Soft Comput (2015) 19:3551–3569
DOI 10.1007/s00500-014-1308-7

FOCUS

DECMO2: a robust hybrid and adaptive multi-objective evolutionary algorithm

Alexandru-Ciprian Zăvoianu · Edwin Lughofer · Gerd Bramerdorfer · Wolfgang Amrhein · Erich Peter Klement

Published online: 4 June 2014

Abstract We describe a hybrid and adaptive coevolutionary optimization method that can efficiently solve a wide range of multi-objective optimization problems (MOOPs) as it successfully combines positive traits from three main classes of multi-objective evolutionary algorithms (MOEAs): classical approaches that use Pareto-based selection for survival criteria, approaches that rely on differential evolution, and decomposition-based strategies. A key part of our hybrid evolutionary approach lies in the proposed fitness sharing mechanism that is able to smoothly transfer information between the coevolved subpopulations without negatively impacting the specific evolutionary process behavior that characterizes each subpopulation. The proposed MOEA also features an adaptive allocation of fitness evaluations between the coevolved populations to increase robustness and favor the evolutionary search strategy that proves more successful for solving the MOOP at hand. Apart from the new evolutionary algorithm, this paper also contains the description of a new hypervolume and racing-based methodology aimed at providing practitioners from the field of multi-objective optimization with a simple means of analyzing/reporting the general comparative run-time performance of multi-objective optimization algorithms over large problem sets.

Communicated by E. Lughofer.

A.-C. Zăvoianu (✉) · E. Lughofer · E. P. Klement
Department of Knowledge-based Mathematical Systems/Fuzzy Logic Laboratory Linz-Hagenberg,
Johannes Kepler University of Linz, Linz, Austria
e-mail: ciprian.zavoianu@jku.at

A.-C. Zăvoianu · G. Bramerdorfer · W. Amrhein · E. P. Klement
LCM, Linz Center of Mechatronics, Linz, Austria

G. Bramerdorfer · W. Amrhein
Institute for Electrical Drives and Power Electronics,
Johannes Kepler University of Linz, Linz, Austria

Keywords Evolutionary computation · Hybrid multi-objective optimization · Coevolution · Adaptive allocation of fitness evaluations · Performance analysis methodology for MOOPs

1 Introduction

A multi-objective optimization problem (MOOP) can be defined as:

$$\text{minimize } O(x) = (o_1(x), \ldots, o_m(x))^{\mathrm{T}}, \tag{1}$$

where $x \in D$, D is called the decision (variable) space, $O : D \to \mathbb{R}^m$ consists of m single-objective functions that need to be minimized and R^m is called the objective space. In many cases the decision space of the MOOP is itself multidimensional, e.g., $D = \mathbb{R}^n$.

Usually, MOOPs do not have a single solution. This is because the objectives to be minimized ($o_1 \ldots o_m$ from (1)) are often conflicting in nature (e.g., cost vs. quality, risk vs. return on investment) and no $x \in D$ is able to simultaneously minimize all of them. To define a complete solution for a MOOP, we must first introduce the notions of Pareto dominance and Pareto optimality. When considering two solution candidates $x, y \in D$, solution x is said to Pareto-dominate solution y (notation: $x \preceq y$) if and only if $o_i(x) \leq o_i(y)$ for every $i \in \{1, \ldots, m\}$ and $o_j(x) < o_j(y)$ for at least one $j \in \{1, \ldots, m\}$ (i.e., x is better than y with regard to at least one objective and is not worse than y with regard to any objective). A solution candidate $x^* \in D$ with the property that there exists no $y \in D$ such that $y \preceq x^*$ is called a Pareto-optimal solution to (1). The set that reunites all the Pareto-optimal solutions is called the Pareto-optimal set (PS) and this set is the complete solution of the MOOP. The pro-

jection of the Pareto set on the objective space is called the Pareto front (PF).

Since for many problems, the PS is unknown and may contain an infinity of solutions, in real-life applications, decision makers often use the Pareto non-dominated set (PN) which contains a fixed number of solution candidates that are able to offer a good approximation of the PS. Therefore, finding high-quality Pareto non-dominated sets is the goal of most multi-objective optimization algorithms (MOOAs). Section 4 contains a detailed discussion regarding quality assessment in the case of PNs.

General research tasks in industrial environments often deal with highly dimensional ($6 \leq n \leq 60$) multiple-objective ($2 \leq m \leq 6$) optimization problems (MOOPs) that also may display very lengthy optimization run-times. This is because these industrial optimization scenarios require fitness evaluation functions that are extremely computationally intensive. For instance, in Yagoubi et al. (2011) MOOAs are used for the optimization of combustion in a diesel engine and the fitness evaluations require the usage of software emulators. In Jannot et al. (2011), finite element simulations are used during the fitness evaluation of an industrial MOOP from the field of electrical drive design. In these cases, despite using modern solving techniques from the field of soft computing like response surface methods, particle swarm optimization, and evolutionary algorithms, for many real-life MOOPs, a single optimization run can take several days, even when distributing the computations over a computer cluster.

As we strive to significantly reduce the run-times required to solve industrial MOOPs, our experience is grounded in three research lines:

- Applying non-linear surrogate modeling techniques on-the fly to significantly reduce the dependency on computationally intensive fitness evaluations (Zăvoianu et al. 2013a);
- deciding what type of parallelization/distribution method is more likely to deliver the best results taking into consideration the MOOAs that are used and the particularities of the hardware and software architecture (Zăvoianu et al. 2013c);
- trying to develop a new MOOA that generally requires fewer fitness evaluations to reach an acceptable solution, regardless of the specific MOOP considered, and that is robust with regard to its parameterization (Zăvoianu et al. 2013b);

While the third research direction is quite general and thus appeals to a considerable larger audience than the former two, it is also, by far, the most challenging. In the present article, building on past findings, we describe the results of our latest efforts directed towards developing an efficient and robust multi-objective optimization algorithm based on a hybrid and adaptive evolutionary model.

The remainder of this paper is organized as follows: Sect. 2 contains a short review on multi-objective evolutionary algorithms, Sect. 3 contains the detailed description of DECMO2, Sect. 4 presents our ideas on how to perform a general comparison of run-time MOOA performance over large problem sets and a formal description of what we understand by the syntagm "robust and efficient" in the context of MOOAs, Sect. 5 contains a comparative analysis of the performance of DECMO2 versus four other MOOAs when considering a wide range of artificial and real-life MOOPs, and Sect. 6 concludes the paper with a summary of achievements and some perspectives for future work.

2 Multi-objective evolutionary algorithms

Because of their inherent ability to produce complete Pareto non-dominated sets over single runs, multi-objective evolutionary algorithms (MOEAs) are a particular type of MOOAs that have emerged as one of the most successful soft computing models for solving MOOPs (Coello et al. 2007).

Among the early (by now, classical) MOEAs, NSGA-II (Deb et al. 2002a) and SPEA2 (Zitzler et al. 2002) 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 make use of similar, two-tier, selection for survival operators that combine Pareto ranking (primary quality measure) and crowding indices (equality discriminant). The names of these Pareto-based selection for survival operators are: non-dominated sorting (for NSGA-II) and environmental selection (for SPEA2). Canonically, both NSGA-II and SPEA2 also use the same genetic operators: simulated binary crossover—SBX (Deb and Agrawal 1995) and polynomial mutation—PM (Deb and Goyal 1996).

More modern MOEAs, like DEMO (Robič and Filipič 2005) and GDE3 (Kukkonen and Lampinen 2005) intend to exploit the very good performance exhibited by differential evolution (DE) operators (see Price et al. 1997) and replaced the SBX and polynomial mutation operators with various DE variants but maintained the elitist Pareto-based selection for survival mechanisms introduced by NSGA-II and SPEA2. Convergence benchmark tests (Robič and Filipič 2005; Kukkonen and Lampinen 2009) show that differential evolution can help MOEAs to explore the decision space far more efficiently for several classes of MOOPs.

Decomposition is the basic strategy behind many traditional mathematical programming methods for solving MOOPs. The idea is to transform the MOOP (as defined in

(1)) into a number of single-objective optimization problems, in each of which the objective is an aggregation of all the $o_i(x), i \in \{1, \ldots, m\}, x \in D$. Provided that the aggregation function is well defined, by combining the solutions of these single-objective optimization problems, one obtains a Pareto non-dominated set that approximates the solution of the initial MOOP. Miettinen (1999) provides a valuable review of several methods for constructing suitable aggregation functions. However, solving a different single-objective optimization problem for each solution in the PN is quite inefficient. A major breakthrough was achieved with the introduction of MOEA/D in Zhang and Li (2007) and its DE-based variant (MOEA/D-DE) in Li and Zhang (2009). This evolutionary algorithm decomposes a multi-objective optimization problem into a number of single-objective optimization subproblems that are then simultaneously optimized. Each subproblem is optimized through means of (restricted) evolutionary computation by only using information from several of its neighboring subproblems. It is noteworthy that MOEA/D proposes a different paradigm to multi-objective optimization than most of the previous MOEAs, and that this approach has proven quite successful, especially when dealing with problems with complicated Pareto-optimal sets. A version of MOEA/D (see Zhang et al. 2009) won the CEC2009 Competition dedicated to multi-objective optimization. As such, MOEA/D is considered state of the art by many researchers in the field.

In Zăvoianu et al. (2013b), we described DECMO—a hybrid multi-objective evolutionary algorithm based on cooperative coevolution that was able to effectively incorporate the pros of both individual search strategies upon which it was constructed. The idea was to simultaneously evolve two different subpopulations of equal size: subpopulation P was evolved using the SPEA2 evolutionary model, while subpopulation Q was evolved using DEMO/GDE3 principles. After various experiments, we discovered that a dual fitness sharing mechanism is able to induce the most stable behavior and to achieve competitive results. The DECMO fitness sharing mechanism consists of:

- Generational weak sharing stages (i.e., trying to insert in each subpopulation one random offspring generated in the complementary subpopulation);
- fixed interval strong sharing stages (i.e., constructing an elite subset of individuals from $A = P \cup Q$ and reinserting this subset in P and Q with the intent of spreading the best performing individuals across both subpopulations);

The aforementioned elite subset construction and the insertion and reinsertion operations are all performed by applying Pareto-based selection for survival operators (non-dominated sorting or environmental selection).

DECMO can be considered a successful proof of concept as it displayed a good performance on a benchmark composed of several artificial test problems (the coevolutionary algorithm was consistently able to replicate the behavior of the best performing individual strategy and, for some problems, even surpassed it).

3 Our proposal: DECMO2

In this section, we describe DECMO2, a new and significantly improved variant of our coevolutionary MOEA. Apart from Pareto-based elitism, differential evolution and coevolution, DECMO2 has two more key building blocks (integration of a decomposition strategy and search adaptation) and initial results show that it is able to compete with, and sometimes outperform, state-of-the-art approaches like MOEA/D and GDE3 over a wide range of multi-objective optimization problems.

Like its predecessor, DECMO2 is a hybrid method that uses two coevolved subpopulations of equal and fixed size. The first one, P ($|P| = P_{size}$), is evolved using the standard SPEA2 evolutionary model. The second subpopulation, Q ($|Q| = Q_{size}$), is evolved using differential evolution principles. Apart from these, DECMO2 also makes use of an external archive, A, maintained according to a decomposition-based strategy. The coevolutionary mechanism is redesigned to allow for an effective combination of all three search strategies and of a search adaptation mechanism.

We now proceed to describe the five building blocks of the DECMO2 multi-objective optimization algorithm and, finally, in Sect. 3.6 we present the algorithmic description of our hybrid evolutionary approach.

3.1 Pareto-based elitism

The cornerstone of the SPEA2 model (used in DECMO2 to evolve subpopulation P) is the environmental selection (for survival) operator introduced in Zitzler et al. (2002). Because we make extensive reference to it, we shall mark it with $E_{sel}(Pop, count)$, with the understanding that we refer to the procedure through which we select a subset of maximum $count$ individuals from an original set Pop. The first step is to assign a general rank to each individual $x, x \in Pop$. A lower value of this general rank indicates a higher quality individual. This general rank is the sum of two metrics, the raw rank $r(x)$ (2) and the density $d(x)$ (3). To compute the raw rank, each individual $x, x \in Pop$ is initially assigned a strength value $s(x)$ representing the number of solutions it Pareto-dominates in Pop. The raw rank assigned to x is computed by summing the strengths of all the individuals in

the population that Pareto-dominate individual x, i.e.,

$$r(x) = \sum_{y \in Pop\,:\, y \preceq x} s(y). \tag{2}$$

The density $d(x)$ of individual x is computed as the inverse of the distance to the k-th nearest neighbor, i.e.,

$$d(x) = \frac{1}{\text{dist}_E(x,k) + 2} \tag{3}$$

where $\text{dist}_E(x,k)$ is the Euclidean distance in objective space between individual x and its k-th nearest neighbor with $k = \sqrt{|Pop|}$. After each individual in *Pop* has been ranked, we simply select the first *count* individuals with lowest general rank values. The only noteworthy detail is that the $E_{\text{sel}}(Pop, count)$ variant we use across DECMO2 first removes all duplicate values from *Pop* and then begins the ranking process.

In DECMO2, at each generation t, $t \geq 1$, from the current subpopulation P, we use binary tournament selection, SBX and polynomial mutation to create a new offspring population P'. We then proceed to construct the union of the parent and offspring populations: $P' = P' \cup P$. Finally, the population of the next generation is obtained after applying the elitist environmental selection operator to extract the best individuals from this union: $P = E_{\text{sel}}(P', P_{\text{size}})$.

3.2 Differential evolution

Differential evolution is a global, population-based, stochastic optimization method introduced in Storn and Price (1997). By design, DE is especially suitable for continuous optimization problems that have real-valued objective functions. Like most evolutionary techniques, DE starts with a random initial population that is then gradually improved by means of selection, mutation and crossover operations.

In the case of DECMO2, at each generation t, $t \geq 1$, subpopulation Q will be evolved using the DE/rand/1/bin strategy according to an evolutionary model that is very similar to the ones proposed in DEMO (Robič and Filipič 2005) and GDE3 (Kukkonen and Lampinen 2005).

At first we perform the initialization: $Q' = \Phi$ and $Q'' = Q$. Afterwards, as long as $Q'' \neq \Phi$, we randomly select $x \in Q''$ and:

- firstly, we construct the mutant vector v using the rand/1 part of the DE strategy by randomly selecting three individuals $z_1, z_2, z_3 \in Q$ such that $z_1 \neq z_2 \neq z_3 \neq x$ and then computing:

$$v = z_1 + F(z_2 - z_3) \tag{4}$$

  where $F > 0$ is a control parameter.
- secondly, we generate the trial vector y using the binomial crossover part of the DE strategy:

$$y_i = \begin{cases} v_i & \text{if } U^i < CR \text{ or } i = j \\ x_i & \text{if } U^i \geq CR \text{ and } i \neq j \end{cases}, \tag{5}$$

  where j is a randomly chosen integer from $\{1, \ldots, n\}$, $U^1, \ldots, U^n$ are independent random variable uniformly distributed in [0, 1], and $CR \in [0, 1]$ is a control parameter. n is the dimensionality of the decision space (D) of the MOOP we wish to solve.
- thirdly, we remove x from the list of individuals that we must evolve in the current generation (i.e., $Q'' = Q'' \setminus \{x\}$) and update Q':

$$Q' = \begin{cases} Q' \cup \{x\} & \text{if } x \preceq y \\ Q' \cup \{y\} & \text{if } y \preceq x \\ Q' \cup \{x\} \cup \{y\} & \text{if } x \not\preceq y \text{ and } y \not\preceq x \end{cases} \tag{6}$$

At the end of the previously described cycle, it is highly likely that $|Q'| > Q_{size}$ because when x and y are not dominating each other, both individuals are added to Q' [i.e., the third case from (6)]. To obey the fixed subpopulation size design principle, when computing the population of the next generation, we apply the environmental selection operator [i.e., $Q = E_{\text{sel}}(Q', Q_{\text{size}})$].

3.3 Decomposition-based archive

Apart from the two coevolved populations, DECMO2 also uses an archive population, A, that is maintained according to a decomposition approach that is based on the Chebyshev distance.

Let us mark with:

- $z^* = (z_1^*, \ldots, z_m^*)$ the current optimal reference point of (1). More formally, $z_i^* = \min\left\{o_i(x) | x \in D^E\right\}$ for each $i \in \{1, \ldots, m\}$ when $D^E \subset D$ is the set containing all the individuals that have been evaluated during the evolutionary search till the current moment.
- $\lambda^i = (\lambda_1^i, \ldots, \lambda_m^i)$, $\lambda_j^i \geq 0$ for all $j \in \{1, \ldots, m\}$ and $\sum_{j=1}^m \lambda_j^i = 1$ and $i \in \{1, \ldots, |A|\}$ an arbitrary objective weight vector;
- $d_{\text{Cheb}}(x, \lambda^i) = \max_{1 \leq j \leq m} \left\{\lambda_j^i |o_i(x) - z_i^*|\right\}$, $x \in D$ the weighted Chebyshev distance between an individual $x \in D$ and the current optimal reference point;

For any MOOP we wish to solve, we consider a total of $|A|$ uniformly spread weight vectors: $\lambda^1, \ldots, \lambda^{|A|}$. These vectors are generated before the beginning of the evolutionary process and remain constant throughout the entire optimization run. When using them in the d_{Cheb} distance, these weight

vectors are the means through which we define the decomposition of the original MOOP problem into a number of $|A|$ single-objective optimization problems. As such, at any given moment during the optimization, archive A is organized as a set of pairs (2-tuples) $\langle \lambda^i, y^i \rangle$, $y^i \in D$, where λ^i is fixed and $y^i \in D^E$ has the property that it tries to minimize $d_{\text{Cheb}}(y^i, \lambda^i)$.

Given a current individual x that has just been generated during the optimization run, after performing the standard fitness evaluation:

- we update the reference point z^*;
- we construct A'—the improvable subset of the current archive set:

$$A' = \left\{ y^i | \exists \langle \lambda^i, y^i \rangle \in A : d_{\text{Cheb}}(x, \lambda^i) < d_{\text{Cheb}}(y^i, \lambda^i) \right\} \tag{7}$$

- if $A' \neq \Phi$, we:
  - mark with y^* that individual in A' that has the property that $\delta_{\text{Cheb}} = d_{\text{Cheb}}(y^*, \lambda^*) - d_{\text{Cheb}}(x, \lambda^*)$ is maximal (i.e., we apply a greedy selection principle);
  - update the archive by replacing the most improvable individual (i.e., $A = A \setminus \langle \lambda^*, y^* \rangle$) with the current individual: $A = A \cup \langle \lambda^*, x \rangle$;

It is worthy to note that the working principles behind the decomposition-based archive are inspired and fairly similar to those proposed by MOGLS [see Jaszkiewicz (2002)] and especially MOEA/D.

3.4 Search adaptation

By design, nearly all evolutionary models are adaptive in the sense that, by promoting "a survival of the fittest" strategy, these algorithms are forcing the evolved population to "adapt" with each passing generation (i.e., retain the genetic features that are beneficial for solving the current problem).

The central idea of the DECMO algorithm (Zăvoianu et al. 2013b) was to combine the different search behaviors of classical MOEAs that rely on SBX and PM with that of newer approaches that use DE operators. This was done in light of strong empirical evidence that one evolutionary model was by far better than the other one (when using standard parameterization) on several well-known problems [i.e., an occurrence subject to the No Free Lunch Theorem by Wolpert and Macready (1997)]. By effectively incorporating both search strategies, DECMO displayed a good average performance and proved its ability to adapt on a meta level (i.e., to mimic the best strategy for the problem at hand).

To improve the aforementioned results, for DECMO2 we designed a mechanism that is aimed to directly bias the coevolutionary process towards the particular search strategy that is more successful during the current part of the run. This is implemented by dynamically allocating at each odd generation $t, t \geq 1$ and $t \in \{2k+1 : k \in \mathbb{Z}\}$ an extra number (B_{size}) of bonus individuals that are to be created and evaluated by the search strategy that was able to achieve the highest ratio of archive insertions in the previous (even-numbered) generation. Therefore, at each even generation, we are computing:

- ϕ^P—the archive insertion ratio achieved by the P_{size} offspring generated in subpopulation P via tournament selection, SBX and PM;
- ϕ^Q—the archive insertion ratio achieved by the Q_{size} offspring generated in subpopulation Q via DE/rand/1/bin;
- ϕ^A—the archive insertion ratio achieved when creating B_{size} offspring by applying DE/rand/1/bin on individuals selected directly from A. When creating offspring directly from A, the parent individuals required by the DE/rand/1/bin strategy are selected such as to correspond to the B_{size} single-objective optimization problems (i.e., 2-tuples) that have not been updated for the longest periods.

Taking into account previous notation and descriptions, if, at an arbitrary even generation t, $\phi^P > \phi^Q$ and $\phi^P > \phi^A$, at generation $t+1$, the size of the offspring population (i.e., P') will be set to $P_{\text{size}} + B_{\text{size}}$. Likewise, if, at an arbitrary even generation t, $\phi^Q > \phi^P$ and $\phi^Q > \phi^A$, at generation $t+1$, after the stopping criterion is initially met (i.e., $Q'' = \Phi$), Q'' will be re-initialized with a (smaller) set containing B_{size} individuals randomly extracted from Q_t and the offspring generation process will resume until Q'' becomes void again. If neither of the previous two success conditions are met, then the B_{size} bonus offspring of generation $t+1$ will be created by applying DE/rand/1/bin on individuals selected directly from A.

As we have defined all the individual population subdivision in DECMO2, now, we can also give all the formulae that describe the relation between the size of the archive and the sizes of the two coevolved populations:

$$\begin{cases} |A| = P_{\text{size}} + Q_{\text{size}} + B_{\text{size}} \\ P_{\text{size}} = Q_{\text{size}} \\ B_{\text{size}} = \frac{|A|}{10} \end{cases} \tag{8}$$

3.5 Cooperative coevolution

Coevolution is a concept inspired from biological systems where two or more species (that are in a symbiotic, parasitic

or predatory relationship) gradually force each other to adapt (evolve) to either increase the efficiency of their symbiotic relationship or survive. For a valuable overview please see Chapter 6 from Luke (2013).

In the field of soft computing, coevolution is usually applied to population-based optimization methods and is implemented using subpopulations that are evolved simultaneously. N-population cooperative coevolution is a particular type of coevolutionary process that is modeled according to symbiotic relationships occurring in nature. The central idea is to break up complicated high-dimensional search spaces into N, much simpler, subspaces that are to be explored by independent (sub)populations. To discover high-quality solutions, it is necessary to (occasionally) share information regarding fitness between the different populations.

In DECMO2, the particular way in which we apply cooperative coevolution does not implement the previously described search space partitioning concept. In our case, both P and Q explore the same search space, i.e., D. Instead, our approach profits from two general (complementary) characteristics of the coevolutionary concept:

- it helps to maintain diversity in the evolutionary system;
- it enables the rapid dissemination of elite solutions;

According to the descriptions from the previous four subsections, at the end of every generation t, $t \geq 1$, the subpopulations P and Q that will be involved in the next evolutionary cycle (i.e., that of generation $t+1$) have been computed and archive A is in an up-to-date state. The very last step before starting the computation cycles of generation $t+1$ consists of a fitness sharing stage between the three main population subdivisions: P, Q, and A. The purpose of this stage is to make sure that the best global solutions found till now are given the chance to be present in both coevolved populations.

The first step is to generate C-an elite subset with the property: $|C| = B_{\text{size}}$, where B_{size} has been defined in the previous subsection. This elite subset is easily constructed by first performing the union $C = P \cup Q \cup A$ and then applying the environmental selection operator: $C = E_{\text{sel}}(C, B_{\text{size}})$. The second step of the fitness sharing stage is to try to introduce the individuals of this elite subset into the subpopulations P and Q of the next generation. This is also done through the usage of the environmental selection operator (as defined in Sect. 3.1):

- $P = P \cup C$ and $P = E_{\text{sel}}(P, P_{\text{size}})$;
- $Q = Q \cup C$ and $Q = E_{\text{sel}}(Q, Q_{\text{size}})$;

3.6 The main DECMO2 loop

The initialization stage (i.e., "generation 0") and the main computational loop of our hybrid MOEA are presented in Algorithm 1. There are three input parameters: *MOOP*—the definition of the problem to be solved, *archS*—the size of the archive A (i.e., $|A|$), and *maxT*—the maximum number of generations to be evolved. The algorithm returns a *Pareto non-dominated set* (PN) of size *archS*.

There are seven auxiliary methods that we use across Algorithm 1:

- ExtractSizes(*archS*)—this function computes P_{size}, Q_{size}, and B_{size} from the call argument $archS$ (which equals $|A|$), by solving (8);

Algorithm 1 Description of the DECMO2 hybrid multi-objective evolutionary algorithm

function DECMO2($MOOP$, $archS$, $maxT$)
  $P, Q \leftarrow \Phi$
  $\langle P_{size}, Q_{size}, B_{size} \rangle \leftarrow$ **ExtractSizes**($archS$)
  $A \leftarrow$ **InitializeArchive**($MOOP$, $archS$)
  $i \leftarrow 1$
  while $i \leq archS$ **do**
    $x \leftarrow$ **CreateIndividual**($MOOP$)
    InsertIntoArchive(A, x)
    if $i \leq P_{size}$ **then**
      $P \leftarrow P \cup \{x\}$
    else
      if $i \leq P_{size} + Q_{size}$ and $i > P_{size}$ **then**
        $Q \leftarrow Q \cup \{x\}$
      end if
    end if
    $i \leftarrow i + 1$
  end while
  $\phi^P, \phi^Q, \phi^A \leftarrow 1$
  $t \leftarrow 1$
  while $t \neq maxT$ **do**
    if $t \in \{2k+1 : k \in \mathbb{Z}\}$ **then**
      if $\phi^P > \phi^Q$ and $\phi^P > \phi^A$ **then**
        $P_{size} = |P| + B_{size}$
      end if
      if $\phi^Q > \phi^P$ and $\phi^Q > \phi^A$ **then**
        $Q_{size} = |Q| + B_{size}$
      end if
    end if
    $A_{size} \leftarrow archS - P_{size} - Q_{size}$
    $\langle P, \phi^P \rangle \leftarrow$ **EvolveNextGenSPEA2**(P, P_{size})
    $\langle Q, \phi^Q \rangle \leftarrow$ **EvolveNextGenDE**(Q, Q_{size})
    $\phi^A \leftarrow$ **EvolveArchiveInd**(A, A_{size})
    $P_{size} = |P|$
    $Q_{size} = |Q|$
    $C \leftarrow P \cup Q \cup A$
    $C \leftarrow E_{sel}(C, B_{size})$
    $P \leftarrow P \cup C$
    $P \leftarrow E_{sel}(P, P_{size})$
    $Q \leftarrow Q \cup C$
    $Q \leftarrow E_{sel}(Q, Q_{size})$
    $t \leftarrow t + 1$
  end while
  $C \leftarrow P \cup Q \cup A$
  $C \leftarrow E_{sel}(C, archS)$
  return C
end function

- InitializeArchive(*MOOP, archS*)—considering the notations from Sect. 3.3, this function first creates a total of archS uniformly spread weight vectors (i.e., $\lambda^1, \ldots, \lambda^{archS}$) with the dimensionality required by the given *MOOP*. It then proceeds to create and return an incomplete archive of the form: $A = \{\langle \lambda^1, \rangle, \ldots, \langle \lambda^{archS}, \rangle\}$;
- CreateIndividual(*MOOP*)—this function returns a randomly created individual that encodes a possible solution for the given *MOOP*;
- InsertIntoArchive(A, x)—this procedure looks if there are any incomplete pairs (i.e., of the form $\langle \lambda^i, \rangle$ with $i \in \{1, \ldots, |A|\}$) in archive A, and, if such a pair is found, it updates the archive: $A = A \setminus \langle \lambda^i, \rangle$ and $A = A \cup \langle \lambda^i, x \rangle$;
- EvolveNextGenSPEA2(P, P_{size})—this function uses the SPEA2 evolutionary model described in Sect. 3.1 to evolve the solution set passed as the first call argument (i.e., P) for one generation. It returns two entities: (1) a new, evolved, population of size $|P|$ and (2) the archive insertion ratio achieved by the P_{size} offspring that were created during the evolutionary process.
- EvolveNextGenDE(Q, Q_{size})—this function uses the DE-based evolutionary cycle described in Sect. 3.2 to evolve the set passed as the first call argument (i.e., Q) for one generation. It also returns two entities: (1) a new, evolved, population of size $|Q|$ and (2) the archive insertion ratio achieved by the Q_{size} offspring that were created during the evolutionary cycle.
- EvolveArchiveInd(A, A_{size})—this function uses the *DE/rand/1/bin strategy* (i.e., the combination of (4) and (5)) to create a number of A_{size} offspring using only individuals directly selected from the pairs that make up archive A. Each offspring individual created at this step is considered for the archive update procedure described in Sect. 3.3. The archive insertion ratio achieved when considering the A_{size} generated offspring is the only entity returned by this function. If $A_{\text{size}} = 0$, the function returns the value 0.

When considering the above description, one of the major shortcomings of our proposed multi-objective optimization method is evident: high structural and computational complexity. As we strived to create an efficient hybrid starting from three different evolutionary approaches for solving MOOPs, ending up with a fairly complex optimization procedure was something to be expected. However, it should be noted that, apart from the parameterizations required by the genetic operators we rely on (i.e., SBX, PM and DE), our approach remains quite robust, as it does not require any extra parameters. In Sect. 5, we present solid evidence that DECMO2 displays a very good average performance on a wide range of MOOPs and we think that this more than compensates for the complexity of our method.

4 Comparing the performance of MOOAs

As with most (meta)heuristic approaches, when talking about the performance of a multi-objective evolutionary algorithm, three criteria are primarily considered and usually need to be balanced:

- the quality of the generated solution, i.e., how well does the PN returned at the end of the optimization run approximate the PF of the MOOP to be solved?
- the convergence speed, i.e., what is the number of fitness evaluations (notation: nfe) that must be performed during the optimization run to reach a PN of acceptable quality?
- the generality of the algorithm, i.e., is the proposed method able to display the previous two criteria on a wide range of problems?

It should be noted that the above three criteria can be applied to evaluate any multi-objective optimization algorithm. For example, very fine-grained grid searches over the entire decision space will likely produce the best results with regard to the quality and generality criteria, but such approaches display excessively poor convergence speeds, which render them useless in most cases.

Over the years, several metrics for assessing the PN quality criterion have been proposed. A comprehensive analysis and review of most of these metrics can be found in Zitzler et al. (2003). Some of the more popular metrics are: the generational distance (GD) and the inverted generational distance (IGD) proposed in Van Veldhuizen and Lamont (1998) and the hypervolume metric ($\mathcal{H}$) proposed in Zitzler (1999). The latter has the added advantage that it is the only PN quality comparison metric for which we have theoretical proof of a monotonic behavior [see Fleischer (2003)]. As such, by design, the PF of any MOOP has the highest achievable $\mathcal{H}$ value. The monotonic property of $\mathcal{H}$ can be understood in the sense that, given two Pareto non-dominates sets, PN_A and PN_B, if $\mathcal{H}(\text{PN}_A) > \mathcal{H}(\text{PN}_B)$, we can be certain that PN_A "is not worse than" PN_B [see Zitzler et al. (2003) for details]. Furthermore, when comparing with GD and IGD, the hypervolume is easier to compute when the PF of the MOOP is unknown (as it is the case with most real-life problems).

Measuring the convergence speed is a truly trivial task once one has a clear idea of how to define acceptable quality in the case of Pareto non-dominated sets. Unfortunately this definition is highly domain-dependent and sometimes it also depends on the experience (or even subjective opinions) of the decision maker. For example, in many publication from the field of multi-objective optimization, acceptable quality means a (nearly) perfect approximation of the PF of a given benchmark MOOP. When considering real-life applications of multi-objective optimization, a PN may be deemed of having an acceptable quality if it "is not worse than" any other

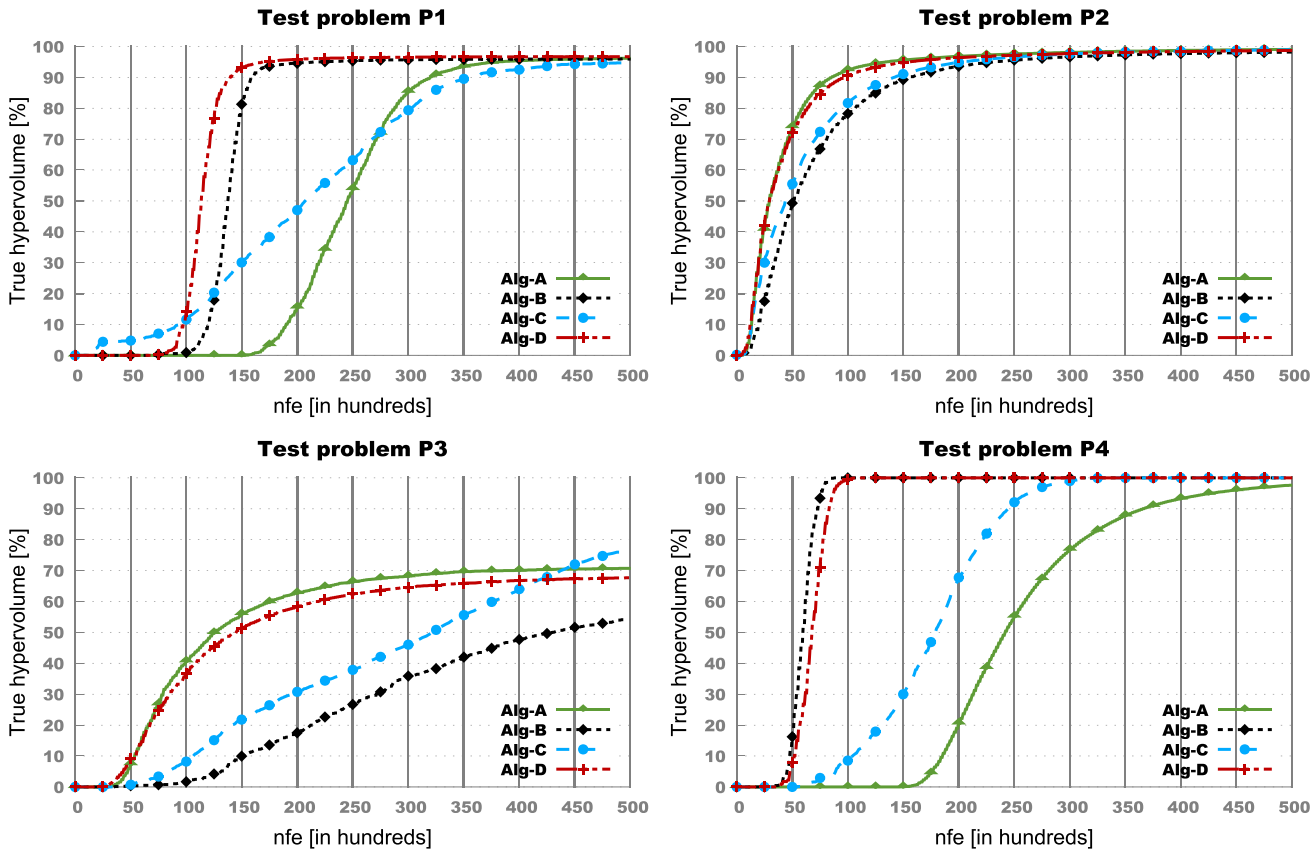

Fig. 1 Run-time $\mathcal{H}$-measured performance on the four problems considered in our toy test set

PN ever discovered for the considered MOOP (even though it is actually a rather poor approximation of the PF).

In light of the very computationally intensive nature of the fitness functions required by the industrial MOOPs we aim to solve, our idea of how to best balance quality, convergence speed and generality in order to assess the performance of a MOOA is that: given an arbitrary MOOP and a maximal number of fitness evaluations ($nfe_{\max}$) that we are willing to execute, the analyzed MOEA displays the best possible performance if, for any $nfe \leq nfe_{\max}$, the PN obtained after performing nfe fitness evaluations "is not worse than" the PN that might have been obtained by any another available method after also performing nfe fitness evaluations.

Although quite vague at a first glance, the previous statement is the base from which we developed a practical ranking framework for multi-objective optimization algorithms. This framework is described in the next subsection and it can offer practitioners valuable insight regarding the relative performance of different:

- multi-objective optimization algorithms;
- parameterization settings for a given MOOA;

4.1 A racing-based ranking of performance in the context of MOOPs

Let us consider a toy example in which we wish to compare the performance of four different multi-objective optimization algorithms (Alg-A, Alg-B, Alg-C, and Alg-D) on a limited test set that consists of four benchmark MOOPs (P1, P2, P3, and P4) with known PFs. For each optimization run we perform 50,000 fitness evaluations. A more or less standard approach would be to perform several independent runs for each MOOA–MOOP pair and assess the quality and convergence behavior by computing some metric over averaged results. For example, the plots from Fig. 1 display the average run-time $\mathcal{H}$-measured performance of the four algorithms when considering 25 independent runs for each test. For every run, the data points were obtained by calculating the $\mathcal{H}$ of the current MOOA population after every 100 fitness evaluations and afterwards computing the percentage ratio (notation: $\mathcal{H}$-%-ratio) obtained when comparing these values against $\mathcal{H}$(PF), where PF denotes the Pareto front of the MOOP that is solved. As a side note, we shall write "fully" (with quotes) when referring to a MOOA that is able to solve a MOOP (i.e., $\mathcal{H}$-%-ratio $\approx$ 100 %) to emphasize the fact

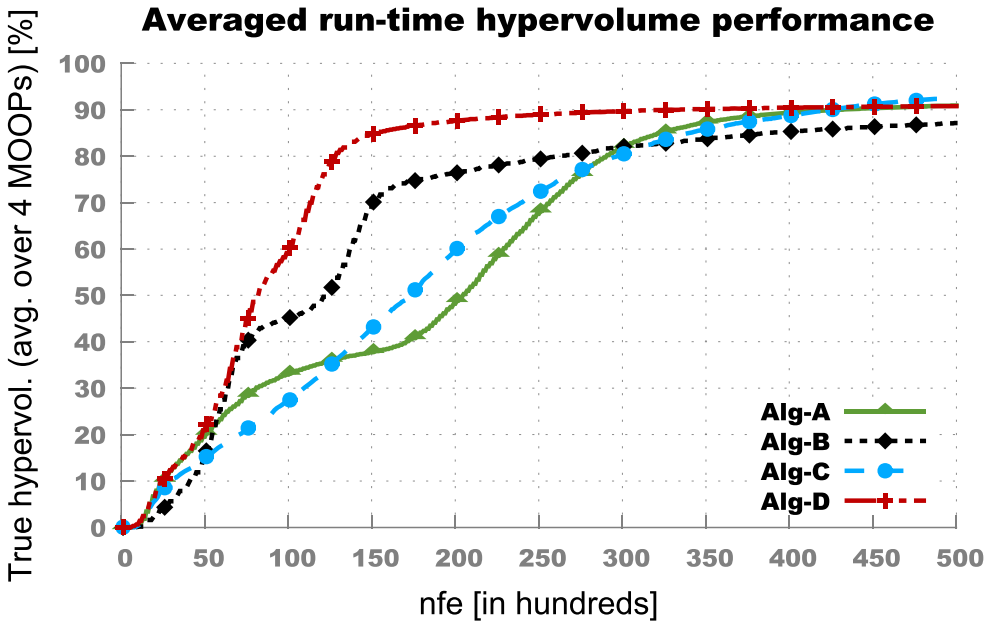

Fig. 2 Averaged run-time $\mathcal{H}$-measured performance over the entire toy test set

that, in most cases (e.g., all continuous MOOPs), a PN cannot be (by definition) more than a near perfect approximation of the PF.

To quickly assess the general performance of the four tested MOOAs (w.r.t. the example test set), it is very intuitive to plot the $\mathcal{H}$-%-ratio-measured performance, averaged over the entire problem set (e.g., Fig. 2). Such a chart is very useful as it clearly shows:

- which algorithm generally starts to converge faster (e.g., Alg-D in the case of our example);
- which algorithm has the best average performance at the end of the runs (e.g., Alg-C);
- which algorithms seem to have a somewhat similar convergence behavior during (a certain part of) the optimization run. For example, Alg-D and Alg-B converge quite fast (average $\mathcal{H}$-%-ratio $\geq$ 70 % after 15,000 fitness evaluations) while Alg-A and Alg-C converge slower, but reach slightly better average results at the end of the experiment (after 50,000 fitness evaluations). Knowing this and assuming that the used MOOP test set is relevant for real-life situations, in practice, we would prefer Alg-D or even Alg-B over Alg-C/Alg-A when faced with a limited number of fitness evaluations and we would use Alg-C or Alg-A if there would be no such limitation.

Nevertheless, the average $\mathcal{H}$ plot from Fig. 2 is also misleading because, through averaging, it helps to mask very bad performance. As all four algorithms display average $\mathcal{H}$-%-ratio values between 85 and 95 % after 40000 fitness evaluations, we might believe that their general performance is largely similar when it comes to the solutions discovered towards the end of each run. In fact, the very good performance of Alg-B on P1, P2 and P4 helps to cover up the very poor behavior on problem P3. Similarly, the fact that all four algorithms are (sooner or later) each able to fully converge on one MOOP is also concealed. Although in our very simple example, both problems can be solved by independently consulting the relative performance of the four MOOAs on each MOOP via numerical/visual inspection of $\mathcal{H}$-related performance, in rigorous performance comparison contexts, involving tens of MOOPs and several MOOAs, such a case-by-case approach is very tedious, and, in the late stages of convergence (where most good algorithms find PNs of roughly similar quality), it can also become useless.

Our idea for simplifying the comparison process is to interpret the run-time hypervolume plot for each MOOP as if it depicts the results of a multi-stage race between the four MOOAs. The goal is to reach a $\mathcal{H}$-%-ratio $\approx$ 100 as fast as possible (i.e., "fully" converge after the lowest possible nfe). The secondary goals are to have the highest $\mathcal{H}$-%-ratio at the end of each stage in the race. Therefore, it makes sense to imagine a basic ranking schema where, at the end of each stage, the analyzed MOOAs are ranked in ascending order of their $\mathcal{H}$-%-ratio starting with the worst performer. In our toy example, 4 is assigned for the smallest $\mathcal{H}$-%-ratio value and 1 for the highest. There are two exceptions from this rule:

- if the $\mathcal{H}$-%-ratio at a certain stage is higher than 99 % (i.e., the obtained PN dominates more than 99 % of the objective space that is dominated by the PF), the analyzed

Table 1 Ranks corresponding to the run-time $\mathcal{H}$ plots presented in Fig. 1

Problem	Rank computation stages based on $\mathcal{H}$-%-ratios											
	0	1	2	3	4	5	6	7	8	9	10	μ_P
	Ranks achieved by Alg-A											
P1	5	5	5	5	4	4	3	3	3	2	2	3.73
P2	5	1	1	1	1	1	1	1	1	1	0	1.27
P3	5	2	1	1	1	1	1	1	1	2	2	1.64
P4	5	5	5	5	4	4	4	4	4	4	4	4.36
μ_S	*5.00*	*3.25*	*3.00*	*3.00*	*2.50*	*2.50*	*2.25*	*2.25*	*2.25*	*2.25*	*2.00*	
	$\mu_A = 2.75, \mu_F = 2.00$											
	Ranks achieved by Alg-B											
P1	5	5	5	2	2	2	2	2	2	3	3	3.00
P2	5	4	4	4	4	4	4	4	4	4	4	4.09
P3	5	5	4	4	4	4	4	4	4	4	4	4.18
P4	5	1	0	0	0	0	0	0	0	0	0	0.55
μ_S	*5.00*	*3.75*	*3.25*	*2.50*	*2.50*	*2.50*	*2.50*	*2.50*	*2.50*	*2.75*	*2.75*	
	$\mu_A = 2.95, \mu_F = 2.75$											
	Ranks achieved by Alg-C											
P1	5	1	1	3	3	3	4	4	4	4	4	3.27
P2	5	5	3	3	3	3	3	2	2	2	2	2.82
P3	5	5	3	3	3	3	3	3	3	1	1	3.00
P4	5	5	3	3	3	3	3	0	0	0	0	2.27
μ_S	*5.00*	*3.50*	*2.50*	*3.00*	*3.00*	*3.00*	*3.25*	*2.25*	*2.25*	*1.75*	*1.75*	
	$\mu_A = 2.84, \mu_F = 1.75$											
	Ranks achieved by Alg-D											
P1	5	5	2	1	1	1	1	1	1	1	1	1.82
P2	5	2	2	2	2	2	2	3	3	3	3	2.64
P3	5	1	2	2	2	2	2	2	2	3	3	2.36
P4	5	2	0	0	0	0	0	0	0	0	0	0.64
μ_S	*5.00*	*2.5*	*1.5*	*1.25*	*1.25*	*1.25*	*1.25*	*1.5*	*1.5*	*1.75*	*1.75*	
	$\mu_A = 1.86, \mu_F = 1.75$											

For each algorithm, the values in italics are used to create the left-side plot from Fig. 3

algorithm is assigned the rank 0. This is how we mark (reward) "full" convergence.
- if the $\mathcal{H}$-%-ratio at a certain stage is lower than 1 %, the analyzed algorithm is assigned a rank which equals one plus the total number of analyzed MOOAs (i.e., five in our case). This is how we mark (penalize) a MOOA that has not yet produced a relevant PN, i.e., a MOOA that has not started to converge.

In the toy example, the comparison stages are equidistant (i.e., they are set after every 5000 fitness evaluations) and, in Fig. 1, each stage of the race is marked with a vertical solid grey line. The rank information we obtained is presented in Table 1. For each MOOA, the table also presents four average ranks:

- μ_P—the average rank achieved by the MOOA on an individual problem. A value closer to 0 indicates that the algorithm displays a good performance on the problem. μ_P can be used to rapidly/automatically identify those problems on which a MOOA performs very well (i.e., "fully" converges very fast) or poorly (i.e., does not "fully" converge, converges very slowly, etc.).
- μ_S—is the average rank across the entire test set at a given stage (i.e., after a fixed number of nfes). These are useful as we shall combine them to display the dynamics of the relative MOOA performance over time.
- μ_F—is the average rank across the entire test in the final stage (i.e., close to the end of the optimization). The MOOA that has the smallest value of μ_F was able to "fully" converge or discover higher quality PNs on more problems than its competitors.

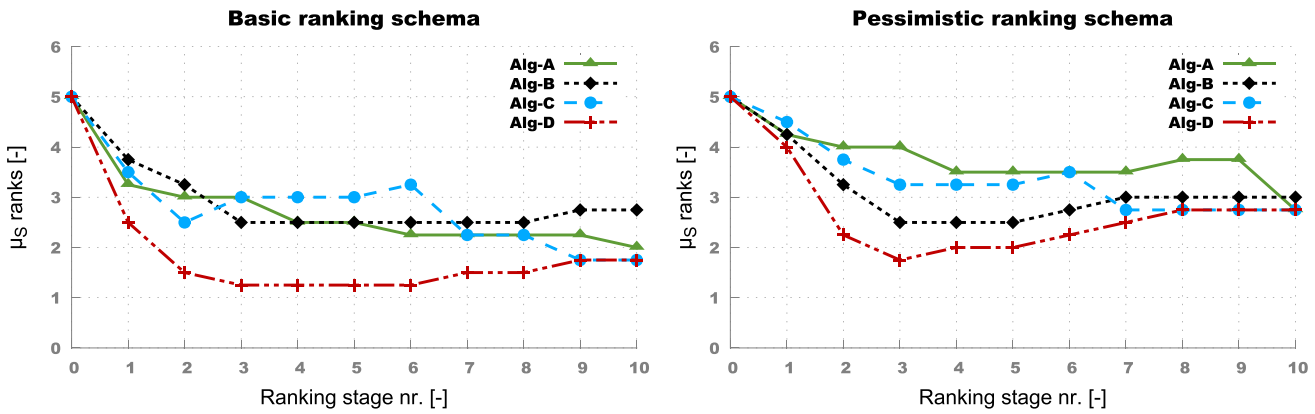

Fig. 3 HRPCs obtained when applying the racing-based ranking methodology on the toy test set

– μ_A—is the overall average rank achieved by the MOOA during the comparison. The value of μ_A can be used to single out the MOOAs that tend to generally outperform their counterparts.

In the left-side plot from Fig. 3 we use the μ_S values to plot hypervolume-ranked performance curves (HRPCs). We feel that by introducing HRPCs, we are providing practitioners in the field of multi-objective optimization with an extremely useful tool for helping to rapidly assess the general comparative performance of MOEAs (especially over test sets containing many MOOPs). The basic ranking schema ignores the magnitudes of the differences in performance and favors the algorithm that is able to perform very well on the highest number of MOOPs from the considered test set. When considering the HRPCs computed for the toy comparison context (left-side plot from Fig. 3), the data points corresponding to the last 2 ranking stages indicate that:

– there is a good balance between the number of MOOPs on which Alg-D and Alg-C perform well by the end of the optimization runs;
– Alg-A has managed to converge on at least one MOOP right before the final stage as passing from a rank of 1 to a rank of 0 is the only explanation for a drop of average rank between the two stages that does not influence the average ranks of the other three MOOAs.

The main advantage of HRPCs is that they can be easily adjusted in order to outline certain MOOA performance characteristics by making small changes in the required ranking procedure. For example, using the same run-time information that was plotted in Fig. 1, we could focus our MOOA comparison on analyzing if there are large differences in performance between the tested algorithms by imposing that: at a given stage, the difference between two $\mathcal{H}$-%-ratios must be higher than 10 % in order to justify a rank improvement, i.e., we impose a $\mathcal{H}$-ranking threshold of 10 %. According to the this modification, if at a certain stage the four MOOAs have the $\mathcal{H}$-%-ratios (64, 78, 84, 99.5 %), they will be assigned the ranks (4, 3, 3, 0). The HRPCs obtained when applying this, very pessimistic, ranking schema are presented in the right-side plot from Fig. 3 and:

– the data points corresponding to the last 3 ranking stages confirm that Alg-D and Alg-C have an average similar convergence behavior towards the end of the runs;
– the data points corresponding to the last ranking stage indicate that Alg-A seems to perform much worse (i.e., difference in $\mathcal{H}$-%-ratio > 10 %) than the other 3 MOOAs on at least one extra MOOP;

We have devised this racing and hypervolume-based ranking methodology that may combine information from:

– several HRPC plots (computed using different ranking schemata);
– associated μ_P, μ_F, μ_A values;
– the plot of the averaged $\mathcal{H}$-measured performance over the entire problem test

to easily observe/report the general performance characteristics of the MOOAs we wish to analyze over large problem sets.

4.2 On robustness and efficiency in MOEAs

As mentioned in the introductory part, our main interests with regards to multi-objective optimization algorithms are related to enhancing these methods to improve the run-times of industrial optimizations that rely on very computationally intensive fitness evaluation functions. In the particular case of MOOAs that can be parameterized (e.g., most MOEAs), the prohibitive optimization run-times that occur when solv-

ing industrial MOOPs usually make systematic parameter tuning approaches virtually impossible. This means that we strongly prefer to rely on MOEAs that are very robust with regard to their control parameters, meaning that they generally perform well on a wide range of problems when using the parameterization settings recommended in the literature.

The second important characteristic that we demand from a MOEA is efficiency. In non-academic terms, the idiom "bang for the buck" encapsulates very well the essence of this characteristic and, in light of the concepts presented till this point, we consider that "$\mathcal{H}$ for nfe" is a good equivalent, especially when dealing with very lengthy run-times induced by computationally intensive fitness evaluation functions. The only condition is that, in the case of MOEAs, efficiency must be stable (i.e., displayed throughout the duration of the optimization run), general (i.e., displayed on a wide range of MOOPs), and, because of the stochastic nature of evolutionary methods, must be supported by averaged results over many independent runs.

The new MOOA racing-based ranking comparison framework, which we introduced in the previous subsection, is able to offer insights with regard to both the efficiency and robustness of a given MOEA provided that we construct comparison contexts where:

- we compare the given MOEA against MOOAs that are themselves regarded as being generally successful (i.e., they are state of the art);
- we maintain a fixed parameterization of the tested algorithms;
- the test sets contain a sufficient number of MOOPs with different characteristics;
- we apply appropriate ranking schemata;

In the next section, we obey these rules to construct comparison contexts that help to tune MOEA/D-DE and to evaluate the robustness and efficiency of DECMO2.

5 Tests regarding the performance of DECMO2

To evaluate the performance of DECMO2, we consider two types of comparisons:

- the first one aims to estimate the robustness and efficiency of our hybrid and adaptive MOEA by applying the new comparison methodology we proposed in Sect. 4 on a test set consisting of 20 artificial benchmark problems;
- the second comparison is a case study regarding the convergence behavior of DECMO2 and SPEA2 on two industrial MOOPs from the field of electrical drive design optimization. Both problems require very computationally intensive fitness evaluation functions.

The 20 artificial benchmark problems we aggregated in our test set are:

- DTLZ1, DTLZ2, DTLZ4, DTLZ6, and DTLZ7 from the problem set proposed in Deb et al. (2002b);
- KSW10—a classic optimization problem with 10 variables and two objectives based on Kursawe's function described in Kursawe (1991);
- all nine problems from the LZ09 problem set described in Li and Zhang (2009);
- WFG1, WFG4 and WFG8 from the problem set proposed in Huband et al. (2003);
- ZDT3 and ZDT6 from the problem set described in Zitzler et al. (2000);

When applying the race-based ranking methodology, we defined ranking stages after every 1000 fitness evaluations with the first ranking evaluation taking place at "generation 0" (i.e., we evaluated the randomly generated initial population of the MOEAs). We performed 50 independent runs for each MOEA–MOOP pair to obtain the hypervolume information based on which the rankings were computed. We applied two types of ranking schemata:

- the *basic* ranking schema which is identical to the one described in Sect. 4.1;
- the *pess-Thr* ranking schema which has the same working principles as the *pessimistic ranking schema* presented in Sect. 4.1. *Thr* is the $\mathcal{H}$-ranking threshold. For example, a pess-5 ranking schema uses a $\mathcal{H}$-ranking threshold of 5 %.

The algorithms we compared DECMO2 against (using the race-based ranking methodology) are SPEA2, GDE3, MOEA/D-DE [the Zhang et al. (2009) version], and DECMO. In the case of the first three algorithms we relied on implementations available in the jMetal package [see Durillo and Nebro (2011)]. We fixed the number of fitness evaluations to 50,000. Across all runs we used MOEA parameterizations that are in accordance with those recommended in the literature. For SPEA2, we used a population and archive size of 200, 0.9 for the crossover probability and 20 for the crossover distribution index of SBX, $1/n$ for the mutation probability (where n is the number of variables of the MOOP to be solved) 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 strategy. For MOEA/D-DE we used a population size of 500 and all other parameters were set as described in Zhang et al. (2009). For DECMO we used a size of 100 for each coevolved subpopulation, the same SBX and PM parameterizations used for SPEA2 and the settings CR = 0.2 and $F = 0.5$ for the DE/rand/1/bin strategy. In the case of DECMO2 we used an archive size of 200 and

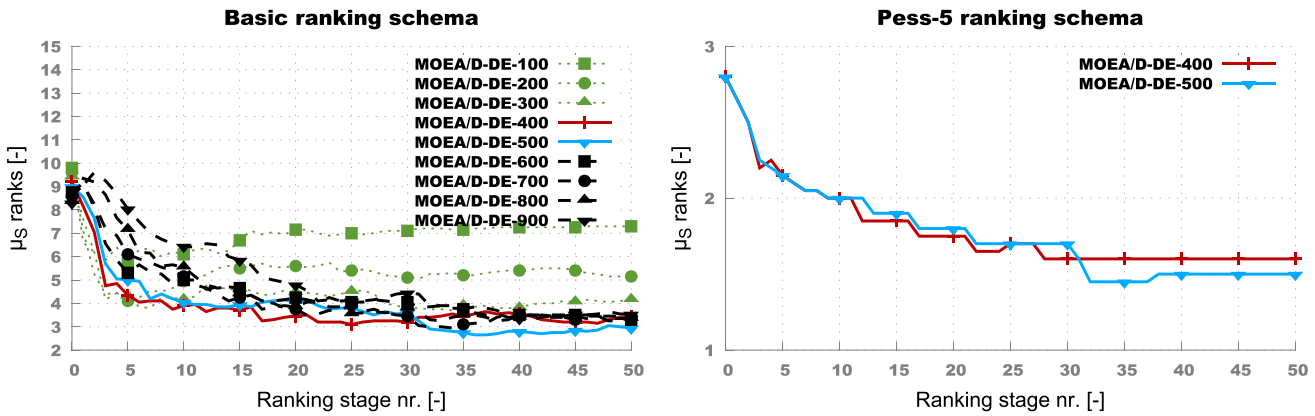

Fig. 4 HRPCs obtained when testing the impact of various archive sizes on MOEA/D-DE

– in the case of subpopulation P: 1.0 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;
– in the case of subpopulation Q: the settings CR = 0.2 and $F = 0.5$ for the DE/rand/1/bin strategy;
– in case of (the bonus) individuals evolved directly from A: the settings CR = 1.0 and $F = 0.5$ for the DE/rand/1/bin strategy;

In the case of DECMO and DECMO2, the control parameters for the DE/rand/1/bin strategy used in subpopulation Q are chosen such as to maintain a good trade between exploration and intensification ($F = 0.5$) and, as shown in Zaharie (2009), stimulate a minor increase in population diversity (CR = 0.2). When evolving bonus individuals from A, CR is set to 1.0 to stimulate population diversity to the maximum inside the highly elitist archive.

While it is fair that all MOEAs except MOEAD/D-DE should use the same population size since they are all constructed around the Pareto-based elitism paradigm, in the case of MOEAD/D-DE, the size of the archive was set at 500 after using the race-based ranking methodology to estimate the relative performance achieved by nine different archive sizes (from 100 to 900). We applied the basic ranking schema and obtained the HRPCs that are presented in the left-side plot from Fig. 4. These HRPCs indicate that, on average, over the 20 considered benchmark MOOPs:

– when using an archive size of 500, MOEAD/D-DE is able to achieve the best results towards the end of the optimization run (i.e., between the ranking stages 31 and 50);
– when using an archive size of 400, MOEAD/D-DE is able to achieve the best results during the middle of the run (i.e., between the ranking stages 8 and 30);

Having two strong candidates, we applied again the race-based ranking methodology (this time using a pess-5 ranking schema on only MOEAD/D-DE-400 and MOEAD/D-DE-500. The obtained HRPCs are presented in the right-side plot from Fig. 4 and they indicate that, on average, differences between the two methods are greater during the end of the run than during the middle part of the run. As such, we decided that, when keeping every other parameter fixed, an archive size of 500 would enable MOEA/D-DE to achieve the best overall performance on our benchmark problem set.

5.1 Results on artificial benchmark problems

Because in several future statements we shall use the phrase "on average" to refer to conclusions drawn from various results we present, it is important to clearly state what we mean by this. Considering that we have experimented with 5 different MOEAs over 20 different MOOPs and that we made 50 independent runs for each MOEA–MOOP combination, at every stage of our race-based ranking procedure we have assigned ranks based on 2,000 (when comparing only two MOEAs) to 5,000 (when comparing all five) hypervolume measurements. Since the HRPCs are based on 51 ranking stages, each of them aggregates information from 102,000 (plots with two HRPCs) to 255,000 independent hypervolume measurements. To construct the plot of the averaged $\mathcal{H}$-measured performance of all five algorithms over the entire benchmark problem set (i.e., the plot from Fig. 5), we sampled $\mathcal{H}$ values after 1,000 fitness evaluations on each independent run. Therefore, this plot is based on 2,500,000 independent data points.

Figure 6 contains four subplots with the HRPCs obtained by the five algorithms we tested with over the entire artificial problem set. In addition, Table 2 presents the μ_F and μ_A values achieved by each tested MOEA when applying the pess-1 ranking schema. With regard to "full" convergence (i.e., reaching $\mathcal{H}$-%-ratio $> 99\,\%$), SPEA2 was able to achieve it on 3 problems, GDE3 on 4 problems, MOEA/D-DE and DECMO on 5 problems, and DECMO2 on 7 prob-

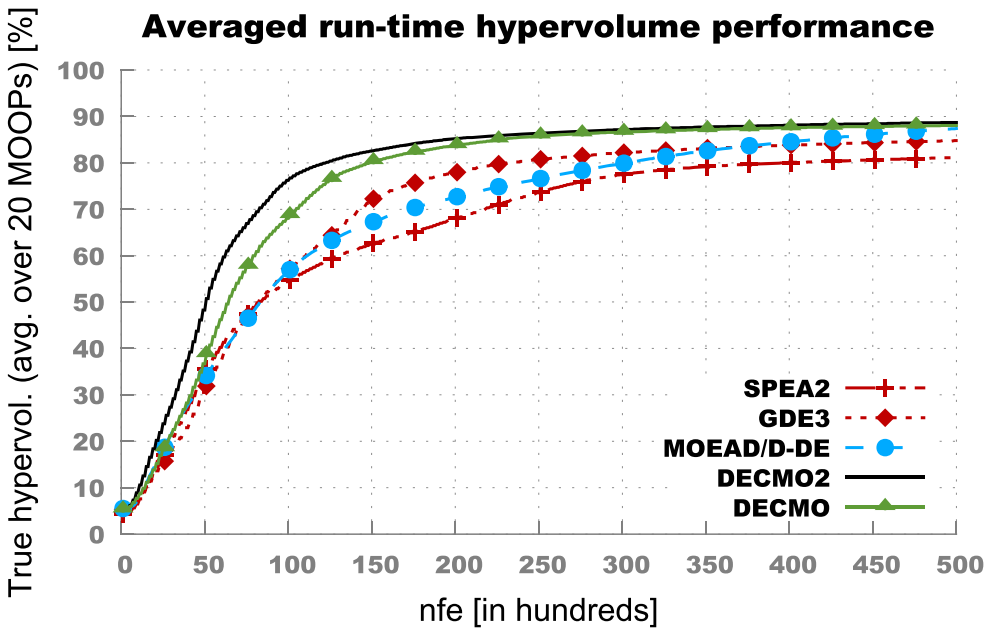

Fig. 5 Averaged run-time $\mathcal{H}$-measured performance over 20 artificial benchmark MOOPs

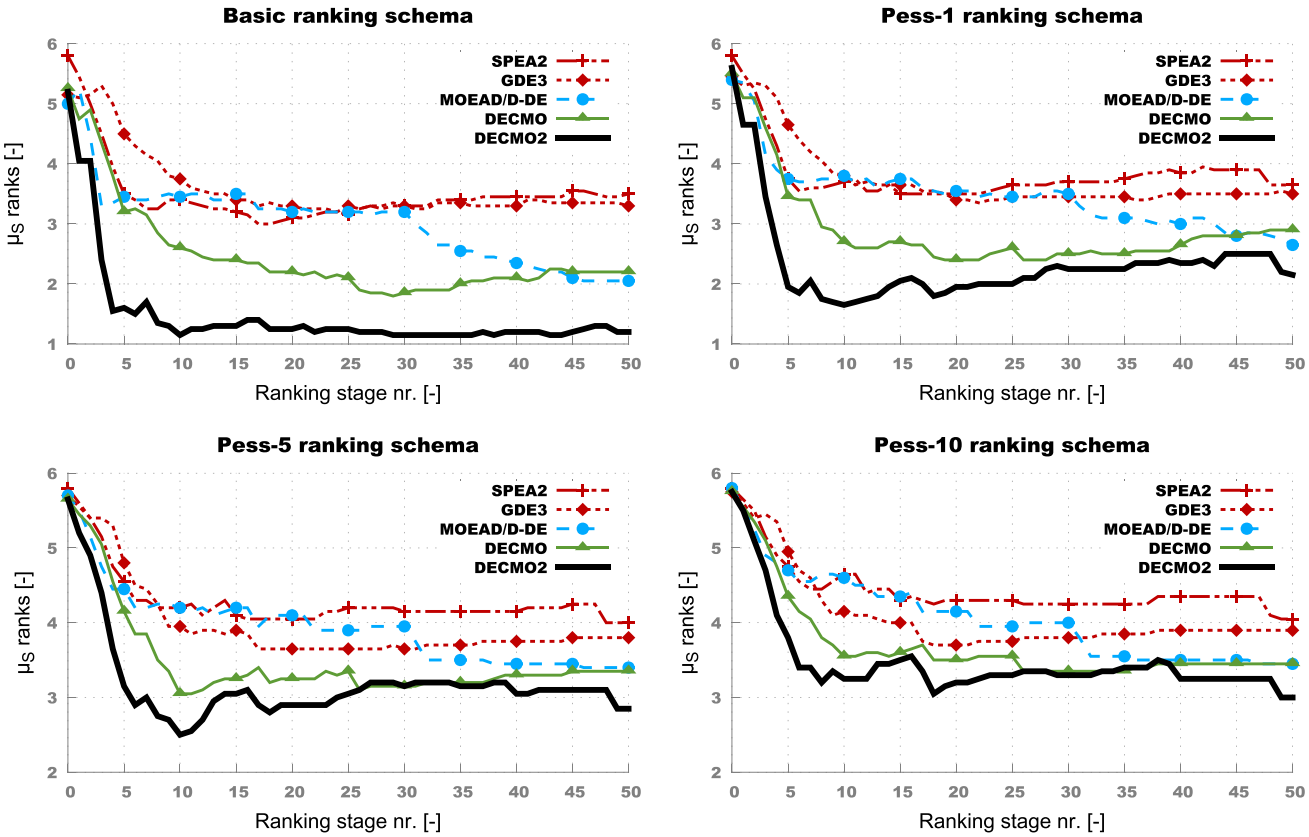

Fig. 6 HRPCs obtained when comparing DECMO2 with four other MOEAs over 20 artificial benchmark MOOPs

Table 2 The average ranks achieved by the five tested MOEAs over the benchmark problem set when applying a pess-1 ranking schema

Algorithm	μ_F	μ_A
SPEA2	3.6500	3.8265
GDE3	3.5000	3.7490
MOEA/D-DE	2.6500	3.4775
DECMO	2.9000	2.8902
DECMO2	*2.1500*	*2.3294*

The best values are italicized

lems. LZ09-F1 is the only MOOP on which DECMO2 was unable to achieve "full" convergence, but another algorithm, namely MOEA/D-DE, managed to do so.

Taking into account the setup of our tests and the arguments from Sect. 4.2, all the previously mentioned results allow us to conclude that DECMO2 is an efficient and robust MOEA.

Although all the HRPCs from Fig. 6 and all the hypervolume average values plotted in Fig. 5 show that, on average (and at every stage of the run), our method produces better hypervolumes than the other MOEAs we have compared against, it is extremely important to interpret this information in combination with the implications of the monotonicity of the $\mathcal{H}$ metric mentioned in the introduction of Sect. 4. As such, the strongest statement that we can make based on the obtained results is that: on average, DECMO2 is not worst than any of the other four MOEAs during any stage of the optimization run. But, based on the presented results, the same statement cannot be made for any of the other four algorithms. In light of this, (for the considered comparison context/test settings) we can we can weakly argue that, on average, DECMO2 is the best choice among the five tested MOEAs.

In accordance with the previous line of arguments, and taking into account the HRPCs obtained with the pess-5 and pess-10 ranking schemata, we can also conclude that, on average, especially in the initial phases of the optimization runs, DECMO2 displays a convergence speed that is not outperformed by any of the other MOEAs. We believe that this feature makes our hybrid algorithm a very strong candidate for MOOPs where the solver is limited in the number of fitness evaluations that it can perform per optimization run.

In Fig. 7, we plot the HRPCs obtained when only comparing DECMO2 to MOEA/D-DE. They indicate clearly that, on average, our hybrid and adaptive MOEA displays a better convergence behavior during the early part of the run and that MOEA/D-DE is, more or less, able to generally match the performance of DECMO2 towards the end of the run.

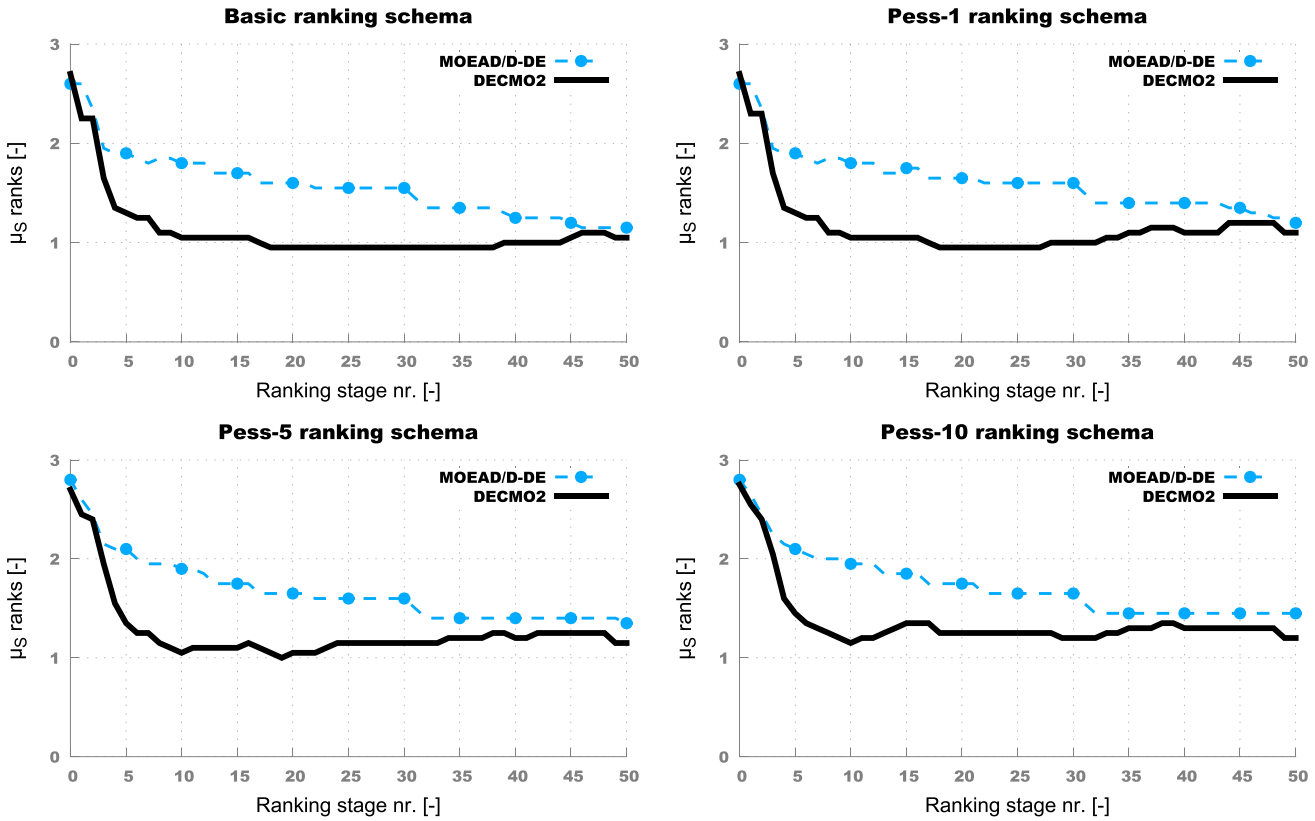

Fig. 7 HRPCs obtained when comparing DECMO2 with MOEA/D-DE over 20 artificial benchmark MOOPs

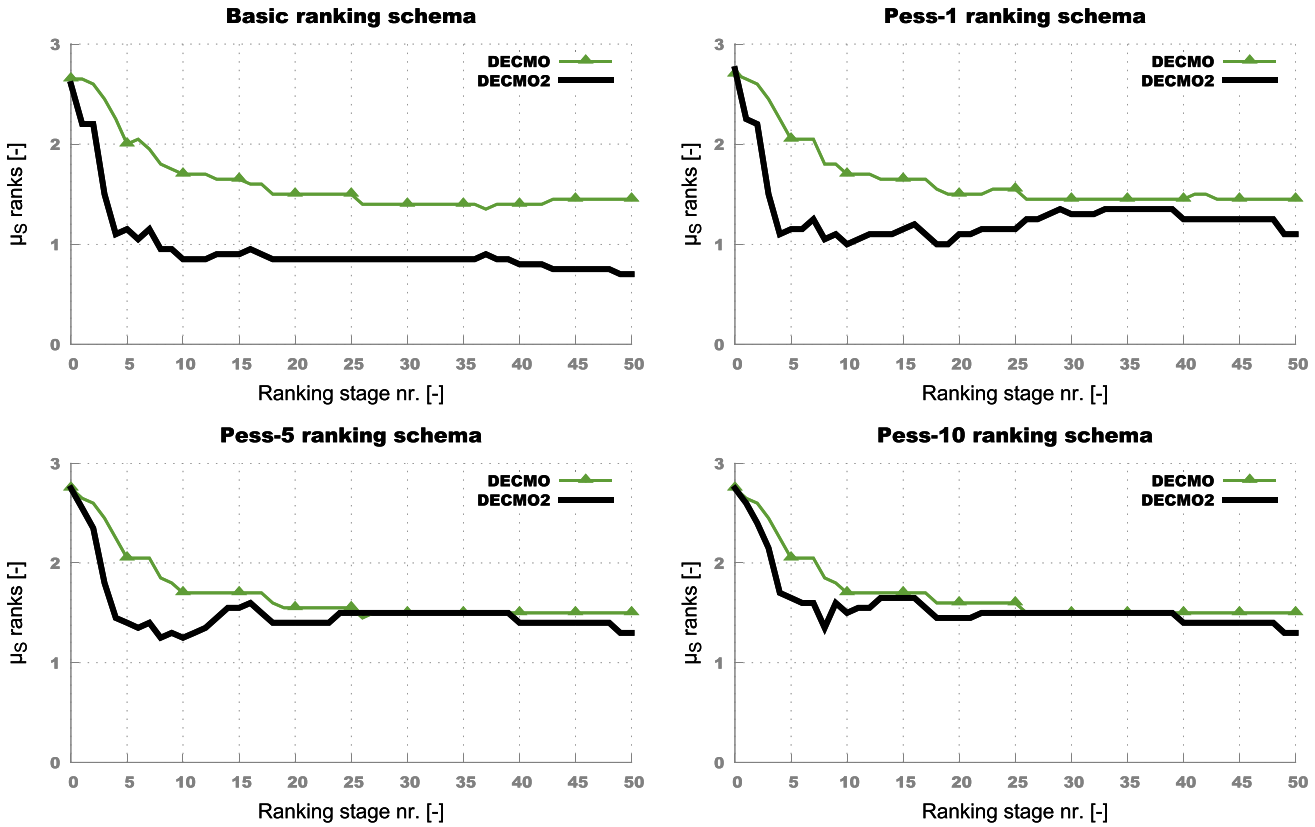

Fig. 8 HRPCs obtained when comparing DECMO2 with DECMO over 20 artificial benchmark MOOPs

In Fig. 8, we plot the HRPCs obtained when only comparing DECMO2 to DECMO. The HRPCs corresponding to the basic and pess-1 ranking schemata indicate that, in comparison with its predecessor, on average, DECMO2 displays at least some small improvements throughout the entire optimization run. When applying the pess-5 and pess-10 ranking schemata, DECMO2 only shows an improved average performance during the early part of the run and a slightly better average performance towards the end of the run. Nevertheless, the general idea is that by adding a decomposition-based strategy and an adaptive allocation of fitness evaluations, we were able to increase the overall performance of our initial coevolutionary method and enable it to successfully compete with a very well known and successful multi-objective optimizer like MOEA/D-DE over a wide range of artificial MOOPs.

5.2 Case study: electrical drive design

Although extremely valuable for the algorithm design, prototyping and parameter tuning stages, as we are primarily motivated by practical applications of multi-objective optimization, we generally regard the assessment of MOOA performance on artificial MOOPs as a mere means to an end. The final objective is to obtain a robust MOOA that is able to successfully tackle real-life MOOPs.

Using the same parameterizations we experimented with on the artificial problem set, we applied DECMO2 on two fairly complicated MOOPs from the field of electrical drive design, allowing for 10,000 fitness evaluations per run. In both problems, the goal is to configure 22 real-valued parameters to simultaneously optimize 4 objectives regarding cost and efficiency. For each problem, to evaluate the quality of a single design, we must perform a series of computationally intensive operations consisting of a meshing stage and one or more finite element simulations. The overall impact of these required simulations is that, even when distributing the fitness evaluations over a high throughput cluster computing environment, performing 10,000 fitness evaluations takes between 6 and 7 days to complete.

Because of the extremely long run-times, we only performed two independent runs with DECMO2 for each problem and saved information regarding the best found solutions after every 100 fitness evaluations. For both industrial MOOPs we also have (historical) run-time quality information from optimizations conducted with SPEA2 (two independent runs for each MOOP). Using as reference the best known sets of solutions for both problems, in Fig. 9, we present the run-time $\mathcal{H}$-measured performance of DECMO2

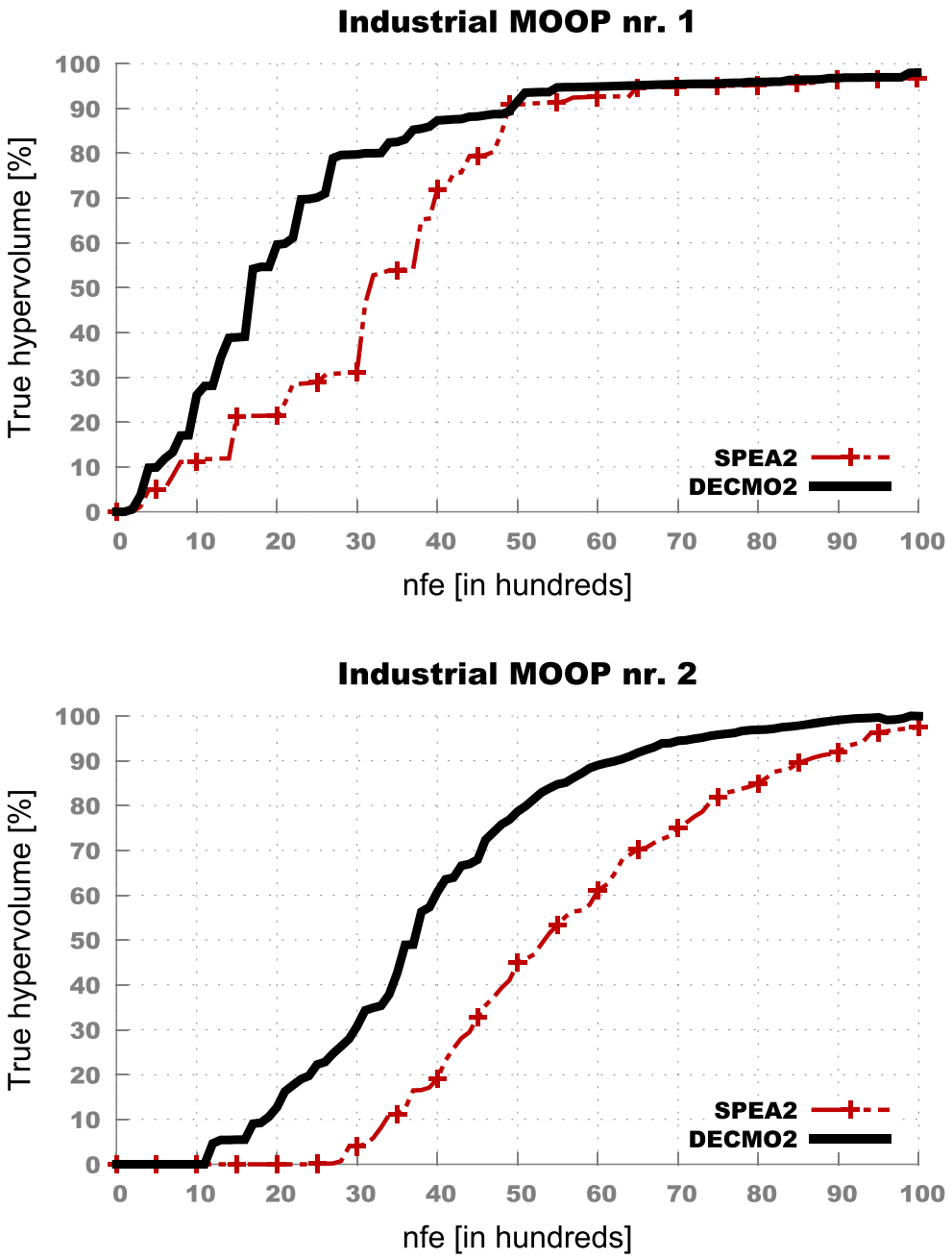

Fig. 9 Run-time $\mathcal{H}$-measured performance of DECMO2 and SPEA2 on two industrial MOOPs

and SPEA2 on the two industrial problems. The results indicate that DECMO2 is able to converge faster. In this particular case, the faster convergence of DECMO2 roughly translates into finding PNs that have the same $\mathcal{H}$ values as those that were discovered one day later when using SPEA2.

6 Conclusion

In this paper, we have described DECMO2, a hybrid multi-objective optimization algorithm that uses coevolution to successfully combine three different principles for solving MOOPs: Pareto-based dominance, differential evolution and decomposition-based strategies. DECMO2 also incorporates an adaptive allocation of fitness evaluations to accelerate convergence by rewarding the incorporated evolutionary model that is able to deliver the best performance during a given part of the optimization run.

A considerable part of the present paper (Sect. 4.1) is dedicated to introducing a new methodology aimed at providing practitioners from the field of multi-objective optimization with a simple means of analyzing/reporting the general com-

parative run-time performance of MOOAs over large problem sets. This methodology is largely based on a racing perspective over averaged hypervolume measures and can be used either to fine tune algorithms over given problem sets or to analyze the relative robustness and efficiency of MOOAs (see the discussion from Sect. 4.2).

In Sect. 5, we present results using the newly introduced MOOA comparison methodology that substantiates the claim that DECMO2 displays both robustness and efficiency when comparing against four other MOEAs (SPEA2, GDE3, MOEA/D-DE and DECMO) over a challenging benchmark of 20 artificial MOOPs from different well known problem sets. The results section also contains a small case study regarding the comparative performance of DECMO2 and SPEA2 on two real-life industrial MOOPs that feature computationally intensive fitness evaluation functions. The results of this study confirm the general characteristic of DECMO2 to converge fast.

In light of all the presented results, we finally argue that DECMO2 is a valuable addition to the ever-growing set of MOEAs and that, despite its structural complexity, this hybrid evolutionary algorithm is very robust with regard to its parameterization and, therefore, especially suited for solving real-life MOOPs that have computationally intensive fitness evaluation functions.

With regard to DECMO2, future work will revolve around developing a steady-state asynchronous version of the algorithm and around testing and analyzing the comparative performance on more industrial MOOPs. We also plan to extend our racing-based MOOA comparison methodology by designing a ranking schema that uses statistical significance testing.

Acknowledgments This work was conducted within LCM GmbH 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 partners. The authors thank all involved partners for their support.

References

Coello C, Lamont G, Van Veldhuisen D (2007) Evolutionary algorithms for solving multi-objective problems. Genetic and evolutionary computation series. Springer, New York

Deb K, Agrawal RB (1995) Simulated binary crossover for continuous search space. Complex Syst 9:115–148

Deb K, Goyal M (1996) A combined genetic adaptive search (GeneAS) for engineering design. Comput Sci Inf 26:30–45

Deb K, Pratap A, Agarwal S, Meyarivan T (2002a) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197

Deb K, Thiele L, Laumanns M, Zitzler E (2002b) Scalable multi-objective optimization test problems. In: IEEE congress on evolutionary computation (CEC 2002), IEEE Press, pp 825–830

Durillo JJ, Nebro AJ (2011) JMETAL: a Java framework for multi-objective optimization. Adv Eng Softw 42:760–771

Fleischer M (2003) The measure of Pareto optima. Applications to multi-objective metaheuristics. In: International conference on evolutionary multi-criterion optimization (EMO 2003), Springer, pp 519–533

Huband S, Barone L, While L, Hingston P (2005) A scalable multi-objective test problem toolkit. In: Evolutionary multi-criterion optimization (EMO 2005), lecture notes in computer science, vol 3410

Jannot X, Vannier J, Marchand C, Gabsi M, Saint-Michel J, Sadarnac D (2011) Multiphysic modeling of a high-speed interior permanent-magnet synchronous machine for a multiobjective optimal design. IEEE Trans Energy Conv 26(2):457–467. doi:10.1109/TEC.2010.2090156

Jaszkiewicz A (2002) On the performance of multiple-objective genetic local search on the 0/1 knapsack problem A comparative experiment. IEEE Trans Evol Comput 6(4):402–412

Kukkonen S, Lampinen J (2005) GDE3: the third evolution step of generalized differential evolution. In: IEEE congress on evolutionary computation (CEC 2005), IEEE Press, pp 443–450

Kukkonen S, Lampinen J (2009) 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), IEEE Press, pp 1943–1950

Kursawe F (1991) A variant of evolution strategies for vector optimization. In: Workshop on parallel problem solving from nature (PPSN I), Springer, lecture notes in computer science, vol 496, pp 193–197

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

Luke S (2013) Essentials of metaheuristics, 2nd edn. Lulu. http://cs.gmu.edu/~sean/book/metaheuristics/

Miettinen K (1999) Nonlinear multiobjective optimization. Kluwer Academic Publishers, Boston/London/Dordrecht

Price K, Storn R, Lampinen J (1997) Differential evolution. Springer, Berlin/Heidelberg

Robič T, Filipič B (2005) DEMO: differential evolution for multiobjective optimization. International conference on evolutionary multi-criterion optimization (EMO (2005) Springer. Springer, Berlin/Heidelberg, pp 520–533

Storn R, Price KV (1997) Differential evolution—a simple and effcient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359

Van Veldhuizen D, Lamont G (1998) Multiobjective evolutionary algorithm research: A history and analysis, tech. rep. tr-98-03. Tech. rep., Dept. Elec. Comput. Eng., Graduate School of Eng., Air Force Inst. Technol., Wright-Patterson, AFB, OH

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Yagoubi M, Thobois L, Schoenauert M (2011) Asynchronous evolutionary multi-objective algorithms with heterogeneous evaluation costs. In: IEEE congress on evolutionary computation (CEC 2011), pp 21–28. doi:10.1109/CEC.2011.5949593

Zaharie D (2009) Influence of crossover on the behavior of differential evolution algorithms. Appl Soft Comput 9(3):1126–1138

Zhang Q, Li H (2007) MOEA/D: a multi-objective evolutionary algorithm based on decomposition. IEEE Tran Evol Comput 11(6):712–731

Zhang Q, Liu W, Li H (2009) The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. Technical report, School of CS and EE, University of Essex

Zitzler E (1999) Evolutionary algorithms for multiobjective optimization: Methods and applications. Ph.D. thesis, Swiss Federal Institute of Technology

Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8(2):173–195

Zitzler E, Laumanns M, Thiele L (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), pp 95–100

Zitzler E, Thiele L, Laumanns M, Fonseca CM, da Fonseca VG (2003) Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans Evol Comput 7(2):117–132

Zăvoianu AC, Bramerdorfer G, Lughofer E, Silber S, Amrhein W, Klement EP (2013) Hybridization of multi-objective evolutionary algorithms and artificial neural networks for optimizing the performance of electrical drives. Eng Appl Artif Intel 26(8):1781–1794. doi:10.1016/j.engappai.2013.06.002

Zăvoianu AC, Lughofer E, Amrhein W, Klement EP (2013) Efficient multi-objective optimization using 2-population cooperative coevolution. Computer aided aystems theory—EUROCAST, (2013) vol 8111. Lecture notes in computer science, Springer, Berlin Heidelberg, pp 251–258

Zăvoianu AC, Lughofer E, Koppelstätter W, Weidenholzer G, Amrhein W, Klement EP (2013c) On the performance of master-slave parallelization methods for multi-objective evolutionary algorithms. In: International conference on artificial intelligence and soft computing (ICAISC 2013), Springer, pp 122–134