AN INTELLIGENT MARSHALING PLAN BASED ON MULTI-POSITIONAL DESIRED LAYOUT IN CONTAINER YARD TERMINALS

Yoichi Hirashima
Dept. Media Science, Fac. Information Science and Technology, Osaka Institute of Technology
1-79-1 Kitayama, Hirakata-City, Osaka, 573-0196 Japan

Keywords: Container marshaling, Block stacking problem, Q-learning, Reinforcement learning, Binary tree.

Abstract: This paper proposes a new scheduling method for a marshaling in the container yard terminal. The proposed method is derived based on Q-Learning algorithm considering the *desired position* of containers that are to be loaded into a ship. In the method, 3 processes can be optimized simultaneously: rearrangement order of containers, layout of containers assuring explicit transfer of container to the *desired position*, and removal plan for preparing the rearrange operation. Moreover, the proposed method generates several desired positions for each container, so that the learning performance of the method can be improved as compared to the conventional methods. In general, at container yard terminals, containers are stacked in the arrival order. Containers have to be loaded into the ship in a certain order, since each container has its own shipping destination and it cannot be rearranged after loading. Therefore, containers have to be rearranged from the initial arrangement into the desired arrangement before shipping. In the problem, the number of container-arrangements increases by the exponential rate with increase of total count of containers, and the rearrangement process occupies large part of total run time of material handling operation at the terminal. For this problem, conventional methods require enormous time and cost to derive an admissible result. In order to show effectiveness of the proposed method, computer simulations for several examples are conducted.

1 INTRODUCTION

In recent years, the number of shipping containers grows rapidly, and operations for layout-rearrangement of container stacks occupy a large part of the total run time of shipping at container terminals. Since containers are moved by a transfer crane driven by human operator, and thus, the container operation is important to reduce cost, run time, and environmental burden of material handling systems (Siberholz et al., 1991). Commonly, materials are packed into containers and each container has its own shipping destination. Containers have to be loaded into a ship in a certain desired order because they cannot be rearranged in the ship. Thus, containers must be rearranged before loading if the initial layout is different from the desired layout. Containers carried in the terminal are stacked randomly in a certain area called bay and a set of bays are called yard. When the number of containers for shipping is large, the rearrangement operation is complex and takes long time to achieve the desired layout of containers. Therefore the rearrangement process occupies a large part of the total run time of shipping. The rearrangement process conducted within a bay is called marshaling.

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 movements of containers is often required to achieve a desired layout, and results (the number of container-movements) that are derived from similar layouts can be quite different. Problems of this type have been solved by using techniques of optimization, such as genetic algorithm (GA) and multi agent method (Koza, 1992; Minagawa and Kakazu, 1997). These methods can successfuly yield some solutions for block stacking problems. However, they adopt the environmental model different from the marshaling process, and do not assure to obtain the desired layout of containers.

The Q-learning (Watkins and Dayan, 1992) is known to be effective for learning under unknown environment. In the Q-learning for generating marshaling plan, all the estimates of evaluation-values for pairs of the layout and movement of containers are calculated. These values are called “Q-value” and Q-

Hirashima Y. (2007).
AN INTELLIGENT MARSHALING PLAN BASED ON MULTI-POSITIONAL DESIRED LAYOUT IN CONTAINER YARD TERMINALS.
In *Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics*, pages 234-239
DOI: 10.5220/0001643902340239

table is a look-up table that stores Q-values. The input of the Q-table is the plant state and the output is a Q-value corresponding to the input. A movement is selected with a certain probability that is calculated by using the magnitude of Q-values. Then, the Q-value corresponding to the selected movement is updated based on the result of the movement. The optimal pattern of container movements can be obtained by selecting the movement that has the largest Q-value at each state-movement pair, when Q-values reflect the number of container movements to achieve the desired layout. However, conventional Q-table has to store evaluation-values for all the state-movement pairs. Therefore, the conventional reinforcement learning method, Q-learning, has great difficulties for solving the marshaling problem, due to its huge number of learning iterations and states required to obtain admissible operation of containers (Baum, 1999). Recently, a Q-learning method that can generate marshaling plan has been proposed (Hirashima et al., 1999). 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 degraded.

In this paper, a new reinforcement learning system to generate a marshaling plan is proposed. The learning process in the proposed method is consisted of two stages: ① determination of rearrangement order, ② selection of destination for removal containers. Learning algorithms in these stages are independent to each other and Q-values in one stage are referred from the other stage. That is, Q-values are discounted according to the number of container movement and Q-table for rearrangement is constructed by using Q-values for movements of container, so that Q-values reflect the total number of container movements required to obtain a desired layout. Moreover, in the end of stage ①, selected container is rearranged into the desired position so that every trial can achieve the desired layout. 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 terminal consists of containers, yard areas, yard transfer cranes, auto-guided vehicles, and port crane. Containers are carried by trucks and each container is

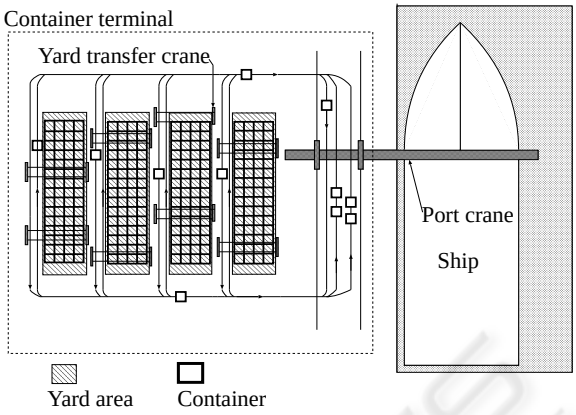

Figure 1: Container terminal.

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 $\mathrm{c}_i\,(i = 1, \cdots, k)$. A position of each container is discriminated by using discrete position numbers, $1, \cdots, n_{\mathrm{y}} \cdot m_{\mathrm{y}}$. Then, the position of the container c_i is described by $x_i\,(1 \leq i \leq k, 1 \leq x_i \leq m_{\mathrm{y}} \cdot n_{\mathrm{y}})$, and the state of a bay is determined by the vector, $x = [x_1, \cdots, x_k]$.

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. 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 $m_{\mathrm{y}} = n_{\mathrm{y}} = 3, k = 9$. In the figure, containers are loaded in the ship in the descendent order. Then, containers $\mathrm{c}_7, \mathrm{c}_8, \mathrm{c}_9$ 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, $\mathrm{c}_4, \mathrm{c}_5, \mathrm{c}_6$ are in the Group2, and $\mathrm{c}_1, \mathrm{c}_2, \mathrm{c}_3$ 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.

In addition to the grouping explained above, a "heap shaped group" for n_{y} containers at the top of stacks in original the desired-layout (group 1) is generated as follows:

1. n_{y} containers in group 1 can be placed at any

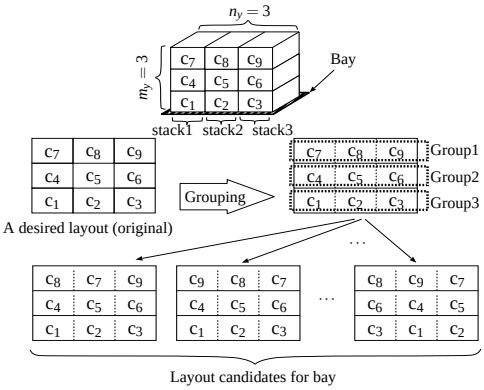

Figure 2: Layouts for bay.

stacks if their height is same as the original one.

2. Each of them can be stacked on other $n_{\mathrm{y}}-1$ containers when both of followings are satisfied:
   (a) They are placed at the top of each stack in the original disired-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.

2.2 Marshaling Process

The marshaling 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, $c_{\mathrm{a}}(t)$ denotes the container to be rearranged at t in the stage ①. $c_{\mathrm{a}}(t)$ is selected from candidates $\mathrm{c}_{y_{i_1}}$ $(i_1 = 1,\cdots,n_{\mathrm{y}}^2)$ 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 $c_{\mathrm{a}}(t)$ is thus $n_{\mathrm{y}} \times n_{\mathrm{y}}$. In the stage ②, the container to be removed at t is $c_{\mathrm{b}}(t)$ and is selected from two containers $\mathrm{c}_{y_{i_2}}$ $(i_2 = 1,2)$ on the top of stacks. c_{y_1} is on the $c_{\mathrm{a}}(t)$ and c_{y_2} is on the destination of $c_{\mathrm{a}}(t)$. Then, in the stage ②, $c_{\mathrm{b}}(t)$ 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 $(j = 1,\cdots,n_{\mathrm{y}}-2)$. $c_{\mathrm{a}}(t)$ is rearranged to its desired position after all the $\mathrm{c}_{y_{i_2}}$s are removed. Thus, a state transition of the bay is described as follows:

$$x_{t+1} = \begin{cases} f(x_t, c_{\mathrm{a}}(t)) & (\text{stage } ①) \\ f(x_t, c_{\mathrm{b}}(t), u(t)) & (\text{stage } ②) \end{cases} \quad (1)$$

where $f(\cdot)$ denotes that removal is processed and x_{t+1} is the state determined only by $c_{\mathrm{a}}(t), c_{\mathrm{b}}(t)$ and $u(t)$ at the previous state x_t. Therefore, the marshaling 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. $k \leq m_{\mathrm{y}} n_{\mathrm{y}} - 2m_{\mathrm{y}} + 1$

The maximum number of containers that must removed before rearrangement of $c_{\mathrm{a}}(t)$ is $2m_{\mathrm{y}}-1$ because the height of each stack is limited to m_{y}. Thus, assumption (4) assures the existence of space for removing all the $c_{\mathrm{b}}(t)$, and $c_{\mathrm{a}}(t)$ can be placed at the desired position from any state x_t.

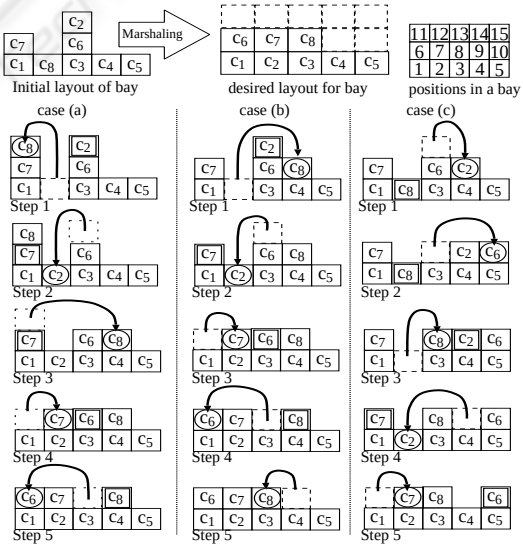

Figure 3: Marshaling process.

Figure 3 shows 3 examples of marshaling process, where $m_{\mathrm{y}} = 3, n_{\mathrm{y}} = 5, k = 8$. Positions of containers are discriminated by integers $1,\cdots,15$. 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 □ denotes c_1, a container marked with a ◯ 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 marshaling 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) achives at step4. If extended groups are considered, cases (a),(b) achive goal layouts at step2 and case (c) achives 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 marshaling 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 MARSHALING 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 $c_{y_{i_1}}$ $(i_1 = 1, \cdots, n_y^2)$. In the same way, evaluation values are used in the selection of the container to be removed c_b and its destination u_j $(j = 1, \cdots, n_y - 2)$. Candidates of c_b is $c_{y_{i_2}}$ $(i_2 = 1, \cdots, n_y)$. 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 lth 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 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:

$$\begin{aligned} &Q_3(l, x_t, c_a(t), c_b(t), u(t)) = \\ &\quad (1-\alpha) Q_3(l-1, x_t, c_a(t), c_b(t), u(t)) \\ &\quad + \alpha [R + V_{t+1}] \\ &V_t = \begin{cases} \gamma \max_{y_{i_1}} Q_1(l, x_t, c_{y_{i_1}}) & (\text{stage } ①) \\ \gamma \max_{y_{i_2}} Q_2(l, x_t, c_a(t), c_{y_{i_2}}) & (\text{stage } ②) \end{cases} \end{aligned} \tag{2}$$

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 $(j = 1 \cdots n_y - 2)$, 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,

$$\begin{aligned} &Q_2(l, x, c_a(t), c_b(t)) = \\ &\quad \max_j Q_3(l, x, c_a(t), c_b(t), u_j). \end{aligned} \tag{3}$$

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

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

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 $1 - \varepsilon (0 < \varepsilon < 1)$, and with probability ε, a container and a movement are selected randomly.

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

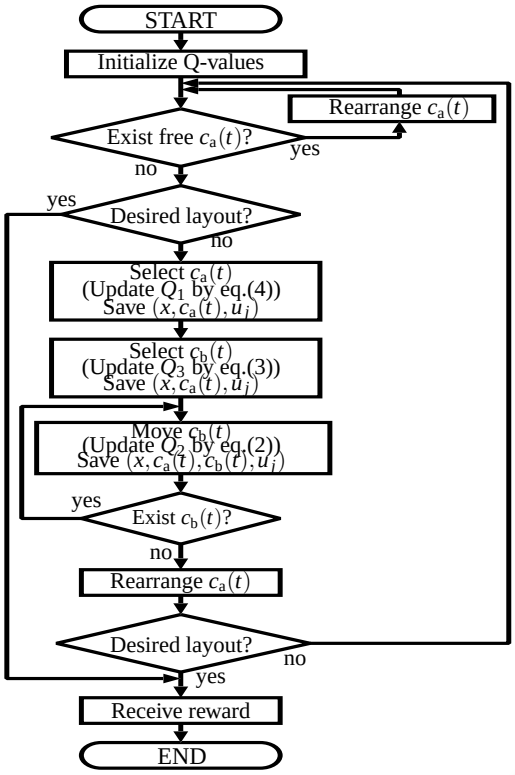

Figure 4: Flowchart of the learning algorithm.

[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 4.

4 SIMULATIONS

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

(A) proposed method considering grouping with heap,

(B) proposed method considering original grouping,

(C) a learning method using eqs. (2)-(4) as the update rule without grouping (Hirashima et al., 2005),

(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 $c_a(t)$ (Motoyama et al., 2001).

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 marshaling environment in real container terminals. Containers are assumed to be loaded in a ship in descendant order from c_{18} to c_1. Figure 5 shows a desired layout for the two cases, and figure 6 shows corresponding initial layout for each case. Other parameters are put as $\alpha = 0.8, \gamma = 0.8, R = 1.0, \varepsilon = 0.8, s = 15$.

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 marshaling as well as learning performance to solve the problem.

Results for case 2 are shown in Fig. 7. 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

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)	18	19.10	23	24.40
(B)	20	20.40	25	26.20
Method (C)	34	35.05	35	38.85
(D)	38	46.90	50	64.00
(E)	148	206.4	203	254.0

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 marshaling process as compared to the layout generated by method (B). Desired layouts generated by methods (A),(B) are depicted in the Fig.8 for case 2.

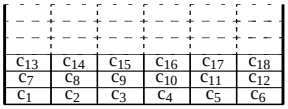

Figure 5: A desired layout for cases 1,2.

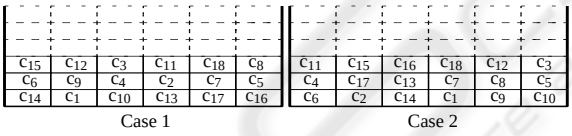

Figure 6: Initial layouts for cases 1,2.

5 CONCLUSIONS

A new reinforcement learning system for marshaling 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.

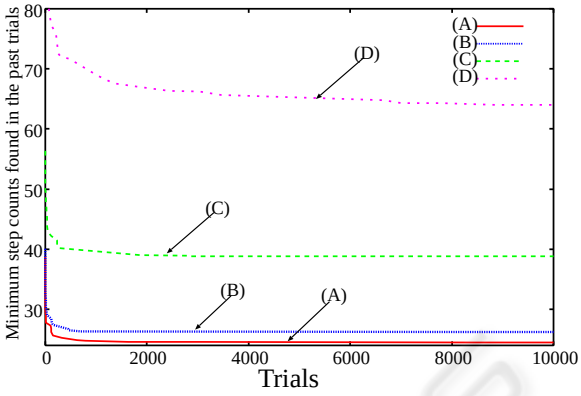

Figure 7: Performance comparison for case 2.

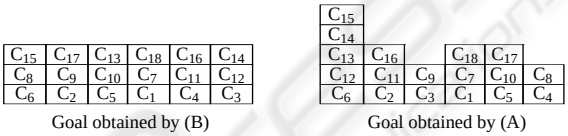

Figure 8: Final layouts of the best solutions for case 2.

REFERENCES

Baum, E. B. (1999). Toward a model of intelligence as an economy of agents. *Machine Learning*, 35:155–185.

Hirashima, Y., Iiguni, Y., Inoue, A., and Masuda, S. (1999). Q-learning algorithm using an adaptive-sized q-table. *Proc. IEEE Conf. Decision and Control*, pages 1599–1604.

Hirashima, Y., takeda, K., Furuya, O., Inoue, A., and Deng, M. (2005). A new method for marshaling plan using a reinforcement learning considering desired layout of containers in terminals. *Preprint of 16th IFAC World Congress*, pages We–E16–TO/2.

Koza, J. R. (1992). *Genetic Programming : On Programming Computers by means of Natural Selection and Genetics*. MIT Press.

Minagawa, M. and Kakazu, Y. (1997). An approach to the block stacking problem by multi agent cooperation. *Trans. Jpn. Soc. Mech. Eng. (in Japanese)*, C-63(608):231–240.

Motoyama, S., Hirashima, Y., Takeda, K., and Inoue, A. (2001). A marshalling plan for container terminals based on reinforce-ment learning. *Proc. of Inter. Sympo. on Advanced Control of Industrial Processes*, pages 631–636.

Siberholz, M. B., Golden, B. L., and Baker, K. (1991). Using simulation to study the impact of work rules on productivity at marine container terminals. *Computers Oper. Res.*, 18(5):433–452.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.