Proceedings of the International MultiConference of Engineers and Computer Scientists 2010 Vol I,
IMECS 2010, March 17 - 19, 2010, Hong Kong

A Reinforcement Learning System for Transfer Scheduling of Freight Cars in a Train

Yoichi Hirashima *

Abstract—**In this paper, a Q-Learning method for trasfer scheduling of freight cars in a train is proposed. In the proposed method, the number of freight-movements in order to line freights in the desired order is reflected by evaluation value for each pair of freight-layout and removal-destination at a freight yard. The best transfer scheduling can be derived by selecting the removal-action of freight that has the best evaluation value at each freight-layout.**

Keywords: Scheduling, Container Transfer Problem, Q-Learning, Freight train, Marshalling

1 INTRODUCTION

In recent years, logistics with freight train is took attentions in ecological aspects as compared to goods transportation with tracks. A freight train consists of several railway cars, and each car has one or several containers. Commonly, goods are packed into containers and each container in a freight train has its own destination. Since freight trains can transport goods only between railway stations, modal shifts are required for delivering them to area that has no railway. In intermodal transports from the road and the rail, containers carried into the station are loaded on freight cars in the arriving order. The initial layout of freight cars is thus random. For efficient shift, the desirable layout should be determined considering destination of container. Then, freight cars must be rearranged before jointing to the freight train.

In general, the rearrangement process is conducted in a freight yard that consists of a main-track and several sub-tracks. Freight cars are initially placed on sub tracks, rearranged, and lined into the main track. Although similar problems are treated by mathematical programming and genetic algorithm[1, 2, 3, 4], they do not evaluate the number of movements of freight cars.

In this paper, a new scheduling method is proposed in order to rearrange and line freight cars by the desirable order onto the main track. In the proposed method, the focus is centered on to reduce the number of car-movements that achieves desirable order on the main track. The optimal layout of freight cars in the main track is derived based on the destination of freight cars. This yields several desirable layouts of freight cars in the main track, and the optimal layout that can achieve the smallest number of car-movements is obtained by autonomous learning. Simultaneously, the optimal sequence of car-movements that can achieve the desired layout is obtained by autonomous learning. In other words, each freight car has several desired positions in the main track. Also, the feature is considered in the learning algorithm, so that, at each arrangement on sub track, an evaluation value represents the smallest number of car-movements to achieve the best layout on the main track. The learning algorithm is derived based on the Q-Learning[5], which is known as one of the well established realization algorithm of the reinforcement learning.

In the learning algorithm, the state is defined by using a layout of freight cars, the car to be moved, and the destination of the removed car. An evaluation value called Q-value is assigned to each state, and the evaluation value is calculated by several update rules based on the Q-Learning algorithm. Update rules are independent to each other and the Q-value in one update rule is referred from another update rule, so that Q-values are discounted according to the number of car-movements. Consequently, Q-values at each state represent the total number of car-movements required to achieve the best layout from the state. Moreover, in the proposed method, only referred Q-values are stored by using table look-up technique, and the table is dynamically constructed by binary tree in order to obtain the best solution with feasible memory space. In order to show effectiveness of the proposed method, computer simulations are conducted for two methods.

2 PROBLEM DESCRIPTION

The yard consist of 1 main track and m sub tracks. Define k as the number of freight cars placed on the sub tracks, and they are carried to the main track by the desirable order based on their destination. In the yard, a locomotive moves freight cars from sub track to sub track or from sub track to main track. The movement of freight cars from sub track to sub track is called removal, and the car-movement from sub track to main track is called rearrangement. For simplicity, the maximum number of freight cars that each sub track can have is assumed to be n, the ith car is recognized by an unique symbol c_i ($i = 1, \cdots, k$), and the number of sub tracks is l. Fig.1 shows the outline of freight yard in the case $k = 30, m = n = 6$. In the figure, track $\mathrm{T_m}$ denotes the main track, and other tracks A, B, C, D, E, F are sub tracks. The main track is linked with sub

*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

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

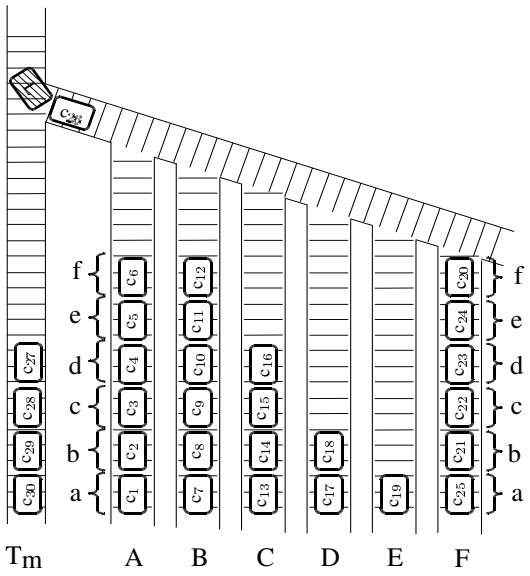

Figure 1: Freight yard

tracks by a joint track, which is used for moving cars between sub tracks, or for moving them from a sub track to the main track. In the figure, freight cars are moved from sub tracks, and lined in the main track by the descending order, that is, rearrangement starts with c_{30} and finishes with c_1. When the locomotive L moves a certain car, other cars locating between the locomotive and the car to be moved must be removed to other sub tracks. This operation is called removal. Then, if $k \leq n \cdot m - (n-1)$ is satisfied for keeping adequate space to conduct removal process, every car can be rearranged to the main track.

In each sub track, positions of cars are defined by n rows. Every position has unique position number represented by $m \cdot n$ integers. Fig.2 shows an example of position index for $k = 30, m = n = 6$ and the layout of cars for fig.1.

31	32	33	34	35	36
25	26	27	28	29	30
19	20	21	22	23	24
13	14	15	16	17	18
7	8	9	10	11	12
1	2	3	4	5	6

Position index

c_6	c_{12}				c_{24}
c_5	c_{11}				c_{23}
c_4	c_{10}	c_{16}			c_{22}
c_3	c_9	c_{15}			c_{21}
c_2	c_8	c_{14}	c_{18}		c_{20}
c_1	c_7	c_{13}	c_{17}	c_{19}	c_{25}

Yard layout

Figure 2: Example of position index and yard state

In Fig.2, the position "aA" that is located at row "a" in the sub track A has the position number 1, and the position "fF" has the position number 36. For unified representation of layout of car in sub tracks, cars are placed from the row "a" in every track, and newly placed car is jointed with the adjacent freight car. In the figure, in order to rearrange c_{25}, $c_{24}, c_{23}, c_{22}, c_{21}$, and c_{20} have to be removed to other sub tracks. Then, since $k \leq n \cdot m - (n-1)$ is satisfied, c_{25} can be moved even when all the other cars are placed in sub tracks.

In the freight yard, define $x_i (1 \leq x_i \leq n \cdot m, i = 1, \cdots, k)$ as the position number of the car c_i, and $\boldsymbol{s} = [x_1, \cdots, x_k]$ as the state vector of the sub tracks. For example, in Fig.2, the state is represented by $\boldsymbol{s} = [1, 7, 13, 19, 25, 31, 2, 8, 14, 20, 26, 32, 3, 9, 15, 21, 4, 10, 5, 12, 18, 24, 30, 36, 6]$.
A trial of the rearrange process starts with the initial layout, rearranging freight cars according to the desirable layout in the main track, and finishs when all the cars are rearranged to the main track.

3 DESIRED LAYOUT IN THE MAIN TRACK

In the main track, freight cars that have the same destination are placed at the neighboring positions. In this case, removal operations of these cars are not required at the destination regardless of layouts of these cars. In order to consider this feature in the desired layout in the main track, a group is organized by cars that have the same destination, and these cars can be placed at any positions in the group. Then, for each destination, make a corresponding group, and the order of groups lined in the main track is predetermined by destinations. This feature yields several desired layouts in the main track.

Fig.3 depicts examples of desirable layouts of cars and the desired layout of groups in the main track. In the figured, freight cars $c_1, \cdots, c_6$ to the destination$_1$ make group$_1$, $c_7, \cdots, c_{18}$ to the destination$_2$ make group$_2$, $c_{19}, \cdots, c_{25}$ to the destination$_3$ make group$_3$, and $c_{26}, \cdots, c_{30}$ to the destination$_4$ make group$_4$. Groups$_{1,2,3,4}$ are lined by ascending order in the main track, which make a desirable layout. In the figure, examples of layout in group$_1$ are in the dashed square.

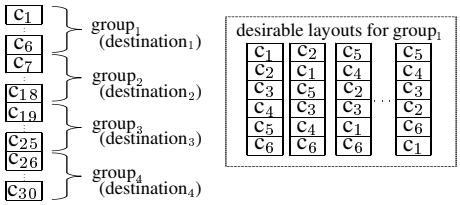

Figure 3: Example of groups

4 REARRANGEMENT PROCESS

The rearrangement process for cars consists of following 4 operations :

(1) selection of a freight car to be rearranged into the main track,

(2) selection of a removal destinations of the cars on the selected car in (1),

(3) removal of the cars to the selected sub track,

(4) rearrangement of the selected car.

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

These operations are repeated until one of desirable layouts is achieved in the main track, and the series of operations from the initial state to the desirable layout is define as a trial.

In the operation (1), each gourp has the predetermined position in the main track, candidates of the gargo to be rearranged can be determined by freight cars that have already rearranged to the main track. These candidates belongs to the same group, and can be rearranged by any order.

Now, define r as the number of groups, g_l as the number of freight cars in group$_l (1 \leq l \leq r)$, and $u_{j_1} (1 \leq j_1 \leq g_l)$ as candidates of cars to be rearranged to the main track.

In the operation (2), the removal destination of car located on the car to be rearranged is selected. Then, defining $u_{j_2} (g_l + 1 \leq j_2 \leq g_l + m - 1)$ as candidates of the destination, excluding the sub track that has the car to be removed, and the number of candidates is $m - 1$.

When rearranging car that has no car to be removed on it is exist, its rearrangement precede any removals. In the case that several cars can be rearranged without a removal, rearrangements are repeated until all the candidates for rearrangement requires at least one removal. If several candidates for rearrangement require no removal, the order of selection is random, because any orders satisfy the desirable layout of groups in the main track. In this case, the arrangement of cars in sub tracks obtained after rearrangements is unique, so that the movements count of cars has no corelation with rearrngement orders of cars that require no removal.

Fig.4 shows an example of arrangement in sub tracks existing candidates for rearranging cars that require no removal. In the figure, $r = 2$, where $\mathrm{c}_1, \mathrm{c}_2, \mathrm{c}_3, \mathrm{c}_4$ are in group$_1$, $\mathrm{c}_5, \mathrm{c}_6, \mathrm{c}_7, \mathrm{c}_8$ are in group$_2$, and group$_1$ must be rearranged first to the main track. In each group, any layouts of cars can be acceptable. In "Case1" of the example, the rearrangement order of cars that require no removal is $\mathrm{c}_1, \mathrm{c}_2, \mathrm{c}_3, \mathrm{c}_4$, and in "Case2", the order is $\mathrm{c}_3, \mathrm{c}_2, \mathrm{c}_1, \mathrm{c}_4$. Although 2 cases have defferent orders of rearrangemt, the arrangements of cars in sub tracks and the numbers of movements of cars have no difference.

5 LEARNING ALGORITHM

Define c_T as the car to be rearranged, $s(t)$ as the state at time t, $s^\dagger(t) = [s(t), \mathrm{c}_T]$ and Q_1, Q_2 as evaluation values for $(s^\dagger(t), u_{j_1})$, $(s^\dagger(t), u_{j_2})$, respectively. $Q_1(s^\dagger(t), u_{j_1})$ and $Q_2(s^\dagger(t), u_{j_2})$ are updated by following rules:

$$Q_1(s(t), u_{j_1}) \leftarrow \begin{cases} \max Q_2(s^\dagger(t), u_{j_2}) \quad \text{ⓐ} \\ \quad \text{(next operation is removal)}, \\ \max Q_1(s^\dagger(t), u_{j_1}) \quad \text{ⓑ} \\ \quad \text{(repetitive rearrangement)}, \end{cases} \tag{1}$$

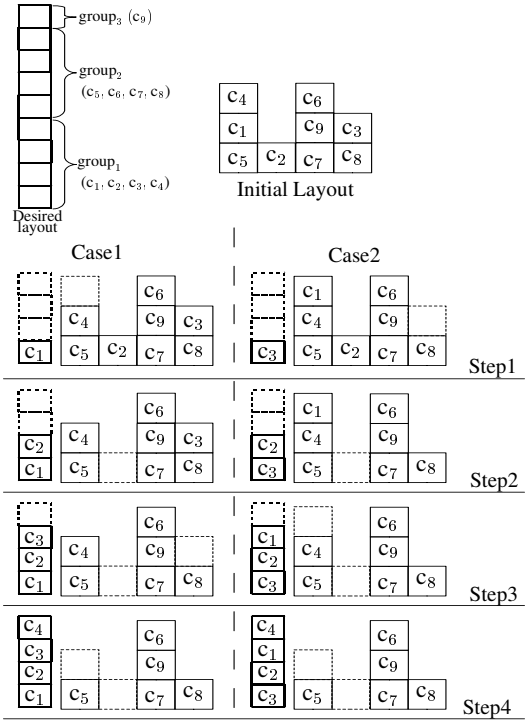

Figure 4: Direct rearrangements

$$Q_2(s^\dagger(t), u_{j_2}) \leftarrow \begin{cases} (1-\alpha) Q_2(s^\dagger(t), u_{j_2}) + \alpha[R \\ \quad + \gamma \max Q_1(s(t+1), u_{j_1}(t+1))] \ \text{ⓐ} \\ \quad \text{(next operation is rearrangement)} \\ (1-\alpha) Q_2(s^\dagger(t), u_{j_2}) + \alpha[R \\ \quad + \gamma \max Q_2(s^\dagger(t+1), u_{j_2}(t+1))] \ \text{ⓑ} \\ \quad \text{(repetitive removal)} \end{cases} \tag{2}$$

where α is the learning rate, γ is the discount factor, and R is the reward that is given when one of desirable layout is achieved.

Propagating Q-values by using eqs.(1),(2), Q-values are discounted according to the number of removals of cars. In other words, by selecting the removal destination that has the largest Q-value, the number of removals can be reduced. In the learning stages, each u_j $(n_\mathrm{c} + 1 \leq j \leq n_\mathrm{c} + n_\mathrm{y} - 1)$ is selected by the following probability:

$$P(s^\dagger, u_j) = \frac{\exp(Q_{t-1}(s^\dagger, u_j)/T)}{\sum_u \exp(Q_{t-1}(s^\dagger, u)/T)}, \quad (n_\mathrm{c} + 1 \leq j \leq n_\mathrm{c} + n_\mathrm{y} - 1) \tag{3}$$

The proposed learning algorithm can be summarized as follows:

1. Initialize all the Q-values as 0

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

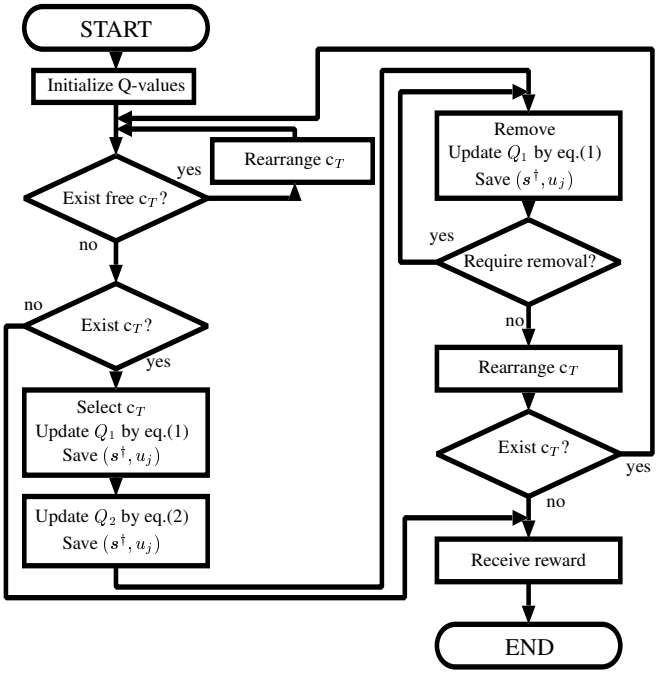

Figure 5: Flowchart of the learning algorithm

2. When no cars are placed on candidates of c_T, all of them are rearranged
3. If no cars are in sub tracks, go to 9, otherwise go to 4
4. ⓐ Determine c_T among the candidates by roulette selection (probabilities are calculated by eq. (3)),

   ⓑ putting reward as $R = 0$,

   ⓒ update the corresponding $Q_1(s^\dagger, u_{j_1})$ by eq. (1),

   ⓓ store $(s^\dagger, u_{j_1})$
5. If cars to be removed exists, update the corresponding $Q_2(s^\dagger, u_{j_2})$ by eq. (2a), and store $(s^\dagger, u_{j_2})$, otherwise update the $Q(s^\dagger, u_{j_2})$ by eq. (2b) and go to 2
6. ⓐ If the car to be removed exists, remove it. The destination of the car to be removed is determined by roulette selection (probabilities are calculated by eq. (3)),

   ⓑ update the corresponding $Q_1(s^\dagger, u_{j_1})$ by eq.(1),

   ⓒ store the state $(s^\dagger, u_{j_1})$,

   ⓓ repeat 6-ⓐ∼ 6-ⓒ until all the cars on c_T are removed
7. Rearrange the c_T
8. If there exist cars, go to 2
9. If all the cars are rearranged, the reward R is given, and update a Q-value according to the last movement of car.

Also, flowchart of the proposed learning algorithm is shown in Fig.5.

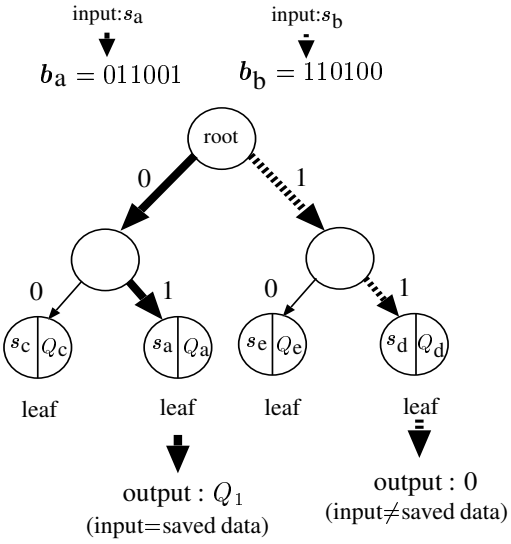

Figure 6: Structure of look-up table

6 DATA STRUCTURE OF LOOK-UP TABLE FOR Q-VALUE

In the learning algorithm explained previous section, the table lookup method is used for storing and referring Q-values. Since the state of the sub tracks is replesented by $s = [x_1, \cdots, x_k]$, $(1 \leq x_i \leq n \cdot m, i = 1, \cdots, k)$, the state space requires $(nm)^k$ memory units to store Q-values for each selection. Thus, the number of memory units increases by the exponential rate with increase of the total number of cars k.

In realistic problems the number of car is often large, so that huge memory size is required in order to store Q-values for all the states. Therefore, in the proposed method, only Q-values corresponding states that have been searched are stored. Binary tree is constructed dynamically during the course of the learning for storing Q-values.

6.1 Specification of a Q-value

In the following, the method to specify a Q-value stored in a look-up table is explained. The input of the table is (s, u_j), and the output is a Q-value. Assuming I is the order of binary description of $m \cdot n$, the Q-value corresponding to a state s is specified.

$b_i = b_{i1} \cdots b_{iI}$ $(b_{ij} = 0, 1 \quad j = 1, \cdots, I)$ is defined as the binary description of x_i $(i = 1, \cdots, k)$. Then, the binary description of s can be described by $b = b_1 \cdots b_k$ of order $(k+1)I$. That is, a binary tree of depth $(k+1)I$ can represent s. At each node of the binary tree, by assigning 0 to left descendant of the node and 1 to right descendant, b_{ij} can specify the descendant at the node of depth $I(i-1)+j$. Each leaf of the tree stores a state and corresponding Q-value. Given an input to the look-up table, the leaf corresponding to the input is specified by a search using b. When the input

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

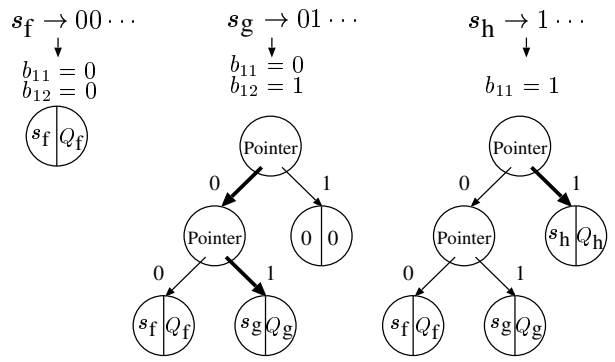

Figure 7: Construction process of Q-table

corresponds to the value stored by the leaf, the look-up table outputs the Q-value stored by the leaf. Otherwise, the table outputs 0. Fig.6 depicts a Q-table constructed by binary tree in the case of $k = m = 2, n = 3, I = 3$. In the figure, inputs $\boldsymbol{s}_{\mathrm{a}} = [1, 3], \boldsymbol{s}_{\mathrm{b}} = [6, 4]$ are given to the look-up table. Since $\boldsymbol{b}_{\mathrm{a}} = [011001]$, in the former case, left, right descendants are specified from the root, the leaf stores the same state as the input $\boldsymbol{s}_{\mathrm{a}}$, and thus outputs Q_1. While, in the latter case, since $\boldsymbol{b}_{\mathrm{b}} = [110100]$, right, right descendants are specified, and the leaf stores $\boldsymbol{s}_{\mathrm{c}} (\neq \boldsymbol{s}_{\mathrm{b}})$. That is, the state that leaf has is different from the input, and thus 0 is output from the look-up table.

6.2 Constitution of look-up table

Initially, the tree has only root that has pointer to an empty leaf. When a Q-value corresponding to the input state is different from initial value, the state and the Q-value are stored in the leaf. Then, if a new state that has the same b_{11} as the stored state, 2 consecutive memory units for storing pointer to leafs storing data of state and Q-value are newly allocated. The pointer to the newly allocated memory units is stored in the ascendant node (in this case, root). These operations are repeated until b_{ij} of the new state is different from that of the stored state. If the new state has the different b_{ij}, 1 memory unit is newly allocated for storing the new state and Q-value in addition to 2 memory units for storing pointer to leaf. The pointer to these 2 memory units is stored to the ascendant node. Pointers to leafs are stored in accordance with b_{ij} in 2 memory units that are newly allocated. For nodes that has no descendant nodes or leaf, 0 is stored as the initial value to indicate being empty.

Whenever the state that has updated Q-value appears, the input and the state stored in the specified leaf are compared. When they have the same value, the stored Q-value is update. Otherwise, excluding b_{ij} used to specifiy the stored state, 2 memory units are newly allocated until first difference between 2 states appears in b_{ij}, and then, a unit is allocated as a leaf to store 2 pairs of state and Q-value.

Fig.7 shows an example of the look-up table for the proposed learning algorithm. In the figure, inputs $\boldsymbol{s}_{\mathrm{f}} = [1, 3], \boldsymbol{s}_{\mathrm{g}} = [2, 1], \boldsymbol{s}_{\mathrm{h}} = [4, 1]$ are given to the table. Then, binary descriptions of $\boldsymbol{s}_{\mathrm{f}}, \boldsymbol{s}_{\mathrm{g}}, \boldsymbol{s}_{\mathrm{h}}$ are $\boldsymbol{b}_{\mathrm{f}} = [b_{11}b_{12}b_{13}b_{21}b_{22}b_{23}] = 001011, \boldsymbol{b}_{\mathrm{f}}\beta = 010001, \boldsymbol{b}_{\mathrm{g}} = 100001$, respectively. First, $\boldsymbol{s}_{\mathrm{f}}$ is stored in the leaf pointed by the root. Next, when $\boldsymbol{s}_{\mathrm{g}}$ is given, 2 memory units are newly allocated. Since b_{11} of $\boldsymbol{s}_{\mathrm{f}}$ is 0, $\boldsymbol{s}_{\mathrm{f}}$ and Q_α are stored in the left leaf. $\boldsymbol{s}_{\mathrm{f}}, \boldsymbol{s}_{\mathrm{g}}$ have the same b_{11}, and thus, 2 memory units are allocated additionally. Since they have difference in b_{12}, $\boldsymbol{s}_{\mathrm{f}}$ and Q_{f} are stored left leaf, and $\boldsymbol{s}_{\mathrm{g}}$ and Q_{g} are stored right leaf. Finally, when $\boldsymbol{s}_{\mathrm{h}}$ is given, $\boldsymbol{s}_{\mathrm{h}}$ and Q_{h} are stored right leaf at the height 1 according $b_{11} = 1$.

The algorithm for look-up table construction is described below.

1. Calculate $\boldsymbol{b}$ from $\boldsymbol{s}$ and initialize $i = j = 1$
2. If a memory unit corresponding to b_{ij} is a leaf then go to (3), and if it is node then go to (4)
3. update i, j by
$$\begin{cases} j \leftarrow j+1, & i \leftarrow i & (j < I), \\ j \leftarrow 1, & i \leftarrow i+1 & (j = I), \end{cases} \quad (4)$$
and go to (2).
4. Conduct eq.(4) again, allocate 2 nodes for expanding a tree, and 1 leaf for storing state and Q-value. Then, copy data from original leaf into corresponding leaf, and store the pointers indicating a new leaf and nodes into ascendant nodes.
5. If b_{ij} has the same number as the state stored in the leaf, go to (4). Otherwise store the new input and Q-value into the corresponding leaf.

7 COMPUTER SIMULATIONS

Computer simulations are conducted for $m = 12, n = 6, k = 36$. The initial arrangement of cars in sub tracks is described in Fig.8, desirable layout considering groups in the main track is depicted in Fig.9. In this case, the rearrantement order of groups is $\mathrm{group}_1, \mathrm{group}_2, \mathrm{group}_3, \mathrm{group}_4$. Cars $\mathrm{c}_1, \cdots, \mathrm{c}_9$ are in group_1, $\mathrm{c}_{10}, \cdots, \mathrm{c}_{18}$ are in group_2, $\mathrm{c}_{19}, \cdots, \mathrm{c}_{27}$ are in group_3, and $\mathrm{c}_{28}, \cdots, \mathrm{c}_{36}$ are in group_4. Other parameters are set as $\alpha = 0.9, \gamma = 0.9, R = 1.0$.

Fig.10 shows the results for the case 1. In the figure, horizontal axis expresses the number of trials and the vertical axis expresses the minimum number of removals of cars to achieve a desirable layout found in the past trials. Each results is averaged over 10 independent simulations. As the number of trials increases, the number of removals reduces, and within 1200 trials, all simulations derive the layout of main track, rearrangement order of cars, removal destination of cars to achieve the best solution that include one removal. Fig.11 shows one of best layout in the main track obtained by the proposed method. In the figure, positions of cagos in the same group are exchanged so that the number of removals required

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

to achieve the layout of groups in the main track is reduced. Thus, in the proposed method, the layout of main track, the rearrangement order of cars, and the removal destination of cars are simultaneously optimized by the autonomous learning.

c_{10}											
c_1	c_4	c_7	c_{11}		c_{16}	c_{19}	c_{22}	c_{25}	c_{28}	c_{31}	c_{34}
c_2	c_5	c_8	c_{12}	c_{14}	c_{17}	c_{20}	c_{23}	c_{26}	c_{29}	c_{32}	c_{35}
c_3	c_6	c_9	c_{13}	c_{15}	c_{18}	c_{21}	c_{24}	c_{27}	c_{30}	c_{33}	c_{36}

Figure 8: Initial layout

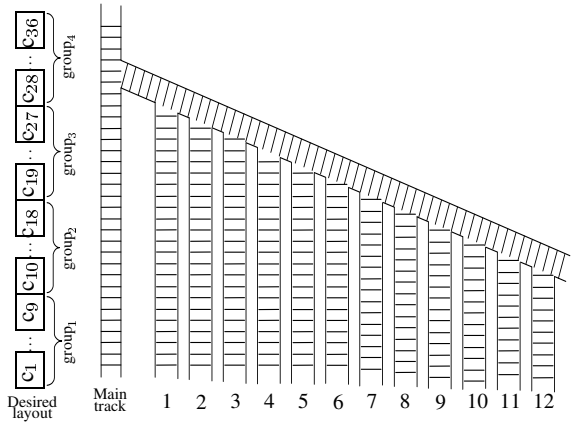

Figure 9: Yard setting

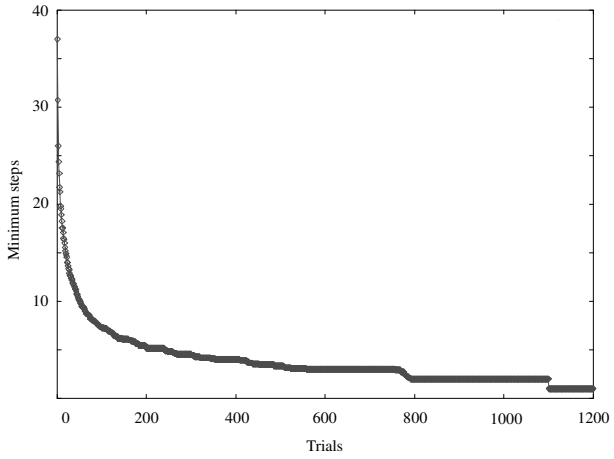

Figure 10: Minimum steps averaged over 10 simulations

8 CONCLUSIONS

A new scheduling method has been proposed in order to rearrange and line cars in the desirable order onto the main track. The learning algorithm of the proposed method is derived based on the reinforcement learning, considering group of cars. In order to obtain the best solution with feasible memory space, in the proposed method, only referred Q-values are

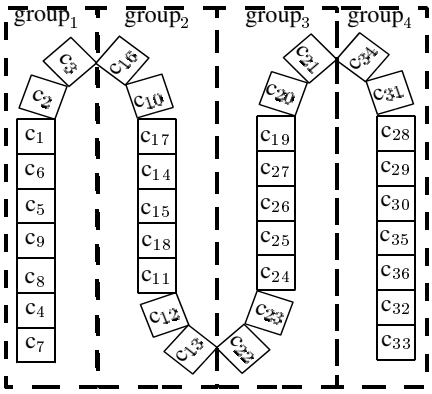

Figure 11: Final layout

stored in look-up tables constructed dynamically by using binary tree. In computer simulations, by using the proposed method, the layout of main track, the rearrangement order of cars, and the removal destination of cars to achieve the optimal solution has been obtained simultaneously.

References

[1] L.G. Kroon, R.M. Lentink and A. Schrijver : Shunting of passenger train units: an integrated approach, Transportation Science Vol. 42, pp. 436–449 (2008)

[2] N. TOMII, and Z. L. Jian : Depot shunting scheduling with combining genetic algorithm and PERT, Proceedings of 7th International Conference on Computer Aided Design, Manufacture and Operation in the Railway and Other Advanced Mass Transit Systems, pp. 437–446 (2000)

[3] S. He, R. Song and S.S. Chaudhry : Fuzzy dispatching model and genetic algorithms for railyards operations, European Journal of Operational Research, Vol. 124, No. 2, pp. 307-331 (2000)

[4] E. Dahlhaus, F. Manne, M. Miller and J. Ryan : Algorithms for Combinatorial Problems Related to Train Marshalling, In Proceedings of AWOCA 2000, pp7–16 (2000)

[5] Watkins, C. J. C. H., Dayan, P., Q-learning, Machine Learning, Vol.8, pp. 279–292, 1992.

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)