2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

SIGNRL: A Population-based Reinforcement Learning Method for Continuous Control*

Daniel F. Zambrano-Gutierrez*, Alberto C. Molina-Porras†, Emmanuel Ovalle-Magallanes†,
Iván Amaya*, José C. Ortiz-Bayliss*, Juan G. Avina-Cervantes†, and Jorge M. Cruz-Duarte*,
*Tecnologico de Monterrey, Monterrey 64849, Mexico, E-mails: {A00836756, jcobayliss, iamaya2, jorge.cruz}@tec.mx
†University of Guanajuato, Salamanca 36885, Mexico, E-mails: {a.molinaporras, e.ovallemagallanes, avina}@ugto.mx

***Abstract*—In engineering processes that require continuous control, it is common to face significant challenges. Addressing these challenges through explicit modeling can take much work and effort. For this reason, Reinforcement Learning (RL) has gained popularity as a feasible strategy for solving this problem. In this context, various value-based methodologies, policies, or combinations have been employed to obtain an optimal learning policy. However, problems such as convergence to local maxima and high variance in training persist. In addition, computational time and cost increase in complex environments, so more robust RL methodologies are required. This paper proposes a Swarm Intelligence Guided Neural Reinforcement Learning (SIGNRL) algorithm, which uses Particle Swarm Optimization as a multi-agent parameter explorer to find the optimal policy. Numerical results obtained in the OpenAI Gym Cart-Pole environment show that SIGNRL, with its gradient-free learning, exhibits good convergence and lower variance in continuous control tasks.**

***Index Terms*—Reinforcement Learning, Artificial Neural Networks, Particle Swarm Optimization, Control Systems.**

I. Introduction

In real-world applications, it is common to come across the need to implement control systems correctly. This is due to the dynamics of systems that, without a controller, exhibit undesirable features [1]. However, finding the optimal parameters for specialized controllers to address these tasks can be complex and laborious [2, 3]. In addition, the inherent complexity of many continuous control systems can generate unexpected or undesirable results, increasing the challenge of implementing an efficient and reliable solution. Moreover, nonlinear effects and system disturbances can lead to unexpected behaviors [4], requiring careful attention in the design and commissioning of controllers. To overcome this problem, alternative approaches based on Reinforcement Learning (RL) have been proposed in the literature to control system dynamics [5]. RL adopts the structured framework of Markov Decision Processes to outline the agent-environment interaction in terms of states, actions, and rewards, inspired by how humans and animals learn through trial and error when interacting with their surroundings.

RL generally exploits diverse techniques, such as value-based, policy-based, or both, to learn an optimal policy [6, 7]. Value-based methods aim to estimate different state-action pairs using such information to select the actions. These value-based methods effectively manage tasks involving continuous states by employing approximation functions. However, they face a significant limitation in handling tasks with continuous actions due to the inherent difficulty in identifying the optimal function across the action space [8].

In contrast, policy-based methods directly learn an optimal policy without explicitly estimating values. In this case, the policy can be represented as a parameterized function that can receive states and generate the best action possible in a continuous domain. In Deep RL, this parameterized policy is defined as a Feed-forward Neural Network (FNN) [9] optimized in the training process by Policy Gradient (PG) algorithms [10]. For continuous control tasks, several PG-based methods have been studied [11, 12]. However, these alternatives present some drawbacks when the environmental complexity increases, generally related to data insufficiency and training inefficiency. The episodic control concept was introduced to overcome this problem, where episodic training applied with PG achieved a satisfactory generalization for continuous control, requiring less training time [13]. Still, PG-based methods can converge to a local maximum instead of a global optimum and exhibit high variance in their training process [14]. Some gradient-free Population-based Reinforcement Learning (PBRL) methods have been proposed to avoid these common weaknesses of policy gradient algorithms [8, 15–17]. Population-based training methods have effectively taught robust policies in model-free continuous control problems. They leverage the benefits of multi-agent parallelism and explore the extensive search space commonly encountered in optimization problems.

This work proposes an alternative PBRL training method integrating Particle Swarm Optimization (PSO) with neural parameterized policies, where the best policy selection depends on the highest average cumulative rewards to guarantee the best long-term performance. Based on episodic learning, the proposed methodology is efficient for continuous control tasks in environments with continuous states and action domains. Besides, through the PSO exploration, the system found optimal policies capable of executing a feasible response to external disturbances.

*Daniel F. Zambrano-Gutierrez, Alberto Carlos Molina-Porras, and Emmanuel Ovalle-Magallanes thank the Consejo Nacional de Humanidades, Ciencia y Tecnología (CONAHCyT) for the financial support provided through their fellowships 1046000, 1156648, and 626154, respectively. Daniel F. Zambrano-Gutierrez and Jorge M. Cruz-Duarte also thank the Tecnológico de Monterrey for their support under grant FAP 2275.

II. Theoretical Foundations

A. Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning paradigm that focuses on training agents to make decisions in a known (model-based) or unknown (model-free) environment to maximize a cumulative reward. In model-free RL, an agent learns by interacting with an unknown dynamic environment over several episodes or continued tasks but defined by discrete time steps [18–20]. It disposes of no task models nor prior knowledge of the dynamics. At each time step, the agent observes the current state s_t of the environment, takes an action a, and then observes the next predicted state s_{t+1}, receiving feedback as a reward r_t until reaching a terminus state or a time limit. The agent aims to find an optimal policy π, which maps from states to actions while maximizing the cumulative long-term rewards [21].

For model-free problems with continuous states and actions, Policy Gradient (PG) methods have become a popular solution for policy-based RL applications. PG methods learn a parameterized policy $\pi(a|s,\boldsymbol{\psi})$ w.r.t. the long-term expected return by gradient descent [10]. This policy is responsible for selecting an action a given the environment's state s with parameter vector $\boldsymbol{\psi} \in \mathbb{R}^d$, where d is the number of parameters in $\boldsymbol{\psi}$, without consulting a value function. A parameterized value function $v(s,\boldsymbol{\phi})$ with $\boldsymbol{\phi} \in \mathbb{R}^d$, may still be used to learn the policy parameter but is not required for action selection after the training process [21].

B. Population-Based Reinforcement Learning

Instead of PG methods, Population-based Reinforcement Learning (PBRL) algorithms [16] update a population of N agents (policies) defined by $\{\pi_{\boldsymbol{\psi}_i}\}_{i=1,\dots,N}$ that explore the environment in parallel and learn from each other to improve their collective performance. The key idea behind PBRL is to maintain and evolve a diverse set of policies to explore different regions of the policy space simultaneously [22]. As the agents interact with the environment and collect experience, the population is updated based on performance or some selection mechanism, encouraging better policies to thrive while pruning out less successful ones.

This approach helps mitigate some PG issues like getting stuck in local optima and improves the chances of finding a globally optimal or near-optimal policy [8].

C. Cart-Pole Problem

A Cart-Pole environment comprises, in its simplest form, an unactuated joint attaching a pole to a cart moving along a frictionless track. The problem, proposed in [23], consists of balancing such a pole, an inverted pendulum, placed upright on the cart. An external force should be applied in the same or opposite direction of the cart displacement to hold the pole position. Fig. 1 illustrates the Cart-Pole dynamics implemented in this work based on [23, 24].

Thus, the Cart-Pole system dynamic can be modeled using the expressions reported in [24], which depend on the angular pole position θ [rad] and the cart position x [m]; see Fig. 1.

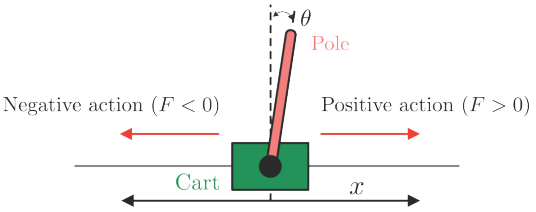

Fig. 1: Sketch of the Cart-Pole mechanical system.

Velocities, accelerations, masses, the pole length, an gravitational acceleration are also relevant, but we avoid extending their formulation for brevity. For this implementation, we used the parameter values defined by OpenAI Gym [25]. Notwithstanding, it is essential to mention that F [N] in Fig. 1 stands for the external force applied to the cart, linked with the action performed by the controller or agent.

Therefore, in the reinforcement learning context, the intervals for the observation parameters used to describe the environment behavior (and the states) are summarized in TABLE I. Besides, the actions $a_t \in A$ for this environment correspond to the moving force exerted in the cart described as $A = [-20, 20]$ N, where the positive forces represent a move to the right and the negative a move to the left.

TABLE I: Intervals of the Cart-Pole observation parameters.

Parameter	Symbol [Unit]	Min. Value	Max. Value
Cart position	x [m]	-4.8	4.8
Cart velocity	$\dot{x}$ [m/s]	$-\infty$	∞
Pole angular position	θ [rad]	-0.418	0.418
Pole angular velocity	$\dot{\theta}$ [rad/s]	$-\infty$	∞

Recall that the agent aims to keep the pole angular position θ as near as possible to zero during the entire episode, trying not to move the cart far from the center at $x = 0$ m. For every step in an episode, a reward r_t is defined by

$$r_t = \boldsymbol{w} \cdot \boldsymbol{g}(s_{t+1}|s_t), \tag{1}$$

with

$$\boldsymbol{g}(s_{t+1}|s_t) = \left(R, \frac{\theta_{s_t} - \theta_{s_{t+1}}}{\theta_{\max}}, \frac{x_{s_t} - x_{s_{t+1}}}{x_{\max}}\right)^{\intercal}, \tag{2}$$

where $\boldsymbol{w} \in \mathbb{R}_+^P$ is a weight vector and $\boldsymbol{g}(s_{t+1}|s_t)\colon S \mapsto \mathbb{R}^P$ is the reward criteria function, defined according to the number of reward criteria P. Without loss of generality, we considered $P = 3$ and $\boldsymbol{w} = (1, 100, 5)^{\intercal}$ chosen to balance the reward value r_t. Plus, for the first reward criterion in (2), we considered that R is given by

$$R = \begin{cases} 1, & \text{if } |\theta| \leq 12^\circ \wedge |x| \leq 2.4 \text{ m}, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

which is a simple and effective way to indicating the cart and pole are within the desired positions. The second and third criteria correspond to how good that transition is regarding the pole angle and cart position, respectively.

In addition, it is worth mentioning that the episode is halted if either the absolute pole angular position θ exceeds 12° (≈ 0.2094 rad), the absolute cart position x is greater than

2.4 m or the episode time t is long lasting more than 200 steps.

III. Proposed Approach

This work proposes a population-based reinforcement learning model using Particle Swarm Optimization (PSO) to explore the policy space and a simple Feed-forward Neural Network (FNN) for modeling policies. Such a technique, called Swarm Intelligence Guided Neural Reinforcement Learning (SIGNRL), can handle continuous states and actions for complex system controllers. Fig. 2 summarizes the overall framework where PSO leads exploration that creates and modifies the population of policies based on their cumulative rewards. After that, the actor based on FNN executes the policies and interacts with the environment. The remainder of this section details each of the elements of the proposed methodology.

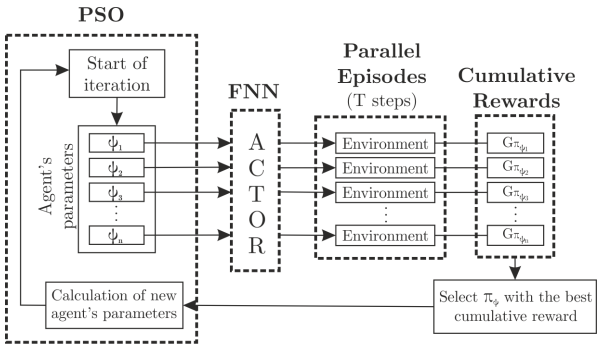

Fig. 2: Overall framework of the proposed SIGNRL algorithm.

A. Policy Iteration

As usual, the actor is an agent who perceives observations from the environment and provides the best available action [21]. This actor comprises an FNN architecture built with four input neurons (one per each observation in TABLE I), 20 hidden neurons with a hyperbolic tangent (`tanh`) activation function, and one output neuron also with a `tanh` activation function scaled to generate the action a. So, for the layer sizes, we get $L_0 = 4$, $L_1 = 20$, and $L_2 = 1$. The weight matrices for this FNN are given by $\boldsymbol{W}_1 \in \mathbb{R}_+^{L_0 \times L_1}$ and $\boldsymbol{W}_2 \in \mathbb{R}_+^{L_1 \times L_2}$, and the corresponding bias vectors $\boldsymbol{b}_1 \in \mathbb{R}^{L_1}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{L_2}$, describe the actor's policy $\pi(a|s;\boldsymbol{\psi})$ with,

$$\boldsymbol{\psi} = (\mathrm{vec}(\boldsymbol{W}_1)^\intercal, (\boldsymbol{b}_1)^\intercal, \mathrm{vec}(\boldsymbol{W}_2)^\intercal, (\boldsymbol{b}_2)^\intercal)^\intercal, \tag{4}$$

since $\boldsymbol{\psi} \in \mathbb{R}^{(L_0+1)L_1+(L_1+1)L_2}$ denotes the number of hyper-parameters in the FNN model and $\mathrm{vec}\colon \mathbb{R}^{n\times m} \mapsto \mathbb{R}^{nm}$ is a vectorization operation. For this particular implementation it is $(L_0+1)L_1+(L_1+1)L_2 = 121$. Nevertheless, we expressed it in that fashion to facilitate its extension and analysis.

The proposed method corresponds to a policy iteration algorithm, where the policy $\pi(a_t|s_t;\boldsymbol{\psi})$ defines the agent behavior. The impact of each action the agent takes on the environment is evaluated using the reward r_t in (1), whose value depends on the properties of the environment. This reward is obtained at each step of an episode and represents the short-term effect of the agent's actions.

However, a more valuable long-term performance in agent training is highly recommended to avoid the drawbacks of short-term rewards. Therefore, we propose using the Episodic Control approach by defining a finite episode, delimited by T steps, to interact with the environment. Indeed, the episode's duration can be variable depending on each agent; longer episodes indicate better agent performance, avoiding a premature terminal state. Consequently, long-term rewards,

$$G_{\pi_{\boldsymbol{\psi}}} = \textstyle\sum_{k=0}^{T-1} \gamma^k r_{k+1}, \tag{5}$$

correspond to the cumulative of no discounted rewards ($\gamma = 1$) obtained by the agent in a complete episode starting in s_0.

B. Training based on Swarm Intelligence

FNN hyper-parameters in the proposed scheme are estimated using Particle Swarm Optimization (PSO). Pseudocode 1 describes this optimizer, which renders a set of potential solutions subsequently provided to the actor and tested in the environment during a certain number of episodes (iterations) until the best possible solution is obtained.

Pseudocode 1 SIGNRL algorithm

Input: Number of agents N_A, number of episodes N_E, maximum number of steps per episode $T_{\max}$, FNN architecture for $\pi(a|s;\boldsymbol{\psi})$, and problem environment `env`.
Output: Parameters of best policy obtained $\boldsymbol{\psi}_*$

Initialize the global best fitness $f_* \leftarrow 0$
for $i \leftarrow 1$ to N_A **do**
  Initialize randomly the FNN hyper-parameters $\boldsymbol{\psi_i}$
  Initialize the i^{th} individual best fitness $f_{p_i} \leftarrow 0$
  Initialize the average reward value $\mathrm{ACR}_i \leftarrow 0$
for $j \leftarrow 1$ to N_E **do**
  parallel for $i \leftarrow 1$ to N_A **do**
    $s_0 \leftarrow$ `env.reset()` ▷ Get initial observation
    for $t \leftarrow 1$ to $T_{\max}$ **do**
      $a_{i,t} \leftarrow \pi(s_t, \boldsymbol{\psi}_i)$ ▷ Select an action
      $r_t, s_{t+1} \leftarrow$ `env.step(`$a_{i,t}$`)`
      Determine $G_{\pi_{\boldsymbol{\psi}_i}}$ with (5)
    if $f(\pi_{\psi_i}) \leq f_{p_i}$ **then** ▷ Using $f(\pi_{\psi_i})$ from (6)
      $f_{p_i} \leftarrow f(\pi_{\psi_i})$ and $\boldsymbol{p}_i \leftarrow \boldsymbol{\psi}_i$
    Update ACR_i using (9)
  for $i \leftarrow 1$ to N_A **do**
    $f_* \leftarrow f_{p_i}$ and $\boldsymbol{p}_* \leftarrow \boldsymbol{p}_i$ **if** $f_{p_i} \leq f_*$
    $\mathrm{ACR}_* \leftarrow \mathrm{ACR}_i$ and $\boldsymbol{\psi}_* \leftarrow \boldsymbol{p}_i$ **if** $\mathrm{ACR}_i \leq \mathrm{ACR}_*$
    Update $\boldsymbol{\psi}_i$ using (7) and (8)

Now, instead of modifying a single policy through exploration, PSO uses a population of N_A agents, such as $\boldsymbol{\Psi} = \{\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_{N_A}\}$, containing the hyper-parameters described in (4). These agents represent the candidate policies associated with a particular expected reward. Moreover, the fitness function for the PSO procedure is defined as,

$$f(\pi_{\boldsymbol{\psi}}) = -G_{\pi_{\boldsymbol{\psi}}}, \tag{6}$$

where G_{π_ψ} is the cumulative reward obtained by each agent in an entire episode during the current iteration.

Therefore, the agents evolve following a process guided by PSO, considering the best-known performance in the swarm at each iteration. It aims to improve each agent position (directly related to the FNN-based actor hyperparameters) so that the new population can generate policies with better rewards. This is possible using the well-known PSO's expressions [26], such as

$$\boldsymbol{v}_{i+1} \leftarrow \alpha_0 \boldsymbol{v}_i + \alpha_1 \boldsymbol{r}_1 \odot (\boldsymbol{p}_i - \boldsymbol{\psi}_i) + \alpha_2 \boldsymbol{r}_2 \odot (\boldsymbol{p}_* - \boldsymbol{\psi}_i)\,, \quad (7)$$

$$\boldsymbol{\psi}_{i+1} \leftarrow \boldsymbol{\psi}_i + \boldsymbol{v}_i, \quad (8)$$

where the inertial, self-confidence and swarm-confidence factors were set as $\alpha_0 = 0.9$, $\alpha_1 = 0.5$, and $\alpha_2 = 0.3$, respectively. In addition, $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are two vectors with i.i.d. random variables uniformly distributed in $[0, 1]$ and $\odot$ is the Hadamard-Schur product.

Lastly, PSO finishes the procedure when reaching the total number of episodes. Nevertheless, we implement an alternative performance metric to the fitness function in (6) based on the Average Cumulative Reward (ACR). So, the (ACR_*) for the i^{th} agent is updated each episode via

$$\text{ACR}_i \leftarrow \left((j-1)\text{ACR}_i + G_{\pi_{\psi_i}}\right)/j, \quad (9)$$

since $j = 1, \ldots, N_E$ corresponds to the episode counter. It is important to note that this function is only used for the final agent selection to obtain the best long-term performance. This alternative metric allows selecting the best agent fitness, which corresponds to the best Average Cumulative Reward (ACR_*).

IV. Experimental Results

All numerical case studies were performed in Python v3.9 running on an Intel® Xeon® Gold 6326 CPU @2.90 Hz, with 32 cores and 32 threads, with 32 GB of RAM. The Feedforward Neural Network (FNN) model was built using Tensorflow v2.13.0, and the Particle Swarm Optimization (PSO) algorithm was implemented using the PySwarms v1.3.0 framework. Moreover, the asynchronous multiprocessing module based on the Python `concurrent` library was utilized to improve the training process.

To implement the SIGNRL algorithm, we defined 500 episodes with 30 agents and a maximum of 200 steps for each agent. Furthermore, all data this work achieved is freely available at https://github.com/Danielfz14/SIGNRL_RL.

A. Training details

The SIGNRL algorithm was tested in 50 independent runs, wherein the parameters of policy networks were initialized using Glorot Uniform initialization at the start of each run. As illustrated in Fig. 3, the cumulative reward G_{π_ψ} obtained by the best agent of each run in each training episode shows that 500 episodes were enough to achieve a high convergence near a maximum cumulative reward of 207.

On the other hand, the average cumulative reward per episode from the first episode to the current episode in Fig. 4 exhibits an increasing trend in the learning process. This

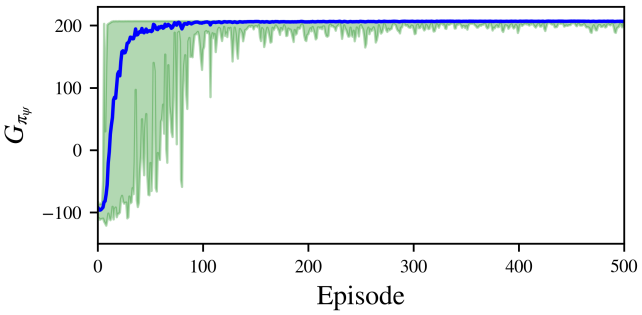

Fig. 3: Cumulative reward per episode of best agents obtained in 50 independent training runs.

curve shows that most of the agent's learning occurs in the first 200 episodes, reflecting a high learning speed.

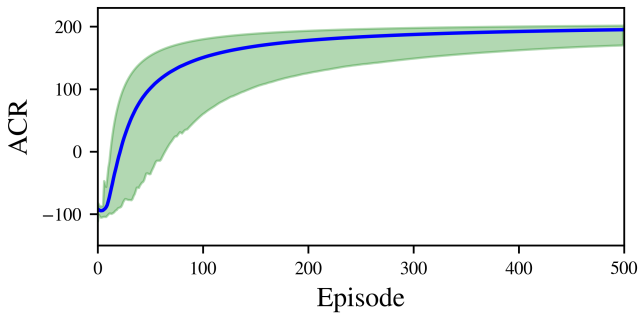

Fig. 4: Average cumulative reward per episode from the first episode to the current episode of best agents obtained in 50 independent training runs.

B. Performance validation

We tested the agents' performance obtained by the SIGNRL methodology to validate our approach under three scenarios. The first one corresponded to a set of short episodes with the same duration as training episodes; the second scenario considered episodes ten times longer than the first scenario; and the third one had extended episodes of the same duration as the second scenario but including disturbances due to external forces perceived by the cart. These disturbances were implemented according to $F = a + df$, where a is the action exerted by the agent and df is the disturbance composed of Dirac delta and Heaviside signals.

We conducted a repeatability test to verify the agent's performance. It consisted of running each scenario 50 times per agent with random initial states. The Average Mean Squared Error (AMSE) of pole angle position θ and cart position x to the reference was calculated for an entire episode, such as

$$\text{AMSE}_y = \frac{1}{N} \sum_{j=1}^{N} \mathbb{E}\left\{\|\boldsymbol{y}_j\|^2\right\}, \quad (10)$$

since $y = \{\theta, x\}$. For the expectation $\mathbb{E}$, we use a uniform distribution $f(\tau) = 1/\tau$ and consider the number of episodes performed N; we set it to 50 episodes per agent. $\tau = T_j$

corresponds to the total time-steps performed by the agent in the j^{th} episode. Besides, $y_k \in \boldsymbol{y}$ is the position, either θ_k or x_k, in k^{th} step; *i.e.*, $\boldsymbol{y} = (y_1, \ldots, y_\tau)^\intercal$.

As both θ and x are essential observations to measure the long-term behavior of agents, we implemented a metric to quantify the agent's performance based on the Median (med) and the Inter-Quartile Range (iqr) of each AMSE, as shown

$$\xi = \sum_{y=\{\theta,x\}} \text{med}(\text{AMSE}_y) + \text{iqr}(\text{AMSE}_y). \qquad (11)$$

Table II shows the repeatability performance of each best agent obtained by our proposal in each scenario. The logarithm of ξ indicates the capability to keep the pole angle and cart position as close to zero as possible, and $[\%]$ means the percent of episodes finished successfully. A smaller $\log(\xi)$ indicates that the agent performed better in the total or most of the episodes executed. Conversely, larger values show that the agent presented more significant difficulties in keeping θ or x stable close to zero. Agents that offered at least one uncompleted episode were highlighted in red, and the agent with the best global performance was highlighted in blue.

The repeatability test shows that 100% of the best agents completed the short episode scenario, 80% of agents had a good performance in the long episode, demonstrating an acceptable generalization, and 72% responded correctly to the situation with disturbances.

Fig. 5 plots the environment observations and actions of the best global agent tested in the long episode scenario with disturbances. It proves that although θ and x started in random initial values, the agent could create the actions needed to stabilize quickly θ and x as near as possible to zero in the entire episode. Indeed, the best agent generated the correct actions that compensated for the effect of additional forces of disturbances.

V. Conclusions

This paper introduced an alternative approach for Population-based Reinforcement Learning using the PSO algorithm for exploring policy parameters. The experimental results prove that the SIGNRL method proposed offers a straightforward and gradient-free framework for addressing continuous control problems in reinforcement learning.

The main limitation of using a finite episode in the training process for RL systems is the possibility of finding a policy with an outperforming result in the episodic training sessions that do not necessarily guarantee the best long-term performance for continuous operation outside of that episode's boundaries. The results with actors trained in a short episode and tested in episodes ten times longer show that the selection of policy parameters based on the best average cumulative reward with the SIGNRL method can be a feasible alternative to avoid the limitation of episodic training when it is necessary to select the best policy for long-term operation. The conducted experiments also show that our proposal requires low training times and exhibits high convergence rates with environments involving continuous states and actions.

TABLE II: Repeatability results for each best agent obtained with RL-PSO model proposed in 50 intents. Agents that rendered at least one uncompleted episode are highlighted in red, while those with the best global performance are in blue.

	Scenario					
Episodes:	**Short**		**Long**		**Long**	
Disturbances:	**No**		**No**		**Yes**	
Agent's Id.	**log(ξ)**	**[%]**	**log(ξ)**	**[%]**	**log(ξ)**	**[%]**
1	-4.389	100.0	-5.41	100.0	-4.257	100.0
2	-3.485	100.0	-2.563	100.0	-2.601	100.0
3	-4.25	100.0	-3.009	100.0	**0.05**	**0.0**
4	**-4.852**	**100.0**	**-5.953**	**100.0**	**-5.484**	**100.0**
5	-3.638	100.0	-4.362	100.0	**-2.86**	**82.0**
6	-3.55	100.0	**-0.519**	**66.0**	**-0.342**	**62.0**
7	-4.539	100.0	-4.302	100.0	**-4.355**	**90.0**
8	-4.59	100.0	-5.461	100.0	-3.72	100.0
9	-3.909	100.0	-3.121	100.0	-3.096	100.0
10	-4.234	100.0	-2.969	100.0	-3.387	100.0
11	-4.745	100.0	-5.329	100.0	-4.003	100.0
12	-2.932	100.0	**0.187**	**0.0**	**0.081**	**2.0**
13	-3.781	100.0	**0.146**	**0.0**	**0.142**	**0.0**
14	-3.694	100.0	-1.867	100.0	-1.753	100.0
15	-2.863	100.0	**0.241**	**0.0**	**0.258**	**0.0**
16	-4.811	100.0	-5.77	100.0	-4.625	100.0
17	-4.503	100.0	-5.654	100.0	-4.582	100.0
18	-3.512	100.0	-2.246	100.0	**0.06**	**4.0**
19	-4.742	100.0	-5.934	100.0	-4.884	100.0
20	-3.057	100.0	-1.865	100.0	-2.123	100.0
21	-3.068	100.0	**0.208**	**0.0**	**0.213**	**0.0**
22	-4.735	100.0	-5.458	100.0	-4.05	100.0
23	-3.519	100.0	**0.12**	**0.0**	**0.093**	**0.0**
24	-4.75	100.0	-5.771	100.0	-4.541	100.0
25	-4.395	100.0	-4.633	100.0	-4.456	100.0
26	-3.681	100.0	-3.313	100.0	-2.552	100.0
27	-4.367	100.0	-3.088	100.0	-3.588	100.0
28	-4.698	100.0	-5.609	100.0	-3.978	100.0
29	-3.765	100.0	**0.252**	**0.0**	**0.193**	**0.0**
30	-3.494	100.0	**0.192**	**0.0**	**0.198**	**0.0**
31	-4.002	100.0	-2.431	100.0	-3.211	100.0
32	-4.316	100.0	-4.534	100.0	-4.206	100.0
33	-3.857	100.0	**0.233**	**0.0**	**0.227**	**0.0**
34	-4.571	100.0	-5.234	100.0	-4.967	100.0
35	-4.537	100.0	-3.209	100.0	**-0.629**	**0.0**
36	-4.742	100.0	-5.864	100.0	-3.771	100.0
37	-4.338	100.0	-2.976	100.0	-3.072	100.0
38	-4.564	100.0	-5.344	100.0	-4.756	100.0
39	-3.829	100.0	**0.354**	**40.0**	**0.15**	**4.0**
40	-4.854	100.0	-5.829	100.0	-4.957	100.0
41	-4.431	100.0	-2.954	100.0	-3.328	100.0
42	-4.37	100.0	-3.127	100.0	-3.148	100.0
43	-4.189	100.0	-4.67	100.0	-4.005	100.0
44	-4.149	100.0	-3.41	100.0	**-3.334**	**86.0**
45	-3.538	100.0	-3.049	100.0	-3.039	100.0
46	-3.534	100.0	**0.178**	**2.0**	**0.158**	**2.0**
47	-4.298	100.0	**-1.11**	**70.0**	**0.079**	**0.0**
48	-4.541	100.0	-5.457	100.0	-4.866	100.0
49	-4.535	100.0	-5.02	100.0	-4.27	100.0
50	-4.127	100.0	-4.656	100.0	-3.611	100.0

In addition, the performance of the best agents in long episodes with disturbances shows that even when the agent was not trained to manage a disturbance situation, it could generate the correct actions to compensate for the unexpected forces introduced throughout the test. Therefore, it shows the robustness and adaptability of the best agents found

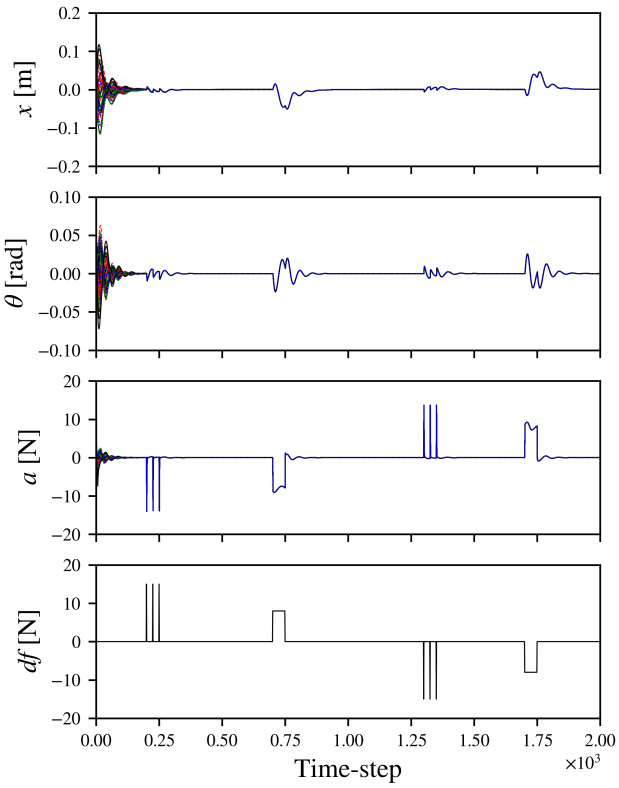

Fig. 5: Cart position, pole angle, and actions executed by the best agent, obtained with the proposed SIGNRL method, in 50 episodes of 2000 time steps with external disturbances.

by the proposed method and the capability of the SIGNRL algorithm to find a long-term stable solution with adequate generalization capabilities across continuous tasks.

REFERENCES

[1] D. Zambrano-Gutierrez, J. Cruz-Duarte, and H. Castañeda, "Automatic hyper-heuristic to generate heuristic-based adaptive sliding mode controller tuners for buck-boost converters," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1482–1489, 2023.

[2] S. Fong, S. Deb, and A. Chaudhary, "A review of metaheuristics in robotics," *Computers & Electrical Engineering*, vol. 43, pp. 278–291, 2015.

[3] C. Caraveo, F. Valdez, and O. Castillo, "Optimization of fuzzy controller design using a new bee colony algorithm with fuzzy dynamic parameter adaptation," *Applied Soft Computing*, vol. 43, pp. 131–142, 2016.

[4] S. Pashaei and M. Badamchizadeh, "A new fractional-order sliding mode controller via a nonlinear disturbance observer for a class of dynamical systems with mismatched disturbances," *ISA transactions*, vol. 63, pp. 39–48, 2016.

[5] R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," *Computers & Chemical Engineering*, vol. 139, p. 106886, 2020.

[6] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Bridging the gap between value and policy based reinforcement learning," *Advances in neural information processing systems*, vol. 30, 2017.

[7] M. Yu and S. Sun, "Policy-based reinforcement learning for time series anomaly detection," *Engineering Applications of Artificial Intelligence*, vol. 95, p. 103919, 2020.

[8] T. Liu, L. Li, G. Shao, X. Wu, and M. Huang, "A novel policy gradient algorithm with pso-based parameter exploration for continuous control," *Engineering Applications of Artificial Intelligence*, vol. 90, p. 103525, 2020.

[9] M. H. Sazli, "A brief review of feed-forward neural networks," *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, vol. 50, no. 01, 2006.

[10] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 2004.

[11] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, 09 2015.

[12] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[13] Z. Zhang, J. Chen, Z. Chen, and W. Li, "Asynchronous episodic deep deterministic policy gradient: Toward continuous control in computationally complex environments," *IEEE Transactions on Cybernetics*, vol. 51, no. 2, pp. 604–613, 2021.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[15] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *arXiv preprint arXiv:1712.06567*, 2017.

[16] K. W. Pretorius and N. Pillay, "Population based reinforcement learning," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, 2021.

[17] N. Grinsztajn, D. Furelos-Blanco, and T. D. Barrett, "Population-based reinforcement learning for combinatorial optimization," *arXiv preprint arXiv:2210.03475*, 2022.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[19] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 01 2016.

[20] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *2012 American Control Conference (ACC)*, pp. 2177–2182, 2012.

[21] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 2018.

[22] A. Flajolet, C. B. Monroc, K. Beguir, and T. Pierrot, "Fast population-based reinforcement learning on a single machine," in *International Conference on Machine Learning*, pp. 6533–6547, PMLR, 2022.

[23] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, 1983.

[24] R. V. Florian, "Correct equations for the dynamics of the cart-pole system," in *Center for Cognitive and Neural Studies (Coneural)*, 08 2005.

[25] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[26] M. Jain, V. Saihjpal, N. Singh, and S. B. Singh, "An overview of variants and advancements of pso algorithm," *Applied Sciences*, vol. 12, no. 17, p. 8392, 2022.