

Towards Automatic Facility Layout Design Using Reinforcement Learning

Hikaru Ikeda, Hiroyuki Nakagawa, Tatsuhiro Tsuchiya Graduate School of Information Science and Technology Osaka University, Japan {h-ikeda, nakagawa, t-tutiya}@ist.osaka-u.ac.jp

Abstract—The accuracy and perfection of layout designing significantly depend on the designer's ability. Quick and nearoptimal designs are very difficult to create. In this study, we propose an automatic design mechanism that can more easily design layouts for various unit groups and sites using reinforcement learning. Specifically, we devised a mechanism to deploy units to be able to fill the largest rectangular space in the current site. We aim to successfully deploy given units within a given site by filling a part of the site. We apply the mechanism to the three sets of units in benchmark problems. The performance was evaluated by changing the learning parameters and iteration count. Consequently, it was possible to produce a layout that successfully deployed units within a given one-floor site.

Index Terms—reinforcement learning, machine learning, layout design, facility layout problem

I. INTRODUCTION

THE FACILITY layout problem (FLP) [1], [2] is an optimization problem of deploying equipment and machines in a facility. The solution to this problem depends on the designer's ability. Therefore, various methods and procedures have been proposed to solve the FLP. For example, Genetic Algorithm (GA), Genetic Programming (GP), and deep learning are used. In this study, we propose a layout design support mechanism using reinforcement learning. Reinforcement learning is a machine learning method in which learning is performed without training data. An agent learns through trial and error. It makes it possible to obtain layouts for different conditions from training ones. We make learning agent possible to create a layout in various environment by predetermining the maximum layout area, and generating layouts for the sites that are less than that. The object of this study is to clarify whether reinforcement learning is effective for solving the FLP.

The structure of this paper proceeds as follows. Section 2 explains the layout design, current problems, and reinforcement learning as the background of this study. Section 3 explains the layout design mechanism using reinforcement learning. Section 4 describes the results of the experiments conducted to demonstrate the effectiveness of the proposed mechanism and performance. Section 5 describes the discussion based on the experiments presented in Section 4. Section 6 describes the prospects, such as the problems found and future issues, and Section 7 concludes the paper.

II. BACKGROUND

A. Layout design

The FLP is a well studied combinatorial optimization problem which arises in a variety of problems such as layout design of the deployment and flow lines of equipment, materials, parts, work-in-process, workers, circuit board design, warehouses, backboard wiring problems [3]. An efficient and rational layout is desirable for the layout design process. The purpose of layout design is to obtain a layout that is close to optimal under the specified conditions.

Facility layout design has been studied for a long time [4]. Many of them use GA [5] and GP [6]. GA is a programming technique which forms its basis from the biological evolution. GA uses the principles of selection and evolution to produce several solutions to a given problem. GP is considered to be a variant of GA, and used to evolve abstractions of knowledge, like mathematical expressions or rule-based systems. Venkatesh and Jim [7] use GA to reduce the sum of the product of the three factors of material handling cost, which are: the volume of material handling (frequency of journeys); the cost of material handling, and the distance travelled. José et al. [8] introduced biased random-key GA for the unequal area facility layout problem where a set of rectangular facilities with given area requirements has to be placed, without overlapping, on a rectangular floor space. The role of GA is to evolve the encoded parameters that represent the facility placement sequence, the vector of facility aspect ratios, and the position of the first facility. Stanislas [9] introduced to apply AI to floor plans analysis and generation. His ultimate goal is threefold: (1) to generate floor plans. (2) to qualify floor plans. (3) to allow users to browse through generated design options. He have chosen nested Generative Adversarial Neural Networks or GANs. Luisa et al. [10] introduced Firefly algorithm (FA), which was designed to solve continuous optimization problems. From the map of the layout that contains the (x, y) coordinates corresponding to the location of each of the nodes or workstations that are distributed on the plant, and paths between nodes are defined, the system solves layout problem as traveling salesman problem (TSP). Jing et al. [11] introduced the ant colony optimization (ACO) algorithm, which is a bio-inspired optimization algorithm based on the behavior of natural ants that succeed in finding the shortest paths from their nest to food sources by communicating via pheromone trails. Singh et al. [12] introduced a layout generating method using the space-filling curve. The space-filling curve [13] describes plane spaces based on the trajectory of an individual curve. The layout was created by allocating the required area based on the space-filling curve of the site. In these current layout design, the techniques and methods are not standardized.

B. Reinforcement learning

Reinforcement learning [14] is a machine learning method that implement optimal system control through trial and error by the system itself. Machine learning methods are generally classified into five types: supervised learning, unsupervised learning, self-supervised learning, semi-supervised learning, and reinforcement learning. While supervised, unsupervised self-supervised and semi-supervised learning require training data for learning, reinforcement learning obtain the data generated from its own experience. A reinforcement learning program developer creates a high learning efficiency model instead of preparing learning data. An agent in reinforcement learning learns the behavior to maximize the score set for the objective in an environment.

Reinforcement learning uses the following concepts.

- agent: subject of learning.
- environment: virtual space in which the agent acts.
- *state*: situation which the agent is located.
- action: what the agent does in a state.
- reward: value obtained by an action of the agent.
- return: sum of rewards that can be earned in the future.
- *policy*: guidelines for choosing actions.
- *episode*: the entire sequence of learning.
- trial: a learning step of an episode.
- *action-value function*: a function that represents the value of actions in the current state.

In reinforcement learning, rewards and returns are similar but different concepts. Reward is the invariant value given as a score for an action that causes a transition to the next state. Return, on the other hand, is the expected total value of the reward that will be finally obtained. In reinforcement learning, an agent is created that does not simply select an action with a large immediate reward but evolves to choose the action toward the one with the highest return. Fig. 1 shows a flow of reinforcement learning. The agent acts according to the policy in the environment. This action causes a transition from the current state to the next state. The reward is obtained based on the results of the action. The learning proceeds by updating the action-value function [15] and changing the state based on this reward. Returns is calculated by rewards and action-value function is calculated by returns. The agent continues the trial several times and creates an episode. The series of actions follow the Markov decision process [16]. The Markov decision process has the characteristic that the probability distribution of the future state depends only on the current state and not on all past states. This is called the Markov property.



Fig. 1. Flow of reinforcement learning.

There are three main learning methods for reinforcement learning: dynamic programming (DP) method [17], Monte Carlo (MC) method [18], and time difference (TD) method [19]. The DP method is a method of obtaining optimal behavior by solving the Bellman equation when each parameter of the system is known. The MC method was used to obtain the optimal behavior from the real reward obtained by repeating the trial. The TD method combines DP and MC methods to obtain the optimal behavior by solving the Bellman equation [20], [21] from the obtained reward. Currently, many learning methods have been developed based on these three methods. For example, DQN [22], SARSA [23], SAC [24] and Policy Gradients [25]. In particular, DQN using the TD method is currently a major part of reinforcement learning [14].

The Bellman equation is expressed as below:

$$V(S_t) = r(S_{t+1}) + \gamma V(s_{t+1})$$
(1)

- S_t : t-th states.
- $V(S_t)$: return in S_t ,
- $r(S_{t+1})$: reward obtained when transitioning to state S_{t+1}
- γ : discount factor

The return is expressed by the obtained reward in the current state and the return in the future state. This means the sum of the discounted rewards from all future states until the end of the trial. The discount factor γ is used to express the influence on the return and is set from 0 to 1. The closer the value is to 0, the less is the influence on the past states.

This Bellman equation is the basis of many reinforcement learning methods, and the above mentioned three methods can also be expressed using this equation.

Several studies have been conducted on reinforcement learning and space control, including layout problems. Xinhan et al. [26] introduced a method of searching for an appropriate arrangement while moving furniture in a room using a deep Q-Network (DQN) [27]. While this method can be arranged so as not to interfere with the function of the room, it can be arranged according to any policy. However it can only learn one piece of furniture at a time, which means that it is not possible to assume the arrangement of multiple units. Matthias et al. [28] study the layout for four functional units next to the logistic lane where the material is transported by vehicles using reinforcement learning. Their layout is generated to be optimized regarding a single planning objective, i.e., the transportation time. Richa et al [29] introduced reinforcement learning framework for selecting and sequencing containers to load onto ships in ports. The goal is to minimize the number of crane movements required to load a given ship. We can regard the problem as an assignment problem in which the order of assignments is important and therefore the reward is dependent of the order. Azalia et al [31] introduced a new graph placement method based on reinforcement learning, and demonstrate state-of-the-art results on chip floorplanning. They show that their method can generate chip floorplans that are comparable or superior to human experts in under six hours, whereas humans take months to produce acceptable floorplans for modern accelerators. Ruizhen et al. [30] introduced the transport-and-pack (TAP) problem, a frequently encountered instance of real-world packing. They developed a neural optimization solution based on reinforcement learning. Given an initial spatial configuration of boxes, they seek an efficient method to iteratively transport and pack the boxes compactly into a target container. Xinhan et al. [32] explore the interior graphics scenes design task as a Markov decision process, which is solved by deep reinforcement learning. Their goal is to generate an accurate layout for the furniture in the indoor graphics scenes simulation. Peter et al. [33] described the recent surge of research interest in Artificial Intelligence. They regard machine learning techniques including reinforcement learning as the most promising approach to facility layout research. López et al. [34] developed a virtual reality application that creates a production line for electric drills in a virtual space using deep reinforcement learning. Their PCG method can reduce the resources required for development and can personalize the 3-Dimentional virtual environments. Amine et al. [35] developed a Mixed Integer Programming (MIP) robust model for a form of Multi-floor layout problem. In the experiment, they created facility layouts reflecting the relationship between the cellar containing main storages and upper floors in which departments will be located in predetermined locations.

III. THE PROPOSED MECHANISM

We propose a layout-deployment mechanism using a reinforcement learning technique to deploy given units within a given site as a solution for FLP. An agent in our mechanism arranges the units on a flat site and proposes an efficient layout. This section explains the specific environment settings and algorithms of the proposed mechanism.

A. The mechanism specifications

Our mechanism assumes the following constraints.

- 1) Each functional space of the facility is modeled as a unit.
- 2) All of the units are represented as rectangles. In this study, a three-dimensional structure is not considered, that is, only one-floor layouts are considered.
- The condition of the site is managed as shown in Fig.
 Let the upper left block be (0, 0) and associate the







Fig. 3. An example site setting.

two-dimensional array with the block whose right and bottom directions are positive.

4) If the width and the depth of a unit are different, we should consider a 90-degree-rotated deployment.

B. Layout design mechanism

Site setting

As shown in Fig. 2, the top-left block of the site is the origin, the rightward direction is the x-axis plus direction, and the downward direction is the y-axis plus direction. The site is managed as a two-dimensional array of integers, which is assigned as zero when a unit is not deployed on it. When the agent deployed a unit, the array value of the corresponding location was updated to the number of units deployed. In the layout method, the agent can only create a layout that matches the site size initially set. In this mechanism, we set the maximum depth and width of the site, separately from the depth and width of the site used for layout. Since the states are determined based on remaining area in the maximum site, the agent can use the learning results for sites of other sizes. In Fig. 3, the entire grid is the maximum extent of the site, and the maximum area minus the gray area is the area used for the layout.

Deployment mechanism

Unlike Go or Shogi, it is difficult to determine the placement of units on a site because there are various sizes of units. To solve this problem, a starting point was introduced. The starting point is set according to the current conditions of the site, and the units are deployed based on the starting point. The agent searches the site and defines the next starting point as the point that forms the largest rectangle with no units

Remaining and maximum rectangular area



Fig. 4. Movement of the start point.

located. Fig. 4 shows an example of the starting point change. The first starting point was set to the origin. The dark blue area is already deployed, and the pale blue area is empty. In Fig. 4 the block at (0,2) is chosen as the next starting point.

Learning process

The problem to be dealt with in this study is the successful deployment of units within a given site. It may be difficult to completely evaluate the quality of unit deployment deployed in the middle of the trial. Therefore, the MC method, which allows learning to proceed by trying to achieve the end state and then evaluating each arrangement, is used in this study. The MC method requires longer time for learning but can generate a reliable solution. From the same reason, the MC method was also used in Mahjong [36], Chess [37], Shogi [38], and Go [39]. Since these game agents must evaluate solutions in tree structures, they use MC trees [40]. In our study, a table of action-value-function (Fig. 5) is created first. This function is called the action-value table and managed as a two-dimensional array, as shown in Fig. 5. One index of the array is state which is represented by the size and the maximum rectangular in the remaining area. Another index is action which is represented by the size of the unit that fits the site. All the values in the table were initialized to zero. The value in action-value-function is expressed as the average of the return in the state obtained in each trial.

One trial involved creating a layout for a set of units. The unit sets are randomly generated with the same number as the size of the site. The ϵ -greedy method [41], [42] was used to select the unit. The agent selects a random unit with a probability of ϵ (Exploration) and selects a unit with the highest action-value with a probability of 1 - ϵ (Exploration). The ϵ -greedy method can choose unlearned actions in exploration that will never choose in exploitation. This method is commonly used with reinforcement learning. When selecting a unit, the agents consider whether units with different vertical and horizontal lengths can be arranged as units with interchanged lengths.

Fig. 6 shows the calculation of return. When the mechanism finds that none of the units can be deployed, the mechanism finishes the current trial and regards the number of blocks located by units as the reward value of the trial (Reward N in Fig. 6). The score of the final state N is 0 (Score N in Fig. 6). Return was calculated using these values and the Bellman equation. The values of Score N-1 in Fig. 6 (return at state N)

1*1 1*2 1*3 1*4 5*4 ↑Unit 20 20 0 0 0 0 0 1*1 20 12.70 0 0 0 15.5 1*20 19 0 0 0 0 0 0 1*1 : : 0 0 0 0 0 1*1

Fig. 5. Action-value table.



Fig. 6. Calculation of learning value.

were calculated as the sum of the reward N and the product of the score N and discount factor γ . The values of Score K in Fig. 6 (return at state K) are calculated as the sum of the reward K+1 and the product of the score K+1 and the discount factor γ . Returns are updated by returning to the first state in the trial. According to the above process, the return value of a trial is calculated using the following formula:

$$Score_{k} = R_{k+1} + \gamma * Score_{k+1}$$
$$= \gamma^{N-k-1} * R_{N}$$
(2)

- N: Number of deployed units until final state.
- Score_k: the return in k-th state.
- R_N : the reward in k-th state.
- γ : discount factor.

Each value in the action-value table (Fig. 5) is calculated by the average of each return. The agent updates the action-value table for each trial and adjust the unit selection criteria.

learning mechanism

Fig. 7 shows the flow of one trial in an episode. First, the agent generates a set of units. Second, the agent selects a unit using ϵ -greedy method. Third, the agent deploys the unit based on the present starting point. Finally, the agent changes the staring point according to the site situation. The agent continues this procedure for as many units and calculates each value in the action-value table.

The agent learns the procedure to select the best from the set of units for the current site by repeating the trial.



Fig. 7. Flow of one trial in our mechanism.

IV. EXPERIMENT

This section represents the contents and results of the experiment. Two experiments were conducted to demonstrate the effectiveness and efficiency of the proposed mechanism. First, the proposed mechanism was applied to the three sets of units in the benchmark problems [12] to affirm whether the proposed mechanism can appropriately obtain a layout that satisfies these conditions. A one-floor plan is considered. Thereafter, we measure whether the learning will become faster by executing learning multiple times for each set of units to affirm the possibility to deep learning for the combinations of existing units. Learning was conducted according to the learning process described in Section III-B. The variables ϵ used in the ϵ -greedy method and the variables γ used in the learning value calculation are $\epsilon = 0.9$ and $\gamma = 0.9$, respectively.

A. Experiment 1 : Effectiveness of the proposed mechanism

In this experiment, the proposed mechanism was applied to the benchmark problems to demonstrate the effectiveness. The site has 7×5 blocks, with a total size of 35 blocks. Learning was performed for 50,000,000-unit sets, and each unit set randomly generated the same number of units as the site and the size that fits in the site for each trial. The study [12], which describes the benchmark problems, illustrates layouts that deploy units using a space-filling curve (Fig. 8). Because the proposed mechanism handles rectangular units, the L-shaped units that appear in the layout of the benchmark problem are handled by dividing them as appropriate.



Fig. 8. Layouts of the benchmark problem illustrated in [12].

The results of the generation are presented in Fig. 9. It was observed that the layout in which all the units on each floor fit in the site could be generated. The maximum change in learning value is shown in Fig. 10. These change of the values decrease with learning and converge to a constant value as the learning stabilizes. In Experiment 1, the maximum change decreased from 35 to approximately 25. The decrease of the graph started at approximately 20,000,000 times. The execution time was about 12 h for learning with the learning model for 50,000,000 times used this time, while the generation time was less than 0.1 s.

B. Experiment 2 : Changing the iteration count

In this experiment, the efficiency of repeated learning was examined for one set of units. It is ensured that the layout can be generated when the iteration counts of one set of units are changed. The agent is trained to repeat 100 times and 1,000 times for one set of units, and the layout is derived in the same way as in Experiment 1. Learning is performed for 500,000-unit sets were repeated 100 times, and for 50,000-unit sets were repeated 1,000 times. First, the case of continuous learning of one set of units is examined. Second, the case of repeated learning for all prepared sets of units is examined.

1) Experiment 2 Result 1: The generation result in the case of learning one set of units continuously is shown in Fig.

Fig. 9. Unit layout when learning 1 * 50,000,000 times.

10

6

6

9

g

4

5

8

6

11

8

8

3

3

12

12

6

15

16

3

8

4

2

5

15

35

30

25

20

15

10

5

0

ò

The maximum change



2*107

3*10

4*107

5*107

i*107

11. The results were derived using a layout that could be completely satisfied at all iterations. The maximum change in the action-value function is shown in Fig. 12. This graph is similar to the graph shown in Fig. 10.

2) Experiment 2 Result 2: The generation result in the case of repeating learning prepared sets of units is shown in Fig. 13. In this case, a layout that was completely satisfied was derived for all the repeated numbers. The maximum change

$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		_				_	_	_		_		_	
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	I	1	I	2	3	I	3		6		6	ſ	10
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	I	1	I	14	3	I	3		6	I	6	I	10
1 7 8 8 9 11 12 1 15 7 8 8 9 11 13 1 15 7 8 8 9 11 13 1 1 2 3 3 4 4 10 1 1 2 3 3 5 5 5 1 1 2 3 3 5 5 5 1 1 6 6 7 7 8 8 1 1 6 6 7 7 8 8 1 1 2 2 3 3 6 6 1 1 2 2 4 5 9 15 1 1 2 2 4 5 9 15 1 1 2 2 7 8 13 16 1 1 2 2 7 8 13 16 1<	I	1	I	4	5	I	5		6	L	6	I	12
1 15 7 8 8 9 11 13 1 1 2 3 3 4 4 10 1 1 2 3 3 5 5 5 1 1 2 3 3 5 5 5 1 1 2 3 3 5 5 5 1 1 6 6 7 7 8 8 1 1 6 6 7 7 9 9 1 1 2 2 3 3 6 6 1 1 2 2 4 5 9 9 1 1 2 2 4 5 9 15 1 1 2 2 7 8 13 16 1 1 2 2 7 8 13 16 1 1 1 1 1 1 1 1 1	I	1	I	7	8	I	8		9		11	I	12
1 1 2 3 3 4 4 10 1 1 2 3 3 5 5 5 1 1 2 3 3 5 5 5 1 1 2 3 3 5 5 5 1 1 6 6 7 7 8 8 1 1 6 6 7 7 9 9 1 1 2 2 3 3 6 6 1 1 2 2 4 5 9 9 1 1 2 2 4 5 9 15 1 1 2 2 4 5 9 15 1 1 2 2 7 8 13 16 1 1 1 1 1 1 1 1 1 1 1	I	15	1	7	8	I	8		9		11	I	13
1 1 2 3 3 4 4 10 1 1 2 3 3 5 5 5 1 1 2 3 3 5 5 5 1 1 2 3 3 5 5 5 1 1 2 3 3 5 5 5 1 1 6 6 7 7 8 8 1 1 6 6 7 7 9 9 1 1 2 2 3 3 6 6 1 1 2 2 4 5 9 15 1 1 2 2 4 5 9 15 1 1 2 2 7 8 13 16 1 1 1 1 1 1 1 1 1 1 1 1													
1 1 2 3 3 5 5 5 1 1 2 3 3 5 5 5 1 1 2 3 3 5 5 5 1 1 6 6 7 7 8 8 1 1 6 6 7 7 9 9 1 1 2 2 3 3 6 6 1 1 2 2 4 5 9 15 1 1 2 2 4 5 9 15 1 1 2 2 7 8 13 16 1 1 2 2 7 8 13 16 1 1 1 1 1 1 1 1 1		1		2	3		3		4		4		10
1 1 2 3 3 5 5 5 1 1 6 6 7 7 8 8 1 1 6 6 7 7 9 9 1 1 2 2 3 3 6 6 1 1 2 2 4 5 6 6 1 1 2 2 4 5 9 15 1 1 2 2 7 8 13 16 1 1 2 2 7 8 13 16 1 10 11 12 14 17 18	I	1	I	2	3	I	3		5		5	I	5
1 1 6 6 7 7 8 8 1 1 6 6 7 7 9 9 1 1 2 2 3 6 6 6 1 1 2 2 4 5 6 6 1 1 2 2 4 5 9 15 1 1 2 2 7 8 13 16 1 1 2 2 7 8 13 16 1 10 11 12 14 17 18	I	1	I	2	3	1	3		5		5		5
1 11 6 6 7 7 9 9 1 1 2 2 3 3 6 6 1 1 2 2 4 5 6 6 1 1 2 2 4 5 9 15 1 1 2 2 4 5 9 15 1 1 2 2 7 8 13 16 1 10 11 12 14 17 18	I	1	I	6	6	I	7		7		8	I	8
1 1 2 2 3 3 6 6 1 1 2 2 4 5 6 6 1 1 2 2 4 5 9 15 1 2 2 4 5 9 15 1 2 2 7 8 13 16 1 10 11 12 14 17 18	1	11		6	6	I	7		7	[9		9
1 1 2 2 3 3 6 6 1 1 2 2 4 5 6 6 1 1 2 2 4 5 9 15 1 1 2 2 4 5 9 15 1 1 2 2 7 8 13 16 1 10 11 12 14 17 18		_		_		-	_	-		-			
1 2 2 4 5 6 6 1 2 2 4 5 9 15 1 2 2 4 5 9 15 1 1 2 2 7 8 13 16 1 10 11 12 14 17 18	I	1	1	2	2	I	3		3		6	I	6
1 2 2 4 5 9 15 1 2 2 7 8 13 16 1 10 11 12 14 17 18	I	1	I	2	2		4		5	Ú	6	l	6
1 2 2 7 8 13 16 1 10 11 12 14 17 18	I	1	I	2	2	I	4		5	1	9		15
	I	1	I	2	2	I	7	I	8	[13		16
	I	1	I	10	11		12	1	14		17	I	18

Fig. 11. Unit layout when learning 100*500,000 times.

in the action-value function is shown in Fig. 14. This graph is similar to the graph illustrated in Fig. 10 and 12.



Fig. 12. Maximum change of action-value for 100* 500,000 learning.

V. DISCUSSION

A. Unit Deployment

According to Experiments 1 and 2, large units tend to be deployed earlier in the trial, and one-block-size units tend to be deployed later in the layout generated by the proposed mechanism (Fig. 9, Fig. 11, Fig. 13). The agent learns to



Fig. 13. Unit layout when learning 500,000*100 times.



Fig. 14. Maximum change of action-value for 500,000*100 learning.

get more rewards and changes the starting point that forms the largest rectangle in which no units are located. The agent regards the number of blocks located by units as the reward value of the trial. Therefore, the agent's priority is to try to deploy larger units. This tendency reflects the proposed mechanism. Additionally, proposed mechanism is superior to some methods because the mechanism can specify the shape of units. For example, in JLAV model [43] units are approximated by a circle to determine the positions. Each unit can specify

 TABLE I

 Fastest generation count and average.

	1*	2*	3*	10*
Avg. number	54587	94497.33	50315	54104
	20*	100*	200*	300*
Avg. number	120883.33	83282.5	66385.33	74313066
	400*	500*	1000*	*100
A 1	10(510,000	150(05.1((010(01)	1001/66
Avg. number	196/13.833	153687.166	217674	175466.66
Avg. number	*1000	153687.166	21/6/4	175466.66



The iteration count

Fig. 15. Correlation between the number of the iteration counts and the number of trials until the fastest layout generations. Horizontal axis represents the iterarion count of learnings and Vertical axis represents number of trials until the fastest generation.

the size but can not specify the shape. From these viewpoints, reinforcement learning is effective to the FLP.

B. Repeated learning effect

Although Experiment 2 obtained the result when the iteration count was changed, we did not observe the large differences in the maximum change of action-value. Therefore, an additional experiment was conducted to demonstrate the differences of the number of iteration counts. To measure the learning ability of the agent, we measured how many learning trials were repeated before the fastest fulfilled layout could be generated for the three sets of units. A graph of the average trial times is shown in Fig. 15. As shown in Fig. 15, an outlier is generated when repeated 300 times. However the other parts of the graph tended to increase monotonically. It is considered that such a result was obtained because it takes time to simply encounter an unknown combination of units as the number of iterations counts increases. As the number of iterations increases, the possibility that the unit common to the desired unit set does not appear in learning increases. Therefore, it is better not to repeat learning using the same set of units.

C. Degree of learning

We conduct 50,000,000-iteration learning to perform sufficient learning in Experiment2 1 and 2. The layouts can be

300

250

200

150

100

50

The number of fulfilled layout

Fig. 16. Total amount of changes of 50,000,000-iteration learning.

generated by at most 2,000,000-iteration learning in Table I. According to Fig. 10, Fig. 12, and Fig. 14, the maximum change in the action-value decreased a few numbers from 35. Because the maximum change in the action-value is taken from all the changes in the action-value table, large changes are frequently observed in the graphs. Therefore, to avoid the effect of large changes, the total number of changes was graphed. Fig 16 shows a graph of the total amount of change of 50,000,000-iteration learning. According to Fig. 16, the number of changes is decreased from 200 and converged to around 25 when the agent learned about 10,000,000 times. The investigation shows that the approximate degree of learning can be measured by observing the total number of changes.

D. Unit order

Layouts were only generated with the same order sets of units in Experiment2 1 and 2. Therefore, an additional experiment was conducted to examine whether the agent can generate layouts when the order of the set of units is changed. Hundred different orders of units were prepared, and the agents were made to generate their layouts at each time step of learning. The experimental results are presented in Fig. 17. 50,000,000-iteration learning was conducted and check the layout of each order of sets of units for every 1000 learnings finished. The figure shows the total number of fulfilled layouts in three floors. From the figure, we could demonstrate that the learning with the proposed mechanism gradually increase the possibility of successful layout generation.

VI. OUTLOOK

The problems and functions that we would like to implement in future studies are described below.

Enlarged table

The action-value table is created by the number of blocks that has no unit and the units that can be deployed at the site as the index. Therefore, the size of the table is the product of the number of each index. Because the number of units increases in response to the site area, the size of the table increases exponentially. Therefore, if the site area becomes too large, it may not be possible to secure the memory area. The study by Seongwoo [44] describes the efficiency of memory in deep Q-learning, and it is mentioned as one solution to this problem. Therefore, we will have to use deep Q-learning to tackle the problem of not only reducing calculation time, but also memory shortage.

Extension in three-dimensional direction

10000

Most studies on layout creation are for flat sites, similar to the proposed mechanism. However, many facilities have a three-dimensional structure to make functions and activities efficient. This is a common practice in the areas with high land cost or concerns of the available land [45]. In the future, we would like to expand on the creation of a three-dimensional site layout. Helbor et al. [46] and Hahn et al. [47] studied the layout of the facility over multiple layers and stated that it is required to multi-floor layout the functions and activities of the facility at a lower cost.

Correlation between units

The proposed mechanism creates a layout that successfully deploys the given units within a given site. In actual facilities, each unit has a function and meaning, and each unit must be adjacent or should not be close. Therefore, we would like to set the relationship between the units and modify that the agent can learn that the units can be deployed close to each other. Yifei et al. [48], [49] completed the warehouse layout based on the current situation of comprehensive related relationships and warehouse problems using the SLP method. When the SLP method is used to analyze the non-logistics relationship between the operation units, it depends on several evaluation indicators. In the process of determining the degree of correlation, the subjective idea of the designer must be



Fig. 17. Number of fulfilled layout using 100 orders of units.

The iteration count devided by 1000

20000

30000

40000

50000

mixed. Therefore, introducing correlation between units is a very time-consuming modification.

Accelerate learning

In the learning of this mechanism, it took approximately 12 h to learn 50,000,000 times. This result is evidently slow. For example, deep reinforcement learning, which uses a neural network to represent a table of learning values as an approximate function and performs learning, can reduce the time required for table reference and creation. Simultaneously, it is possible to speed up and reduce the load on the memory area of the learning table. Zhiang et al. [50] proposed a whole-building-energy-model-based deep reinforcement learning control framework. The building energy model (BEM), which is a detailed physics-based model used to predict a building's thermal and energy behaviors, has been widely used for building design decision support. Therefore, a deep neural network is expected to contribute to higher speeds.

Utilization of learned agents

To use the learned agent for generating the layout in other size sites, the maximum deployable site is set, and the remaining site is calculated from it. Therefore, its effectiveness was not affirmed in the experiments. If the same agent can generate layouts at different sites, the time and cost can be significantly reduced. Felipe et al. [51], [52] stated that learning a task from scratch every time is impractical because of the huge sample complexity of reinforcement learning algorithms.

VII. CONCLUSION

A mechanism that generates a facility layout using reinforcement learning was proposed. The proposed mechanism generates a layout plan that can successfully deploy the given units within a given site. Based on the results, reinforcement learning was considered to be effective to FLP. In addition, training the agent by providing additional information on the maximum rectangle in which no units are located, the agent can generate the layout for a set of various sets of units, and the effectiveness of learning when the iteration counts are changed was examined. Consequently, it was discovered that the agent tends to learn faster by providing new learning data for each trial without repetition. However, some situations could not be tested in the experiment. First, we could not examine whether the agent could learn and generate layouts at the multi-floor site. Second, we could not examine the layout generation with interrelationship among the units. The problem with generating the layout at the multi-floor site is one of the most important points to be examined for efficiency in the layout of warehouses and distribution facilities in the real world. In addition, speeding up learning using multi-agent [53], [54] or parallelization [55], [56], and comparing the MC method with other reinforcement learning methods, such as Q-learning, are future challenges. We would like to focus on these problems in future studies.

VIII. ACKNOWLEDGEMENTS

Kajima Corporation supported this work. We would like to thank Hiromasa Akagi and Fumi Sekimoto, who work with Kajima Corporation, for helpful discussions. We gratefully thank three anonymous referees for their constructive comments that helped improve the paper.

REFERENCES

- Andrew Kusiak, Sunderesh S.Heragu, 1987, The facility layout problem, European Journal of Operational Research 29, 229-251, https://doi.org/ 10.1016/0377-2217(87)90238-4
- [2] Sunderesh S.Heragu, AndrewKusiak, 1991, Efficient models for the facility layout problem, European Journal of Operational Research, https://doi.org/10.1016/0377-2217(91)90088-D
- [3] S.P.Singh, R.R.K.Sharma, 2006, A review of different approaches to the facility layout problems, The International Journal of Advanced Manufacturing Technology volume 30, pages 425–433, https://doi.org/ 10.1007/s00170-005-0087-9
- [4] Kar Yan Tam, 1992, Genetic algorithms, function optimization, and facility layout design, European Journal of Operational Research Volume 63 issue 2, https://doi.org/10.1016/0377-2217(92)90034-7
- [5] Anita Thengade, Rucha Dondal, 2012, Genetic Algorithm Survey Paper, MPGI National Multi Conference 2012, ISSN: 0975 - 8887
- [6] Pedro G. Espejo, Sebastian Ventura, Francisco Herrera, 2010, A Survey on the Application of Genetic Programming to Classification, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews, Volume 40, Issue 2, 10.1109/TSMCC.2009.2033566
- [7] Venkatesh Dixit, Jim Lawlor, 2019, Modified genetic algorithm for automated facility layout design, International Journal of Advance Research, Ideas and Innovations in Technology, Volume 5, Issue 3, ISSN: 2454-132X
- [8] José Fernando Gonçalvesa, Mauricio G.C.Resende, 2015, A biased random-key genetic algorithm for the unequal area facility layout problem, European Journal of Operational Research, Volume 246, https://doi.org/10.1016/j.ejor.2015.04.029
- [9] Stanislas Chaillou, 2019, AI and Architecture An Experimental Perspective, The Routledge Companion to Artificial Intelligence in Architecture, ISBN:9780367824259
- [10] Luisa Fernanda Vargas-Pardo, Frank Nixon Giraldo-Ramos, 2021, Firefly algorithm for facility layout problemoptimization, Visión electrónica, https://doi.org/10.14483/issn.2248-4728
- [11] Jing fa, Liuab JunLiu, 2019, Applying multi-objective ant colony optimization algorithm for solving the unequal area facility layout problems, Applied Soft Computing, Volume 74, https://doi.org/10.1016/ j.asoc.2018.10.012
- [12] Russell D.Meller, Yavuz A.Bozer, 1997, Alternative Approaches to Solve the Multi-Floor Facility Layout Problem, Journal of Manufacturing Systems, Volume 16, Issue 3, https://doi.org/10.1016/S0278-6125(97)88887-5,
- [13] Arthur R.Butz, 1969, Convergence with Hilbert's Space Filling Curve, Journal of Computer and System Sciences, https://doi.org/10.1016/ S0022-0000(69)80010-3
- [14] L.P.Kaelbling, M.L.Littman, A.W.Moore, 1996, Reinforcement Learning: A Survey, JAIR, https://doi.org/10.1613/jair.301
- [15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller, 2013, Playing Atari with Deep Reinforcement Learning, NIPS Deep Learning Workshop, https://doi.org/10.48550/arXiv.1312.5602
- [16] Yu-Jui Liu, Shin-Ming Cheng, Yu-Lin Hsueh, 2017, eNB Selection for Machine Type Communications Using Reinforcement Learning Based Markov Decision Process, /url10.1109/TVT.2017.2730230
- [17] Frank L. Lewis, Draguna Vrabie, 2009, Reinforcement learning and adaptive dynamic programming for feedback control, IEEE Circuits and Systems Magazine Volume9 Issue3, 10.1109/MCAS.2009.933854
- [18] F. Llorente, L. Martino, J. Read, D. Delgado, 2021, A survey of Monte Carlo methods for noisy and costly densities with application to reinforcement learning, https://doi.org/10.48550/arXiv.2108.00490
- [19] P. Cichosz, 1995, Truncating Temporal Differences: On the Efficient Implementation of TD(lambda) for Reinforcement Learning, 2017, JournalofArti cialIntelligenceResearch2, IEEE Transactions on Vehicular Technology, Volume 66, No. 12, https://doi.org/10.1613/jair.135

- [20] Harmon, Mance E., Harmon, Stephanie S., 1997, Reinforcement Learning: A Tutorial., January,
- [21] E. N. Barron, H. Ishii, 1989, The Bellman equation for minimizing the maximum cost, Nonlinear Analysis, Theory, Methods and Applocations, https://doi.org/10.1016/0362-546X(89)90096-5
- [22] Christopher J. C. H. Watkins, Peter Dayan, 1992, Q-Learning, https: //doi.org/10.1007/BF00992698 Machine Learning, 8, 279-292
- [23] Ali Asghari, Mohammad Karim Sohrabi, Farzin Yaghmaee, 2021, Task scheduling, resource provisioning, and load balancing on scientifc workfows using parallel SARSA reinforcement learning agents and genetic algorithm, The Journal of Supercomputing, https://doi.org/10. 1007/s11227-020-03364-1
- [24] Feng Ding, Guanfeng Ma, Zhikui Chen, Jing Gao, Peng Li, 2021, Averaged Soft Actor-Critic for Deep Reinforcement Learning, Complexity, vol.2021, https://doi.org/10.1155/2021/6658724
- [25] Seyed Sajad Mousavi, Michael Schukat1, Enda Howley, 2017, Traffic light control using deep policy-gradient and value-functionbasedreinforcement learning, IET Intelligent Transport Systems, https: //doi.org/10.1049/iet-its.2017.0153
- [26] Xinhan Di, Pengqian Yu, IHome Company, IBM Research, 2021, Deep Reinforcement Learning for Producing Furniture Layout in Indoor Scenes, Cornell University, https://doi.org/10.48550/arXiv.2101.07462
- [27] Vincent Francois-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, Joelle Pineau, 2018, An Introduction to Deep Reinforcement Learning, Foundations and Trends in Machine Learning, Volume 11, https://doi.org/10.1561/2200000071
- [28] Matthias Klar, Moritz Glatt, Jan C. Aurich, 2021, An implementation of a reinforcement learning based algorithm for factory layout planning, Manufacturing Letters, Volume 30, October, https://doi.org/10.1016/j. mfglet.2021.08.,
- [29] Richa Verma, Sarmimala Saikia, Harshad Khadilkar, Puneet Agarwal, Gautam Shrof, Ashwin Srinivasan, 2019, A Reinforcement Learning Framework for Container Selection and Ship Load Sequencing in Ports, Autonomous Agents and Multiagent Systems,
- [30] Ruizhen Hu, Juzhan Xu, Bin Chen, Minglun Gong, Hao Zhang, Hui Huang, 2020, TAP-Net: Transport-and-Pack using Reinforcement Learning, ACM Transactions on Graphics, Volume 39, December, https: //doi.org/10.1145/3414685.3417796
- [31] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter, Jeff Dean, 2021, A graph placement methodology for fast chip design, Nature, volume 594, pages207–212, https://doi.org/10.1038/ s41586-021-03544-w
- [32] Xinhan Di, Pengqian Yu, 2021, Multi-Agent Reinforcement Learning of 3D Furniture Layout Simulation in Indoor Graphics Scenes, ICLR SimDL Workshop, https://doi.org/10.48550/arXiv.2102.09137
- [33] Peter Burggraf, Johannes Wagner, Benjamin Heinbach, 2021, Bibliometric Study on the Use of Machine Learning as Resolution Technique for Facility Layout Problems, IEEE Access, Volume 9, 10.1109/ACCESS. 2021.3054563
- [34] Christian E. López, James Cunningham, Omar Ashour, Conrad S. Tucker, 2020, Deep Reinforcement Learning for Procedural Content Generation of 3D Virtual Environments, Journal of Computing and Information Science in Engineering, https://doi.org/10.1115/1.4046293
- [35] Niloufar Izadinia, Kourosh Eshghi, Mohammad Hassan Salmani, A robust model for multi-floor layout problem, 2014, Computers and Industrial Engineering 78, http://dx.doi.org/10.1016/j.cie.2014.09.023
- [36] Junjie Li, Sotetsu Koyamada, Qiwei Ye, Guoqing Liu, Chao Wang, Ruihan Yang, Li Zhao, Tao Qin, Tie-Yan Liu, Hsiao-Wuen Hon, 2020, Suphx: Mastering Mahjong with Deep Reinforcement Learning, Cornell University, https://doi.org/10.48550/arXiv.2003.13590
- [37] Matthew Lai, 2015, Giraffe: Using Deep Reinforcement Learning to Play Chess, partial fulfilment of the requirements for the MSc Degree in Advanced Computing of Imperial College, https://doi.org/10.48550/ arXiv.1509.01549
- [38] Adrian Goldwaser, Michael Thielscher, 2020, Deep Reinforcement Learning for General Game Playing, The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20), https://doi.org/10.1609/aaai. v34i02.5533

- [39] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis, 2018, A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, Science, 10.1126/science. aar6404,
- [40] Guillaume Chaslot, Sander Bakkes, Istvan Szita, Pieter Spronck, 2008, Monte-Carlo Tree Search: A New Framework for Game AI, Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, https://ojs.aaai.org/index.php/AIIDE/article/view/18700
- [41] Yahui Liu, Buyang Cao, Hehua Li, Improving ant colony optimization algorithm with epsilon greedy and Levy flight, 2021, Complex and Intelligent Systems 17111722, https://doi.org/10.1007/s40747-020-00138-3
- [42] Tailong Yang, Shuyan Zhang, Cuixia Li, 2021, A multi-objective hyperheuristic algorithm based on adaptive epsilon-greedy selection, Complex and Intelligent Systems, https://doi.org/10.1007/s40747-020-00230-8
- [43] Abbas Ahmadi, Mohammad Reza Akbari Jokar, 2016, An efficient multiple-stage mathematical programming method for advanced single and multi-floor facility layout problems, Applied Mathematical Modelling, Volume 40, Issues 9–10, Pages 5605-5620, https://doi.org/10. 1016/j.apm.2016.01.014
- [44] Seongwoo Lee, Joonho Seon, Chanuk Kyeong, Soohyun Kim, Youngghyu Sun, Jinyoung Kim, 2021, Novel Energy Trading System Based on Deep-Reinforcement Learning in Microgrids, https://doi.org/10.3390/ en14175515,
- [45] Amine Drira, Henri Pierreval, SoniaHajri-Gabouj, 2007, Facility layout problems: A survey, Annual Reviews in Control, Volume 31, Issue 2, https://doi.org/10.1016/j.arcontrol.2007.04.001
- [46] Stefan Helber, Daniel Bohme, Farid Oucherif, Svenja Lagershausen, Steffen Kasper, 2015, A hierarchical facility layout planning approach for large and complex hospitals, Flexible Services and Manufacturing Journal, pp 5-29, https://doi.org/10.1007/s10696-015-9214-6
- [47] Peter Hahn, J.MacGregor Smith, Yi-Rong Zhu, 2008, The Multi-Story Space Assignment Problem, Annals of Operations Research,pp77-103, https://doi.org/10.1007/s10479-008-0474-3
- [48] Yifei Zhang, 2021, The design of the warehouse layout based on the non-logistics analysis of SLP, E3S Web of Conferences 253, https://doi. org/10.1051/e3sconf/202125303035
- [49] Yifei Zhang, 2020, Research on layout planning of disinfection tableware distribution center based on SLP method, MATEC Web of Conferences 325, https://doi.org/10.1051/matecconf/202032503004
- [50] Zhiang Zhang, Adrian Chongb, Yuqi Panc, Chenlu Zhanga, Khee Poh Lam, 2019, Whole building energy model for HVAC optimal control: A practical framework based on deep reinforcement learning, Energy and Buildings, Volume 199, Pages 472-490, https://doi.org/10.1016/j. enbuild.2019.07.029
- [51] Felipe Leno Da Silva, Anna Helena Reali Costa, 2019, A Survey on Transfer Learning for MultiagentReinforcement Learning Systems, Journal of Artificial Intelligence Research 64, https://doi.org/10.1613/ jair.1.11396
- [52] Felipe Leno Da Silva, Matthew E. Taylor, Anna Helena Reali Costa, 2018, Autonomously Reusing Knowledge in Multiagent Reinforcement Learning, Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18),
- [53] Wei Du, Shifei Ding, 2020, A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications, Artificial Intelligence Review, https://doi.org/10.1007/s10462-020-09938-y
- [54] Ingy Elsayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, Lu Feng, 2021, Safe Multi-Agent Reinforcement Learning via Shielding, Autonomous Agents and Multiagent Systems, https://doi. org/10.48550/arXiv.2101.11196
- [55] Alfredo V. Clemente, Humberto N. Castejon, Arjun Chandra, 2017, EFFICIENT PARALLEL METHODS FOR DEEP REINFORCEMENT LEARNING, Cornell University, https://doi.org/10.48550/arXiv.1705. 04862
- [56] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas, Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, Shane Legg, Volodymyr Mnih, Koray Kavukcuoglu, David Silver, 2015, Massively Parallel Methods for Deep Reinforcement Learning, the Deep Learning Workshop, International Conference on Machine Learning, https://doi.org/10.48550/arXiv. 1507.04296