

A Q-learning System for Container Marshalling with Group-Based Learning Model at Container Yard Terminals

Yoichi Hirashima *

Abstract—This paper addresses scheduling problems on the material handling operation at marine container-yard terminals. The layout, removal order and removal destination of containers are simultaneously optimized in order to reduce the waiting time for a vessel. The schedule of container-movements is derived by autonomous learning method based on a new learning model considering container-groups and corresponding Q-Learning algorithm. In the proposed method, the layout and movements of containers are described based on the Markov Decision Process (MDP), and a state is represented by a container-layout with a selection of a container to be removed or a selection of destination on where the removed container are placed. Then, a state transition arises from a container-movement, a selection of container-destination, or a selection of container to be removed. Only the container-movement takes a cost, and a series of container-movements with selections of destination and order of containers is evaluated by a total amount of costs. As a consequent, the total amount of costs reflects the number of container-movements that is required to achieve desired container-layout. After adequate autonomous learning, the optimum schedule for material handling operation can be obtained by selecting a series of container-movements that has the best evaluation. In the problem, the number of container-arrangements increases by the exponential rate with increase of total count of containers. Therefore, conventional methods have great difficulties to determine desirable movements of containers in order to reduce the run time for shipping.

Keywords: *Scheduling, Container Transfer Problem, Q-Learning, Block Stacking, Reinforcement Learning*

1 Introduction

In recent years, the number of shipping containers grows rapidly, and in many container yard terminals, increasing throughput of material handling operation becomes important issue as well as decreasing the turnaround times of vessels. Material handling operation for loading containers into a vessel is highly complex, and the complexity grows at an exponential rate according to the growth of the number of containers, the operation occupy a large part of the total run time of shipping at container terminals. Thus, improving throughput of the material handling operation for loading container on a vessel is one of main interests at marine terminals. Com-

monly, materials are packed into containers and each container in a vessel has its own position determined by the destination, weight, owner, and so on [1, 2]. Then, containers have to be loaded into a ship in a certain desired order because they cannot be rearranged in the ship. Therefore, containers must be rearranged before loading if the initial layout is different from the desired layout. Containers carried into the terminal are stacked randomly in a certain area called bay and a set of bays are called yard. The rearrangement process conducted within a bay is called marshalling.

In the problem, the number of stacks in each bay is predetermined and the maximum number of containers in a stack is limited. Containers are moved by a transfer crane and the destination stack for the container in a bay is selected from the stacks being in the same bay. In this case, a long series of container movements is often required to achieve a desired layout, and results that are derived from similar initial layouts can be quite different. Although some methods, such as genetic algorithm (GA) and multi agent method [3, 4] have been proposed for solving block stacking problems, environmental models adopted in these methods are different from the marshalling process, and do not apply directly to obtain the desired layout of containers.

Another candidate for solving the problem is the reinforcement learning [5], which is known to be effective for learning under unknown environment that has the Markov Property. The Q-learning, one of the realization algorithm for the reinforcement learning can be applied to generate marshalling plan, when all the estimates of evaluation-values for pairs of the layout and container movement are obtained. These values are called "Q-value". The optimal series of container movements can be obtained by selecting the movement that has the best evaluation for each layout. However, conventional Q-learning has to store evaluation-values for all the layout-movement pairs. Therefore, the conventional Q-learning has great difficulties for solving the marshalling problem, due to its huge number of learning iterations required to obtain admissible plan [6]. Recently, a Q-learning method that can generate marshalling plan has been proposed [7]. Although these methods were effective several cases, the desired layout was not achievable for every trial so that the early-phase performances of learning process can be spoiled. To conquer the drawback, the environmental model that assures the reacha-

* Faculty of Information Science and Technology, Osaka Institute of Technology, 1-79-1, Kita-yama, Hirakata City, Osaka, 573-0196, Japan. Tel/Fax: +86-72-866-5187 Email: hirash-y@is.oit.ac.jp

bility to the desired layout in each trial is proposed [8]. In addition, the environmental model considering groups of containers [9] is shown to be effective to improve the learning performance.

This paper proposes a new environmental model integrated in Q-learning method for marshalling plan in order to improve learning performances. The learning process in the proposed method is consisted of two stages: 1. determination of rearrangement order, 2. selection of destination for removal containers. In both stages, candidates are extended including all the candidates in conventional methods [8, 9], so that the method can find better marshalling plan as compared to conventional methods. In addition, Q-values in one stage are referred from the learning algorithm in the other stage. Stages are repeated sequentially in accordance with container movements and Q-values are discounted according to the number of container movements, then Q-values reflect the total number of container movements. Consequently, selecting the best Q-values leads the best series of container movements required to obtain a desired layout. Moreover, each rearranged container is placed into the desired position so that every trial can achieve one of desired layouts. In addition, in the proposed method, each container has several desired positions in the final layout, and the feature is considered in the learning algorithm. Thus, the early-phase performances of the learning process can be improved.

Finally, effectiveness of the proposed method is shown by computer simulations for several cases.

2 PROBLEM DESCRIPTION

Fig.1 shows an example of container yard terminal. The ter-

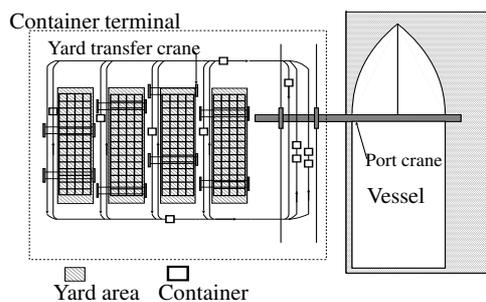


Figure 1: Container terminal

minal consists of containers, yard areas, yard transfer cranes, auto-guided vehicles, and port crane. Containers are carried by trucks and each container is stacked in a corresponding area called bay and a set of bays constitutes a yard area. Each bay has n_y stacks that m_y containers can be laden, the number of containers in a bay is k , and the number of bays depends on the number of containers. Each container is recognized by an unique name c_i ($i = 1, \dots, k$). A position of each container is discriminated by using discrete position numbers, $1, \dots, n_y \cdot m_y$. Then, the position of the container c_i is described by x_i ($1 \leq i \leq k, 1 \leq x_i \leq m_y \cdot n_y$), and the state of a bay is determined by the vector, $x = [x_1, \dots, x_k]$.

2.1 Grouping

The desired layout in a bay is generated based on the loading order of containers that are moved from the bay to a ship. In this case, the container to be loaded into the ship can be anywhere in the bay if it is on top of a stack. This feature yields several desired layouts for the bay.

2.1.1 Groups in horizontal direction

In the addressed problem, when containers on different stacks are placed at the same height in the bay, it is assumed that the positions of such containers can be exchanged. Fig.2 shows an example of desired layouts, where $m_y = n_y = 3, k = 9$. In the figure, containers are loaded in the ship in the descendent order. Then, containers c_7, c_8, c_9 are in the same group (group₁), and their positions are exchanged because the loading order can be kept unchanged after the exchange of positions. In the same way, c_4, c_5, c_6 are in the group₂, and c_1, c_2, c_3 are in the group₃ where positions of containers can be exchanged. Consequently several candidates for desired layout of the bay are generated from the original desired-layout.

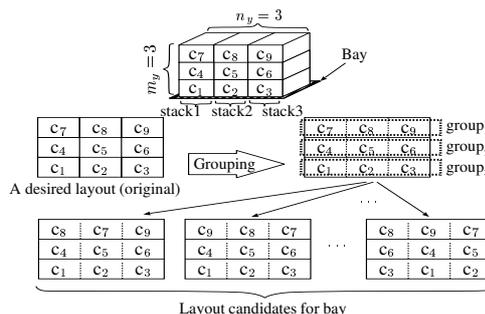


Figure 2: Layouts for bay

2.1.2 Heap shaped group

In addition to the grouping in the horizontal direction, a “heap shaped group” for n_y containers at the top of stacks in original the desired-layout (group₁) is generated as follows:

1. n_y containers in group₁ can be placed at any stacks if their height is same as the original one.
2. Each of them can be stacked on other $n_y - 1$ containers when both of followings are satisfied:
 - (a) They are placed at the top of each stack in the original desired-layout,
 - (b) The container to be stacked is loaded into the ship before other containers being under the container.

Other groups are the same as ones in the original grouping, so that the grouping with heap contains all the desired layout in the original grouping.

Fig.3 depicts an example of heap grouping for $k = 9, n_y = 3$. In the figure, containers are loaded into a vessel by the order c_9, c_8, c_7, \dots . Then, c_9 can be placed on c_7 and c_8 , c_8 can be placed on c_7 , so that the number of desired layouts is increased.

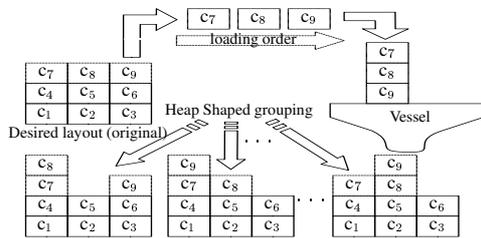


Figure 3: Heap shaped group

2.1.3 Overlapped group

As the main contribution of the paper, the horizontal groups are extended by overlapping adjacent groups to each other. Groups are overlapped by exchanging members in different groups. When group_j is located on group_i, members being in the overlapping area, which can be placed in adjacent group are determined by the following rule :

1. A container c_l in group_j can be placed in group_i if l satisfies

$$l < \frac{n_y}{2} + (k - jn_y),$$

when $i > j$ and loading is conducted with descending order from c_k to c_1 .

2. A container c_r in group_i can be placed in group_j if r satisfies

$$r > \frac{n_y}{2} + (k - in_y),$$

when $i > j$ and loading is conducted with descending order from c_k to c_1 .

Fig.4 shows an example of overlapped group for $k = 9, n_y = 3$. In the example, members of group₁ are $\{c_9, c_8, c_7\}$, ones of group₂ are $\{c_6, c_5, c_4\}$, and ones of group₃ are $\{c_3, c_2, c_1\}$. In group₂ c_4 can be placed in group₃ because c_4 satisfies the rule 1, and in group₁, c_3 can be placed in group₂ because c_3 satisfies the rule 2. When loading is conducted descending order, c_4 can be placed under c_5 and c_6 . c_3 can be placed on c_1 and c_2 . This feature augments the number of candidates for optimum layout as shown in the figure.

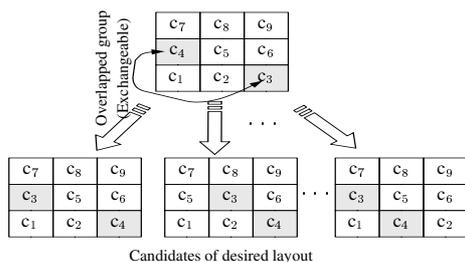


Figure 4: Overlapped group

2.2 Marshalling process

The marshalling process consists of 2 stages: ① selection of a container to be rearranged, and ② removal of the containers on the selected container in ①. After these stages, rearrangement of the selected container is conducted. In the stage ②, the removed container is placed on the destination stack selected from stacks being in the same bay. When a container is rearranged, n_y positions that are at the same height in a bay can be candidates for the destination. In addition, n_y containers can be placed for each candidate of the destination. Then, defining t as the time step, $c_a(t)$ denotes the container to be rearranged at t in the stage ①. $c_a(t)$ is selected from candidates $c_{y_{i_1}}$ ($i_1 = 1, \dots, n_y^2$) that are at the same height in a desired layout. A candidate of destination exists at a bottom position that has undesired container in each corresponding stack. The maximum number of such stacks is n_y , and they can have n_y containers as candidates, since the proposed method considers groups in the desired position. The number of candidates of $c_a(t)$ is thus $n_y \times n_y$. In the stage ②, the container to be removed at t is $c_b(t)$ and is selected from two containers $c_{y_{i_2}}$ ($i_2 = 1, 2$) on the top of stacks. c_{y_1} is on the $c_a(t)$ and c_{y_2} is on the destination of $c_a(t)$. Then, in the stage ②, $c_b(t)$ is removed to one of the other stacks in the same bay, and the destination stack $u(t)$ at time t is selected from the candidates u_j ($j = 1, \dots, n_y - 2$). $c_a(t)$ is rearranged to its desired position after all the $c_{y_{i_2}}$ s are removed. Thus, a state transition of the bay is described as follows:

$$x_{t+1} = \begin{cases} f(x_t, c_a(t)) & \text{(stage ①)} \\ f(x_t, c_b(t), u(t)) & \text{(stage ②)} \end{cases} \quad (1)$$

where $f(\cdot)$ denotes that removal is processed and x_{t+1} is the state determined only by $c_a(t), c_b(t)$ and $u(t)$ at the previous state x_t . Therefore, the marshalling plan can be treated as the Markov Decision Process.

Additional assumptions are listed below:

1. The bay is 2-dimensional.
2. Each container has the same size.
3. The goal position of the target container must be located where all containers under the target container are placed at their own goal positions.
4. $k \leq m_y n_y - 2m_y + 1$

The maximum number of containers that must removed before rearrangement of $c_a(t)$ is $2m_y - 1$ because the height of each stack is limited to m_y . Thus, assumption (4) assures the existence of space for removing all the $c_b(t)$, and $c_a(t)$ can be placed at the desired position from any state x_t .

Figure 5 shows 3 examples of marshalling process, where $m_y = 3, n_y = 5, k = 8$. Positions of containers are discriminated by integers $1, \dots, 15$. The first container to be loaded is c_8 and containers must be loaded by descendent order until c_1 is loaded. In the figure, a container marked with a \square denotes

c_1 , a container marked with a \bigcirc is removed one, and an arrowed line links source and destination positions of removed container. Cases (a),(b) have the same order of rearrangement, c_2, c_7, c_6 , and the removal destinations are different. Whereas, case (c) has the different order of rearrangement, c_8, c_2, c_7 . When no groups are considered in desired arrangement, case (b) requires 5 steps to complete the marshalling process, and other cases require one more step. Thus, the total number of movements of container can be changed by the destination of the container to be removed as well as the rearrangement order of containers.

If groups are considered in desired arrangement, case (b) achieves a goal layout at step2, case (a) achieves at step3, case (c) achieves at step4. If extended groups are considered, cases (a),(b) achieve goal layouts at step2 and case (c) achieves at step4. Since extended goal layouts include the non-extended goal layouts, and since non-extended goal layouts include a non-grouping goal layout, equivalent or better marshalling plan can be generated by using the extended goal notion as compared to plans generated by other goal notions.

The objective of the problem is to find the best series of movements which transfers every container from an initial position to the goal position. The goal state is generated from the shipping order that is predetermined according to destinations of containers. A series of movements that leads a initial state into the goal state is defined as an episode. The best episode is the series of movements having the smallest number of movements of containers to achieve the goal state.

3 REINFORCEMENT LEARNING FOR MARSHALLING PLAN

3.1 Update rule of Q-values

In the selection of c_a , the container to be rearranged, an evaluation value is used for each candidate $c_{y_{i_1}}$ ($i_1 = 1, \dots, r$, where r is the number of candidates. In the same way, evaluation values are used in the selection of the container to be removed c_b and its destination u_j ($j = 1, \dots, n_y - 2$). Candidates of c_b is $c_{y_{i_2}}$ ($i_2 = 1, \dots, n_y$). The evaluation value for the selection of $c_{y_{i_1}}$, $c_{y_{i_2}}$ and u_j at the state x are called Q-values, and a set of Q-values is called Q-table. At the l th episode, the Q-value for selecting $c_{y_{i_1}}$ is defined as $Q_1(l, x, c_{y_{i_1}})$, the Q-value for selecting $c_{y_{i_2}}$ is defined as $Q_2(l, x, c_{y_{i_1}}, c_{y_{i_2}})$ and the Q-value for selecting u_j is defined as $Q_3(l, x, c_{y_{i_1}}, c_{y_{i_2}}, u_j)$. The initial value for both Q_1, Q_2, Q_3 is assumed to be 0.

In this method, a large amount of memory space is required to store all the Q-values referred in every episode. In order to reduce the required memory size, the length of episode that corresponding Q-values are stored should be limited, since long episode often includes ineffective movements of container. In the following, update rule of Q_3 is described. When a series of n movements of container achieves the goal state x_n from an initial state x_0 , all the referred Q-values from x_0

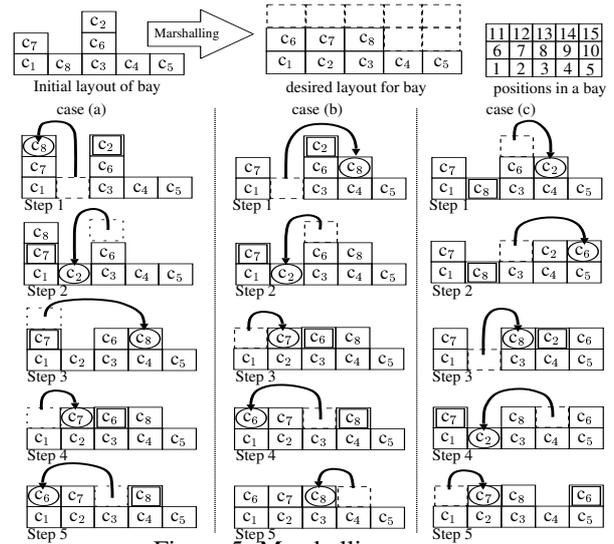


Figure 5: Marshalling process

to x_n are updated. Then, defining L as the total counts of container-movements for the corresponding episode, L_{min} as the smallest value of L found in the past episodes, and s as the parameter determining the threshold, Q_3 is updated when $L < L_{min} + s$ ($s > 0$) is satisfied by the following equation:

$$Q_3(l, x_t, c_a(t), c_b(t), u(t)) = (1 - \alpha)Q_3(l - 1, x_t, c_a(t), c_b(t), u(t)) + \alpha[R + V_{t+1}] \quad (2)$$

$$V_t = \begin{cases} \gamma \max_{y_{i_1}} Q_1(l, x_t, c_{y_{i_1}}) & \text{(stage ①)} \\ \gamma \max_{y_{i_2}} Q_2(l, x_t, c_a(t), c_{y_{i_2}}) & \text{(stage ②)} \end{cases}$$

where γ denotes the discount factor and α is the learning rate. Reward R is given only when the desired layout has been achieved. L_{min} is assumed to be infinity at the initial state, and updated when $L < L_{min}$ by the following equation: $L = L_{min}$.

In the selection of $c_b(t)$, the evaluation value $Q_3(l, x, c_a(t), c_b(t), u_j)$ can be referred for all the u_j ($j = 1 \dots n_y - 2$), and the state x does not change. Thus, the maximum value of $Q_3(l, x, c_a(t), c_b(t), u_j)$ is copied to $Q_1(l, x, c(t))$, that is,

$$Q_2(l, x, c_a(t), c_b(t)) = \max_j Q_3(l, x, c_a(t), c_b(t), u_j). \quad (3)$$

In the selection of $c_a(t)$, the evaluation value $Q_1(l, x, c_a(t))$ is updated by the following equations:

$$Q_1(l, x_t, c_a(t)) = \begin{cases} \max_{y_{i_1}} Q_1(l, x_t, c_{y_{i_1}}) + R & \text{(stage ①)} \\ \max_{y_{i_2}} Q_2(l, x_t, c_a(t), c_{y_{i_2}}) & \text{(stage ②)} \end{cases} \quad (4)$$

In order to select actions, the "ε-greedy" method is used. In the "ε-greedy" method, $c_a(t), c_b(t)$ and a movement that have the largest $Q_1(l, x, c_a(t)), Q_2(l, x, c_a(t), c_b(t))$ and $Q_3(l, x, c_a(t), c_b(t), u_j)$ are selected with probability $1 - \epsilon$ ($0 < \epsilon < 1$), and with probability ϵ , a container and a movement are selected randomly.

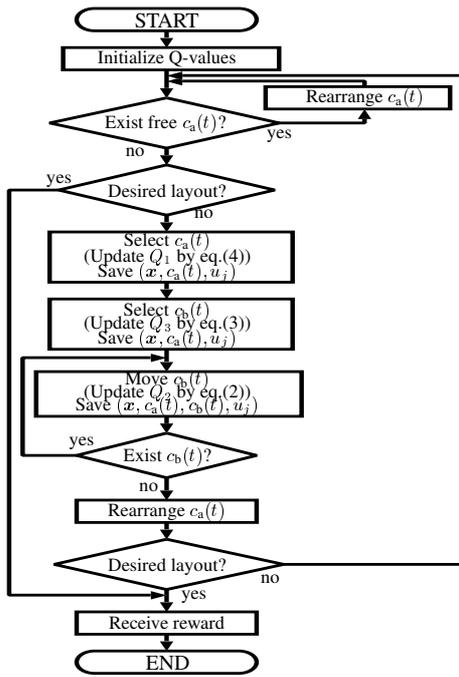


Figure 6: Flowchart of the learning algorithm

3.2 Learning algorithm

By using the update rule, restricted movements and goal states explained above, the learning process is described as follows:

- [1]. Count the number of containers being in the goal positions and store it as n
- [2]. If $n = k$, go to [10]
- [3]. Select $c_a(t)$ to be rearranged
- [4]. Store $(x, c_a(t))$
- [5]. Select $c_b(t)$ to be removed
- [6]. Store $(x, c_a(t), c_b(t))$
- [7]. Select destination position u_j for $c_b(t)$
- [8]. Store $(x, c_a(t), c_b(t), u_j)$
- [9]. Remove $c_b(t)$ and go to [5] if another $c_b(t)$ exists, otherwise go to [1]
- [10]. Update all the Q-values referred from the initial state to the goal state according to eqs. (2), (3)

A flow chart of the learning algorithm is depicted in Figure 6.

4 SIMULATIONS

Computer simulations are conducted for 2 cases, and learning performances are compared for following two methods:

- (A) proposed method using 3 grouping method,
- (B) proposed method only using horizontal and heap shaped grouping,

- (C) a learning method using eqs. (2)-(4) as the update rule without grouping [8],
- (D) method (E) considering original grouping.
- (E) a learning method using eqs. (2),(3) as the update rule, which has no selection of the desired position of $c_a(t)$ [10].

In methods (D),(E), although the stage ② has the same process as in the method (A), the container to be rearranged, $c_a(t)$, is simply selected from containers being on top of stacks. The learning process used in methods (D),(E) is as follows:

- [1]. The number of containers being on the desired positions is defined as k_B and count k_B
- [2]. If $k_B = k$, go to [6] else go to [3],
- [3]. Select $c_a(t)$ by using ϵ -greedy method,
- [4]. Select a destination of $c_a(t)$ from the top of stacks by using ϵ -greedy method,
- [5]. Store the state and go to [1],
- [6]. Update all the Q-values referred in the episode by eqs. (2),(3).

Since methods (D),(E) do not search explicitly the desired position for each container, each episode is not assured to achieve the desired layout in the early-phase of learning.

In methods (A)-(E), parameters in the yard are set as $k = 18, m_y = n_y = 6$ that are typical values of marshalling environment in real container terminals. Containers are assumed to be loaded in a ship in descendant order from c_{18} to c_1 . Figure 7 shows a desired layout for the two cases, and figure 8 shows corresponding initial layout for each case. Other parameters are put as $\alpha = 0.8, \gamma = 0.8, R = 1.0, \epsilon = 0.8, s = 15$.

Results for case 2 are shown in Fig. 9. In the figure, horizontal axis shows the number of trials, and vertical axis shows the minimum number of movements of containers found in the past trials. Each result is averaged over 20 independent simulations. In both cases, solutions that is obtained by methods (A),(B) and (C) is much better as compared to methods (D),(E) in the early-phase of learning, because methods (A),(B),(C) can achieve the desired layout in every trial, whereas methods (D),(E) cannot. Also, methods (A),(B) successfully reduces the number of trials in order to achieve the specific count of container-movements as compared to method (C), since methods (A),(B) considers grouping and finds desirable layouts than can easily diminish the number of movements of container in the early-phase learning. Moreover, at 10000th trail the number of movements of containers in method (A) is smaller as compared to that in method (B) because, among the extended layouts, method (A) obtained better desired layouts for improving the marshalling process as compared to the layout generated by method (B). Desired layouts generated by methods (A),(B) are depicted in the Fig.10 for case 2.

Table 1: The best solution of each method for cases 1, 2

	Case 1		Case 2	
	min. counts	ave. value	min. counts	ave. value
(A)	16	16.90	22	23.00
(B)	18	19.10	23	24.40
Method (C)	34	35.05	35	38.85
(D)	38	46.90	50	64.00
(E)	148	206.4	203	254.0

The container-movement counts of the best solution and its averaged value for each method are described in Table1. Averaged values are calculated over 20 independent simulations. Among the methods, method (A) derives the best solution with the smallest container-movements. Therefore method (A) can improve the solution for marshalling as well as learning performance to solve the problem.

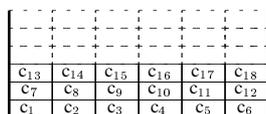


Figure 7: A desired layout for cases 1,2

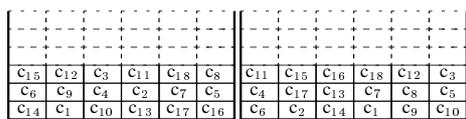


Figure 8: Initial layouts for cases 1,2

5 CONCLUSIONS

A new reinforcement learning system for marshalling plan at container terminals has been proposed. Each container has several desired positions that are in the same group, and the learning algorithm is designed to considering the feature.

In simulations, the proposed method could find solutions that had smaller number of movements of containers as compared to conventional methods. Moreover, since the proposed method achieves the desired layout in each trial as well as learns the desirable layout, the method can generate solutions with the smaller number of trials as compared to the conventional method.

References

[1] Siberholz, M. B., Golden, B. L., Baker, K., "Using Simulation to Study the Impact of Work Rules on Productivity at Marine Container Terminals", *Computers Oper. Res.*, V.18, N.5, pp. 433-452, 1991.
 [2] Günther, H.-O., Kim, K. H., *Container Terminals and Automated Transport Systems*, Springer, pp. 184-206, 2005.

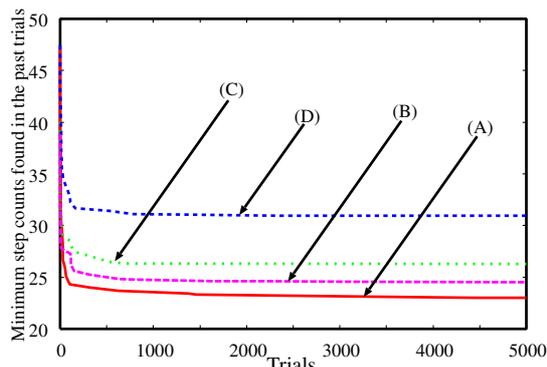


Figure 9: Performance comparison for case 2

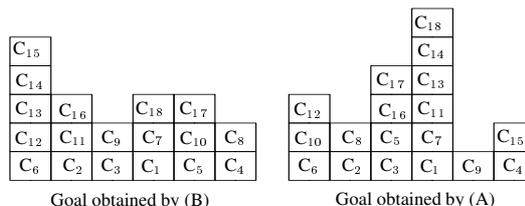


Figure 10: Final layouts of the best solutions for case 2

[3] Koza, J. R., *Genetic Programming : On Programming Computers by means of Natural Selection and Genetics*, MIT Press, 1992.
 [4] Minagawa, M., Kakazu, Y., "An Approach to the Block Stacking Problem by Multi Agent Cooperation", *Trans. Jpn. Soc. Mech. Eng. (in Japanese)*, V.C-63, N.608, pp. 231-240, 1997.
 [5] Watkins, C. J. C. H., Dayan, P., "Q-learning", *Machine Learning*, V.8, pp. 279-292, 1992.
 [6] Baum, E. B., "Toward a Model of Intelligence as an Economy of Agents", *Machine Learning*, V.35, pp. 155-185, 1999.
 [7] Hirashima, Y., Iiguni, Y., Inoue, A., Masuda, S., "Q-Learning Algorithm Using an Adaptive-Sized Q-table", *Proc. IEEE Conf. Decision and Control*, pp. 1599-1604, 1999.
 [8] Hirashima, Y., Takeda, K., Furuya, O., Inoue, A., Deng, M., "A New Method for Marshaling Plan Using a Reinforcement Learning Considering Desired Layout of Containers in Terminals", *Preprint of 16th IFAC World Congress*, We-E16-TO/2, 2005.
 [9] Hirashima, Y., Ishikawa, N., Takeda, K., "A New Reinforcement Learning for Group-Based Marshaling Plan Considering Desired Layout of Containers in Port Terminals", *International Conference on Networking, Sensing, and Control*, pp. 670-675, 2006.
 [10] Motoyama, S., Hirashima, Y., Takeda, K., Inoue, A., "A marshalling plan for container terminals based on reinforcement learning", *Proc. of Inter. Sympo. on Advanced Control of Industrial Processes*, pp. 631-636, 2001.