Integrating Reinforcement Learning and Optimization Task: Evaluating an Agent to Dynamically Select PSO Communication Topology

Rodrigo Cesar Lira[1](✉), Mariana Macedo[2], Hugo Valadares Siqueira[3], and Carmelo Bastos-Filho[1]

[1] University of Pernambuco, Pernambuco, Brazil
{rcls,carmelofilho}@ecomp.poli.br
[2] University of Toulouse, Occitania, France
mmacedo@biocomplexlab.org
[3] Federal University of Technology, Paraná, Brazil
hugosiqueira@utfpr.edu.br

Abstract. A recent study catalogued hundreds of meta-heuristics proposed over the past three decades in Swarm Intelligence (SI) literature. This scenario makes it difficult for the practitioner to choose the most suitable meta-heuristic (RL) for a specific problem. This paper shows that Reinforcement Learning could be a powerful tool for SI. First, we describe a Reinforcement Learning environment to solve an optimization problem. Then, we investigate the usage of Proximal Policy Optimization to dynamically set the Particle Swarm Optimization topology accordingly to the simulation states. Our RL proposal reached competitive fitness values, even when evaluated in non-trained scenarios. In addition, we show the actions' distribution by simulation in the Rastrigin. The paper demonstrates how RL could be integrated to improve meta-heuristics capabilities, opening new research paths where RL will be used to improve meta-heuristics or select them accordingly to their strengths.

Keywords: Particle Swarm Optimization · Proximal Policy Optimization · Reinforcement Learning

1 Introduction

Swarm Intelligence (SI) is a branch of Computational Intelligence (CI) that comprises approaches inspired by the intelligent behaviour that arises in the interaction among living beings to solve complex problems [6]. These agents behave mostly without supervision, and their actions have a stochastic component according to the perception of their surroundings [5,23].

After three decades of the emergence of the research area, the number of meta-heuristics based on swarms increased from a few to hundreds [3]. Covid-19, African Buffalo, Buses and the FIFA World Cup are recent inspirations used in the proposal of new algorithms [3]. Knowing that even the same meta-heuristics can achieve weak results if we choose non-optimal hyperparameters. The current scenario makes complex the task of finding the most suitable meta-heuristic for each problem [10].

In order to overcome the manual and laborious task of choosing a meta-heuristic for a specific problem, a new approach based on meta-learning has emerged in the literature called Learning to Optimize (L2O) [13]. Recently, Yue et al. [4] created an L2O proposal by extending a point-based optimizer into a population-based optimizer. Gomes et al. [8] proposed the Learning-to-Optimize Partially Observable Markov Decision Process (LTO-POMDP), a framework based on reinforcement meta-learning to solve black-box problems.

On the other hand, we have Reinforcement Learning (RL) becoming an emergent machine learning tool to improve swarm intelligence capabilities. RL is a family of machine learning methods that can evolve from simple feedback. Seyyedabbasi [19,20] used a classical reinforcement learning technique, called Q-Learning, to guide the behaviour of several swarm intelligence algorithms, creating RL_{I-GWO}, RL_{Ex-GWO}, RL_{WOA} and RLSCSO. Wu et al. [24] employed reinforcement learning to improve PSO convergence, controlling the social and cognitive components. Almonacid integrated reinforcement learning and optimization problem, creating a method that automatically creates evolutionary meta-heuristics using reinforcement learning, named AutoMH [1]. Nevertheless, to the best of our knowledge, there is no approach to dynamically change the use of swarm communication topology using RL in the literature.

In this paper, we investigate the usage of Deep Reinforcement Learning (DRL) to dynamically set the Particle Swarm Optimization (PSO) [11] topology. We choose PSO because it is a well-known meta-heuristic widely used to solve several optimization problems [14,21]. In our approach, we first create a Reinforcement Learning environment where a PSO is integrated to solve an optimization task. Then, we train the RL agent to select the best-suited PSO topology. Finally, we evaluate the results in different scenarios.

This paper is divided as follows: Sect. 2 briefly describes Particle Swarm Optimization. Section 3 describes the methodology and parameterization for the experiments. Section 4 presents our findings and results. Lastly, Sect. 5 finishes the paper with some discussions and conclusions.

2 Particle Swarm Optimization

In 1995, Kennedy and Eberhart proposed Particle Swarm Optimization (PSO) [11], a novel algorithm to solve continuous optimization problems. PSO is composed of a set of particles (i.e., the agents) that move around the search space, influenced by the best solution that they individually have found (the cognitive component) and the best solution that any particle in their neighbourhood has found (social component).

In their movement, the particles update their velocity and position using Eq. 1 and Eq. 2, respectively [2].

$$\vec{v}_i(t+1) = \chi\big\{\vec{v}_i(t) + c_1 r_1[\vec{p}_i(t) - \vec{x}_i(t)] + \vec{c}_2 r_2[\vec{n}_i(t) - \vec{x}_i(t)]\big\} \tag{1}$$

$$\vec{x}_i(t+1) = \vec{v}_i(t+1) + \vec{x}_i(t), \tag{2}$$

where $\vec{x}_i(t)$ and $\vec{v}_i(t)$ are the position and velocity in the iteration t of the particle i, respectively; c_1 is the cognitive acceleration coefficient, and c_2 is the social acceleration coefficient; r_1 and r_2 are random uniform numbers; $\vec{p}_i(t)$, and $\vec{n}_i(t)$ are, respectively, the best position found by the particle, and the best position found by any neighbour until the current iteration. Finally, χ is the constrictor factor, a mechanism to ensure convergence [21] defined in Eq. 3.

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \varphi = c_1 + c_2. \tag{3}$$

PSO pseudocode is shown in Algorithm 1. While the stop criterion is not reached, each particle moves around the search space using Eq. 1 and 2. After each movement, the particle updates its best position, and the best position found by the neighbours is also updated.

Algorithm 1. PSO Pseudocode

```
1: Initialize the population randomly
2: while stop criterion is not reached do
3:     for each particle do
4:         Update particle velocity
5:         Update particle position
6:         Evaluate particle position
7:         Update $\vec{p}_i$ and $\vec{n}_i$
8:     end for
9: end while
10: Return the best solution
```

In the PSO, the communication topology is an essential hyperparameter because it defines the set of particles that will share information (neighbourhood). In a fully connected topology, the agents will quickly share information. A topology with limited sharing delays information propagation. Thus, communication topology directly affects the flow of information within the swarm and, consequently, the algorithm's performance.

Global (*gbest*) and Local (*lbest*) are two well-known topologies in the literature [21]. For the Global topology, each particle broadcasts information to the whole swarm. For the Local topology, each particle has a limited number of neighbours with whom they can communicate [16]. Local and Global topologies are static, meaning the neighbours do not change over the iterations.

Oliveira et al. proposed a new topology based in a dynamic neighbourhood [15]. The topology uses a mechanism that creates new connections between particles when the swarm stagnates. Every iteration, the particle that does not improve its $\vec{p}_i(t)$, increments a counter (p_k-$failure$) by one. When a threshold (p_k-$failure^T$) is reached, the particle modifies its neighbourhood by connecting to a new particle selected by the roulette wheel. p_k-$failure$ is set to zero in two situations: (i) every iteration where the particle improves its solution or (ii) after a new connection. On average, the dynamic topology showed better results than Local and Global.

3 Methodology

We developed a reinforcement learning environment using OpenAI Gym that encapsulates a PSO simulation to solve the optimization problem. The environment is one of the main components of a Reinforcement Learning task. In a trial-and-error process, the RL agent interacts with it, selecting an action and receiving feedback.

The RL agent is integrated into the PSO simulation by setting the PSO communication topology in our experiments. The agent acts in the environment by choosing Local, Global, or Dynamic topology and evaluating the improvement in the simulation. An action is taken every *step_ decision* iterations, an environment hyperparameter. We used *step_ decision=5*, meaning an action after five PSO iterations. There are 200 time steps where the RL agent interacts with the environment since in the PSO simulation, we used $max\_iteration = 1000$ as the stop criterion. In our preliminaries simulations, *step_ decision=5* performed better and faster than *step_ decision=1*. Figure 1 depicts our simulation using the RL environment.

While the stop criterion is not reached, i.e. the episode is not over, the agent acts in the environment selecting the topology. Next, the agent receives a set of states representing the current simulation scenario and the reward that measures the action's effectiveness after a run of 5 iterations. The reward is calculated using the function proposed by Schuchardt et al. [17], showed in Eq. 4. It considers that multiple minor improvements should have the same effect as one considerable improvement to find the best solution.

$$reward(t) = \alpha_r \log_{10} \frac{F_{\max}(t)}{F_{\max}(t-1)} \tag{4}$$

where α_r is a scale factor that is set to 1 as indicated by the authors for solving continuous optimization, and $F_{\max}(t)$ is the fitness value of the fittest particle in time step t. The proposed reward function maximizes the maximum fitness gain between actions. However, as we deal with the problem as a minimization task, we transform the fitness using Eq. 5 [17].

$$F(x_i) = \frac{1}{\max(f(x_i), 10^{-20})} \tag{5}$$

where $f(x)$ is the value of our fitness function for the particle x_i. max() is used to avoid math errors in the division when the fitness is zero. Therefore, the reward function works in minimization and maximization problems.

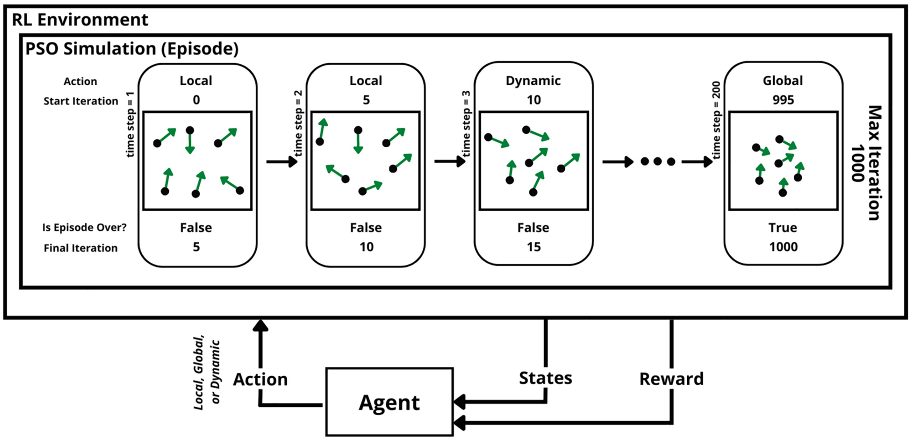

Fig. 1. Reinforcement Learning environment created for solving an optimization problem through Particle Swarm Optimization. Every time step, the RL agent selects a topology and evaluates its effectiveness.

We used a set of features retrieved from the simulation to inform the agent about the state. We based on a set of variables described in previous works [12,22], e.g. normalized euclidean distance among agents, remaining simulation budget, and the number of agents improved in the previous iterations. Additionally, we included a new feature that maps the topology chosen in the current action.

We choose Proximal Policy Optimization (PPO), a Reinforcement Learning agent famous for its stability and reliability [18]. We used PPO implementation from RLLib with default hyperparameters, being $\gamma = 0.99$ (discount factor), $timesteps\_per\_iteration = 20{,}000$, $train\_batch\_size = 4{,}000$, $num\_sgd\_iter = 30$, $clip\_param = 0.3$, $KL\_init = 0.2$, $KL\_target = 0.01$, $GAE\_parameter = 1.0$, $vf\_loss\_coeff = 1.0$, $entropy\_coeff = 0$, $learning\_rate =$ 5e-05. During our preliminaries simulations, we also trained the Rainbow agent [9], but we removed it from our setup due to the lack of competitive results, supposedly a tuning problem.

We trained PPO during 500 iterations (epochs) to solve the Rastrigin function in two scenarios, 10 and 30 dimensions. In each training iteration, we run 600 PSO simulations (episodes). In the PSO simulation we used 20 particles, $c_1 = c_2 = 2.05$, $p_k\text{-}failure^T = 1$. Local topology allows communication with two neighbours (i.e., Ring topology). We used standard particle swarm optimization, but our proposal is not PSO version-dependent. We can switch to a different PSO version by simply changing the PSO code.

After training, we tested the agent in Rastrigin, Shifted Rastrigin and Shifted Rotated Rastrigin benchmark functions [7]. For each scenario, we run 50 simulations, collecting the best fitness and sequence of actions the RL agent takes.

4 Results

We present our results organized into two parts. First, we show the performance of PSO using RL to choose the topology dynamically compared to PSO using Local, Global or Dynamic topologies. Next, we evaluate the actions chosen by the approach using RL agent in Rastrigin.

Figure 2 shows the results found in Rastrigin with 10 (A) and 30 (B) dimensions, respectively. The box plots indicate that our RL approach found competitive results in each function, indicating that PPO learned to choose the suited PSO topology according to the simulation state.

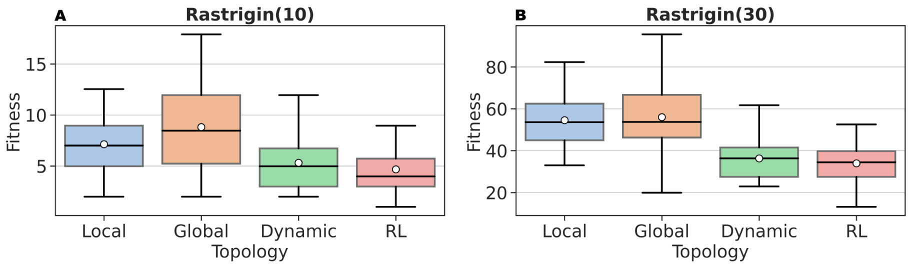

Fig. 2. Box plot of fitness values reached by our proposal (RL), and PSO using Local, Global and Dynamic topology in Rastrigin with 10 dimensions.

Even training only in Rastrigin, we tested the proposal in two variations to evaluate the model generalization when used in a non-simulated scenario. Figures 3, and 4 show the results found in the scenarios using Shifted Rastrigin and Shifted Rotated Rastrigin with 10 and 30 dimensions. Even though the RL agent was not trained with rotation and shifting, it could select good sequences of topologies that took it to reach low fitness values in these scenarios, even when we increase the complexity (Figs. 3 B and 4 B), we see our RL approach reaching good fitness results when compared with PSO using Local, Global or Dynamic topologies.

Figure 5 shows the convergence in the most complex trained scenario (30 dimensions). We see the same behaviour in Rastrigin with 10 dimensions. Our proposal and Dynamic topology in both scenarios reach the best solution around the final iteration. Differently, all approaches converge quickly to the final solution in Shifted Rastrigin and Shifted Rotated Rastrigin (non-trained scenarios).

We applied the Wilcoxon test to compare the efficiency across topologies using a confidence rate of 95%, as shown in Table 1. In the table, '–' indicates

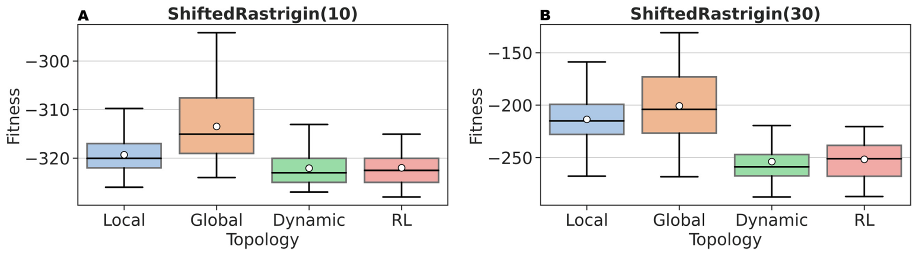

Fig. 3. Box plot of fitness values reached by our proposal (RL), and PSO using Local, Global and Dynamic topology in Shifted Rastrigin with 10 (A) and 30 (B) dimensions.

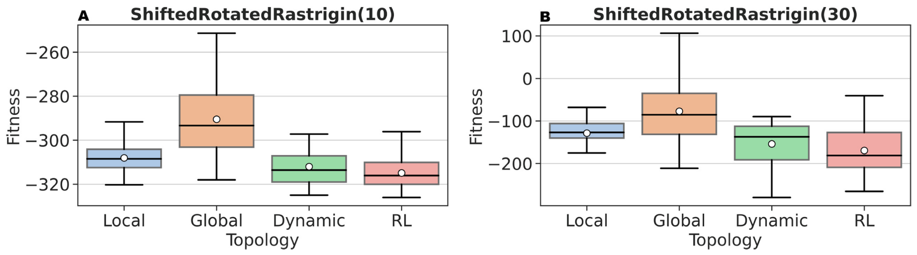

Fig. 4. Box plot of fitness values reached by our proposal (RL), and PSO using Local, Global and Dynamic topology in Shifted Rotated Rastrigin with 10 (A) and 30 (B) dimensions.

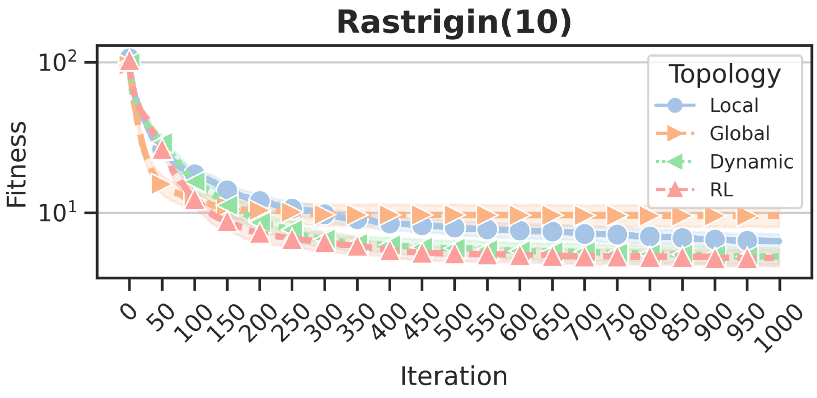

Fig. 5. Fitness per iteration for RL, and PSO using Local, Global and Dynamic topology in Rastrigin with 30 dimensions.

that there is no statistical difference between the results found by the topologies, '▲' indicates the proposal achieved better results than the other topology, and '▼' represents that our proposal reached worse results than the topology compared. We can conclude from this analysis that the RL approach performs better than Local and Global topologies for every simulated scenario. Our proposal overcomes the Dynamic topology only in Shifted Rotated Rastrigin with 30 dimensions. While in other scenarios, they are equivalent, with no statistical differences.

Table 1. Results of fitness values and Wilcoxon test with a confidence level of 95% comparing the RL with the other algorithms using 10 and 30 dimensions in Rastrigin, Shifted Rastrigin and Shifted Rotated Rastrigin.

		10 dimensions			
Function		RL	Local	Global	Dynamic
Rastrigin	Mean Fitness	4.68	7.14	8.82	5.31
	STD	2.30	3.01	4.17	2.70
	Wilcoxon		▲	▲	–
Shifted Rastrigin	Mean Fitness	−322.00	−319.32	−313.48	−322.08
	STD	4.59	3.81	7.65	3.86
	Wilcoxon		▲	▲	–
Shifted Rotated Rastrigin	Mean Fitness	−314.88	−308.07	−290.47	−311.99
	STD	6.93	6.48	16.11	10.00
	Wilcoxon		▲	▲	–
		30 dimensions			
Function		RL	Local	Global	Dynamic
Rastrigin	Mean Fitness	33.94	54.56	56.01	36.28
	STD	8.39	11.94	15.59	8.84
	Wilcoxon		▲	▲	–
Shifted Rastrigin	Mean Fitness	−251.67	−213.52	−200.68	−253.84
	STD	18.57	23.77	35.50	23.76
	Wilcoxon		▲	▲	–
Shifted Rotated Rastrigin	Mean Fitness	−169.35	−128.81	−77.30	−153.95
	STD	56.12	32.31	72.35	51.61
	Wilcoxon		▲	▲	▲

In addition, we evaluated the set of actions taken in each simulation and created Fig. 6. Figure 6 A shows that Dynamic topology was predominantly chosen in Rastrigin with ten dimensions. Nevertheless, in Rastrigin with 30 dimensions, Local and Dynamic topologies were selected almost similarly during the 200-time steps (Fig. 6 B). As the dimensions increase, the complexity grows, forcing the agent to behave differently.

Figure 7 shows the actions' distribution by simulation in Rastrigin. In Fig. 7 A, Dynamic (blue bar) is the most selected topology in each simulation. In turn, Local (green bar) and Dynamic are selected in a similar quantity in Fig. 7 B. Global topology (orange bar) is the less frequent choice in both scenarios.

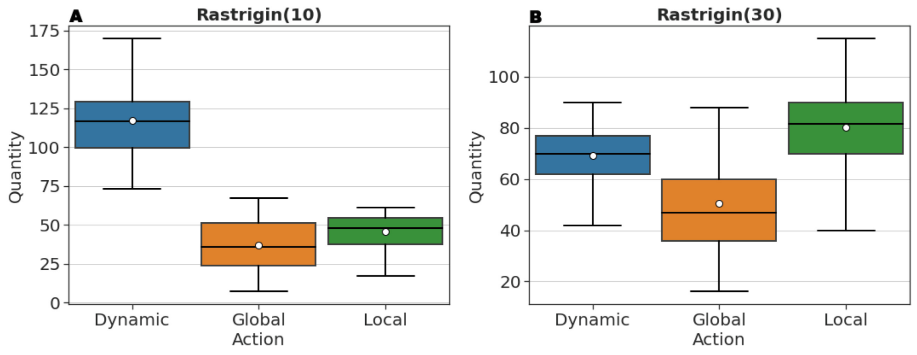

Fig. 6. Box plot of actions taken by RL agent in Rastrigin with 10 (A) and 30 (B) dimensions.

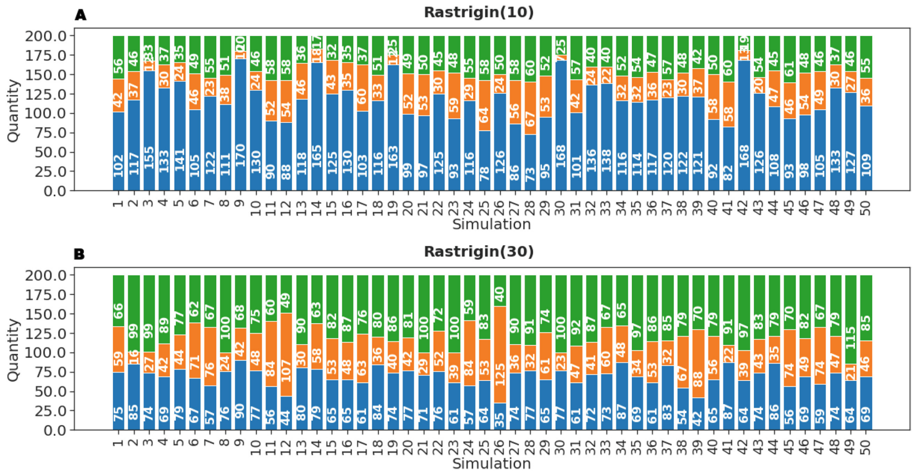

Fig. 7. Actions' distribution by simulation in Rastrigin with 10 (A) and 30 (B) dimensions.

5 Conclusions

We used Proximal Policy Optimization to set the Particle Swarm Optimization topology dynamically. Firstly, we developed an RL environment with a PSO simulation where the RL agent selects topology every five iterations. Then, we trained the agent using the Rastrigin function with 10 and 30 dimensions.

The proposal reached competitive results, overcoming PSO using Local and Global topology. Both scenarios' results were statistically equivalent compared to PSO using Dynamic topology. Even when evaluated in a non-simulated scenario, such as Shifted Rastrigin and Shifted Rotated Rastrigin, the RL approach reached good results, showing generalization capability. We also showed the choices' distribution by simulation using the Rastrigin benchmark function.

The results lead us to believe that the level of the problem's complexity forces the agent to act differently.

Despite our limited scenarios, the results found in this paper are essential for creating the foundation of new research paths where RL will be used to improve meta-heuristics or select them accordingly to their strengths. We utilized the standard PSO algorithm in our simulations, but our method could be applied to any PSO version. We aim to add new problem features in the environment state for future work to improve the agent's generalization. In addition, we plan to investigate in-depth the impact of simulation hyperparameters in different scenarios.

Acknowledgements. The authors thank the Brazilian National Council for Scientific and Technological Development (CNPq), processes number 40558/2018-5, 315298/2020-0, and Araucaria Foundation, process number 51497, for their financial support. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

1. Almonacid, B.: Automh: Automatically create evolutionary metaheuristic algorithms using reinforcement learning. Entropy **24**(7) (2022). https://doi.org/10.3390/e24070957
2. Belotti, J.T., et al.: Air pollution epidemiology: A simplified generalized linear model approach optimized by bio-inspired metaheuristics. Environ. Res. **191**, 110106 (2020). https://doi.org/10.1016/j.envres.2020.110106
3. Campelo, F., Aranha, C.: EC Bestiary: A bestiary of evolutionary, swarm and other metaphor-based algorithms (2018). https://doi.org/10.5281/zenodo.1293352
4. Cao, Y., Chen, T., Wang, Z., Shen, Y.: Learning to optimize in swarms. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook (2019)
5. Chandrasekaran, K., Simon, S.P., Padhy, N.P.: Binary real coded firefly algorithm for solving unit commitment problem. Inf. Sci. **249**, 67–84 (2013). https://doi.org/10.1016/j.ins.2013.06.022
6. Cruz, D.P.F., Maia, R.D., De Castro, L.N.: A critical discussion into the core of swarm intelligence algorithms. Evolution. Intell. **12**(2), 189–200 (2019). https://doi.org/10.1007/s12065-019-00209-6
7. Ding, K., Tan, Y.: A cuda-based real parameter optimization benchmark (2014). https://doi.org/10.48550/ARXIV.1407.7737
8. Gomes, H.S., Léger, B., Gagné, C.: Meta learning black-box population-based optimizers (2021). https://doi.org/10.48550/ARXIV.2103.03526
9. Hessel, M., et al.: Rainbow : Combining Improvements in Deep Reinforcement Learning (2013)
10. Houssein, E.H., Gad, A.G., Hussain, K., Suganthan, P.N.: Major advances in particle swarm optimization: Theory, analysis, and application. Swarm Evolution. Comput. **63**, 100868 (2021). https://doi.org/10.1016/j.swevo.2021.100868
11. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks. vol. 4, pp. 1942–1948 (1995). https://doi.org/10.1109/ICNN.1995.488968

12. de Lacerda, M.G.P., de Lima Neto, F.B., Ludermir, T.B., Kuchen, H.: Out-of-the-box parameter control for evolutionary and swarm-based algorithms with distributed reinforcement learning. Swarm Intell. (2023). https://doi.org/10.1007/s11721-022-00222-z
13. Li, K., Malik, J.: Learning to optimize (2016). https://doi.org/10.48550/ARXIV.1606.01885
14. Macedo, M., et al.: Overview on binary optimization using swarm-inspired algorithms. IEEE Access **9**, 149814–149858 (2021). https://doi.org/10.1109/ACCESS.2021.3124710
15. Oliveira, M., Bastos-Filho, C.J.A., Menezes, R.: Using network science to define a dynamic communication topology for particle swarm optimizers. In: Complex Networks, pp. 39–47. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-30287-9_5
16. Santos, P., et al.: Application of PSO-based clustering algorithms on educational databases. In: 2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI), pp. 1–6. IEEE (2017). https://doi.org/10.1109/LA-CCI.2017.8285690
17. Schuchardt, J., Golkov, V., Cremers, D.: Learning to evolve (2019). https://doi.org/10.48550/ARXIV.1905.03389
18. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017). https://doi.org/10.48550/ARXIV.1707.06347
19. Seyyedabbasi, A.: A reinforcement learning-based metaheuristic algorithm for solving global optimization problems. Adv. Eng. Softw. **178**, 103411 (2023). https://doi.org/10.1016/j.advengsoft.2023.103411
20. Seyyedabbasi, A., Aliyev, R., Kiani, F., Gulle, M.U., Basyildiz, H., Shah, M.A.: Hybrid algorithms based on combining reinforcement learning and metaheuristic methods to solve global optimization problems. Knowl. Based Syst. **223**, 107044 (2021). https://doi.org/10.1016/j.knosys.2021.107044
21. Shami, T.M., El-Saleh, A.A., Alswaitti, M., Al-Tashi, Q., Summakieh, M.A., Mirjalili, S.: Particle swarm optimization: A comprehensive survey. IEEE Access **10**, 10031–10061 (2022). https://doi.org/10.1109/ACCESS.2022.3142859
22. Sharma, M., Komninos, A., López-Ibáñez, M., Kazakov, D.: Deep reinforcement learning based parameter control in differential evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19), pp. 709–717. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3321707.3321813
23. Shunmugapriya, P., Kanmani, S.: A hybrid algorithm using ant and bee colony optimization for feature selection and classification (ac-abc hybrid). Swarm Evolution. Comput. **36**, 27–36 (2017). https://doi.org/10.1016/j.swevo.2017.04.002
24. Wu, D., Wang, G.G.: Employing reinforcement learning to enhance particle swarm optimization methods. Eng. Optimiz. **54**(2), 329–348 (2022). https://doi.org/10.1080/0305215X.2020.1867120