Two Enhancements for Improving the Convergence Speed of a Robust Multi-Objective Coevolutionary Algorithm

Alexandru-Ciprian Zăvoianu*
Department of Knowledge-Based Mathematical Systems
Johannes Kepler University Linz
4040 Linz, Austria
ciprian.zavoianu@jku.at

Susanne Saminger-Platz
Department of Knowledge-Based Mathematical Systems
Johannes Kepler University Linz
4040 Linz, Austria
susanne.saminger-platz@jku.at

Edwin Lughofer
Department of Knowledge-Based Mathematical Systems
Johannes Kepler University Linz
4040 Linz, Austria
edwin.lughofer@jku.at

Wolfgang Amrhein
Institute for Electrical Drives and Power Electronics
Johannes Kepler University Linz
4040 Linz, Austria
wolfgang.amrhein@jku.at

ABSTRACT

We describe two enhancements that significantly improve the rapid convergence behavior of DECMO2 – a previously proposed robust coevolutionary algorithm that integrates three different multi-objective space exploration paradigms: differential evolution, two-tier Pareto-based selection for survival and decomposition-based evolutionary guidance. The first enhancement is a refined active search adaptation mechanism that relies on run-time sub-population performance indicators to estimate the convergence stage and dynamically adjust and steer certain parts of the coevolutionary process in order to improve its overall efficiency. The second enhancement consists in a directional intensification operator that is applied in the early part of the run during the decomposition-based search phases. This operator creates new random local linear individuals based on the recent historically successful solution candidates of a given directional decomposition vector. As the two efficiency-related enhancements are complementary, our results show that the resulting coevolutionary algorithm is a highly competitive improvement of the baseline strategy when considering a comprehensive test set aggregated from 25 (standard) benchmark multi-objective optimization problems.

CCS CONCEPTS

• **Theory of computation → Bio-inspired optimization**; Interactive computation; • **Applied computing → Multi-criterion optimization and decision-making**; • **General and reference** → *Performance*;

KEYWORDS

evolutionary multi-objective optimization, differential evolution, coevolution, sub-population dynamics, run-time performance assessment methodology

ACM Reference Format:
Alexandru-Ciprian Zăvoianu, Susanne Saminger-Platz, Edwin Lughofer, and Wolfgang Amrhein. 2018. Two Enhancements for Improving the Convergence Speed of a Robust Multi-Objective Coevolutionary Algorithm. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan,* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205549

1 INTRODUCTION

Generally, a multi-objective optimization problem (MOOP) can be defined as:

$$\text{minimize } F(x) = (f_1(x), \ldots, f_m(x))^T, \tag{1}$$

where the decision (variable) space is multi-dimensional (i.e., $x \in D^n \subset \mathbb{R}^n$) and $F : D^n \rightarrow \mathbb{R}^m$ contains m single-objective functions that need to be minimized simultaneously. If $m \geq 4$, one is usually said to deal with a many-objective optimization problem [11], so for the type of problems we focus on in the present work, $2 \leq m \leq 3$. The likely conflicting nature of the m individual objectives means that, apart from the inherent optimization-related difficulties (non-linear and/or multimodal and/or non-differentiable functions to be optimized), an important challenge related to MOOPs is that their solution comes in the form of a Pareto-optimal set (PS). The PS reunites all the individual solution candidates $x^* \in D^n$ with the property that there is no single element in D^n that is better than x^* with regard to all the considered objectives. Since for many MOOPs the PS is unknown or contains an infinity of elements, multi-objective solvers aim to discover high-quality Pareto non-dominated sets (PNs) that contain a limited number of solution candidates but are able to provide a near-perfect PS approximation.

During the past two decades, in light of their intrinsic ability to produce entire PNs after a single run, multi-objective optimization algorithms (MOEAs) have emerged as one of the most successful methods for solving MOOPs [1]. Although numerous valuable contributions have been made over the years, three main evolutionary

*Address all correspondence related to this article to this author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '18, July 15–19, 2018, Kyoto, Japan
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205549

computation paradigms stand out when considering their overall success, applicability and community acceptance:

(1) guiding the evolutionary process through a highly elitist selection for survival mechanism [5, 31] based on a primary Pareto non-dominance metric and a secondary criteria related to the distribution of individual solutions in objective space (i.e., a niching strategy), as originally suggested in [8, 9];
(2) integrating differential evolution (DE) strategies to further improve the performance of established evolutionary models [13, 28];
(3) decomposing the original MOOP in a sufficiently large number of well-spaced single-objective sub-problems (defined using classical objective aggregation techniques [17]) that are to be optimized simultaneously by the evolutionary process [27].

1.1 Motivation and approach

Our main motivation for the present research stems from the fact that (like all optimization methods) even state-of-the-art MOEAs are still bound by the "No Free Lunch Theorems for Optimization" (NFL) [25], which means that (at least) a parameter turning stage would be required in order for a given MOEA to deliver the best possible optimization performance an a given MOOP. Furthermore, even though the best performing MOEAs are quite robust, whenever restricting oneself to the usage of literature recommended parameterization settings, heterogeneous convergence behaviors can easily be observed on well studied benchmark MOOPs.

The NFL implications are particularly problematic for the ever increasing number of MOEA practitioners because many of them:

(1) are not experts in multi-objective evolutionary computation methods and they apply MOEA solvers using fixed (i.e., recommended) parameterizations;
(2) tackle complicated industrial problems where even state-of-the-art surrogate-based modeling techniques [19] are not fully applicable when wishing to reduce the reliance on intensive simulations [18] and thus the total number of optimization runs and of individuals that can be evaluated during these runs is rather limited;
(3) often adopt a form of interactive usage of MOEA-based optimization where (for computationally-intensive problems) the solver is stopped as soon as it has discovered a sufficiently good/interesting PN [21].

Given some of the insights in [26], there have been surprisingly few attempts to alleviate the aforementioned practical and theoretical concerns via coevolutionary strategies. The approaches described in [2], [24] and [32] represent some of the exceptions and the DECMO2 algorithm [32] is of particular interest since it is based on the simple idea of integrating all the three major multi-objective evolutionary paradigms inside a coevolutionary process that aims to deliver robustness: i.e., generally effective run-time convergence over a wide range of MOOPs when using a fixed parameterization. In the present work, we describe two complementary enhancements that further improve the convergence speed of DECMO2 without impacting its robustness.

2 COEVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION

Algorithm 1 contains the high-level algorithmic description of **DECMO2++**, the new enhanced multi-objective coevolutionary optimization strategy we propose.

The algorithmic description of the original DECMO2 coevolutionary strategy can also be inferred easily from Algorithm 1 by simply ignoring lines 8 to 16 and 30 to 43 as they correspond to the two novel enhancements described in Sections 2.2 and 2.3.

2.1 Baseline strategy

The main feature of the original DECMO2 coevolutionary strategy is that it effectively integrates three different multi-objective optimization search space exploration paradigms that have proven their individual strengths on multiple (and often complementary) types of MOOPs. Each individual evolutionary paradigm incorporated in DECMO2 operates on a separate set of individuals and there is a fitness sharing stage at the end of each generation.

Let us mark with P the first coevolved sub-population. Its members are generated via the SPEA2 [31] evolutionary model that is based on the *environmental selection* (notation $E_{sel}(Set, count)$) operator – a selection for survival mechanism that filters (i.e., returns *count* elite individuals from) an input Set of individuals based on Pareto dominance as a primary metric and a crowding distance in objective space as a secondary metric. Apart from E_{sel}, this evolutionary paradigm also makes use of the simulated binary crossover (SBX) [4] and polynomial mutation (PM) [3] genetic operators that were popularized by NSGA-II [5]. In Algorithm 1 line 44, the function **EvoGenSPEA2**(S, *count*) implements the SPEA2 evolutionary model to evolve *count* offspring from a parent set S and then applies the E_{sel} operator to select a total of $|S|$ solution candidates (from the combined set that reunites all parents and offspring) that will form the parent set of the next generation.

It is noteworthy that DECMO2 also uses E_{sel} as an active part of its fitness sharing stage and as such, a slightly modified version of the operator is needed in order to filter objective-wise duplicates from the returned set of elite individuals. Without this modification, repeated applications of the operator would lead to premature convergence on several test problems.

The second coevolved sub-population (marked by Q) implements an evolutionary model that resembles the ones initially proposed by the GDE3 [13] and DEMO [20] multi-objective solvers. The main idea is to exploit the very good performance displayed by the differential evolution paradigm [22] on continuous optimization problems with real-valued objective functions. This is achieved by replacing the SBX and PM operators with the *DE/rand/1/bin* strategy while preserving the Pareto-based selection for survival mechanism (i.e., E_{sel}) during the evolutionary cycle. In Algorithm 1, the function **EvoGenDE** implements the above described evolutionary model to generate and select future members of sub-population Q.

The third MOO paradigm incorporated in DECMO2 comes in the form of an archive (marked by A) that is maintained according to a decomposition-based principle similar to the one proposed in MOGLS [12] and popularized by the highly successful solver MOEA/D [27]. Even though at certain times a limited number of new individuals are evolved directly from the archive using

Algorithm 1 The new DECMO2++ coevolutionary model

1: **function DECMO2++**($problem$, a_{size}, $stopCriterion$)
2: $\langle p_{size}, q_{size}, b_{size} \rangle \leftarrow$ **ComputeSizes**(a_{size})
3: $\langle A, P, Q \rangle \leftarrow$ **Initialize**($problem, a_{size}, p_{size}, q_{size}$)
4: $\phi^P, \phi^Q, \phi^A \leftarrow 1 \ \wedge \ stage \leftarrow$ "$early$" $\wedge \ testGen \leftarrow true$
5: $p_b, q_b, \leftarrow (b_{size}/2) \ \wedge \ a_b, a_{wins} \leftarrow 0$
6: **while** $\neg stopCriterion$ **do**
7: **if** $testGen$ **and** $stage \neq$ "$late$" **then**
8: **if** $\phi^P < \phi^Q$ **and** $\phi^P < \phi^A$ **then**
9: $p_b \leftarrow 0 \ \wedge \ q_b, a_b \leftarrow (b_{size}/2)$
10: **end if**
11: **if** $\phi^Q < \phi^P$ **and** $\phi^Q < \phi^A$ **then**
12: $q_b \leftarrow 0 \ \wedge \ p_b, a_b \leftarrow (b_{size}/2)$
13: **end if**
14: **if** $\phi^A < \phi^P$ **and** $\phi^A < \phi^Q$ **then**
15: $a_b \leftarrow 0 \ \wedge \ p_b, q_b \leftarrow (b_{size}/2)$
16: **end if**
17: **if** $\phi^P > \phi^Q$ **and** $\phi^P > \phi^A$ **then**
18: $p_b \leftarrow b_{size} \ \wedge \ q_b, a_b \leftarrow 0$
19: **end if**
20: **if** $\phi^Q > \phi^P$ **and** $\phi^Q > \phi^A$ **then**
21: $q_b \leftarrow b_{size} \ \wedge \ p_b, a_b \leftarrow 0$
22: **end if**
23: **if** $\phi^A > \phi^P$ **and** $\phi^A > \phi^Q$ **then**
24: $a_b \leftarrow b_{size} \ \wedge \ p_b, q_b \leftarrow 0$
25: **end if**
26: **else**
27: $p_b, q_b \leftarrow 0 \ \wedge \ a_b \leftarrow b_{size}$
28: **end if**
29: $testGen \leftarrow \neg testGen$
30: **if** $stage =$ "$early$" **and** $\phi^P < 0.5$ **and** $\phi^Q < 0.5$ **then**
31: $b_{size} \leftarrow (b_{size}/2) \ \wedge \ stage \leftarrow$ "$middle$"
32: $p_{size}, q_{size} \leftarrow p_{size} + (b_{size}/2)$
33: **end if**
34: **if** $stage =$ "$middle$" **then**
35: **if** $\phi^P + \phi^Q < \phi^A$ **then**
36: $a_{wins} \leftarrow a_{wins} + 1$
37: **if** $a_{wins} = 5$ **then**
38: $stage \leftarrow$ "$late$"
39: **end if**
40: **else**
41: $a_{wins} \leftarrow 0$
42: **end if**
43: **end if**
44: $\langle P, \phi^P \rangle \leftarrow$ **EvoGenSPEA2**($P, \ p_{size} + p_b$)
45: $\langle Q, \phi^Q \rangle \leftarrow$ **EvoGenDE**($Q, \ q_{size} + q_b$)
46: $\phi^A \leftarrow$ **EvoDirArchiveInd**($A, \ a_b, \ stage$)
47: $E \leftarrow E_{sel}(P \cup Q \cup A, b_{size})$
48: $P \leftarrow E_{sel}(P \cup E, p_{size})\rangle$
49: $Q \leftarrow E_{sel}(Q \cup E, q_{size})$
50: **end while**
51: **return** $E_{sel}(P \cup Q \cup A, a_{size})$
52: **end function**

a *DE/rand/1/bin* strategy (i.e., the **EvoDirArchiveInd** function from Algorithm 1), A has a mainly passive purpose (hence our preference not to also refer to it as a sub-population) as its role is to preserve an accurate well-spread approximation of the PS. Spacing inside A is maintained with the help of a weighted Tschebyscheff distance measure, one of the well-known classical approaches [17] of tackling multi-objective optimization. As such, the archive is organized as a list of ordered pairs $\langle \lambda^i, x^i \rangle$ where a total of $|A|$ objective weight vectors are uniformly spread (and kept fixed throughout the optimization) and, during the run, x^i denotes the individual that can best minimize the objective space λ^i-weighted Tschebyscheff distance w.r.t. the current estimate of the optimal reference point.

The last important key feature of DECMO2 is an active run-time search adaptation mechanism that is designed to pivot towards the evolutionary paradigm that was more successful during the previous generation. This is done by rewarding (at each even-numbered generation) the part of the algorithm that implements the successful paradigm and allowing it generate an extra number of $b_{size} = \frac{1}{10} \cdot a_{size}$ bonus individuals. In order to determine the run-time comparative effectiveness of the three evolutionary paradigms, DECMO2 proposes a basic self-diagnostics process that uses as main indicators the archive insertion ratios achieved by the coevolved sub-populations (at each odd-numbered generation). In in Algorithm 1, these ratios are marked with ϕ^P, ϕ^Q and ϕ^A and they show the percentage of the offspring created inside the P and Q sub-populations or directly from the archive A that were also fit enough to warrant an archive insertion – i.e., that were able to bring an improvement to the current decomposition-based PN approximation. The original reward-schema proposed by DECMO2 is illustrated in Algorithm 1 on lines 17 to 25 where p_b, q_b, a_b denote the number of bonus individuals allocated to each paradigm in the next generation.

Apart from generating the $a_{size} = |A|$ uniformly spread weight vectors (each containing the number of objectives specified by the given $problem$), the **Initialize**($problem, a_{size}, p_{size}, q_{size}$) function (Algorithm 1, line 3) generates random uniformly distributed individuals and splits them into equally sized initial sub-populations such that $p_{size} = |P| = q_{size} = |Q| = \frac{a_{size} - b_{size}}{2}$.

2.2 Improved run-time search adaptation

The first enhancement of the original DECMO2 coevolutionary model comes in the form of an improved run-time search adaptation strategy that better exploits the comparative performance of the contained evolutionary paradigms. This search adaptation strategy is built on top of a more refined self-diagnostics process that also attempts to distinguish during the optimization run the current stage of convergence: "$early$" , "$middle$", or "$late$".

In contrast to the original DECMO2 search adaptation strategy, the improved version allows for a split of the b_{size} (generational) bonus evaluations that are to be awarded to the best performing evolutionary paradigm. As such, lines 8 to 16 of Algorithm 1 were added to complement the original strategy (lines 17 to 25) and enable it to deal with the case when one evolutionary paradigm underperforms but the other two paradigms achieve equal insertion ratios. This more refined split of the performance reward enables the integration of a larger number of bonus individuals at the start

of the optimization. Thus, the function **ComputeSizes**(a_{size}) on line 2 of Algorithm 1 uses the formulae:

$$\begin{cases} b_{size} & = \frac{a_{size}}{5} \\ p_{size} & = q_{size} = \frac{a_{size} - b_{size}}{2} \end{cases} \quad (2)$$

One empirical observation that was integrated in the improved search adaptation strategy is that the comparative performance of the three insertion ratios can also be used to distinguish between different stages of convergence. Thus, the first time the insertion ratios of both sub-populations fall below 0.5 (i.e., more than half of the generated offspring are not archive-wise competitive), the coevolutionary process is estimated to have entered the "*middle*" stage of convergence (lines 30 to 33 of Algorithm 1). During this stage, the number of bonus individuals is halved as both sub-populations are estimated to under-perform.

It is important to notice that, similarly to the original DECMO2 strategy, b_{size} also denotes the number of elite individuals (selected via the E_{sel} operator) that are responsible for fitness sharing at the end of each generation (lines 47 to 49 of Algorithm 1). By dynamically reducing b_{size} during the run, we are also influencing the fitness sharing process by passing fewer elite members between the sub-populations when the algorithm reaches a stage where highly dominating individuals are increasingly harder to find. By doing this, we aim to reduce external selection-pressure on the sub-populations and allow them to better exploit their internal evolutionary model.

If during the the middle stage of convergence there is a consistent trend (e.g., five consecutive generations) in which generating individuals directly from the decomposition-based archive is considerably more successful (i.e., $\phi^P + \phi^Q < \phi^A$), we consider that the optimization run entering a "*late*" stage of convergence (lines 34 to 43 of Algorithm 1). From this moment, until the end of the run, at every generation, the bonus individuals will only be used to generate new individuals directly from the directional archive (lines 27 and 46 of Algorithm 1).

2.3 Directional intensification operator

Given a weight vector $\lambda^i, 1 \leq i \leq |A|$ used by the directional archive A, the directional intensification operator we propose creates a new individual by applying a rather simple local linear recombination strategy on the last two historically successful solution candidates for λ^i.

More formally, if $\langle \lambda^i, y \rangle \in A$ and x was the individual that y replaced when the latter was inserted in A, using the directional intensification operator we obtain a new individual z for which:

$$z_i = \begin{cases} y_i + F2 \cdot (y_i - x_i) & \text{if} \quad U^i < CR2 \\ y_i & \text{if} \quad U^i \geq CR2 \end{cases}, \forall 1 \leq i \leq n \quad (3)$$

where, $U^1, \ldots, U^n$ are independent random variable uniformly distributed in $[0, 1]$ and $CR2 \in [0, 1]$ and $F2 > 0$ are control parameters. It is easy to remark that the proposed operator has strong similarities to a differential evolution approach (i.e., *rand/1/bin*) but its overall design aims at local intensification centered on the given weight vector λ^i. For example, when using the setting $CR2 = 0.5$, one half of the decision variables encoded in z will match exactly those encoded in y (the current best option for λ^i) while the other half of the variables in z will try to "speculate" the difference between y and x (the previous best option for λ^i) by enlarging local variable-wise differences by a factor of $F2$. In order to reduce the danger of overshooting (and then reflecting against) variable-wise lower and upper limits, we recommend the setting $F2 = 0.5$.

Using a set of limited but systematic benchmark tests, we have determined that applying this directional intensification operator generally brings the most benefits in the early stages of the optimization runs when the the individuals able to deliver consecutive directional-wise improvements are more heterogeneous. As such, in Algorithm 1 line 46, **EvoDirArchiveInd**(A, a_b, *stage*) exclusively uses the above described operator to generate a_b new individuals directly from the archive A – each for a different randomly selected weight vector – when *stage* = "*early*". Alteratively, the original DECMO2 *de/rand/1/bin* strategy is used inside **EvoDirArchiveInd** whenever the algorithm is at a different stage of the run or if the weight vector randomly selected for improvement contains less than 2 historically successful solution candidates.

3 PERFORMANCE EVALUATION

In order to empirically support our claim that the two proposed enhancements increase the convergence speed of DECMO2 we:

(1) performed experiments on a comprehensive test set aggregated from 25 benchmark MOOPs:
   - DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ6 and DTLZ7 from the problem set proposed in [6];
   - KSW10 – Kursawe's function with 10 variables and 2 objectives [14];
   - all nine problems from the LZ09 problem set [15];
   - WFG1, WFG2, WGF3, WFG4, WFG7, WFG8 and WFG9 from the problem set proposed in [10];
   - ZDT3 and ZDT6 from the problem set described in [30];
(2) made 100 independent repeats for each MOOP-MOEA combination and only report over averaged results;
(3) applied statistical significance testing when analyzing the differences between results
(4) have used the run-time hypervolume-based racing methodology that was introduced in [32] alongside DECMO2 in order to illustrate application-wise MOEA robustness and to easily highlight the differences in the comparative run-time performance of the tested MOEAs over an entire benchmark problem set

3.1 Racing methodology for comparing run-time MOEA performance / robustness

The main idea of the racing-based methodology proposed in [32] is to continuously measure a given unary PN quality indicator throughout the entire optimization runs on a given MOOP and, at pre-defined (equally-spaced) comparison stages, to award ranks to competing MOEAs based on their individual performance indicator achievement and a predefined ranking schema. For example, when wishing to compare between n_s algorithms, according to a *basic (natural) ranking schema*, at any given comparison stage, the worst performer will receive the rank n_s, and the best performing

algorithm will receive the rank of 1. Since MOEAs are highly stochastic processes, at each stage, the comparisons must take into consideration the average value of the performance indicator over multiple runs. In order to assess the comparative performance on an entire benchmark set, one simply needs to average (MOEA-wise and stage-wise) the ranks achieved for each individual MOOP from the benchmark.

The unary PN quality indicator that is recommended in [32] is the (normalized) hypervolume metric [29] (notation Ind_H) as it is the only unary indicator for which there is theoretical proof of a monotonic convergence behavior [7], and the hypervolume is easier to compute than the generational distance or the inverse generational distance [23] for problems with unknown PSs.

One of the main advantages of the Ind_H-based racing methodology is that (when using different ranking schemata) it can help to easily highlight some contrasts in the run-time convergence behavior (across the entire benchmark problem set) and thus it complements the information provided by a basic averaging of Ind_H performance across the benchmark set. This is especially true when plotting the average ranks in order to obtain so called run-time hypervolume-ranked performance curves (**HRPCs**).

For our numerical experiments, apart from the basic ranking schema, keeping in line with [32], at each comparison stage, we make two-by-two MOEA comparisons in increasing order of performance in which we:

- employ a *pessimistic ranking schema* that only awards a better rank when the difference between the average hypervolume-measured performance of two MOEAs is larger than a predefined threshold th – e.g., $th = 0.01$ would require a 1% difference between average hypervolume values at a given stage of comparison in order to award different ranks for that stage;
- employ a *statistical ranking schema* that only awards different ranks when the stage-wise differences between the average performance of two MOEAs is statistically significant when using a one-sided Mann-Whitney-Wilcoxon test [16] with a considered significance level of 0.025;
- adopt a bonus / penalty system that (a) awards the exceptional rank of 0 to recompense a MOEA that is estimated to have fully converged on a problem (i.e., $Ind_H > 0.99$) and (b) awards a penalty rank of $n_s + 1$ when a MOEA has not been able to discover a remotely relevant PS approximation (i.e., $Ind_H < 0.01$).

Independent of the MOOP to be solved, we fixed to 50,000 the total number of fitness evaluations (nfe) that each MOEA is allowed to perform during an optimization run. We considered 51 ranking stages: one after each 1000 fitness evaluations and one extra (i.e., ranking stage no. 0) after the first generation.

3.2 Tested algorithms and parameterization

We have chosen to compare the performance of the new DECMO2++ coevolutionary model with:

- DECMO2 - the baseline that we aim to improve over;
- MOEA/D-DE with Dynamic Resource Allocation [28] - a highly efficient solver based on decomposition that is known to deliver state-of-the-art solutions for many MOOPs;
- SPEA2 and GDE3 - the two Pareto dominance based MOEAs that originally proposed the search paradigms incorporated by the coevolved subpopulations of DECMO2++;
- NSGA-II [5] - historically, the best known and most successful MOEA (based on number of citations).

All the tested algorithms were parameterized using their literature recommended settings. For SPEA2 and NSGA-II we used an archive and population size of 200, 0.9 for the crossover probability and 20 for the crossover distribution index of SBX, $1/n$ for the mutation probability and 20 for the mutation distribution index of PM. For GDE3, we used a population size of 200 and the settings $CR = 0.3$ and $F = 0.5$ for the DE/rand/1/bin evolutionary strategy.

In the case of MOEA/D-DE we used an archive with $N = 300$ weight vectors for bi-objective MOOPs and $N = 595$ weight vectors for MOOPs with three objectives. The neighbourhood size was fixed to $T = 0.1 \cdot N$ and the neighborhood recombination factor δ was set to 0.9. The DE/rand/1/bin strategy was parameterized with the settings $CR = 1.0$ and $F = 0.5$ and the PM operator used the SPEA2 / NSGA-II standard parameterization.

In case of both DECMO2 and DECMO2++ we used an archive size of 200, the above described SPEA2 / NSGA-II standard parameterization for sub-population P and $CR = 0.2$ and $F = 0.5$ for the DE/rand/1/bin strategy used to evolve sub-population Q. When evolving individuals directly from A, we used the setting $CR = 1.0$ and $F = 0.5$ if the DE/rand/1/bin strategy was employed and $CR2 = 0.5$ and $F2 = 0.5$ if the individual was produced using the newly introduced directional intensification operator.

4 RESULTS AND INTERPRETATION

The average Ind_H-measured performance over the 25 benchmark MOOPs is presented in Figure 1 and it indicates that:

- the two coevolutionary approaches do exhibit a comparative rapid convergence behavior (with a small advantage for DECMO++) that does not seem to hinder their ability to reach competitive results at the end of the experiment;
- MOEA/D-DE DRA has a rather average performance in the early phases of the optimization run but catches up with the coevolutionary approaches towards the end;
- the four best performing MOEAs when $nfe > 20,000$ integrate differential evolution operators.

These benchmark-wide performance-related insights are confirmed by the HRPCs from Figure 2 that were obtained when comparing the 6 tested MOEAs using the run-time Ind_H-measured performance and different ranking schemata. Furthermore, the two pessimistic HRPCs indicate that the observed difference in performance between DECMO2++ and DECMO2 is somewhat considerable (i.e., statistically significant and observable even for $th = 0.05$) in the early phases of the optimization runs (i.e., $nfe < 10,000$) but negligible afterwards.

HRPCs offer the clearest and most relevant insights when analyzing the run-time performance of only two competing algorithms (as adding other algorithms in the mix can affect / bias a ranking that relies on successive two-by-two comparisons). Thus, Figure 3 focuses on a direct comparison between DECMO2++ and MOEA/D-DE DRA. The similar performance of the two algorithm towards the end of the optimization runs when using the basic and statistical

ranking schemata is in slight contrast with average Ind_H values from Figure 1 and with the results indicated by the pessimistic rankings in Figure 3. This indicates that over the entire benchmark dataset, when $nfe > 40,000$:

- there is a roughly equal number of problems on which one MOEA dominates the other;
- on a few of the test problems where DECMO2++ performs better, the differences in average Ind_H achievement are slightly larger than on the problems where MOEA/D-DE DRA ranks significantly better.

The HRPC plots from Figure 4 focus on the direct comparison between DECMO2++ and DECMO2 and support the claim that the two enhancements described in Sections 2.2 and 2.3 are able to visibly improve the convergence speed in the early stages of the optimization runs without impacting the robustness of the baseline coevolutionary strategy.

5 CONCLUSIONS AND FUTURE WORK

In this paper we described how to generally enhance the rapid convergence behavior of DECMO2 – a robust multi-objective coevolutionary algorithm – by redesigning its active search adaptation mechanism in order to facilitate the run-time estimation of the convergence stage and the subsequent application of a new genetic operator that, by design, aims to better exploit the directional-decomposition paradigm by intensifying local search in the early phases of the optimization run.

Since the performance of DECMO2++ is virtually identical to that of the baseline strategy during the middle and late stages of the run, future plans focus on redesigning the entire coevolutionary model such that, after existing the early stage of convergence, the solver will award a much more active role to the decomposition-based paradigm as this strategy seems to have a competitive advantage towards the end of the runs (or whenever the number of available fitness evaluations is high enough).

ACKNOWLEDGMENTS

This work was conducted within the Linz Center of Mechatronics (LCM) as a part of the COMET K2 program of the Austrian government. The COMET K2 projects at LCM are kindly supported by the Austrian and Upper Austrian governments and the participating scientific and industrial partners. The authors thank all involved partners for their support.

REFERENCES

[1] C. A. Coello Coello and G. B. Lamont. 2004. *Applications of multi-objective evolutionary algorithms.* World Scientific.
[2] C. A. Coello Coello and M. R. Sierra. 2003. A coevolutionary multi-objective evolutionary algorithm. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, Vol. 1. IEEE, 482–489.
[3] K. Deb. 2001. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons.
[4] K. Deb and R. B. Agrawal. 1995. Simulated binary crossover for continuous search space. *Complex Systems* 9 (1995), 115–148.
[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.
[6] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. 2002. Scalable multi-objective optimization test problems. In *IEEE Congress on Evolutionary Computation (CEC 2002)*. IEEE Press, 825–830.
[7] M. Fleischer. 2003. The Measure of Pareto Optima: Applications to Multi-objective Metaheuristics. In *International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*. Springer, 519–533.
[8] D. E. Goldberg. 1989. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Reading, MA.
[9] D. E. Goldberg and J. Richardson. 1987. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms.* Hillsdale, NJ: Lawrence Erlbaum, 41–49.
[10] S. Huband, L. Barone, L. While, and P. Hingston. 2005. A Scalable Multi-Objective Test Problem Toolkit. In *Evolutionary Multi-Criterion Optimization (EMO 2005) (Lecture Notes in Computer Science)*, Vol. 3410. 280–295.
[11] H. Ishibuchi, N. Tsukamoto, and Y. Nojima. 2008. Evolutionary many-objective optimization: A short review. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on.* IEEE, 2419–2426.
[12] A. Jaszkiewicz. 2002. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - A comparative experiment. *IEEE Transactions on Evolutionary Computation* 6, 4 (2002), 402–412.
[13] S. Kukkonen and J. Lampinen. 2005. GDE3: The third evolution step of generalized differential evolution. In *IEEE Congress on Evolutionary Computation (CEC 2005)*. IEEE Press, 443–450.
[14] F. Kursawe. 1991. 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. Springer, 193–197.
[15] H. Li and Q. Zhang. 2009. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation* 13, 2 (2009), 284–302.
[16] H. B. Mann and D. R. Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics* 18, 1 (1947), 50–60.
[17] K. Miettinen. 1999. *Nonlinear Multiobjective Optimization.* Kluwer Academic Publishers.
[18] A. Oyama, T. Kohira, H. Kemmotsu, T. Tatsukawa, and T. Watanabe. 2017. Simultaneous Structure Design Optimization of Multiple Car Models Using the K Computer. In *IEEE Symposium Series on Computational Intelligence.*
[19] M. Pilát and R. Neruda. 2014. Hypervolume-based local search in multi-objective evolutionary optimization. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation.* ACM, 637–644.
[20] T. Robič and B. Filipič. 2005. DEMO: Differential evolution for multiobjective optimization. In *International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*. Springer, Springer Berlin / Heidelberg, 520–533.
[21] S. Silber, G. Bramerdorfer, A. Dorninger, A. Fohler, J. Gerstmayr, W. KoppelstÃđtter, D. Reischl, G. Weidenholzer, and S. Weitzhofer. 2016. Coupled optimization in MagOpt. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 230, 4 (2016), 291–299.
[22] R. M. Storn and K. V. Price. 1997. Differential Evolution - a simple and effcient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11, 4 (December 1997), 341–359.
[23] D. A. Van Veldhuizen and G. B. Lamont. 1998. *Multiobjective Evolutionary Algorithm Research: A History and Analysis, Tech. Rep. TR-98-03.* Technical Report. Dept. Elec. Comput. Eng., Graduate School of Eng., Air Force Inst. Technol., Wright-Patterson, AFB, OH.
[24] J. A. Vrugt and B. A. Robinson. 2007. Improved evolutionary optimization from genetically adaptive multimethod search. *Proceedings of the National Academy of Sciences* 104, 3 (2007), 708–711.
[25] D. H. Wolpert and W. G. Macready. 1997. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation* 1, 1 (1997), 67–82.
[26] D. H. Wolpert and W. G. Macready. 2005. Coevolutionary free lunches. *Evolutionary Computation, IEEE Transactions on* 9, 6 (2005), 721–735.
[27] Q. Zhang and H. Li. 2007. MOEA/D: A multi-objective evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation* 11, 6 (December 2007), 712–731.
[28] Q. Zhang, W. Liu, and H. Li. 2009. *The Performance of a New Version of MOEA/D on CEC09 Unconstrained MOP Test Instances.* Technical Report. School of CS & EE, University of Essex.
[29] E. Zitzler. 1999. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications.* Ph.D. Dissertation. Swiss Federal Institute of Technology.
[30] E. Zitzler, K. Deb, and L. Thiele. 2000. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation* 8, 2 (2000), 173–195.
[31] E. Zitzler, M. Laumanns, and L. Thiele. 2002. 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)*. International Center for Numerical Methods in Engineering (CIMNE), 95–100.
[32] A.-C. Zăvoianu, E. Lughofer, G. Bramerdorfer, W. Amrhein, and E. P. Klement. 2014. DECMO2: a robust hybrid and adaptive multi-objective evolutionary algorithm. *Soft Computing* 19, 12 (2014), 3551–3569.

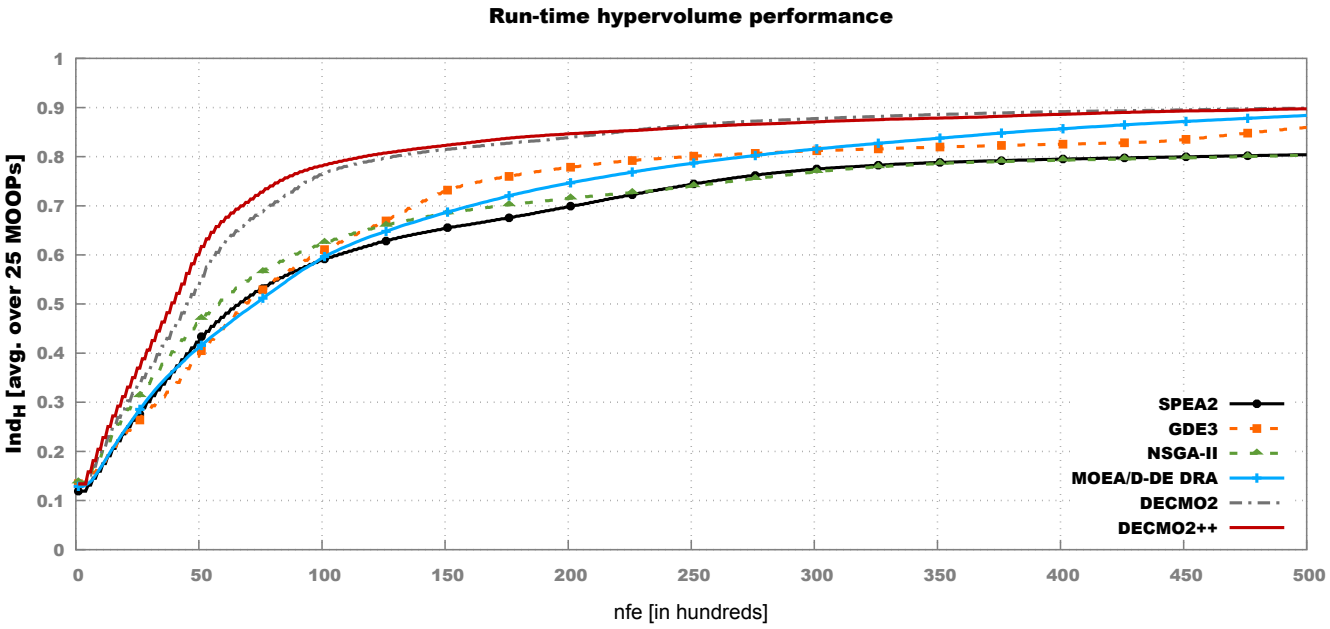

Figure 1: Run-time Ind_H-measured performance averaged over 25 artificial benchmark MOOPs

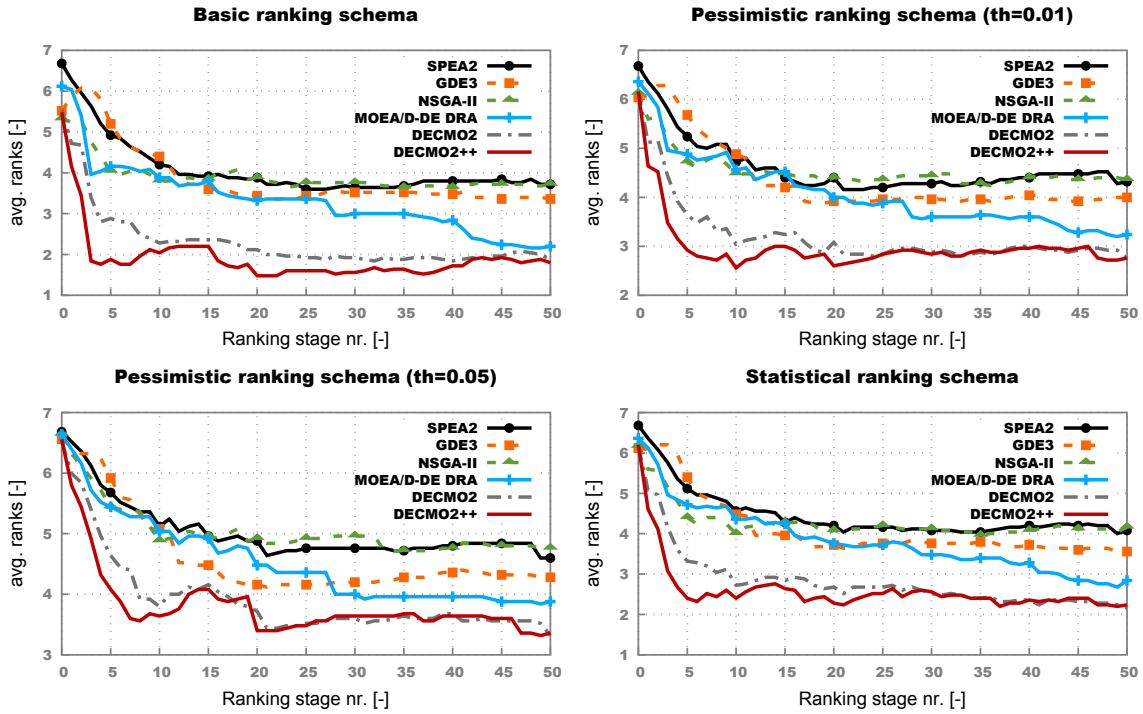

Figure 2: HRPCs obtained when comparing DECMO2++ with five other MOEAs across the 25 benchmark MOOPs.

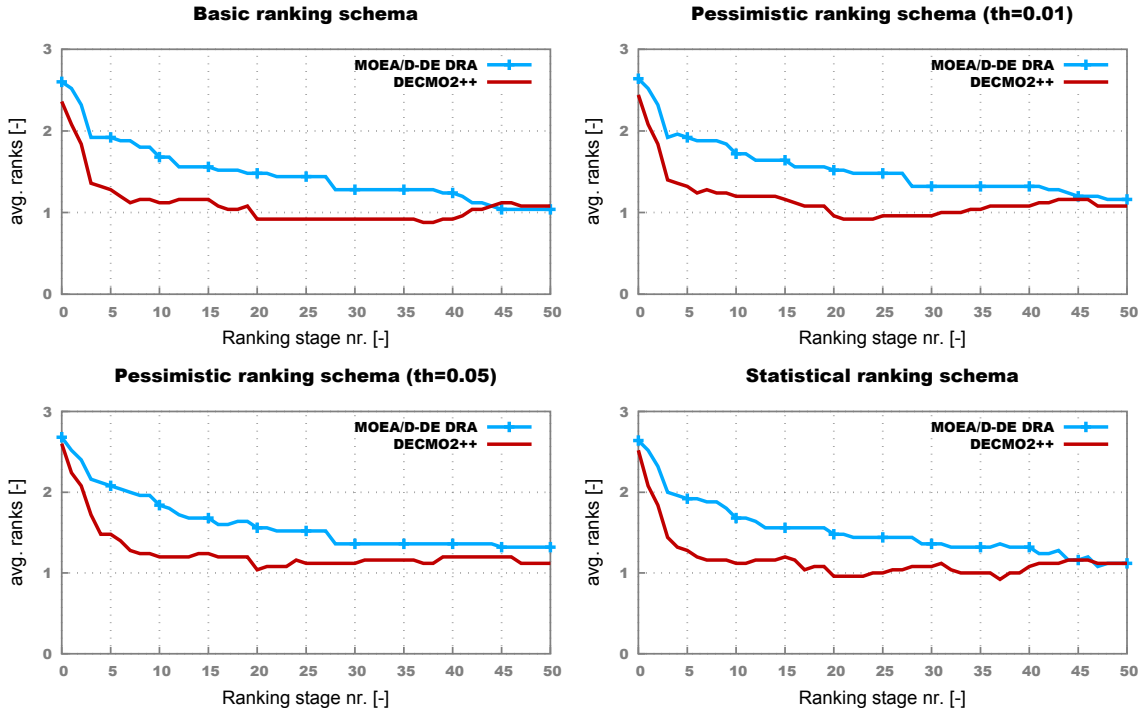

Figure 3: HRPCs obtained when comparing DECMO2++ with MOEA/D-DE DRA across the 25 benchmark MOOPs.

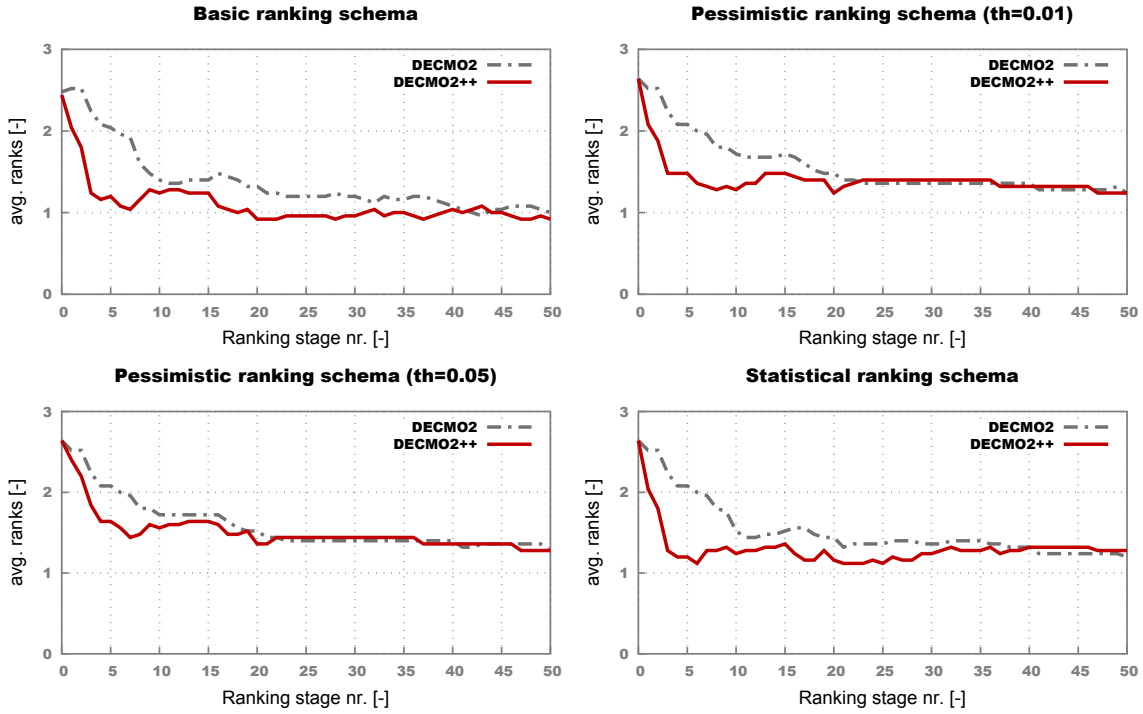

Figure 4: HRPCs obtained when comparing DECMO2++ with DECMO2 across the 25 benchmark MOOPs.