Understanding Multi-objective Evolutionary Algorithms through Component Oriented Design

Claus Aranha[†], Felipe Campelo[††], Lucas S. Batista[††],

筑波大学システム情報工学系[†], Federal University of Minas Gerais, Brazil[††],

1 Introduction

Evolutionary Algorithms (EAs) are powerful meta heuristic methods that can be divided into multiple constituent components such as selection operators, variation operators, and stop criteria.

Traditionally, EAs have been proposed as monolithic "novel algorithms", in the worst cases using metaphor-based nomenclature that masks similarities between different methods, such as "Gray Wolf Optimization" or "Cat Swarm Optimization". We argue that this focus presents at least two main drawbacks: (i) it obscures the exact contribution of each proposed method in comparison with former and concurrent approaches; and (ii) it does not lend itself well to the study of individual contributions of algorithmic components. These issues can lead to duplication of efforts and, incidentally, to a multiplication of methods in the field [16)].

Recently, however, there has been an effort towards a more component oriented approach to algorithmic design and analysis. In this approach, an EA optimizer is seen not as a monolithic bloc, but rather as a composition of multiple, specialized components. This component oriented approach to algorithm investigation and development allows researchers to identify more clearly the level of contribution of each component to the overall performance of the algorithm. They also allow users to more easily implement and test each component, streamlining the development, adaptation, and test of new ideas, as well as the reproducibility of results. Moreover, this approach also allows for automated algorithm generation and fine tuning of parameters based on existing components [1)].

In this paper, we discuss how a component oriented view can be used to provide a more transparent understanding of new developments, promoting the exchange of ideas and the reproducibility of results. To illustrate this idea, we describe the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [21)] in a component-oriented framework. We show how this framework can be used to facilitate the comparison of different algorithms, the automated parameter tuning, and even the automated discovery of new algorithms. This framework is available as an open source R library, so that other researchers can also apply this methodology to their own works.

This paper is organized as follows: Section 2 reviews the MOEA/D algorithm and describes the component oriented framework. Section 3 shows the case study that illustrates how to use the proposed framework to auto-configure and compare different algorithms using the same component oriented standard. Finally, Section 4 concludes the paper, describing the open source R package and future directions for this work.

2 Component Oriented MOEA/D

2.1 Background

The Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D) approaches the Multi-objective Optimization Problem (MOP) by generating a set of scalar sub-problems. Each sub-problem is a weighted linear combination of the original problems. This process of generating scalar sub-problems is called a *decomposition*. If decomposed correctly, the optimization of each

Understanding Multi-objective Evolutionary Algorithms through Component Oriented Design

[†] Claus Aranha(caranha@cs.tsukuba.ac.jp)
[††] Felipe Campelo(fcampelo@ufmg.br)
[††] Lucas S. Batista(lusoba@ufmg.br)

Faculty of Systems and Information Engineering, University of Tsukuba (†)

Department of Electrical Engineering, Universidade Federal de Minas Gerais (††)

sub-problem in parallel generates a set of non-dominated solutions to the original MOP.

The MOEA/D in its current form was proposed by Zhang and Li [21], while earlier examples of decomposition approaches can be traced to Ishibuchi [9] and Murata [13]. In recent years, many researcers have tried to improve on the MOEA/D, and develop new algorithms derived from it. Trived et al. [19] made a recent survey of MOEAs based on decomposition.

In this context we present a component-oriented formulation of the MOEA/D that tries to tie together these different works under a similar language. In this framework, we define the different algorithms as different combinations of a unified pool of components. These components can be independently added, removed, modified or recombined from the *algorithmic composition*, either manually, or by automated testing and tuning programs.

2.2 The MOEA/D Component-wise

Let $\mathbf{f}(\mathbf{x})$ be a continuous Multi Objective Problem with m objectives, subject to inequality and equality constraints. The goal of the MOEA/D is to find a set of N solutions $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N\}$ that approximates the Pareto Front by solving a set of N scalar *sub-problems* which decompose the original MOP.

In the proposed framework, we re-define the MOEA/D and its derivations as a *composition* of multiple functions, each performing a different role in the method. Algorithm 1 summarizes the relationships between the different component types. In more specific terms, we can describe four stages in an algorithmic composition of the MOEA/D.

First, the algorithm uses a *Decomposition Strategy* to generate the sub problems from the original MOP. This generates a set of N *weight vector* which defines the decomposition into subproblems. To each subproblem is assigned a particular *incumbent solution* $x_i \in \mathbf{X}$.

Second, the set $\mathbf{X}$ of solutions is generated. In the first iteration, this set is generated randomly. In subsequent iterations it is generated based on a *Variation Stack*. The variation stack is composed of a set of *Variation Operators*, which are applied sequentially to $\mathbf{X}$ in order to generate new solutions. Our definition of a variation operator includes *Repair Operators* and *Local Search Operators* as special cases of variation operators in the variation stack.

Additionally, a *Neighborhood Assignment Strategy* is often employed to improve the performance of algorithm. It defines limits to the exchange of information between incumbent solutions when executing the variation stack.

Third, the solutions are evaluated on the original subproblem, and then a *Objective Scaling Strategy* and an *Aggregation Function* are applied. Objective Scaling defines how to treat differences in the ranges of objective values. Aggregation Function describes how the weight vectors are translated into the objective value of the sub problems. The simplest approach is simply a weighted sum of the weight vector defined by the decomposition strategy, but more complex methods do exist.

After this, an *Update Strategy* is used to determine whether which of the new solutions generated by the variation stack are assigned to which subproblems as incumbents. In this stage the neighborhood assignment is also used to limit the exchange of information between subproblems.

Finally, a *Termination Criteria* defines at what point the algorithm stops, usually based on total number of evaluations, or running time.

The open source R package *MOEADr*, made available by the authors in the CRAN repository[2], offers an implementation of this framework, including examples of some components from each class, drawn both from MOEA/D literature and the wider body of Multi objective Evolutionary Algorithms in general. While a detailed description of their implementation is outside the scope of this work, a list of components from the literature included in the package is summarized in Table 1.

When implementing components in this framework, special care must be taken to guarantee the modularity of the definitions so that each component is independent from design choices made for the others. This characteristic allows the free exchange of components while guaranteeing the correct flow of the MOEA/D, at the cost of some

Algorithm 1 Component-wise MOEA/D structure

```
Require: Objective functions $\mathbf{f}(\cdot)$; Constraint functions $\mathbf{g}(\cdot)$; Component-specific input parameters;
 1: $t \leftarrow 0$; $run \leftarrow$ TRUE
 2: Generate initial population $\mathbf{X}^{(t)}$ by random sampling.
 3: Generate weights $\Lambda$                                  ▷ Decomposition strategies
 4: while $run$ do
 5:     Define or update neighborhoods $B$                      ▷ Neighborhood assignment strategies
 6:     Copy incumbent solution set $\mathbf{X}^{(t)}$ into $\mathbf{X}'^{(t)}$
 7:     for each variation operator $\mathbf{v} \in \mathcal{V}$ do
 8:         $\mathbf{X}'^{(t)} \leftarrow v(\mathbf{X}'^{(t)})$                         ▷ Variation Stack
 9:     end for
10:     Evaluate solutions in $\mathbf{X}^{(t)}$ and $\mathbf{X}'^{(t)}$     ▷ Aggregation functions and Constraint handling
11:     Define next population $\mathbf{X}^{(t+1)}$                ▷ Update strategies
12:     Update $run$ flag; $t \leftarrow t + 1$                     ▷ Stop criteria
13: end while
14: return $\mathbf{X}^{(\mathbf{t})}$; $\mathbf{f}\left(\mathbf{X}^{(t)}\right)$
```

implementation overhead. This also simplifies the use of automated algorithm assembly and tuning methods, as well as efforts for replicating and testing algorithms from the literature.

3 Case Study

We present a case study that demonstrate some possible applications of the component wise approach to the study and development of MOEA/Ds. We focus on two aspects that we believe may be of immediate interest for researchers: fast replication of existing methods, and automated algorithm assembly and tuning.

3.1 Replicating Published Variants

One key aspect of scientific research that is often challenging in the field of evolutionary computation is the ability to independently replicate published methods and results. Reproducibility is essential, not only for promoting faster development of the field, but also to allow independent researchers to quickly and easily audit published methods so that inconsistencies can be quickly detected and corrected.

In this aspect, using component-based approaches, such as the one proposed in this work, can facilitate the task of expressing and studying new contributions to the MOEA literature. The *MOEADr* package allows researchers to quickly test new proposed improvements by providing a standard implementation of existing modules from the literature and, therefore, requiring minimal additional implementation in order to reproduce existing variants. Together with other packages available in the *R* ecosystem, such as *smoof* or *emoa*, it is possible for researchers to reproduce whole experimental sections from the published literature relatively easily.

As an example, we present the component-wise description of two classic MOEA/D versions in Table 2: the first MOEA/D presented in Section V-E of Zhang and Li's work[21)] , and the MOEA/D-DE presented in Li and Zhang's 2009 paper[10)] . By expressing these two methods in terms of component values, their similarities become immediately apparent, and their differences are similarly highlighted.

It is interesting to highlight how the component-wise modeling helps a straightforward comparison of MOEA/D variants. By expressing the original MOEA/D and the MOEA/D-DE as in Table 2, attention is quickly drawn to what actually differs between the two methods – the replacement of SBX by Differential Mutation, the possibility of out-of-neighborhood sampling for variation ($\delta_p = 0.9$), and the use of the restricted neighborhood replacement to alleviate the greediness of the orig-

Table 1 Components currently available in the *MOEADr* package

Component Class	**Name**	**User Parameters**
Decomposition Method	SLD[21]	$h \in \mathbb{Z}_{>0}$
	MSLD[5]	$\mathbf{h} \in \mathbb{Z}_{>0}^{K}$; $\tau \in (0,1]^{K}$
	Uniform[18]	$N \in \mathbb{Z}_{>0}$
Scalar Aggregation Function	WS	–
	WT[12]	–
	AWT[14]	–
	PBI[21]	$\theta^{pbi} \in \mathbb{R}_{>0}$
	iPBI[15]	$\theta^{ipbi} \in \mathbb{R}_{>0}$
Objective Scaling	–	$type \in \{none;\ simple\}$
Neighborhood Assignment[4, 8]	–	$type \in \left\{by\ \lambda_i;\ by\ \mathbf{x}_i^{(t)}\right\}$
		$\delta_p \in [0,1]$
Variation Operators	SBX recombination[7]	$\eta_{\mathtt{X}} \in \mathbb{R}_{>0}$; $p_{\mathtt{X}} \in [0,1]$
	Polynomial mutation[6]	$\eta_{\mathtt{M}} \in \mathbb{R}_{>0}$; $p_{\mathtt{M}} \in [0,1]$
	Differential mutation[17]	$\phi \in \mathbb{R}_{>0}$ $basis \in \{rand;\ mean;\ wgi\}$
	Binomial recombination[17]	$\rho \in [0,1]$
	Truncation	–
	Local search[3, 18]	$type \in \{tpqa;\ dvls\}$ $\tau_{ls} \in \mathbb{Z}_{>0}$; $\gamma_{ls} \in [0,1]$ $\epsilon \in \mathbb{R}_{>0}$ (if $type = tpqa$)
Update Strategy	Standard[21]	–
	Restricted[10]	$nr \in \mathbb{Z}_{>0}$
	Best[20]	$nr \in \mathbb{Z}_{>0}$; $T_r \in \mathbb{Z}_{>0}$
Constraint Handling	Penalty functions	$\beta_v \in \mathbb{R}_{>0}$
	VBR	$type \in \{ts;\ sr;\ vt\}$ $p_f \in [0,1]$ (if $type = sr$)
Termination Criteria	Evaluations	$max_{eval} \in \mathbb{Z}_{>0}$
	Iterations	$max_{iter} \in \mathbb{Z}_{>0}$
	Time	$max_{time} \in \mathbb{R}_{>0}$

Table 2 Original MOEA/D vs. MOEA/D-DE

	MOEA/D [21]	MOEA/D-DE [10]
Decomp.	SLD	
Agg. Fun.	WT	
Scaling	none	
Neigh.	by λ, $T = 20$	
	$\delta_p = 1.0$	$\delta_p = 0.9$
Variation	SBX $\eta_{\mathtt{X}} = 20$; $p_{\mathtt{X}} = 1$	Diff. mut *basis* = "*rand*"; $\phi = 0.5$
	Polynomial mutation ($\eta_{\mathtt{M}} = 20$; $p_{\mathtt{M}} = 1/n_v$)	
Update	Standard	Restric. $nr = 2$
Constr.	none	
Stop	Number of Iterations	

inal MOEA/D update method. In other words, we cease to see these two methods as two different algorithms, and now see them as different compositions of the same base algorithm, which facilitates their comparison and analysis.

3.2 Automated Assembly and Tuning

Next we illustrate how the component-based framework could be used to facilitate the automated assembly and tuning of the MOEA/D. In this case study we use a set of benchmark problems as a training base to select a promising algorithmic configuration. The set of training problems is composed of ten test problems from the CEC 2009 competition, with dimensions ranging from 20 to 60.

The pool of components avialable for selection, from those listed in Table 1, is as follows: SLD or Uniform for the decomposition strategy; WT, PIB or AWT for the scalar aggregation function; weight-based or incumbent solution-based neighborhood assignment strategy; standard, restricted or best sub problem update; and stop criteria of $100,000$ evaluations.

The the variation stack pool required a bit more care. We defined a variation stack with five component slots, where the first two could be any of SBX, Polynomial Mutation, Differential Mutation, or Binomial Recombination; the third slot could be any of those, or "none", the fourth could be one of the local search operators, or "none", and the fifth operator was fixed as a trucation repair operator.

To select the components out of this pool, as well as each component's parameters, we used the *Iterated Racing* procedure[11] (irace). A total of $20,000$ runs were allocated for the procedure, and the inverted generational distance (IGD) was used as a measure of quality for the candidate compositions.

The irace procedure returned seven "final" configurations, which are summarized in Table 3. The "consensus" column indicates the proportion of these final configurations that had the specified values. As can be seen from the table, a unanimous consensus was obtained for almost all the components selected. The few exceptions are shown in Figure 1, where we can see that even when full consensus was not achieved, the solutions converged around a few values.

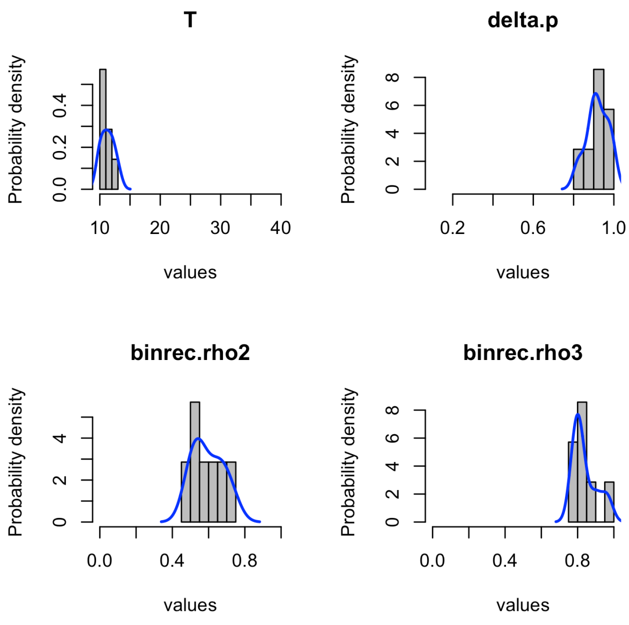

Fig. 1 Values of the numeric parameters returned by Iterated Racing.

Table 3 Final MOEA/D configuration returned by Iterated Racing.

	Value	**Consensus**
Decomp.	SLD	1.00
Agg. Fun.	AWT	1.00
Scaling	simple	Fixed
Neigh.	by $\mathbf{x}$	1.00
	$T = 11$	see Fig. 1
	$\delta_p = 0.909$	see Fig. 1
Var.	Diff. mut.	1.00
	$basis$ = "$rand$"	1.00
	$\phi \sim U(0, 1)$	Fixed
	Binom. recomb.	1.00
	$\rho_2 = 0.495$	see Fig. 1
	Binom. recomb.	1.00
	$\rho_3 = 0.899$	see Fig. 1
	Truncate	Fixed
Update	Restricted	1.00
	$nr = 1$	1.00

Let us discuss some interesting characteristics of this configuration. First we immediately observe that it selected two identical Binomial Recombination components for the variation stack, with different values for the ρ parameter. Using the definition of the Binomial Recombination and some calculations we can verify that this is equivalent of a single application of the operator using $\rho = 0.445$. This indicates that using some sort of parsimony pressure might be useful for future uses of this automated composition technique. We also note that the final composition used a smaller neighborhood size than what is usually found in the literature ($T = 11$), and that a very strict neighborhood update parameter ($nr = 1$). This indicate that the algorithm is aggressive in trying to limit the locality of the exchange of information, which might indicate an advantage to trying to maintain diversity in the population.

4 Conclusion

In this paper, we proposed a new formulation for the MOEA/D algorithm based on the concept of component based software architecture. In this formulation, the algorithm is broken up into independent components that can be separately replaced or configured. We showed that, using this configuration, it is possible to more directly compare and analyze different "algorithms" by identifying their common and diverging points. We also showed an example of using the component oriented architecture to automatically develop new algorithmic configurations.

To support this proposal, the authors have published an open-source *R* package that implements the proposed framework [2)]. This package includes components derived from many recent works on MOEA/D and other MOEAs. We expect that this package may help current and future researchers in the field to perform more rigorous comparisons of existing algorithmic compositions, and to develop new components based on current ideas.

Our current interest is to use this package to further explore the automatic generation of MOEAD/R algorithmic compositions. A small sample of this idea was presented in section 3.2 of this paper. Due to time constraints, this sample was limited regarding the size of the variation stack, and the types of problems explored. Also, we did not investigate issues such as parsimony of the composition. We intend to solve these issues in our future works.

Acknowledgements

This research was partially funded by Brazilian agencies Fapemig (grant TEC-APQ-01099-16) and CNPq (grant 404988/2016-4).

参考文献

1) Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. To DE or not to DE? multi-objective differential evolution revisited from a component-wise perspective. In *Lecture Notes in Computer Science*, pages 48–63. Springer Nature, 2015.

2) Felipe Campelo and Claus Aranha. *MOEADr: Component-Wise MOEA/D Implementation*, 2017. R package version 0.2.2.9012.

3) Bili Chen, Wenhua Zeng, Yangbin Lin, and Defu Zhang. A new local search-based multiobjective optimization algorithm. *IEEE Trans. Evolutionary Computation*, 19(1):50–73, 2015.

4) Tsung-Che Chiang and Yung-Pin Lai. MOEA/D-AMS: Improving MOEA/D by an adaptive mating selection mechanism. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 1473–1480, 2011.

5) K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. *IEEE Trans. Evolutionary Computation*, 18(4):577–601, 2014.

6) Kalyanmoy Deb and Samir Agrawal. A niched-penalty approach for constraint handling in genetic algorithms. In *Artificial Neural Nets and Genetic Algorithms*, pages 235–243. Springer Science + Business Media, 1999.

7) Kalyanmoy Deb and Hans-Georg Beyer. Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation*, 9(2):197–221, 2001.

8) H. Ishibuchi, N. Akedo, and Y. Nojima. Relation between neighborhood size and MOEA/D performance on many-objective problems. In *Evolutionary Multi-Criterion Optimization (EMO)*, volume 7811 of *Lecture Notes in Computer Science*, pages 459–474. Springer, 2013.

9) H. Ishibuchi and T. Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 28(3):392–403, 1998.

10) H. Li and Q. Zhang. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Trans. Evolutionary Computation*, 13(2):284–302, 2009.

11) Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

12) Kaisa Miettinen. *Nonlinear Multiobjective Optimization.* Kluwer Academic Publishers, Boston, 1999.

13) T. Murata and M. Gen. Cellular genetic algorithm for multi-objective optimization. In *Proc. 4th Asian Fuzzy System Symposium*, pages 538–542, 2000.

14) Yutao Qi, Xiaoliang Ma, Fang Liu, Licheng Jiao, Jianyong Sun, and Jianshe Wu. MOEA/D with adaptive weight adjustment. *Evolutionary Computation*, 22(2):231–264, 2014.

15) Hiroyuki Sato. Inverted PBI in MOEA/D and its impact on the search performance on multi and many-objective optimization. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 645–652, Vancouver, BC, Canada, 2014.

16) Kenneth Sörensen. Metaheuristics - the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18, 2013.

17) Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

18) Yan-yan Tan, Yong-chang Jiao, Hong Li, and Xin-kuan Wang. A modification to MOEA/D-DE for multiobjective optimization problems with complicated Pareto sets. *Information Sciences*, 213:14–38, 2012.

19) Anupam Trivedi, Dipti Srinivasan, Krishnendu Sanyal, and Abhiroop Ghosh. A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Trans. Evolutionary Computation*, PP(99):1–23, 2016.

20) Z. Wang, Q. Zhang, M. Gong, and A. Zhou. A replacement strategy for balancing convergence and diversity in MOEA/D. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 2132–2139, Beijing, China, 2014.

21) Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evolutionary Computation*, 11(6):712–731, 2007.