Moead-framework: a modular MOEA/D Python framework

Geoffrey Pruvost¹, Bilel Derbel¹, and Arnaud Liefooghe¹

¹ Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France

Summary

The Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) is a general-purpose algorithm for approximating the Pareto set of multi-objective optimization problems (Zhang & Li, 2007). It decomposes the original multi-objective problem into a number of single-objective optimization sub-problems and then uses an evolutionary process to optimize these sub-problems simultaneously and cooperatively. MOEA/D is a state-of-the-art algorithm in aggregation-based approaches for multi-objective optimization.

The goal of the moead-framework Python package is to provide a modular framework for scientists and researchers interested in experimenting with MOEA/D and its numerous variants.

Statement of Need

The MOEA/D algorithm is now considered as a framework. MOEA/D is the basis of many variants that improve or add new components to improve MOEA/D performance. The first version of MOEA/D and its most famous variants (Li & Zhang, 2009; Zhang et al., 2009) are implemented in recent multi-objective optimization software such as pymoo (Blank & Deb, 2020), pygmo (Biscani & Izzo, 2020) and jMetal (Nebro et al., 2015). These implementations offer many state-of-the-art algorithms, visualization tools or parallelization abstraction, but they are not modular enough to test easily all MOEA/D components. The modular R package MOEADr (Campelo et al., 2020) focuses on MOEA/D and allows the definition of different variants for each component of MOEA/D. While some modular frameworks already exist in Python for evolutionary algorithms such as DEAP (Fortin et al., 2012) or ModEA (van Rijn et al., 2016), these do not (easily) support implementing MOEA/D variants. Instead, they focus mostly on single-objective optimization and CMA-ES variants respectively.

With the moead-framework package, we aim to provide the modularity of the MOEADr package by using the flexibility of Python. Indeed, we want to allow the user to update the behavior of MOEA/D components in their research works without being limited by the package itself. The package is focused on a modular architecture for easily adding, updating or testing the components of MOEA/D and for customizing how components interact with each other. Indeed, in contrast with other existing implementations, moead-framework does not limit the users with a limited number of components available as parameters (8 components are available in MOEADr). Users can easily restructure the 10 existing components of the moead-framework and include new ones to easily add new features without altering existing components. Components are not only customizable with parameters as with MOEADr, but in fact they can be added with the inheritance mechanism on the main run() method of each algorithm.

For example, the moead-framework package was used for creating novel sub-problem selection strategies and analyzing them (Pruvost, Derbel, Liefooghe, Li, et al., 2020), and for rewriting
the component used to generate new candidate (offspring) solutions with a variant based on Walsh surrogates (Pruvost, Derbel, Leflooghe, Verel, et al., 2020).

<table>
<thead>
<tr>
<th>Software</th>
<th>Can add a new algorithm</th>
<th>Can modify the components of the algorithms in a modular way</th>
<th>Can add components to algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>moead-framework</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MOEADr</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>pymoo</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>pygmo</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>jMetal</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

**Documentation**

The documentation is available at the following URL: moead-framework.github.io/framework/.

A complete example and all components are described in details. Two tutorials are made available for the user to experiment with their own multi-objective optimization problem and to implement their own MOEA/D variants.

**Acknowledgements**

This work was supported by the French national research agency (ANR-16-CE23-0013-01) and the Research Grants Council of Hong Kong (RGC Project No. A-CityU101/16).

**References**


Pruvost, G., Derbel, B., Leflooghe, A., Li, K., & Zhang, Q. (2020). On the combined impact of population size and sub-problem selection in MOEA/d. In L. Paquette & C. Zarges (Eds.), *Evolutionary computation in combinatorial optimization* (pp. 131–147). Springer International Publishing. [https://doi.org/10.1007/978-3-030-43680-3_9](https://doi.org/10.1007/978-3-030-43680-3_9)

