Efficient Optimization of Many Objectives by Approximation-Guided Evolution

Markus Wagner[c,*], Karl Bringmann[a], Tobias Friedrich[b], Frank Neumann[c]

[a] *Max-Planck-Institut für Informatik, Saarbrücken, Germany*
[b] *Friedrich-Schiller-Universität Jena, Germany*
[c] *University of Adelaide, 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 framework for evolutionary multi-objective optimization that allows to work with a formal notion of approximation. This approximation-guided evolutionary algorithm (AGE) has a worst-case runtime linear in the number of objectives and works with an archive that is an approximation of the non-dominated objective vectors seen during the run of the algorithm. Our experimental results show that AGE finds competitive or better solutions not only regarding the achieved approximation, but also regarding the total hypervolume. For all considered test problems, even for many (i.e., more than ten) dimensions, AGE discovers a good approximation of the Pareto front. This is not the case for established algorithms such as NSGA-II, SPEA2, and SMS-EMOA. In this paper we compare AGE with two additional algorithms that use very fast hypervolume-approximations to guide their search. This significantly speeds up the runtime of the hypervolume-based algorithms, which now allows a comparison of the underlying selection schemes.

Keywords: Multi-objective optimization, approximation, comparative study

1. Introduction

Real-world optimization problems are usually very complex and hard to solve due to different circumstances such as constraints, complex function evaluations

*Corresponding author. Phone: +61 8 8313 5405, fax: +61 8 8313 4366,
email: markus.wagner@adelaide.edu.au.

that can only be done by simulations, or multiple objectives. 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 so-called Pareto front of a MOO problem consists of the function values representing the different trade-offs with respect to the given objective functions. The set of compromise solutions that is the outcome of a MOO run is an approximation of this Pareto front, and the idea of this posteriori approach is that afterwards the decision maker selects an efficient solution from this set. Multi-objective optimization is regarded 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 [16]. 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. 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. [11]) on evolutionary multi-objective optimization (EMO) just states that the set of compromise solutions (i) should be close to the true Pareto front, (ii) should cover the complete Pareto front, and (iii) should be uniformly distributed. There are different evolutionary algorithms for multi-objective optimization such as NSGA-II [12], SPEA2 [31], or IBEA [29], 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 approximations (see [9, 10, 14, 24–26]). Laumanns et al. [22] 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 [30] that measures the volume of the dominated portion of the objective space. Hypervolume-based algorithms such as MO-CMA-ES [20] 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]. It is even NP-hard to determine which individual gives approximately the least contribution to the total hypervolume [6].

We introduce an efficient framework of an evolutionary algorithm for MOO that works with a formal notion of approximation and improves the approximation quality during its iterative process. The algorithm can be applied to a wide range of notions of approximation that 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 introduce a basic approximation guided evolutionary algorithm which already performs very well for problems with many objectives. One drawback of the basic approach is that the archive size might grow tremendously during the run of the algorithm. In order to deal with this, we propose to work with an approximative archive which keeps at each time step only an ε-approximation of all solutions seen so far. We do this by incorporating the ε-dominance approach of Laumanns et al. [22] into the algorithm. Furthermore, we introduce a powerful parent selection scheme which especially increases the performance of our algorithm for problems with just a few objectives by given the algorithm a stronger focus on the extreme points on the Pareto front.

We show on a set of well established benchmark problems that our approach is highly successful in obtaining high quality approximations according to the formal

definition. Comparing our results to state of the art multi-objective algorithms such as NSGA-II, SPEA2, IBEA, and SMS-EMOA, we show that our algorithm typically gives better results, especially for high dimensional problems.

In our experimental study, we measure the quality of the results obtained not only in terms of the approximation quality but also with respect to the achieved hypervolume. Our experiments show that the examined hypervolume-based algorithms can sometimes achieve a larger hypervolume than our algorithm AGE, but AGE is the only one considered that finds a competitive hypervolume for all functions. Hence our algorithm not only performs better regarding our formal definition of approximation on problems with many objectives, but it is also competitive (or better, depending on the function) regarding the hypervolume.

This article is based on its previous conference publications. The basic AGE algorithms has been introduced in [7]. The archive approximation has been presented in [28] and different parent selection schemes for AGE have been examined and discussed in [27].

The outline of this paper is as follows. We introduce some basic definitions in Section 2. The main idea of approximation guided evolution and the basic AGE algorithm are presented in Section 3. In Section 6 we show how to speed up the approach by using an approximative archive and discuss different parent selection schemes in Section 5. We present our experimental results in Section 8 and finish with a summary and some concluding remarks.

2. Preliminaries

Multi-objective optimization deals with the optimization of several (often conflicting) objective functions. The different objective functions usually constitute a minimization or maximization problem on their own. Optimizing with respect to all given objective functions, there is usually no single optimal objective function vector, but a set of vectors representing the different trade-offs that are imposed by the objective functions.

Without loss of generality, we consider minimization problems with d objective functions, where $d \geqslant 2$ holds. Each objective function $f_i \colon S \mapsto \mathbb{R}$, $1 \leqslant i \leqslant d$, 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:

$$
\begin{aligned}
x \preceq y \;&:\Leftrightarrow\; x_i \leqslant y_i \text{ for all } 1 \leqslant i \leqslant d, \\
x \prec y \;&:\Leftrightarrow\; x \preceq y \text{ and } x \neq y.
\end{aligned}
$$

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

Subroutine 1: Measure approximation quality of a population

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

```
1  S ← ∅;
2  foreach a ∈ A do
3      δ ← ∞;
4      foreach p ∈ P do
5          ρ ← −∞;
6          for i ← 1 to d do
7              ρ ← max{ρ, a_i − p_i};
8          δ ← min{δ, ρ};
9      S ← S ∪ {δ};
10 sort S decreasingly;
11 return S;
```

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

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

In this paper, we only consider additive approximations. However, our approach can be easily adapted to multiplicative approximations. This this case, the term $s_i - t_i$ in Definition 1 has to be replaced by s_i / t_i.

3. Basic Algorithm

Our aim is to minimize the additive approximation value 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. As its definition is based on maximum and miminum values, $\alpha(A, P)$ only measures the approximation of points that are worst approximated. Consequently, it does not take into account approximation values for points that are not "worst approximated". We will illustrate this with a very simple example. Let us consider a two-dimensional space with an archive $A = \{(1, 2), (2, 1), (3, 0)\}$ and a population $P = \{(2, 1)\}$. Then, $\alpha(A, P) = 1$ due to archive points $(1, 2)$ and $(3, 0)$. Even if points such as $(3, 0)$ or $(2.5, 0.5)$ are added to the population, the approximation value will remain $\alpha(A, P) = 1$ because of the worst approximated archive point $(1, 2)$, even though the approximation of $(3, 0)$ is significantly improved.

Subroutine 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 p ≺ a then delete a from A;
    if a ⪯ p then dominated ← true;
if not dominated then add p to A;
return A;
```

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*, meaning that we take the lexicographically smallest sequence when we face several sequences.[1] Note that this is a refinement 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. Subroutine 1 states the pseudo-code for computing $S_\alpha(A, P)$ for a given archive A and a population P.

We are now ready to describe the basic AGE algorithm. It works with the vector $S_\alpha(A, P)$ and tries to minimize it with respect to the lexicographical order. Depending on the optimization process the archive A changes and stores at each point in time for each non-dominated objective vector one single solution.

The basic AGE algorithm shown in Algorithm 3 works with a parent population of μ individuals and produces in each generation λ offspring.

A newly produced offspring p is added to the archive A if it is not dominated by any other solution found so far. If it is added to the archive, all solutions that are dominated by p are removed from A (see Subroutine 2). In order to obtain the next parent population, the set consisting of the union of the parent and offspring is considered. From this set, the individual p for which $S_\alpha(A, P \setminus \{p\})$ is lexicographically smallest is removed iteratively until a population of size μ is obtained. Note that in contrast to many other evolutionary algorithms (like [22] or all hypervolume-based algorithms), the basic AGE algorithm needs no meta-parameters besides the population sizes μ and λ.

[1] $(a_1, \ldots, a_{|A|}) <_{\text{lex}} (b_1, \ldots, b_{|A|}) \Leftrightarrow (a_1 < b_1)$ or $\big((a_1 = b_1)$ and $(a_2, \ldots, a_{|A|}) <_{\text{lex}} (b_2, \ldots, b_{|A|})\big)$

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

```
Initialize population P with μ random individuals;
Set archive A ← P;
foreach generation do
    Initialize offspring population O ← ∅;
    for j ← 1 to λ do
        Select two random individuals from P;
        Apply crossover and mutation;
        Add new individual to O, if it is not dominated by any individual
        from P;
    foreach p ∈ O do
        Insert offspring p in archive A with Subroutine 2;
    Add offspring to population, i.e., P ← P ∪ O;
    while |P| > μ do
        foreach p ∈ P do
            Compute S_α(A, P \ {p}) with Subroutine 1;
        Remove p from P for which S_α(A, P \ {p}) is lexicographically
        smallest;
```

We now analyze the runtime of the basic AGE algorithm in dependence of μ, λ, the archive size A, and the number of function evaluations N of the algorithm. 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 population of λ individuals. This means for N function evaluations, that is, N generated points overall, we get a total runtime of

$$\mathcal{O}(N\,(\mu+\lambda)\,|A|\,(d\,(\mu+\lambda)+\log|A|)) \tag{1}$$

As we can see, this basic algorithm becomes very slow due to the $(\mu+\lambda)^2$ factor when, e.g., $\mu+\lambda = 200$ is chosen. However, this algorithm works well (in the sense of runtime) for very small population and offspring sizes.

The following three sections describe three successive improvements for this basic framework of approximation guided evolution.

4. Improved Approximation Calculation

It can be observed that the selection phase is the most costly step in one iteration of the basic AGE given in Algorithm 3 as it has to evaluate the points of the parent

and offspring population against the archive. We can obtain a significant speed-up for each generation of the algorithm by cleverly updating the approximation value of the points in the archive that are affected by the removal of a point from the set consisting of the parents and offspring.

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 $B := \{a \in A \mid p_1(a) \in \{p,q\}\}$. Now, if we delete p from the population 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 minimal $\beta(p)$. This is shown in Subroutine 4, which replaces lines 12–15 of Algorithm 3.[2]

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 AGE with Subroutine 4. For one generation, i.e., for producing and processing λ offspring with one run of Subroutine 4, AGE 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 can estimate the expected runtime for one generation by $\mathcal{O}(d\,(\mu+\lambda)\,|A| + \lambda((\mu+\lambda) + d|P| \cdot |A|/|P|))$, which simplifies to $\mathcal{O}(d(\mu+\lambda)|A|)$ as $|A| \geqslant \mu+\lambda$. 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 $|A| \leqslant N$. For any $\lambda = \mathcal{O}(\mu)$,

[2] AGE with this selection scheme was called "Fast AGE" in [7].

Subroutine 4: Fast approximation-guided selection

input : Population P, Archive A, μ
output: μ individuals from P which (greedily) give best approximation of archive A

1 **foreach** $a \in A$ **do**
2 $\quad p_1(a) \leftarrow \text{argmin}_{p \in P}\, \alpha(\{a\}, \{p\})$;
3 $\quad p_2(a) \leftarrow \text{argmin}_{p_1(a) \neq p \in P}\, \alpha(\{a\}, \{p\})$;
4 $\quad \alpha_1(a) \leftarrow \min_{p \in P}\, \alpha(\{a\}, \{p\})$;
5 $\quad \alpha_2(a) \leftarrow \min_{p_1(a) \neq p \in P}\, \alpha(\{a\}, \{p\})$;
6 **foreach** $p \in P$ **do**
7 $\quad \beta(p) \leftarrow \max_{a \in A}\{\alpha_2(a) \mid p_1(a) = p\}$;
8 **while** $|P| > \mu$ **do**
9 $\quad$ Remove p^* from P with $\beta(p)$ minimal;
10 $\quad$ **foreach** $a \in A$ with $p_1(a) = p^*$ **do**
11 $\quad\quad$ Compute $p_1(a), p_2(a), \alpha_1(a), \alpha_2(a)$ as done above in lines 2–5;
12 $\quad\quad \beta(p_1(a)) \leftarrow \max\{\beta(p_1(a)), \alpha_2(a)\}$;

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 estimate of the expected runtime of $\mathcal{O}(d|A|N)$.

5. Improved parent selection

Quite interestingly, and despite the basic AGE's good performance on problems with many objectives (as shown in [7]), it is clearly outperformed by other algorithms in several cases, when the problem has just two or three objectives. The key discovery is that the random parent selection of the basic AGE is free of any bias. For problems with many objectives, this is not a problem, and can even be seen as its biggest advantage. For problems with just a few objectives, however, it is well known that one can do better than random selection, such as selection based on crowding distance, hypervolume contribution, etc. Such strategies then select potential candidates based on their relative position in the current population. For the basic AGE, the lack of this bias means that solutions can be picked for parents that are not necessarily candidates with high potential. Consequently, it is not surprising to see that the basic AGE is outperformed by algorithms that do well with their parent selection strategy, *if* their strategy is effective in the respective d-dimensional objective space.

We improve the basic AGE's performance, subject to the following conditions:

1. The introduced computation time required to select parents should be polynomial in the number of objectives d.
2. The selection mechanism should significantly improve the performance on problems with few objectives, while not influencing the performance on problems with many objectives.
3. The selection scheme should favour individuals that have the potential to improve the approximation quality.

Note that most hypervolume-based algorithms, such as SMS-EMOA and MO-CMA-ES, violate condition (1), as some of the computations that are associated with the selection process take time exponential in d. However, we have to note that it is possible to deal with this drawback by approximating the hypervolume, as shown and demonstrated in [8]. Nevertheless, as the maximization of the hypervolume can interfere with our goal of improving the approximative quality, we do not consider such approaches.

Also note that the exclusive use of domination based-criteria is problematic. Assuming a general d-dimensional unbounded space (with $d \geqslant 2$), then a point in this space dominates $1/2^d$ of the volume. Obviously then, a pure dominance check in high-dimensional spaces is extremely likely to fail. Or, when interpreted the other way around, this means that a check of the dominance relation between two solutions is extremely unlikely to bring up any additional information about the relative quality between these two solutions.

It is relatively easy to design algorithms that easily discover points at the fringe of the Pareto front. With these fringe points (or points that are very close to the fringe), the decision maker can get an idea about the achievable ranges for each objective. However, the problem of finding points "between" those fringe points proves to be much more difficult. Selection mechanisms for the (fitness-based) parent selection and the offspring selection tend to have different biases that result in different preferences for fringe points or central points, depending on the "shape" of the intermediate populations and on the shape of the true Pareto front. With an increasing number of dimensions, this problem becomes even more apparent, as solutions should evenly cover the front, while not concentrating only on extreme points.

We choose the best-performing selection scheme from [27], which works as follows in each generation. In the first step, the population is "pre-processed" (see Algorithm 5): the population is split into fronts of non-dominating solutions[3], and then solutions in the front i have a probability of $1/i$ of staying in the population. Thus, we increase the selection pressure, and solutions that are dominated multiple times are less likely to be selected as a potential parent. Additionally, we determine

[3]Iteratively, all non-dominated solutions are identified and then removed (as one front), which results in potentially several fronts of dominating solutions—see NSGA & NSGA-II [12].

Subroutine 5: Pre-processing of the population for the subsequent parent selection

input : Population P
output: Pre-processed population Q

```
1 Q ← ∅;
2 Split P into non-dominating fronts F_1, ..., F_i;
3 foreach front F_j, 1 ⩽ j ⩽ i do
4 |   foreach p ∈ F_j do
5 |   |   Add p to Q with probability 1/j;
6 Split Q into non-dominating fronts G_1, ..., G_k;
7 foreach front G_j, 1 ⩽ j ⩽ k do
8 |   Compute the crowding distances for the solutions in G_j;
9 return Q;
```

the crowding distances for the points in the reduced population. In the second step of the selection scheme, a binary tournament is performed where solutions of higher crowding distance are preferred. The crowding distance helps to pick diverse parents when the number of objectives is low.

Note that the size of the pre-processed population is not deterministic. It is only guaranteed to contain the entire first front (see Lin 5 of Algorithm 5).

6. Approximating the archive

The basic AGE algorithms stores all objective vectors into the archive that are currently not dominated by any other objective vector produced so far. It is common for multi-objective optimization problems that the number of such trade-offs can be very large, i.e. exponential with respect to the given input size in the case of discrete optimization or even infinite for continuous optimization problems. As, in the worst case, the archive size $|A|$ can grow linearly in the number of fitness function evaluations, the runtime given in Equation (1) becomes quadratic in the number of generated points N. We therefore want to work with an archive of reduced size which can lead to a significant speed up of the algorithm.

In this section, we show how we adapt the ε-dominance approach Laumanns et al. [22] in order to approximate the different points seen so far during the run of the algorithm. This archive is significantly reduced and therefore leads to a faster algorithm.

Algorithm 6: $(\mu + \lambda)$-Approximation Guided EA (AGE)

```
Initialize population P with μ random individuals;
Set ε_grid the resolution of the approximative archive A_{ε_grid};
foreach p ∈ P do
    Insert offspring floor(p) in the approximative archive A_{ε_grid} such that only
    non-dominated solutions remain;
foreach generation do
    Initialize offspring population O ← ∅;
    for j ← 1 to λ do
        Select two individuals from the pre-processed P (see Section 5);
        Apply crossover and mutation;
        Add new individual to O;
    foreach p ∈ O do
        Insert offspring floor(p) in the approximative archive A_{ε_grid} such that
        only non-dominated solutions remain;
        Discard offspring p if it is dominated by any point increment(a),
        a ∈ A;
    Add offspring to population, i.e., P ← P ∪ O;
    Apply fast approximation-guided selection of Subroutine 4 to P and
    obtain population of size μ;
```

Subroutine 7: Function *floor*

input : d-dimensional objective vector x, archive parameter ε_{grid}
output: Corresponding vector v on the ε-grid

for $i = 1$ **to** d **do** $v[i] \leftarrow \left\lfloor \frac{x[i]}{\varepsilon_{grid}} \right\rfloor$;

Subroutine 8: Function *increment*

input : d-dimensional vector x, archive parameter ε_{grid}
output: Corresponding vector v that has each of its components increased by 1

for $i = 1$ **to** d **do** $v[i] \leftarrow o[i] + 1$;

6.1. Archive approximation

In order to approximate the archive, we are facing a problem that is similar to the original problem of multi-objective optimisation, namely a set of solutions is sought that nicely represents the true set of compromise solutions.

We reuse AGE's own main idea of maintaining a small set that approximates the true Pareto front. By approximating the archive as well in a controlled manner, we can guarantee a maximum size of the archive, which directly translates into a bound with respect to the runtime of AGE when considering a fixed number of iterations.

Our archive approximation is based on the idea of ε-dominance introduced in Laumanns et al. [22]. Instead of using an archive A^t that stores at any point in time t the whole set of non-dominated objective vectors, we are using an archive $A^{(t)}_{\varepsilon_{grid}}$ that stores an additive ε-approximation of the non-dominated objective vectors produced until time step t.

In order to maintain such an approximation during the run of the algorithm, a grid on the objective space is used to pick a small set of representatives (based on ε-dominance). We reuse the *update*-mechanism from [22], and thus can maintain the ε-Pareto set $A^{(t)}_{\varepsilon_{grid}}$ of the set $A^{(t)}$ of all solutions seen so far. Due to [22], the size is bounded by

$$\left|A^{(t)}_{\varepsilon_{grid}}\right| \leqslant \prod_{j=1}^{d-1} \left\lfloor \frac{K}{\varepsilon_{grid}} \right\rfloor$$

where

$$K = \max_{i=1}^{d} \left(\max_{s \in S} f_i(s) \right)$$

is the maximum function value attainable among all objective functions.

We parameterize our algorithm by the desired approximation quality $\varepsilon_{grid} \geqslant 0$ of the archive with respect to the seen objective vectors. AGE is shown in Algorithm 6, and it uses the helper functions given in Subroutines 7 and 8. The latter is used to perform a relaxed dominance check on the offspring p in Line 13. A strict dominance check here would require an offspring to be not dominated by any point in the entire archive. However, as the archive approximates all the solutions seen so far (via the flooring), it might very unlikely, or even impossible, to find solutions that pass the strict dominance test.

6.2. Impact of archive approximation on running times

The algorithm works at each time step t with an approximation $A^{(t)}_{\varepsilon_{grid}}$ of the set of non-dominated points A^t seen until time step t. Note, that setting $\varepsilon_{grid} = 0$ implies the basic AGE approach that stores every non-dominated objective vector. We now investigate the effect of working with different archives sizes (determined by the choice of ε_{grid}) in AGE. Our goal is to understand the effect of the choice of this parameter on the actual archive size used during the run of the algorithm as well as on the approximation quality obtained by AGE.

Next, we outline the results of our experimental investigation of the influence of approximative archives on the runtime and the solution qualities. Note, that

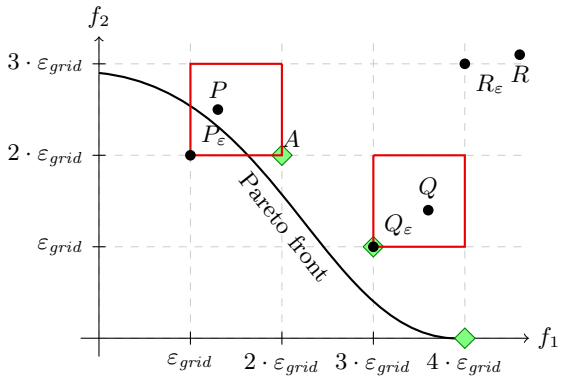

Figure 1: The newly generated points P, Q, and R are shown with their corresponding *additive ε-approximations* P_ε, Q_ε, and R_ε. Both objectives f_1 and f_2 are to be minimised, and the current *approximative archive* is represented by ◆. Only P_ε will be added to the *approximative archive*, replacing A. Both P and Q will be candidates for the selection process to form the next population.

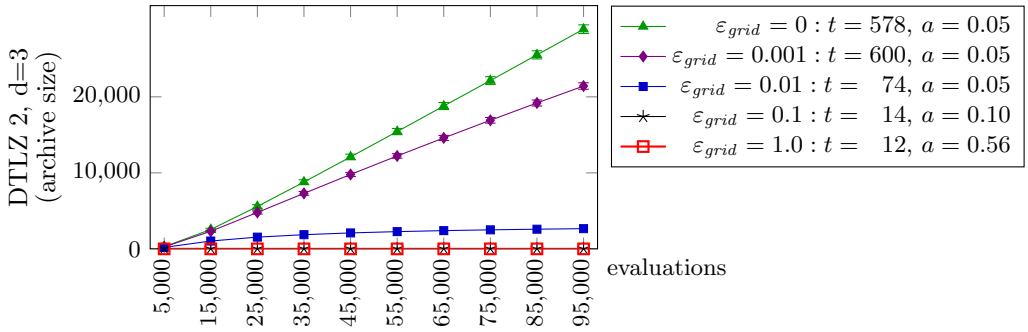

Figure 2: Influence of ε_{grid} on the archive size, the runtime, and the final quality. Shown are the means of the archive sizes, and their standard deviation is shown as error bars. Additionally, the means of the runtime t in seconds and the achieved additive approximation a of the true Pareto front are listed (smaller values are better). Note: the archive can grow linearly with the number of solutions generated, even when problem have just $d = 3$ objectives.

the computational complexity of AGE is linear in the number of objectives. The algorithm was implemented in the jMetal framework [15] and is publicly available.[4]

The parameter setup of AGE is as follows. We use polynomial mutation and the simulated binary crossover [1] in order to create new offspring. Both variation operators are widely used in MOO algorithms [12, 18, 31] and they are transformations of bit-string operators to bounded real-valued domains. The distribution param-

[4] `http://cs.adelaide.edu.au/~optlog/research/age.php`

eters associated with the operators are $\eta_m = 20.0$ and $\eta_c = 20.0$. The crossover operator is biased towards the creation of offspring that are close to the parents, and is applied with $p_c = 0.9$. The mutation operator has a special explorative effect for MOO problems, and is applied with $p_m = 1/(\text{number of variables})$.[5] Population size is set to $\mu = 100$ and $\lambda = 100$, and each run uses 100,000 fitness evaluations. We assess the quality of the final population using the additive approximation measure ([7]). First, we draw one million points of the mathematically described true Pareto front uniformly at random. Then we compute the additive approximation that the final population achieved for this sample of the true Pareto front.

Exemplary, we show in Figure 2 the results averaged over 100 independent runs for DTLZ 2 with d=3. Note that the archive grows very quickly in the case of $\varepsilon_{grid} = 0$, where every non-dominated point is stored. Without sacrificing solution quality, a speed-up by a factor of 7.8 is achieved with $\varepsilon_{grid} = 0.01$. Additional speed-ups can be achieved, but it is then up to the decision maker to balance the computation speed and the solution quality. More results can be found in [28]. For example, for DTLZ 4 with 20 objectives: "a speed-up by a factor of over 250 can be achieved, while achieving even better quality solutions as well."

The choice of ε_{grid} can have a significant impact on the final approximation, which is why we consider several values in the final experiments. In particular, with "coarser" archives, the number of points that represent a particular region decreases. Since the fast approximation-guided selection will at first consider only the single best approximating solution per cuboid, fewer points will actually represent that region. If the cuboids end up very large with the choice of a larger value of ε_{grid}, then the remaining solutions of the population (that are not the best approximating ones) are not necessarily distributed in way that results in a good approximation.

7. Discussion on impact of algorithm components

The AGE algorithm consists of different components that make the approach successful. In this section, we would like to discuss them further in detail such that practitioners become aware of the different contributions.

The framework of AGE has two components that speed up the computation. The first one is the faster approximation calculation described in Section 4 which gives a runtime speed-up compared to the basic approach without any impairment in terms of quality, i.e., the same set of solutions is computed. This improvement in terms of running time should always be incorporated as it does not come with any disadvantage compared to the basic framework (as described in Section 3).

[5]Note that other setups can be used, including different recombination and exploration operators. However, this is beyond the scope of this article as we focus on the comparison of the algorithms.

A further significant speed-up is obtained by working with an approximative archive as outlined in Section 6. Here, the parameter ε_{grid} determines the size of the archive during the run of the algorithm. This component imposes a trade-off in running time and approximation behavior as an increasing value of ε_{grid} leads to a speed-up of the approach at the expense of a worsening in the approximation. Setting the parameter ε_{grid} is crucial for the success of the algorithm and a good choice is dependent on the given multi-objective problem. Our experimental studies on ε_{grid} have shown that the value can be chosen to gain very significant speed-ups with almost no impairment in terms of quality.

While the faster approximation calculation and the approximative archive mainly aim for a speed up of the algorithm, the improved parent selection introduced in Section 5 aims for a better spread of the population in the objective space. As the runtime of AGE is mainly determined by the size of its archive different methods can be exploited without having a huge impact on the running time. Different parent selection methods have been examined in [27] and the best performing one is integrated into the final AGE algorithm.

8. Experimental investigations

In this section, we compare AGE to well-known evolutionary multi-objective algorithms on commonly used benchmark functions. We first study low-dimensional problems and later pay special attention to problems with many dimensions. We judge the algorithms by the approximation quality and the hypervolume that they achieve. AGE is investigated for different values of ε in order to study the effect of working with an approximative archive on the quality of the results.

8.1. Low-dimensional problems

In our first study, we investigate the performance of AGE on problems with few objectives. We use the jMetal framework [15] to compare AGE with the established algorithms IBEA [29], NSGA-II [12], SMS-EMOA [17], and SPEA2 [31] on the benchmark families WFG [19] and LZ [23], and DTLZ [13]. For each of the problems, the objective values are within "roughly" the same ranges. When facing a problem with significantly differing ranges, we recommend ()as we do for other algorithms) to rescale the objectives for the algorithms into comparable ranges, as mechanisms like hypervolume or density computations will not necessarily produce "evenly spread" outcomes as intended by the respective algorithms' authors.

It is important to note that we limit the calculations of the algorithms to a maximum of 50,000/100,000/150,000 fitness evaluations for WFG/DTLZ/LZ *and* to a maximum computation time of 4 hours per run, as the runtime of some algorithms increases exponentially with respect to the size of the objective space. The further parameter setup of the algorithms is as follows. Parents are selected through a binary tournament. We will present our results for population sizes $\mu = 100$ and

$\lambda = 100$ and average the results over 100 independent runs. The AGE test setup has been outlined in Section 6.2.

We assess the algorithms by examining their final populations. To measure the quality of the final population, we consider the additive approximation and the *hypervolume* [30]. The latter is very popular in the performance assessment of evolutionary multi-objective algorithms and measures the volume of the dominated portion of the objective space relative to a reference point r. For the quality assessment on the WFG and LZ functions, we compute the achieved additive approximations and the hypervolumes with respect to the Pareto fronts given in the jMetal package. For the DTLZ functions, we compute the additive approximations as described in Section 6.2. For the hypervolume computations for DTLZ 1 we choose $r = 0.5^d$, and $r = 1^d$ for all other benchmark problems. We approximate the achieved hypervolume with an FPRAS [3], which has a relative error of not more than 2% with probability at 1/1000. The volumes shown for DTLZ 1 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 $\mu \to \infty$ as a reference.

Results. The benchmarking results for the different algorithms are shown in Figures 3 and 4 and are a clear evidence for AGE's excellent performance. AGE ranks among the best algorithms on the low-dimensional WFG and LZ functions (see Figure 3). This holds for the additive approximation quality as well as for the achieved hypervolumes. Interestingly, NSGA-II (—⋆—), which normally performs rather well on such problems, is beaten in almost all cases. AGE performs very similarly for the different used approximative archive settings ($\varepsilon_{grid} = 0$: —○—, $\varepsilon_{grid} = 0.1$: —○—, $\varepsilon_{grid} = 0.01$: —○—). This confirms that working with an approximative archive usually leads to a significant speed-up without a detrimental effect on the solution quality.

Our investigations on the DTLZ family (see Figure 4) prove to be more differentiating between the different type of algorithms. The DTLZ family can be scaled with the number of objectives and therefore enables us to investigate the impact for problems with more than two objectives. With an increasing number of objectives, the benefits and drawbacks of the algorithms' underlying mechanisms become more apparent. We can summarize the experimental results in the following way.

- AGE ($\varepsilon_{grid} = 0$: —○—, $\varepsilon_{grid} = 0.1$: —○—, $\varepsilon_{grid} = 0.01$: —○—) shows a very good performance on all DTLZ variants. It is either the best performing algorithm, or in many cases, it shows at least competitive performance.

- It is interesting to see that even though AGE incorporates the crowding distance idea from NSGA-II (—▲—) for a fitness assignment, it is not influenced

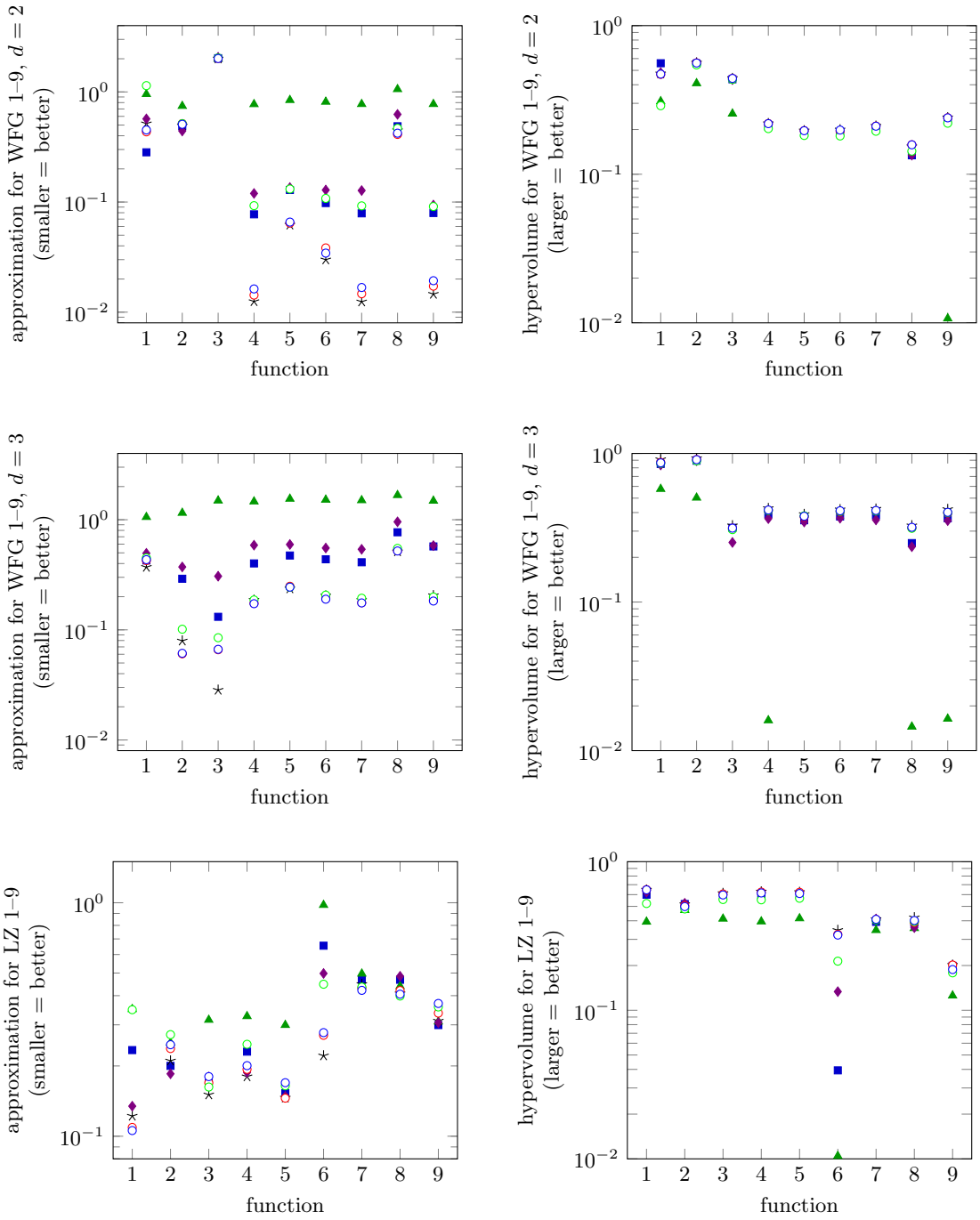

Figure 3: Comparison of the performance of our AGE ($\varepsilon_{grid} = 0$: —○—, $\varepsilon_{grid} = 0.1$: —○—, $\varepsilon_{grid} = 0.01$: —○—) with IBEA (—■—), NSGA-II (—▲—), SMS-EMOA (—★—), and SPEA2 (—♦—) on the test function classes WFG (2 and 3 dimensions) and LZ (2 dimensions for LZ 1–5 and LZ 7–9; 3 dimensions for LZ 6). The figures show the average of 100 repetitions each. Only non-zero hypervolume values are shown.

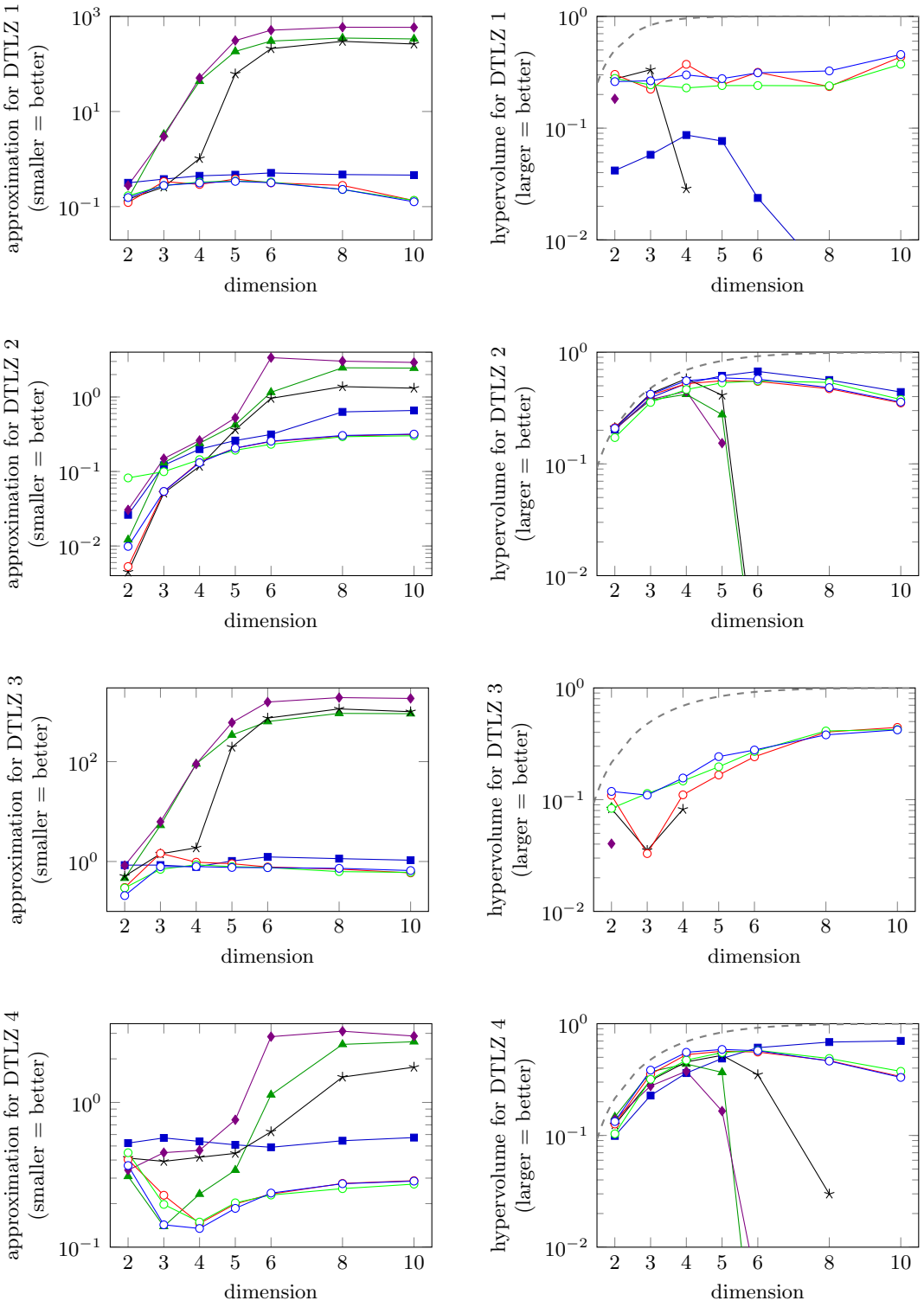

Figure 4: Comparison of the performance of AGE ($\varepsilon_{grid} = 0$: —○—, $\varepsilon_{grid} = 0.1$: —○—, $\varepsilon_{grid} = 0.01$: —○—) with IBEA (—■—), NSGA-II (—▲—), SMS-EMOA (—★—), and SPEA2 (—♦—) on DTLZ test functions with $d \leqslant 10$ dimensions. The figures show the average of 100 repetitions each. We limit the computations per repetition to a maximum of 100,000 evaluations and to a maximum computation time of 4 hours. Only non-zero hypervolume values are shown. For reference, we also plot (- - -) the maximum hypervolume achievable for $\mu \to \infty$.

by its detrimental effects in higher dimensional objective spaces. This is a consequence of how the next generation is formed (i.e., based on contribution to the approximation quality achieved with respect to the archive, see Line 16 of Algorithm 6).

- Remarkably, NSGA-II (—▲—), SMS-EMOA (—★—), and SPEA2 (—♦—) are unable to find the front of the higher-dimensional DTLZ 1 and DTLZ 3 variants. This results in extremely large approximation values and zero hypervolume values. In particular, the mechanisms used by NSGA-II (—▲—) and SPEA2 (—♦—) are inadequate for higher-dimensional spaces, and both algorithms push their population too far out to the boundaries for high dimensions.

- For higher dimensions ($d \geqslant 5$) IBEA (—■—) is AGE's strongest competitor. However, its performance is not consistent for all functions and its runtime does not scale well with increasing dimension. The same holds for SMS-EMOA (—★—), which uses an exponential-time algorithm to internally determine the hypervolume. It did not finish a single generation for $d \geqslant 8$ and only performs around $5,000$ iterations within four hours for $d = 5$.

8.2. High-dimensional problems

Encouraged by the good performance of AGE on lower-dimensional test problems, we also study high-dimensional problems with dimensions $d > 10$. It is known that the classical algorithms SPEA2 and NSGA-II deteriorate with an increasing number of objectives. Also for SMS-EMOA we observed runtime issues for higher-dimensional spaces. For a meaningful comparison we therefore neglect these algorithms for higher-dimensional test problems and instead compare AGE with two recent EMOA specifically designed for high-dimensional problems. In particular, we compare AGE with two hypervolume-based algorithms that use fast approximations of the hypervolume to guide their search, namely MO-CMA-ES [20] and SMS-EMOA [17], which are both implemented in the Shark Machine Learning Library [21]. Note that we again include IBEA in this final comparison due to its good performance on DTLZ 2 and 4 for lower number of objectives.

Among the studied test problems, only DTLZ [13] allows scaling to an arbitrary number of objective space dimensions. We therefore study DTLZ1–4 for up to 20 dimensions. The test setup remains unchanged, with the difference that we limit the calculations of the algorithms to a maximum of 250,000 fitness evaluations *and* to a maximum computation time of 24 hours per run, due to the increased difficulty. As this is our final test, we also compared the algorithms using the Wilcoxon-Mann-Whitney two-sample rank-sum test. If we call a comparison "statistically significant", it is significant at the 99% confidence level. The results are shown in Figure 5 and summarized as follows.

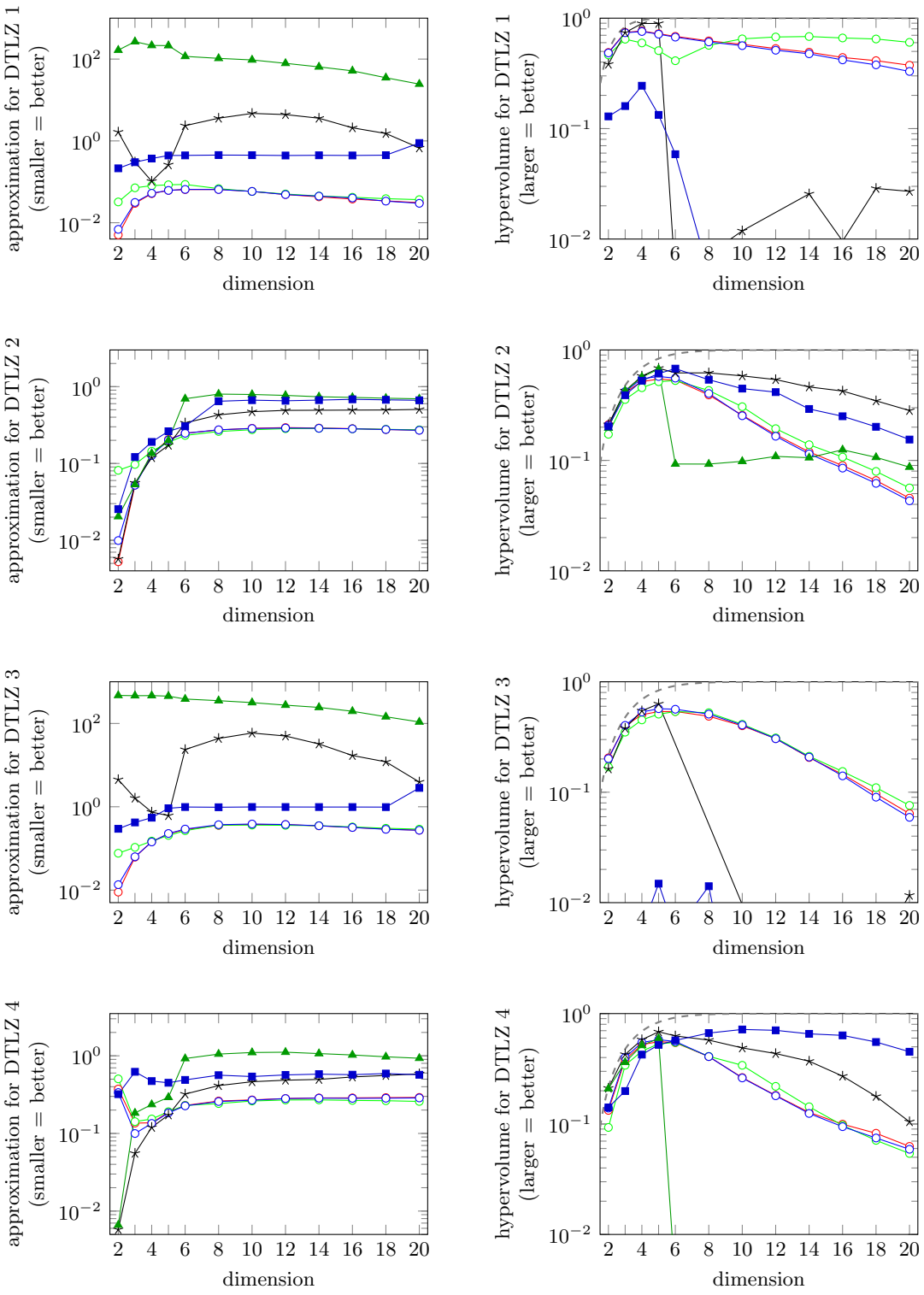

Figure 5: Comparison of the performance of AGE ($\varepsilon_{grid} = 0$: —○—, $\varepsilon_{grid} = 0.1$: —○—, $\varepsilon_{grid} = 0.01$: —○—) with IBEA (—■—), SMS-EMOA (—★—), and MO-CMA-ES (—▲—) on the DTLZ test functions with varying dimension $d = 2 \ldots 20$. The used implementations of SMS-EMOA and MO-CMA-ES use very fast approximation algorithms to compute the hypervolume to improve their running time. The figures show the averages of 100 repetitions each. We limit the computations per repetition to a maximum of 250,000 evaluations and to a maximum computation time of 24 hours. Only non-zero hypervolume values are shown. For reference, we also plot (- - -) the maximum hypervolume achievable for $\mu \to \infty$.

- On all higher-dimensional ($d \geqslant 6$) test problems, AGE achieves (statistically significantly) the best approximation. MO-CMA-ES (—▲—) and SMS-EMOA (—★—) fail at achieving good approximations on DTLZ 1 and 3. On these, IBEA (—■—) performs relatively well, but we observed runtime issues for the twenty-dimensional spaces.

- AGE achieves statistically significantly better approximations than IBEA on all functions. Compared to MO-CMA-ES (—▲—), AGE achieves statistically significantly better approximations on DTLZ1/3 (all dimensions), DTLZ2 ($d \geqslant 6$), DTLZ4 ($d \geqslant 4$). The best competitor in low dimensions ($d \leqslant 5$) is SMS-EMOA. However, in also in low dimensions AGE is either competitive or still better than the other algorithms.

- The hypervolume-based algorithms, MO-CMA-ES (—▲—), SMS-EMOA (—★—) and IBEA (—■—), sometimes achieve slightly larger hypervolumes for DTLZ 2 and DTLZ 4, but fail completely on DTLZ 1 and DTLZ 3. AGE achieves statistically significantly higher hypervolume than IBEA (—■—)and MO-CMA-ES (—▲—) on DTLZ 1 and DTLZ 3 for all dimensions. The same holds compared to SMS-EMOA (—★—) for $d \geqslant 6$.

- All aformentioned observations hold for all three variants of AGE. The performance of AGE is very similar for the different used approximative archive settings ($\varepsilon_{grid} = 0$: —○—, $\varepsilon_{grid} = 0.1$: —○—, $\varepsilon_{grid} = 0.01$: —○—). While the grid size has a significant impact on the runtime of AGE (cf. Section 6.2), it apparently has little impact on the approximation quality. Counting the number of test functions where the achieved approximation of one variant statistically significantly outperforms another variant, we can still derive a total ordering: $\varepsilon_{grid} = 0.01$ performed 75× better (48× worse, 21× insignificant) than $\varepsilon_{grid} = 0$, which performed 72× better (51× worse, 21× insignificant) than $\varepsilon_{grid} = 0.1$.

8.3. Distribution of the obtained solutions

In the following, we show and comment about the distribution of the obtained solutions, in variable and objective space.

First, we investigate the diversity in the variable space. For the DTLZ functions, the variables $x_1 ... x_{30} \in [0, 1]$ of these problems are divided in diversity related variables (x_1, ..., x_{m-1}), and convergence related variables (x_m, ..., x_{30}). In Figure 6 we show the distributions of several such variables on four DTLZ functions are shown. The data is based on 100 independent runs, from which we take from the populations the means and standard deviations of different variables. We then show the respective means and standard deviations of that data.

For all problems we can see that a wide range of values for the diversity related variables is achieved and maintained. For example, the standard deviations of x_1

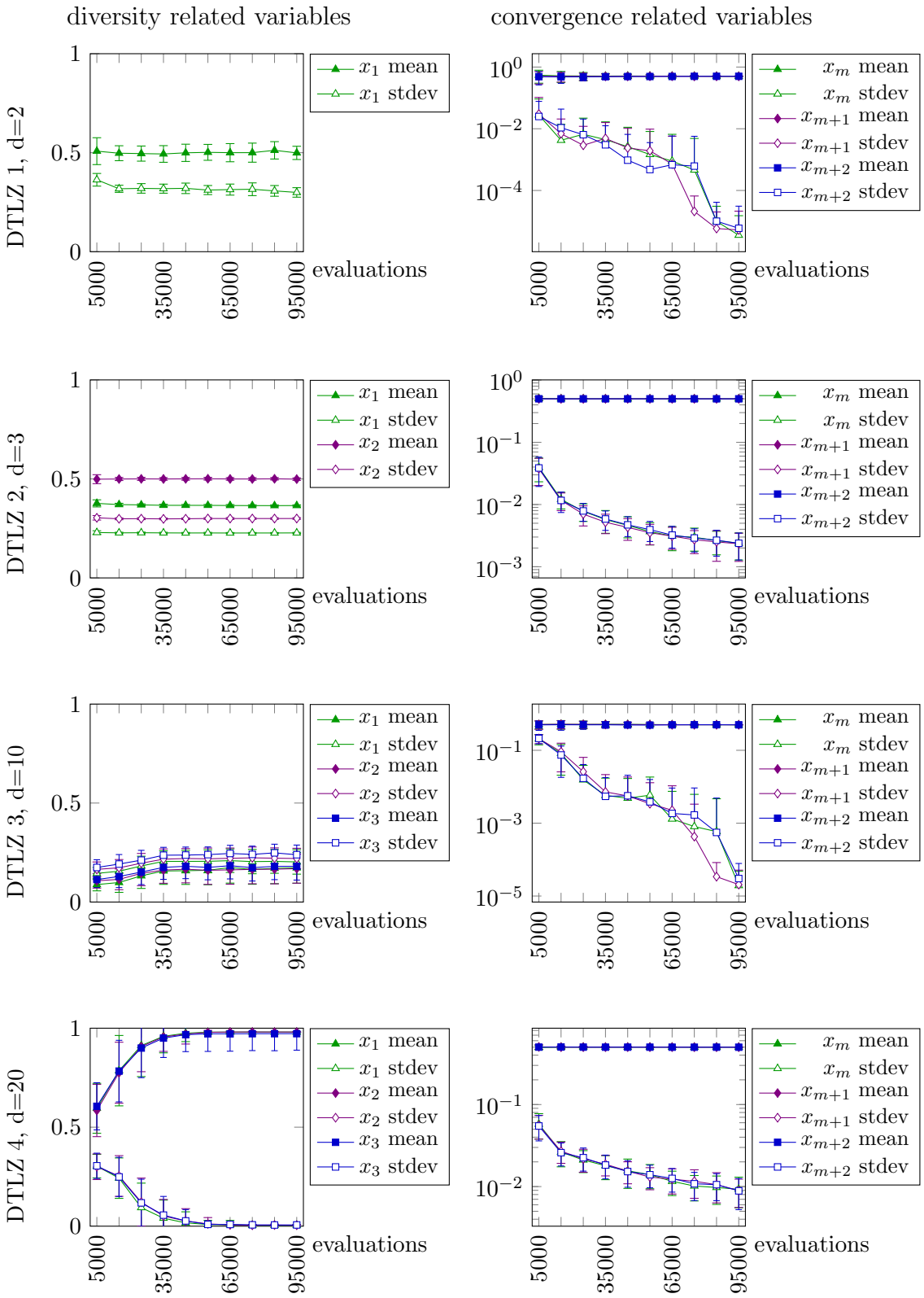

Figure 6: Distribution of different variables. We show the means and standard deviation (stdev) of the 'means of a population' for 100 independent runs of AGE with $\varepsilon_{grid} = 0.01$. The error bars show the standard deviation of the respective measure.

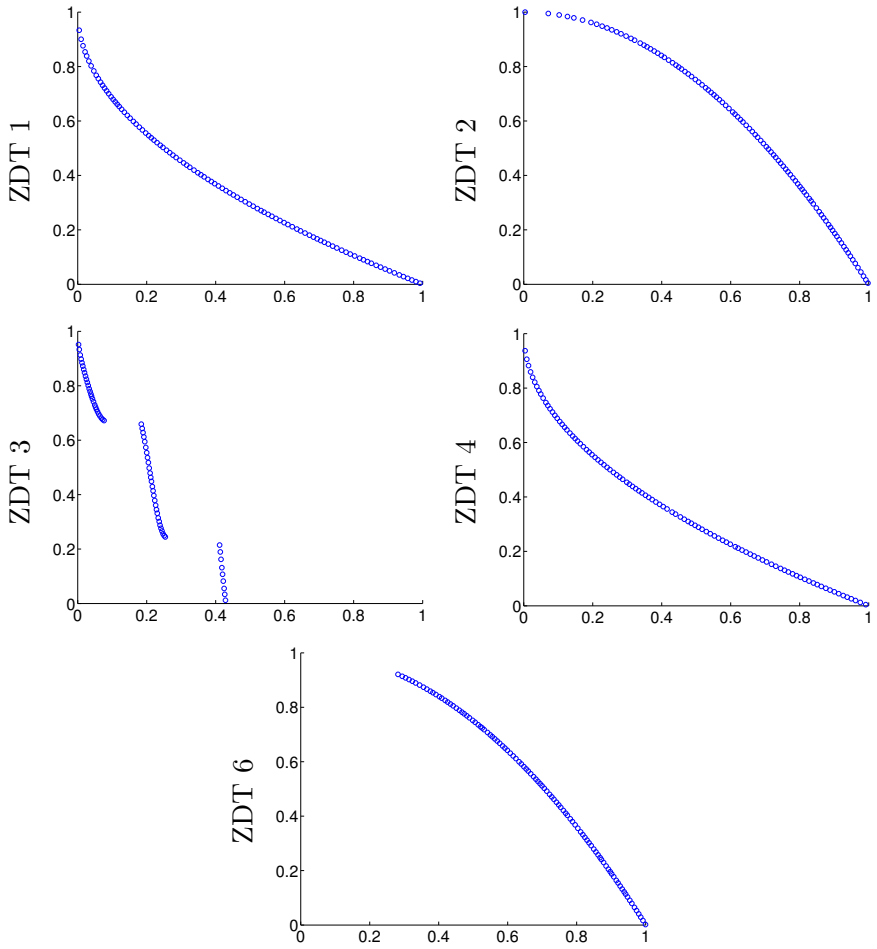

Figure 7: Distribution of the obtained solutions (examples from the ZDT family, all with $d = 2$).

within the individual populations is high (given the valid range $x_1 \in [0, 1]$) and the means of the populations are stable across multiple runs. During the runs, diversity is achieved and maintained, which is expressed in stable statistics along the x-axis.

The analysis of the convergence related variables reveals an entirely different, but expected, behaviour. The initial diversity of these variables collapses very quickly and continues to decrease as optimisation progresses. This can be seen best in the standard deviations that continue to plummet. Amongst different runs, there is little variation, as the small standard deviation of the standard deviations reveals.

Next, we show in Figures 7 and 8 randomly picked final populations in the objective space.

The solutions for the two-objective problems are very uniformly distributed. Note that "uniformity" is with respect to the additive approximation used: for example in the case of DTLZ 2, d=2 a single solution near the bottom right corner

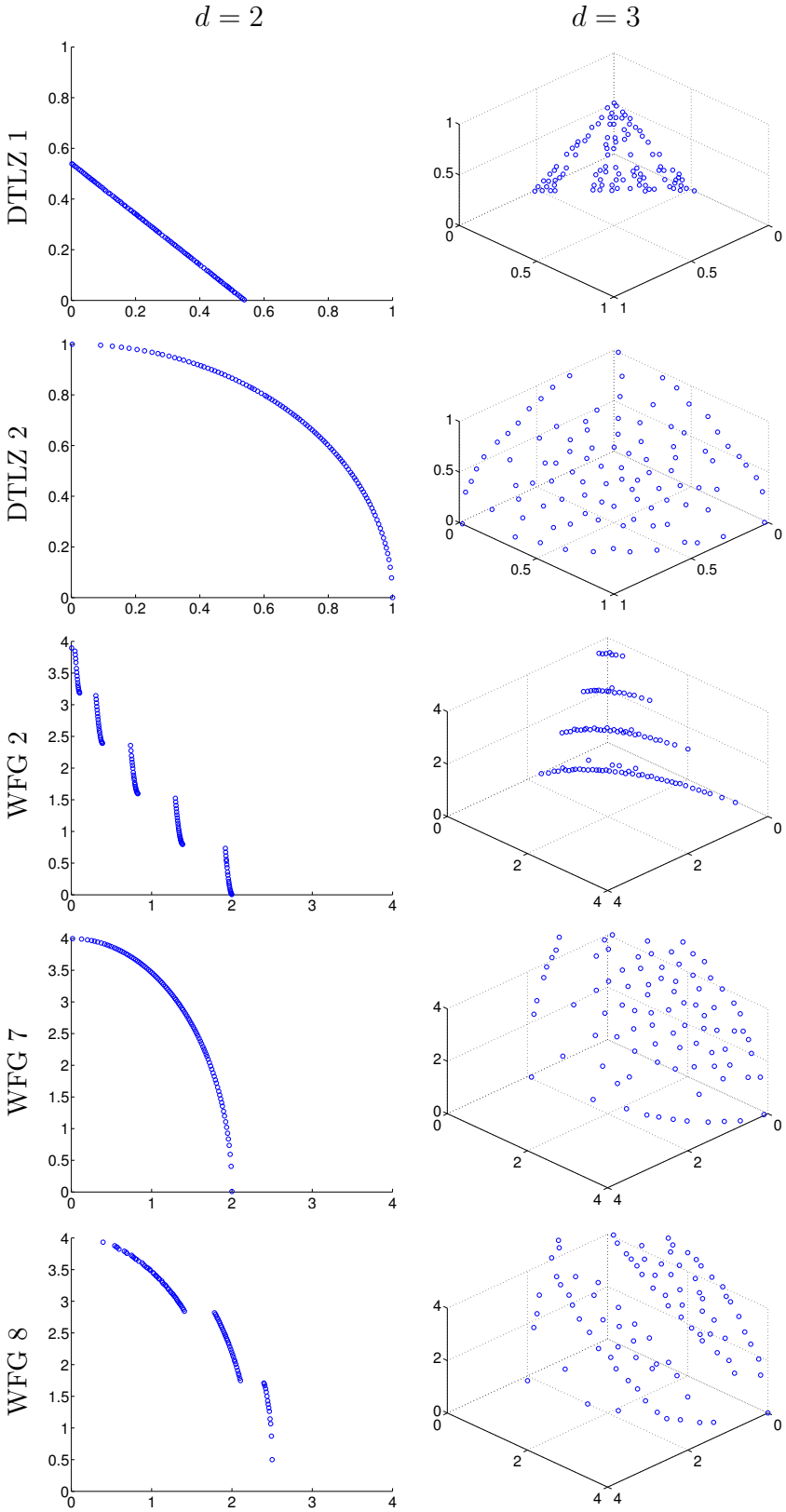

Figure 8: Distribution of the obtained solutions (examples from the DTLZ and WFG family).

can additively approximate a larger part of the true Pareto front than a single solution near the centre of the true Pareto front (near $<0.71, 0.71>$). The same effect can be observed for ZDT 3 and WFG 8.

Similarly, we can often observe quite uniform distributions of the solutions for the three-objective problems. Note that we do not know what the optimal final distributions for the different problems would be, given a fixed population size of $\mu = 100$.

9. Conclusions

Evolutionary algorithms are frequently used to solve multi-objective optimization problems. Often, it is very hard to formally define the optimization goal that current state-of-the-art approaches work with. We have presented an 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. The basic framework of AGE works with an archive which stores every non-dominated objective vector and uses this archive to judge the quality of newly produced solutions. In order to increase performance of this basic variant, we introduced an approximative archive and a parent selection scheme which increases performance for low dimensional problems.

The experimental results show that AGE efficiently solves problems with few and with many conflicting objectives.[6] Its computation time increases only linearly with the number of objectives. Given a fixed time budget, AGE outperforms current state-of-the-art approaches (including those using fast hypervolume-approximations) in terms of the desired additive approximation on standard benchmark functions for more than four objectives. On functions with two and three objectives, it lies level with the best approaches. Additionally, it also performs competitive or better regarding the covered hypervolume, depending on the function. This holds in particular for problems with many objectives, which most other algorithms have difficulties dealing with. The choice of the approximative archive (determined by the choice of ε) mainly determines the computational cost of the algorithm but has no major effect on the quality of the outcome for the investigated choices of ε. Thus we can observe runtime reductions by a factor of up to 250 without sacrificing the final solution quality.

In summary, AGE is an efficient approach to solve multi-objective problems with few and many objectives. It enables practitioners now to add objectives with only minor consequences, and to explore problems for even higher dimensions.

[6] The source code is available under `http://cs.adelaide.edu.au/~optlog/research/age.php`

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 on 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] K. Bringmann and T. Friedrich. Approximating the least hypervolume contributor: NP-hard in general, but fast in practice. *Theoretical Computer Science*, 425:104–116, 2012.

[7] K. Bringmann, T. Friedrich, F. Neumann, and M. Wagner. Approximation-guided evolutionary multi-objective optimization. In *Proc. 22nd International Joint Conferences on Artificial Intelligence (IJCAI '11)*, pages 1198–1203, 2011.

[8] K. Bringmann, T. Friedrich, C. Igel, and T. Voß. Speeding up many-objective optimization by Monte Carlo approximations. *Artificial Intelligence*, 2013+.

[9] T. C. E. Cheng, A. Janiak, and M. Y. Kovalyov. Bicriterion single machine scheduling with resource dependent processing times. *SIAM J. on Optimization*, 8:617–630, 1998.

[10] 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.

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

[12] 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.

[13] 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.

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

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

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

[17] 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.

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

[19] S. Huband, L. Barone, R. L. While, and P. Hingston. A scalable multiobjective test problem toolkit. In *Proc. Evolutionary Multi-Criterion Optimization (EMO '05)*, volume 3410 of *LNCS*, pages 280–295. Springer, 2005.

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

[21] C. Igel, T. Glasmachers, and V. Heidrich-Meisner. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.

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

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

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

[25] C. H. Papadimitriou and M. Yannakakis. Multiobjective query optimization. In *Proc. 20th ACM Symposium on Principles of Database Systems (PODS '01)*, pages 52–59, 2001.

[26] S. Vassilvitskii and M. Yannakakis. Efficiently computing succinct trade-off curves. *Theor. Comput. Sci.*, 348:334–356, 2005.

[27] M. Wagner and T. Friedrich. Efficient parent selection for approximation-guided evolutionary multi-objective optimization. In *Proc. IEEE Congress on Evolutionary Computation (CEC '13)*, pages 1846–1853. IEEE, 2013.

[28] M. Wagner and F. Neumann. A fast approximation-guided evolutionary multiobjective algorithm. In *Proc. 15th Annual Conference on Genetic and Evolutionary Computation Conference (GECCO '13)*, pages 687–694. ACM, 2013.

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

[30] 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.

[31] 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.