Randomized Parameter Settings for a Pool-based Particle Swarm Optimization Algorithm

A Comparison Between Dynamic Adaptation of Parameters and Randomized Parameterization

Amaury Hernandez-Aguila
Tijuana Institute of Technology
amherag@tectijuana.edu.mx

Mario Garcia-Valdez
Tijuana Institute of Technology
mario@tectijuana.edu.mx

Juan Julian Merelo-Guervos
University of Granada
jmerelo@geneura.ugr.es

Oscar Castillo
Tijuana Institute of Technology
ocastillo@tectijuana.edu.mx

ABSTRACT

This work makes a comparison between different parameter tuning strategies and a strategy based on randomized parameterization in a pool-based model for the particle swarm optimization algorithm. The proposed method is compared against strategies that implement dynamic adaptation of parameters through the use of fuzzy inference systems. The experiments show results that support a hypothesis stating that the use of randomized parameterization can make a pool-based particle swarm optimization algorithm perform as well as its dynamically adapted counterpart.

CCS CONCEPTS

•Computing methodologies → Optimization algorithms; Parallel algorithms;

KEYWORDS

Particle swarm optimization, parallelization, parameter tuning

ACM Reference format:
Amaury Hernandez-Aguila, Mario Garcia-Valdez, Juan Julian Merelo-Guervos, and Oscar Castillo. 2017. Randomized Parameter Settings for a Pool-based Particle Swarm Optimization Algorithm. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 2 pages.
DOI: http://dx.doi.org/10.1145/3067695.3076100

1 INTRODUCTION

This work proposes the use of a randomized parameter setting (RPS) strategy to determine the values of the parameters in particle swarm optimization (PSO) instances in a pool-based model. The benefit of implementing this strategy is that one does not need to perform an optimization of the parameters in a PSO instance using a computationally demanding method, as all the parameters are set to random values. The drawback for this approach could be that setting the parameters in a randomly fashion is not going to be as effective as using a dynamic adapter, another optimization method, or even set the parameters manually. This work presents an experiment that support the claim that the proposed method could be as effective as a dynamic adapter, specifically a fuzzy inference system.

GECCO '17 Companion, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3067695.3076100

2 PROPOSED METHOD

The proposed method is designed to accelerate the convergence of the PSO algorithm. The hypothesis that was taken into consideration for this method is that extending PSO to a parallel architecture should increase the position diversity of the particles, and avoid premature convergence around local optima and should yield better results because of this, as explained by S. Cheng in [1]. Furthermore, each device in the parallel architecture that is searching for a solution is configured with a random set of parameters, as proposed by Y. Gong and A. Fukunaga in [2]. This aproach, as [2] expresses it, exploits the fact that with a sufficient number of instances of an EA, there is a high probability that one of the instances will have a set of randomized parameters that performs well on a given problem.

3 EXPERIMENTS

In order to compare the proposed method presented in Section 2, a set of other parameter tuning strategies were defined. These strategies are based on the ones presented in the work by Melin et al. [4], where a dynamic adaptation of the cognitive and the social factors in the particle swarm optimization algorithm is performed using a fuzzy inference system.

The objective of the work in [4] was to determine what combination of inputs was better to perform a dynamic adaptation of parameters in a PSO. In the case of the present work, the authors propose that a randomized parameterization will achieve better or similar results than any of the proposed strategies in [4], supporting the idea that a good parameter tuning in EAs can be achieved by randomized parameterization.

To give shorter descriptions of the dynamic adaptation strategies, it can be mentioned that all of the membership functions are described by triangular functions. For a triangular membership function, one has to define three points (a, b, and c), where $f(a) = 0$, $f(b) = 1$, $f(c) = 0$, and $a <= b <= c$. The social and the cognitive factors, represented by $c1$ and $c2$, range in domain from 0 to 3, as values in this interval are recommended in the literature [3]. The rule bases for the strategies are obtained from [4].

The first strategy, called Fuzzy PSO 1, is configured as follows. The first antecedent, *iterations*, has a domain from 0 to 1, which

represents the percentage of iterations completed at a certain point, and is described by three adjectives: low, medium, and high. In the case of *low*, $a = 0$, $b = 0$, and $c = 0.5$; for *medium*, $a = 0$, $b = 0.5$, and $c = 1$; and for *high*, $a = 0.5$, $b = 1$, and $c = 1$. The second antecedent, *diversity*, has a domain from 0 to 1, and is described by three adjectives: low, medium, and high. In the case of *low*, $a = 0$, $b = 0$, and $c = 0.5$; for *medium*, $a = 0$, $b = 0.5$, and $c = 1$; and for *high*, $a = 0.5$, $b = 1$, and $c = 1$. For the consequents, *c1* and **c2**, both have a domain from 0 to 3, and are described by five adjectives each: low, medium-low, medium, medium-high and high. In the case of *low*, $a = 0$, $b = 0.5$, and $c = 1$; for *medium-low*, $a = 0.5$, $b = 1$, and $c = 1.5$; for *medium*, $a = 1$, $b = 1.5$, and $c = 2$; for *medium-high*, $a = 1.5$, $b = 2$, and $c = 2.5$; lastly, for *high*, $a = 2$, $b = 2.5$, and $c = 3$.

The second strategy, called Fuzzy PSO 2, is similar to Fuzzy PSO 1, with a slight difference in the antecedents. Instead of *iterations* and *diversity*, it uses *iterations* and *error*. This antecedent, *error*, has a domain from 0 to 1, and is described by three adjectives: low, medium, and high. In the case of *low*, $a = 0$, $b = 0$, and $c = 0.5$; for *medium*, $a = 0$, $b = 0.5$, and $c = 1$; and for *high*, $a = 0.5$, $b = 1$, and $c = 1$. The remaining antecedent and consequents are described the same way as in Fuzzy PSO 1.

For the third strategy, called Fuzzy PSO 3, *iterations*, *diversity* and *error* are considered as antecedents, and *c1* and *c2* as the consequents. All of them are defined the same as in Fuzzy PSO 1 and Fuzzy PSO 2.

In the case of the proposed method, five instances of PSO are created, which are going to be drawing individuals from the population in the server pool. Every time an instance obtains a new sample of individuals, it performs an evolutionary process of only one generation, using randomized values for *c1* and *c2*. The sample size will be $N/5$, meaning that each device will have an equally sized sample. The samples always gather the individuals by random choice.

For each of the strategies mentioned in this Section, an initial population of 200 individuals is randomly generated, and they will be evolved for 100 iterations.

The number of evaluations required for each method to reach certain threshold of error is recorded. In order to avoid converging to said threshold too soon or too late, a simple PSO with $c1 = 1$ and $c2 = 3$ is run 100 times, and the error obtained by the 90th percentile is used as the threshold. The reason behind this is that if a very low threshold is used, the strategies could never converge to it, and every strategy would report similar number of evaluations. Likewise, if a very high threshold is used, the strategies could converge to it too early, and every strategy would also report similar number of evaluations.

4 RESULTS

The basic procedure is to perform comparisons between each strategy against the proposed method, called PB-RPS PSO (pool-based randomized parameter settings PSO) in the results table. These comparisons take into consideration the number of evaluations required to achieve certain threshold, as is explained in Section 3.

For the comparisons, 100 runs of each strategy are performed, and the number of evaluations is recorded after each run. The average and standard deviations are calculated and used to perform hypothesis tests. In the case of the comparisons in terms of diversity, 30 runs of each strategy are performed, for 100 generations. After each generation finishes its evolutionary process, the diversity in the particles is recorded, giving as result a record of 100 diversities for each of the 30 runs. In order to obtain the final averages and standard deviations for each strategy, the average and standard deviation of the 100 generations is calculated for each of the 30 runs, and the average is calculated for these 30 averages and 30 standard deviations, and they are used to perform hypothesis tests.

For the diversity comparisons, the hypothesis tests are considering that the proposed method obtains a higher diversity in its population. In the case of the number of evaluations, what is being considered is that the proposed method obtains a lower number than the other strategies. The results tables

Table 1: Comparison of Number of Evaluations for Each Method Using the Rastrigin Function Benchmark

Method	μ	SD	n	t-Value	CI
PSO	4585.93	6756.02	100	3.3529	> 99.8%
Fuzzy PSO 1	3148.97	5625.51	100	1.6683	> 90%
Fuzzy PSO 2	2332.24	4056.50	100	0.5252	-
Fuzzy PSO 3	2648.31	4808.98	100	1.0102	-
PB-RPS PSO	2055.44	3363.98	100		

Table 1 shows the results for the Rastrigin benchmark function. This Table shows t-Values and confidence intervals to illustrate how well the proposed method performed against the corresponding strategy. If a hyphen is printed instead of a confidence interval, this means that the proposed method performed similarly than the corresponding strategy (a t-Value corresponding to a confidence interval of less than 80% was calculated).

5 CONCLUSIONS

The results obtained in this work support the use of randomized parameterization in the particle swarm optimization algorithm when used in a pool-based model. A high diversity in the particles should help the algorithm to not fall into local minima prematurely, and a low number of evaluations means less computational power is needed in order to obtain satisfactory results.

REFERENCES

[1] Shi Cheng. 2013. *Population diversity in particle swarm optimization: Definition, observation, control, and application.* Ph.D. Dissertation. University of Liverpool.
[2] Yiyuan Gong and Alex Fukunaga. 2011. Distributed island-model genetic algorithms using heterogeneous parameter settings. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 820–827.
[3] James Kenndy and RC Eberhart. 1995. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, Vol. 4. 1942–1948.
[4] Patricia Melin, Frumen Olivas, Oscar Castillo, Fevrier Valdez, Jose Soria, and Mario Valdez. 2013. Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic. *Expert Systems with Applications* 40, 8 (2013), 3196–3206.