Distributed population-based optimization for a fuzzy path tracking controller

Mario García-Valdez$^{1[0000-0002-2593-1114]}$, Alejandra Mancilla$^{1[0000-0003-0430-8152]}$, Oscar Castillo$^{1[0000-0002-7385-5689]}$, and Juan Julián Merelo-Guervós$^{2[0000-0002-1385-9741]}$

^{1} Tijuana Institute of Technology / Tecnólogico Nacional de México, Tijuana, Baja Califorina,Calzada Tecnológico S/N, 22414, Mexico {`mario, alejandra.mancilla`}`@tectijuana.edu.mx,ocastillo@tectijuana.mx`

^{2} Department of Computer Engineering, Robotics and Automation, University of Granada, Spain `jmerelo@ugr.es`

Abstract. We propose a distributed bio-inspired algorithm to optimize the parameters of a fuzzy controller, using a multi-population multi-algorithmic approach. The proposed algorithm can mix different meta-heuristics, one for each population, with the aim of keeping the total population diversity high. To validate the speed-up benefit of our proposal, we optimized the membership functions of a fuzzy controller for the trajectory tracking of a mobile autonomous robot using populations running only Genetic Algorithms and Particle Swarm Optimization, and a mixed configuration where both algorithms run at the same time. Results indicate that even when we use random heterogeneous parameter configuration for the distributed populations, we obtain an error similar to other works while significantly reducing the execution time. *This is a summary of our work "Distributed and asynchronous population-based optimization applied to the optimal design of fuzzy controllers" published in Symmetry in 2023. [2]*

Keywords: Fuzzy Control · Bio-inpired Algorithms · Distribuited Algorithms.

1 Introduction

Fuzzy logic control has been one of the most successful intelligent control techniques [1]. One advantage being the use of a fuzzy inference system (FIS) that models the system using fuzzy rules as the knowladge base. These rules have human interpretation but need to be automatically or manually tuned, to handle the particularities of real-world problems. In the case of FISs, the core components are the fuzzy propositions constructing the fuzzy rules, these propositions are, in turn, constructed using fuzzy variables and fuzzy terms that are defined using MFs. Optimizing the MFs parameters is challenging; as we need to execute one or more simulations to establish the performance of just one configuration.

Our main contribution is to propose a distributed optimization method that speeds up the tuning of fuzzy control optimization problems with respect to sequential versions, without increasing the complexity for the user. This work proposes a multi-population, distributed optimization method considering current practices in constructing highly scalable, resilient, and replicable systems. Moreover, the architecture is capable of executing several bio-inspired algorithms simultaneously. We also address the problem of setting the parameters for each population, this is an important factor in multi-populations algorithms because it has been found in other works to have an impact on the execution time and the optimization results. We compare two strategies: the homogeneous setting using the same parameters in all populations, and the other, a heterogeneous strategy using distinct parameters for all populations.

2 Proposed Method

In this model, we have a multi-population cycle in which a set of populations P, each in a current state j, and a set (W) of w workers, is added to a message queue. Then each population ($population_i$) is pulled set of populations by one of the workers. Then, after executing a metaheuristic, it sends the evolved population (X_{j+1}) to the *mixer*. The the *mixer* swaps some candidate solutions between two populations, changing their state again and sending the resulting populations again to the input message queue to be processed again by some worker, closing the cycle. Each population has several parameters, including the algoritm to be executed on that particular population, and the parameters of that metaheuristic. A single population is executed several times during the lifetime of the algorithm, and this number is limited by the parameter nc (number of cycles). A schematic of an execution cycle is shown in Figure 1. A worker is an agent running in a single process or thread. The worker pulls population messages from a queue and executes the specified metaheuristic.

3 Experimental Results

To validate the algorithm's speed-up and optimization capabilities, we selected the computationally demanding task of tuning a path-tracking fuzzy controller for a bicycle-like mobile robot. The fuzzy controller we optimize in this work has two input variables θ_e and e, indicating the angular error and the distance to the desired path. The controller has a single output ω, used for steering the front wheel. All three variables have the same granularity using five fuzzy terms: *high negative*, *medium negative*, *low*, *medium positive*, and *high positive*. The sign of the error indicates if the angle is to the left or right of the path. We used 15 parameters for the MFs. To test each controller, they had to guide the robot thru three paths. These paths where defined using a cubic splines. The average RMSE obtained from these simulations are the fitness of each controller. We selected the GA and PSO algorithms for the multi-population-based algorithms, we selected the GA as lower-end algorithm specilized in conbinatorial optimization and on the other hand, the PSO algorithm which had the best results in previous experiments. Furthermore, in previous works, but using continuous optimization

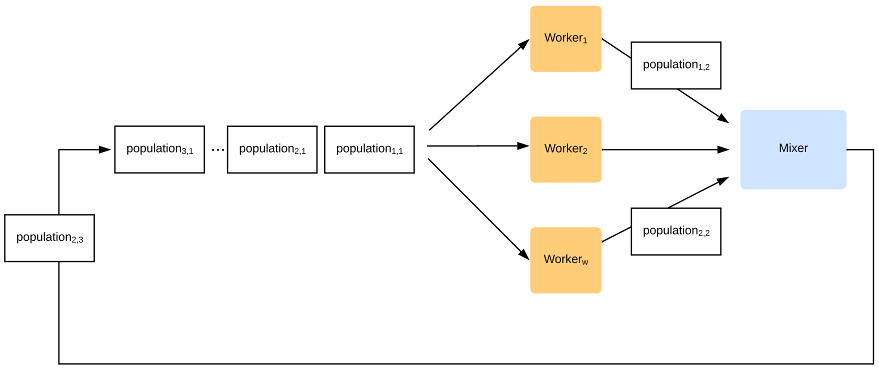

Fig. 1. The multi-population cycle consists of a set of populations P, each in a current state j, and a set (W) of w workers. Each population $(population_i)$ is received by one of the workers. Then, after executing a metaheuristic, it sends the evolved population (X_{j+1}) to the *mixer*. In turn, the *mixer* swaps some candidate solutions between two populations, changing their state again and sending the resulting populations again to be processed by workers, closing the cycle.

benchmarks, the combination of both metaheuristics performed better than any of them in isolation.

When selecting the mataheuristic's parameters, we also experimented with two strategies, one stablishing the same parameters for all populations, these parameters are selected after running more than 30 trial experiments, we call this an homogeneous parametrization. The other option is using a heterogeneous strategy proposed by Gong et al. [3] that randomly sets the parameters of each population. For the GA, we used a tournament selection with $(k = 3)$, a Gaussian mutation with $\mu = 0.0$ and $\sigma = 0.2$, and a probability of 0.3. We use a one-point crossover with a probability of 0.7. For the sequential PSO algorithm, we used a minimum and maximum speeds of -0.25 and 0.25, respectively, and $C_1 = 2$ and $C_2 = 2$. In the sequential case, both algorithms have a single population of 50 candidate solutions and run for 20 iterations. In the sequential version, the number of function evaluations is 1000. This number is obtained by multiplying the population size by the number of iterations. For the multi-population versions of the GA and PSO algorithms. We used the same parameters as the sequential versions for the homogeneous parametrization. We used the same parameters for the heterogeneous versions as before, except for the following parameters affecting the exploration-exploitation balance in the GA and PSO algorithms. The mutation probability is selected for the GA from the $[0.1, 0.5]$ range and the crossover probability from $[0.3, 0.9]$. For the PSO algorithm, we also selected the minimum speed between $[-0.30, -0.20]$ and the maximum from $[0.20, 0.30]$. Both C_1 and C_2 are in the $[1.0, 2.0]$ range. For all the distributed versions, we set the number of populations (islands) to seven, each with a population size of nine. Each worker will execute four iterations (generations) of the algorithm. All populations will complete four cycles; this means they will pass through the mixer module four times. As a result, the total number of function evaluations is 1008. About the same as the sequential, single-population versions. The results are shown in Table 1.

Table 1. Results from 30 observations of the execution time for each of the presented algorithms. The execution time is expressed in seconds. The table shows the results for the sequential (first two columns) and distributed versions of the multi-population-based algorithms: GA, PSO, and the combined version PSO-GA. The homogeneous versions are on the left side and heterogeneous versions on the right.

	Sequential		Distributed					
			Homogeneous			Heterogeneous		
	GA	PSO	GA	PSO	PSO-GA	GA	PSO	PSO-GA
Speedup	-	-	4.74	6.86	6.60	5.05	6.84	6.95
Avg.	1999.34	2851.61	421.67	415.64	431.52	395.75	415.79	409.90
StD.	82.44	278.73	28.54	23.47	22.60	26.54	20.47	24.85
Min	1876.40	2457.98	364.65	365.81	394.61	340.65	374.83	372.20
Max	2253.71	3822.01	526.76	467.47	478.80	461.46	461.96	465.27

4 Concluding Remarks

These results confirm that a multi-population-based strategy offers a convenient speedup while keeping the results very similar to their sequential counterparts. Moreover, the combined algorithm offers better execution times, combining the continuous optimization capabilities of a PSO algorithm with the faster GA metaheuristic. Moreover, the randomized configuration parameters heterogeneous strategy gave better execution times when using the PSO-GA variant.

References

1. Driankov, D., Saffiotti, A.: Fuzzy logic techniques for autonomous vehicle navigation, vol. 61. Physica (2013)
2. García-Valdez, M., Mancilla, A., Castillo, O., Merelo-Guervós, J.J.: Distributed and asynchronous population-based optimization applied to the optimal design of fuzzy controllers. Symmetry **15**(2) (2023). https://doi.org/10.3390/sym15020467, https://www.mdpi.com/2073-8994/15/2/467
3. Gong, Y., Fukunaga, A.: Distributed island-model genetic algorithms using heterogeneous parameter settings. In: 2011 IEEE Congress of Evolutionary Computation (CEC). pp. 820–827. IEEE (2011)