The Journal of Open Source Software

Metaheuristics: A Julia Package for Single- and Multi-Objective Optimization

Jesús-Adolfo Mejía-de-Dios [1¶] **and Efrén Mezura-Montes** [1]

1 Artificial Intelligence Research Institute, University of Veracruz, MEXICO ¶ Corresponding author

DOI: 10.21105/joss.04723

Software

- Review
- Repository
- Archive

Editor: Vissarion Fisikopoulos

Reviewers:

- @idoby
- @PaulDebus

Submitted: 14 July 2022
Published: 26 October 2022

Summary

Metaheuristics is a Julia package that implements metaheuristic algorithms for solving global optimization problems that can contain constraints and either single or multiple objectives. The package exposes an easy-to-use API to enable testing without requiring lengthy configuration for choosing among the implemented optimizers. For example, to optimize an objective function $f(x)$ where solution x is within the corresponding bounds using the Evolutionary Centers Algorithm optimizer, the user can call `optimize(f, bounds, ECA())`. Moreover, Metaheuristics provides common features required by the evolutionary computing community such as challenging test problems, performance indicators, and other notable utility functions. This paper presents the main features followed by some examples to illustrate the usage of this package for optimization.

Statement of need

Real-world problems often require sophisticated methods to solve. Metaheuristics are stochastic algorithms that approximate optimal solutions quickly where not all of the mathematical properties of the problem are known. Our package implements state-of-the-art algorithms for constrained, multi-, and many-objective optimization. It also includes many other utility functions such as performance indicators, scalable benchmark problems, constraint handling techniques, and multi-criteria decision-making methodologies. Although similar software has been proposed in different programming languages such as Python (Blank & Deb, 2020), MATLAB (Tian et al., 2017), C/C++ (Biscani & Izzo, 2020), and Java (Nebro et al., 2015), among others (Johnson, 2022), Metaheuristics is the first package in Julia containing ready-to-use metaheuristic algorithms and utility functions with a uniform API.

There are also some packages implemented in Julia for global optimization. For example, `Optim.jl` (Mogensen & Riseth, 2018) implements global optimizers such as Particle Swarm Optimization, `BlackBoxOptim.jl` (Feldt, 2022) implements a couple of stochastic heuristics for black-box optimization, `Evolutionary.jl` (Wilde et al., 2021) is a framework for evolutionary computing, and `CMAEvolutionStrategy.jl` (Brea, 2022) implements a CMA Evolution Strategy. Unlike these packages, Metaheuristics can handle equality and inequality constraints and supports multi-objective problems.

Main Features

This section describes the primary features of Metaheuristics. First, Metaheuristics implements a consistent and intuitive API to approximate the optimal solutions of the following minimization problem:

$$\min_{x \in X} f(x)$$

subject to

$$g_j(x) \leq 0, \ j = 1, 2, \ldots, J;$$
$$h_l(x) = 0, \ l = 1, 2, \ldots, L;$$

where $x_{i,\min} \leq x_i \leq x_{i,\max}$, i.e., $x \in X = \prod_{i=1}^{D} [x_{i,\min}, x_{i,\max}]$. f can be either single- or multi-objective.

Implemented Metaheuristics

Implemented metaheuristic algorithms are detailed in **Table 1**. Algorithms for single-objective optimization include Evolutionary Centers Algorithm (ECA) (Mejía-de-Dios & Mezura-Montes, 2019), Differential Evolution (DE) (Price, 2013), Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995), Artificial Bee Colony (ABC) (Karaboga & Basturk, 2007), Gravitational Search Algorithm (GSA) (Mirjalili & Gandomi, 2017), Simulated Annealing (SA) (Van L. & Aarts, 1987), Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016), and Machine Coded Compact Genetic Algorithms (MCCGA) (Satman & Akadal, 2020). Metaheuristics also includes multi-objective optimization algorithms such as a Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D-DE) (Li & Zhang, 2008), Non-dominated Sorting Genetic Algorithms (NSGA-II,-III)(Deb et al., 2002; Deb & Jain, 2014), S-Metric Selection Evolutionary Multi-objective Algorithm (SMS-EMOA) (Emmerich et al., 2005), Improved Strength Pareto Evolutionary Algorithm (SPEA2) (Eckart Zitzler et al., 2001)m and Coevolutionary Framework for Constrained Multiobjective Optimization (CCMO) (Tian et al., 2021).

Table 1: Implemented optimizers so far. Here, "✓", "×" and "—", respectively mean that the feature in the corresponding column is available, unavailable, or can be enabled by changing default parameters.

Algorithm	Objective	Constraint Handling	Batch Evaluation	Authors
ECA	Single	✓	✓	Mejía-de-Dios & Mezura-Montes (2019)
DE	Single	✓	✓	Price (2013)
PSO	Single	✓	✓	Kennedy & Eberhart (1995)
ABC	Single	×	×	Karaboga & Basturk (2007)
GSA	Single	×	✓	Mirjalili & Gandomi (2017)
SA	Single	✓	×	Van L. & Aarts (1987)
WOA	Single	✓	✓	Mirjalili & Gandomi (2017)
MCCGA	Single	×	×	Satman & Akadal (2020)
GA	Single	✓	✓	Goldberg (2002)
MOEA/D-DE	Multi	—	×	Li & Zhang (2008)
NSGA-II	Multi	✓	✓	Deb et al. (2002)
SMS-EMOA	Multi	✓	✓	Emmerich et al. (2005)
SPEA2	Multi	✓	✓	Eckart Zitzler et al. (2001)
CCMO	Multi	✓	✓	Tian et al. (2021)
NSGA-III	Many	✓	✓	Deb & Jain (2014)

Performance Indicators

Performance indicators are used to assess algorithms with respect to the quality of the corresponding outcomes (E. Zitzler et al., 2003). Metaheuristics implements Generational Distance (GD), Inverted Generational Distance (IGD), IGD+, Covering Indicator (C-metric),

Hypervolume (HV), Averaged Hausdorff distance (also known as Δ_p), Spacing Indicator, and the ε-indicator.

Multi-Criteria Decision-Making

Metaheuristics also provides algorithms for when Multi-Criteria Decision-Making (MCDM) has to be performed after an optimizer has reported a set of non-dominated solutions. The implemented MCDM techniques include Compromise Programming (Ringuest, 1992) and Region of Interest Archiving (de-la-Cruz-Martínez et al., 2023). Metaheuristics also includes an interface to the JMcDM package (Satman et al., 2021), which implements many further MCDM techniques.

Installation and Usage

Metaheuristics can be installed through the Julia package manager by using the add command in the Pkg prompt:

```
pkg> add Metaheuristics
```

Or, equivalently, via the `Pkg` API:

```
julia> import Pkg
julia> Pkg.add("Metaheuristics")
```

Now, let us consider the following minimization problem:

Minimize:

$$f(x) = 10D + \sum_{i=1}^{D} x_i^2 - 10\cos(2\pi x_i)$$

with box constraints (`bounds`) $x \in [-10, 10]^5$.

We can optimize the objective function f with solution x within the desired bounds as follows:

```
julia> f(x) = 10length(x) + sum( x.^2 - 10cos.(2π*x) )

julia> bounds = [-10ones(5) 10ones(5)]

julia> result = optimize(f, bounds, DE(CR=0.5))
+=========== RESULT ==========+
  iteration: 550
    minimum: 0
  minimizer: [-1.5469066028117595e-9,..., 3.797322900567224e-9]
    f calls: 27500
 total time: 0.0388 s
stop reason: Small difference of objective function values.
+============================+
```

`optimize(f, bounds, OPTIMIZER)` is used to approximate an optimal solution, where `OPTIMIZER` can be selected from the implemented metaheuristics (see **Table 1**). Note that `OPTIMIZER` is a stochastic procedure and therefore each run may show different outputs.

Finally, potential users are encouraged to read the documentation for more details, options, and examples.

Acknowledgements

The first author acknowledges support from the Mexican Council for Science and Technology (CONACyT) through a scholarship to pursue graduate studies at the University of Veracruz,

MEXICO.

References

Biscani, F., & Izzo, D. (2020). A parallel global multiobjective framework for optimization: pagmo. *Journal of Open Source Software*, *5*(53), 2338. https://doi.org/10.21105/joss.02338

Blank, J., & Deb, K. (2020). Pymoo: Multi-objective optimization in python. *IEEE Access*, *8*, 89497–89509. https://doi.org/10.1109/ACCESS.2020.2990567

Brea, J. (2022). CMAEvolutionStrategy.jl: CMA Evolution Strategy in Julia. In *GitHub repository*. GitHub. https://github.com/jbrea/CMAEvolutionStrategy.jl

Deb, K., & Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, *18*(4), 577–601. https://doi.org/10.1109/TEVC.2013.2281535

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017

de-la-Cruz-Martínez, S. J., Mejía-de-Dios, J. A., & Mezura-Montes, E. (2023). Efficient archiving method for handling preferences in constrained multi-objective evolutionary optimization. In J. A. Zapata-Cortes, C. Sánchez-Ramírez, G. Alor-Hernández, & J. L. García-Alcaraz (Eds.), *Handbook on Decision Making: Volume 3: Trends and Challenges in Intelligent Decision Support Systems* (pp. 93–119). Springer International Publishing. https://doi.org/10.1007/978-3-031-08246-7_5

Emmerich, M., Beume, N., & Naujoks, B. (2005). An EMO algorithm using the hypervolume measure as selection criterion. In *Lecture Notes in Computer Science* (pp. 62–76). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-31880-4_5

Feldt, R. (2022). BlackBoxOptim.jl: Black-box optimization for Julia. In *GitHub repository*. GitHub. https://github.com/robertfeldt/BlackBoxOptim.jl

Goldberg, D. E. (2002). *The design of innovation: Lessons from and for competent genetic algorithms* (Vol. 1). Springer.

Johnson, S. G. (2022). The NLopt nonlinear-optimization package. In *GitHub repository*. GitHub. http://github.com/stevengj/nlopt

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459–471. https://doi.org/10.1007/s10898-007-9149-x

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95-International Conference on Neural Networks*, *4*, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

Li, H., & Zhang, Q. (2008). Multiobjective optimization problems with complicated Pareto sets, MOEA/d and NSGA-II. *IEEE Transactions on Evolutionary Computation*, *13*(2), 284–302. https://doi.org/10.1109/tevc.2008.925798

Mejía-de-Dios, J.-A., & Mezura-Montes, E. (2019). A new evolutionary optimization method based on center of mass. In K. Deep, M. Jain, & S. Salhi (Eds.), *Decision Science in Action: Theory and Applications of Modern Decision Analytic Optimisation* (pp. 65–74). Springer Singapore. https://doi.org/10.1007/978-981-13-0860-4_6

Mirjalili, S., & Gandomi, A. H. (2017). Chaotic gravitational constants for the gravitational search algorithm. *Applied Soft Computing*, *53*, 407–419. https://doi.org/10.1016/j.asoc.2017.01.008

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008

Mogensen, P. K., & Riseth, A. N. (2018). Optim: A mathematical optimization package for Julia. *Journal of Open Source Software*, *3*(24), 615. https://doi.org/10.21105/joss.00615

Nebro, A. J., Durillo, J. J., & Vergne, M. (2015). Redesigning the JMetal multi-objective optimization framework. *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 1093–1100. https://doi.org/10.1145/2739482.2768462

Price, K. V. (2013). Differential evolution. In I. Zelinka, V. Snás Sel, & A. Abraham (Eds.), *Handbook of Optimization: From Classical to Modern Approach* (pp. 187–214). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-30504-7_8

Ringuest, J. L. (1992). Compromise programming. In *Multiobjective Optimization: Behavioral and Computational Considerations* (pp. 51–59). Springer US. https://doi.org/10.1007/978-1-4615-3612-3_4

Satman, M. H., & Akadal, E. (2020). Machine coded compact genetic algorithms for real parameter optimization problems. *Alphanumeric Journal*, *8*(1), 43–58. https://doi.org/10.17093/alphanumeric.576919

Satman, M. H., Yıldırım, B. F., & Kuruca, E. (2021). JMcDM: A Julia package for multiple-criteria decision-making tools. *Journal of Open Source Software*, *6*(65), 3430. https://doi.org/10.21105/joss.03430

Tian, Y., Cheng, R., Zhang, X., & Jin, Y. (2017). PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. *IEEE Computational Intelligence Magazine*, *12*(4), 73–87.

Tian, Y., Zhang, T., Xiao, J., Zhang, X., & Jin, Y. (2021). A coevolutionary framework for constrained multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, *25*(1), 102–116. https://doi.org/10.1109/TEVC.2020.3004012

Van L., P. J. M., & Aarts, E. H. L. (1987). Simulated annealing. In *Simulated Annealing: Theory and Aminor cpplications* (pp. 7–15). Springer.

Wilde, A., Thiem, D., Leo, Gupta, A., Haselgrove, A., Molina, D., Honeypot95, matago, Rogerluo, & Churavy, V. (2021). *Evolutionary.jl* (Version v0.10.0) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.5110647

Zitzler, Eckart, Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-Report*, *103*.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & Fonseca, V. G. da. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, *7*(2), 117–132. https://doi.org/10.1109/TEVC.2003.810758