2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 1-5, 2023. Detroit, USA

C^2: Co-design of Robots via Concurrent-Network Coupling Online and Offline Reinforcement Learning

Ci Chen[1], Pingyu Xiang[1], Haojian Lu[1], Yue Wang[1], Rong Xiong[1]

***Abstract*— With the increasing computing power, using data-driven approaches to co-design a robot's morphology and controller has become a promising way. However, most existing data-driven methods require training the controller for each morphology to calculate fitness, which is time-consuming. In contrast, the dual-network framework utilizes data collected by individual networks under a specific morphology to train a population network that provides a surrogate function for morphology optimization. This approach replaces the traditional evaluation of a diverse set of candidates, thereby speeding up the training. Despite considerable results, the online training of both networks impedes their performance. To address this issue, we propose a concurrent network framework that combines online and offline reinforcement learning (RL) methods. By leveraging the behavior cloning term in a flexible manner, we achieve an effective combination of both networks. We conducted multiple sets of comparative experiments in the simulator and found that the proposed method effectively addresses issues present in the dual-network framework, leading to overall algorithmic performance improvement. Furthermore, we validated the algorithm on a real robot, demonstrating its feasibility in a practical application.**

I. Introduction

A robot's performance depends on its mechanical structure as well as its control proficiency, which are inherently interrelated. While robot locomotion control has achieved remarkable success, the design of a robot's structure still heavily relies on the experience of engineers. Recently, analytic dynamics model-based approaches [1]–[4] have emerged to address the co-design problem of robots. However, such methods require the establishment of dynamic models and the implementation of equality or inequality constraints, which necessitates a significant amount of tedious human engineering and expert knowledge.

With the increased computing power, numerous data-driven algorithms [5]–[9] have emerged to address co-design problems. Most of these algorithms [5]–[8] adopt bi-level approaches. The lower level trains policies under specific morphology candidates from scratch to calculate fitness, while the upper level selects a new morphology based on fitness. Such processes require significant time investments. Improving optimization efficiency is a worthwhile pursuit. The dual-network architecture proposed by [9] offers a solution that learns a surrogate function conditioned on morphology parameters to evaluate candidate fitness, avoiding the need to train each candidate from scratch. Specifically, it includes an individual network and a population network. The former interacts with the environment under a specific morphology, while the latter integrates interactive data from various morphologies to provide goals for morphology optimization.

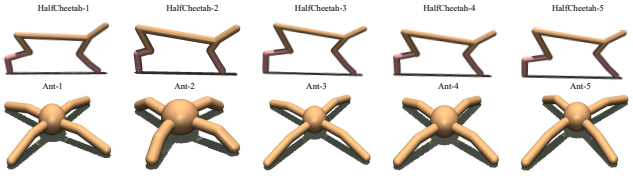

Fig. 1. Agents under different morphological parameters, the upper row is HalfCheetah, and the lower row is Ant.

While the dual-network architecture has achieved notable success, it has several severe limitations. Firstly, as the population network is updated without direct interaction with the environment, *exploration errors* may arise in such an offline setting, leading to an inaccurate estimation of fitness during morphology optimization. Secondly, similar to the offline-to-online setting [10]–[12], the population network's parameters can be used to initialize the individual network. But such procedures lead to performance collapses caused by sudden state-action *distribution shifts*. To address these issues, we propose the concurrent-network architecture, which emphasizes the effective combination of offline and online networks. Specifically, we use a policy-constraint method to train the population network offline, which helps alleviate the exploration error and ensures the general policy learned by the population network is more reliable. Besides, we use an adaptive behavior cloning term to train the individual network online, which mitigates the influence of distribution shifts in the early stages of training and ensures the agent's exploration in the later stages. To verify the effectiveness of our proposed methods, we perform two simulation tasks and one physical task. In summary, our contributions are as follows:

- Aiming at the task that can be modeled as bi-level optimization problems, such as the co-design of robots, we introduced the concurrent network that integrates individual network and population network under the purview of Bayesian optimization. The individual network is trained online to solve the lower-level optimization task, while the population network is trained offline to provide the objective of upper-level optimization. By combining the offline and online training approaches, we are able to leverage the data in the most efficient manner.

This work was supported in part by the National Key R&D Program of China under Grant 2021ZD0114500. Corresponding author: Yue Wang (wangyue@iipc.zju.edu.cn)

[1]Ci Chen, Pingyu Xiang, Haojian Lu, Yue Wang, Rong Xiong are with the State Key Laboratory of Industrial Control and Technology, Zhejiang University, Hangzhou 310027, China.

2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-9190-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IROS55552.2023.10341983

- We conduct simulation experiments on two typical legged robot locomotion tasks to evaluate the proposed method. The results demonstrate that the method can significantly reduce exploration errors and mitigate state-action distribution shifts, leading to noticeable improvements in optimization performance.
- We construct a detachable four-legged robot and conduct hardware experiments to verify the effectiveness of our proposed method.

II. Related Works

A. Co-design for robots

The co-optimization of morphology and policy can be classified into two categories: analytic dynamics model-based approaches and data-driven approaches. In the first category of approaches [1]–[4], [1] pointed out that the design and motion parameters of robots need to satisfy various equality and inequality constraints, which can form an implicitly-defined manifold. It applies the implicit function theorem to derive the relationships among the design and motion parameters. In [4], the design and control parameters are selected by a genetic optimizer and used to establish the dynamic and friction models. The trajectory optimization is then performed, and the final costs serve as the optimized objective of the genetic algorithm. However, among these approaches, the establishment of motion equations and the design of equality and inequality constraints require careful human engineering, and the constraints may vary for different types of robots. In the data-driven approaches, the control policy and morphology parameters are learned in a trial-and-error manner, which makes them independent of the robot's dynamics. Based on whether the topology changes, these approaches can be divided into two categories: morphology-changing and unchanging approaches. Morphology-changing methods [6]–[8], [13] change the robot's topology, whereas unchanging methods [5], [9], [14]–[16] do not. A representative example of a morphology-changing approach is [8], which collects experience via a distribution and asynchronous manner and uses the learned average final rewards as the fitness for tournament-based evolution. For unchanging morphology methods, [5] maintain a distribution over designs and use the reinforcement learning algorithm to optimize a control policy to maximize the expected reward over the design distribution. Furthermore, several works have used differential simulators to perform co-design, such as [14], [15] for soft robot tasks and [16] for contact-rich manipulation tasks. The topology-changing methods are not feasible to deploy in the physical environment because the optimized robot structures are often asymmetric, and the motor position layout may not be reasonable. In this paper, we focus on co-design unchanging topology problems, propose a novel framework, and construct a four-legged robot to perform hardware validation.

B. Offline reinforcement learning

The goal of offline reinforcement learning (RL) is to derive better strategies solely from static datasets. The main challenge of offline RL is exploration error, which arises from learned policies that may produce out-of-distribution actions [17], [18]. To reduce exploration error, previous works can be mainly divided into three categories. The first category is policy-constraint methods, which aim to restrict the learned policy to only access data similar to interaction tuples in the datasets. This category can be further divided into explicit [17], [19], [20], implicit [10], [21], [22], and importance sampling [23]–[25] methods. The second category is conservative methods [26]–[28], with CQL [26] and Fisher-BRC [27] being state-of-the-art approaches in this category. All of the aforementioned approaches are model-free offline reinforcement learning methods. In addition, there is a type of model-based method, where the basic principle is to generate data through the model and penalize generated data that deviates from the dataset by measuring the uncertainty of the model prediction. Representative methods in this category include [29] and [30].

III. Preliminaries

A. Problem formulation

Co-design for robots can be formulated as a bi-level optimization problem:

$$\begin{aligned} &\max_{\xi\in\Xi} F(\pi^*(\xi),\xi) \\ s.t.\quad &\pi^*(\xi)=\arg\max_{\pi\in\Pi} J(\pi,\xi) \end{aligned} \tag{1}$$

where π is the control policy, ξ is the morphology parameters. $F(\cdot)$ and $J(\cdot)$ are objective functions of the upper and lower layers, respectively. The lower-level optimization is performed first to obtain optimized policies under pre-defined morphology parameters. Then, the upper-level optimization is performed to obtain the optimized morphology among the morphology parameter search space based on the fitness acquired by the optimized policies. It is worth noting that the bi-level framework can be utilized not only for addressing the co-design problem of robots but also for adapting the parameter distribution of the simulator to tackle the sim-to-real problem, as demonstrated in [31] and [32]. The interaction between the robot and its environment can be modeled as an extension of the Markov Decision Process (MDP) conditioned by ξ. This can be represented by the tuple $\langle S, A, P, R, \gamma\rangle$, where S and A represent the state and action space, P and R denote the dynamics and reward function, and $\gamma \in [0, 1)$ indicates the discount factor. At each time step t, an agent selects an action $a_t \in A$ under the state $s_t \in S$ according to the policy $\pi(\cdot|s_t, \xi)$ and receives a reward $r_t = r(s_t, a_t, \xi)$. The environment then transitions to a new state s_{t+1} following the transition model $p(s_{t+1}|s_t, a_t, \xi)$. The objective of lower-level optimization is to optimize the control policy to maximize the expectation of the accumulative rewards conditioned on a specific morphology parameter $\bar{\xi}$. The evaluation is depicted as follows:

$$J(\pi,\bar{\xi}) = \left[\sum_{t=0}^{H}\gamma^t r(s_t,a_t,\bar{\xi})\middle| a_t\sim\pi(\cdot|s_t,\bar{\xi})\right] \tag{2}$$

where H refers to the horizon. In (2), ξ is fixed and π is to be optimized. Similarly, the objective of the upper-level is to

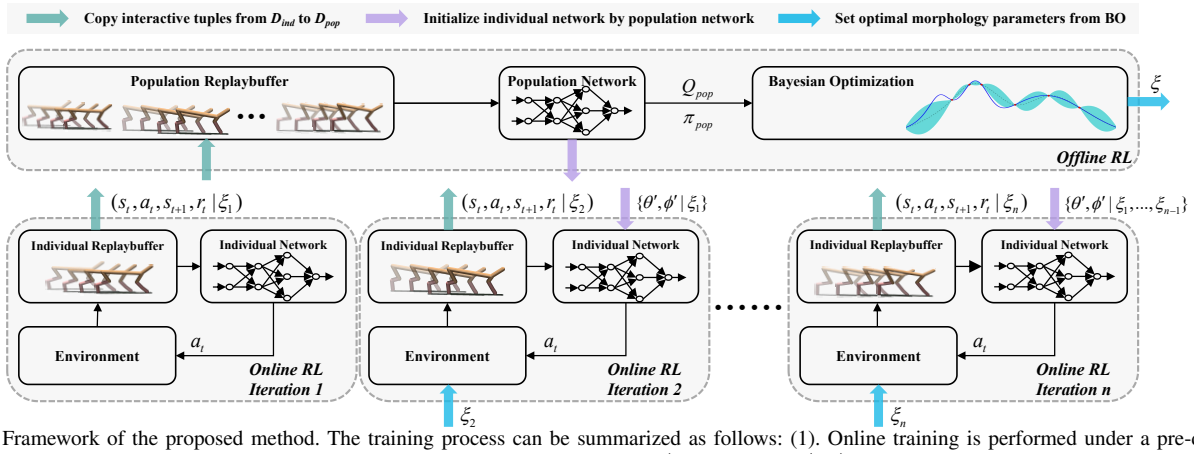

Fig. 2. Framework of the proposed method. The training process can be summarized as follows: (1). Online training is performed under a pre-designed morphology ξ_1, as illustrated in ***Online RL Iteration 1***. (2). The interaction tuples $(s_t, a_t, s_{t+1}, r_t | \xi_1)$ obtained during the online training are used to train the population network to provide a surrogate function for Bayesian optimization. The optimized morphology ξ_2 is then obtained through Bayesian optimization, as depicted in ***Offline RL***. (3). ξ_2 obtained by (2) is utilized as the new morphology parameters, and the population network's parameters are used to initialize the individual network's parameters in ***Online RL Iteration 2***. Online training is then performed again under ξ_2. (4). The interaction tuples $(s_t, a_t, s_{t+1}, r_t | \xi_2)$ obtained during this iteration are also stored in the replay buffer D_{pop}. Then, the interaction tuples obtained from both ξ_1 and ξ_2 are used to train the population network to obtain the new configuration ξ_3, and the process is repeated.

find the optimized morphology parameter ξ^* that maximizes the expectation of the cumulative rewards given the optimal policy $\pi^*(\xi)$. The evaluation function is described as follows:

$$F(\pi^*(\xi), \xi) = \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t, \xi) \middle| a_t \sim \pi^*(\cdot|s_t, \xi)\right] \quad (3)$$

In (3), π is fixed, and ξ is the parameter to be optimized.

B. The dual-network framework

Obtaining $\pi^*(\xi)$ in equation (3) is time-consuming. To tackle this issue, [9] introduced the dual-network framework. This framework employs two identical networks: the individual network and the population network. The individual network interacts with the environment, and the interaction tuples are stored in the replay buffers D_{ind} and D_{pop}. During each iteration of morphology optimization, D_{ind} is cleared, and D_{pop} retains all interaction tuples from various morphologies. This enables the population network to generalize better across different morphologies. When presented with new morphology parameters, the population network estimates Q-values as the surrogate function for morphology optimization. Specifically, the Q-value of the initial state s_0 is utilized. Thus, the objective of morphology optimization, as expressed in equation (3), can be rephrased as follows:

$$\xi^* \approx \arg\max_{\xi \in \Xi} \mathbb{E}[Q_{pop}(s_0, a_0, \xi) | a_0 \sim \pi_{pop}(\cdot|s_0, \xi)] \quad (4)$$

In this way, the problem of finding optimal morphological parameters can be transformed into training networks that can predict the Q-values of different morphology for given initial states. It is worth noting that using a single network is infeasible to achieve both generalizations to provide policies for upper-level optimization and optimality under a specific morphology to ensure the quality of interaction tuples.

IV. Method

A. Exploration error and distribution shift

In the dual-network framework [9], it is necessary to perform value estimation when training the population network, as shown below:

$$\begin{aligned} Q_{pop}(s_t, a_t, \xi) &\leftarrow r_t + \gamma Q_{pop}(s_{t+1}, a_{t+1}, \xi) \\ a_{t+1} &\sim \pi_{pop}(\cdot|s_{t+1}, \xi) \end{aligned} \quad (5)$$

The policy function π_{pop} obtains a_{t+1} based on s_{t+1} and ξ, which come from interaction tuple collected by the individual network. Subsequently, the Q function Q_{pop} provides $Q_{pop}(s_{t+1}, a_{t+1}, \xi)$, and the target Q-value $Q_{pop}(s_t, a_t, \xi)$ is calculated. If the population network interacts with the environment itself, when Q_{pop} overestimates the state-action pair, π_{pop} may collect data in the uncertainty region, and the erroneous value estimate can be corrected. However, since the population network is updated by data from the static dataset, a_{t+1} selected by π_{pop} may be suboptimal, and the distribution of (s_{t+1}, a_{t+1}, ξ) may differ significantly from that of the replay buffer, leading to an incorrect estimation of the target Q-value and the failure of the Q-learning-based algorithm. This process is known as *exploration error*. Therefore, the Q function Q_{pop} cannot serve as a reliable surrogate function for morphology optimization, which is critical for the dual-network architecture proposed in [9].

In the co-design task with unchanged topology, the robot morphology parameters remain in a feasible region, allowing the population network to provide a pre-trained feasible policy for the individual network, which can be considered an offline-to-online problem [10]–[12]. Nevertheless, as the morphology parameters are modified during optimization, the individual network is likely to encounter unfamiliar state-action regions. This results in a sudden *distribution shift* between offline and online data, which can lead to inaccurate Q-value estimates [10]. As a result, the policy may be updated in an arbitrary direction, which could compromise the well-trained initial policy from the population network.

B. Policy-constraint method for offline RL

To handle the *exploration error* problem, we utilize a policy-constraint method TD3BC [20] to train the population network. Although simple, it can still achieve or exceed other complex state-of-the-art offline RL methods

[26], [27]. Specifically, when calculating the Actor's loss function, we add the behavior cloning term to promote the actions obtained by the policy to approach the actions in the dataset, thereby reducing the error estimation of the Q-value. The Actor's loss function is as follows (In practical implementation, we concatenate ξ and state s_t together. For notational clarity, we omit ξ in the remainder of the paper):

$$J_\pi(\phi') = -\mathbb{E}_{(s_t,a_t)\sim D_{pop}}[\frac{Q_{\theta'}(s_t,\pi_{\phi'}(s_t))}{\frac{1}{N}\sum_{(s_i,a_i)}|Q_{\theta'}(s_i,a_i)|} - \alpha(\pi_{\phi'}(s_t)-a_t)^2] \tag{6}$$

where θ' and ϕ' represent the network parameters of Critic and Actor of the population network, respectively. To balance the values of the two terms in (6), the Q-value is normalized, and α is added to control the weights of the behavior cloning term, we use $\alpha = 0.4$ in our experiments.

C. Adaptive behavior cloning term for offline-to-online

When training the population (offline) network, we use α to balance the trade-off between the reinforcement learning (RL) target and the behavior cloning term. This approach has inspired us to dynamically adjust α during individual network training. Intuitively, the α should be high when the policy inherits from the population network is already near-optimal and α should be low when the policy has to be significantly improved. To achieve this, we employ a control mechanism similar to a proportional-derivative (PD) controller [12]. To separate from that of the population network, we assign the weight β to the behavior cloning term of the individual network, and the Actor's loss function is presented below:

$$J_\pi(\phi) = -\mathbb{E}_{(s_t,a_t)\sim D_{ind}}[\frac{Q_\theta(s_t,\pi_\phi(s_t))}{\frac{1}{N}\sum_{(s_i,a_i)}|Q_\theta(s_i,a_i)|} - \beta(\pi_\phi(s_t)-a_t)^2] \tag{7}$$

where θ and ϕ represent the network parameters of Critic and Actor of the individual network, respectively. More specifically, the value of β is made up of two components. The proportional component is determined by the discrepancy between the current episodic return $R_{current}$ and the target return R_{target}, while the derivative component is determined by the difference in returns between the current episode $R_{current}$ and the previous episode R_{last}. The formula can be expressed as follows:

$$\Delta\beta = K_p(R_{current} - R_{target}) + K_d \cdot \max(0, R_{last} - R_{current}) \tag{8}$$

where K_p and K_d are weights of two terms, R_{target} is a hyperparameter that needs to be set manually according to different tasks.

D. Bayesian optimization for morphology selection

Among the concurrent-network, the population network is trained to synthesize data from different morphologies. Hence we deem it can fit the Q function when meeting a new morphology configuration.

$$\begin{aligned} F(\pi^*(\xi),\xi) &\approx F(\pi_{pop},\xi) \\ &= \mathbb{E}[Q_{pop}(s_0,a_0,\xi)|a_0 \sim \pi_{pop}(\cdot|s_0,\xi)] \end{aligned} \tag{9}$$

As the morphology parameters we used are in continuous space, we employ a Gaussian Process to model (9). This model, denoted as $\mathcal{M}: \xi \mapsto F(\pi_{pop},\xi)$, is trained and utilized to calculate the acquisition function $\psi_i(\xi)$. Specifically, we adopt the Gaussian Process Upper Confidence Bound (GP-UCB) [33] technique in our method. During each round of optimization, the optimization results are as follows:

$$\xi_i = \arg\max_{\xi\in\Xi} \psi_i(\xi) = \arg\max_{\xi\in\Xi} \mu_{i-1}(\xi) + \kappa^{\frac{1}{2}}\sigma_{i-1}(\xi) \tag{10}$$

where $\mu_{i-1}(\xi)$ and $\sigma^2_{i-1}(\xi)$ are the mean and variance of model $\mathcal{M}$ respectively. κ is a hyperparameter that controls the balance between exploration and exploitation. The subscript i indicates the number of times Bayesian optimization is conducted during a single morphology optimization process. In summary, the algorithm framework is presented in Fig.2, and the corresponding pseudo-code is as follows.

Algorithm 1 Bayesian Optimization Augmented by the Concurrent-Network

Initialize replay buffers: D_{pop}, D_{ind}, D_{init};
for each iteration **do**
  Initialize and empty D_{ind};
  $\xi = \xi_{new}$;
  for every training episode **do**
    for t in episode length T **do**
      Interact with the environment: $a_t \sim \pi_{ind}(s_t)$;
      Get next state s_{t+1} and reward r_t;
      Store (s_t, a_t, r_t, s_{t+1}) to D_{pop} and D_{ind};
      Store initial states s_0 to D_{init};
    end for
    Set $R_{last} = R_{current}$ and $R_{current} = \sum_{t=0}^{T} r_t$;
    Update weight β according to E.q.(8);
    for n in update numbers **do**
      Train population network with random batches from D_{pop} according to E.q.(6);
      Train individual network with random batches from D_{ind} according to E.q.(7);
    end for
  end for
  for i in BO update numbers **do**
    Find ξ_i by optimizing acquisition function over the GP according to E.q.(10);
    Sample initial states s_0 from D_{init};
    Calculate the objective value $F(\pi_{pop},\xi_i)$ according to E.q.(9);
    Augment $D_{BO}^{1:i} = \{D_{BO}^{1:i-1}, (\xi_i, F(\pi_{pop},\xi_i))\}$ and update the GP;
  end for
  $\xi_{new} = \arg\max_i F(\pi_{pop},\xi_i)$
end for

V. Experiments

In this section, we design several experiments to answer the following questions:

- Does the policy-constraint method mitigate the *exploration error* resulting from the population network's lack of interaction with the environment?

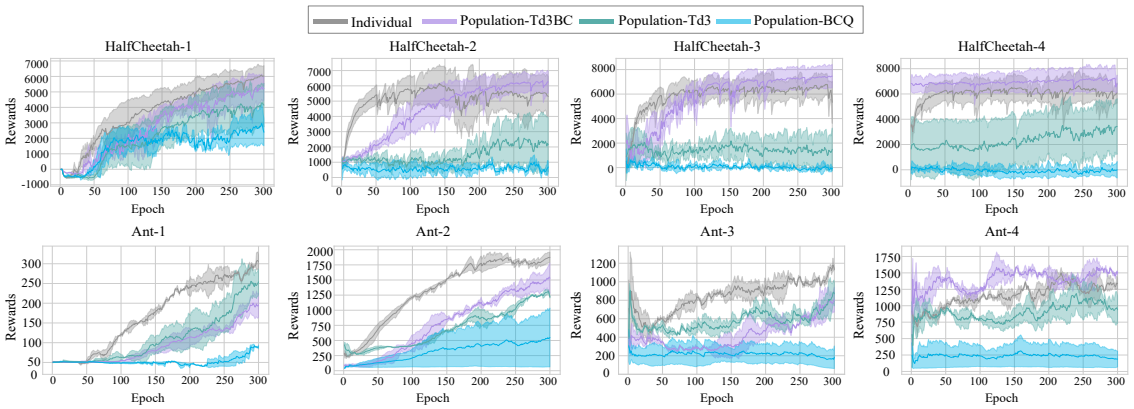

Fig. 3. Policy-constraint method analysis results. The x-axis represents the low-level episode, while the y-axis shows the accumulated rewards of that episode. We plot the mean and standard deviation across three runs.

- Does introducing the adaptive behavior cloning term alleviate the performance degradation caused by the sudden state-action *distribution shift* when initializing individual networks with parameters from population network?
- Does the proposed method exhibit a significant improvement in optimization performance when compared to the original dual-network method [9]?
- Do the optimization results remain valid when tested in physical experiments?

A. *Legged robot tasks in Simulation*

Setup: We investigate the performance of our proposed method on two legged-robot tasks: HalfCheetah and Ant. The former is a 2D motion task, while the latter is a 3D task. We modify the length of the robots' legs by changing the corresponding XML files. The training is conducted on an NVIDIA GeForce GTX 2080ti GPU, with the number of epochs set to 300 to balance efficiency and performance.

Policy-constraint method analysis: In this part, we design comparative experiments to answer the first question. Specifically, we manually configure four sets of morphological parameters (HalfCheetah 1-4, Ant 1-4 in Fig.1). The population network adopts three distinct implementations, as described below:

- **Population-TD3BC.** The proposed method, in which the population network is trained by TD3BC [20].
- **Population-BCQ.** Adopots another offline RL method BCQ [17] to train the population network. Specifically, a generative model is utilized to generate actions that are expected within the distribution range of actions in the replay buffer.
- **Population-TD3.** Train the population network with the TD3 [34] method, which is without offline settings.

The rewards obtained by the individual network and three types of population networks are depicted in Fig.3. To ensure fairness, the individual network is trained using the TD3 algorithm [34] and its parameters are not initialized by the population network anymore. For HalfCheetah-1 and Ant-1, both the individual network and population networks are trained using the same data (as only the data under the first group are utilized at the beginning), and the individual network achieves the highest rewards, indicating the presence of exploration errors problem. By comparing the subsequent groups, it is found that the rewards of Population-TD3BC gradually converge towards those of the individual network and even surpass them during training. The rewards of Population-BCQ are lower than those of Population-TD3, possibly due to the introduction of additional generative networks that could result in a training slowdown. These experiments demonstrate that the proposed method effectively mitigates the exploration error problem in the offline setting, enabling the population network to provide more reliable estimations for upper-layer morphology optimization.

Adaptive behavior cloning term analysis: We conduct four comparative experiments to answer the second question.

- **No Copy.** Initialize the individual network with random parameters.
- **Direct Copy.** Copy the parameters of the population network directly to the individual network.
- **Fixed Term.** Fix β in (7) to reduce the performance drop caused by the distribution shift.
- **Adaptive Term.** The proposed method, β is adjusted dynamically as the training progresses.

The results are presented in Fig.4. We use five groups of morphological parameters (HalfCheetah1-5, Ant1-5 in Fig. 1). The training starts with the first group and ends with the fifth group. Since there are no parameters transmitted in the first group, we exclude its results. From Fig. 4, we observe that the initial rewards of the four methods are similar in the second group. As training progresses, it is evident that No Copy has the lowest rewards among the initial rewards. Due to the distribution shift, Direct Copy's initial rewards are neither high. The initial rewards of Adaptive Term and Fixed Term are relatively high, indicating that these two methods can alleviate the performance drop caused by the distribution shift. However, as the fixed behavior cloning term limits the agents' exploration, the rewards of the Fixed Term at the end of the epoch are not as high as those of the proposed method. The above experiments demonstrate that the proposed method can reduce the initial performance drop while enabling the agent to maintain a high degree of exploration, resulting in higher rewards than other methods.

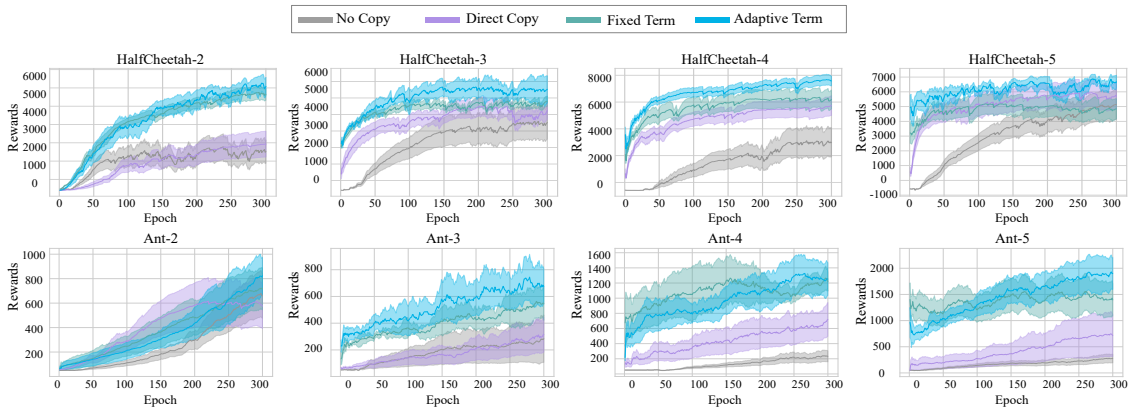

Fig. 4. Adaptive behavior cloning term analysis results. The x-axis represents the low-level episode, while the y-axis shows the accumulated rewards for that episode. We plot the mean and standard deviation across three runs.

It is worth noting that although the parameters of Direct Copy, Fixed Term, and Adaptive Term are copied from the same population network, the curves in Fig.4 are obtained from the evaluation stage (after the training stage), and the initial network parameters change after the training stage. Therefore, the initial rewards may not have the same values, as reported in other offline-to-online works [11], [12].

Morphology optimization results: We compare the morphology optimization performance of the proposed method to that of four baselines. To facilitate analysis, we include the morphology optimization results of the four-legged robot in the simulation environment in this section.

- **Coadapt_SP.** The original implementation of the dual-network framework, in which both the individual network and population network are trained by the online RL algorithm SAC [35], and the morphology is optimized by particle swarm optimization (PSO) method.
- **Coadapt_TP.** Replaces the original SAC algorithm with the TD3 algorithm. (By adjusting the hyperparameters, we want to ensure that the results of Coadapt_SP can be replicated.)
- **Coadapt_TB.** Replace PSO in Coadapt_TP with Bayesian optimization.
- **Random Sampling.** Sample designs uniformly at random within the parameter ranges.

The cumulative rewards under optimized morphology with the corresponding controller are shown in Tab.I, where the symbol "#" represents morphology optimization iterations, and the p-values of each two methods are placed between the two rows. We conduct independent T-tests between Coadapt_SP and Coadapt_TP for the three tasks, and all p-values are higher than the threshold of 0.05, indicating that by selecting hyperparameters, we ensure that the results are independent of the algorithm choice. Additionally, we perform independent T-tests between Coadapt_TP and Coadapt_TB for the three tasks. The p-values for the Four-legged robot and Halfcheetah are less than 0.05, while that of the Ant is greater than 0.05. Moreover, all mean values of Coadapt_TB are greater than those of Coadapt_TP. Thus, we can conclude that Bayesian optimization is more suitable for our task most of the time. The p-values between Coadapt_TB and the Proposed method are all less than the threshold, indicating that the introduction of the concurrent-network architecture is indeed effective in the co-design task. Furthermore, the rewards of the Proposed method have a relatively steady upward trend during the optimization process, demonstrating the effectiveness of the proposed improvements compared to other baselines. As the lower bound of optimization, the Random Sampling method has the lowest rewards, which is in line with our expectations.

B. Legged robot task in real world

In this section, we examine the feasibility of the proposed method in the real world by utilizing a four-legged robot. At the lower level, we combine RL with Central Pattern Generator (CPG) [36], [37], and define the action of RL as the phase difference of CPG to train gaits. Initially, a simulation model identical to the physical robot is constructed, and the proposed algorithm is then implemented in the simulator.

The optimization results of the simulation are displayed in Tab.I, demonstrating similar performance to that of HalfCheetah and Ant tasks. The results of the gait (policy) optimization are presented in Fig.5. We compare the trained gait with three classical gaits (walk, trot, and pace). In each row of Fig.5, eight instances are recorded from left to right, corresponding to the first to eighth seconds when the robot starts to move. It is apparent that the fastest gait is the trained one, which affirms the efficacy of policy optimization.

The results of the morphology optimization are depicted in Fig.6. We adopt the optimal gait for the original and optimized morphology configurations and capture the positions reached by each robot during the same time interval (approximately 0.6s). It is evident that the robot with the optimized morphology configuration reached the end within the allotted time, whereas the robot with the initial morphology configuration only reached position 3. The findings demonstrate that the optimized morphology configuration can indeed enhance the robot's motion performance. Additionally, we select two optimized morphology parameters - front leg length, and rear leg length, as the x-axis and y-axis, respectively, and plot them in Fig.7. Upon analyzing it, we observe that all optimized points remain in the upper left corner, suggesting that longer rear legs ($i.e.$, a forward center

TABLE I

MORPHOLOGY OPTIMIZATION RESULTS

Environment	Method	#2	#4	#6	#8	#10	#12	#14	#16	#18	#20	Mean	p-value
HalfCheetah	Coadapt_SP	5924.76 ±374.5	5655.03 ±644.03	5812.25 ±1260.22	7015.14 ±339.66	6728.55 ±172.12	7170.83 ±107.02	7287.99 ±139.61	7369.74 ±30.23	7626.74 ±419.68	7614.88 ±192.26	6820.6	0.1968
	Coadapt_TP	**7178.99 ±629.97**	7129.19 ±706.58	7436.12 ±433.62	7433.26 ±322.11	6778.18 ±103.72	6910.45 ±442.8	7347.12 ±372.43	7285.93 ±44.24	6977.98 ±353.95	7071.89 ±414.43	7154.92	6.1399×10^{-4}
	Coadapt_TB	6930.2 ±394.19	**7738.43 ±960.86**	8122.24 ±614.25	8235.96 ±343.09	8021.7 ±257.36	7980.29 ±58.02	7676.38 ±142.35	7433.61 ±207.37	7681.02 ±422.12	7545.66 ±131.21	7736.55	2.0961×10^{-3}
	Proposed method	7144.15 ±227.66	7524.32 ±261.51	**8341.74 ±405.17**	**9129.20 ±433.11**	**9133.28 ±385.74**	**9231.53 ±517.56**	**9294.37 ±521.32**	**9373.68 ±420.81**	**9115.31 ±495.18**	**9127.19 ±484.34**	**8741.48**	6.5502×10^{-12}
	Random Sampling	4238.71 ±369.64	4838.95 ±190.70	5100.78 ±1062.57	4148.00 ±660.25	4400.39 ±101.54	4567.07 ±584.55	4585.07 ±66.55	4458.30 ±1190.42	4504.45 ±722.85	4756.85 ±1027.57	4559.86	
Ant	Coadapt_SP	3148.73 ±319.88	3116.11 ±528.07	3810.01 ±237.98	4178.09 ±356.03	4170.36 ±347.52	4400.92 ±365.0	4249.47 ±405.0	3933.58 ±259.84	3713.71 ±225.22	2938.52 ±679.30	3765.95	0.5982
	Coadapt_TP	3507.22 ±238.45	3678.96 ±227.16	3761.81 ±287.83	3565.20 ±225.75	3914.69 ±330.73	4165.95 ±202.42	4018.89 ±316.82	4144.58 ±180.11	4033.96 ±282.52	3844.03 ±175.74	3863.53	0.8766
	Coadapt_TB	3243.30 ±347.53	3741.11 ±276.01	4046.61 ±218.76	3964.59 ±44.86	4260.58 ±289.96	4082.87 ±321.40	4108.96 ±353.13	3883.07 ±381.47	3905.95 ±463.52	3585.50 ±358.13	3882.25	1.2397×10^{-6}
	Proposed method	**4200.89 ±429.03**	**4496.12 ±116.18**	**4675.72 ±226.38**	**4733.33 ±318.72**	**4984.87 ±322.91**	**4846.29 ±271.03**	**4936.87 ±195.32**	**5057.18 ±51.42**	**5001.50 ±185.93**	**5094.02 ±163.47**	**4802.68**	6.5943×10^{-11}
	Random Sampling	2832.19 ±193.83	2349.64 ±201.47	2972.70 ±193.05	3003.18 ±51.14	3368.03 ±240.22	3322.97 ±175.37	3342.69 ±37.40	2810.71 ±137.81	2871.90 ±173.53	3089.62 ±222.96	2996.36	
Four-legged robot	Coadapt_SP	5387.32 ±1190.59	6191.51 ±708.47	6101.63 ±544.73	6527.56 ±478.78	6747.27 ±673.71	6964.32 ±212.86	6292.54 ±543.68	6116.06 ±501.15	6162.80 ±624.79	4808.85 ±2522.27	6129.98	0.5135
	Coadapt_TP	6097.78 ±367.85	6627.54 ±138.04	6172.61 ±862.58	6074.53 ±1162.6	6545.82 ±487.69	6237.02 ±722.41	6492.50 ±464.20	5827.08 ±1223.99	6184.13 ±932.55	6470.44 ±415.83	6272.94	1.8359×10^{-3}
	Coadapt_TB	6461.69 ±190.38	6572.23 ±434.52	6452.36 ±234.14	6669.88 ±330.47	6640.23 ±380.55	6337.23 ±263.05	6682.14 ±395.27	6872.21 ±329.48	6772.9 ±218.83	6994.41 ±208.39	6645.53	3.0324×10^{-10}
	Proposed method	**7496.97 ±549.98**	**7841.40 ±332.96**	**8055.03 ±660.95**	**8103.69 ±398.92**	**8103.93 ±155.72**	**8264.59 ±318.35**	**8396.84 ±88.05**	**8525.23 ±252.80**	**8273.15 ±163.17**	**8640.00 ±169.70**	**8170.09**	3.7904×10^{-11}
	Random sampling	3303.81 ±1885.33	4109.96 ±2447.13	2247.02 ±589.59	4334.36 ±2581.22	4808.80 ±2136.08	2795.10 ±1872.26	4048.07 ±1153.41	2440.10 ±1668.91	4390.83 ±1777.40	4680.92 ±2524.79	3715.90	

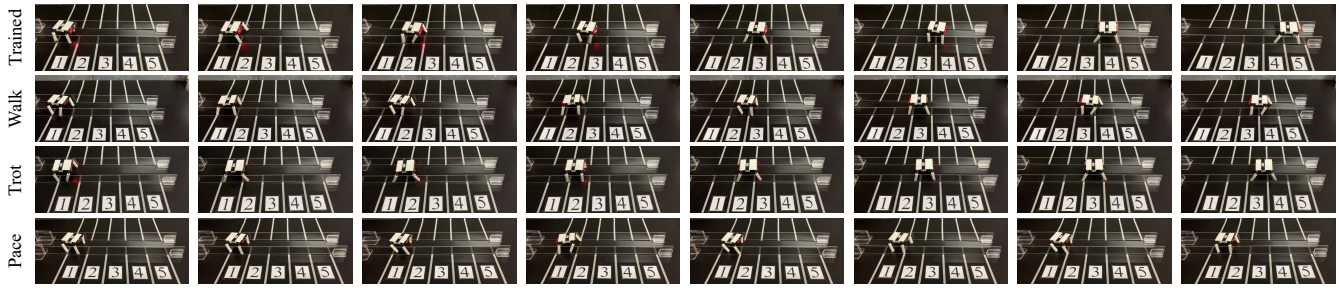

Fig. 5. The gait optimization results, with the name of the gait are displayed on the left.

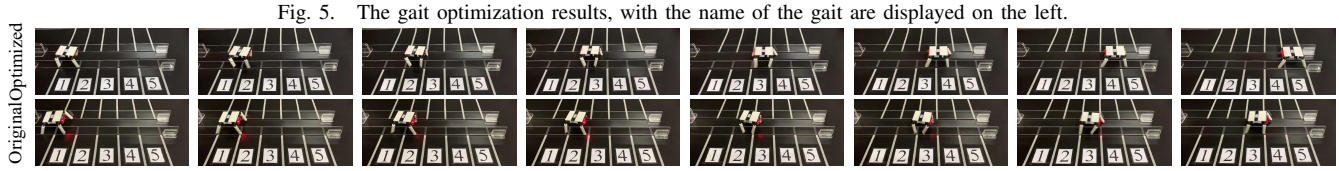

Fig. 6. The morphology optimization results. The first row displays the optimized morphology, while the second row shows the original morphology.

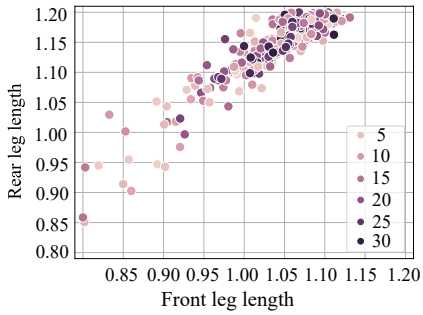

Fig. 7. The morphology optimization result, the colors of points represent the iterations of morphology optimization.

of mass) will enable the robot to run faster. This finding is also consistent with our prior knowledge. Both outcomes confirm the effectiveness of the proposed method.

VI. CONCLUSIONS

In this paper, we propose the concurrent network, a simple yet effective method to solve problems that can be modeled as bi-level optimization, such as policy and morphology co-design of robots. In which the population network is trained offline to solve the upper-level task, and the individual network is trained online to solve the lower-level task. By leveraging the behavior cloning term flexibly, an effective combination of both networks is achieved. We validate the proposed method through extensive simulation and real-world experiments, showing its superiority over baseline algorithms. Furthermore, the proposed method can optimize not only continuous but also discrete variables by replacing Bayesian optimization based on Gaussian Process with Bayesian optimization based on Random Forest, without changing the network architecture. The current limitation is that the proposed method has only been verified on an open-loop control system of a physical robot with a simple structure. In future work, we will continue to optimize the physical robot and install some sensors to form a closed-loop control system to adapt to the changing environment, such as locomotion in the presence of uneven terrain, obstacle, variations in friction, etc.

REFERENCES

[1] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Joint optimization of robot design and motion parameters using the implicit function theorem." in *Robotics: Science and systems*, vol. 8, 2017.

[2] M. Geilinger, R. Poranne, R. Desai, B. Thomaszewski, and S. Coros, "Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–12, 2018.

[3] M. Geilinger, S. Winberg, and S. Coros, "A computational framework for designing skilled legged-wheeled robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3674–3681, 2020.

[4] G. Fadini, T. Flayols, A. Del Prete, N. Mansard, and P. Souères, "Computational design of energy-efficient legged robots: Optimizing for size and actuators," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9898–9904.

[5] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter, "Jointly learning to construct and control agents using deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9798–9805.

[6] T. Wang, Y. Zhou, S. Fidler, and J. Ba, "Neural graph evolution: Towards efficient automatic robot design," *arXiv preprint arXiv:1906.05370*, 2019.

[7] D. J. Hejna III, P. Abbeel, and L. Pinto, "Task-agnostic morphology evolution," *arXiv preprint arXiv:2102.13100*, 2021.

[8] A. Gupta, S. Savarese, S. Ganguli, and L. Fei-Fei, "Embodied intelligence via learning and evolution," *Nature communications*, vol. 12, no. 1, pp. 1–12, 2021.

[9] K. S. Luck, H. B. Amor, and R. Calandra, "Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 854–869.

[10] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[11] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin, "Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble," in *Conference on Robot Learning*. PMLR, 2022, pp. 1702–1712.

[12] Y. Zhao, R. Boney, A. Ilin, J. Kannala, and J. Pajarinen, "Adaptive behavior cloning regularization for stable offline-to-online reinforcement learning," 2021.

[13] A. Zhao, J. Xu, M. Konaković-Luković, J. Hughes, A. Spielberg, D. Rus, and W. Matusik, "Robogrammar: graph grammar for terrain-optimized robot design," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–16, 2020.

[14] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "Chainqueen: A real-time differentiable physical simulator for soft robotics," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6265–6271.

[15] P. Ma, T. Du, J. Z. Zhang, K. Wu, A. Spielberg, R. K. Katzschmann, and W. Matusik, "Diffaqua: A differentiable computational design pipeline for soft underwater swimmers with shape interpolation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–14, 2021.

[16] J. Xu, T. Chen, L. Zlokapa, M. Foshey, W. Matusik, S. Sueda, and P. Agrawal, "An end-to-end differentiable framework for contact-aware robot design," *arXiv preprint arXiv:2107.07501*, 2021.

[17] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2052–2062.

[18] A. Kumar, J. Hong, A. Singh, and S. Levine, "When should we prefer offline reinforcement learning over behavioral cloning?" *arXiv preprint arXiv:2204.05618*, 2022.

[19] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[20] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[21] N. Y. Siegel, J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess, and M. Riedmiller, "Keep doing what worked: Behavioral modelling priors for offline reinforcement learning," *arXiv preprint arXiv:2002.08396*, 2020.

[22] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," *arXiv preprint arXiv:1910.00177*, 2019.

[23] Y. Liu, A. Swaminathan, A. Agarwal, and E. Brunskill, "Off-policy policy gradient with state distribution correction," *arXiv preprint arXiv:1904.08473*, 2019.

[24] A. Swaminathan and T. Joachims, "Batch learning from logged bandit feedback through counterfactual risk minimization," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1731–1755, 2015.

[25] O. Nachum, B. Dai, I. Kostrikov, Y. Chow, L. Li, and D. Schuurmans, "Algaedice: Policy gradient from arbitrary experience," *arXiv preprint arXiv:1912.02074*, 2019.

[26] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[27] I. Kostrikov, R. Fergus, J. Tompson, and O. Nachum, "Offline reinforcement learning with fisher divergence critic regularization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5774–5783.

[28] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn, "Combo: Conservative offline model-based policy optimization," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[29] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims, "Morel: Model-based offline reinforcement learning," *Advances in neural information processing systems*, vol. 33, pp. 21 810–21 823, 2020.

[30] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma, "Mopo: Model-based offline policy optimization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14 129–14 142, 2020.

[31] F. Muratore, C. Eilers, M. Gienger, and J. Peters, "Data-efficient domain randomization with bayesian optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 911–918, 2021.

[32] F. Muratore, T. Gruner, F. Wiese, B. Belousov, M. Gienger, and J. Peters, "Neural posterior domain randomization," in *Conference on Robot Learning*. PMLR, 2022, pp. 1532–1542.

[33] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *arXiv preprint arXiv:0912.3995*, 2009.

[34] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[35] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[36] C. Wang, G. Xie, L. Wang, and M. Cao, "Cpg-based locomotion control of a robotic fish: Using linear oscillators and reducing control parameters via pso," *International Journal of Innovative Computing Information and Control*, vol. 7, no. 7B, pp. 4237–4249, 2011.

[37] A. Crespi, D. Lachat, A. Pasquier, and A. J. Ijspeert, "Controlling swimming and crawling in a fish robot using a central pattern generator," *Autonomous Robots*, vol. 25, no. 1, pp. 3–13, 2008.