Fuzzy Self-Tuning PSO: Single-objective global optimization without moving a finger

Marco S. Nobile[1] [0000−0002−7692−7203]

Eindhoven University of Technology, Eindhoven, The Netherlands
m.s.nobile@tue.nl

Keywords: Fuzzy Self-Tuning PSO · Particle Swarm Optimization · global optimization · swarm intelligence · python.

1 Introduction

Particle Swarm Optimization (PSO) is an effective algorithm for non-linear and complex high-dimensional problems. One major drawback of PSO is that its performance strongly depends on the choice of its hyper-parameters (i.e., inertia, cognitive and social factors, minimum and maximum velocity), Fuzzy Self-Tuning PSO (FST-PSO) is a novel swarm intelligence population-based meta-heuristic able to tune all settings at run-time, yielding a completely settings-free version of PSO [7]. The main innovation of FST-PSO is that each particle adjusts its own hyper-settings, using a fuzzy reasoner that uses as input variables to the proximity to the (current) best particle in the swarm and the improvement with respect to the previous iteration. In this paper, I provide a brief explanation about how to install and exploit the python implementation of FST-PSO, highlighting the simplicity of this library.

Competitive with state of the art methods [7], FST-PSO have been successfully applied to several real-world problems and disciplines, e.g., systems biology [8, 13], cancer research [9, 6, 10], molecular dynamics [2], geology [12], computational neurosciences [11], epidemiology [5], fuzzy clustering [3], and fuzzy modeling [4]. In general, FST-PSO can be applied in all situations in which some real-valued parameters of a model must be optimized with respect of some objective function (e.g., parameter estimation, calibration, training of neural networks, and regression problems).

2 Installation and usage

Although the source code is available for download on GITHUB at the address `https://github.com/aresio/fst-pso`, the easiest way to install FST-PSO is by using pip, by using the command `pip install fst-pso`. Once FST-PSO is installed in the system, it can be used in a python script to minimize a single objective function inside a bounded search-space. Both information must be provided by the user. For example, let us assume to be interested in optimizing the Ackley function in the search space $[-5, 5]^D$. Listing 1.1 shows how to perform this optimization using FST-PSO.

Listing 1.1. 'Example of FST-PSO usage.'

```
1 from fstpso import FuzzyPSO
2 FP = FuzzyPSO()
3 D = 10 # dimensions of the problem
4 FP.set_fitness(Ackley)
5 FP.set_search_space([[-5, 5]]*D)
6 best_sol, best_fit = FP.solve_with_fstpso()
```

The first step is to import and create a `FuzzyPSO` object (lines 1–2). Then, the user must specify the fitness function to be minimized[1] (line 4) and the search space in which particles will move (line 5) using the `set_fitness` and `set_search_space` methods, respectively. Finally, the optimization can be launched by using the `solve_with_fstpso` method (line 6) which returns the best solution found and its fitness. In order to work with FST-PSO, the fitness function must have two characteristics: it receives as argument a particle and returns as output the fitness value of that particle.

Please note that this is all the information that is needed to optimize any function with FST-PSO: as a matter of fact, the algorithm automatically determines the swarm size according to an internal heuristics based on the number of dimensions. However, since the optimal choice for the number of particle is strongly problem-dependent, if can be forced by the user by using the `set_swarm_size` method. Similarly, as a default FST-PSO executes a maximum of 100 iterations. This settings can be override by specifying the optional keyword argument `max_iter` to the `solve_with_fstpso` method. Finally, FST-PSO assume a minimization problem; however, a maximization problem can be easily turned into a minimization problem by changing the sign of the fitness value (e.g., using a decorator).

FST-PSO accepts several optional parameters and arguments (e.g., non-uniform initialization of particles, the possibility of disengaging some fuzzy rules, distributing the fitness evaluations over some high-performance computing facility). A summary of currently supported options can be found at the following address: `https://github.com/aresio/fst-pso/wiki`. Notably, FST-PSO offers the possibility of providing some initial "educated guesses" for the particles, a functionality that was exploited by Swarm-CG [2] developers to accelerate the convergence to optimal molecular structures in coarse-grained simulations of nano-materials.

3 Future developments

FST-PSO is a settings-free meta-heuristics that was designed to perform single-objective optimization (in particular, minimization). However, many complex

[1] Of course, the `Ackley` function, used in this example, must have been defined before.

real-world problems require the simultaneous optimization of multiple conflicting objective functions. Under these circumstances, multi-objective optimization is more suitable to obtain a good approximation of the Pareto front of optimal dominating candidate solutions. The main future development of FST-PSO will be the integration of the velocity update formulas of multi-objective variants of PSO (notably, MOPSO [1]), in order to make FST-PSO able to tackle this class of problems.

References

1. Coello, C.C., Lechuga, M.S.: MOPSO: A proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC). vol. 2, pp. 1051–1056. IEEE (2002)
2. Empereur-Mot, C., Pesce, L., Bochicchio, D., Perego, C., Pavan, G.M.: Swarm-CG: Automatic Parametrization of Bonded Terms in Coarse-Grained Models of Simple to Complex Molecules via Fuzzy Self-Tuning Particle Swarm Optimization (2020)
3. Fuchs, C., Spolaor, S., Nobile, M.S., Kaymak, U.: A swarm intelligence approach to avoid local optima in fuzzy c-means clustering. In: 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). pp. 1–6. IEEE (2019)
4. Fuchs, C., Spolaor, S., Nobile, M.S., Kaymak, U.: pyFUME: a Python package for fuzzy model estimation. In: 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). pp. 1–8. IEEE (2020)
5. Gallese, C., Falletti, E., Nobile, M.S., Ferrario, L., Schettini, F., Foglia, E.: Preventing litigation with a predictive model ofcovid-19 icus occupancy. In: Fourth Annual Workshop on Applications of Artificial Intelligence in the Legal Industry, IEEE BigData 2020. IEEE (2020)
6. Nobile, M., Nisoli, E., Vlachou, T., Spolaor, S., Cazzaniga, P., Mauri, G., Pelicci, P.G., Besozzi, D.: cuProCell: GPU-accelerated analysis of cell proliferation with flow cytometry data. IEEE Journal of Biomedical and Health Informatics (2020)
7. Nobile, M.S., Cazzaniga, P., Besozzi, D., Colombo, R., Mauri, G., Pasi, G.: Fuzzy Self-Tuning PSO: A settings-free algorithm for global optimization. Swarm and evolutionary computation **39**, 70–85 (2018)
8. Nobile, M.S., Tangherloni, A., Rundo, L., Spolaor, S., Besozzi, D., Mauri, G., Cazzaniga, P.: Computational intelligence for parameter estimation of biochemical systems. In: Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8. IEEE (2018)
9. Nobile, M.S., Vlachou, T., Spolaor, S., Bossi, D., Cazzaniga, P., Lanfrancone, L., Mauri, G., Pelicci, P.G., Besozzi, D.: Modeling cell proliferation in human acute myeloid leukemia xenografts. Bioinformatics **35**(18), 3378–3386 (2019)
10. Nobile, M.S., Vlachou, T., Spolaor, S., Cazzaniga, P., Mauri, G., Pelicci, P.G., Besozzi, D.: ProCell: Investigating cell proliferation with Swarm Intelligence. In: 2019 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB). pp. 1–8. IEEE (2019)
11. Papetti, D.M., Spolaor, S., Besozzi, D., Cazzaniga, P., Antoniotti, M., Nobile, M.S.: On the automatic calibration of fully analogical spiking neuromorphic chips. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2020)

12. Soltani-Moghadam, S., Tatar, M., Komeazi, A.: An improved 1-D crustal velocity model for the Central Alborz (Iran) using Particle Swarm Optimization algorithm. Physics of the Earth and Planetary Interiors **292**, 87–99 (2019)
13. Totis, N., Tagherloni, A., Beccuti, M., Cazzaniga, P., Nobile, M.S., Besozzi, D., Pennisi, M., Pappalardo, F.: Efficient and settings-free calibration of detailed kinetic metabolic models with enzyme isoforms characterization. In: International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics. pp. 187–202. Springer (2018)