An MDP based Approach to Solve Quay Crane Scheduling Problem under Uncertainty

Tianyi Gu, Christopher Amato
Department of Computer Science, University of New Hampshire,
Durham, NH 03824, USA
tg1034@wildcats.unh.edu, camato@cs.unh.edu

Abstract

One issue of container terminal operation planning is quay crane scheduling problem (QCSP). The number of time-conflict-tasks for yard cranes (YC) in yard operation is considered as one of the important measures for the evaluation of the level of fluency for quay crane scheduling by terminal experts. In this project, an MDP model for QCSP will is developed. A UCT algorithm is designed to solve the model. Several numerical tests will be conducted to show the performance of the proposed approach.

Introduction

Since the introduction of the container in April 1956, when Malcolm McLean moved fifty-eight 35 foot containers from Newark to Houston by a refitted oil tanker, container flows have increased continuously. Annually, about 108 million cargo containers are transported through seaports around the world, constituting the most critical component of global trade. Between 1990 and 2015, the total number of full containers shipped internationally is expected to grow from 28.7 million to 177.6 million (United Nations: ESCAP, 2007). A simple calculation shows that there are enough containers on the planet to build more than two 8-foot-high walls around the equator (Taggart, 1999).

Containerization has become the main driver for intermodal freight transport, which involves the transportation of freight in containers of standard dimensions (20 ft equivalent unit (1 TEU), 40 ft (2 TEU), 45 ft (high-cube)), using multiple modes of transportation such as ships, trucks, trains, or barges without any handling of the freight itself when changing modes (Crainic and Kim, 2007). Bundling freight in containers reduces cargo handling, and thereby improves security, reduces damages and losses, and allows freight to be transported faster (Agerschou et al., 1983). In the chain of intercontinental transport, container terminals are of special importance since all containers pass through at least one of them during their drayage. Container terminals are the nodes where different modalities meet to transport containers.

Seaport container terminals are essential nodes in sea cargo transportation networks. The marine container industry has grown dramatically in the last three decades, and container transportation has become a predominant mode of inter-continental cargo traffic. Container terminals, which are multi-modal interfaces that connect sea transportation with land transport, have an important role. To exploit economies of scale, the size of container ships has significantly increased during the last few decades. A large container ship typically requires the lifting of thousands of containers at the terminal during one call. Since running a container ship involves a major capital investment and significant daily operating costs, customer service has become the principal issue at container terminals. More container terminals are seeking to improve their throughput and reduce the turnaround time of vessels. As such, the operational efficiency of container terminals in handling containers passing through them plays a critical role in a globalized world economy. Many models and algorithms have been developed to address various decision problems in container terminals to help improve operational efficiency. One issue of container terminal operation planning is quay crane scheduling problem (QCSP). The number of time-conflict-tasks for yard cranes (YC) in yard operation is considered as one of the important measures for the evaluation of the level of fluency for quay crane scheduling by terminal experts.

This paper present an MDP based approach to solve the Quay Crane Scheduling Problem under Uncertainty. Our reasons for choosing MDP based approach as a solution approach are as follows: Firstly, because of the complexity of the problem, we can imagine that the QCSP model will be a NP-hard problem. It is deemed unable to obtain optimal solutions for large-scale problems. Hence, heuristic algorithms are wildly used to obtain near-optimal solutions efficiently. However, because of the numerous constraints with high degree of irregularity, it is difficult to evaluate a scheduling scheme in the process of heuristic algorithms. Secondly, today it is widely observable by operators in real world container terminal that it is better to generate quay scheduling plan before generate ship stowage plan. Because if people plan for container vessel location without the information of QC working sequence, a lot of good scheduling chance will be kill. But, if we want to generate QC scheduling sequence without stowage plan, uncertainty is difficult to tackle by the analytical model alone. Thirdly, in practice, how to schedule QC according to real-time condition is important. We can use an online MDP solver to handle this problem. Each time after all QC performs an action, the yard response (as indicated by its new state) is used for planning next action. The objective is to develop a decision-making policy for selecting the appropriate action rule for each QC. By this approach, the optimal moves for QCs can be obtained. In this project, an MDP model for QCSP is developed. A UCT algorithm is designed to solve the model. Several numerical tests are conducted to show the performance of this approach.

The remainder of the paper is organized as follows. We first provide background on quay crane scheduling problem as well as our approach which is based on Markov decision processes (MDPs) and MDPs solver algorithms. We then discuss our MDP based approach for solving quay crane scheduling problem under uncertainty. In section 4, we describe our experimental results, showing that this approach can solve both small size problem and large size problem. Finally, we supply an overview of the related work on QCSP and then conclude.

Background

Quay Crane Scheduling Problem

Container handling equipment includes quay cranes (QCs), yard cranes (YCs), and yard trucks (YTs). These systems are shown in Figures 1a–c, and are used to tranship containers from ships to barges, trucks and trains. Sea container terminals are divided into several areas such as seaside, landside, stacking, and internal transport areas that cater to seaside and landside operations. At a container terminal, QCs load and unload containers from ships berthed along the quay at the seaside. QCs pick up or drop off containers on YTs which transport containers from the seaside to the stacking area where YCs take over. The typical work flow (Figure 2) during a loading operation for container departure that are retrieved from the yard and loaded to a vessel

(a) QC (b) YC

(c) YT

Figure 1: A top view of a container terminal and material handling equipment

(YV), is as follows. One YC (yard crane) picks up a container from a container block and loads it onto an YT (yard truck). The YT then transports the container to a QC (quay crane), which loads the container onto the vessel. The unloading operation for arrival containers is the reverse of the foregoing: a container is unloaded from a vessel to the yard (VY).

Seaside operations planning consists of ship berthing operations (berth planning and quay crane scheduling), and loading and unloading of containers onto ships. Further, the stowage planning where the sequence of loading and unloading containers in a ship is optimized plays a critical role in the seaside operations planning.

In container terminals, the principal objective of terminal operators is to minimize the duration of time for which a vessel stays at the terminal. Short ship sojourn times minimize the risk of violating the ship's time window. It also allows negotiating better time slot lengths with the ship liners. The prime objective of ship liners is to have the ship sailing as much as possible as berthing time is considered as cost. The order in which containers are unloaded and loaded by each QC can significantly alter the handling time of a vessel. Thus, optimal sequencing of tasks performed by each QC is necessary. This problem is widely known as the quay crane scheduling problem (QCSP). In terminal operations, QCs are typically the most constrained resources. Hence, optimal schedules can maximize throughput, and minimize ship handling time (ship make span). Several constraints need to be satisfied during the schedule generation process, such as preventing crane crossovers

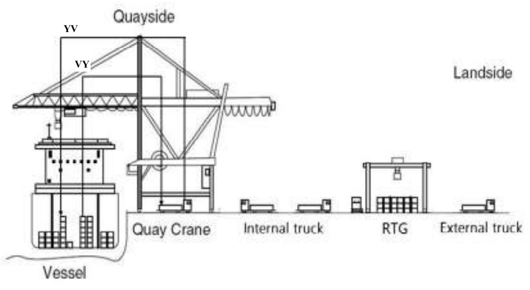


Figure 2: Typical work flow at a container terminal

(structural constraint imposed on cranes and crane trajectory), maintaining a minimum distance between cranes (neighborhood constraint), time separation of containers that need to be stacked in the same location (job separation constraint), and ensuring that unloading transactions within a ship bay precede loading transactions (precedence constraint defined by the stowage plan). Multiple optimization formulations have been developed with variations in task attributes (single or multiple bays), crane attributes (initial and final positions of the cranes, operational time windows), and interference attributes.

Markov decision processes

To generate quay crane scheduling plan without detail ship stowage plan, we choose to use Markvo Decision Processes (MDPs) as the general framework. MDPs are a common method for modeling sequential decision-making with stochastic actions. We generate quay crane scheduling policy for an MDP through sampling-based techniques.

A finite discrete-time fully observable MDP is a tuple <*S, A, T, R*>, where:

- *S*, a finite set of all possible states of the system;
- *A*, a finite set of all actions an agent can take;
- *T*, a transition function, a mapping specifying the probability $T(s_1,a,s_2)$ of going to state s_2 if action a is executed when the agent is in state s_1.
- *R*, a reward function that gives a finite numeric reward value $R(s,a)$ obtained when the system take action *a* in state *s*.

An MDP unfolds over a series of steps. At each step, the agent observes the current state, *s*, chooses an action, *a*, and then receives an immediate reward that depends on the state and action, $R(s,a)$. The agent begins in the initial state s_0, which is assumed to be known. The state transitions according to the distribution P as given above and the process continues. The goal is to find a policy, which is a mapping, π, from states to actions, that maximizes the sum of rewards over the steps of the problem. In this paper, we consider the finite horizon problem which unfolds over a finite number of steps.

Algorithms to solve MDP

Value iteration

To find the optimal policy π we will compute the value vector *U*. For each state $s \in S$, $U(s)$ is the expected cost of reaching the target state, using the best possible sequence of actions starting at state *s*.

We can start from an initial value vector $U_0(s) = 0$ for all $s \in S$. Then the update step is

$$U_{t+1}(s) = \max_{a\in A}\left\{ R(s,a) + \sum_{s'\in S} P(s'\mid s,a)U_t(s') \right\} \quad (1)$$

In other words, we maximize the sum of the local reward of taking some action, along with the expected reward from the possible new states. The policy is updated at each step by

$$\pi_{t+1}(s) = \arg\max_{a\in A}\left\{ R(s,a) + \sum_{s'\in S} P(s'\mid s,a)U_t(s') \right\} \quad (2)$$

The value iteration algorithm is guaranteed to converge.

We can also apply a discounting factor γ to reflect the fact that rewards incurred in the future are worth less than ones incurred in the present. If $0 \le \gamma \le 1$ then the value iteration update is simply

$$U_{t+1}(s) = \max_{a\in A}\left\{ R(s,a) + \gamma\sum_{s'\in S} P(s'\mid s,a)U_t(s') \right\} \quad (3)$$

Real-time Dynamic Programming

RTDP is a simple DP algorithm that involves a sequence of trials or runs, each starting in the initial state s_0 and ending in a goal state. Each RTDP trial is the result of simulating the greedy policy π_V while updating the values $V(s)$ using (4) over the states *s* that are visited.

$$V(s) := \max_{a\in A} R(s,a) + \sum_{s'\in S} P(s'\mid s,a)V(s') \quad (4)$$

Thus, RTDP is an asynchronous value iteration algorithm in which a single state is selected for update in each iteration. This state corresponds to the state visited by the simulation where successor states are selected with their corresponding probability. This combination is powerful, and makes RTDP different than pure greedy search and general asynchronous value iteration.

Note that unlike asynchronous value iteration, in RTDP there is no requirement that all states be updated infinitely often. Indeed, some states may not be updated at all. On the other hand, the proof of optimality lies in the fact that the relevant states will be so updated.

The convergence of RTDP is asymptotic, and indeed, the number of trials before convergence is not bounded (e.g., low probability transitions are taken eventually but there is always a positive probability they will not be taken after any arbitrarily large number of steps). Thus for practical reasons, like for value iteration, we define the termination of RTDP in terms of a given bound $\varepsilon > 0$ on the residuals.

UCT: Upper Confidence bounds on Trees

Solving an MDP system using value iteration can become computationally intensive on large examples because each update step necessarily reads and changes every element of the value and policy vectors.

An alternative approach is to use Monte Carlo planning. The algorithm alternates randomly between trying new actions at each state (in order to search for better policies) or using the current best policy to improve the estimate of the policy. The policy vector π is not explicitly computed. Instead we have the state-action vector $Q(s, a)$, the average seen reward of taking action a when in state s.

The UCT algorithm just changes the way that actions are selected during a rollout. The algorithm used is called Upper Confidence Bounds (UCB). Imagine that we have K gambling machines with arbitrary reward distributions $P_1,\ldots, P_K$. At each time step we can play any machine j that we choose, and receive a reward according to the distribution P_j. We would like a strategy that maximizes our total reward over n plays. Since the distributions P_j are fixed but unknown, we want to avoid sampling too many times from machines with low reward. In other words, we have to balance exploration versus exploitation.

The UCB1 strategy is:

1. Play each machine once.
2. Play machine j that maximizes $\overline{x}_j + \sqrt{\frac{2\ln n}{n_j}}$ where $\overline{x}_j$ is the average reward from machine j, machine j has been played n_j times so far, and n is the total number of plays so far.

The $\overline{x}_j$ term gives preference to machines that have performed well in past plays, while the $\sqrt{2\ln n / n_j}$ term gives preference to machines that have not been played many times so far, relative to ln n.

The UCT algorithm is the same as Monte Carlo planning except that UCB is used to select the action at each tree node. To do this we have to keep track of the average reward so far (i.e. up to time t) of taking action a from state s, denoted $Q_t(s, a)$. We also track $N_s(t)$, the number of times that state s has been visited up to time t. If we are at a tree node s then the action is chosen using the UCB rule

$$a^* = \arg\max_{a\in A}\left\{Q_t(s,a) + \alpha\sqrt{\frac{\ln N_s(t)}{N_{s,a}(t)}}\right\} \tag{5}$$

Where α is a constant that has to be chosen empirically. A badly chosen α can have a huge effect on the convergence of the algorithm.

An MDP based approach for QCSP

The basis of our approach is to fully model the quay crane scheduling problem by Markov decision process. We use a sampling-based algorithm (UCT) to solve the MDP model. During quay crane planning, all QC choose the best joint action according to the accumulate Q value of each action in UCT tree.

Modeling QCSP as MDP

The QC assignment is the precondition of QCSP, which means we know each QC will work on which vessel bays. Vessel bays is consisted of vessel slots (Figure 3), so we know each QC will work on which vessel slots. As we described in section 2, unloading operations is the reverse of loading operations, so we just discuss loading operations here. We assume the configuration of yard blocks will not be changed during the planning period, which means there will be no new container transfer to every block and no pre-upload task will be cancel.

State space

We define the state space with the loaded container numbers in each bay slot and the pre-load container numbers of each container type in each block. These two type of container numbers were chosen because they provide the full information about the loading progress status.

States are then discretized over the possible values. Let v_i be the loaded container numbers in vessel slot i, b_i^t be the t type pre-load container numbers in block i. Then the state could be describe as follow:

$$S : \left\langle v_1, v_2, v_3, \ldots, v_i \mid b_1^1, b_1^2, \ldots, b_1^t, b_2^1, \ldots, b_2^t, \ldots b_i^1, \ldots, b_i^t \right\rangle$$

For example, if we have 3 slots, 2 blocks, and 2 container types , the state <11, 7, 1, 9, 1, 4, 3, 3> (Figure 4) means 11 containers have been loaded to slot 1, 7 containers have been loaded to slot2, 1container have been loaded to slot 3, 9 containers have been loaded to slot4, and for all type 1 containers, there are 1 containers left in block 1 wait to be loaded, there are 3 containers left in block 2 wait to be loaded, and For all type 2 containers, there are 4 containers left in block 1 wait to be loaded, there are 3 containers left in block 2 wait to be loaded.

Action space

An action is a joint action of all QCs movements in next step. We defined a_i^k as the action of QC k moving to bay slot i. Then $A : \left\langle a_i^1, a_i^2, \ldots, a_i^k \right\rangle$ is the joint action in this

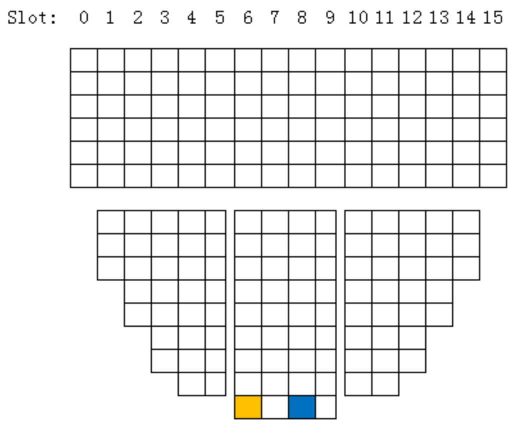


Figure 3: Vessel slots of vessel bay

model. For example, if we have two QCs, the action <2,6> means QC1 move to slot2 and QC2 move to slot6 in next step. For those QC do not need to move, we use a dummy action -1 to represent.

Transition model

We define conflict index c_i for each legal next state i. Conflict index is the aggregate conflict container numbers for every block when transfer from previous state. All legal next states are ranked by their conflict index. We define the transition probability as 0.9 for the highest rank state and as 0.1 divided by the remainder for all other legal states. Legal states can satisfy both previous states and action on logic. In real world container terminal, operators always like to clean those blocks with less containers. According to this operation rule, for those states have same conflict index, the state with less containers will get higher transition probability.

$$P\left(s'|(s,a)\right)=\begin{cases} 0.9 & \begin{matrix} \textit{If } s' \textit{ is the highest rank state} \\ \left(\begin{matrix} \textit{choose the } s' \text{ with } \textit{less containers} \\ \textit{if several states all rank first} \end{matrix}\right) \end{matrix} \\ \dfrac{0.1}{\textit{Number of legal next states}-1} & \textit{Else} \end{cases}$$

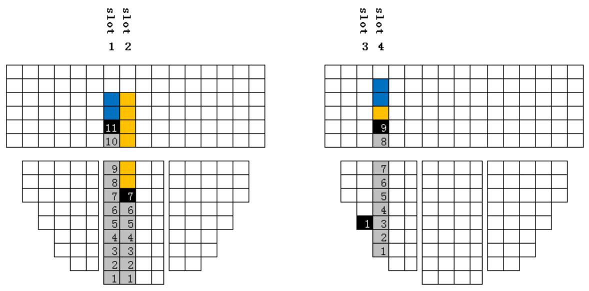
Reward model

We define the immediate reward as follow:

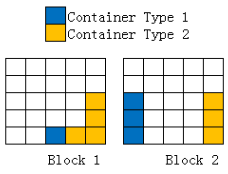
$$R(s)=\begin{cases} 100 & \textit{If } s \textit{ is final state} \\ -1 & \textit{Else} \end{cases}$$

UCT

UCT is a planning algorithm that can successfully cope even with exponential transition functions. Exponential transition functions plague other MDP solvers ultimately because these solvers attempt to enumerate the domain of the transition function for various state-action pairs, e.g., when summing over the transition probabilities or when generating action outcomes to build determinizations. Being a Monte Carlo planning technique, UCT only needs to be able to efficiently sample from the transition function



(a) Slots



(b) Blocks

Figure 4: An example of state space

and the reward function.

Our UCT is presented in Algs. 1-5. The algorithm starts with a given initial state, in which all the slots are empty and all the blocks are full, and the final state, in which, all the slots are filled and all the blocks are empty.

1)Main UCT (*Alg. 1, Lines 2-4*): UCT works by sampling a number of trajectories through the state space and build UCT tree. In each trial, the algorithm insert a new state node into UCT tree by selection policy, and get a Q-value by playing rollout on this state, and updates Q-value approximations for the state-action pairs the trajectories visit.

2)Selection policy (*Alg. 2, Lines 2-5*): In every state s, UCT selects an action a' based on UCB value (*Alg .3*). The algorithm then samples an outcome of a' and continues the trial until find a new state s which is not in the UCT tree and add this node to UCT tree. Then the algorithm will play a rollout on this state (*Alg .4*) and get a Q-value.

3)Backup (*Alg. 5, Lines 2-6*): For each state s in the search trajectory, UCT maintains a counter ns for the number of times state s has been visited by the algorithm. The n_s counter is incremented every time a rollout passes through s. For each state-action pair<s,a>, UCT maintains a counter $n_{s,a}$ of how many times UCT selected action a when visiting state s. Clearly, for each s $n_s=\sum_{a\in A} n_{s,a}$. Finally, for each <s,a>, UCT keeps track of an approximation of the Q-value of a in s equal to the average reward accumulated by past rollouts after visiting s and choosing a in s.

```
Algorithm 1: UCTSEARCH()
1    While there is time left do:
2        s'←TREEPOLICY(s)
```

```
3        Q← PLAYSIMULATOR(s')
4        BACKUP(s',Q)
5     end while
6     return BESTACTION(s)
```

```
Algorithm 2: TREEPOLICY(s)
1     While s is in UCT Tree do:
2        a←BESTACTION(s)
3        s←sample according to T(s'|s,a)
4     end while
5     add s into UCT Tree
6     return s
```

```
Algorithm 3: BESTACTION(s)
1     return $\arg\max_{a\in A}\left\{\hat{Q}(s,a)+c\sqrt{\frac{\ln(n_s)}{n_{s,a}}}\right\}$
```

Experiments

To demonstrate the applicability of out approaches, we performed several different scale problems. All our experiments are performed on a Quad-Core Intel i-7 processor. We use 100 as the execution horizon for small size problem and 1000 for large size problem, although reaching the goal earlier will stop the execution. The planning horizon for a decision is an input to each problem. All UCT rollouts use random actions. The exploration constant c for UCB equation is set as the same magnitude of current Q value at the node. How to set this automatically is an important question for future work.

Small and large size cases study

In our small size case, there are 5 vessel slots, in each slot there are 3 containers to be loaded. In yard area, there are 3 yard blocks. There are 2 QCs assigned to finish all load tasks. There are 2 container types. Table 1 is the configuration of all slots. The number in each slot is the container identify number and the number in the parenthesis is the type of this container. Table 2 is the configuration of all blocks. The numbers in parenthesis indicate the number of containers of each container type in each block. Table 3 is the QC assignment.

The result is shown in Figure 5. By ten joints action of 2 QCs, the loading operation can be finished without yard conflict. The action sequence of each QC is shown in Table 4.

```
Algorithm 4: PLAYROLLOUT (s)
1     while s is not final state do:
2        a ← randomly pick an action in s
3        s←sample according to T(s'|s,a)
4        Q←Q+R(s)
5     end while
6     return Q
```

```
Algorithm 5: BACKUP (s,Q)
1     while s is not NULL do:
2        a ← parent of s
3        s ←parent of a
4        $\hat{Q}(s,a)\leftarrow\frac{n_{s,a}\hat{Q}(s,a)+Q}{n_{s,a}+1}$
5        $n_s \leftarrow n_s + 1$
6        $n_{s,a} \leftarrow n_{s,a} + 1$
```

In our large size case, there are 10 vessel slots, in each slot there are 10 containers to be loaded. In yard area, there are 6 yard blocks. There are 4 QCs assigned to finish all load tasks. There are 8 container types. Table 5 is the configuration of all slots. The number in each slot is the container identify number and the number in the parenthesis is the type of this container. Table 6 is the configuration of all blocks. The numbers in parenthesis indicate the number of containers of each container type in each block. Table 7 is the QC assignment.

The result is shown in Fig. 6. By 30 joint actions of 4 QCs, the loading operation can be finished within only 9 yard conflicts. The action sequence of each QC is shown in table 8.

Simulation

To test the performance of this approach, we test on a mid-size problem. We take an average of 100 trials for each number of nodes are inserted for each decision. In our experiments we find that as the increase of nodes in UCT trees, the result (Table 9) of the algorithm can converge to optimal solution.

Related work

The earliest work on the QCSP is by Daganzo (1989) and Peterkofsky and Dazango (1990). They assume one crane per hold is assigned for each vessel. Daganzo formulates a mixed integer program for the QCSP considering multiple vessels, and presents both exact and approximation methods to solve the problem for small instances. Peterkofsky and Dazango attempt to minimize the delay costs of the vessels and propose a branch and bound method to solve the problem. However, both these studies ignore some key

Table 1: The configuration of all slots (small size)

Slot 0	Slot 1	Slot 2	Slot 3	Slot 4
2(1)	5(0)	8(1)	11(0)	14(1)

1(0)	4(1)	7(0)	10(1)	13(0)
0(1)	3(0)	6(1)	9(0)	12(1)

Table 2: The configuration of all blocks (small size)

Block 0	Block 1	Block 2
0(2)	0(3)	0(2)
1(3)	1(2)	1(3)

Table 3: The QC assignment (small size)

QC	slots
0	0,1
1	2,3,4

Table 4: QC action sequence (small size)

QC	Actions
0	0,0,0,1,1,1
1	3,2,2,3,4,3,4,2,4

characteristics of the QCSP such as interference of QCs, safety distance requirements between adjacent QCs, and precedence relationships between tasks. The QCs are mounted on the same rail track and cannot overtake each other. Some unloading (loading) tasks cannot be executed simultaneously, e.g. if the containers are positioned too close to each other. Further, some tasks need to be performed in sequence. For instance, containers on the deck have to be unloaded before unloading the containers from the hold of the vessel. These limitations are addressed in the papers by Kim and Park (2004) and Lim et al. (2004).

Kim and Park discussed practical QC scheduling in terms of assumed time windows during which QCs are assigned to a vessel. They developed an effective heuristic search algorithm, called GRASP (Greedy Randomized Adaptive Search Procedure), for the QC scheduling problem to eliminate the computational difficulty of the previously branch and bound method. Subsequently, Lee, Wang, and Miao (2008) proposed a genetic algorithm to optimize the handling sequence for QCs, and accounting for interference among QCs.

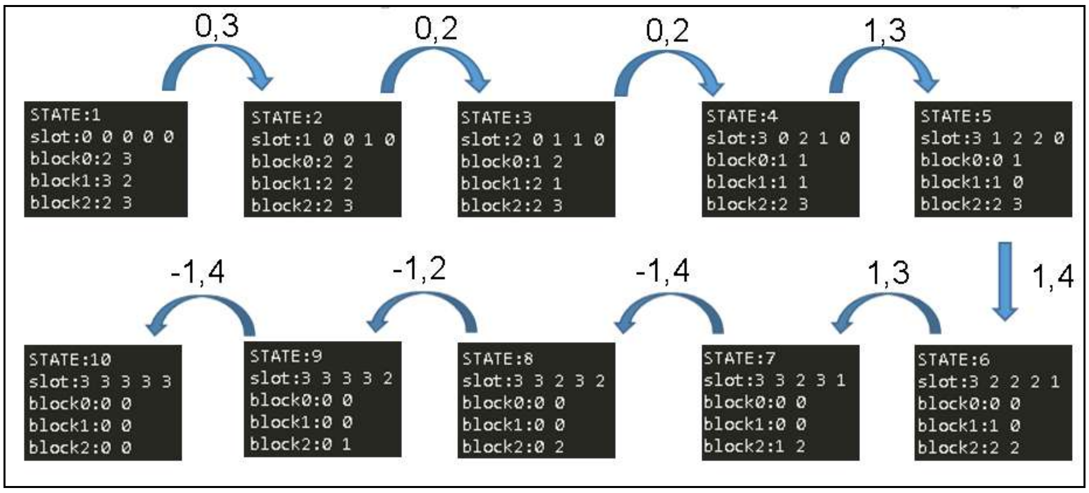
Stahlbock and Voß (2008) studied various CT operation planning problems, Although the container sequence problem (CSP) has not yet been addressed directly, key related issues, such as crane operations planning, dual cycling, and container reshuffling, have been investigated in the context of stowage planning, yard crane scheduling, and QC scheduling. Zhu and Lim (2005) and Liu, Wan, and Wang (2006) also investigated the quay crane scheduling problem.

Goodchild and Daganzo (2006) firstly incorporated crane dual cycling issues into QC scheduling. Their approach concerns the operating process of a single QC at a bay of a container vessel. Every container stack in the considered bay is represented by two tasks that are related by precedence, the first of which is unloading the stack and the other of which is loading the stack. The processing time of these task are determined by the number of containers to be (un-)loaded. Crane dual cycling is realized by the parallelization of unloading and loading of different stacks. The problem is to find a sequence for processing stacks that minimizes the make span of the schedule while maximizing crane dual cycling. Hence, formulate this problem as a two-machine flow shop scheduling problem, which is solved exactly using the rule of Johnson (1954). Goodchild and Daganzo (2007) proved the economic benefits of the QC dual cycling, which are increased crane productivity, berth utilization, and vessel utilization. Zhang and Kim (2009) extended the approach of Good child and Daganzo by considering the effect of hatch covers on QC operations including local search to solve stack-based QC scheduling incorporates dual cycling. Frank and Matthias (2010) proved that the consideration of internal reshuffles shortens vessel handling time more than does crane dual-cycling alone. Zhen, L (2011) studies two tactical level decision problems arising in transshipment hubs: berth template planning that is concerned with allocating berths and quay cranes to arriving vessels, and yard template planning that is concerned with assigning yard storage locations to vessels. Lu Chen (2012) studied the interactions between crane handling and truck transportation in a maritime container terminal by considering them as simultaneous. They formulated the problem as a constraint programming model and developed a three-stage algorithm.

Conclusion

In this paper, we present an MDP based approach to solve the Quay Crane Scheduling Problem under Uncertainty. The MDP model handle the uncertainty of generating quay scheduling plan without ship stowage plan. A UCT algorithm is designed to solve the model to fulfill the real-time condition. This approach is able to obtain the optimal moves for QCs. We show that both small size problem and large size problem can be solved in reasonable time.

In the future, we are interested in extending this line of research in several ways. For instance, according to real world operation, other operation rules can be add to influence the transition probability. Also, many terminal operation related problem can be modeled and solved using AI techniques. We hope that others will also explore more way of using AI techniques in this rich domain.



```
STATE:1
slot:0 0 0 0 0
block0:2 3
block1:3 2
block2:2 3
```

```
STATE:2
slot:1 0 0 1 0
block0:2 2
block1:2 2
block2:2 3
```

```
STATE:3
slot:2 0 1 1 0
block0:1 2
block1:2 1
block2:2 3
```

```
STATE:4
slot:3 0 2 1 0
block0:1 1
block1:1 1
block2:2 3
```

```
STATE:5
slot:3 1 2 2 0
block0:0 1
block1:1 0
block2:2 3
```

```
STATE:6
slot:3 2 2 2 1
block0:0 0
block1:1 0
block2:2 2
```

```
STATE:7
slot:3 3 2 3 1
block0:0 0
block1:0 0
block2:1 2
```

```
STATE:8
slot:3 3 2 3 2
block0:0 0
block1:0 0
block2:0 2
```

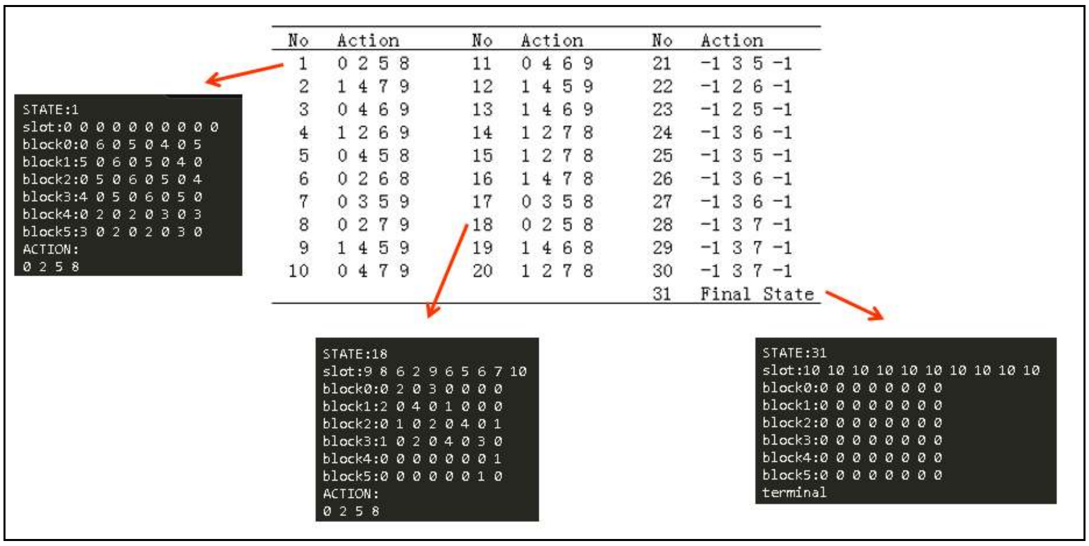
```
STATE:9
slot:3 3 3 3 2
block0:0 0
block1:0 0
block2:0 1
```

```
STATE:10
slot:3 3 3 3 3
block0:0 0
block1:0 0
block2:0 0
```

Figure 5: Scheduling result for small size QCSP

Table 5: The configuration of all slots (large size)

Slot 0	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot6	Slot 7	Slot 8	Slot 9
0(1)	10(3)	20(5)	30(7)	40(1)	50(3)	60(5)	70(7)	80(1)	90(3)
1(2)	11(4)	21(6)	31(0)	41(2)	51(4)	61(6)	71(0)	81(2)	91(4)
2(3)	12(5)	22(7)	32(1)	42(3)	52(5)	62(7)	72(1)	82(3)	92(5)
3(4)	13(6)	23(0)	33(2)	43(4)	53(6)	63(0)	73(2)	83(4)	93(6)
4(5)	14(7)	24(1)	34(3)	44(5)	54(7)	64(1)	74(3)	84(5)	94(7)
5(6)	15(0)	25(2)	35(4)	45(6)	55(0)	65(2)	75(4)	85(6)	95(0)
6(7)	16(1)	26(3)	36(5)	46(7)	56(1)	66(3)	76(5)	86(7)	96(1)
7(0)	17(2)	27(4)	37(6)	47(0)	57(2)	67(4)	77(6)	87(0)	97(2)
8(1)	18(3)	28(5)	38(7)	48(1)	58(3)	68(5)	78(7)	88(1)	98(3)
9(2)	19(4)	29(6)	39(0)	49(2)	59(4)	69(6)	79(0)	89(2)	99(4)



```
STATE:1
slot:0 0 0 0 0 0 0 0 0 0
block0:0 6 0 5 0 4 0 5
block1:5 0 6 0 5 0 4 0
block2:0 5 0 6 0 5 0 4
block3:4 0 5 0 6 0 5 0
block4:0 2 0 2 0 3 0 3
block5:3 0 2 0 2 0 3 0
ACTION:
0 2 5 8
```

No	Action	No	Action	No	Action
1	0 2 5 8	11	0 4 6 9	21	-1 3 5 -1
2	1 4 7 9	12	1 4 5 9	22	-1 2 6 -1
3	0 4 6 9	13	1 4 6 9	23	-1 2 5 -1
4	1 2 6 9	14	1 2 7 8	24	-1 3 6 -1
5	0 4 5 8	15	1 2 7 8	25	-1 3 5 -1
6	0 2 6 8	16	1 4 7 8	26	-1 3 6 -1
7	0 3 5 9	17	0 3 5 8	27	-1 3 6 -1
8	0 2 7 9	18	0 2 5 8	28	-1 3 7 -1
9	1 4 5 9	19	1 4 6 8	29	-1 3 7 -1
10	0 4 7 9	20	1 2 7 8	30	-1 3 7 -1
				31	Final State

```
STATE:18
slot:9 8 6 2 9 6 5 6 7 10
block0:0 2 0 3 0 0 0 0
block1:2 0 4 0 1 0 0 0
block2:0 1 0 2 0 4 0 1
block3:1 0 2 0 4 0 3 0
block4:0 0 0 0 0 0 0 1
block5:0 0 0 0 0 0 1 0
ACTION:
0 2 5 8
```

```
STATE:31
slot:10 10 10 10 10 10 10 10 10 10
block0:0 0 0 0 0 0 0 0
block1:0 0 0 0 0 0 0 0
block2:0 0 0 0 0 0 0 0
block3:0 0 0 0 0 0 0 0
block4:0 0 0 0 0 0 0 0
block5:0 0 0 0 0 0 0 0
terminal
```

Figure 6: Scheduling result for large size QCSP

Table 6: The configuration of all blocks (large size)

Block 0	Block 1	Block 2	Block 3	Block 4	Block 5
0(0)	0(5)	0(0)	0(4)	0(0)	0(3)
1(6)	1(0)	1(5)	1(0)	1(2)	1(0)
2(0)	2(6)	2(0)	2(5)	2(0)	2(2)
3(5)	3(0)	3(6)	3(0)	3(2)	3(0)
4(0)	4(5)	4(0)	4(6)	4(0)	4(2)
5(4)	5(0)	5(5)	5(0)	5(3)	5(0)
6(0)	6(4)	6(0)	6(5)	6(0)	6(3)
7(5)	7(0)	7(4)	7(0)	7(3)	7(0)

Table 7: The QC assignment (large size)

QC	slots
0	0,1
1	2,3,4
2	5,6,7
3	8,9

Table 8: QC action sequence (large size)

QC	Actions
0	0,1,0,1,0,0,0,0,1,0,0,1,1,1,1,1,0,0,1,1
1	2,4,4,2,4,2,3,2,4,4,4,4,4,2,2,4,3,2,4,2,3,2,2,3,3,3,3, 3,3,3
2	5,7,6,6,5,6,5,7,5,7,6,5,6,7,7,7,5,5,6,7,5,6,5,6,5,6,6, 7,7,7
3	8,9,9,9,8,8,9,9,9,9,9,9,9,8,8,8,8,8,8,8

Acknowledgments

The authors would thank the members of the UNH AI Group and CS980 for their insightful comments.

References

United Nations: ESCAP. 2007. *Regional Shipping and Port Development: Container Traffic Forecast 2007 Update.* United Nations: Economic and Social Comission for Asia and the Pacific (ESCAP), New York.

Taggart, S. 1999. The 20-ton packet. *Wired Magazine* 7(10) 246.

Crainic, G. T., K. H. Kim. 2007. Chapter 8 intermodal transportation. C. Barnhart, G. Laporte, eds., Transportation, *Handbooks in Operations Research and Management Science*, vol. 14. Elsevier, 467–537.

Agerschou, H., H. Lundgren, T. S¨orensen, T. Ernst, J. Korsgaard, L. R. Schmidt, W. K. Chi. 1983. *Planning and Design of Ports and Marine Terminals*. John Wiley and Sons, Chichester.

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley-Interscience, 1994.

F. Sailer, M. Buro, and M. Lanctot. Adversarial planning through strategy simulation. *The IEEE Symposium on Computational Intelligence and Games, Honolulu, HI*, 2007.

Levente Kocsis and Csaba SzepesvÃąri. Bandit based monte-carlo planning. In: *ECML-06. Number 4212 in LNCS*, pages 282–293. Springer, 2006. 5, 1

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, May 2002.

Daganzo C (1989) The crane scheduling problem. *Transp Res Part B* 23:159–179

Peterkofsky R, Dazango C (1990) A branch and bound solution method for the crane scheduling problem. *Transp Res Part B* 24:159–172

Kim K, Park Y (2004) A crane scheduling method for port container terminals. *Eur J Oper Res* 156:752–768

Lim A, Rodrigues B, Xiao F, Zhu Y (2004) Crane scheduling with spatial constraints. *Nav Res Logistics* 51:386–406

Lee D, Wang H, Miao L (2008) Quay crane scheduling with non-interference constraints in port container terminals. *Transp Res Part E* 44:124–135

Lee, D. H., Wang, H. Q., & Miao, L. (2008). Quay crane scheduling with non-interference constraints in port container terminals. *Transportation Research Part E*, 44, 124–135.

Stahlbock R, Voß S (2008). Operations research at container terminals: a literature update. *OR Spectrum* (1):1–52

Zhu, Y., & Lim, A. (2005). Crane scheduling with non-crossing constraint. *Journal of the Operational Research Society*, 57(12), 1464–1471.

Liu J, Wan Y-W, Wang L (2006). Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. *Nav Res Logist* 53(1):60–74

Goodchild AV, Daganzo CF (2006). Double-cycling strategies for container ships and their effect on ship loading and unloading operations. *Transp Sci* 40(4):473–483

Goodchild AV, Daganzo CF (2007). Crane double cycling in container ports: planning methods and evaluation. *Transp Res B* 41(8):875–891

Zhang H, Kim KH (2009). Maximizing the number of dual-cycle operations of quay cranes in container terminals. *Comput Indust Eng* 56(3):979–992

Frank Meisel, Matthias Wichmann (2010).Container sequencing for quay cranes with internal reshuffles. *OR Spectrum* (32):569–591

Zhen, L., Chew, E. P., & Lee, L. H. (2011). An Integrated Model for Berth Template and Yard Template Planning in Transshipment Hubs. *Transportation Science*, 45(4), 483-504.

Lu Chen, André Langevin, Zhiqiang Lu (2012). Integrated scheduling of crane handling and truck transportation in a maritime container terminal. *European Journal of Operational Research* 225 (2013) 142-152