k . In addition, there are $k-1$ inter-bay gantry times made by a single crane which will not be made by k cranes. We subtract the smallest $k-1$ inter-bay gantry times from the total time to complete

all container operations by a single crane in the calculation. LB2 is based on the idea that containers in bays within the separation distance d cannot be handled at the same time by two cranes. So LB2 identifies the longest time needed to complete any $d+1$ consecutive bays by a single crane, including gantry time. Then the lower bound for completing these bays is the larger of LB1 and LB2.

The lower bound for $T[k, s, t]$ is the minimum time crane k takes to complete the container operations from $w[s]$ to $w[t]$. It is the total working time for one crane from $w[s]$ to $w[t]$ without considering the interference with the crane on the right. It is represented by $LB_T[k, s, t]$ in Figure 10.

```

BoundedDP(NofQCs, w, startIndex, n, A, LB_A, LB_T)
{
  If (NofQCs is 1) {
    If (A[1, startIndex] == -1) A[1, startIndex] = RightmostQCtime(startIndex);
    Return A[1, startIndex]; }
  Else {
    is = calculatels(startIndex); // Figure 3
    im = calculatelm(NofQCs, n); // Figure 4
    firstDivide = n/NofQCs;
    If (A[NofQCs-1, firstDivide] ≠ -1)
      smallest = A[NofQCs-1, firstDivide];
    else smallest = max(BoundedDP(NofQCs-1, w, firstDivide, n, A, LB_A, LB_T),
      QCtime(m-NofQCs+1, startIndex, firstDivide-1));
    For (j = firstDivide-1 down to is+1) {
      current = PartitionAtJ(NofQCs, w, startIndex, n, A, LB_A, LB_T, j, smallest);
      if (current < smallest) {
        smallest = current; startBay[m-NofQCs+2] = j; }
    }
    For (j = firstDivide+1 to im) {
      current = PartitionAtJ(NofQCs, w, startIndex, n, A, LB_A, LB_T, j, smallest);
      if (current < smallest) {
        smallest = current; startBay[m-NofQCs+2] = j; }
    }
    A[NofQCs, startIndex] = smallest;
    Return smallest;
  }
}

PartitionAtJ(NofQCs, w, startIndex, n, A, LB_A, LB_T, j, currentBest)
{
  If (A[NofQCs-1, j] ≠ -1) // best time for A[NofQCs-1, j] available
    return A[NofQCs-1, j];
  else {
    if (LB_A[NofQCs-1, j] == -1) // LB of A[NofQCs-1, j] not available {
      LB_A[NofQCs-1, j] = LBofWorkTime(NofQCs-1, j);
      LB_T[m-NofQCs-1, startIndex, j-1] = LB_QCT(m-NofQCs+1, startIndex, j-1); }
    if (LB_A[NofQCs-1, j] < currentBest) and (LB_T[m-NofQCs+1, startIndex, j-1] < currentBest)
      return max(BoundedDP(NofQCs-1, w, j, n, A), QCtime(m-NofQCs+1, startIndex, j-1));
    else return maxint; }
}

```

Figure 10: BoundedDP algorithm for QC scheduling.

The bounded 2-level DP algorithm for QC scheduling is the 2-level DP algorithm in Figure 2 with the calls to QCscheduling replaced by calls to BoundedDP. The bounded 2-level DP algorithm is able to find the

optimal partitions of work among a number of QCs under the assumptions in Section 3.1. Safety separation distance between neighbouring QCs is maintained at all times during vessel operations. The gantry movement times for QCs are taken into considering by the scheduling algorithm.

```

LBofWorkTime(k, b)
// computing the lower bound of completing work from w[b] to w[n-1] by k cranes
{
  MaxFTofAllWL = completion time of all workload by one QC including all gantry times;
  LB1 = (MaxFTofAllWL - sum of (k - 1) largest gantry times between neighbouring bays that have work) / k;
  LB2 = 0; // we assume that there must be at least 2 bays between 2 QCs, i.e. d=2
  For each bay y from bay b to the third rightmost bay
    Total3BayWL = total work time of bays y, y+1 and y+2;
    If work time in y and y+1 are both not zero
      add gantry time between y and y+1 to Total3BayWL;
    If work time in y+1 and y+2 are both not zero
      add gantry time between y+1 and y+2 to Total3BayWL;
    If work time in y and y+2 are both not zero but work time in y+1 is zero
      add gantry time between y and y+2 to Total3BayWL;
    if (LB2 < Total3BayWL) LB2 = Total3BayWL;
  return max(LB1, LB2);
}

```

Figure 11: Computation of the lower bound of $A[k, b]$.

3.4 Initial positions and ready times of quay cranes

The last shortcoming of bay-based algorithms is that the initial positions and ready times of QCs are not considered. We show that with a small modification to the bounded DP algorithm, we are able to consider the different ready times of QCs.

In a QC scheduling plan of a vessel, each QC has a starting bay. If a QC is not at its starting bay and is idling, it can start moving to its starting bay and be ready when the vessel's service starts. However a QC may still be working on a neighbouring vessel and can only join the operations of this vessel later. For example, a vessel may have six QCs in its scheduling plan but only four QCs are ready at the start of the vessel service. Two QCs may only join in at some later times. We assume that QCs that are ready to serve the vessel at a later time will join at one end of the vessel or both ends of the vessel. Then the issue of initial positions of QCs is resolved once the ready times of QCs are considered. The times that these QCs are able to reach their respective starting bays of this vessel after serving a neighbouring vessel will be the ready times of these QCs. In our algorithm, the QC that can join a vessel's operation earlier will be assigned to take up jobs at more center bays than another QC that will join later. Therefore the blocking phenomenon described in (Chen et al., 2014) will not happen.

The function `RightmostQCtime()` in Figure 10 is the same as that in Figure 6. It computes the time required by the rightmost QC to complete all operations in the (tentatively) assigned bays with no need to consider QC interference. If this QC has a ready time later than the start of the vessel service, we just

need to add a virtual bay at the end of the work list w and take the QC ready time into consideration in the function.

The function $QCtime()$ in Figure 10 is also the same as that in Figure 6. It simulates the relevant QC (not the rightmost QC) operations and compute a QC's time to complete all operations in the (tentatively) assigned bays with the consideration of interference with the QC on its right. If this QC has a ready time later than the start of the vessel service, we can also insert a virtual bay either before $w[0]$ or after $w[n-1]$ depending on which end of the vessel the QC will join the service. The ready time of the QC is taken as the time the QC completes the work in the virtual bay.

3.5 Examples of results

In this section, we present some example results of the bounded 2-level DP QC scheduling algorithm with identical QC ready times or different QC ready times.

Figure 12: An example of scheduling result for one vessel, 4 QCs.

Figure 12 shows an example output for one vessel served by 4 QCs where QCs are ready at the same time. The horizontal axis shows the time and the vertical axis shows the vessel bays. Each rectangular box shows the bay a QC is at for a period of time indicated by the length of the box. The small zig-zag movements of QC positions (intra-bay movements) show the movements of the QC when it is unloading 40-foot containers at the center position of the bay, then move left (up) to unload 20-foot containers in the left half of the bay, then moves right (down) to unload 20-foot containers in the right half of the bay. Similar zig-zag movements occur for loading. All these are accounted for in the working time of a QC in a bay. It can be seen from the figure that our algorithm allows the sharing of workload in the same bay

by 2 neighbour QCs. The last working bay of QC 2 is the first working bay of QC 3. The last working bay of QC 3 is the first working bay of QC 4.

Figure 13: Scheduling plan where QC4 joins the vessel operation later.

Figure 14: Scheduling plan where QC3 and QC4 join the vessel operation later.

Figure 13 shows an example output for one vessel served by 4 QCs where three QCs are ready at the time when a vessel is berthed and QC4 joins the operation from a neighbouring vessel later. Figure 14 shows an example output for one vessel served by 4 QCs where two QCs are ready at the time when a vessel is berthed and QC 3 and QC4 join the operation from a neighbouring vessel at different times later. Note that towards the end of QC4's operations, QC3 cannot move to its last bay because it is too close to QC4 and therefore has to wait. After QC4 completes its operation, QC4 will be able to move away (downward direction in the figure but not shown), possibly to join the operation of another vessel. QC3 then moves to its last bay and works till it completes its operation. There are eighteen other examples we tested, some of which have QCs to join a vessel's operations from both ends of the vessel. These tested examples show that our algorithm is able to handle the different ready times of QCs correctly.

4 PERFORMANCE EVALUATION

We tested our bounded two-level DP algorithm with real container operation lists planned for real vessels from a shipping company. The algorithms are programmed in C++ language under Microsoft Visual Studio 2010 using Dell Precision T3500, Windows 7 64-bit OS, Intel(R) Xeon(R) CPU with 3.2GHz and 6GB RAM.

We tested 3 groups: 22 vessels with 4 QCs allocated, 40 vessels with 5 QCs allocated and 18 vessels with 6 QCs allocated. Tables 1, 2 and 3 show the total number of container moves and the number of bays that have containers to be handled in each of these cases in the 3 groups. Each container move is a QC loading or unloading operation that involves either one 20-ft container, one 40-ft container or two 20-ft containers.

The evaluation of the algorithm compares the makespan of the loading and unloading operations of a vessel with the lower bound, given the number of QCs. This lower bound is computed by the LBofWorkTime method in Figure 8 where k is the number of QCs allocated to serve the vessel and b is 0. So this gives the lower bound of a vessel's makespan when all QCs are ready at the time when a vessel is berthed. The computational time of the bounded 2-level DP algorithm (B2LDP) is compared with the 2-level DP algorithm (2LDP) to assess how much B2LDP is able to improve the efficiency of 2LDP.

Table 1a: The total number of container moves for each of the 22 vessels, each served by 4 QCs.

636	1918	1916	1900	1866	1867	1870	2614	230
2368	3073	1609	2944	3235	3235	3532	4626	185
2785	2786	2672	1655					

Table 1b: The number of bays that have container moves for each of the 22 vessels, each served by 4 QCs.

12	14	14	14	20	19	18	17	6
17	15	17	17	19	19	19	19	9
18	18	19	13					

Table 2a: The total number of container moves for each of the 40 vessels, each served by 5 QCs.

3190	2718	3295	1870	2991	2991	2991	2990	2991	2990
6329	5652	5065	6310	4513	4440	4083	5014	5065	3701
3651	5958	3014	5420	3361	4725	3068	4734	4722	3941
4734	3013	3790	3493	2718	2805	2795	2663	2692	4468

Table 2b: The number of bays that have container moves for each of the 40 vessels, each served by 5 QCs.

20	19	14	20	18	17	16	18	17	18
22	22	20	22	20	21	19	21	20	21
21	21	19	21	22	21	20	21	21	18
21	17	21	20	19	16	17	18	19	19

Table 3a: The total number of container moves for each of the 18 vessels, each served by 6 QCs.

2990	5448	6418	6643	3729	6299	3544	3928	6168
6435	6679	4722	4129	5250	3929	4918	3080	1859

Table 3b: The number of bays that have container moves for each of the 18 vessels, each served by 6 QCs.

15	20	21	21	21	21	21	21	21
21	21	21	18	21	21	22	20	20

Table 4a: Comparison of vessel makespan (minutes) between the bounded two-level DP algorithm (B2LDP) and the lower bound (LB) for 22 vessels served by 4 QCs. The “%” columns give the percentage differences of B2LDP from LB, i.e. (B2LDP - LB)/LB in %.

LB	B2LDP	%	LB	B2LDP	%	LB	B2LDP	%
1487	1490	0.20%	1314	1343	2.21%	2571	2584	0.51%
1570	1573	0.19%	2364	2367	0.13%	169	184	8.88%
1467	1468	0.07%	1069	1074	0.47%	1515	1596	5.35%
1068.25	1117	4.56%	1681	1740	3.51%	1515.5	1598	5.44%
1081	1097	1.48%	1899	1919	1.05%	1493	1506	0.87%
1130	1202	6.37%	1899	1919	1.05%	377	429	13.79%
1563	1592	1.86%	2490	2493	0.12%	1731	1734	0.17%
259	259	0.00%	1441.49	1467.45	2.65%			

Tables 4a, b and c show the results of makespan (2LDP and B2LDP return the same scheduling results therefore the same makespan). The makespan of a vessel is the length of time taken by all the QCs to finish all the loading and the unloading operations for the vessel. The entries in bold in the tables present the average among all vessels for the respective number of QCs we tested. The results in these tables show that our algorithm is able to generate QC scheduling plans which in most cases are very close to the shortest makespan possible. Among the 80 cases, 68 cases is less than 2% from the lower bound, which means 85% of the cases is less than 2% difference from the lower bound. Seventy two

cases, which is 90% of the cases, is less than 5% from the lower bound. There are 2 cases where B2LDP results are 11.29% and 13.79% respectively from their lower bound. In one case (6 QCs), 4 QCs are working with just the minimum distances between them for most of their working time and among them 2 QCs have to wait for 3 hours before they can move to the next bay to work. For cases where 5 QCs serve a vessel, 38 out of 40 cases, that is, 95% of the cases, the results are less than 2% difference from the lower bound. For cases where 6 QCs serve a vessel, 16 out of 18 cases, that is, about 90% of the cases, the results are less than 1% difference from the lower bound.

Tables 5a, b and c show the computational times taken by the 2-level DP algorithm (2LDP) presented in Figure 2 and Figure 5 and the bounded 2-level DP algorithm (B2LDP) presented in Figure 2 and Figure 7. The entries in bold in the tables present the average among all vessels for the respective number of QCs we tested. The results show that the 2LDP and the B2LDP algorithms are able to compute the QC scheduling plans for all 80 vessels within 2 minutes. Out of the 80 cases, B2LDP is able to shorten the computational time in 79 cases. In the best 64 cases, the improvement is from 10% to 56%. In one case, B2LDP does not do better but actually takes a longer time by 1.95%. This will happen if the lower bound estimates do not reduce the computation of the DP matrix cells. Then the algorithm will pay the cost of computing the lower bounds for many cells. However, as the 80 cases show, this is a rare case.

Table 4b: Comparison of vessel makespan (minutes) between the bounded two-level DP algorithm (B2LDP) and the lower bound (LB) for 40 vessels served by 5 QCs. The “%” columns give the percentage differences of B2LDP from LB, i.e. $(B2LDP - LB)/LB$ in %.

LB	B2LDP	%	LB	B2LDP	%	LB	B2LDP	%
1669	1674	0.30%	2171	2176	0.23%	2485	2488	0.12%
1301	1314	1.00%	2473	2474	0.04%	2384	2389	0.21%
1726	1746	1.16%	2162	2167	0.23%	2200	2203	0.14%
854	900	5.39%	2655	2658	0.11%	2469	2472	0.12%
1467	1470	0.20%	2390	2393	0.13%	1564	1567	0.19%
1487	1490	0.20%	1602.4	1634	1.97%	1637	1640	0.18%
1523	1526	0.20%	1577.2	1649	4.55%	2106	2114	0.38%
1399	1402	0.21%	2524.4	2540	0.62%	1793	1796	0.17%
1435	1438	0.21%	1388	1399	0.79%	1301	1314	1.00%
1537	1542	0.33%	2644	2649	0.19%	1455	1460	0.34%
2891	2896	0.17%	1638	1651	0.79%	1417	1420	0.21%
2516	2557	1.63%	2073	2086	0.63%	1386	1389	0.22%
2390	2393	0.13%	1598	1601	0.19%	1315	1328	0.99%
2985	2990	0.17%	1898.16	1908.75	0.67%			

Table 4c: Comparison of vessel makespan (minutes) between the bounded two-level DP algorithm (B2LDP) and the lower bound (LB) for 18 vessels served by 6 QCs. The “%” columns give the percentage differences of B2LDP from LB, i.e. $(B2LDP - LB)/LB$ in %.

LB	B2LDP	%	LB	B2LDP	%	LB	B2LDP	%
1344	1347	0.22%	2423	2428	0.21%	2405	2408	0.12%
2581	2586	0.19%	2487	2500	0.52%	1411	1416	0.35%
2460	2465	0.20%	1867	1869	0.11%	1431	1434	0.21%
2330	2335	0.21%	1923	1928	0.26%	815	818	0.37%
1396	1399	0.21%	1940	2159	11.29%	2182	2185	0.14%
1586	1589	0.19%	1638	1641	0.18%	1337	1417	5.98%
1917.28	1934.44	0.88%						

Table 5a: Comparison of computational time (seconds) between the two-level DP algorithm (2LDP) and the bounded two-level DP algorithm (B2LDP) for 22 vessels served by 4 QCs. The “%” columns give the percentage improvement of B2LDP from 2LDP, i.e. (2LDP - B2LDP)/2LDP in %.

2LDP	B2LDP	%	2LDP	B2LDP	%	2LDP	B2LDP	%
16.45	11.81	28.21%	18.52	13.76	25.73%	63.07	27.39	56.56%
17.69	13.44	24.04%	16.98	15.65	7.87%	1.16	1.09	6.32%
13.36	7.16	46.44%	12.60	8.89	29.44%	76.83	69.12	10.04%
27.46	13.77	49.84%	26.66	16.14	39.45%	76.58	68.46	10.60%
22.49	15.65	30.42%	35.87	19.55	45.51%	38.22	25.61	32.97%
19.11	11.88	37.82%	33.79	18.51	45.24%	5.13	5.22	-1.95%
24.65	13.95	43.39%	53.14	27.43	48.38%	9.59	7.05	26.45%
1.15	1.07	6.82%	27.75	18.76	29.53%			

Table 5b: Comparison of computational time (seconds) between the two-level DP algorithm (2LDP) and the bounded two-level DP algorithm (B2LDP) for 40 vessels served by 5 QCs. The “%” columns give the percentage improvement of B2LDP from 2LDP, i.e. (2LDP - B2LDP)/2LDP in %.

2LDP	B2LDP	%	2LDP	B2LDP	%	2LDP	B2LDP	%
27.08	17.34	35.96%	67.88	46.8271	31.02%	39.39	25.8085	34.49%
21.39	13.33	37.69%	56.21	36.4416	35.17%	38.12	23.6227	38.03%
12.08	10.00	17.20%	50.03	36.5662	26.91%	28.70	22.0321	23.25%
20.32	11.47	43.57%	61.35	36.7532	40.09%	50.31	36.6074	27.24%
15.22	12.04	20.93%	56.51	29.8349	47.21%	29.09	19.7628	32.05%
15.63	13.41	14.18%	56.42	47.1637	16.40%	51.32	32.7531	36.18%
15.45	11.61	24.88%	31.42	22.6948	27.77%	45.78	42.2233	7.78%
16.18	12.89	20.30%	85.47	72.9766	14.62%	21.39	13.0464	39.02%
15.74	12.62	19.85%	24.48	20.6845	15.52%	75.79	71.9869	5.02%
17.99	13.09	27.22%	80.02	77.8747	2.68%	77.58	73.6515	5.06%
48.14	32.40	32.70%	38.71	20.934	45.92%	21.08	19.9798	5.21%
79.55	61.78	22.34%	45.62	28.2371	38.10%	14.86	8.55445	42.42%
57.99	29.63	48.90%	23.34	16.558	29.05%	77.90	62.3374	19.98%
73.33	47.79	34.83%	42.12	31.13	27.17%			

Table 5c: Comparison of computational time (seconds) between the two-level DP algorithm (2LDP) and the bounded two-level DP algorithm (B2LDP) for 18 vessels served by 6 QCs. The “%” columns give the percentage improvement of B2LDP from 2LDP, i.e. (2LDP - B2LDP)/2LDP in %.

2LDP	B2LDP	%	2LDP	B2LDP	%	2LDP	B2LDP	%
11.41	9.22	19.21%	18.97	16.10	15.14%	27.86	23.99	13.89%
39.89	31.08	22.09%	23.17	19.14	17.40%	119.43	115.43	3.35%
81.59	76.16	6.66%	47.56	44.42	6.61%	21.90	17.51	20.05%
80.95	68.56	15.30%	70.09	68.00	2.99%	39.22	34.58	11.81%
42.66	39.37	7.71%	107.32	100.07	6.76%	17.40	15.43	11.30%
75.51	73.67	2.44%	63.14	53.59	15.12%	12.20	9.51	22.10%
50.02	45.32	12.22%						

5 Conclusion

We proposed a bounded two-level DP algorithm for QC scheduling which considers many realistic operational details. Safety distance between QCs is maintained throughout the whole loading and unloading process of a vessel. All intra-bay and inter-bay gantry times and hatch cover handling times are accounted for. The algorithm is also able to handle situations where not all QCs can start at the same time. Apart from intra-bay movements, the algorithm plans the QCs to move in one direction. The 2-level DP algorithm first finds the optimal bay-based QC scheduling in the first level DP. Then it refines the results by allowing workload sharing between neighbouring QCs in the second level DP. The algorithm retains the simplicity of bay-based QC scheduling approach but overcomes all its shortcomings. The scheduling results are very close to the lower bound. The bound we use in the DP algorithm is able to reduce computational time significantly. Future work includes QC allocation and scheduling for multiple vessels.

Acknowledgement

This project is supported by the Ministry of Education, Singapore, project number RG133/16.

References

- Bierwirth, C., & Meisel, F. 2009.** A fast heuristic for quay crane scheduling with interference constraints. *Journal of Scheduling*, 12(4), pp 345–360.
- Chen, J. H., Lee, D. H. & Goh, M. 2014.** An effective mathematical formulation for the unidirectional cluster-based quay crane scheduling problem. *European Journal of Operational Research*, 232, pp 198–208.
- Chung, S.H. & Chan, F. T.S. 2013.** A workload balancing genetic algorithm for the quay crane scheduling problem. *International Journal of Production Research*, 51(16), pp 4820–4834.
- Kim, K. H. & Park, Y. M. 2004.** A crane scheduling method for port container terminals. *European Journal of Operational Research*, 156(3), pp 752–768.
- Lee, D. H. & Chen, J. H. 2010.** An improved approach for quay crane scheduling with non-crossing constraints. *Engineering Optimization*, 42(1), pp 1–15.
- Lee, D. H., Wang, H. Q., & Miao, L. X. 2008.** Quay crane scheduling with noninterference constraints in port container terminals. *Transportation Research Part E – Logistics and Transportation Review*, 44(1), pp 124–135.
- Legato, P., Trunfio, R., & Meisel, F. 2012.** Modeling and solving rich quay crane scheduling problems. *Computers and Operations Research*, 39(9), pp 2063–2078.
- Lim, A., Rodrigues, B., Xiao, F., & Zhu, Y. 2004.** Crane scheduling with spatial constraints. *Naval Research Logistics*, 51(3), pp 386–406.
- Lim, A., Rodrigues, B., & Xu, Z. 2007.** A m-parallel crane scheduling problem with a non-crossing constraint. *Naval Research Logistics*, 54(2), pp 115–127.
- Liu, J., Wan, Y., & Wang, L. 2006.** Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. *Naval Research Logistics*, 53(1), pp 60–74.
- Meisel, F. 2011.** The quay crane scheduling problem with time windows. *Naval Research Logistics*, 58(7), pp 619–636.

- Meisel, F., & Bierwirth, C. 2011.** A unified approach for the evaluation of quay crane scheduling models and algorithms. *Computers and Operations Research*, 38(3), pp 683–693.
- Moccia, L., Cordeau, J. F., Gaudio, M., & Laporte, G. 2006.** A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. *Naval Research Logistics*, 53(1), pp 45–59.
- Monaco, M., & Sammarra, M. 2011.** Quay crane scheduling with time windows, one-way and spatial constraints. *International Journal of Shipping and Transport Logistics*, 3(4), pp 454–474.
- Sammarra, M., Cordeau, J., Laporte, G., & Monaco, M. 2007.** A tabu search heuristic for the quay crane scheduling problem. *Journal of Scheduling*, 10(4), pp 327–336.
- Tavakkoli-Moghaddam, R., Makui, A., Salahi, S., Bazzazi, M. & Taheri, F. 2009.** An efficient algorithm for solving a new mathematical model for a quay crane scheduling problem in container ports. *Computers & Industrial Engineering*, 56 (1), pp 241–248.
- Zhu, Y., & Lim, A. 2006.** Crane scheduling with non-crossing constraint. *Journal of the Operational Research Society*, 57(12), pp 1464–1471.