2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Simultaneous Planning for Item Picking and Placing by Deep Reinforcement Learning

Tatsuya Tanaka[1], Toshimitsu Kaneko[1], Masahiro Sekine[1], Voot Tangkaratt[2] and Masashi Sugiyama[2,3]

Abstract— **Container loading by a picking robot is an important challenge in the logistics industry. When designing such a robotic system, item picking and placing have been planned individually thus far. However, since the condition of picking an item affects the possible candidates for placing, it is preferable to plan picking and placing simultaneously. In this paper, we propose a deep reinforcement learning (DRL) method for simultaneously planning item picking and placing. A technical challenge in the simultaneous planning is its scalability: even for a practical container size, DRL can be computationally intractable due to large action spaces. To overcome the intractability, we adopt a fully convolutional network for policy approximation and determine the action based only on local information. This enables us to produce a shared policy which can be applied to larger action spaces than the one used for training. We experimentally demonstrate that our method can successfully solve the simultaneous planning problem and achieve a higher occupancy rate than conventional methods.**

I. Introduction

Due to labor shortages, there has been an increasing demand for automation of a container loading problem by using picking robots in the logistics industry. For efficient transportation, it is also required to optimize the occupancy rate of a container. A conventional container loading problem [1] focuses on only how to load a container and an item picking is separately solved. However, in order to optimize the occupancy rate, those operations should be planned simultaneously, since the picking position of an item affects the region in which the item can be placed.

Thus, we design a new container loading problem, simultaneous planning for item picking and placing. The goal of the task is to maximize the occupancy of the container by planing where to pick an item and placing it in the container. To achieve this goal, in this paper we formulate the task by representing the container as a discretized grid space.

In view of planning robots, much research has attempted to use reinforcement learning (RL) to learn a planner [2], [3], [4]. However, it is difficult to apply deep RL (DRL) to our task. This is because actions are combinations of the picking position and placing position, hence the action space becomes too large even for a moderate container size.

In this paper, we propose a method for solving our task by using DRL with the fully convolutional network (FCN) [5]. Our method is based on the property of the task that each action can be approximately determined using only local information. In order to realize this, we apply FCN architecture for the policy. The policy is therefore applicable to larger action space than the one used in the training phase.

The main contributions are summarized as follows. First, we design simultaneous planning for item picking and placing, which brings beneficial effects for optimizing the occupancy rate. Second, we propose a DRL method with FCN for the task. Our method enables us to produce a shared policy which can be applied to action spaces larger than the one used in the training phase. Third, we experimentally demonstrate that our method can learn simultaneous planning and achieve a higher occupancy rate than conventional methods.

II. Related Work

The container loading problem is a classical optimization problem in logistics, and methods such as exact methods and heuristic methods have been proposed [1]. Compared to heuristic methods, the number of exact methods is limited because of the difficulty of representing possible patterns and practical packing constraints [6]. The deepest bottom left method is well known as a heuristic method for the container loading problem. Duan et al. [7] defined the container loading problem as a combinatorial optimization problem with a heuristic method that determines the order of loading items. Since these methods consider only the item placing operation, they cannot plan the picking and loading operations simultaneously. In order to optimize the occupancy rate, those operations should be planned simultaneously.

To overcome the above limitation, methods based on RL have been proposed recent years. Laterre et al. [8] proposed a reward design called the ranked reward using the framework of self-play, and applied it to the container loading problem. The expected reward is maximized by a Monte-Carlo tree search (MCTS) [9] based algorithm which decides the order and placing position of the objects. However, in the MCTS based method, since the computational time is greatly influenced by the size of the action space, there are concerns that the tree search may become difficult when planning picking and loading operations simultaneously. Kundu et al. [10] designed a 2D online container loading problem and proposed a method to solve it by using DRL. Since the method also focuses only on how to load a container, it is intractable to learn the simultaneous planning due to the large action space.

RL using FCN was proposed by Woo et al. [11]. In their method, the Q-function in deep Q-network [12] is

[1]Corporate Research & Development Center, Toshiba Corporation, Japan {tatsuya5.tanaka, toshimitsu.kaneko, masahiro5.sekine}@toshiba.co.jp
[2]Center for Advanced Intelligence Project, RIKEN, Japan
[3]The University of Tokyo, Japan

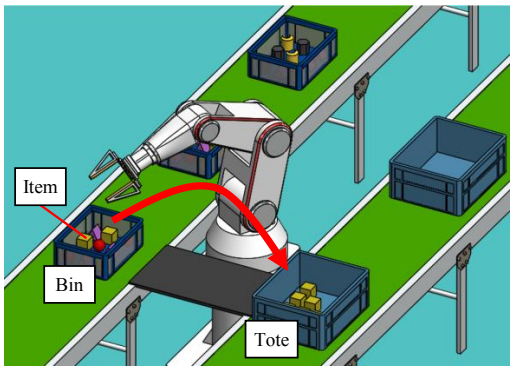


Fig. 1: Simultaneous planning for item picking and placing.

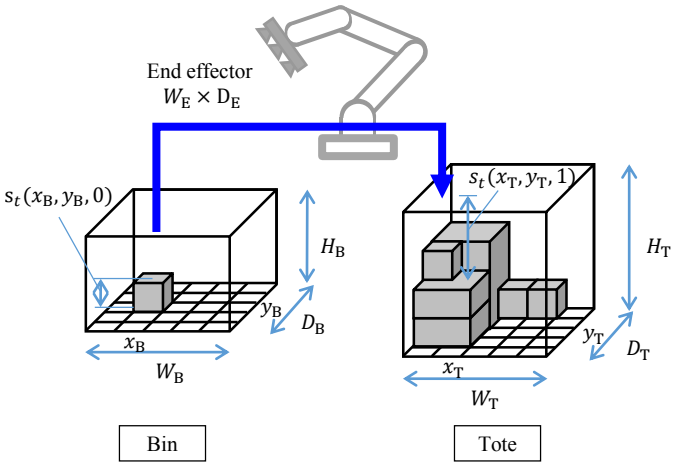


Fig. 2: Definition of variables for the item picking and placing problem.

represented by FCN. The aim is to improve the learning performance by using spatial continuity, and the applicability to other sizes of action space has not been discussed.

III. Preliminaries

In this section, we formalize our target problem.

A. Simultaneous Planning for Item Picking and Placing

In this subsection, we describe the problem of simultaneous planning for item picking and placing (Fig.1). First we define the terms for this task as follows.

- Item: A target object which is packed into the container.
- Bin: A container which carries items from a warehouse.
- Tote: A container which items are placed in.

In this task, we assume that a picking robot picks an item stored in a bin carried from a warehouse by a belt conveyer, and stores it in a tote. To realize automation of this task, a planner of the robot needs to repeatedly plan where to pick an item and place it in the tote. The objective is to maximize the occupancy rate of the tote for efficient transportation and it is also required that the picked item and end effector do not collide with the bin, tote, or items already placed in a tote. In this paper, we assume a vacuum-type end effector, which has been widely used in industry for picking robots [13] and do not consider the reposition of a picked item using working space.

The detailed settings are listed below.

- The surface area of the end effector is a rectangle of size $W_{\mathrm{E}} \times D_{\mathrm{E}}$ and the end effector can pick (adsorb) items in arbitrary positions in this rectangle.
- The robot arm picks an item from a bin and then places it in the tote. This operation continues until the task is finished.
- When the picking and placing operations are completed, the target bin changes to the next bin.
- The shapes of items are restricted to cuboids.
- Each bin has an item, and the item is randomly positioned in the bin.
- The termination conditions of this task are the following.
  - The position of the upper surface of a placed item exceeds that of the tote.
  - The robot arm collides with the bin, tote, or an already placed item.
  - An item is placed in an unstable position (the base area is less than 50% of the item).
- The order of the items cannot be changed.
- The end effector can rotate around the z-axis (i.e., the horizontal plane).

B. Problem Settings

Next we define the RL settings for the task described above. The state s_t at time t consists of depth maps of the bin and tote at time t. The width and depth of the bin are divided into a grid size $W_{\mathrm{B}} \times D_{\mathrm{B}}$, and the depth of the grid position $(x_{\mathrm{B}}, y_{\mathrm{B}})$ at time t is written as $s_t(x_{\mathrm{B}}, y_{\mathrm{B}}, 0)$. The depth of the bin is represented by the height from the bottom of the bin. Similarly, the width and depth of the tote are divided into a grid size $W_{\mathrm{T}} \times D_{\mathrm{T}}$ and the depth of the grid position $(x_{\mathrm{T}}, y_{\mathrm{T}})$ at time t is written as $s_t(x_{\mathrm{T}}, y_{\mathrm{T}}, 1)$. Unlike the bin depth, the tote depth is represented by the remaining height until the tote is full (Fig. 2).

The actions in this task are the picking position in the bin and the placing position and rotation in the tote. We define an action a_t as a discrete variable $0 \leq a_t < N_{\mathrm{a}}$, where $N_{\mathrm{a}}{=}W_{\mathrm{B}} \times D_{\mathrm{B}} \times W_{\mathrm{T}} \times D_{\mathrm{T}} \times R$. Here, R is the possible rotations of the picked object.

The reward is v_t if both of the picking and placing operations are succeeded and 0 otherwise, where v_t is the volume of the item placed into the tote at time t.

The settings of this task are summarized below.

- State $s_t(x, y, b)$: The depth of bin $(b = 0)$ and tote $(b = 1)$.
- Action a_t: An integer corresponding to the picking position, placing position, and rotation.
- Reward r_t: The volume of the item placed successfully (0 if fail).

C. Deep Reinforcement Learning

We use proximal policy optimization (PPO) [14] for solving our task. However, other methods such as advantage

actor critic (A2C) [15] and trust region policy optimization (TRPO) [16] can also be used.

PPO is an actor-critic method, and consists of a policy and a value function. We denote the parameters of these networks as θ_{P} and θ_{V}, respectively. The typical loss function of the actor-critic method is described by the following equation.

$$L_\theta = \mathbb{E}_t[L_t^{\mathrm{P}}(\theta_{\mathrm{P}}) - c_1 L_t^{\mathrm{V}}(\theta_{\mathrm{V}}) + c_2 H(\pi(s_t;\theta_{\mathrm{P}}))], \quad (1)$$

where $L_t^{\mathrm{P}}(\theta_{\mathrm{P}})$ is the loss function for policy gradient [17]: $L_t^{\mathrm{P}}(\theta_{\mathrm{P}}) = \log \pi_{\theta_{\mathrm{P}}}(a_t|s_t)\hat{A}_t$, $\pi_{\theta_{\mathrm{P}}}$ is a policy, $\hat{A}_t$ is an estimator of the advantage function at time step t, $L_t^{\mathrm{V}}(\theta_{\mathrm{V}})$ is a squared-error loss: $L_t^{\mathrm{V}}(\theta_{\mathrm{V}}) = (V_{\theta_{\mathrm{V}}}(s_t) - V_t^{\mathrm{target}})^2$, H is an entropy bonus, and c_1, c_2 are coefficients. The clipped surrogate objective is used for PPO instead of $L_t^{\mathrm{P}}(\theta_{\mathrm{P}})$. For more details, we refer the readers to [14].

IV. Method

If we use a typical RL method to learn the simultaneous planning, it is intractable to learn even for a practical container size due to the large action space. To overcome this intractability, we apply a FCN architecture for a policy and determine the action based on only local information.

A. Policy Sharing between Different Action Spaces

In general, a network consisting of convolutional layers followed by fully connected layers is commonly used for pixel-based state variables [12]. However, since the output size of the final fully connected layer depends on the size of the action space, such a network cannot be directly employed in the current setting. To apply the trained policy to larger action space than the one used in the training phase, we use FCN [5] for approximating the policy. We used 4D convolution for the convolutional layer of the FCN in order to evaluate the value of an action based on the local information of both the bin and tote. The input state $s_t(x, y, b)$ is reshaped and broadcast to the 4D tensor, and concatenated to $s_t(x_{\mathrm{B}}, y_{\mathrm{B}}, x_{\mathrm{T}}, y_{\mathrm{T}}, b)$. Next, $s_t(x_{\mathrm{B}}, y_{\mathrm{B}}, x_{\mathrm{T}}, y_{\mathrm{T}}, b)$ is converted to $\tilde{s}_t(x_{\mathrm{B}}, y_{\mathrm{B}}, x_{\mathrm{T}}, y_{\mathrm{T}}, b)$ and given to the network. The details of the conversion are described in SectionIV-B.

Our method is based on the property of the task that the action can be approximately determined using only local information. Specifically, when evaluating the value of placing an item in a certain position $(x_{\mathrm{T}}, y_{\mathrm{T}})$ in the tote, the state of the neighbor region is thought to have a great influence on the value, whereas regions far from $(x_{\mathrm{T}}, y_{\mathrm{T}})$ are thought to have a small influence. The picking position is also considered to be the same. Therefore, the value of the picking position and placing position/direction can be approximately evaluated from only local information of the state of sufficient size for an item. To realize such an evaluation, we use an FCN architecture to select an action based on the local information on whether it is large enough for the size of the item.

To evaluate the value of the picking/placing position based on local information, it is necessary to set up a receptive field wide enough when the item size and rotation of the end effector are taken into account. Moreover, for applying the trained policy to a setting with larger bin/tote size than the training setting, the bin/tote size for training should be larger than the size of the receptive field. Therefore, the size of the receptive field should satisfy the following conditions.

$$\text{maximum item size} \leq \frac{\text{receptive field size}}{2} + 1, \quad (2)$$

$$\text{receptive field size} \leq \text{bin/tote size}. \quad (3)$$

Fig. 3 shows the network architecture for the policy/value function and parameters that were used in the experiments in Section V-B. In the experiment, we used 4 directions for R which gives directions of $0°$, $90°$, $180°$, and $270°$. The number of channels in the last layer is then $4(= R)$.

The input to the network is padded according to the receptive field size, sucn that the dimension of the output becomes $(W_{\mathrm{B}}, D_{\mathrm{B}}, W_{\mathrm{T}}, D_{\mathrm{T}})$. Moreover, the padding value of bin depth is a negative constant to clearly indicate the outside regin of the bin. The padding value of tote depth is 0 because the outsize of tote can be treated as a fully occupied region.

Although a fully connected layer is used for the value function $V_{\theta_{\mathrm{V}}}(s_t)$ needed for learning as a critic, the calculation of $V_{\theta_{\mathrm{V}}}(s_t)$ is unnecessary for selecting the action. Thus, this has no effect on the application of the policy to different sizes of action spaces.

B. Policy Sharing between Different Depth Value Ranges

Because the depth value of a state depends on the height of the bin/tote, the variation in states increases as the height of the bin/tote increases. To apply the policy to any height of bin/tote, it is necessary to specify the maximum height to cover all depth patterns. However, this increases the training cost and makes training more difficult.

In order to solve this problem, we compute $\pi_{\theta_{\mathrm{P}}}(a|s)$ by considering only a fixed depth range of the bin/tote (Fig. 4). Specifically, the depth value of the tote state is clipped by the following equation.

$$\tilde{s}_t(x_{\mathrm{B}}, y_{\mathrm{B}}, x_{\mathrm{T}}, y_{\mathrm{T}}, b) = s_t(x_{\mathrm{B}}, y_{\mathrm{B}}, x_{\mathrm{T}}, y_{\mathrm{T}}, b) - \max(\max_{x,y} s_t(x, y, b) - H_b^{\max}, 0), \quad (4)$$

where $H_0^{\max}$ and $H_1^{\max}$ are the ranges which are considered during evaluation of estimating the picking/placing position.

This clipping processing makes it possible to compute $\pi_{\theta_{\mathrm{P}}}(a|s)$ by considering only a fixed depth range of the bin/tote.

V. Experiments

In our experiments, we used curriculum learning [18] for training. In curriculum learning, the curriculum begins from a shallow tote, and the height is increased gradually. It means that the state starts from the condition where the tote was almost filled at an early stage of the curriculum. We compared the occupancy rate achieved by the proposed method with other methods. The occupancy rate was evaluated from the average over 100 evaluations. The hyper-parameters used in our experiments are described in Table.I.

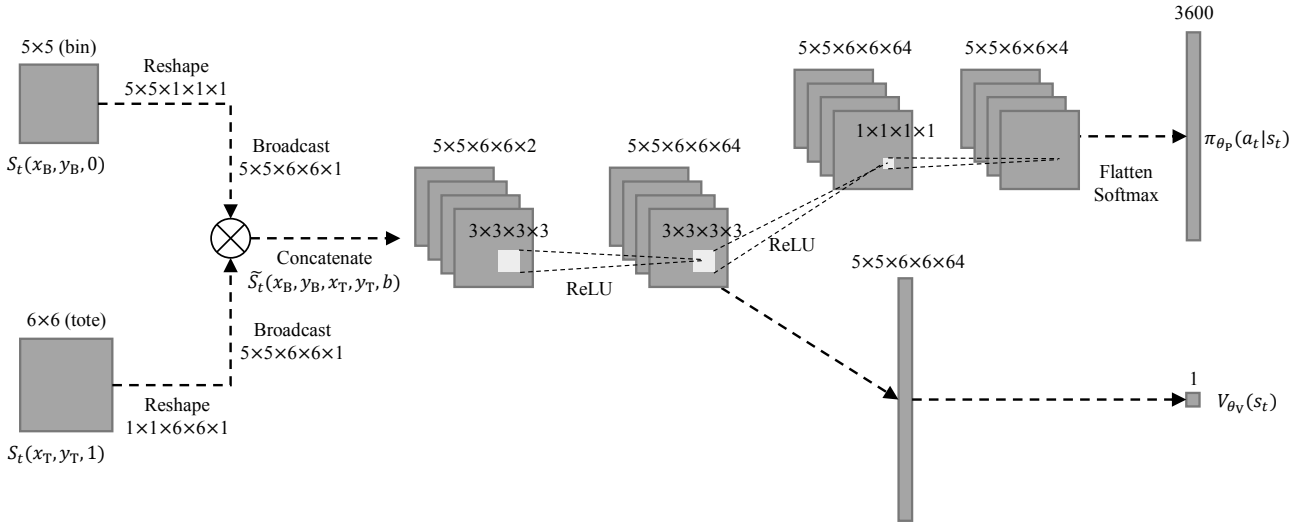
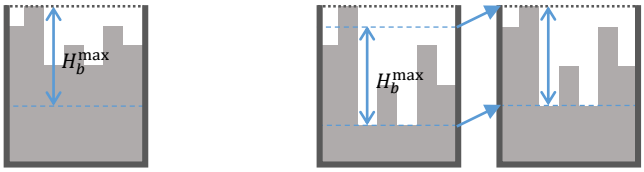


Fig. 3: Network architecture for the policy/value function.



(a)Without clipping (b)With clipping

Fig. 4: Example of depth clipping. (a) If the maximum depth value is below the threshold, the value is not clipped. (b) If the maximum depth value exceeds the threshold, the value is clipped to $H_b^{\max}$.

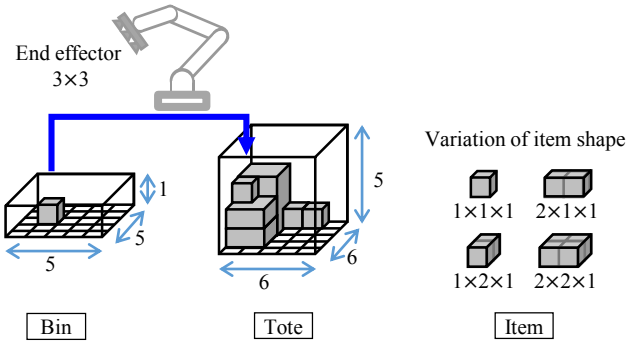


Fig. 5: Training condition.

TABLE I: Hyperparameters used in SectionV.

Hyperparameter	Value
Adam learning rate	2.5×10^{-4}
Minibatch size	32
Discount (γ)	0.99
VF coefficient	0.005
Entropy coefficient	0.002
Number of actors	16 for tote size $6\times6\times*$ 8 for tote size $10\times10\times*$
Effector width/depth H_0^{max}, H_1^{max}	3×3 5
Filters for conv. layer (in order)	64filters, $(3\times3\times3\times3)$, 64filters, $(3\times3\times3\times3)$, 4filters, $(1\times1\times1\times1)$,

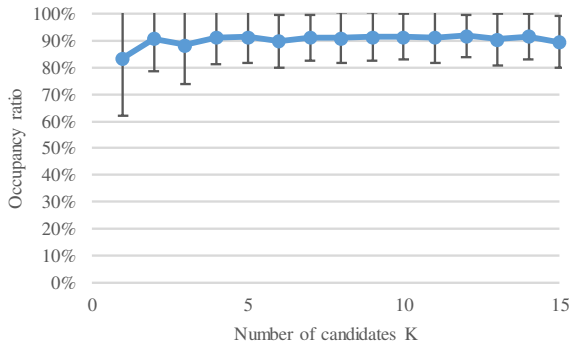


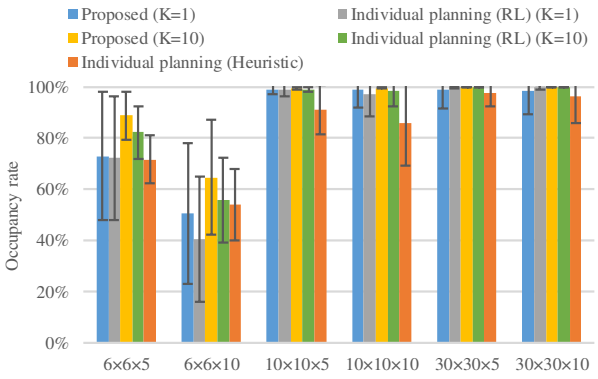
Fig. 6: Relationship between the number of candidates K and occupancy rate.

A. *Comparison with Individual Planning Methods*

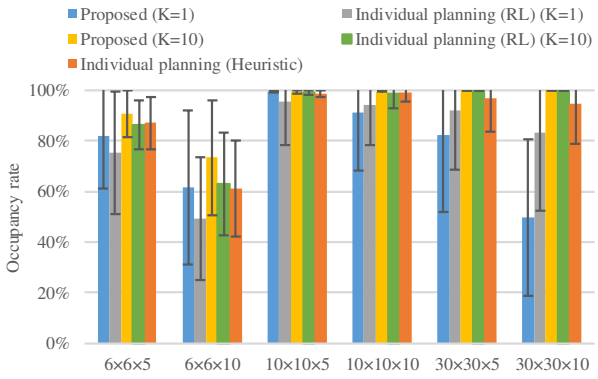
First, we compare the occupancy rate achieved by the proposed method with two methods which plan item picking and placing individually.

- **Proposed:** A simultaneous planning method.
- **Individual planning (RL):** A reinforcement-learning-based individual planning method.
- **Individual planning (Heuristic):** A heuristic-based individual planning method.

Individual planning (RL) is a combination of a heuristic picking rule and a policy for item placing. With the heuristic picking rule, the picking position is selected so that the end effector picks an item at the top-left corner as much as possible. The policy for item placing only decides the placing position and direction in the tote. The network architecture is mostly the same as the proposed method. A feature selection operation, which selects features corresponding to the picking position, is added to right after the last convolution layer of the policy and value function. We do not consider



(a) Number of direction R=2.



(b) Number of direction R=3.

Fig. 7: Comparison with individual planning methods.

a combination of individually trained picking and placing policies in this experiment. This is because the success rate of heuristic picking method is 100% which is always better than that of an individually trained picking policy.

Individual planning (Heuristic) is a combination of heuristic picking rule and deepest bottom left (DBL) method based placing rule, and we used MaxRects algorithm [19] for the DBL method. Then, a placing position is selected based on the DBL metric while satisfying the stability of the placing area (the base area should be greater than 50% of the item).

The policy of the proposed method and Individual planning (RL) were trained on a bin size of 5×5×1, tote size of 6×6×5, and end effector size of 3×3. After the policies are trained, we evaluate the performance on problems with a larger number of grids and heights. In this evaluation, the sizes of items are randomly selected from 4 patterns of {1×1×1, 2×1×1, 1×2×1, and 2×2×1}. We visualized the training condition in Fig.5. In order to assume a restriction of robot motion, we restricted the rotation of end effector(R) for this experiment.

Moreover, in the proposed method and Individual planning (RL), we used the action that succeeds in picking and placing an item from among the actions with the K highest probabilities. The K candidates were selected based on the probability $\pi_{\theta_P}(a|s)$, and these actions were evaluated by using forward simulation. A similar process was performed

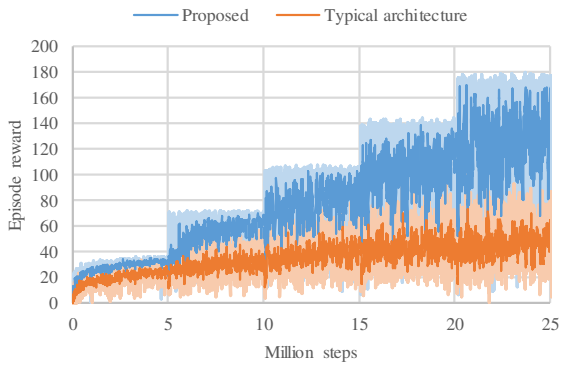


Fig. 8: Learning curve of proposed method.

for the DBL method. The placing position to which placing an item was succeeded was selected from several candidates[1] based on the DBL metric.

In a preliminary experiment, we investigated the relationship between the number K and the occupancy. Fig.6 shows the results. In this experiment, the occupancy rate of proposed method was evaluated for a tote size 6×6×5 and R=3. The occupancy rate was found to increase with the number K. Because the occupancy rate was converged well at about K=10, this value was adopted for subsequent experiments.

Fig.7 shows a comparison of the results with individual planning methods. The occupancy rate of the proposed method exceeded that of the other methods, especially for small tote sizes. Meanwhile, the occupancy rate of the proposed method and Individual planning (RL) exceeded that of the Individual planning (Heuristic) for large tote sizes. This may be because the individual planning methods are unable to place an item in the desired position due to collisions between the end effector and already placed items, that is, the picking position was selected without considering the placing position. These results show that simultaneous planning for picking and placing is important for the container loading problem.

B. Comparison with Reinforcement-Learning-Based methods

Second, we compare the proposed method against the following reinforcement-learning-based methods. Note that these methods do not use the depth clipping described in Section V-A.

- **4D FCN:** The network architecture for the policy/value function is same as proposed method.
- **Typical architecture:** A fully connected layer is added to just before softmax operation of the policy of proposed method.
- **MCTS:** Monte-Carlo tree search. The number of simulations per step was set to 5000.

[1]The number of candidates depends on the shape of unoccupied spaces [19].

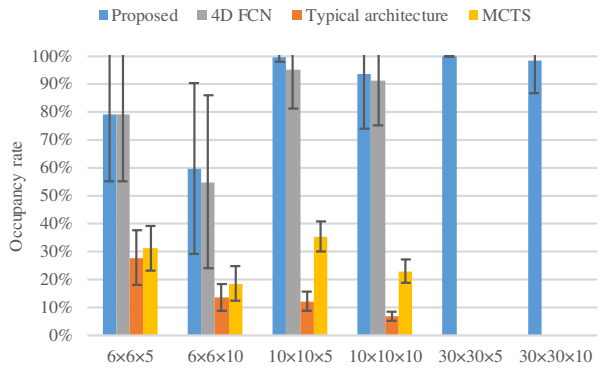


Fig. 9: Comparison with other methods.

The policy of the proposed method is trained on the same conditions as Section V-A. After the policies are trained, we evaluate the performance on problems with a larger number of grids and heights. On the other hand, the policies of 4D FCN and Typical architecture were trained on the same grid size as the one for evaluation. We used the number of directions R=4 and the number of candidates K=1 for all methods.

The learning curves of the proposed method and Typical architecture are shown in Fig.8. The height of the tote was increased by 1 for each curriculum and the number of steps for each curriculum was 5 million steps. The dark line represents a smoothing curve using the 5 moving average. The figure shows our network architecture for the policy provides better performance.

We compared the occupancy rate of our method with other reinforcement-learning-based methods. The average occupancy rate for the 100 evaluations using the trained policy is shown in Fig.9. The policy trained by the proposed method is applicable to settings with larger tote sizes. However, the policies of 4D FCN and Typical architecture could not be trained under conditions of a tote width/depth of 30×30 because of large consumption of GPU memory [2]. Moreover, it was difficult to execute under these conditions using MCTS in a practical amount of time.

It can be seen that the performance of the proposed method does not decrease as the size of totes increases. This result indicates that the proposed method successfully learns a policy that can generalize well to problems with larger tote sizes. The occupancy rate for 6×6×10 was lower than that for 6×6×5. It is thought that this setting is more difficult than 6×6×5 because the end effector collides with the tote or already placed items, and an increase in the number of patterns in which the region cannot be arranged due to an insufficient support area.

As the learning curve in Fig.8, it is difficult for Typical architecture, which uses typical network architecture for the policy, to learn this task due to its low data efficiency. The occupancy rate of 4D FCN was slightly lower than that of the proposed method. We can also see that MCTS does not perform well and achieve very low occupancy rates. This is due to a very large action space which makes it difficult for MCTS to perform sufficient simulations.

[2]We used Titan RTX for our experiments

VI. Conclusion

In this paper, we designed a new task, simultaneous planning for item picking and placing, and proposed a method to solve it by using DRL. Our method enables us to produce a shared policy which can be applied to larger action space settings than those used in the training. Although conventional methods could not solve the task even in the practical settings due to the large action space, our method could solve it by applying the policy trained on the small action space setting to a larger one. Experimental results showed the effectiveness of our method and also it could be trained in a generalized way even under the condition of the rotation of the end effector was constrained. In future work, we will consider planning with movable range of robot and item with different shapes. We will also apply our methods to real physical systems.

References

[1] X. Zhao, J. Bennell, T. Bektaş, and K. Dowsland, "A comparative review of 3d container loading algorithms," *International Transactions in Operational Research*, vol. 23, 05 2014.

[2] Y. Deng, X. Guo, Y. Wei, K. Lu, B. Fang, D. Guo, H. Liu, and F. Sun, "Deep reinforcement learning for robotic pushing and picking in cluttered environment," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 619–626.

[3] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," *arXiv:1803.09956*, 2018.

[4] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, "Learning to manipulate deformable objects without demonstrations," *arXiv:1910.13439*, 2019.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[6] S. Martello, D. Pisinger, and D. Vigo, "The three-dimensional bin packing problem," *Operations Research*, vol. 48, 02 1998.

[7] L. Duan, H. Hu, Y. Qian, Y. Gong, X. Zhang, J. Wei, and Y. Xu, "A multi-task selected learning approach for solving 3d flexible bin packing problem," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019, p. 1386–1394.

[8] A. Laterre, Y. Fu, M. K. Jabri, A.-S. Cohen, D. Kas, K. Hajjar, T. S. Dahl, A. Kerkeni, and K. Beguir, "Ranked reward: Enabling self-play reinforcement learning for combinatorial optimization," *arXiv:1807.01672*, 2018.

[9] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[10] O. Kundu, S. Dutta, and S. Kumar, "Deep-pack: A vision-based 2d online bin packing algorithm with deep reinforcement learning," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2019, pp. 1–7.

[11] S. Woo, J. Yeon, M. Ji, I. C. Moon, and J. Park, "Deep reinforcement learning with fully convolutional neural network to solve an earthwork scheduling problem," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 4236–4242.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[13] R. Bostelman and J. Falco, "Survey of industrial manipulation technologies for autonomous assembly applications," National Institute of Standards and Technology, Tech. Rep. NIST Interagency/Internal Report (NISTIR) - 7844, 2012.
[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.
[15] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, 2016, pp. 1928–1937.
[16] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, 2015, pp. 1889–1897.
[17] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems 12*, 2000, pp. 1057–1063.
[18] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, p. 41–48.
[19] J. Jylänki, "A thousand ways to pack the bin – a practical approach to two-dimensional rectangle bin packing," 2010. [Online]. Available: http://clb.demon. fi/files/RectangleBinPack.pdf (Accessed February 5, 2020)