Applying Reinforcement Learning for Multiple Functions in Swarm Intelligence

André A. V. Escorel Ribeiro[1], Rodrigo Cesar Lira[1(✉)], Mariana Macedo[2], Hugo Valadares Siqueira[3], and Carmelo Bastos-Filho[1]

[1] University of Pernambuco, Pernambuco, Brazil
{aaver,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. Swarm intelligence (SI) algorithms have become popular due to their self-learning characteristics and adaptability to external changes. They can find reasonable solutions to complex problems without in-depth knowledge. Much of the success of these algorithms comes from balancing the exploration and exploitation tasks. This work evaluates the application and performance of a reinforcement learning approach applied to a well-known swarm intelligence algorithm, Particle Swarm Optimization (PSO). We use the reinforcement learning agent Proximal Policy Optimization (PPO) to dynamically change the swarm communication topology according to the problem. We analyze the PSO's behavior, influenced by the reinforcement learning agent, through methods such as interaction networks and fitness analysis. We show that the RL approach can transfer the knowledge learned from one function to other functions, and that dynamic changes of topology over time makes PSO much more efficient than setting only one specific topology, even when using a Dynamic topology. Our results then suggest that changing topologies might be more efficient than having a Dynamic topology, and that indeed Local and Global topologies have an important role in the best swarm performance. Our results take a step further on explaining the performance of SI and automatizing their use for non-experts.

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

1 Introduction

Swarm Intelligence (SI) is a branch of Computational Intelligence in which a collective behavior is exhibited by a group of decentralized and self-organized simple reactive agents interacting with each other and the environment. The

M. C. Naldi and R. A. C. Bianchi (Eds.): BRACIS 2023, LNAI 14196, pp. 197–212, 2023.
https://doi.org/10.1007/978-3-031-45389-2_14

interaction among them generates a collective adaptation to allow them to solve complex problems [7]. The simple reactive agents are represented by positions in the search space with simple historical memories that explore it while exchanging information about their experiences with other agents. Based on the individual's experience and the information received by other members of the swarm, the agents can adapt the behavior in the search space and, with enough time, find and refine reasonable solutions to the presented problems. Swarm-based algorithms emerge as an alternative to classical optimization methods in high-dimensional optimization problems [1]. They are an excellent alternative for optimization problems since they considerably reduce the computational cost and do not require a complete understanding of the problem regarding the characteristics of the search space.

Many SI algorithms are based on animal social behavior metaphors, such as Ant colony optimization (ACO) [2] inspired by the behavior of ant colonies, Particle swarm optimization (PSO) [6] inspired by a flock of birds, Artificial Bee Colony (ABC) [5] inspired by the bee hives. Swarm-based meta-heuristics are applied in several problems, such as nuclear engineering [18] and diagnosing diseases [14], among others. We also find applications to solve tasks related to data science and image and signal processing [13,16,19].

Many efforts have been made to improve the performance of swarm-based algorithms. In the case of PSO, Xu et al. [21], and Wu et al. [20] suggested the application of reinforcement learning due to this method's characteristic of being able to learn which actions are best for a given state. It allows the dynamic modification of the behavior of the PSO through the adjustment of communication topology using a reinforcement learning agent, generating better results for complex problems and increasing the convergence speed. Recently, Lira et al. [8] proposed a self-adaptive metaheuristic that considers the real-time information acquired during execution. For the algorithm to adapt, a reinforcement learning agent collects information and chooses actions that modify the metaheuristic's behavior. Despite the good preliminary results presented by reinforcement learning to create advanced approaches for swarm-based algorithms, it needs to be clarified if behaviors learned in some scenarios could be adapted in other scenarios not experienced by the algorithms.

This paper evaluates the transfer learning capability of a reinforcement learning strategy when different topologies can be selected along the optimization with a PSO algorithm, seeking to understand how the changes influence the various observed metrics and providing information about the learning of reinforcement agents and possible patterns that can be found. We find that RL is able to transfer the knowledge from one function to the other functions, and that changing topologies can be more effective than using a dynamic topology for PSO.

This paper is divided as follows: Sect. 2 briefly describes Particle Swarm Optimization, Proximal Policy Optimization, and Interaction Networks. Section 3 describes the methodology and parameterization for the experiments. Section 4 presents our findings and results, and we finish in Sect. 5 with our conclusions.

2 Background

2.1 Particle Swarm Optimization

In 1995, Kennedy and Eberhart [6] proposed Particle Swarm Optimization (PSO) after observing the social behavior of flocks of animals, such as birds. PSO is one of the metaheuristics of swarm intelligence most known and used in the literature [9]. Particle Swarm Optimization consists of a group of simple agents, called particles, that will be scattered in a search space. While a stopping criterion is not reached, the particles update their velocities and positions at each iteration, keeping the information on the best solutions found by them ($\vec{p}_i$) and the best solutions found by their neighbors ($\vec{n}_i$). This information is used to calculate its next movement within the search space. In each PSO iteration, the velocity (v_i) and position (x_i) information of each particle is updated according to Eqs. 1 and 2. Eventually, with enough iterations, the swarm will likely return a solution approaching the optimum position in the search space.

$$\vec{v}_i(t+1) = \chi\Big\{\vec{v}_i(t) + c_1\epsilon_1[\vec{p}_i(t) - \vec{x}_i(t)] + c_2\epsilon_2[\vec{n}_i(t) - \vec{x}_i(t)]\Big\} \tag{1}$$

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

c_1 and c_2 are the acceleration coefficients, ϵ_1 and ϵ_2 are uniform random numbers, and χ is the constriction factor defined by $\chi = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|}$, where $\varphi = c_1+c_2$.

The communication topology defines the neighborhood in PSO. It describes the relations among particles, influencing the way the swarm behaves. Global (gbest) and Local (lbest) are two well-known topologies for PSO. Global is a fully connected topology where all particles can communicate with the entire swarm, allowing the best solution found to be shared quickly in the entire swarm. The Global topology causes a quick convergence and may not adequately explore the search space. For the Local topology, each particle is connected to k immediate particles in the swarm. It creates a ring-like communication structure using k=2. The particles have information only from the particles next to them so that the sub-swarms can independently converge on several optimal points. Local topology has slower convergence but allows a better exploration of the search space [7]. However, These two topologies are more suitable for specific problems, which led Oliveira et al. [4] to develop a more balanced topology that operates adaptively. In this approach, stagnant particles look for better particles to communicate. This approach adds a new attribute to the particle, called $p_k\text{-}failure$, which is incremented every iteration that the particle does not improve its fitness. If the $p_k\text{-}failure$ value exceeds a threshold ($p_k\text{-}failure^T$), the particle looks for a new neighbor to communicate. The choice of the neighbor for the particle to communicate with is probabilistic, based on roulette wheel selection, so that particles with better fitness have more chances of being chosen.

2.2 Reinforcement Learning

Reinforcement learning (RL) refers to learning which action should be taken in a situation (i.e., state) to achieve one or more goals [14]. At each iteration, a reinforcement learning agent receives observations of the current state scenario and takes an action from a list of actions allowed in that problem. After the action, the agent receives a reward, which measures whether the action was beneficial. Through trial and error, the agent maps and learns the actions that obtained the best rewards for the observed states, seeking to choose the ones that maximize the accumulative reward.

In reinforcement learning, the agent's strategy to map the relationship between the observed states and the actions that must be taken is called policy. The agent aims to find the best strategy (i.e., policy) within the environment that maximizes the reward function. Therefore, the agent can learn which policy works better for a given environment. Reinforcement learning has been used with optimization meta-heuristics, including PSO [14], seeking to improve the algorithm's convergence speed. For applications in continuous and complex problems where mapping the set of states and actions is difficult, it is possible to use a deep reinforcement learning approach. The name deep reinforcement comes from deep learning because of the use of deep neural networks to map the set of states and actions.

Proximal Policy Optimization (PPO) Schulman et al. [17] proposed the Proximal Policy Optimization (PPO), a policy gradient method more stable, efficient, and more straightforward than other predecessors, as Trust Region Policy Optimization (TRPO). PPO works to improve a policy, performing slight modifications. Its main improvements are using clipped surrogate objective, value function clipping, reward and layer scaling, orthogonal, and Adam learning rate [3]. PPO performed well in multiple benchmark problems for Reinforcement Learning [17].

2.3 Interaction Network

Oliveira et al. [11] proposed the Interaction Network (IN) aiming to understand the swarm dynamics better. IN is a framework that assesses the flow of information generated from the agents' interactions. The Interaction Network captures the exchange of information between agents, seeking to understand how the swarm influence each other. Oliveira et al. [12] have demonstrated that using interaction networks helps to compare, for instance, the balance between exploration and exploitation tasks across algorithms. IN is represented using a graph where each node represents an agent, and the edges represent the interactions between agents. The edges can be modelled in many ways, here, we modelled as shown in Eq. 3. In this network, we do not consider how much one particle influenced each other, but who influenced whom over time that signs the communication topology structure [10].

$$I_{t_{i,j}} = \begin{cases} 1 \text{ if } j \, \epsilon \, \vec{n}_i \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

IN can be evaluated individually or by the accumulation of successive networks. Iterations can be accumulated using a Time Window (TW) to capture the social interactions among agents in a frequency of iterations. The TW allows an analysis of which agents were neighbors to each other within an interaction interval so that large time windows make the interaction networks show the interactions that are most repeated. Short TWs contain the most recent interactions, while TWs = 1 contain instant interactions.

3 Methodology

We used the reinforcement learning framework for swarm intelligence, created by Lira et al. [8] based on Python programming language and RLLLib[1]. We chose the PSO in our experiments since it is a widely known and deployed swarm intelligence metaheuristics in the literature [9].

The simulation runs in time steps, allowing different swarm configurations. At the beginning of each time step, the RL agent acts, selecting Local, Global, or Dynamic topology. After a preset number of iterations, the RL agents evaluate the reward and the new simulation state before starting the same cycle until the stop criterion is reached. We used this preset number of iterations equal to 10 in this paper. At the end of the training stage, the agent should be able to recommend the best topologies for functions with similar characteristics. We used PPO [17] as the reinforcement learning agent to solve this problem. The RL agent is responsible for learning the topologies that best adapt to the tested functions and modifying the topology in the PSO based on the characteristics learned during execution.

We used simple and widely used functions to evaluate the algorithms in unimodal and multimodal scenarios for simplicity and first validation. Yet, it is challenging to appropriately cover all needed scenarios to evaluate a methodology [15], we argue that these scenarios are well-explored in the literature in regard to performance [9]. Thus, we selected Schwefel 2.21, Sphere, and Schwefel 2.22 unimodal functions, and Rastrigin, Griewank, and Schwefel multimodal functions. The search space chosen for these functions is based on Plevris and Solozano [15] (Table 1). Then, two instances of RL agents were trained for each type of function per scenario. We trained with Rastrigin for multimodal functions, and we used Schwefel 2.22 for unimodal functions. We expect that the RL learns the characteristics needed from each type of function by training only in one example.

A given metaheuristic may perform well on a function with few dimensions and poorly on a function with multiple dimensions. This problem is called the

[1] https://docs.ray.io/en/latest/rllib/index.html.

Table 1. Description of the benchmark functions used in the simulations.

	Function	Search Space	Equation
Unimodal	Sphere	[−100, 100]	$\sum_{i=1}^{D} \lvert x_i^2 \rvert$
	Schwefel 2.21	[−100, 100]	$max(\lvert x_i \rvert), i \in \{0...\text{D-1}\}$
	Schwefel 2.22	[−100, 100]	$\sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=0}^{D} \lvert x_i \rvert$
Multimodal	Rastrigin	[−5.12, 5.12]	$\sum^{D}{}_{i=1} ({x_i}^2 - 10cos(2\pi x_i) + 10 \cdot D$
	Griewank	[−100, 100]	$\frac{1}{4000} \sum_{i=1}^{D} x^2 - \prod_{i=0}^{D} cos(\frac{x_i}{\sqrt{i}}) + 1$
	Schwefel	[−500, 500]	$418.9829 \cdot D - \sum_{i=1}^{D} x_i sin(\sqrt{\lvert x_i \rvert})$

"Curse of Dimensionality" – a well-known problem in data science that refers to the phenomena that arise when analyzing and organizing information in spaces of many dimensions that do not occur when few dimensions are implemented [15]. Due to this problem, two different scenarios were empirically chosen for the number of dimensions and particles, Scenario 1 with 20 particles and 50 dimensions and Scenario 2 with 10 particles and 25 dimensions. In both scenarios, we used 1000 iterations as the stop criterion for the PSO simulations. These values were chosen to guarantee convergence for multidimensional problems [15], and we point out that the number of particles is smaller than usually used in the literature in order to make the problem more complex with a smaller dimensionality. Additionally, in PSO, we used the parameters $c_1 = c_2 = 2.05$, and $p_k\text{-}failure^T = 1$.

For the evaluation, we selected three metrics: (i) fitness, which is an indication of algorithm success; (ii) the distribution of the selected topology among Local, Global, and Dynamic, i.e., which actions the Reinforcement Learning agent recommended the most; (iii) interaction networks (IN) [10], which will be used to observe the accumulated interactions of the agents during a defined time interval, allowing us to analyse the importance of the selected topologies over time. We execute the PSO without reinforcement, seeking to observe how the topologies perform in each chosen function. This topology performance information is used to compare the results obtained with the reinforcement, so the topologies with the best performance in the tested functions might be the majority of the agent's recommendations.

4 Results

We divided our results into three subsections. First, we show the fitness performance of the RL proposal compared with PSO using Global, Local, and Dynamic topologies. Next, we focus on our proposal, evaluating the topologies chosen in each scenario. Finally, we analyze the Interactions Networks in the RL approach.

4.1 Evaluating the Fitness

In Figs. 1 and 2, we present the boxplots of the best fitness values found in 50 simulations in each scenario. We see that RL applied to PSO reached, in general, better performance than using the PSO with a specific topology for six functions.

For Scenario 1 (Fig. 1), using unimodal functions, we see that the RL performed as well as the best communication topology found for each function. RL could have been more efficient for the multimodal functions. In Schwefel function, it reached the worst results. In Scenario 2 (Fig. 2), we see our approach reaching competitive results. However, the results showed again that the training in Rastrigin led RL to achieve bad results in Schwefel.

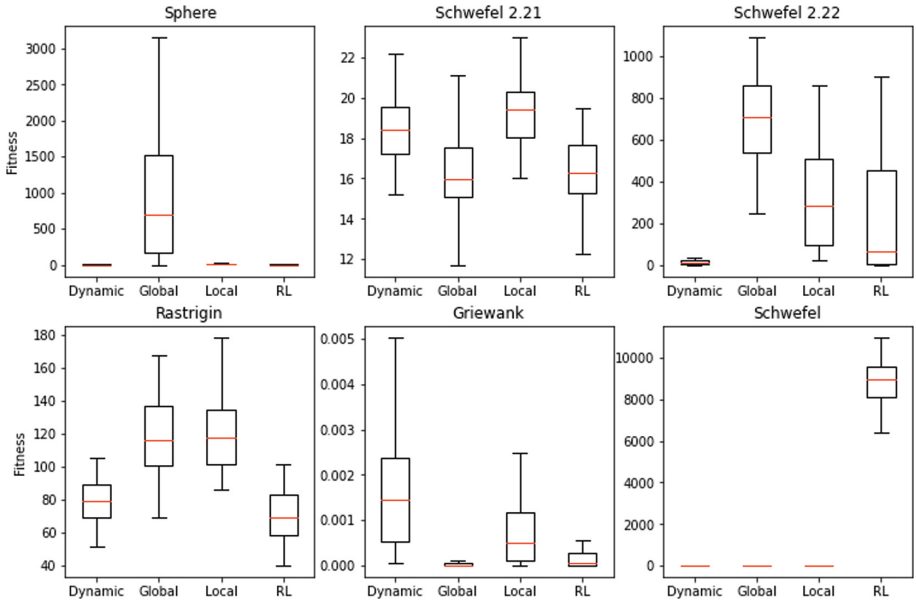

Fig. 1. Boxplot of the best fitness found on 50 simulations of each algorithm in Scenario 1.

We then compared the results using a signal-ranked Wilcoxon test with a confidence rate of 99.9% in Tables 2 and 3. '–' indicates no statistical difference between the solutions, '▲' indicates the RL approach achieved better results than the other algorithm, and '▼' represents that our proposal reached worse results than the algorithm compared. Based on the Wilcoxon test results, we can assure the RL capability for solving different functions, even when we train only one of them with similar characteristics. Only in Schwefel function it did not work well.

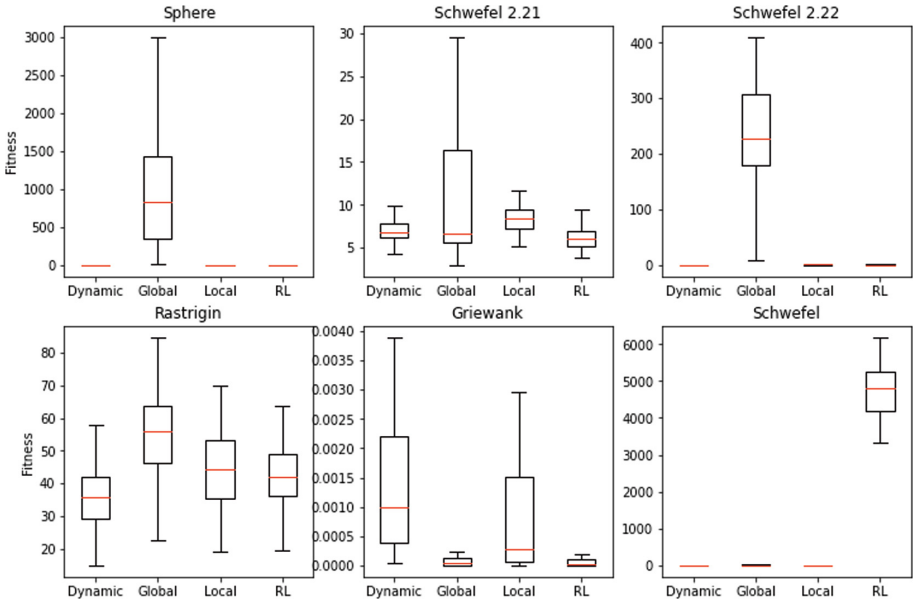

Fig. 2. Boxplot of the best fitness found on 50 simulations of each algorithm in Scenario 2.

Table 2. Results of fitness values and Wilcoxon test with a confidence level of 99.9% comparing the RL with the other algorithms for Scenario 1.

		Scenario 1			
Function		RL	Local	Global	Dynamic
Sphere	Mean Fitness	2.61	15.38	1163.9	2.92
	STD	4.72	9.87	1446.39	6.87
	Wilcoxon		▲	▲	–
Schwefel 2.21	Mean Fitness	16.25	19.25	16.17	18.46
	STD	1.67	1.39	1.83	1.65
	Wilcoxon		▲	–	▲
Schwefel 2.22	Mean Fitness	269.48	312.32	693.32	41.37
	STD	471.29	239.16	225.67	106.0
	Wilcoxon		–	▲	–
Rastrigin	Mean Fitness	71.90	121.60	115.97	80.0
	STD	16.93	23.80	23.97	14.30
	Wilcoxon		▲	▲	–
Griewank	Mean Fitness	0.000349	0.000764	0.040017	0.002407
	STD	0.000714	0.000855	0.279387	0.003118
	Wilcoxon		–	–	▲
Schwefel	Mean Fitness	8902.70	19.25	16.17	18.46
	STD	1031.57	1.39	1.83	1.65
	Wilcoxon		▼	▼	▼

Table 3. Results of fitness values and Wilcoxon test with a confidence level of 99.9% comparing the RL with the other algorithms for Scenario 2.

		Scenario 2			
Function		RL	Local	Global	Dynamic
Sphere	Mean Fitness	9.1e-07	3.5e-03	104.7	2.66e-04
	STD	3.0e-06	4.760e-03	1031.26	9.6e-05
	Wilcoxon		▲	▲	▲
Schwefel 2.21	Mean Fitness	6.28	8.46	12.3	7.05
	STD	1.51	1.71	10.73	1.53
	Wilcoxon		▲	–	–
Schwefel 2.22	Mean Fitness	10.9	58.55	242.55	0.15
	STD	68.82	126.82	114.5	0.44
	Wilcoxon		–	▲	–
Rastrigin	Mean Fitness	43.35	44.48	55.58	36.47
	STD	10.88	12.13	14.57	10.39
	Wilcoxon		–	▲	–
Griewank	Mean Fitness	2.45e-4	0.001318	0.051423	0.001681
	STD	0.000637	2.226e-3	0.2	0.001983
	Wilcoxon		▲	–	▲
Schwefel	Mean Fitness	4731.377060	8.5	12.3	7.05
	STD	733.83	1.71	10.73	1.53
	Wilcoxon		▼	▼	▼

4.2 Evaluation of the Selected Topologies

We now analyse the distribution of the selected topologies by RL over time steps. We expect that even by training on one example of a benchmark function (Schwefel 2.22 or Rastrigin), the RL agent will be able to learn a good policy for solving similar functions.

We plot the percentage of time that a topology was chosen over time step coupled with the best fitness evolution over iteration in Figs. 3 and 4. We can observe that in the first phase, "exploration phase", the Global topology was chosen most of the time, but the "exploitation phase" varies across experiments. The Dynamic topology was most chosen for the "exploitation phase" indicating that the swarm needs more diversity from the connections to improve the fitness. In the "exploration phase", being widely connected is more important than having a diverse set of connections. Therefore, regardless of being unimodal and multimodal functions, it might be true that diversity on the connections is better as the swarm starts to exploit.

The fitness improvement is larger while using the Global topology, but we argue that this is not due to the fact that this topology is more efficient for the swarm. Actually, this might be true because of the easiness of improving

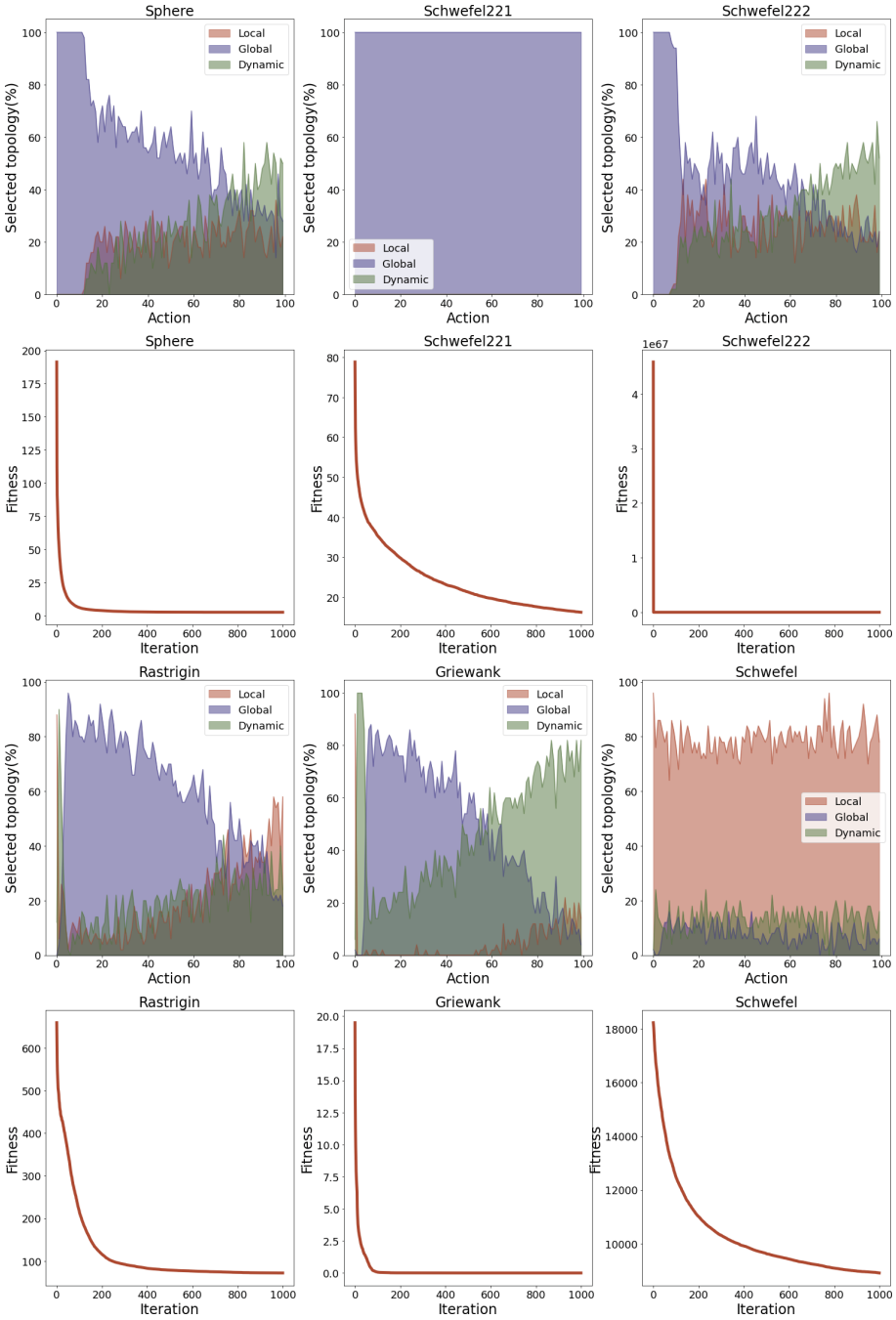

Fig. 3. Percentage of times that a topology was selected by the agent, with its respective fitness evolution on the bottom of each plot for Scenario 1.

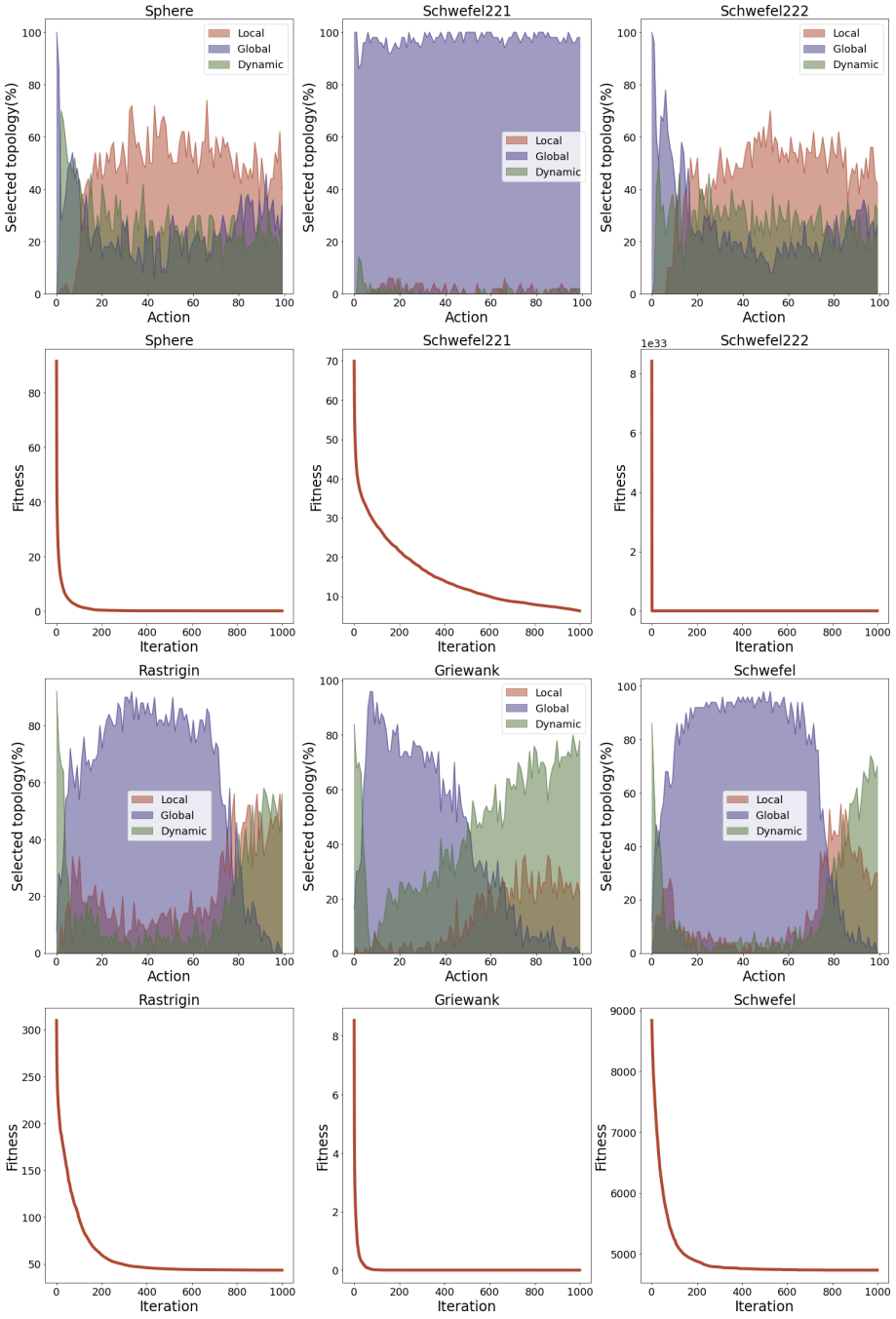

Fig. 4. Percentage of times that a topology was selected by the agent, with its respective fitness evolution on the bottom of each plot for Scenario 2.

in a "exploration phase". We see that for the Schwefel function the swarm did not converge, the chosen topology is the Local, corroborating with the literature that this topology works better than the Global topology for complex multimodal functions.

4.3 Analysing the Interaction Network

We are also interested in understanding how the agents influence each other in their movement over iterations. We use the cumulative Interaction Network (IN) to analyze the social interactions of the best simulation for each experiment in four-time windows: (i) between 0 and 99 iterations, (ii) between 100 and 199 iterations, (iii) between 200 and 299 iterations, and (iv) between 300 and 999 iterations, shown in Figs. 5 and 6. Each line represents the intensity of the influence of one particle on the displacement of the other particles. Therefore, strong lines (yellow-red) indicate particles that strongly influence the swarm, and strong columns represent particles that are strongly influenced by the swarm. We can identify which topology impacted the most across time windows by analyzing the networks. In Sect. 4.2, we see which topologies were more frequently chosen across simulations; here, we can observe which topologies impacted the most on the movement for the best experiments. If we observe strong diagonal lines and

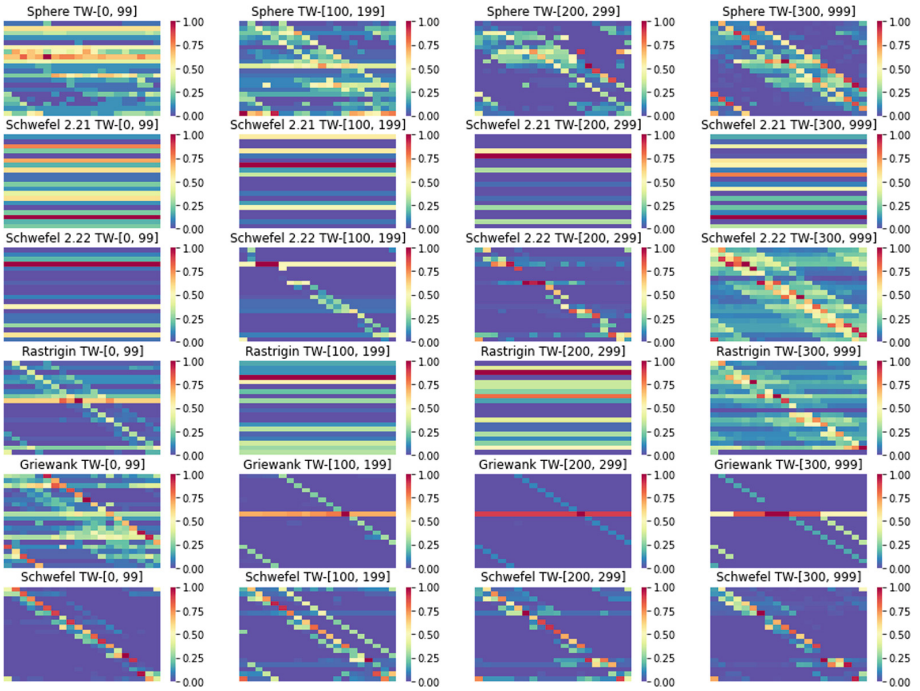

Fig. 5. Interaction Network generated from the simulation with the best fitness for each function of Scenario 1.

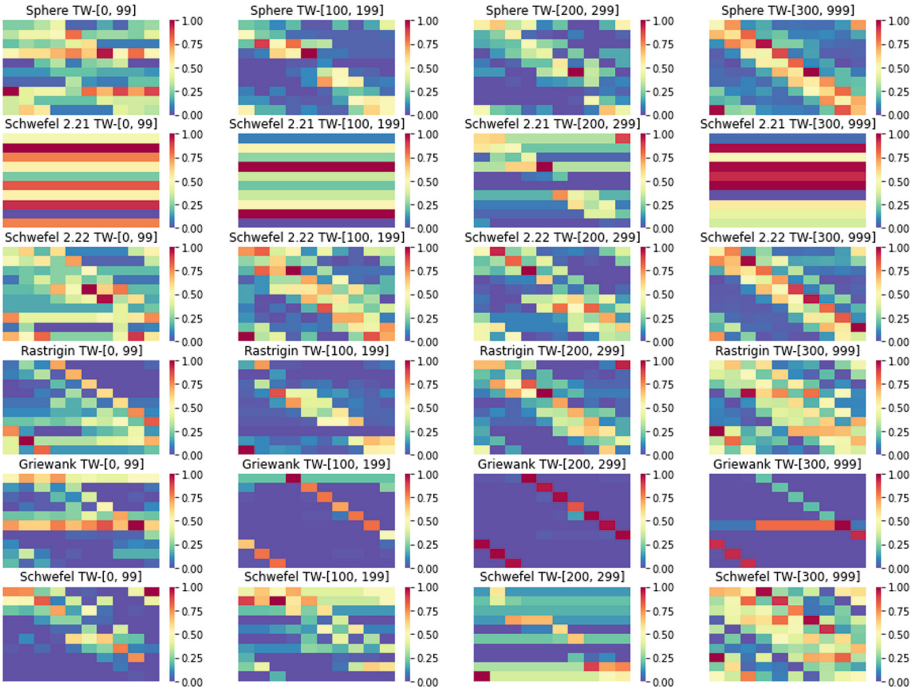

Fig. 6. Interaction Network generated from the simulation with the best fitness for each function of Scenario 2.

random points, the Local, Global, and Dynamic topology substantially affected the displacement, respectively.

We observe that for the best simulations, in Scenario 1 (Figs. 3 and 5), for unimodal functions, the Global topology was more chosen combined with the Dynamic at the end of the simulation which can be observed on the networks. Nevertheless, the Local topology strongly affects the movement from the middle to the end of the simulation (by looking at the diagonals from Sphere and Schwefel 2.22). For the multimodal functions, the Local topology also appears as an essential element for the best simulations, even though it was not the most chosen one for Rastrigin and Griewank functions.

In Scenario 2 (Figs. 4 and 6), we observe some similarities to Scenario 1, but the Dynamic topology is more present on the networks. The importance of the Dynamic topology is in line with its performance, depicted in Table 2. In contrast to the fact that the Global topology was frequently chosen for the multimodal functions, the effect of this topology could have been more substantial than the other topologies.

5 Conclusions

In this paper, we applied RL to the PSO, allowing the swarm to change its communication topology over time. We compared the efficiency of RL when trained on two functions and tested it on two other new functions with similar characteristics. We chose two well-established functions in the literature (Rastrigin and Schwefel 2.22) and tested them on two other multimodal and unimodal functions, respectively.

Using our simulated scenario, we demonstrated that applying Reinforcement Learning in Swarm Intelligence could be efficient across functions. We observed that RL could learn how to adapt to the environment even when not trained in the same function, indicating the capability of transferring learning among functions. Nevertheless, a more comprehensive set of experiments is still essential for drawing stronger conclusions.

Our work was a step further in understanding how to automatize the use of Swarm Intelligence for unknown problems and how to understand the performance and patterns from SI. Swarm Intelligence still requires expertise in the domain, so it is not as straightforward as it can become.

In our future work, we aim to understand more clearly the reason for the topologies selected, seeking to understand why some functions, such as Schwefel, obtained worse results in Reinforcement Learning. It is also necessary to evaluate if the training in a single unimodal or multimodal function is enough for the agent to learn the characteristics presented by the functions. The RL agent may more accurately identify the characteristics of the observed functions using multiple functions with the same characteristics in the training phase.

Acknowledgements. The authors thank the Federal Institute of Pernambuco (IFPE), 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. Mariana Macedo was supported by the Artificial and Natural Intelligence Toulouse Institute (ANITI) - Institut 3iA: ANR-19-PI3A-0004.

References

1. Bansal, J.C., Singh, P.K., Pal, N.R. (eds.): Evolutionary and Swarm Intelligence Algorithms, Studies in Computational Intelligence, vol. 779. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-91341-4
2. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) 2**6**(1), 29–41 (1996). https://doi.org/10.1109/3477.484436
3. Engstrom, L., et al.: Implementation matters in deep policy gradients: a case study on PPO and TRPO (2020)
4. Junior, M.A.C.O., Bastos Filho, C.J.A., Menezes, R.: Using network science to define a dynamic communication topology for particle swarm optimizers. In: Menezes, R., Evsukoff, A., González, M. (eds.) Complex Networks. Studies in Computational Intelligence, vol. 424, pp. 39–47. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-30287-9_5

5. Karaboga, D., et al.: An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes University, Engineering faculty (2005)
6. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN 1995 - International Conference on Neural Network, vol. 4, pp. 1942–1948 (1995). https://doi.org/10.1109/ICNN.1995.488968
7. Kennedy, J.: Swarm Intelligence, pp. 187–219. Springer, US, Boston, MA (2006)
8. Lira, R.C., Macedo, M., Siqueira, H.V., Bastos-Filho, C.: Integrating reinforcement learning and optimization task: Evaluating an agent to dynamically select PSO communication topology. In: Tan, Y., Shi, Y., Luo, W. (eds.) Advances in Swarm Intelligence. ICSI 2023. LNCS, vol. 13969, pp. 38–48. Springer, Cham (2023).https://doi.org/10.1007/978-3-031-36625-3_4
9. 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
10. Oliveira, M., Bastos-Filho, C.J.A., Menezes, R.: Towards a network-based approach to analyze particle swarm optimizers. In: 2014 IEEE Symposium on Swarm Intelligence, pp. 1–8 (2014). https://doi.org/10.1109/SIS.2014.7011791
11. Oliveira, M., Bastos-Filho, C.J.A., Menezes, R.: Using network science to assess particle swarm optimizers. Soc. Netw. Anal. Min. **5**(1), 3 (2015). https://doi.org/10.1007/s13278-015-0245-5
12. Oliveira, M., Pinheiro, D., Andrade, B., Bastos-Filho, C., Menezes, R.: Communication diversity in particle swarm optimizers. In: Dorigo, M., et al. (eds.) ANTS 2016. LNCS, vol. 9882, pp. 77–88. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44427-7_7
13. Parpinelli, R.S., Lopes, H.S.: New inspirations in swarm intelligence: a survey. Int. J. Bio-Inspired Comput. **3(1)**, 1–16 (2011). https://doi.org/10.1504/IJBIC.2011.038700
14. Pervaiz, S., Ul-Qayyum, Z., Bangyal, W.H., Gao, L., Ahmad, J.: A systematic literature review on particle swarm optimization techniques for medical diseases detection. Comput. Math. Methods Med. **2021**, 1–10 (2021). https://doi.org/10.1155/2021/5990999
15. Plevris, V., Solorzano, G.: A collection of 30 multidimensional functions for global optimization benchmarking. Data **7**(4), 46 (2022). https://doi.org/10.3390/data7040046
16. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization: an overview. Swarm Intell. **1**, 33–57 (2007)
17. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017). https://doi.org/10.48550/ARXIV.1707.06347
18. da Silveira Câmara Augusto, J.P., dos Santos Nicolau, A., Schirru, R.: PSO with dynamic topology and random keys method applied to nuclear reactor reload. Progr. Nucl. Energy. **83**, 191–196 (2015). https://doi.org/10.1016/j.pnucene.2015.03.009
19. Wauters, T., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G.: Boosting metaheuristic search using reinforcement learning. In: Talbi, EG. (eds.) Hybrid Metaheuristics. Studies in Computational Intelligence, vol 434, pp. 432–452. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-30671-6_17

20. Wu, D., Wang, G.G.: Employing reinforcement learning to enhance particle swarm optimization methods. Eng. Optim. **54**(2), 329–348 (2022). https://doi.org/10.1080/0305215X.2020.1867120
21. Xu, Y., Pi, D.: A reinforcement learning-based communication topology in particle swarm optimization. Neural Comput. Appl. **32**(14), 10007–10032 (2020). https://doi.org/10.1007/s00521-019-04527-9