Sample-Efficiency, Stability and Generalization Analysis for Deep Reinforcement Learning on Robotic Peg-in-Hole Assembly

Yuelin Deng, Zhimin Hou, Wenhao Yang, and Jing Xu(✉)

Department of Mechanical Engineering, Tsinghua University, Beijing 100084, China
jingxu@tsinghua.edu.cn

Abstract. In the field of robotic assembly, deep reinforcement learning (DRL) has made a great stride in the simulated performance and holds high promise to solve complex robotic manipulation tasks. However, a huge number of efforts are still needed before RL algorithms could be implemented in the real-world tasks directly due to the risky but insufficient interactions. Additionally, there is still a lack of analyzation in the sample-efficiency, stability and generalization ability of RL algorithms. As a result, Sim2Real, analyzing RL algorithms in simulation and then implementing in real-world tasks, has become a promising solution. Peg-in-hole assembly is one of the fundamental forms of the robotic assembly in industrial manufacturing. In the paper, we set up a simulation platform with physical contact models of both single and multiple peg assembly configurations; we then provide the commonly used RL algorithms with an empirical study of the sample-efficiency, stability and generalization, ability; we further propose a new algorithm framework of Actor-Average-Critic (AAC) for better stability and sample-efficiency performance. Besides, we also analyze the existing reinforcement learning with hierarchical structure (HRL) and demonstrate its better generalization ability into new assembly tasks.

Keywords: Deep reinforcement learning · Peg-in-hole assembly

1 Introduction

Peg-in-hole assembly is a fundamental form of industrial assembly tasks. The existing research has achieved a certain level through the conventional controllers e.g., PD and impedance controllers, however, such acquired performance largely depends on contact state recognition. It is still difficult to address the multi-peg assembly problem via conventional controllers due to the complicated contact model [7]. Additionally, modern manufacturing requires not only quantity, but also quality and efficiency which introduce the need for greater stability and generalization ability against environmental uncertainties.

Y. Deng, Z. Hou and W. Yang—Joint first author.

X.-J. Liu et al. (Eds.): ICIRA 2021, LNAI 13014, pp. 393–403, 2021.
https://doi.org/10.1007/978-3-030-89098-8_38

Reinforcement learning (RL) holds the promise to learn assembly skills without contact state recognition [8]. Particularly, deep reinforcement learning (DRL) combined with learning representation skills, has solved a large range of complex robotic manipulation tasks [9]. DRL techniques have been extensively explored to learn the competitive assembly skills [5], however, the sample efficiency, stability, and generalization ability are less discussed. The real-world peg-in-hole assembly with small clearance requires both vision information and accurate force/torque feedback, which guides the assembly movement and avoid failure caused by large contact force. To our best knowledge, there are less existing work on the simulation platform for peg-in-hole assembly tasks basing on force feedback. The first motivation of this paper is to set up a peg-in-hole assembly simulator with physical contact prediction ability. With the contact force detected and read from the force sensor in the simulation platform, the sample-efficiency, stability and generalization ability of the current RL algorithms can then be analyzed.

The sample-efficiency is an essential issue for real-world applications as interactions with real-world environments are at great cost. Compared with the on-policy RL that requires new samples for every training step, off-policy RL can reuse the past experience and reduce the sample complexity [10]. Deep Deterministic Policy Gradient (DDPG) is one of the popular off-policy RL algorithms to learn a high-dimensional deterministic policy. Therefore, many DDPG-like algorithms have been put forward and researched to handle the peg-in-hole assembly tasks [2,15]. Exploration is another essential topic for RL algorithms to reduce the sample complexity and search for the optimal skills, therefore, some approaches were proposed to guide the efficient exploration basing on conventional controllers or human demonstration [15]. Recently, model-based RL assisted with an estimated dynamic environment model is showing a great promise to reduce the need for interactions [9]. Guided policy search (GPS) has shown a competitive performance to learn manipulation skills with the estimated local dynamic model [9].

DRL implemented basing on deep neural networks has achieved great success to solve complex control tasks [9]. However, most of the RL algorithms are sensitive to the selection of hyper parameters [10]. Compared with supervised learning methods, RL training is often unstable without the fixed targets, which is not allowed for real-world tasks. Some practical training techniques have been studied to improve the stability e.g., mini-batch training [10], delayed target network to address the over-estimation issues in function approximation [14]. To address the bad actions brought by random explorations, the residual RL learns advanced skills basing on the pre-trained policy or conventional controllers [6]. Another attractive direction is to reformulate the action space [13], meaning that the action will be constrained by a safe and efficient scheme. Three action spaces(PD control, inverse dynamic, and impedance control) to solve the commonly used robotic tasks are compared and discussed [13].

DRL holds the promise to learn the skills with better robustness and generalization. Random initial positions, peg-hole clearance, and sizes of pegs are often considered as the environmental variables to demonstrate the robustness and generalization ability of DRL methods [2,15]. The adaptability of learned

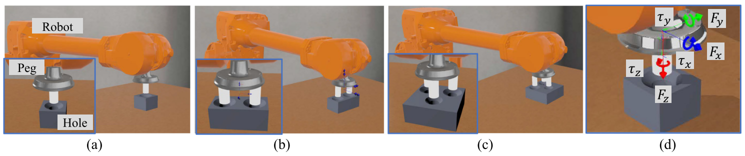

Fig. 1. Structure of peg-in-hole assembly. (a) Single (b) Dual (c) Triple (d) F/T sensor.

assembly skills to the new tasks with different shapes of pegs have been discussed [2]. To improve the task generalization and environment adaptation ability, a deep model fusion framework is proposed to fuse the previously learned knowledge from different assembly environments. Currently, the complex control tasks are sometimes structured hierarchically, hierarchical reinforcement learning (HRL) aims to learn a hierarchical policy with some basic skills, which can be easily transferred to new tasks [4]. However, up to now, the generalization for peg-in-hole assembly has not been discussed sufficiently.

The motivation of this paper is to promote the research for solving peg-in-hole assembly tasks by DRL and provide a guidance to develop the practical RL algorithm. The contributions are concluded as:

1. We set up a force-based simulation platform on Webots to train and test the RL agents for peg-in-hole assembly tasks in different task configurations.
2. We propose a data-efficient actor-average-critic (AAC) RL framework with both better training stability and higher sample-efficiency. We then verify the performance of the proposed algorithm framework and compare it with the methods from existing literature in different peg-in-hole assembly situations (with different clearance, friction, and the number of pegs), through our simulation platform.
3. We evaluate the transfer performance of the assembly skills learned on unseen assembly tasks with different numbers of pegs. We demonstrate that the assembly skills learned by HRL could be transferred to different assembly situations (with different number of pegs) easier than the existing methods.

2 Peg-in-Hole Assembly Platform

The peg-in-hole assembly tasks (including single, dual and triple peg situations) are set up in Webots simulator [1], as shown in Fig. 1(a)–(c).

2.1 Hardware

Robot: There is a variety of manipulators to choose from in Webots e.g., UR, PUMA, ABB IRB, etc. The robots can also be built given the Unified Robot Description Format (URDF). In this paper, ABB IRB 4600, a 6-DOF manipulator is selected to implement the peg-in-hole assembly tasks. All the motors of the robot in Webots are implemented with an underlying PID controller, the robot is controlled by giving the velocity and target angle of each motor.

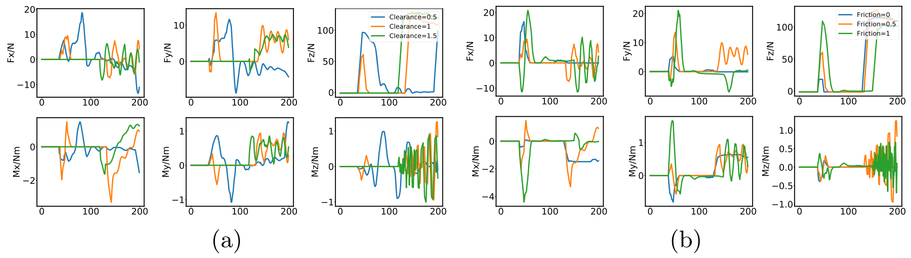

Fig. 2. F/T sensor signals (a) Different peg-hole clearance. (b) Different friction.

F/T Sensor: There is no available multi-axis force sensor in Webots, however, a motor can output force feedback to indicate the required force to hold its position. We built a chain of joints at a single point of the robot end-effector and apply motor behaviour, as shown in Fig. 1(d), which includes three rotational motors and three linear motors.

This chain of joints will serve as a 6-axis F/T sensor to detect the contact forces. However, due to the properties of the simulation engine, this error could be integrated to a noticeable level, which we consider as a uncontrollable random noise. To test the performance of the F/T sensor, a PD controller is applied to control the single peg-in-hole assembly in different situations. The force signals from F/T sensor are plotted as shown in Fig. 2.

Components: Three different numbers of cylindrical peg-hole components (single, dual, and triple) are set up. We assume that each peg-hole pair has the same configuration for dual and triple peg-in-hole assembly tasks. We added chamfers to the hole entrance so that we can focus on the insertion process instead of worrying about an extra search stage to align the peg. Additionally, the Coulomb friction coefficients and the clearance of peg-hole pairs can be changed to simulate different peg-in-hole assembly tasks. The parameters and their adjustable range are listed in Table 1.

Table 1. Adjustable parameters and properties

Name	Adjustable region
Number of peg-hole pairs	1,2,3
Coulomb friction coefficients	0 - inf
Hole size	30 mm
Peg-hole clearance	0.5 mm–1.5 mm
Hole position noise range	[−1, 1]mm

2.2 Software

The Webots platform is compatible with the system including Windows, Linux, and Mac OS. The connection between the simulator and the RL algorithms is implemented by the Python API, which reads the sensor feedback from the simulator as RL input, and outputs the control commands to the robot. Different assembly tasks can be set up via the Webots-Python interface given the parameters in Table 1.

3 RL Algorithms Analysis

3.1 Problem Statement

RL can solve all phases integrally based on the force feedback similar as "feeling" of human beings, which formulates the assembly process as the *Markov Decision Process(MDPs)*. At each time step t, according to the observed state $\boldsymbol{s}_t$, the robot select the "optimal" action $\boldsymbol{a}_t$ based on the policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$. The robot will receive the reward signals r_t from environment to evaluate the state-action pair, and then the environment transit to the next state $\boldsymbol{s}_{t+1} = p(\boldsymbol{s}_{t+1}|\boldsymbol{a}_t, \boldsymbol{s}_t)$. The policy is updated through maximizing the expected cumulative reward, as:

$$\mathbb{E}_{\boldsymbol{a}_t \sim \pi, \boldsymbol{s}_{t+1} \sim p(\cdot)} \left[\sum_{t=0}^{T-1} \gamma^t r_{t+1}(\boldsymbol{s}_t, \boldsymbol{a}_t) \right] \tag{1}$$

where γ denotes the discount factor. The action value function $Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_\Omega[G_t|\boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}]$ is defined to estimate the expected cumulative reward given the trajectory Ω, $G_t = \sum_{i=t}^{T-1} \gamma^{i-t} r_{i+1}(\boldsymbol{s}_i, \boldsymbol{a}_i)$ denotes the cumulative reward of one episode from $t = 0$ to T. The objective can be rewritten as:

$$\mathcal{J}(\pi) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a}) \sim \mathcal{P}_\pi} \left[Q^\pi(\boldsymbol{s}, \boldsymbol{a}) \right] \tag{2}$$

where $\mathcal{P}_\pi$ denotes the state-action marginals of the trajectory distribution induced by a policy π.

As shown in Fig. 1, the state for all the peg-in-hole assembly tasks often consists of the pose(position and orientation) of pegs and the force(force and moment) read from the F/T sensor, therefore, the state denotes as:

$$\boldsymbol{s}_t = [\underbrace{P_t^x, P_t^y, P_t^z}_{\text{position(mm)}}, \underbrace{O_t^x, O_t^y, O_t^z}_{\text{orientation}}(\circ), \underbrace{F_t^x, F_t^y, F_t^z}_{\text{force(N)}}, \underbrace{T_t^x, T_t^y, T_t^z}_{\text{torque(Nm)}}] \tag{3}$$

where the subscripts x, y and z denote the axes of the robot base coordinate. The basic action space $\boldsymbol{a}_t$ is denoted as follows:

$$\boldsymbol{a}_t = [\Delta P_t^x, \Delta P_t^y, \Delta P_t^z, \Delta O_t^x, \Delta O_t^y, \Delta O_t^z] \in \mathbb{R}^6 \tag{4}$$

which are utilized to control the pose of the pegs along the corresponding axis in the robot base coordinate.

A reward function is defined to decrease the assembly steps and avoid the unsafe contact force, as

$$r_t = \begin{cases} -0.1, \ done = False \ and \ safe = True \\ 1 - \dfrac{N}{N_{max}}, \ done = True \ and \ safe = True \\ -1 + \dfrac{D_t}{D_e}, \ safe = False \end{cases} \tag{5}$$

where N_{max} denotes the maximum number of steps in one episode; $N \in [0, N_{max})$ denotes the final number of steps. To encourage fewer number of assembly steps, the robot will receive -0.1 every step. Additionally, the penalty is designed according to the force signals, the assembly will be interrupted with reward -1 once the value of forces exceeding the setting boundary. D_e denotes the desired insertion depth, D_t denotes the final insertion depth when the assembly is not completed due to the forces exceeds the safe boundary.

3.2 Sample-Efficiency

In practice, two directions has achieved success in improving the sample-efficiency.

Residual RL for Control: The robotic assembly tasks can be divided into two parts. The first part is robot-related, which can be solved through the typical controller. And the another part is environment-related, involving the dynamics that can be addressed by RL algorithms [6]. In residual RL, the reward function is often formulated as two parts:

$$r_t = f(\boldsymbol{s}_{r,t}) + g(\boldsymbol{s}_{e,t}) \tag{6}$$

where $\boldsymbol{s}_{r,t}$ is the robot-related state and $f(\boldsymbol{s}_{r,t})$ is the reward to encourage the robot to approach the target position. $g(\boldsymbol{s}_{e,t})$ is the environment-related state in terms of different friction and contact model due to different geometries of assembly components.

Accordingly, the control command also consists of two parts:

$$\boldsymbol{u}_t = \pi_C(\boldsymbol{s}_{r,t}) + \pi_\theta(\boldsymbol{s}_{e,t}) \tag{7}$$

where $\pi_C(\boldsymbol{s}_{r,t})$ is the hand-designed conventional controller without considering the dynamics of environments. $\pi_\theta(\boldsymbol{s}_{e,t})$ is the residual DRL policy to compensate the dynamics of environment and supplement the flexibility of the conventional controller. For peg-in-hole assembly tasks, any RL algorithms (DDPG, TD3, SAC) can be implemented to learn the residual RL policy.

Action Space Reformulation: Action space reformulation can be considered as introducing the prior of policy structure. For different hardware and tasks requirements. PD controller given the parameters $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$, is often applied to feed the control errors without any assumption about the system, as:

$$\begin{aligned} \boldsymbol{u}_t &= \boldsymbol{K}_p \circ \boldsymbol{e}_t + \boldsymbol{K}_d \circ \dot{\boldsymbol{e}}_t \\ \boldsymbol{e}_t &= \boldsymbol{q}_d - \boldsymbol{q}_t, \quad \dot{\boldsymbol{e}}_t = \dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}_t \end{aligned} \tag{8}$$

where $\circ$ denotes the Hadamard product; $\boldsymbol{u}_t$ denotes the control torques in joint-space, therefore, the new action space includes the desired joint angle $\boldsymbol{q}_d$ and the desired joint velocity $\dot{\boldsymbol{q}}_d$.

The contact model of peg-in-hole assembly tasks for impedance controller is often formulated in the task-space, as:

$$\mathcal{M}_d(\ddot{\boldsymbol{x}}_t - \ddot{\boldsymbol{x}}_e) + \mathcal{B}_d(\dot{\boldsymbol{x}}_t - \dot{\boldsymbol{x}}_e) + \mathcal{K}_d(\boldsymbol{x}_t - \boldsymbol{x}_e) = \mathcal{F} \tag{9}$$

where $\mathcal{M}_d$, $\mathcal{B}_d$ and $\mathcal{K}_d$ denotes the desired inertia, damping and stiffness matrix in task-space, respectively. For position-control robot, the control command is position offset in task-space. For torque-control robot, the control command $\boldsymbol{u}$ is actuated torque on joints to compensate the dynamics from robot and feedforward external force.

Consequently, instead of controlling the $\{\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{x}_t, \dot{\boldsymbol{x}}_t, \tau\}$ directly, the reformulated action space often consist of the desired reference value $\{\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d, \boldsymbol{x}_d, \dot{\boldsymbol{x}}_d, \mathcal{F}_d\}$ and the corresponding parameters $\{\boldsymbol{K}_p, \boldsymbol{K}_d\}$.

3.3 Stability

The RL stability can be achieved through reducing the over estimation and providing the stable target value [3,14].

Over Estimation: Over estimation in function approximation is a main factor that introduces instability, which has been addressed in all actor-critic algorithms (e.g., TD3, SAC) [3] to reduce the variance and improve the final performance. For actor-critic architecture, TD3 and DDPG have the same objective via minimizing the loss:

$$1/2\left(r_{t+1} + \gamma Q(\boldsymbol{s}_{t+1}, \mu(\boldsymbol{s}_{t+1})|\boldsymbol{\omega}) - Q(\boldsymbol{s}_t, \boldsymbol{a}_t|\boldsymbol{\omega})\right)^2 \tag{10}$$

where $Q(\cdot|\boldsymbol{\omega})$ is the action value function estimated by a neural network given the state-action pair. $\boldsymbol{a}_{t+1} \sim \mu(\boldsymbol{s}_{t+1}|\boldsymbol{\theta}) \ : \ \mathcal{S} \rightarrow \mathcal{A}$ denotes the deterministic policy estimated by a neural network. The past experience $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_{t+1}, \boldsymbol{s}_{t+1})$ is sampled from replay buffer $\mathcal{M}$. TD3 implements two independent network (Q_1, Q_2) to estimate the action value function, the target value in (10) is corrected as $r_{t+1} + \gamma \min_{i=1,2} Q_i(\boldsymbol{s}_{t+1}, \mu(\boldsymbol{s}_{t+1}|\boldsymbol{\theta})|\boldsymbol{\omega}_i)$. Afterwards, the deterministic policy is updated based on the gradient of action value function with respect to the parameters of μ.

Averaged-Critic: The target network is commonly used trick to provide a stable objective and speed the convergence [10]. The parameters $\bar{\boldsymbol{\theta}}$ target network is slow-updating from the estimated network parameters $\boldsymbol{\theta}$ through the scheme $\bar{\boldsymbol{\theta}} \leftarrow \zeta_\theta \boldsymbol{\theta} + (1-\zeta_\theta)\bar{\boldsymbol{\theta}}$. Another approach to reduce the variance of target approximation error is to replace the target network with the average over the previous learned action value estimation. In this paper, we extend this idea to the continuous control RL algorithms to improve the stability and final performance. The target value in (10) is replaced by the average over the past K estimated Q as:

$$r_{t+1} + \gamma \frac{1}{K} \sum_{i=0}^{K} Q_i(\boldsymbol{s}_{t+1}, \mu(\boldsymbol{s}_{t+1}|\bar{\boldsymbol{\theta}})|\boldsymbol{\omega}_i) \tag{11}$$

where $Q_i(\boldsymbol{s}_{t+1}, \mu(\boldsymbol{s}_{t+1}|\bar{\boldsymbol{\theta}})|\boldsymbol{\omega}_i)$ represents the past ith estimated action-value.

3.4 Generalization Improvement

Two important directions are often adopted to improve the generalization of RL algorithms. **Dynamic randomized:** To improve the robustness to environmental noise, randomized dynamic was utilized to train the RL algorithms [12], the dynamics parameters β are sampled from ρ_β for each episode as $\boldsymbol{s}_{t+1} = p(\boldsymbol{s}_{t+1}|\boldsymbol{a}_t, \boldsymbol{s}_t, \beta)$. The dynamics parameters are as a part of the input to estimate the action value function, which can reduce the variance of the policy gradient via compensating the environmental dynamics.

HRL: Recently, HRL is researched to solve the complex manipulation tasks through learning sub-policies. These learned sub-policies may represents the basic skills to solve the complex tasks, therefore, some HRL work has achieved better performance through transferring the sub-policies [4,11]. Compared to the objective (2), the hierarchical policy is derived with a discrete variable $o \in \mathcal{O}$ as $\pi_h(\boldsymbol{a}_t|\boldsymbol{s}_t) = \sum_{o\in\mathcal{O}} \pi_{\mathcal{O}}(o|\boldsymbol{s}_t)\pi_o(\boldsymbol{a}_t|\boldsymbol{s}_t)$, $\pi_o(\boldsymbol{a}_t|\boldsymbol{s}_t)$ denotes the lower-level policy to generate the action, $\pi_{\mathcal{O}}(o|\boldsymbol{s}_t)$ denotes the upper-level policy to select the lower-level policy given the state. The hierarchical policy $\pi_h(\boldsymbol{a}_t|\boldsymbol{s}_t)$ can be learned given the objective (2) based on the policy gradient theorem and additional constraints.

4 Experiment

We implement the proposed actor-average-critic (AAC) with different average length ($K = 2, 3, 5, 10$), the DDPG [10], TD3 [3] and a HRL algorithm (adInfoHRL) [11] on three different assembly tasks with different clearance and friction setting. For the equal comparison, the hyper-parameters of AAC are selected as TD3, and the hyper-parameters of other algorithms are selected as the original papers, all the algorithms are implemented with action space reformulation in (8).

4.1 Performance: Sample-Efficiency and Stability

As shown in Figs. 3–5, all the algorithms are trained for 10^5 steps with five seeds and cumulative reward is tested every 10^3 steps. Darker lines represents average reward over five seeds. Shaded region shows the standard deviation of average reward. The assembly task becomes more complex from single to triple, DDPG and TD3 cannot perform the assembly tasks effectively, obviously, the proposed AAC with different average length can achieve both the better sample efficiency and stability performance in most of the above assembly tasks.

4.2 Performance: Generalization

To demonstrate the generalization ability, we implement the HRL with different options ($\mathcal{O} = 2, 4, 6$) in dual and triple peg-in-hole assembly tasks. As shown in Fig. 6(a) and 7(a), the HRL with four options can achieve largely better sample efficiency and stability performance than TD3 and DDPG. To demonstrate there kinds of transfer performance, the learned assembly policy is loaded, then tested or re-trained again in a new assembly tasks (Figs. 3–5).

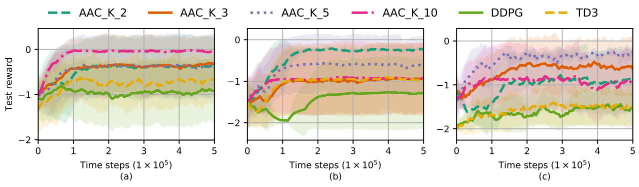

Fig. 3. Single peg-in-hole assembly. (a) Friction-0.5-clearance-1 mm. (b) Friction-0.5-clearance-0.5 mm. (c) Friction-1-clearance-1 mm.

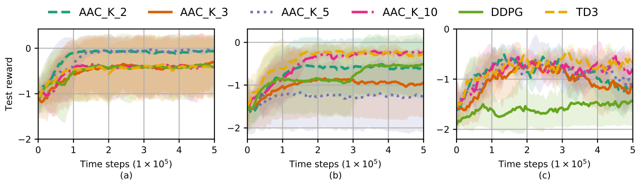

Fig. 4. Dual peg-in-hole assembly. (a) Friction-0.5-clearance-1 mm. (b) Friction-0.5-clearance-0.5 mm. (c) Friction-1-clearance-1 mm.

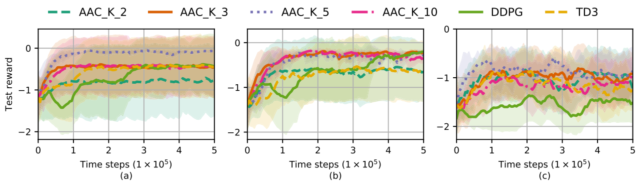

Fig. 5. Triple peg-in-hole assembly. (a) Friction-0.5-clearance-1 mm. (b) Friction-0.5-clearance-0.5 mm. (c) Friction-1-clearance-1 mm.

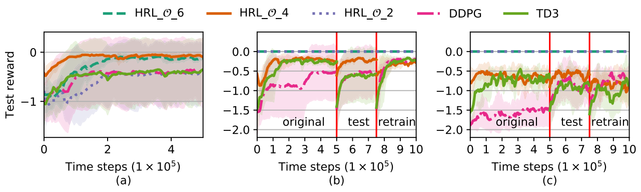

Fig. 6. Dual peg-in-hole assembly. (a) Friction-0.5-clearance-1 mm. (b) Friction-0.5-clearance-0.5 mm. (c) Friction-1-clearance-1 mm.

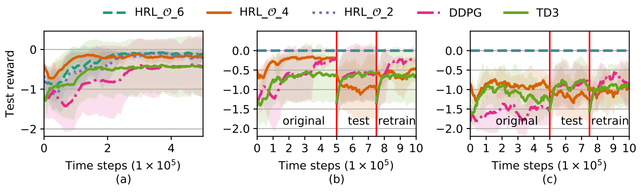

Fig. 7. Triple peg-in-hole assembly. (a) Friction-0.5-clearance-1 mm. (b) Friction-0.5-clearance-0.5 mm. (c) Friction-1-clearance-1 mm.

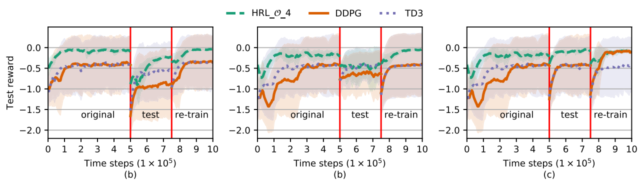

Fig. 8. Different assembly tasks transfer performance. (a) Single-to-dual. (b) Dual-to-triple. (c) Single-to-triple.

Different Clearance: The learned assembly policies in tasks (friction-0.5-clearance-1 mm) are loaded and then tested in new tasks (friction-0.5-clearance-0.5 mm) with smaller clearance for 5×10^4 steps. Additionally, the learned assembly policies are also re-trained in new tasks for 5×10^4 steps. As shown in Fig. 6(b) and 7(b), HRL with four options has achieved better sample-efficiency and stability performance in new assembly tasks.

Different Friction: Likewise, the learned assembly policies in tasks (friction-0.5-clearance-1mm) are loaded and tested in new tasks (friction-1-clearance-1mm) with big friction for 5×10^4 steps. Additionally, the learned assembly policies are also re-trained in new tasks for 5×10^4 steps. As shown in Fig. 6(c) and 7(c), HRL with four options also has achieved better sample-efficiency and stability performance in new tasks, especially in single and dual peg-in-hole assembly tasks.

Different Number of Pegs: The assembly tasks with different number of pegs have different contact model. The learned assembly policy in single peg-in-hole assembly task (friction-0.5-clearance-1 mm) is loaded, then tested and re-trained in dual (see Fig. 8(a)) and triple (see Fig. 8(b)) peg-in-hole assembly tasks with same the clearance and friction settings. Secondly, the learned assembly policy in dual peg-in-hole assembly task (friction-0.5-clearance-1 mm) is loaded, then tested and re-trained in triple peg-in-hole assembly task (friction-0.5-clearance-1 mm) (see Fig. 8(c)). Obviously, the assembly skills learned by HRL have better generalization ability than TD3 and DDPG, which can accelerate the assembly policy learning in new tasks.

5 Conclusion and Discussion

The sample-efficiency, stability, and generalization of RL techniques on peg-in-hole assembly tasks are analyzed and concluded, we propose the AAC framework to improve the sample-efficiency and stability performance, the effectiveness of AAC is demonstrated in three kinds of assembly tasks with 3 different settings, respectively. We compare the generalization of our method and two commonly used RL algorithms in different assembly tasks with three different friction and clearance settings. Consequently, the assembly skills learned by HRL can achieve better generalization performance than commonly used TD3 and DDPG. This lead us to believe that our algorithm have high potential to transfer a policy learned in simulation to real-world efficiently, which will be our future work.

References

1. Webots r2019b (2019). https://www.cyberbotics.com/#webots
2. Fan, Y., Luo, J., Tomizuka, M.: A learning framework for high precision industrial assembly. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 811–817. IEEE (2019)
3. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning, pp. 1582–1591 (2018)
4. Hou, Z., Fei, J., Deng, Y., Xu, J.: Data-efficient hierarchical reinforcement learning for robotic assembly control applications. IEEE Trans. Ind. Electron. **68**(11), 11565–11575 (2020)
5. Inoue, T., De Magistris, G., Munawar, A., Yokoya, T., Tachibana, R.: Deep reinforcement learning for high precision assembly tasks. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 819–825. IEEE (2017)
6. Johannink, T., et al.: Residual reinforcement learning for robot control. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 6023–6029. IEEE (2019)
7. Johannsmeier, L., Gerchow, M., Haddadin, S.: A framework for robot manipulation: skill formalism, meta learning and adaptive control. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 5844–5850. IEEE (2019)
8. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: a survey. Int. J. Robot. Res. **32**(11), 1238–1274 (2013)
9. Levine, S., Wagener, N., Abbeel, P.: Learning contact-rich manipulation skills with guided policy search. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 156–163, May 2015
10. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
11. Osa, T., Tangkaratt, V., Sugiyama, M.: Hierarchical reinforcement learning via advantage-weighted information maximization. In: International Conference on Learning Representations (2019)
12. Peng, X.B., Andrychowicz, M., Zaremba, W., Abbeel, P.: Sim-to-real transfer of robotic control with dynamics randomization. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–8. IEEE (2018)
13. Varin, P., Grossman, L., Kuindersma, S.: A comparison of action spaces for learning manipulation tasks. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6015–6021 (2019)
14. Wu, D., Dong, X., Shen, J., Hoi, S.C.: Reducing estimation bias via triplet-average deep deterministic policy gradient. IEEE Trans. Neural Netw. Learn. Syst. **31**(11), 4933–4945 (2020)
15. Xu, J., Hou, Z., Wang, W., Xu, B., Zhang, K., Chen, K.: Feedback deep deterministic policy gradient with fuzzy reward for robotic multiple peg-in-hole assembly tasks. IEEE Trans. Ind. Inform. **15**(3), 1658–1667 (2018)