Efficient Multi-Objective Optimization using 2-Population Cooperative Coevolution [⋆]

Alexandru-Ciprian Zăvoianu[1,3], Edwin Lughofer[1], Wolfgang Amrhein[2,3], Erich Peter Klement[1,3]

[1] Department of Knowledge-based Mathematical Systems/Fuzzy Logic Laboratory Linz-Hagenberg, Johannes Kepler University of Linz, Austria
[2] Institute for Electrical Drives and Power Electronics, Johannes Kepler University of Linz, Austria
[3] ACCM, Austrian Center of Competence in Mechatronics, Linz, Austria

Abstract. We propose a 2-population cooperative coevolutionary optimization method that can efficiently solve multi-objective optimization problems as it successfully combines positive traits from classic multi-objective evolutionary algorithms and from newer optimization approaches that explore the concept of differential evolution. A key part of the algorithm lies in the proposed dual fitness sharing mechanism that is able to smoothly transfer information between the two coevolved populations without negatively impacting the independent evolutionary process behavior that characterizes each population.

Keywords: continuous multi-objective optimization, evolutionary algorithms, cooperative coevolution, differential evolution

1 Introduction and State-of-the-art

Part of our general research tasks are aimed at optimizing design parameters of electrical drives and deal with highly-dimensional multiple-objective optimization problems (MOOPs) that also display very lengthy run-times [17]. This is because our optimization scenarios require time-intensive design (fitness) evaluation functions that are based on finite element simulations. As such, having a robust and generally efficient (in number of required fitness evaluations) optimization algorithm is very important as it would significantly reduce the optimization run-times.

Like most MOOPs, the problems that we deal with rarely have a single solution and solving them means finding (an approximation of) a set of non-dominated solutions called the *Pareto-optimal set* [1]. Because of their inherent

⋆ This work was conducted in the frame of the research program at the Austrian Center of Competence in Mechatronics (ACCM), a part of the COMET K2 program of the Austrian government. The work-related projects are kindly supported by the Austrian government, the Upper Austrian government and the Johannes Kepler University Linz. The authors thank all involved partners for their support. This publication reflects only the authors' views.

ability to produce complete Pareto-optimal sets over single runs, multi-objective evolutionary algorithms (MOEAs) have emerged as one of the most successful soft computing techniques for solving MOOPs [1].

Among the early MOEAs, NSGA-II [3] and SPEA2 [16] proved to be quite effective and are still widely used in various application domains. At a high level of abstraction, both algorithms can be seen as MOOP orientated implementations of the same paradigm: *the* $(\mu + \lambda)$ *evolutionary strategy*. Moreover, both algorithms are highly elitist and use a two-tier selection for survival function based on Pareto and crowding indices. Canonically, both algorithms also use the same classic evolutionary operators: SBX crossover operator [2] and polynomial mutation [5];

More modern MOEAs (e.g., DEMO [13] and GDE [8]) wanted to exploit the very good performance exhibited by differential evolution (DE) operators [12] and replaced the SBX and polynomial mutation operators with various DE variants. Convergence benchmark tests [13] [9] show that differential evolution can help MOEAs to explore the decision space far more efficiently for several classes of MOOPs. Also, DE-based algorithms seem to be quite stable, as premature convergence can usually be avoided by choosing a good parameterization that stimulates a minor increase in population diversity [14].

For some problems though, MOEAs that use the SBX and polynomial mutation operators still significantly outperform DE-based algorithms. SBX and polynomial mutation also seem to be quite robust with regards to their parameterization as standard values are able to produce good results on a wide range of test problems.

In the next sections of this paper we shall describe our attempt to efficiently combine the two different search paradigms in order to obtain a hybrid optimization method that retains to a good extent all of the above described positive traits: efficient exploration of the search space, stability with regards to premature convergence, and robustness to parameterization.

2 Our Approach

In order to achieve our goal and create an optimization method that has the robustness of classic MOEA algorithms and also profits from the very good performance exhibited by DE operators, we use coevolution. More precisely, we apply 2-population cooperative coevolution where the first population, P, is evolved using the SPEA2 model, while the second population, Q, uses the DEMO/GDE3 evolutionary model coupled with a survival strategy based on *environmental selection* [16]. The general state of our $\underline{d}$ifferential $\underline{e}$volution-based, $\underline{c}$oevolutionary $\underline{m}$ulti-objective $\underline{o}$ptimization algorithm (**DECMO**) at a given generation is obtained by constructing $A = P \cup Q$.

Unsurprisingly, empirical results have shown that the way in which fitness is shared among the two coevolved populations has a crucial impact on the overall success of the method. We obtained the most stable behavior and the best results when using a dual fitness sharing mechanism based on the interleave between,

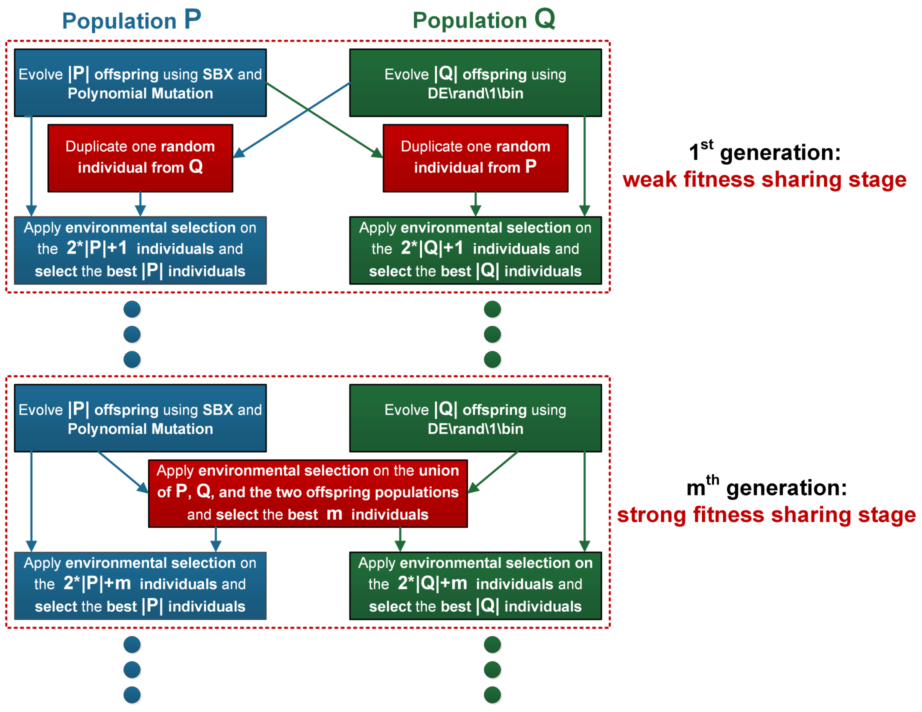
Population P
Population Q
Evolve |P| offspring using SBX and Polynomial Mutation
Evolve |Q| offspring using DE\rand\1\bin
Duplicate one random individual from Q
Duplicate one random individual from P
Apply environmental selection on the 2*|P|+1 individuals and select the best |P| individuals
Apply environmental selection on the 2*|Q|+1 individuals and select the best |Q| individuals
1st generation: weak fitness sharing stage
Evolve |P| offspring using SBX and Polynomial Mutation
Evolve |Q| offspring using DE\rand\1\bin
Apply environmental selection on the union of P, Q, and the two offspring populations and select the best m individuals
Apply environmental selection on the 2*|P|+m individuals and select the best |P| individuals
Apply environmental selection on the 2*|Q|+m individuals and select the best |Q| individuals
mth generation: strong fitness sharing stage

Fig. 1. Diagram of the proposed fitness sharing mechanism

generational, *weak sharing* stages and, fixed interval, *strong sharing* stages. As shown in Figure 1, the weak fitness sharing stage consists of trying to insert in each population one random offspring generated from the complementary population. In the strong sharing stage, which takes place every m generations, an elite subset is constructed by performing the union of A with the two offspring populations and extracting, via environmental selection, the best m individuals. Afterwards, environmental selection is again applied in an attempt to reintroduce (some of) these elite individuals in populations P and Q.

We make extensive use of the environmental selection operator because it inherently has two features that are, by default, beneficial to a coevolutionary process:

- a primary Pareto-based selection criterion - acts both as an inter and intra population fitness indicator
- a secondary crowding-based filtering mechanism - helps to preserve population diversity.

The above two features also characterize the non-dominated sorting [3] survival strategy proposed with NSGA-II. Choosing one method over the other for our initial DECMO prototype was simply a matter of personal preference on behalf of the authors.

The advantage of the previously described fitness sharing mechanism is that it only introduces one extra configuration parameter: m - the control parameter of the strong fitness sharing stage. Furthermore, we have performed several tests and discovered that the setting $m = (|P| + |Q|)/20$ is stable and yields good results.

3 Experimental Setup

For assessing the performance of our coevolutionary approach, we have made runs of up to 50000 individuals (75000 for one special case) using SPEA2, DEMO/DGE3, and DECMO. The population (and archive) sizes were fixed to 200 individuals for the first two methods, while for DECMO we used the setting $|P| = |Q| = 100$. All the results we report are aggregated from 50 independent runs for each MOEA - MOOP combination. In the case of DEMO and GDE3 we consider that conceptually, the overall evolutionary process is nearly identical, but for the sake of clarity we mention that we performed the tests using GDE3, which can also be considered a DEMO variant, namely DEMO^{NS-II}.

Test Problems In order to assess the performance of the three MOEAs we apply them on 10 standard, artificial, test problems from the evolutionary multi-objective literature. These problems have been specifically selected as they propose different degrees of difficulty and different convergence behaviors for the two categories of MOEAs we are interested in (i.e., classical and DE-based). The first five problems we use are selected from the DTLZ set [4], while the other five are KSW10 (based on Kursawe's function [10]), LZ09-F1 and LZ09-F2 - part of the LZ09 problem set [11], which is particularly difficult for classic MOEAs, WFG1[7], and ZDT6 - a problem with a non-uniform search space [15].

MOEA Parameterization We selected the standard parameterization for SPEA2: 0.9 for the crossover probability, 20 for the crossover distribution index, $1/L$ for the mutation probability (where L is the number of variables) and 20 for the mutation distribution index. In the case of DEMO/GDE3, we use a DE/rand/1/bin operator with the settings CR=0.3 and F=0.5 (recommended in [13]). The DE repair strategy we apply replaces any candidate value that is violating a boundary constraint with the boundary value. It should be noted that this particular repair strategy is biased as it offers an advantage in the case of problems where the true Pareto front lies on one of the bounds of the decision space. These settings are also used to parameterize the individual populations P and Q of DECMO. The m-parameter is set, according to the description at the end of Section 2, to a value of 10.

Assessment of Solution Quality The hypervolume associated with a solution set, has the advantage that it is the only Pareto front quality estimation metric for which there is a theoretical proof [6] of a monotonic behavior. For any optimization problem, the *true Pareto front* has the highest achievable hypervolume value. As for our test problems, where the true Pareto front is known, we assess the performance of a MOEA after a given number of fitness evaluations by reporting the hypervolume of the current population to the hypervolume of the true Pareto front.

4 Results - Comparative Performance

The plots in Figure 2 present the average convergence behavior of the three optimization methods. On all the test problems, DECMO displays a search performance that is similar, or even slightly better, than that of the best performing stand-alone strategy. It is noteworthy to highlight the average performance of DECMO on the LZ09-F1 problem where the coevolutionary approach initially displays the same rapid convergence as SPEA2 but switches to a DEMO/GDE3 convergence behavior as the latter seems to be more efficient in the final stage of the runs.

Table 1. Mean and standard deviation information regarding the number of function evaluations (nfe) required in order to reach a late-stage of convergence. The values are computed by considering only those individual runs that are able to reach a late-stage of convergence. The best result for each problem is highlighted and selected by considering only those MOEAs that are able to achieve a successful run ratio (srr) of 1.0 for that problem.

Problem	SPEA2			GDE3			DECMO		
	nfe		srr	nfe		srr	nfe		srr
	μ	σ		μ	σ		μ	σ	
DTLZ1	27908	3489.3	1.00	15068	810.8	1.00	12688	983.3	1.00
DTLZ2	7216	735.0	1.00	10348	923.9	1.00	7576	513.7	1.00
DTLZ4	5573	396.8	0.90	6164	337.3	1.00	5626	356.0	1.00
DTLZ6	69578	1712.0	0.94	4884	259.0	1.00	5980	271.8	1.00
DTLZ7	22364	1247.2	1.00	11940	622.4	1.00	13048	769.4	1.00
KSW10	6872	600.4	1.00	12564	735.9	1.00	7576	831.7	1.00
LZ09-F1	12416	2484.2	1.00	16728	930.9	1.00	12184	866.5	1.00
LZ09-F8	-	-	0.00	-	-	0.00	-	-	0.00
WFG1	39720	8289.1	0.40	21012	1131.7	0.98	21596	1669.2	1.00
ZFT6	33416	909.7	1.00	6884	433.5	1.00	7800	557.0	1.00

In [18] it has been argued that after reaching a solution set that accounts for over 85% of the true hypervolume, a MOEA reaches a *late-stage of convergence* where improvements generally come at a greater cost in terms of fitness evaluations. Because we consider that knowing the number of fitness evaluations (nfe) that are required in order to reach such a late-stage of convergence is very

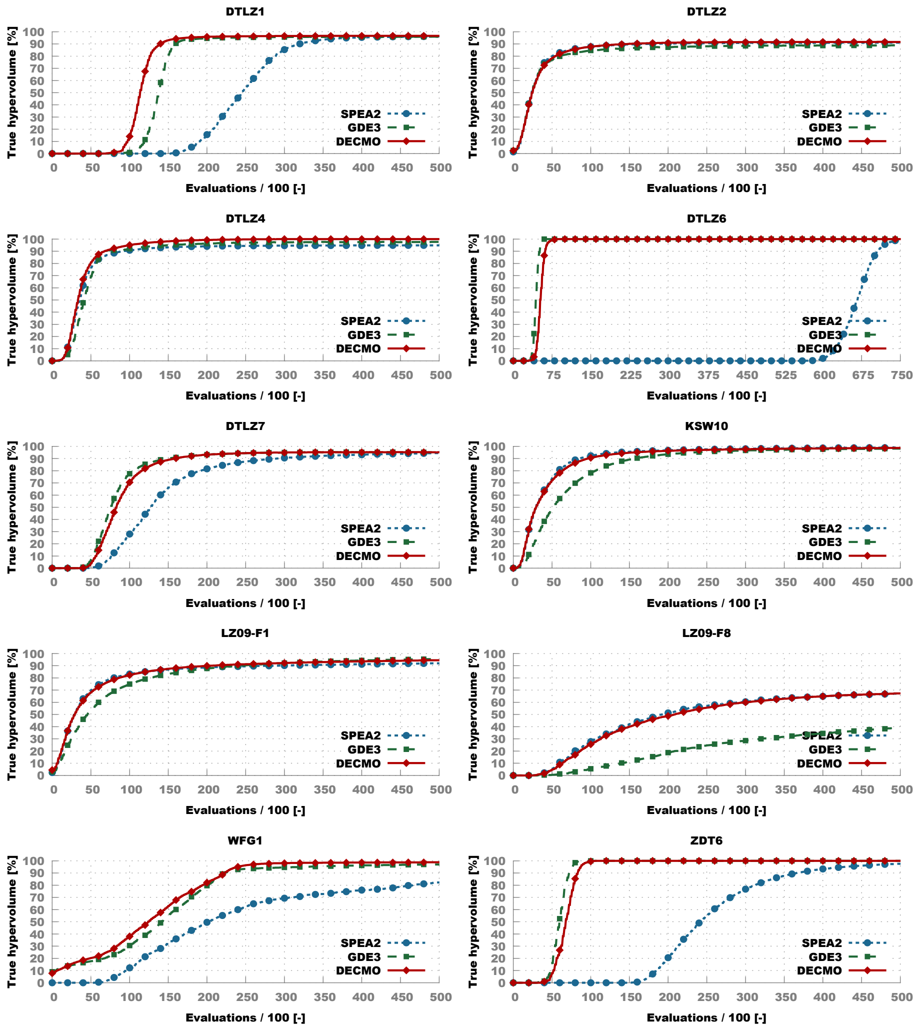

Fig. 2. The convergence behavior of the three MOEAs averaged over 50 runs

helpful in characterizing the search behavior of a multi-objective optimization algorithm, we have also measured this data and present the aggregated values in Table 1.

The results from Table 1 further underline the tendency that DECMO has to replicate the search behavior of the most successful strategy. This seems to be the case even on problems where both search strategies are having difficulties (e.g., LZ09-F8 and WFG1) as all or some of the runs are unable to reach a late-stage of convergence (i.e., srr < 1.0). For every test problem except LZ09-F1 (SPEA2 vs. DECMO), the observed differences between the performance of the

MOEAs that were able to achieve a successful run ratio (srr) of 1.0 are also statistically significant (one-sided Mann-Whitney-Wilcoxon test with a considered significance level of 0.05).

5 Conclusions and Future Work

The results we obtained on 10 benchmark MOOPs indicate that coevolution is quite successful at its aim of constructing a robust average between SBX and DE based methods when using an appropriate fitness sharing mechanism. Furthermore, in general, this combination of two different evolutionary search paradigms also seems to be quite stable and efficient as it is able to perform very well on different optimization scenarios.

Future work will revolve around studying in more depth the qualitative differences between the two coevolved populations and how to use this information in order to improve the overall optimization method. We also plan to develop a steady state asynchronous version of DECMO and to test the performance of the algorithm on MOOPs from the field of electrical drive design.

References

1. Coello, C., Lamont, G., Van Veldhuisen, D.: Evolutionary Algorithms for Solving Multi-Objective Problems. Genetic and Evolutionary Computation Series, Springer (2007)
2. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Systems 9, 115–148 (1995)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182 –197 (2002)
4. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: IEEE Congress on Evolutionary Computation (CEC 2002). pp. 825–830. IEEE Press (2002)
5. Deb, K., Goyal, M.: A combined genetic adaptive search (geneas) for engineering design. Computer Science and Informatics 26, 30–45 (1996)
6. Fleischer, M.: The measure of Pareto optima. applications to multi-objective metaheuristics. In: International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003). pp. 519–533. Springer (2003)
7. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation 10(5), 477–506 (2006)
8. Kukkonen, S., Lampinen, J.: GDE3: The third evolution step of generalized differential evolution. In: IEEE Congress on Evolutionary Computation (CEC 2005). pp. 443–450. IEEE Press (2005)
9. Kukkonen, S., Lampinen, J.: Performance assessment of Generalized Differential Evolution 3 with a given set of constrained multi-objective test problems. In: IEEE Congress on Evolutionary Computation (CEC 2009). pp. 1943–1950. IEEE Press (2009)

10. Kursawe, F.: A variant of evolution strategies for vector optimization. In: Workshop on Parallel Problem Solving from Nature (PPSN I). Lecture Notes in Computer Science, vol. 496, pp. 193–197. Springer (1991)
11. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. IEEE Transactions on Evolutionary Computation 13(2), 284–302 (2009)
12. Price, K., Storn, R., Lampinen, J.: Differential evolution. Springer (1997)
13. Robič, T., Filipič, B.: DEMO: Differential evolution for multiobjective optimization. In: International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005). pp. 520–533. Springer, Springer Berlin / Heidelberg (2005)
14. Zaharie, D.: A comparative analysis of crossover variants in differential evolution. In: Proceedings of the International Multiconference on Computer Science and Information Technology - IMCSIT 2007. pp. 171–181. Wisla: PTI (2007)
15. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary computation 8(2), 173–195 (2000)
16. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001). pp. 95–100. International Center for Numerical Methods in Engineering (CIMNE) (2002)
17. Zăvoianu, A.C., Bramerdorfer, G., Lughofer, E., Silber, S., Amrhein, W., Klement, E.P.: A hybrid soft computing approach for optimizing design parameters of electrical drives. In: International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2012). pp. 347–358. Springer (2013)
18. Zăvoianu, A.C., Lughofer, E., Koppelstätter, W., Weidenholzer, G., Amrhein, W., Klement, E.P.: On the performance of master-slave parallelization methods for multi-objective evolutionary algorithms. In: International Conference on Artificial Intelligence and Soft Computing (ICAISC 2013). pp. 122–134. Springer (2013)