Approximation-Guided Evolutionary Multi-Objective Optimization

Karl Bringmann[1], **Tobias Friedrich**[1], **Frank Neumann**[2], **Markus Wagner**[2]
[1] Max-Planck-Institut für Informatik, Campus E1.4, 66123 Saarbrücken, Germany
[2] School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia

Abstract

Multi-objective optimization problems arise frequently in applications but can often only be solved approximately by heuristic approaches. Evolutionary algorithms have been widely used to tackle multi-objective problems. These algorithms use different measures to ensure diversity in the objective space but are not guided by a formal notion of approximation. We present a new framework of an evolutionary algorithm for multi-objective optimization that allows to work with a formal notion of approximation. Our experimental results show that our approach outperforms state-of-the-art evolutionary algorithms in terms of the quality of the approximation that is obtained in particular for problems with many objectives.

1 Introduction

Most real-world optimization problems are characterized by multiple objectives. As these objectives are often in conflict with each other, the goal of solving a multi-objective optimization (MOO) problem is to find a (not too large) set of compromise solutions. The Pareto front of a MOO problem consists of the function values representing the different trade-offs with respect to the given objective functions. Multi-objective optimization is assumed to be more (or at least as) difficult as single-objective optimization due to the task of computing several solutions. From a computational complexity point of view even simple single-objective problems on weighted graphs like shortest paths or minimum spanning trees become NP-hard when they encounter at least two weight functions [12]. In addition, the size of the Pareto front is often exponential for discrete problems and even infinite for continuous ones.

Due to the hardness of almost all interesting multi-objective problems, different heuristic approaches have been used to tackle them. Among these methods evolutionary algorithms are frequently used as they work at each time step with a set of solutions called population. The population of an evolutionary algorithm for a MOO is used to store desired trade-off solutions for the given problem.

As the size of the Pareto front is often very large, evolutionary algorithms and all other algorithms for MOO have to restrict themselves to a smaller set of solutions. This set of solutions should be a good approximation of the Pareto front. The main question is now how to define approximation. The literature (see e.g. [7]) on evolutionary multi-objective optimization (EMO) just states that the set of compromise solutions

- should be close to the true Pareto front,
- should cover the complete Pareto front, and
- should be uniformly distributed.

There are different evolutionary algorithms for multi-objective optimization such as NSGA-II [8], SPEA2 [20], or IBEA [18] which try to achieve these goals by preferring diverse sets of non-dominated solutions.

However, the above notion of approximation is not a formal definition. Having no formal definition of approximation makes it hard to evaluate and compare algorithms for MOO problems. Therefore, we think that it is necessary to use a formal definition of approximation in this context and evaluate algorithms with respect to this definition.

Different formal notions of approximation have been used to evaluate the quality of algorithms for multi-objective problems from a theoretical point of view. The most common ones are the multiplicative and additive approximation (see [6, 10, 17]). Laumanns et al. [16] have incorporated this notion of approximation in an evolutionary algorithm for MOO. However, this algorithm is mainly of theoretical interest as the desired approximation is determined by a parameter of the algorithm and is not improved over time. Another approach related to a formal notion of approximation is the popular hypervolume indicator [19] which measures the volume of the dominated portion of the objective space. Hypervolume-based algorithms such as MO-CMA-ES [15] or SMS-EMOA [2] are well-established for solving MOO problems. They do not use a formal notion of approximation but it has recently been shown that the worst-case approximation obtained by optimal hypervolume distributions is asymptotically equivalent to the best worst-case approximation achievable by all sets of the same size [4, 5]. The major drawback of the hypervolume approach is that it cannot be computed in time polynomial in the number of objectives unless $P = NP$ [3].

In this paper, we introduce an efficient framework of an evolutionary algorithm for MOO that works with a formal notion of approximation and improves the approximation qual-

ity during its iterative process. The algorithm can be applied to a wide range of notions of approximation which are formally defined. As the algorithm does not have complete knowledge about the true Pareto front, it uses the best knowledge obtained so far during the optimization process.

The intuition for our algorithm is as follows. During the optimization process, the current best set of compromise solutions (usually called "population") gets closer and closer to the Pareto front. Similarly, the set of all non-dominated points seen so far in the objective space (we call this "archive") is getting closer to the Pareto front. Additionally, the archive is getting larger and larger and becoming an increasingly good approximation of the true Pareto front. Assuming that the archive approximates the Pareto front, we then measure the quality of the population by its approximation with respect to the archive. In our algorithm

- any set of feasible solutions constitutes an (potentially bad) approximation of the true Pareto front, and
- we optimize the approximation with respect to all solutions seen so far.

We show that this approach is highly successful in obtaining approximations according to the formal definition. Comparing our results to state of the art approaches such as NSGA-II, SPEA2, and hypervolume based algorithms, we show that our algorithm gives significantly better approximations fulfilling the formal definition.

The outline of this paper is as follows. We introduce some basic definitions in Section 2. The basic algorithm is presented in Section 3 and speed up techniques are discussed in Section 4. We report on our experimental results in Section 5 and finish with some conclusions.

2 Preliminaries

We consider minimization problems with d objective functions, where $d \geq 2$ holds. Each objective function $f_i \colon S \mapsto \mathbb{R}$, $1 \leq i \leq d$, maps from the considered search space S into the real values. In order to simplify the presentation we only work with the dominance relation on the objective space and mention that this relation transfers to the corresponding elements of S.

For two points $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$, with $x, y \in \mathbb{R}^d$ we define the following dominance relation:

$$x \preceq y \ :\Leftrightarrow\ x_i \leq y_i \text{ for all } 1 \leq i \leq d,$$
$$x \prec y \ :\Leftrightarrow\ x \preceq y \text{ and } x \neq y.$$

The typical notions of approximation used in theoretical computer science are multiplicative and additive approximation. We use the following definition

Definition 1. *For finite sets $S, T \subset \mathbb{R}^d$, the additive approximation of T with respect to S is defined as*

$$\alpha(S,T) := \max_{s \in S} \min_{t \in T} \max_{1 \leq i \leq d} (s_i - t_i).$$

This paper only regards additive approximation. However, our approach can be easily adapted to multiplicative approximation where in Definition 1 just the term $s_i - t_i$ has to be replaced by s_i / t_i.

Algorithm 1: Measure approximation quality of a population

input : Archive A, Population P
output: Indicator $S_\alpha(A, P)$

```
S ← ∅;
foreach a ∈ A do
    δ ← ∞;
    foreach p ∈ P do
        ρ ← −∞;
        for i ← 1 to d do
            ρ ← max{ρ, a_i − p_i};
        δ ← min{δ, ρ};
    S ← S ∪ {δ};
sort S decreasingly;
return S;
```

Algorithm 2: Insert point into archive

input : Archive A, Point $p \in \mathbb{R}^d$
output: Archive consisting of the Pareto optimal points of $A \cup \{p\}$

```
dominated ← false;
foreach a ∈ A do
    if a ≺ p then delete a from A;
    if a ⪰ p then dominated ← true;
if not dominated then add p to A;
return A;
```

Our aim is to minimize the additive approximation of the population P we output with respect to the archive A of all points seen so far, i.e., we want to minimize $\alpha(A, P)$. The problem is that $\alpha(A, P)$ is not sensitive to local changes of P. $\alpha(A, P)$ only measures improvements of points which are currently worst approximated.

To get a sensitive indicator which can be used to guide the search, we consider instead the set $\{\alpha(\{a\}, P) \mid a \in A\}$ of all approximations of the points in A. We sort this set decreasingly and call the resulting sequence

$$S_\alpha(A, P) := (\alpha_1, \ldots, \alpha_{|A|}).$$

The first entry α_1 is again $\alpha(A, P)$. Our new goal it then to minimize $S_\alpha(A, P)$ *lexicographically*. Note that this is an augmentation of the order induced by $\alpha(A, P)$: If we have $\alpha(A, P_1) < \alpha(A, P_2)$ then we also have $S_\alpha(A, P_1) <_{\text{lex}} S_\alpha(A, P_2)$. Moreover, this indicator is locally sensitive. Algorithm 1 describes how to calculate it.

3 Simple Algorithm

Given the definition of $S_\alpha(A, P)$, it is easy to come up with an algorithm which minimizes it lexicographically. Algorithm 3 presents such an algorithm. It maintains a population of μ individuals. In each generation, it generates λ new offspring. From the union of the old population and the offspring generation it iteratively removes the individual p for which $S_\alpha(A, P \setminus \{p\})$ is lexicographically smallest. Every

Algorithm 3: Simple $(\mu+\lambda)$-Approximation Guided EA

Initialize population P with μ random individuals;
Set archive $A \leftarrow P$;
foreach *generation* **do**
  Initialize offspring population $O \leftarrow \emptyset$;
  for $j \leftarrow 1$ **to** λ **do**
    Select two random individuals from P;
    Apply crossover and mutation;
    Add new individual to O;
  foreach $p \in O$ **do**
    Insert offspring p in archive A with Algorithm 2;
  Add offsprings to population, i.e., $P \leftarrow P \cup O$;
  while $|P| > \mu$ **do**
    foreach $p \in P$ **do**
      Compute $S_\alpha(A, P \setminus \{p\})$ with Algorithm 1;
    Remove p from P for which $S_\alpha(A, P \setminus \{p\})$ is lexicographically smallest;

Algorithm 4: Fast $(\mu + \lambda)$-Approximation Guided EA

1–11 See lines 1–11 of Algorithm 3
foreach $a \in A$ **do**
  $p_1(a) \leftarrow \operatorname{argmin}_{p\in P} \alpha(\{a\},\{p\})$;
  $p_2(a) \leftarrow \operatorname{argmin}_{p_1(a)\neq p\in P} \alpha(\{a\},\{p\})$;
  $\alpha_1(a) \leftarrow \min_{p\in P} \alpha(\{a\},\{p\})$;
  $\alpha_2(a) \leftarrow \min_{p_1(a)\neq p\in P} \alpha(\{a\},\{p\})$;
foreach $p \in P$ **do**
  $\beta(p) \leftarrow \max_{a\in A}\{\alpha_2(a) \mid p_1(a) = p\}$;
while $|P| > \mu$ **do**
  Remove p^* from P with $\beta(p)$ minimal;
  foreach $a \in A$ with $p_1(a) = p^*$ **do**
    Compute $p_1(a), p_2(a), \alpha_1(a), \alpha_2(a)$ as done above in lines 13–16;
    $\beta(p_1(a)) \leftarrow \max\{\beta(p_1(a)), \alpha_2(a)\}$;

new individual is added to the archive A such that the archive only contains non-dominating solutions. As described in Algorithm 2, this means that (i) a new offspring is only added if it is not dominated by another individual already in A and (ii) individuals in A which are dominated by a new individual are removed. Note that in contrast to many other algorithms (like Laumanns et al. [16] or all hypervolume-based algorithms), our new algorithms needs no meta-parameters.

We now give an upper bound for the runtime of Algorithm 2. One generation consists of producing and processing λ offspring. The main part of the runtime is needed for the $\mathcal{O}(\lambda(\mu+\lambda))$ computations of $S_\alpha(A, P \setminus \{p\})$, each costing $\mathcal{O}(d\,|A|\,(\mu+\lambda) + |A|\log|A|)$. Hence, we get a runtime of $\mathcal{O}(\lambda(\mu+\lambda)\,|A|\,(d\,(\mu+\lambda) + \log|A|))$ for generating an offspring of λ points. This means for N function evaluations, that is, N generated points overall, we get a total runtime of

$$\mathcal{O}(N\,(\mu+\lambda)\,|A|\,(d\,(\mu+\lambda) + \log|A|))$$

This algorithm works well for small population and offspring sizes $\mu+\lambda$, but for e.g. $\mu+\lambda = 100$, it becomes very slow due to the $(\mu+\lambda)^2$ factor.

4 Fast Algorithm

We now show how to speed-up our approach. Let us first assume that the approximations $\alpha(\{a\},\{p\})$ are distinct for all $a \in A$ and $p \in P$. For all $a \in A$ we denote the point $p \in P$ that approximates it best by $p_1(a)$ and the second best by $p_2(a)$. The respective approximations we denote by $\alpha_i(a) := \alpha(\{a\},\{p_i(a)\})$ for $i \in \{1,2\}$. Now, let $p \neq q \in P$ and consider $S_p := S_\alpha(A, P \setminus \{p\})$ and $S_q := S_\alpha(A, P \setminus \{q\})$. Significant for the comparison of the two are only the positions $a \in A$ where S_p or S_q differ from $S := S_\alpha(A,P)$. This is the case for all positions in $B := \{a \in A \mid p_1(a) \in \{p,q\}\}$. If we delete p from P, then the worst approximation of one of the $a \in B$ is the maximum of $\max\{\alpha_2(a) \mid p_1(a) = p\}$ and $\max\{\alpha_1(a) \mid p_1(a) = q\}$. Now observe that if

$$\beta(p) := \max_{a\in A}\{\alpha_2(a) \mid p_1(a) = p\}$$

is smaller than the respective $\beta(q)$, then also the larger term above is smaller, as $\max\{\alpha_1(a) \mid p_1(a) = q\} < \max\{\alpha_2(a) \mid p_1(a) = q\}$. Hence, we end up with the fact that we only have to compare $\beta(p)$ and throw out the point p with $\beta(p)$ minimal (see Algorithm 4).

Recall that we assumed that all approximations $\alpha(\{a\},\{p\})$ with $a \in A$, $p \in P$ are distinct. If this does not hold, we can simply change the indicator $S_\alpha(A,P)$ slightly and insert symmetry breaking terms $a \cdot \varepsilon$, where $\varepsilon > 0$ is an infinitesimal small number. This means that we treat equal approximations as not being equal and hence in some arbitrary order.

We now give an upper bound for the runtime of Algorithm 4. For one generation, i.e., for producing and processing λ offspring, the algorithm needs a runtime of $\mathcal{O}(d\,(\mu+\lambda)\,|A|)$ for computing the values $p_1(a), p_2(a), \alpha_1(a), \alpha_2(a)$ and $\beta(p)$ initially. Then we repeat λ times: We delete the point $p^* \in P$ with $\beta(p)$ minimal in $\mathcal{O}(\mu+\lambda)$, after which we have to recompute the values $p_1(a), p_2(a), \alpha_1(a), \alpha_2(a)$, but only for $a \in A$ with $p_1(a) = p^*$. Observe that we can store a list of these a's during the initial computation and keep these lists up to date with no increase of the asymptotic runtime. Also note that we would expect to find $\mathcal{O}(|A|/|P|)$ points with $p_1(a) = p^*$, while in the worst case there may be up to $\mathcal{O}(|A|)$ such points. Summing up, we get a heuristic runtime for one generation of $\mathcal{O}(d\,(\mu+\lambda)\,|A| + \lambda((\mu+\lambda) + d|P| \cdot |A|/|P|))$ which simplifies to $\mathcal{O}(d(\mu+\lambda)|A|)$ as $|A| \geq \mu+\lambda$. In the worst case we replace $\mathcal{O}(|A|/|P|)$ by $\mathcal{O}(|A|)$ and get a runtime for one generation of $\mathcal{O}(d\lambda(\mu+\lambda)|A|)$. For N fitness evaluations we, therefore, get a runtime of $\mathcal{O}(d(1+\mu/\lambda)|A|N)$ heuristically, and $\mathcal{O}(d(\mu+\lambda)|A|N)$ in the worst case. Note that $|A| \leq N$. For any $\lambda = \mathcal{O}(\mu)$, e.g. $\lambda = 1$ or $\lambda = \mu$, this can be simplified to $\mathcal{O}(d\mu|A|N)$ in both cases, while for $\lambda = \Omega(\mu)$, e.g. $\lambda = \mu$, we get a reduced heuristic runtime of $\mathcal{O}(d|A|N)$.

5 Experimental Study

The fast $(\mu+\lambda)$-version of our algorithm was implemented in the jMetal framework [11]. The code is available upon re-

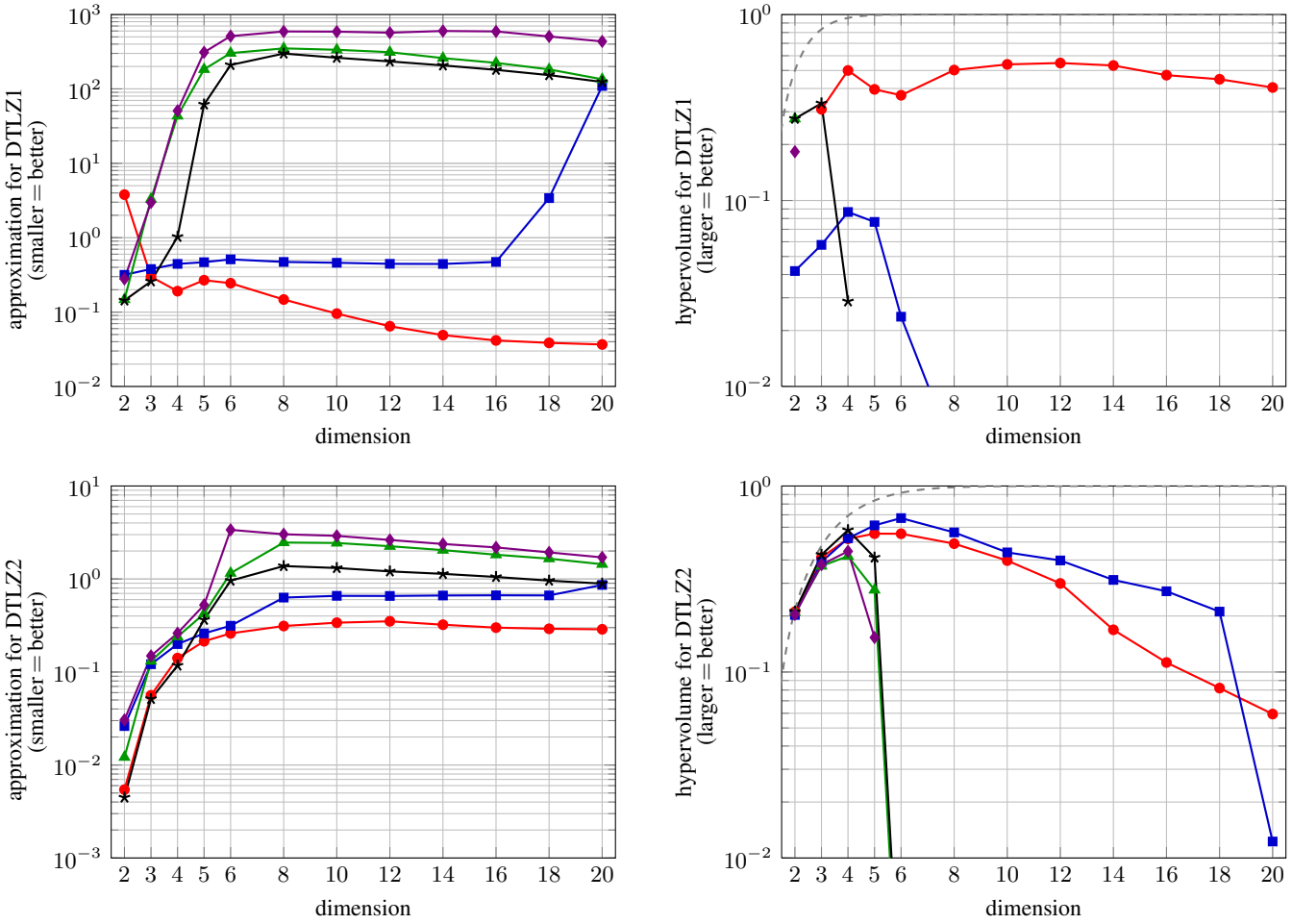
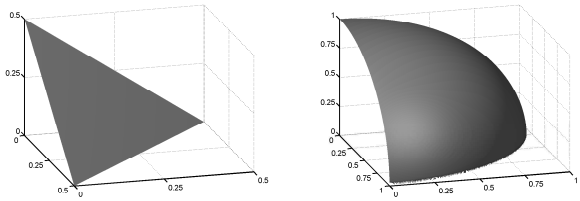


Figure 1: Comparison of the performance of our algorithm AGE (—●—) with IBEA (—■—), NSGA-II (—▲—), SMS-EMOA (—★—), and SPEA2 (—◆—) on the test functions DTLZ1 and DTLZ2 with varying dimension d. The figures show the average of 100 repetitions each. Only non-zero hypervolume values are shown. For reference, we also plot (- - -) the maximum hypervolume achievable for $\mu \to \infty$.

quest. We compared the performance of our AGE algorithm to the established MOO algorithms IBEA [18], NSGA-II [8], SMS-EMOA [13], and SPEA2 [20] on the DTLZ benchmark family [9]. We used the functions DTLZ 1-4, each with 30 function variables and between 2 to 20 objective values/dimensions. Figure 3 shows the Pareto fronts of DTLZ1 and DTLZ2 for three objectives. The fronts of DTLZ3 and DTLZ4 are equivalent to DTLZ2; they only differ in the mapping from the search space to the objective space. We limit the calculations of the algorithms to a maximum of 100,000 fitness evaluations *and* a maximum computation time of four hours per run. Note that the time restriction had to be used as the runtime of some algorithms increases exponentially with respect to the size of the objective space.

The further parameter setup of the algorithms was as follows. Parents were selected through a binary tournament, in which we selected the individual out of two randomly drawn ones with the better approximation of the archive. As variation operators, the polynomial mutation and the simulated binary crossover [1] were applied, which are both used widely in MOO algorithms [8, 14, 20]. The distribution parameters associated with the operators were $\eta_m = 20.0$ and $\eta_c = 20.0$. The crossover operator is biased towards the creation of offspring that are close to the parents, and was applied with $p_c = 0.9$. The mutation operator has a specialized explorative effect for MOO problems, and was applied with $p_m = 1/(\text{number of variables})$.



(a) DTLZ1. (b) DTLZ2.

Figure 3: Visualization of the Pareto fronts for $d = 3$.

Figures 1 and 2 present our results for population size $\mu = 100$ and $\lambda = 100$, averaged over 100 independent runs. We performed the same experiments also for $\mu \in \{25, 50\}$ and observed similar behaviors. We assess the algorithms using

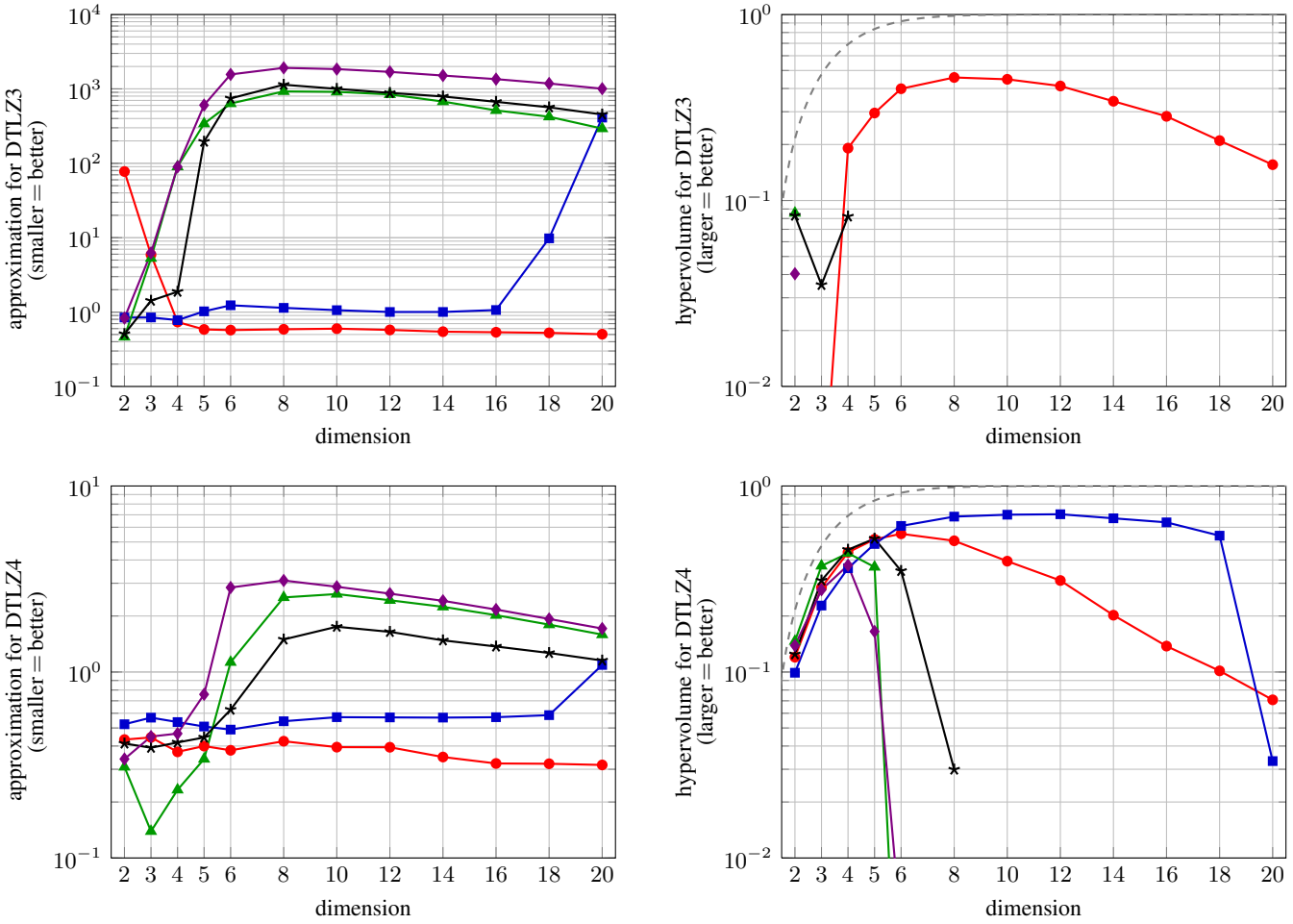


Figure 2: Comparison of the performance of our algorithm AGE (—●—) with IBEA (—■—), NSGA-II (—▲—), SMS-EMOA (—★—), and SPEA2 (—◆—) on the test functions DTLZ3 and DTLZ4 with varying dimension d. The figures show the average of 100 repetitions each. Only non-zero hypervolume values are shown. For reference, we also plot (- - -) the maximum hypervolume achievable for $\mu \to \infty$.

the following measures:

- *Approximation:* We approximate the achieved additive approximation of the known Pareto fronts by first drawing one million points of the front uniformly at random and then computing the approximation which the final population achieved for this set with Algorithm 1.
- The *hypervolume* [19] is a popular performance measure which measures the volume of the dominated portion of the objective space relative to a reference point r. For DTLZ1 we choose $r = 0.5^d$, otherwise $r = 1^d$. We approximate the achieved hypervolume with an FPRAS [3] which has a relative error of more than 2% with probability less than $1/1000$. The volumes shown for DTLZ1 are normalized by the factor 2^d.

As it is very hard to determine the minimum approximation ratio achievable or the maximum hypervolume achievable for all populations of a fixed size μ, we only plot the theoretical maximum hypervolume for $\mu \to \infty$. For this, a simple geometric calculation gives a maximum (rescaled) hypervolume of $1 - 1/d!$ for DTLZ1 and a maximum hypervolume of $1 - 2^{-d}\,\pi^{d/2}/(d/2)!$ for DTLZ2 (with $n! := \Gamma(n+1)$).

For all test functions, our new algorithm AGE (—●—) achieves the best approximation among the competing algorithms for dimensions $d > 5$. We first discuss DTLZ1 and DTLZ3 who are known to be hard to analyze as they contain a very large number of local Pareto-optimal fronts [9]. For both functions, we achieve the best approximation among all tested algorithms for $d > 3$. Remarkably, all other algorithms (besides IBEA (—■—)) are unable to find the front at all for these instances. This results in extremely large approximations and zero hypervolumes. The reason for IBEAs decreasing behaviour for very large dimension ($d \geq 18$) is that it was stopped after four hours and it could not perform $100'000$ iterations. The same holds already for much smaller dimensions in the case of SMS-EMOA (—★—), which uses an exponential-time algorithm to internally determine the hypervolume. It did not finish a single generation for $d \geq 8$ and only performed around $5'000$ iterations within four hours for $d = 5$. This implies that the higher-dimensional approximations plotted for SMS-EMOA (—★—) actually show the approximation of the random initial population. Interestingly, the approximations achieved by NSGA-II (—▲—) and

SPEA2 (—♦—) are even worse as they are tuned for low-dimensional problems and move their population too far out to the boundaries for high dimensions. Our algorithm (—●—) and also NSGA-II (—▲—) and SPEA2 (—♦—) always finished in less than four hours.

The plots of DTLZ2 and DTLZ4 reveal other properties. Here, an approximation of the front seems generally much easier. For small dimensions ($d = 2, 3, 4$), all algorithms find acceptable solutions. However, for larger dimensions again SMS-EMOA (—✶—), NSGA-II (—▲—), and SPEA2 (—♦—) fail for the said reasons. In these cases, our algorithm (—●—) still achieves the best approximation, but for $4 \leq d \leq 18$ (DTLZ2) and $6 \leq d \leq 18$ (DTLZ4), the solutions found by IBEA (—■—) (which uses the hypervolume as an indicator) have a larger hypervolume. The hypervolume of IBEA is only worse for $d = 20$ because it could only perform a few hundred iterations within the four hour time limit.

6 Conclusions

Evolutionary algorithms are frequently used to solve multi-objective optimization problems. Often, it is very hard to formally define the optimization that current state-of-the-art approach work with. We have presented a new evolutionary multi-objective algorithm that works with a formal notion of approximation. The framework of our algorithm allows to work with various formal notions of approximations. Our experimental results show that given a fixed time budget it outperforms current state-of-the-art approaches in terms of the desired additive approximation as well as the covered hypervolume on standard benchmark functions. This holds, in particular, for problems with many objectives, which most other algorithms have difficulties dealing with.

References

[1] R. B. Agrawal and K. Deb. Simulated binary crossover for continuous search space. Technical report, 1994.

[2] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[3] K. Bringmann and T. Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Computational Geometry: Theory and Applications*, 43:601–610, 2010.

[4] K. Bringmann and T. Friedrich. Tight bounds for the approximation ratio of the hypervolume indicator. In *Proc. 11th International Conference Parallel Problem Solving from Nature (PPSN XI)*, volume 6238 of *LNCS*, pages 607–616. Springer, 2010.

[5] K. Bringmann and T. Friedrich. The maximum hypervolume set yields near-optimal approximation. In *Proc. 12th annual Conference on Genetic and Evolutionary Computation (GECCO '10)*, pages 511–518. ACM Press, 2010.

[6] C. Daskalakis, I. Diakonikolas, and M. Yannakakis. How good is the Chord algorithm? In *Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '10)*, pages 978–991, 2010.

[7] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, Chichester, UK, 2001.

[8] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Computation*, 6(2):182–197, 2002.

[9] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary Multiobjective Optimization*, Advanced Information and Knowledge Processing, pages 105–145. 2005.

[10] I. Diakonikolas and M. Yannakakis. Small approximate Pareto sets for biobjective shortest paths and other problems. *SIAM Journal on Computing*, 39:1340–1371, 2009.

[11] J. J. Durillo, A. J. Nebro, and E. Alba. The jMetal framework for multi-objective optimization: Design and architecture. In *Proc. Congress on Evolutionary Computation (CEC '10)*, pages 4138–4325. IEEE Press, 2010.

[12] M. Ehrgott. *Multicriteria optimization*. Berlin, Springer, 2nd edition, 2005.

[13] M. T. M. Emmerich, N. Beume, and B. Naujoks. An EMO algorithm using the hypervolume measure as selection criterion. In *Proc. Third International Conference on Evolutionary Multi-Criterion Optimization (EMO '05)*, pages 62–76. Springer, 2005.

[14] M. Gong, L. Jiao, H. Du, and L. Bo. Multiobjective immune algorithm with nondominated neighbor-based selection. *Evolutionary Computation*, 16(2):225–255, 2008.

[15] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15:1–28, 2007.

[16] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10(3):263–282, 2002.

[17] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proc. 41st annual Symposium on Foundations of Computer Science (FOCS '00)*, pages 86–92. IEEE Press, 2000.

[18] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *Proc. 8th International Conference Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *LNCS*, pages 832–842. Springer, 2004.

[19] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evolutionary Computation*, 3:257–271, 1999.

[20] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In *Proc. Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pages 95–100, 2002.