EvoloPy: An open source Nature-Inspired Optimization Toolbox in Python

First Author Name[1], Second Author Name[1] and Third Author Name[2]
[1]*Institute of Problem Solving, XYZ University, My Street, MyTown, MyCountry*
[2]*Department of Computing, Main University, MySecondTown, MyCountry*
{*f_author, s_author*}*@ips.xyz.edu, t_author@dc.mu.edu*

Keywords: The paper must have at least one keyword. The text must be set to 9-point font size and without the use of bold or italic font style. For more than one keyword, please use a comma as a separator. Keywords must be titlecased.

Abstract: EvoloPy is an open source and cross-platform Python toolbox that implements a wide range of classical and recent nature-inspired metaheuristic algorithms. The goal of this toolbox is to facilitate the use of metaheuristic algorithms by non-specialists coming from different domains. With a simple interface and minimal dependencies, it is easier for researchers and practitioners to utilize EvoloPy for optimizing and benchmarking their own defined problems using the most powerful metaheuristic optimizers in the literature. On the other hand, the design of this toolbox makes it very easy for the researchers in the domain to integrate their own optimizers and compare their performance to the state of art algorithms.

1 INTRODUCTION

Nature-inspired metaheuristic algorithms are considered to be the one of the most efficient approaches to solve optimization problems (Yang, 2013). Nature-inspired metaheuristics imitate natural phenomena such as the the behavior of the nature systems like swarm based algorithms, and evolution based algorithms. Over last three decades, many nature-inspired metaheuristics algorithms have been developed. Swarm intelligence algorithms simulate the natural swarms such as flocks of birds, ant colonies, and schools of fishes such as Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995), Cuckoo Search (CS) (Yang and Deb, 2009), Ant Colony Optimization (ACO) (Korošec and Šilc, 2009), Grey Wolf Optimizer (GWO) (Mirjalili et al., 2014), Multi-Verse Optimizer (MVO) (Mirjalili et al., 2016), Moth-flame optimization (MFO) (Mirjalili, 2015), Whale Optimization Algorithm (WOA) (Mirjalili and Lewis, 2016), Bat Algorithm (BA) (Yang, 2010), and many others. Most of the previous algorithms reaches the best solutions by exchanging the information between the swarm's individuals.

On the other hand, evolutionary based algorithms are inspired by some concepts from the Darwinian theories about evolution and natural selection. Such algorithms include Genetic algorithm (GA) (Sivanandam and Deepa,), Genetic Programming (GP)(Koza, 1992), and Evolution Strategy (ES)(Beyer and Schwefel, 2002). These algorithms use different strategies to evolve and find good solutions for difficult problems.

In the last two decades there have been a growing interest in the applications of these algorithms for solving and optimizing complex problems in different domains.

2 RELATED WORK

Today, there exist many different nature-inspired optimization libraries and toolboxes. Some of them are developed for special type of nature-inspired optimization like GEATbx (Hartmut Pohlheim,), which is a toolbox contains many variants of the genetic algorithms, and genetic Programming. GEATbx is implemented in MATLAB environment and provides many global optimization capabilities. The authors in (Matthew Wall,) developed GAlib library, which contains a set of C++ genetic algorithm tools and operators for different optimization problems. GAlib is built on UNIX platforms and supported parallel experiments. In (Fortin et al., 2012), the authors developed a novel evolutionary computation python

framework called DEAP to simplify the execution of many optimization ideas with parallelisation features. The DEAP Framework includes many nature-inspired algorithms with different variations such as: Genetic Algorithm, Genetic Programing (GP), Evolution Strategies (ES), Particle Swarm Optimization (PSO), Differential Evolution (DE), and multi-objective optimization. In addition, Deap includes Benchmarks modules containing many test functions for evaluations. In (Wagner and Affenzeller, 2004), the author presents a generic optimization environment named HeuristicLab. HeuristicLab is a framework for heuristic, evolutionary algorithms, and machine learning. HeuristicLab includes many algorithms such as: Genetic Programming, Age-layered Population, Structure (ALPS), Evolution Strategy, Genetic Algorithm, Island Genetic Algorithm, Particle Swarm Optimization, Relevant Alleles Preserving GA, and many others.

3 WHY PYTHON?

Python is a general purpose scripting language which has a clear and simple syntax. With the rapid development of mature, advanced and open source scientific computing libraries and packages, Python became as one of the most popular and powerful languages for scientific computing. Moreover, Python has cross-platform runability, which works with different operating systems, and has ability to access libraries written in different programming languages and computing environments. Python also can support small-form devices, embedded systems, and microcontrollers. In addition, Python needs very minimal setup procedure to start with. Python uses modular and object based programming, which is a popular methodology to organize classes, functions, and procedures into hierarchical namespaces. All these reason has turned Python to be a popular language in a huge community of scientists.

4 Design issues

Most of nature inspired metaheuristics are population-based algorithms. These algorithms start by randomly initializing a set of individuals each of which represents a candidate solution. Conventionally, in most of state-of-art evolutionary algorithms frameworks, populations are implemented as 2-dimensional arrays while individuals are implemented as 1-dimensional array. For example an individual I and a population P can be represented as given in equations 1 and 2.

$$I_i = \begin{bmatrix} x_i^1 & x_i^2 & \dots & x_i^d \end{bmatrix} \tag{1}$$

$$P = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^d \\ x_2^1 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \dots & x_n^d \end{bmatrix} \tag{2}$$

In EvoloPy, `Numpy` is chosen to define and represent the populations and individuals in the optimizers. Numpy library is one of the main packages for scientific computing in Python. `Numpy` has many things in common with Matlab. This makes it easier for researchers who are familiar with Matlab use EvoloPy.

The `Numpy`s array class `ndarray` is used to define 1-dimensional and 2-dimensional arrays that represent individuals and populations, respectively. For example, to define a randomly generated initial population with 50 individuals and 100 dimensions the following python line of code can be used

```
initialPop = np.random.rand(500,100)
```

Moreover, with `Numpy` it is possible to perform vector operations in simple and compact syntax. For example, a common operation is used in many metaheuristic algorithms is to check the boundaries of the elements of the individuals after updating them. This can be performed all at once with `Numpy` as follows:

```
newPop=numpy.clip(oldPop, lb, ub)
```

Where `lb` and `ub` are the upper and lower bounds respectively.

5 Comparison with Matlab

In this section we compare the metaheuristic algorithms implemented so far in EvoloPy with their peers in Matlab based on their running time. The idea here is to measure the running time of the main loop of each algorithm in EvoloPy and Matlab with an equal number of function evaluations. The experiments were performed on on a personal machine with an Intel(R) Core(TM) i5-2400 CPU at 3.10Ghz and a memory of 4 GB running Windows 7 professional 32bit operating system. In all experiments, the population size and the number of iterations was set for all algorithms to 50, and 100, respectively. In order to study the effect of the size of the problem to be solved on the running time of the algorithms, all optimizer are executed using different dimension sizes ranging from a small size of 50 elements reaching up to 20,000 in both implementations, Matlab and EvoloPy.

ACKNOWLEDGEMENTS

REFERENCES

Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies – a comprehensive introduction. *Natural Computing*, 1(1):3–52.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175.

Hartmut Pohlheim. Geatbx - the genetic and evolutionary algorithm toolbox for matlab.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4.

Korošec, P. and Šilc, J. (2009). A distributed ant-based algorithm for numerical optimization. In *Proceedings of the 2009 workshop on Bio-inspired algorithms for distributed systems - BADS 09*. Association for Computing Machinery (ACM).

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

Matthew Wall. Galib: A c++ library of genetic algorithm components.

Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems*, 89:228 – 249.

Mirjalili, S. and Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, 95:51 – 67.

Mirjalili, S., Mirjalili, S. M., and Hatamlou, A. (2016). Multi-verse optimizer: a nature-inspired algorithm for global optimization. *Neural Computing and Applications*, 27(2):495–513.

Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69:46 – 61.

Sivanandam, S. and Deepa, S. Genetic algorithms. In *Introduction to Genetic Algorithms*, pages 15–37. Springer Science Business Media.

Wagner, S. and Affenzeller, M. (2004). The heuristiclab optimization environment. Technical report, University of Applied Sciences Upper Austria.

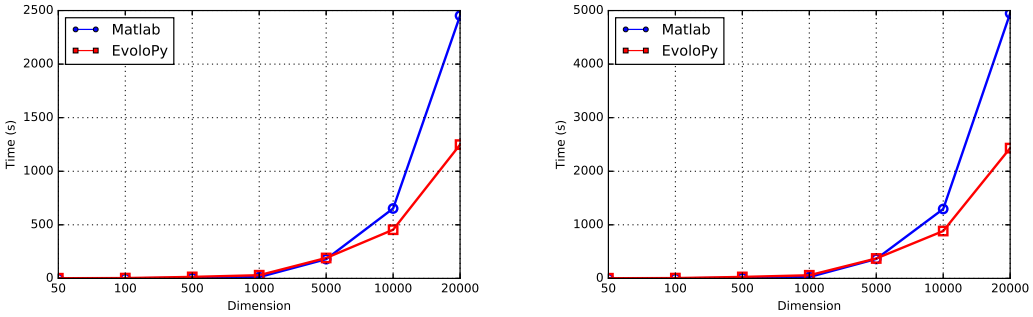
Yang, X.-S. (2010). *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, chapter A New Metaheuristic Bat-Inspired Algorithm, pages 65–74. Springer Berlin Heidelberg, Berlin, Heidelberg.

Yang, X.-S. (2013). Metaheuristic optimization: Nature-inspired algorithms and applications. In *Studies in Computational Intelligence*, pages 405–420. Springer Science Business Media.

Yang, X. S. and Deb, S. (2009). Cuckoo search via levy flights. In *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 210–214.

APPENDIX

If any, *section*{APPENDIX*}*



(a) BAT (b) CS

(c) FPA (d) GWO

(e) MFO (f) MVO

(g) PSO (h) WOA

Figure 1: Comparison between between the metaheuristic optimizers in EvoloPy and Matlab based on execution time of the main iteration.