Model Parameter Adaptive Instance-Based Policy Optimization for Episodic Control Tasks of Nonholonomic Systems

Kyotaro Ohashi
Graduate School of Science and Technology,
Shinshu University
17w2014e@shinshu-u.ac.jp

Natsuki Fujiyoshi
Graduate School of Science and Engineering,
Shinshu University
shinshuclub@gmail.com

Naoki Sakamoto
Graduate School of Medicine, Science and Technology,
Shinshu University
18hs202g@shinshu-u.ac.jp

Youhei Akimoto
Faculty of Engineering, Information and Systems,
University of Tsukuba
akimoto@cs.tsukuba.ac.jp

ABSTRACT

Evolutionary Computation (EC) attracts more and more attention in Reinforcement Learning (RL) with successful applications such as robot control. Instance-Based Policy (IBP) is a promising alternative to policy representations based on Artificial Neural Networks (ANNs). The IBP has been reported superior to continuous policy representations such as ANNs in the stabilization control of nonholonomic systems due to its nature of bang-bang type control, and its understandability. A difficulty in applying an EC based policy optimization to an RL task is to choose appropriate hyper-parameters such as the network structure in ANNs and the parameters of EC. The same applies to the IBP, where the critical parameter is the number of instances that determines mode flexibility. In this paper, we propose a novel RL method combining the IBP representation and optimization by the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), which is a state-of-the-art general-purpose search algorithm for black-box continuous optimization. The proposed method, called IBP-CMA, is a direct policy search that adapts the number of instances during the learning process and activates instances that do not contribute to the output. In the simulation, the IBP-CMA is compared with an ANN-based RL, CMA-TWEANN.

CCS CONCEPTS

• **Computing methodologies** → **Reinforcement learning**; • **Mathematics of computing** → **Continuous optimization**;

KEYWORDS

Reinforcement learning, instance based policy, CMA-ES, episodic tasks, hyper-parameter adaptation

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07…$15.00
https://doi.org/10.1145/3205651.3208295

ACM Reference Format:
Kyotaro Ohashi, Natsuki Fujiyoshi, Naoki Sakamoto, and Youhei Akimoto. 2018. Model Parameter Adaptive Instance-Based Policy Optimization for Episodic Control Tasks of Nonholonomic Systems. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205651.3208295

1 INTRODUCTION

Reinforcement Learning (RL) is a framework to learn *policies* that are mappings from a state-space to an action-space through interaction with an environment [5, 13]. Representative methods of RL include *value function method* [13] (e.g., Q-learning [16]) and Direct Policy Search (DPS) [8]. Recently, Evolutionary Computation (EC) methods draw more and more attention in RL communities [10, 17] with successful applications such as robot control tasks [2, 15].

In EC communities, policy representations based on Artificial Neural Networks (ANNs) have been widely investigated, due to their high flexibility. A difficulty when applying ANN based policy optimization is to tune hyper-parameters regarding the network topology and the EC method. For example, one needs to choose the number of layers and number of neurons in each layer beforehand. However, the tuning of such parameters requires trial-and-error and is often prohibitively expensive. NeuroEvolution of Augmenting Topologies (NEAT) and its improvements [6, 11, 12] have been proposed to adapt the connection weights as well as the network topology during the learning process. It relaxes the hyper-parameter tuning of the ANN-based RL methods. To further improve the accuracy of the control policy and achieve a parameter-less RL framework, the CMA-TWEANN [9] employs the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), which is a state-of-the-art search algorithm for black-box continuous optimization, as the optimization kernel of the connection weights.

Instance-Based Policy (IBP), also called exemplar-based policy [3–5], is a promising alternative to policy representations based on ANNs. The IBP maintains a set of *instances*, each of which consists of a state vector and some information such as a promising action at the given state. The learning agent determines the next action based on the nearest instance from the current state of itself. The instances (i.e., the state vector and action vector) are optimized so as to maximize/minimize a reward/cost, respectively. The IBP has been reported superior to continuous policy representations such

as ANNs in the stabilization control tasks of nonholonomic systems due to its nature of bang-bang type control [7, 14]. However, the same difficulty as in the ANN-based policy optimization needs to be solved, i.e., hyper-parameter tuning of the model and the optimization algorithm. In the IBP, the critical parameter is the number of instances that determines mode flexibility and the number of parameters to be optimized. Moreover, the optimization method for the IBP faces the following difficulties: inter- and intra-instance dependencies and the redundancies of the policy representation, the latter of which is difficult for a general-purpose optimization method to deal with.

In this paper, we propose a novel RL method combining the IBP representation and the CMA-ES, named the IBP-CMA, for episodic control tasks of nonholonomic systems. The IBP-CMA optimizes the instance vectors by the CMA-ES and adapts the number of instances during the learning process. Since the CMA-ES excels at treating variable dependencies, it is expected that the CMA-ES fits well the IBP optimization. To treat the redundancies of the IBP and automate the hyper-parameter tuning, we incorporate a mechanism to adapt the number of instances. The IBP-CMA forms a quasi-parameter-free RL method.

This paper begins with describing the IBP and its difficulties in Section 2. We propose the IBP-CMA in Section 3. In Section 4, we apply the IBP-CMA to RL tasks and compare its performance with the CMA-TWEANN [9]. Section 5 discusses the conclusions of this paper and future work.

2 INSTANCE-BASED POLICY

The IBP consists of pieces of information paired with a specific state, e.g., "In the state $\boldsymbol{s}_i$, the agent should take the action $\boldsymbol{a}_i$.". A piece of information is called an *instance*. In this study, the instance is always a pair of a state vector and an action vector. Given the observed state $\boldsymbol{s}$ of the agent in the environment, the agent refers to the instance that is closest to $\boldsymbol{s}$, called *reference instance*, and takes the action of the instance. The IBP is applicable as long as the distance is defined in the state-space, i.e., it does not matter whether the state-action space is continuous or discrete, so the IBP has a broad application scope. In this study, however, we focus on the real state-action space, which is a typical case in control tasks.

2.1 Policy Representation

The IBP consists of L instances $I = \{I_1, \ldots, I_L\}$. The lth instance I_l composed of a state vector $\boldsymbol{s}_l \in \mathbb{R}^{N_{\text{state}}}$ and an action vector $\boldsymbol{a}_l \in \mathbb{R}^{N_{\text{action}}}$. Given the state $\boldsymbol{s}$ of the agent, the *reference instance* $I_{\bar{l}}$ is the nearest instance from $\boldsymbol{s}$, that is $\bar{l} = \operatorname{argmin}_{l=1,\ldots,L} \|\boldsymbol{s}_l - \boldsymbol{s}\|$. The action of the agent is the action of the reference instance corresponding to the current state. The policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{L(N_{\text{state}}+N_{\text{action}})}$ is a column vector, which is obtained by stacking the state and action vectors of each instance one to the other.

2.2 Episodic Tasks

In this paper, we deal with episodic tasks where one episode is T step. Let $\boldsymbol{s}_t$ be the state of the agent at time t. The agent determines action $\boldsymbol{a}_t$ according to policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t, \boldsymbol{\theta})$. The transition of the agent from $\boldsymbol{s}_t$ to the next state $\boldsymbol{s}_{t+1}$ by taking action $\boldsymbol{a}_t$ is defined by a (usually unknown) transition probability $P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$. The agent then receives the immediate reward (cost) r_t at each step. Let $R = \sum_{i=1}^{T} r_i$ be the cumulative reward (cost) up to time T. The objective of policy optimization is to find the policy parameter $\boldsymbol{\theta}$ that maximizes or minimizes the expected value $J(\boldsymbol{\theta}) = \mathbb{E}[R] = \sum_{i=1}^{T} \mathbb{E}[r_i]$ of R. From the optimization viewpoint, our design variables are the policy parameter $\boldsymbol{\theta}$, and our (possibly noisy) objective function is J.

2.3 Benefits of IBP

Comparing to the policies based on ANNs, an effect of each policy parameter is localized in the IBP. Change of value in one policy parameter affects the policy globally in ANNs, whereas the trajectory of the agent remains unchanged until the agent refers to the instance that is changed. For that reason, one can change the policy locally, which is desired when solving a nonholonomic control task consisting of multiple phases, e.g., swinging-up and stabilizing inverted pendulums. One can even divide a task into several sub-tasks, learn an IBP for each sub-tasks, and combine these IBPs in a straight-forward way. Moreover, it is much easier to derive an insight into the obtained policy due to its simple representation. The understandability of the policy representation is a key to knowledge discovery after the optimization process.

2.4 Difficulties in IBP

In the IBP optimization, there are roughly two difficulties. In this section, we describe each difficulty and its factors.

2.4.1 Difficulty when applying IBP: Tuning of the number of instances. As well as most of the other policy representation models, a hyper-parameter tuning is required when applying IBP based RL to an unknown task. In the IBP, the user must set an appropriate number of instances L. However, we cannot know the optimal number of instances in general control problems. Therefore, it is a herculean task to tune the number of instances. Too many instances increase extra computational costs, whereas too few instances lead to a crude policy representation. Moreover, too many instances result in many instances that called *inactive instance*, and the IBP optimization algorithm will suffer from the problem described below.

2.4.2 Difficulty when optimizing IBP: Redundancies in the policy parameter space. Optimization of policy parameters often suffers from the redundant parameterization of a policy. The performance J of the agent depends only on the policy $\pi(\cdot|\cdot, \boldsymbol{\theta})$. However, if a map $\boldsymbol{\theta} \mapsto \pi(\cdot|\cdot, \boldsymbol{\theta})$ is not bijective, i.e., all the parameters in a subset of the parameter space is mapped to the same policy, the performance is flat in the subset. Moreover, if two parameter vectors are mapped to distinct policies that have overlapping domains whose images are the same, the resulting trajectories of the agents are identical, and they are considered identical on the given episodic task. These features create a plateau in the objective function landscape, where an optimization algorithm can get stuck. Most recent policy representation models have redundant parameterization, and its optimization algorithm needs to cope with this issue.

In the IBP, there are mainly three sources of the redundancies as follows. The first source is the invariance of the permutation of instances in the parameter vector. The parameter vector $\boldsymbol{\theta} = \begin{bmatrix} I_1 & \ldots & I_L \end{bmatrix}^{\mathrm{T}}$ represents the same policy for any permutations of the elements that exist $L!$ ways. This means that the policy does

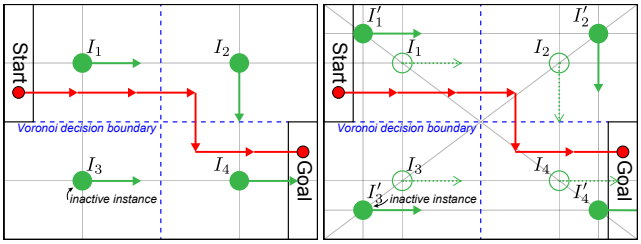


Figure 1: Policy representation and the sources of the redundancies in IBP.

not change for any indexing of instances (I_l) in Figure 1a. The second source is the fact that the policy is unchanged if the *Voronoi decision boundary* constructed by the instances is unchanged even if the parameters are moved (Figure 1b). Conversely, even if the optimization algorithm changes the state parameters of only one instance, the policy representation may change significantly. That is, the IBP is a non-separable problem, which is a factor that makes optimization difficult and a reason that we employ the CMA-ES to optimize the IBP. The third and the most severe source is the existence of *inactive instances*. In the following, we focus on the issue of inactive instances.

An instance such as I_3 in Figure 1a, that the agent never refers to during the episode of the task is called an *inactive instance*. Even if the optimization algorithm changes the action parameters of the inactive instances, it does not affect the state transition of the agent. Also, even if the algorithm changes the state parameters, as long as the instance is in the inactive region (unreferenced region), it does not affect the state transition. Technically, a change of an inactive instance results in a different policy, however, the change never affects the trajectory as long as we consider the same task.

Compared with the above two sources of redundancies, the existence of inactive instances produces a stronger redundancy for stochastic algorithms. From the measure-theoretic perspective, a set of policy parameters in the first two sources that are mapped to the same policy usually has zero Lebesgue measure. Therefore, a stochastic algorithm will never generate multiple candidate solutions in one such set. On the other hand, the existence of inactive instances produces nonzero Lebesgue measure sets.

Once we have an inactive instance, it is difficult for a general-purpose search algorithm to *activate* such an instance. First of all, a small fluctuation of an inactive instance will not affect the trajectory of the agent, hence its reward. Therefore, it is difficult to find a direction to make the instance active. A stochastic search algorithm may produce a jump of an inactive instance into a region where it will be referred during an episode, but it is not always likely. Even if the state vector of the instance is generated in the active region, since the action vector is not optimized for the current state vector, it likely results in poorer performance than other solutions that keep it inactive. Therefore, a search algorithm is unlikely to receive a selection bias to move the inactive instance toward a region where the instance will be referred during an episode and improve its performance.

In previous studies [3, 7], to address the issue of inactive instances, re-initialization and small fluctuation of inactive instances are performed. However, due to the reason as mentioned above, none of them works effectively. Moreover, In [7] that have addressed the IBP optimization in a continuous state-action space, the instance placement is optimized by a real-coded genetic algorithm. A crossover operator is defined in an instance-wise way. That is, the variable dependencies between instances are not taken into account. In this paper, we propose to use CMA-ES, which is often more robust and can better deal with variable dependencies than the real-coded genetic algorithm.

3 THE IBP-CMA

We propose a novel algorithm specialized for the IBP optimization for episodic tasks. The resulting algorithm is called *IBP-CMA*. It combines (1+1)-CMA-ES with a novel mechanism to activate inactive instances and to adapt the number of instances.

3.1 Notation

The policy parameter $\boldsymbol{\theta}$ which is the design vector of the (1+1)-CMA-ES is a $L(N_{\text{state}} + N_{\text{action}})$ dimensional vector. Further, the parent solution $\boldsymbol{x}$ and the evolution path $\boldsymbol{p}$ of the (1+1)-CMA-ES are vectors as follows

$$\boldsymbol{x} = \left[\boldsymbol{x}_1 \cdots \boldsymbol{x}_L\right]^{\mathrm{T}} \tag{1}$$

$$\boldsymbol{x}_l = \left[\boldsymbol{s}_l \quad \boldsymbol{a}_l\right] = \left[s_1^l \ldots s_{N_{\text{state}}}^l \; a_1^l \ldots a_{N_{\text{action}}}^l\right] \tag{2}$$

$$\boldsymbol{p} = \left[\boldsymbol{p}_1 \ldots \boldsymbol{p}_L\right]^{\mathrm{T}} \tag{3}$$

$$\boldsymbol{p}_l = \left[p_1^l \ldots p_{N_{\text{state}}}^l \; p_1^l \ldots p_{N_{\text{action}}}^l\right] \quad , \tag{4}$$

where $\boldsymbol{x}_l$ and $\boldsymbol{p}_l (l \in \{1, \ldots, L^{(t)}\})$ are $(N_{\text{state}} + N_{\text{action}})$ dimensional partial vectors of $\boldsymbol{x}$ and $\boldsymbol{p}$, respectively, corresponding to the lth instance in the parent solution $\boldsymbol{x}$. The covariance matrix $\mathbf{C} = \mathbf{A}\mathbf{A}^{\mathrm{T}}$ can be divided into blocks of covariance matrix between each instance as follows

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}^{1,1} & \cdots & \mathbf{C}^{1,L} \\ \vdots & \ddots & \vdots \\ \mathbf{C}^{L,1} & \cdots & \mathbf{C}^{L,L} \end{bmatrix} \tag{5}$$

$$\mathbf{C}^{i,j} = \begin{bmatrix} C^{i,j}_{s_1^i, s_1^j} & \cdots & C^{i,j}_{s_1^i, a^j_{N_{\text{action}}}} \\ \vdots & \ddots & \vdots \\ C^{i,j}_{a^i_{N_{\text{action}}}, s_1^j} & \cdots & C^{i,j}_{a^i_{N_{\text{action}}}, a^j_{N_{\text{action}}}} \end{bmatrix} \quad , \tag{6}$$

where $\mathbf{C}^{i,j}$ is a $(N_{\text{state}} + N_{\text{action}}) \times (N_{\text{state}} + N_{\text{action}})$ covariance matrix between instances I_i and I_j.

3.2 The (1+1)-CMA-ES

Our underlying algorithm is the (1+1)-CMA-ES [1], which is a variant of covariance matrix adaptation evolution strategy with elitism. Algorithm 1 provides the pseudo-code of the (1+1)-CMA-ES with active covariance matrix adaptation. The (1+1)-CMA-ES is a stochastic algorithm using the multivariate normal distribution which learns the covariance matrix during the optimization process. The adaptation of the covariance matrix is the essence in efficiently solving ill-conditioned and non-separable problems, which the IBP optimization forms as we described above. The main reason of the elitism, i.e., (1+1) selection, is to simplify the mechanism for

Algorithm 1 (1+1)-CMA-ES

Require: $n \in \mathbb{N}^+$, $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$, $\sigma^{(0)} \in \mathbb{R}^+$, $\mathbf{A}^{(0)} \in \mathbb{R}^{n\times n}$, $\mathbf{A}_{\text{inv}}^{(0)} \in \mathbb{R}^{n\times n}$

1: **Set:** $c_p = 1/12$, $c = 2/(n+2)$, $c_{\text{cov}}^+ = 2/(n^2+6)$, $c_{\text{cov}}^- = 0.4/(n^{1.6}+1)$, $d_\sigma = 1 + n/2$, $P_{\text{target}} = 2/11$, $P_{\text{thresh}} = 0.44$, $k = 5$
2: $\boldsymbol{x} \leftarrow \boldsymbol{x}^{(0)}$, $\sigma \leftarrow \sigma^{(0)}$, $\mathbf{A} \leftarrow \mathbf{A}^{(0)}$, $\mathbf{A}_{\text{inv}} \leftarrow \mathbf{A}_{\text{inv}}^{(0)}$
3: $P_{\text{succ}} \leftarrow 0$, $\boldsymbol{p} \leftarrow \mathbf{0}$
4: **while** not termination condition satisfied **do**
5: $\quad \boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
6: $\quad \boldsymbol{y} = \boldsymbol{x} + \sigma \mathbf{A}\boldsymbol{z}$
7: $\quad$ **if** $f(\boldsymbol{y}) \leqslant f(\boldsymbol{x})$ **then**
8: $\quad\quad \boldsymbol{x} \leftarrow \boldsymbol{y}$
9: $\quad\quad P_{\text{succ}} \leftarrow (1-c_P)P_{\text{succ}} + c_P$
10: $\quad\quad \boldsymbol{p} \leftarrow (1-c)\boldsymbol{p} + \sqrt{c(2-c)}\mathbf{A}\boldsymbol{z}$
11: $\quad\quad \boldsymbol{w} = \mathbf{A}_{\text{inv}}\boldsymbol{p}$
12: $\quad\quad a = \sqrt{1 - c_{\text{cov}}^+}$
13: $\quad\quad b = \dfrac{\sqrt{1-c_{\text{cov}}^+}}{\|\boldsymbol{w}\|^2}\left(\sqrt{1 + \dfrac{c_{\text{cov}}^+}{1-c_{\text{cov}}^+}\|\boldsymbol{w}\|^2} - 1\right)$
14: $\quad\quad \mathbf{A} \leftarrow a\mathbf{A} + b\,[\mathbf{A}\boldsymbol{w}]\,\boldsymbol{w}^{\mathrm{T}}$
15: $\quad\quad \mathbf{A}_{\text{inv}} \leftarrow \dfrac{1}{a}\mathbf{A}_{\text{inv}} - \dfrac{b}{a^2 + ab\|\boldsymbol{w}\|^2}\boldsymbol{w}\left[\boldsymbol{w}^{\mathrm{T}}\mathbf{A}_{\text{inv}}\right]$
16: $\quad$ **else**
17: $\quad\quad P_{\text{succ}} \leftarrow (1-c_P)\,P_{\text{succ}}$
18: $\quad$ **end if**
19: $\quad \sigma \leftarrow \sigma \exp\left(\dfrac{1}{d_\sigma}\dfrac{P_{\text{succ}} - P_{\text{target}}}{1 - P_{\text{target}}}\right)$
20: $\quad$ **if** $f(\boldsymbol{y}) > f(k\text{th-Order Ancestor})$ **then**
21: $\quad\quad a = \sqrt{1 + c_{\text{cov}}^-}$
22: $\quad\quad b = \dfrac{\sqrt{1+c_{\text{cov}}^-}}{\|\boldsymbol{z}\|^2}\left(\sqrt{1 - \dfrac{c_{\text{cov}}^-}{1+c_{\text{cov}}^-}\|\boldsymbol{z}\|^2} - 1\right)$
23: $\quad\quad \mathbf{A} \leftarrow a\mathbf{A} + b\,[\mathbf{A}\boldsymbol{w}]\,\boldsymbol{w}^{\mathrm{T}}$
24: $\quad\quad \mathbf{A}_{\text{inv}} \leftarrow \dfrac{1}{a}\mathbf{A}_{\text{inv}} - \dfrac{b}{a^2 + ab\|\boldsymbol{w}\|^2}\boldsymbol{w}\left[\boldsymbol{w}^{\mathrm{T}}\mathbf{A}_{\text{inv}}\right]$
25: $\quad$ **end if**
26: **end while**

adaptation of the number of instances. In the (1+1) strategy we can simply regard the parental solution, i.e., the mean vector of the multivariate normal distribution, as an RL agent.

3.3 IBP-CMA

The main idea of the IBP-CMA is described as follows. We try to keep one inactive instance as a guide to keep, increase, or decrease the number of instances. If all the instances are kept active during episodes, we add an instance to refine the policy. As we described in the previous section, random placement of a new instance will result in leaving it inactive. To address the issue, we *clone* an active instance. The policy after adding an instance in this way is then unchanged. This drastically improves the probability of making a new instance active, and the resulting trajectory improved. If more than one instance is kept inactive for a while, we drop all but one of the inactive instances. If one but only one instance is kept inactive, we try to make it active by cloning an active instance. Increasing or decreasing the number of instances, $L^{(t)}$, at each iteration means that the search space dimension for the CMA-ES, $n = L^{(t)}(N_{\text{state}} + N_{\text{action}})$, changes at each iteration. The dynamic parameters and strategy parameters of the CMA-ES will be updated accordingly.

3.3.1 Inactive Instance Detection. By definition instances that are not referred to during an episode are said to be *inactive*. Due to its stochastic nature, active instances in the parental solution can be inactive by chance and vice versa. To detect instances that are really inactive and determine if we would like to add, delete or activate instances, we introduce counters to v_{add}, v_{del}, and v_{act} that are initialized to zero at the beginning.

After line 6 in Algorithm 1, we evaluate $\boldsymbol{y}$, i.e., we perform an episode using the IBP represented by $\boldsymbol{y}$. Then, this information whether each instance in $\boldsymbol{y}$ is active (i.e., referred by the agent even once during the episode) or not is gathered during the episode.

(a) Check if each instance is active or not. Let $\bar{A}_l^y$ be `True` if the lth instance of $\boldsymbol{y}$ is active, `False` otherwise, and let $\bar{A}^y = \{\bar{A}_1, \ldots, \bar{A}_{L^{(t)}}\}$ be called the activity information of $\boldsymbol{y}$. The activity information of $\boldsymbol{x}$ is kept from the previous iteration, where an instance added or activated in the previous iteration is treated as inactive. The activity information $\bar{A}$ at tth iteration is $\bar{A}^y$ if $f(\boldsymbol{y}) \leqslant f(\boldsymbol{x})$, $\bar{A}^x$ otherwise.

(b) Update the number τ_l of iterations that the lth instance remains inactive as

$$\tau_l \leftarrow \begin{cases} \tau_l + 1 & \text{if } \bar{A}_l = \text{False} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

and let $\tau = \{\tau_1, \ldots, \tau_{L^{(t)}}\}$.

(c) Update the counters v_{add}, v_{del}, and v_{act} as follows

$$v_{\text{add}} \leftarrow \begin{cases} v_{\text{add}} + 1 & \text{if there is no False in } \bar{A} \\ 0 & \text{otherwise;} \end{cases} \tag{8}$$

$$v_{\text{del}} \leftarrow \begin{cases} v_{\text{del}} + 1 & \text{if there are more than one False in } \bar{A} \\ 0 & \text{otherwise;} \end{cases} \tag{9}$$

$$v_{\text{act}} \leftarrow \begin{cases} v_{\text{act}} + 1 & \text{if there is just one False in } \bar{A} \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Note that only one of them can be nonzero at the same time.

3.3.2 Adding an Instance. Based on the counters, we will add, delete, or activate instances. After line 25 of Algorithm 1, we will execute Section 3.3.2, Section 3.3.3, or Section 3.3.4. If none of the conditions are satisfied, we set $L^{(t+1)} = L^{(t)}$ and continue to the next iteration.

If $v_{\text{add}} \geqslant G_{\text{add}}$ is satisfied for a given constant G_{add}, the algorithm executes the following:

(a-1) Select an instance with probability

$$P(I_k) = \frac{c_k'}{\sum_{l=1}^{L^{(t)}} c_l'} \quad (k = 1, \ldots, L^{(t)}) \ , \tag{11}$$

where c_l is the number of steps that the lth instance of $\boldsymbol{x}$ is referred, and $c_l' = 0$ if $c_l = 1$ and c_l otherwise[1].

(a-2) Add an instance by cloning the selected instance I_ρ, and update dynamic parameters of the CMA-ES as

$$L^{(t+1)} \leftarrow L^{(t)} + 1 \tag{12}$$

$$\boldsymbol{x} \leftarrow \begin{bmatrix} \boldsymbol{x} & \boldsymbol{x}_\rho \end{bmatrix}, \ \mathbf{C} \leftarrow \begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}^{\rho,\rho} \end{bmatrix}, \ \boldsymbol{p} \leftarrow \begin{bmatrix} \boldsymbol{p} & \mathbf{0} \end{bmatrix} \ . \tag{13}$$

[1] If the number of steps per episode of the task is equal to 1, then $c_l' = c_l$.

Note that the resulting $\boldsymbol{x}$, $\boldsymbol{p}$ and $\mathbf{C}$ are of $L^{(t+1)}(N_{\text{state}} + N_{\text{action}})$ dimension.

(a-3) Update the strategy parameters depending on the dimension, n, as follows

$$n \leftarrow L^{(t+1)}(N_{\text{state}} + N_{\text{action}}) \tag{14}$$

$$c \leftarrow \tfrac{2}{n+2},\ c_{\text{cov}}^{+} \leftarrow \tfrac{2}{n^2+6},\ c_{\text{cov}}^{-} \leftarrow \tfrac{0.4}{n^{1.6}+1},\ d_\sigma \leftarrow 1 + \tfrac{n}{2}\ . \tag{15}$$

(a-4) Reset $v_{\text{add}} \leftarrow 0$.

3.3.3 Deleting Instances. Instead, if $v_{\text{del}} \geqslant G_{\text{del}}$ for a given constant G_{del}, the algorithm executes the following:

(b-1) Select the indices of instances to remove. Let Ξ be the set of indices of inactive instances with $\tau_l > 0$, and let $\bar{\ell} = \text{argmin}_{l\in\Xi}\,\tau_l$ be the index of inactive instances that has the least τ_l value. If there are more than one such indices, we choose one of them at random.

(b-2) Remove the inactive instances except $\bar{\ell}$, i.e., $\Xi\backslash\{\bar{\ell}\}$, by striking off the blocks of $\boldsymbol{x}$ and $\boldsymbol{p}$ and the columns and rows of $\mathbf{C}$ corresponding to the removed instances. Update $L^{(t+1)} = L^{(t)} - |\Xi| + 1$, where $|\Xi|$ is the cardinality of Ξ.

(b-3) Update the strategy parameters depending on the dimension, n, analogously to (a-3).

(b-4) Reset $v_{\text{del}} \leftarrow 0$ and $\tau_l \leftarrow 0$ for all l.

3.3.4 Activate an Instance. If $v_{\text{act}} \geqslant G_{\text{act}}$ for a given constant G_{act}, the algorithm executes the following:

(c-1) Get the index of the inactive instance ξ with $\tau_\xi > 0$. Such an index must exist only one.

(c-2) Select the index of an active instance with probability (11), and let ρ denotes it.

(c-3) Activate the inactive instance I_ξ by cloning I_ρ, which is done by updating the dynamic parameters of the CMA-ES as

$$\boldsymbol{x}_\xi \leftarrow \boldsymbol{x}_\rho \tag{16}$$

$$\mathbf{C}^{\xi,\xi} \leftarrow \mathbf{C}^{\rho,\rho},\ \mathbf{C}^{\xi,l} \leftarrow 0,\ \mathbf{C}^{l,\xi} \leftarrow \mathbf{0} \quad \forall l \in [\![1, L^{(t)}]\!] \setminus \{\xi\} \tag{17}$$

$$\boldsymbol{p}_\xi \leftarrow \mathbf{0}\ . \tag{18}$$

(c-4) Reset $v_{\text{act}} \leftarrow 0$ and $\tau_l \leftarrow 0$ for all l.

3.4 Static-IBP-CMA

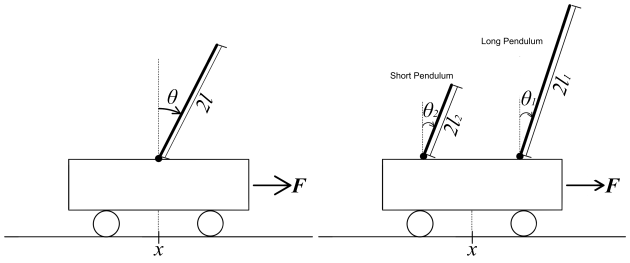
To evaluate the effect of the instance activation in Section 3.3.4, we test a variant of the IBP-CMA without the adaptation of the number of instances, called Static-IBP-CMA.

The difference from the IBP-CMA is listed below. The step (c) in the inactive instance detection (Section 3.3.1) and the mechanisms to add and delete instances (Section 3.3.2 and Section 3.3.3) are omitted. The activation mechanism (Section 3.3.4) is performed if there is an instance satisfying $\tau_l \geqslant G_{\text{inactive}}$ for a given G_{inactive}, where step (c-1) is replaced with the following:

(c-1) Get an index ξ of an inactive instance with $\tau_\xi \geqslant G_{\text{inactive}}$. If there are more than one inactive instances, we choose one at random.

4 PERFORMANCE EVALUATION

In this section, we apply the IBP-CMA to three benchmark tasks and see how effective the proposed mechanism is. We also compare the IBP-CMA with CMA-TWEANN [9], which is an RL method combining the $(\mu/\mu_{\text{w}},\lambda)$-CMA-ES and an ANN with evolving topology. A comparison with the existing IBP optimization algorithms [3, 7] has been omitted due to space limitation. In our preliminary study, we observed that the CMA-ES based IBP optimization tends to reach the same objective value about ten times faster than the existing real-coded genetic algorithm proposed in [7].



(a) Cart Pole **(b) PDIP**

Figure 2: Control Tasks

4.1 CMA-TWEANN

The CMA-TWEANN evolves a recurrent neural network and optimizes the connection weights with $(\mu/\mu_{\text{w}},\lambda)$-CMA-ES. The CMA-TWEANN adds a node and an edge to the network with constant probabilities $P(\alpha_n)$ and $P(\alpha_e)$, respectively, at each CMA-ES iteration. When a node or an edge is added, the connection weights are set to zero so that the policy is unchanged. This idea is similar to how we add an instance in the IBP-CMA. Importantly, the CMA-TWEANN lower bounds the step size σ by a given constant σ_{min} to prevent the CMA-ES from converging. For its details, refer to [9].

4.2 Benchmark tasks

4.2.1 Simple Task. Simple Task is an artificial task where the optimal number of instances and the optimal instance parameters are trivial. Our motivation here is to check how effective the instance activation mechanism of the IBP-CMA and the adaptation of the number of instances are.

Simple Task is not a control task. The state transition is deterministic independently of the action of the agent. At each time step $t = 1, \ldots, T$, the learning agent is located at $s_t \in \mathbb{R}^2$. The agent takes an action $a_t \in \mathbb{R}^2$ and receives an immediate cost $r_t = \|s_t + a_t\|$. The objective is then to minimize the cumulative cost in T steps, i.e., $R = \sum_{t=1}^{T} r_t$.

The optimal policy for this task is that the action corresponding to s_t is $-s_t$ for $t = 1, \ldots, T$, and the minimum cumulative cost is $R = 0$. The state vector at each time step is given by $s_t = [t-1, t-1]$ for $t = 1, \ldots, T$. In the IBP, the optimal policy is achieved only if $L \geqslant T$. If $L > T$, it is guaranteed that there are $L - T$ inactive instances. In the experiments described later, T is set to $\{2, 10\}$.

4.2.2 Cart Pole. Cart Pole is a typical control task of nonholonomic systems, which is often used as RL task. In this task, the optimal number of instances is unknown.

Cart Pole consists of a horizontal rail, a cart that moves on the rail, and a pendulum that rotates around the fulcrum on the side of the cart, as shown in Figure 2a. We assume that the friction

between the pendulum and the fulcrum and between the cart and the rail are negligible. Let M and m[kg] be the mass of the cart and the pendulum, respectively. We assume that the mass distributes uniformly in each material. In this task, the observable states are the position x and the velocity $\dot{x}$ of the cart and the angle θ and the angular velocity $\dot{\theta}$ of the pendulum, and the action is the driving force F[N] of the cart. The state transition is defined by

$$\ddot{x} = \frac{mg\sin\theta\cos\theta - \frac{4}{3}(F + ml\dot{\theta}^2\sin\theta)}{m\cos^2\theta - \frac{4}{3}(M+m)} \ , \tag{19}$$

$$\ddot{\theta} = 3(g\sin\theta - \ddot{x}\cos\theta)/(4l) \ . \tag{20}$$

Here the gravitational acceleration is $g = 9.8$ [m/s^2], and we let $M = 1$[kg], $m = 0.1$[kg], and $l = 0.5$[m].

In this paper, let one episode of this task is $T = 500$ steps, and one step is 0.01 seconds, i.e., the agent observes the state and changes the action every 0.01 seconds according to itself policy. The initial state of this task is $(x, \dot{x}, \theta, \dot{\theta}) = (0, 0, \pi, 0)$. The driving force F takes a value in $[-10, 10]$, and it is limited to 10 and -10 by the environment if the agent try to take a greater value in plus and minus, respectively.

The immediate reward r_t in each time step $t = 1, \ldots, T$ is given with

$$r_t = \cos\theta + r_{\mathrm{PN}t} \ . \tag{21}$$

where $r_{\mathrm{PN}t}$ defines the penalty to prevent the cart from moving far away from the origin,

$$r_{\mathrm{PN}t} = \begin{cases} 0 & \text{if } |x| < 5.0 \\ -1000 & \text{otherwise} \end{cases} \ . \tag{22}$$

The objective value to be minimized is the average of the immediate reward r_t,

$$R = \frac{1}{T}\sum_{i=1}^{T} r_t \ . \tag{23}$$

4.2.3 Parallel-type Double Inverted Pendulum (PDIP). PDIP is another typical control task of nonholonomic systems. The optimal number of instances is unknown for this task as well as the Cart Pole task. The only difference from the Cart Pole task is that the cart has two pendulums with different lengths and different mass, as shown in Figure 2b. Let M[kg], m_1[kg], and m_2[kg] be the mass of the cart, the long-pendulum, and the short-pendulum, respectively, and assume that the mass distribute uniformly in each material.

In this task, the observable states are the position x and the velocity $\dot{x}$ of the cart and the angle θ_1, θ_2 and the angular velocity $\dot{\theta_1}$, $\dot{\theta_2}$ of each pendulum, and the action is the driving force F of the cart. The state transition is defined as

$$\ddot{x} = \frac{\sum_{i=1}^{2} m_i g\sin\theta_i\cos\theta_i - \frac{4}{3}(F + \sum_{i=1}^{2} l_i m_i \dot{\theta}_i^2\sin\theta_i)}{\sum_{i=1}^{2} m_i\cos^2\theta_i - \frac{4}{3}(M + \sum_{i=1}^{2} m_i)} \ , \tag{24}$$

$$\ddot{\theta}_i = 3(g\sin\theta_i - \ddot{x}\cos\theta_i)/(4l_i) \ , \tag{25}$$

where we set $M = 1$[kg], $m_1 = 0.2$[kg], $m_2 = 0.1$[kg], $l_1 = 1.0$[m], $l_2 = 0.5$[m] in this paper. Analogously to the Cart Pole task, we set $T = 500$, and one step is 0.01 seconds. The initial state of this task is $(x, \dot{x}, \theta_1, \dot{\theta_1}, \theta_2, \dot{\theta_2}) = (0, 0, \pi, 0, \pi, 0)$. The driving force F takes a value is limited in $[-10, 10]$ in the same way as in Cart Pole.

Table 1: Parameter settings of each algorithm

(1+1)-CMA-ES:	$L = \begin{cases} T & \text{(Simple Task)} \\ 20 & \text{(Cart Pole, PDIP)} \end{cases}$
IBP-CMA:	$L^{(0)} = 2$
	$G_{\mathrm{act}} = \{1, 10\}$
	$G_{\mathrm{add}} = \{5, 20\}$
	$G_{\mathrm{del}} = \{5, 20\}$
Static-IBP-CMA:	$G_{\mathrm{inactive}} = \{1, 10\}$
	$L = \begin{cases} T & \text{(Simple Task)} \\ 20 & \text{(Cart Pole, PDIP)} \end{cases}$
CMA-TWEANN:	$P(\alpha_n) = \{0.005, 0.01\}$
	$P(\alpha_e) = \{0.1, 0.2\}$

The immediate reward r_t in each time step $t = 1, \ldots, T$ is given with

$$r_t = \cos\theta_1 + \cos\theta_2 + r_{\mathrm{PN}t} \tag{26}$$

$$r_{\mathrm{PN}t} = \begin{cases} 0 & \text{if } |x| < 2.4 \\ -1000 & \text{otherwise} \end{cases} \tag{27}$$

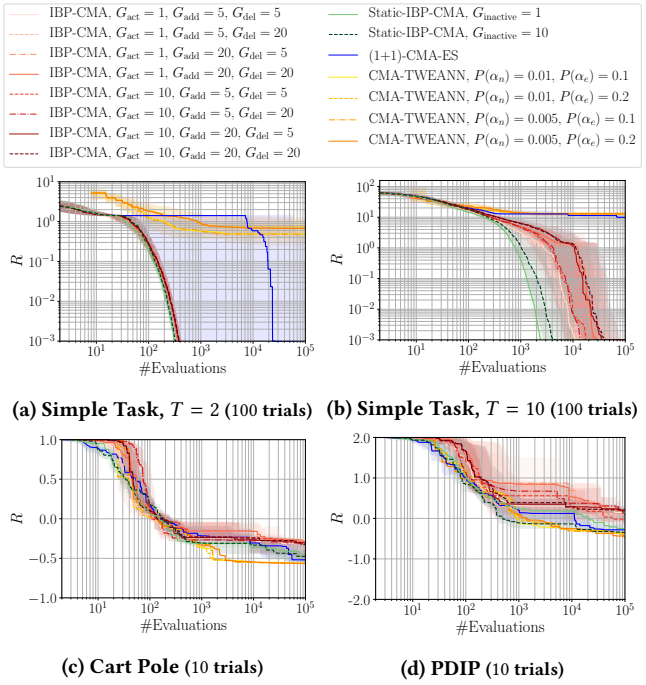
and the objective value to be minimized is the average R of the immediate reward r_t, computed as Eq. (23).

4.3 Results and Discussion

In this section, we show results of applying the (1+1)-CMA-ES, the IBP-CMA, the Static-IBP-CMA and the CMA-TWEANN to the benchmark tasks described in Section 4.2, respectively. Parameter settings for each algorithm are displayed in Table 1. For applying the (1+1)-CMA-ES and the Static-IBP-CMA to the Cart Pole and the PDIP, respectively, the number of instances is $L = 20$ that is the number exhibits the best performance in preliminary experiments for $L \in \{2, 5, 10, 20\}$. In the Simple Task, the number of instances is $L = T$ that is the optimum number for this task. For the CMA-TWEANN, we use the parameter settings in [9]. A single trial is terminated after the maximum number of function evaluations 10^5 have passed, and the algorithm restarts multiple times until it spends 10^5 function evaluations, where a restart is performed if the search distribution is regarded as converging. In the Simple Task, a single trial is terminated if the total cost R (R-value) reaches the target $R = 10^{-3}$ even if the number of function evaluations does not pass the maximum. The number of trials is 100 for the Simple Task, and it is 10 for the Cart Pole and the PDIP.

Figure 3 shows the median, the lower-quartile and the upper-quartile of the R-value over 100/10 trials in each task. Figure 4 shows the number of instances that the IBP-CMA obtains the best R-value at each number of function evaluations among all trials. Figure 5 shows the behaviors of the adaptation of the number of instances for three trials in the IBP-CMA (for $G_{\mathrm{act}} = 10, G_{\mathrm{add}} = 5, G_{\mathrm{del}} = 5$).

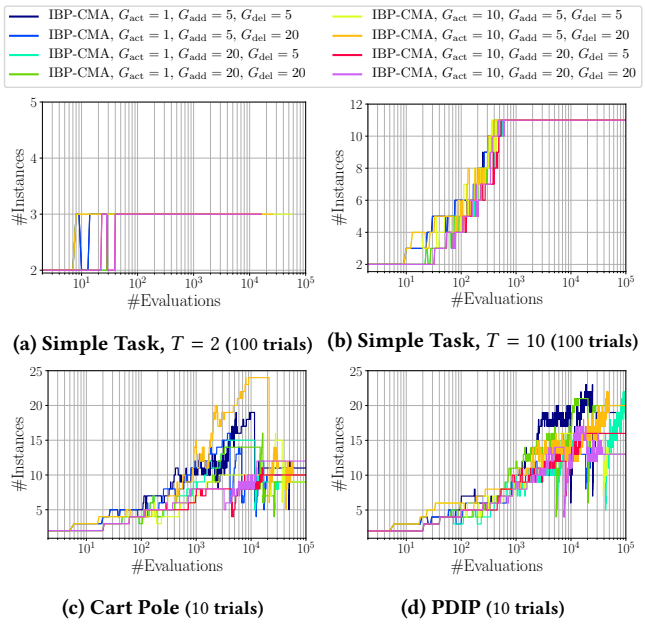
4.3.1 Simple Task. Figures 3a ($T = 2$) and 3b ($T = 10$) show that the R-values of the CMA-TWEANN never reached 10^{-3}. Note that the action of the agent associates to each state s_i needs to be sufficiently close to its optimal value $-s_i$ to reach the target in this task. It requires a fine tuning of the policy parameters. In the CMA-TWEANN, an edge or a node is added with probability



(a) Simple Task, $T = 2$ (100 trials) **(b) Simple Task, $T = 10$ (100 trials)**

(c) Cart Pole (10 trials) **(d) PDIP (10 trials)**

Figure 3: The median and interquartile range of best objective values for each setting.



(a) Simple Task, $T = 2$ (100 trials) **(b) Simple Task, $T = 10$ (100 trials)**

(c) Cart Pole (10 trials) **(d) PDIP (10 trials)**

Figure 4: The number of instances in the trial that the IBP-CMA obtained the best performance.

$P(\alpha_e)$ or $P(\alpha_n)$, respectively, and the search space dimension is incremented by 2 or 1. Therefore, the search space dimension is increased each iteration by $2P(\alpha_n)+P(\alpha_e)$ in expectation. Moreover, the standard deviation of the sampling distribution is lower bounded in the CMA-TWEANN. Both makes difficult for the CMA-ES to perform exploitation. Therefore, the CMA-TWEANN could not take advantage of the effectiveness of the CMA-ES.

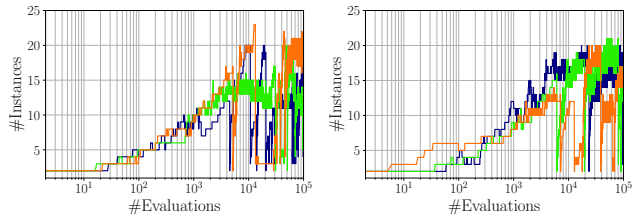
For $T = 2$, the median of the R-values of the (1+1)-CMA-ES reached the target 10^{-3}, however, the interquartile range of this algorithm spread widely. Moreover, for $T = 10$, the R-values of the (1+1)-CMA-ES never reaches 10^{-3}. In contrast, the R-values of the IBP-CMA and the Static-IBP-CMA reached 10^{-3} in all trials both for $T = 2, 10$. The (1+1)-CMA-ES failed to activate all the instances as expected, whereas the proposed activation mechanism in the IBP-CMA and the Static-IBP-CMA worked effectively.

Figures 4a and 4b show that the IBP-CMA obtains $L = T + 1$ instances in the end in all settings. Since the optimal number of instances for this task is $L = T$ and L is adapted in the IBP-CMA so as to keep just one instance inactive, it is the desired result. Figure 3b shows that the smaller G_{act} and G_{add} of the IBP-CMA and $G_{inactive}$ of the Static-IBP-CMA are, the better performance was exhibited. It may be true for $G_{inactive}$ in general, i.e., fast activation of instances results in better performance. However, too fast increase or decrease of the number of instances may disturb the optimization of the policy parameters by the CMA-ES in general control tasks. We further investigate the effect in the sequel.

4.3.2 Cart Pole and PDIP. As we mentioned in the previous section, the CMA-TWEANN typically can not perform exploitation of the connection weights and it behaves as if it is a random search. Nonetheless, the CMA-TWEANN reached the best function values in Figures 3c (the Cart Pole) and 3d (the PDIP). Thanks to its redundancy, it is not difficult (even for a random search) to find better connection weights than the current best if the number of edges are not too large. Note however that displayed are the best-so-far function values. We observed in all cases that the CMA-TWEANN sometimes improved the best-so-far value but generally kept sampling worse function values, and started worsen the function values in the end. On the other hand, the IBP-CMA typically tended to converge towards the best solution.

Comparing the IBP-CMA and the Static-IBP-CMA, the R-value of the Static-IBP-CMA tended to decrease faster. The difference was remarkable in the PDIP. The reason may be described as follows. The Static-IBP-CMA runs with $L = 20$, which is considered a reasonable choice based on our preliminary study, whereas the IBP-CMA started with $L = 2$ in each restart. Hence, the number of instances needed to increase. However, in Figure 5, we observed that the IBP-CMA spent more than 10^3 function evaluations till it obtained more than 15 instances. Therefore, we compromise on the speed of the optimization for removing the need for tuning of the number of instances. One should set the initial L to a reasonable value of the number of instance if known. Moreover, the performance is expected to improve if we take over the number of instances obtained in the previous search when the IBP-CMA performs a restart.

Finally, we discuss the hyper-parameter setting in the IBP-CMA. First of all, the differences in the best R-values between different parameter values for G_{add}, G_{del} and G_{act} were negligible for the Cart Pole task, whereas slight differences were observed for the PDIP task. A small G_{add} is desired to increase the number of instances



(a) Cart Pole **(b) PDIP**

Figure 5: Behavior of adaptation of the number of instances in the IBP-CMA, $G_{\text{act}} = 10, G_{\text{add}} = 5, G_{\text{del}} = 5$ (3 trials).

quickly since we started with $L = 2$. On the other hand, the effect of G_{del} did not appear. For the PDIP tasks, $G_{\text{act}} = 10$ demonstrated a better R-value than $G_{\text{act}} = 1$. A frequent activation of instances results in resetting the component of the covariance matrix, and may lead to a slow convergence. Looking at the inter-quartile ranges of R-values in Figures 3c and 3d, however, the difference was not statistically significant.

5 CONCLUSION

In this paper, we investigated the use of the instance-based policy (IBP) for episodic control tasks of nonholonomic systems. Motivated by the fact that the IBP potentially well suits for stabilization tasks of nonholonomic systems due to its bang-bang type control, we developed an optimization algorithm for the IBP that can address the difficulties of the IBP optimization, i.e., variable dependencies, existence of inactive instances, and tuning of the number of instances. The resulting algorithm, IBP-CMA, is based on the CMA-ES with elitist selection along with the mechanism to adapt the number of instances and activate instances.

The proposed algorithm have been applied to an artificial non-control task and two control tasks, and compared with an existing learning algorithm using an ANN, namely the CMA-TWEANN. Both the advantages and the disadvantages of the IBP-CMA over the CMA-TWEANN have been revealed. Moreover, the hyper-parameters introduced in this work, G_{act}, G_{add}, and G_{del}, were investigated. We conclude that the IBP-CMA is robust against the choice of these values. Our current choice would be $G_{\text{act}} = 1$, $G_{\text{add}} = G_{\text{del}} = 5$, independently of tasks. However, the current defaults are not yet widely investigated. A future work must investigate the sensitivity of the default hyper-parameter values for different control tasks.

A possible room for improvement of the IBP-CMA have been spotted in the experiments. An important future work would be to develop the restart strategy for the IBP-CMA. As we mentioned in the discussion of the experimental results, we restarted the IBP-CMA if the search distribution is considered converged or diverged, by resetting all the internal parameters including the number of instances, L. However, the number of instances achieved in the preceding restart is expected useful information. Restarting without resetting the number of instances will be rather useful. A further investigation is required in this direction.

As we mentioned in the paper, one of the benefits of the IBP representation is that the effect of change of the policy parameter is localized, whereas a single parameter change affects the overall policy in usual policy representations such as an ANN. It allows us to decompose a task into subtasks, adapt the IBP for each subtask, combine them easily later. The effect of decomposition is not yet studied. It is another direction of our future work.

REFERENCES

[1] Dirk V Arnold and Nikolaus Hansen. 2010. Active Covariance Matrix Adaptation for the (1+1)-CMA-ES. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. ACM, 385–392.
[2] Konstantinos Chatzilygeroudis, Roberto Rama, Rituraj Kaushik, Dorian Goepp, Vassilis Vassiliades, and Jean-Baptiste Mouret. 2017. Black-box Data-efficient Policy Search for Robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 51–58.
[3] Kokolo Ikeda. 2005. Exemplar-based Direct Policy Search with Evolutionary Optimization. In *2005 IEEE Congress on Evolutionary Computation*, Vol. 3. IEEE, 2357–2364.
[4] Kokolo Ikeda, Shigenobu Kobayashi, and Hajime Kita. 2010. Exemplar-Based Policy with Selectable Strategies and its Optimization Using GA. *Transactions of the Japanese Society for Artificial Intelligence* 25, 2 (2010), 351–362. (in Japanese).
[5] Kokolo Ikeda and Isao Ono. 2013. Instance Based Policy Representation and Its Evolutionary Optimization(-Special Issue- New Trends of Population-Based Machine Learning). *SYSTEMS, CONTROL AND INFORMATION* 57, 10 (2013), 415–420. (in Japanese).
[6] Jan Hendrik Metzen, Mark Edgington, Yohannes Kassahun, and Frank Kirchner. 2008. Analysis of an Evolutionary Reinforcement Learning Method in a Multiagent Domain. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, 291–298.
[7] A. Miyamae, Jun Sakuma, Isao Ono, and Shigenobu Kobayashi. 2009. Optimization of Instance-Based Policy Based on Real-Coded Genetic Algorithms. In *IEEE Conference on Soft Computing in Industrial Applications, SMCia'08*. IEEE, 338–343.
[8] David E Moriarty, Alan C Schultz, and John J Grefenstette. 1999. Evolutionary Algorithms for Reinforcement Learning. *J. Artif. Intell. Res.(JAIR)* 11 (1999), 241–276.
[9] Hirotaka Moriguchi and Shinichi Honiden. 2012. CMA-TWEANN: Efficient Optimization of Neural Networks via Self-adaptation and Seamless Augmentation. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. ACM, 903–910.
[10] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. 2017. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. (2017). arXiv:1703.03864
[11] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. 2009. A Hypercube-based Encoding for Evolving Large-scale Neural Networks. *Artificial Life* 15, 2 (2009), 185–212.
[12] Kenneth O Stanley and Risto Miikkulainen. 2002. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation* 10, 2 (2002), 99–127.
[13] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. Vol. 1. MIT press Cambridge.
[14] Chikao Tsuchiya, Yusuke Shiokawa, Kokolo Ikeda, Jun Sakuma, Isao Ono, and Shigenobu Kobayashi. 2006. SLIP: A Sophisticated Learner for Instance-based Policy using Hybrid GA. *Transactions of the Society of Instrument and Control Engineers* 42, 12 (2006), 1344–1352. (in Japanese).
[15] Daniel Urieli, Patrick MacAlpine, Shivaram Kalyanakrishnan, Yinon Bentor, and Peter Stone. 2011. On Optimizing Interdependent Skills: A Case Study in Simulated 3D Humanoid Robot Soccer. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 2. International Foundation for Autonomous Agents and Multiagent Systems, 769–776.
[16] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3-4 (1992), 279–292.
[17] Shimon Whiteson and Peter Stone. 2006. Evolutionary Function Approximation for Reinforcement Learning. *Journal of Machine Learning Research* 7, May (2006), 877–917.