HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization

Johannes Bader and Eckart Zitzler
Computer Engineering and Networks Laboratory, ETH Zurich, 8092 Zurich, Switzerland
{johannes.bader,eckart.zitzler}@tik.ee.ethz.ch

TIK-Report No. 286
November 26, 2008

***Abstract*—In the field of evolutionary multi-criterion optimization, the hypervolume indicator is the only single set quality measure that is known to be strictly monotonic with regard to Pareto dominance: whenever a Pareto set approximation entirely dominates another one, then also the indicator value of the former will be better. This property is of high interest and relevance for problems involving a large number of objective functions. However, the high computational effort required for hypervolume calculation has so far prevented to fully exploit the potential of this indicator; current hypervolume-based search algorithms are limited to problems with only a few objectives.**

This paper addresses this issue and proposes a fast search algorithm that uses Monte Carlo simulation to approximate the exact hypervolume values. The main idea is that not the actual indicator values are important, but rather the rankings of solutions induced by the hypervolume indicator. In detail, we present HypE, a hypervolume estimation algorithm for multiobjective optimization, by which the accuracy of the estimates and the available computing resources can be traded off; thereby, not only many-objective problems become feasible with hypervolume-based search, but also the runtime can be flexibly adapted. Moreover, we show how the same principle can be used to statistically compare the outcomes of different multiobjective optimizers with respect to the hypervolume—so far, statistical testing has been restricted to scenarios with few objectives. The experimental results indicate that HypE is highly effective for many-objective problems in comparison to existing multiobjective evolutionary algorithms.

HypE is available for download at http://www.tik.ee.ethz.ch/sop/download/supplementary/hype/.

I. Motivation

By far most studies in the field of evolutionary multiobjective optimization (EMO) are concerned with the following set problem: find a set of solutions that as a whole represents a good approximation of the Pareto-optimal set. To this end, the original multiobjective problem consisting of

- the decision space X,
- the objective space $Z = \mathbb{R}^n$,
- a vector function $f = (f_1, f_2, \dots, f_n)$ comprising n objective functions $f_i : X \rightarrow \mathbb{R}$, which are without loss of generality to be minimized, and
- a relation $\leq$ on Z, which induces a preference relation $\preceq$ on X with $a \preceq b :\Leftrightarrow f(a) \leq f(b)$ for $a, b \in X$,

is usually transformed into a single-objective set problem [46]. The search space Ψ of the resulting set problem includes all possible Pareto set approximations[1], i.e., Ψ contains all multisets over X. The preference relation $\preceq$ can be used to define a corresponding set preference relation $\preccurlyeq$ on Ψ where

$$A \preccurlyeq B :\Leftrightarrow \forall b \in B \, \exists a \in A : \; a \preceq b \qquad (1)$$

for all Pareto set approximations $A, B \in \Psi$. In the following, we will assume that weak Pareto dominance is the underlying preference relation, cf. [46].[2]

A key question when tackling such a set problem is how to define the optimization criterion. Many multiobjective evolutionary algorithms (MOEAs) implement a combination of Pareto dominance on sets and a diversity measure based on Euclidean distance in the objective space, e.g., NSGA-II [13] and SPEA2 [42]. While these methods have been successfully employed in various biobjective optimization scenarios, they appear to have difficulties when the number of objectives increases [34]. As a consequence, researchers have tried to develop alternative concepts, and a recent trend is to use set quality measures, also denoted as quality indicators, for search—so far, they have mainly been used for performance assessment. Of particular interest in this context is the hypervolume indicator [43], [45] as it is the only quality indicator known to be fully sensitive to Pareto dominance—a property especially desirable when many objective functions are involved.

Several hypervolume-based MOEAs have been proposed meanwhile, e.g., [15], [23], [9], but their main drawback is their extreme computational overhead. Although there have been recent studies presenting improved algorithms for hypervolume calculation, currently high-dimensional problems with six or more objectives are infeasible for these MOEAs. Therefore, the question is whether and how fast hypervolume-based search algorithms can be designed that exploit the

[1]Here, a Pareto set approximation may also contain dominated solutions as well as duplicates, in contrast to the notation in [47].

[2]For reasons of simplicity, we will use the term 'u weakly dominates v' resp. 'u dominates v' independently of whether u and v are elements of X, Z, or Ψ. For instance, A weakly dominates b with $A \in \Psi$ and $b \in X$ means $A \preccurlyeq \{b\}$ and a dominates z with $a \in X$ and $z \in Z$ means $f(a) \leq z \wedge z \not\leq f(a)$.

advantages of the hypervolume indicator and at the same time are scalable with respect to the number of objectives.

A first attempt in this direction has been presented in [1]. The main idea is to estimate—by means of Monte Carlo simulation—the ranking of the individuals that is induced by the hypervolume indicator and not to determine the exact indicator values. This paper proposes an advanced method called HypE (Hypervolume Estimation Algorithm for Multiobjective Optimization) that is based on the same idea, but uses more effective fitness assignment and sampling strategies. In detail, the main contributions of this work can be summarized as follows:

1) A novel method to assign fitness values to individuals based on the hypervolume indicator—for both mating and environmental selection;
2) A hypervolume-based search algorithm (HypE) using Monte Carlo simulation that can be applied to problems with arbitrarily many objectives;
3) A statistical testing procedure that allows to compare the outcomes of different multiobjective optimizers with respect to the hypervolume indicator in many-objective scenarios.

As we will show in the follwing, the proposed search algorithm can be easily tuned regarding the available computing resources and the number of objectives involved. Thereby, it opens a new perspective on how to treat many-objective problems, and the presented concepts may also be helpful for other types of quality indicators to be integrated in the optimization process.

II. A Brief Review of Hypervolume-Related Research

The hypervolume indicator was originally proposed and employed in [44], [45] to quantitatively compare the outcomes of different MOEAs. In these two first publications, the indicator was denoted as ‘size of the space covered’, and later also other terms such as ‘hyperarea metric’ [33], ‘S-metric’ [38], ‘hypervolume indicator’ [47], and hypervolume measure [4] were used. Besides the names, there are also different definitions available, based on polytopes [45], the Lebesgue measure [27], [26], [18], or the attainment function [40].

As to hypervolume calculation, the first algorithms [39], [26] operated recursively and in each recursion step the number of objectives was decremented; the underlying principle is known as ‘hypervolume by slicing objectives’ approach [36]. While the method used in [44], [45] was never published (only the source code is publicly available [39]), Knowles independently proposed and described a similar method in [26]. A few years later, this approach was the first time studied systematically and heuristics to accelerate the computation were proposed in [36]. All these algorithms have a worst-case runtime complexity that is exponential in the number of objecives, more specifically $\mathcal{O}(N^{n-1})$ where N is the number of solutions considered [26], [36]. A different approach was presented by Fleischer [18] who mistakenly claimed a polynomial worst-case runtime complexity—While [35] showed that it is exponential in n as well. Recently, advanced algorithms for hypervolume calculation have been proposed, a dimension-sweep method [19] with a worst-case runtime complexity of $\mathcal{O}(N^{n-2}\log N)$, and a specialized algorithm related to the Klee measure problem [5] the runtime of which is in the worst case of order $\mathcal{O}(N\log N + N^{n/2})$. Furthermore, Yang and Ding [37] described an algorithm for which they claim a worst-case runtime complexity of $\mathcal{O}((n/2)^N)$. The fact that there is no exact polynomial algorithm available gave rise to the hypothesis that this problem in general is hard to solve, although the tighest known lower bound is of order $\Omega(N\log N)$ [3]. New results substantiate this hypothesis: Bringmann and Friedrich [8] have proven that the problem of computing the hypervolume is $\#P$-complete, i.e., it is expected that no polynomial algorithm exists since this would imply $NP = P$.

The complexity of the hypervolume calculation in terms of programming and computation time may explain why this measure was seldom used until 2003. However, this changed with the advent of theoretical studies that provided evidence for a unique property of this indicator [24], [47], [18]: it is the only indicator known to be strictly monotonic with respect to Pareto dominance and thereby guaranteeing that the Pareto-optimal front achieves the maximum hypervolume possible, while any worse set will be assigned a worse indicator value. This property is especially desirable with many-objective problems and since classical MOEAs have been shown to have difficulties in such scenarios [34], a trend can be observed in the literature to directly use the hypervolume indicator for search.

Knowles and Corne [26], [25] were the first to propose the integration of the hypervolume indicator into the optimization process. In particular, they described a strategy to maintain a separate, bounded archive of nondominated solutions based on the hypervolume indicator. Huband et al. [22] presented an MOEA which includes a modified SPEA2 environmental selection procedure where a hypervolume-related measure replaces the original density estimation technique. In [41], the binary hypervolume indicator was used to compare individuals and to assign corresponding fitness values within a general indicator-based evolutionary algorithm (IBEA). The first MOEA tailored specifically to the hypervolume indicator was described in [15]; it combines nondominated sorting with the hypervolume indicator and considers one offspring per generation (steady state). Similar fitness assignment strategies were later adopted in [40], [23], and also other search algorithms were proposed where the hypervolume indicator is partially used for search guidance [29], [28]. Moreover, specific aspects like hypervolume-based environmental selection [7], cf. Section III-B, and explicit gradient determination for hypervolume landscapes [16] have been investigated recently.

To date, the hypervolume indicator is one of the most popular set quality measures. For instance, almost one fourth of the papers published in the proceedings of the EMO 2007 conference [30] report on the use of or are dedicated to the hypervolume indicator. However, there are still two major drawbacks that current research acitivities try to tackle: (i) the high computation effort and (ii) the bias of the indicator in terms of user preferences. The former issue has been addressed in different ways: by automatically reducing the number of objectives [9] and by approximating the indicator values using Monte Carlo methods [17], [1], [11]. Everson et al. [17] used a basic Monte Carlo technique for performance assessment in order to estimate the values of the binary hypervolume indicator [38]; with their approach the error ratio is not polynomially bounded. In contrast, the scheme presented in [8] is a fully polynomial randomized approximation scheme where the error ratio is polynomial in the input size. The issue of statistically comparing hypervolume estimates was not addressed in these two papers. Another study [1]—a precursor study for the present paper—employed Monte Carlo simulation for fast hypervolume-based search. As to the bias issue, first proof-of-principle results have been presented in [40] that demonstrate that and how the hypervolume indicator can be adapted to different user preferences.

III. Hypervolume-Based Fitness Assignment

When considering the hypervolume indicator as the objective function of the underlying set problem, the main question is how to make use of this measure within a multiobjective optimizer to guide the search. In the context of an MOEA, this refers to selection and one can distinguish two situations:

1) The selection of solutions to be varied (mating selection).
2) The selection of solutions to be kept in memory (environmental selection).

Since the indicator as such operates on (multi)sets of solutions, while selection considers single solutions, a strategy for assigning fitness values to solutions is required. Most hypervolume-based algorithms first perform a nondominated sorting and then rank solutions within a particular front according to the hypervolume loss that results from the removal of a specific solution [25], [15], [23], [1]. In the following, we propose a generalized fitness assignment strategy that takes into account the entire objective space weakly dominated by a population. We will first provide a basic scheme for mating selection and then present an extension for environmental selection. Afterwards, we briefly discuss how the fitness values can be computed exactly using a slightly modified hypervolume calculation algorithm.

A. Basic Scheme for Mating Selection

To begin with, we formally define the hypervolume indicator as a basis for the following discussions. Different definitions can be found in the literature, and we here use the one from [46] which draws upon the Lebesgue measure as proposed in [27] and considers a reference set of objective vectors.

Definition III.1. *Let $A \in \Psi$ be a Pareto set approximation and $R \subset Z$ be a reference set of mutually nondominating objective vectors. Then the hypervolume indicator I_H can be defined as*

$$I_H(A, R) := \lambda(H(A, R)) \tag{2}$$

where

$$H(A, R) := \{z \in Z\,;\, \exists a \in A\, \exists r \in R : f(a) \leq z \leq r\} \tag{3}$$

and λ is the Lebesgue measure with $\lambda(H(A, R)) = \int_{\mathbb{R}^n} \mathbf{1}_{H(A,R)}(z)dz$ and $\mathbf{1}_{H(A,R)}$ being the characteristic function of $H(A, R)$.

The set $H(A, R)$ denotes the set of objective vectors that are enclosed by the front $f(A)$ given by A and the reference set R.

The subspace $H(A, R)$ of Z can be further split into partitions $H(S, A, R)$, each associated with a specific subset $S \subseteq A$:

$$H(S, A, R) := [\bigcap_{s \in S} H(\{s\}, R)] \setminus [\bigcup_{a \in A \setminus S} H(\{a\}, R)] \tag{4}$$

The set $H(S, A, R) \subseteq Z$ represents the portion of the objective space that is jointly weakly dominated by the solutions in S and not weakly dominated by any other solution in A. It holds

$$\dot{\bigcup_{S \subseteq A}} H(S, A, R) = H(A, R) \tag{5}$$

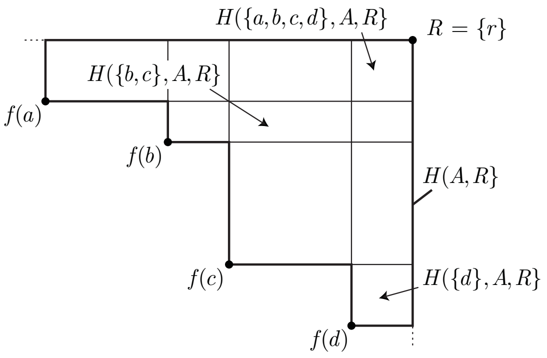
which is illustrated in Fig. 1(a). That the partitions are disjoint can be easily shown: Assume that there are two non-identical subsets S_1, S_2 of A for which $H(S_1, A, R) \cap H(S_2, A, R) \neq \emptyset$; since the sets are not identical, there exists with loss of generality an element $a \in S_1$ which is not contained in S_2; from the above definition follows that $H(\{a\}, R) \supseteq H(S_1, A, R)$ and therefore $H(\{a\}, R) \cap H(S_2, A, R) \neq \emptyset$; the latter statement leads to a contradiction since $H(\{a\}, R)$ cannot be part of $H(S_2, A, R)$ when $a \notin S_2$.

In practice, it is infeasible to determine all distinct $H(S, A, R)$ due to combinatorial explosion. Instead, we will consider a more compact splitting of the dominated objective space that refers to single solutions:

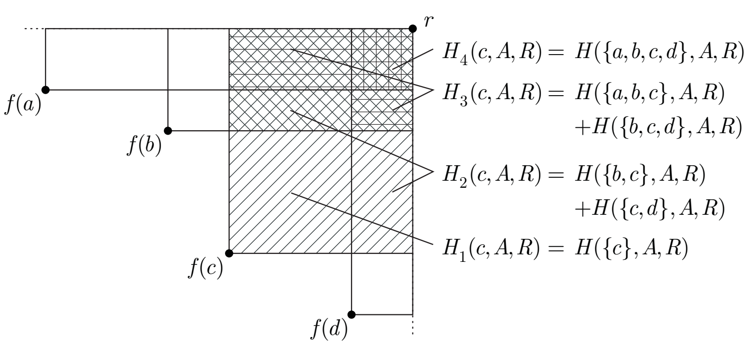
$$H_i(a, A, R) := \bigcup_{\substack{S \subseteq A \\ a \in S \\ |S| = i}} H(S, A, R) \tag{6}$$

According to this definition, $H_i(a, A, R)$ stands for the portion of the objective space that is jointly and solely weakly dominated by a and any $i - 1$ further solutions from A, see Fig. 1(b). Note that the sets $H_1(a, A, R), H_2(a, A, R), \ldots, H_{|A|}(a, A, R)$ are disjoint for a given $a \in A$, i.e., $\dot{\bigcup}_{1 \leq i \leq |A|} H_i(a, A, R) = H(\{a\}, R)$, while the sets $H_i(a, A, R)$ and $H_i(b, A, R)$ may be overlapping for fixed i and different solutions $a, b \in A$. This slightly different notion has reduced the number of subspaces to be considered from $2^{|A|}$ for $H(S, A, R)$ to $|A|^2$ for $H_i(a, A, R)$.

Now, given an arbitrary population $P \in \Psi$ one obtains for each solution a contained in P a vector $(\lambda(H_1(a, P, R)), \lambda(H_2(a, P, R)), \ldots, \lambda(H_{|P|}(a, P, R)))$ of hypervolume contributions. These vectors can be used to assign fitness values to solutions; Subsection III-C describes how the corresponding values $\lambda(H_i(a, A, R))$ can be computed. While most hypervolume-based search algorithms only take the first components, i.e., $\lambda(H_1(a, P, R))$, into



(a) The relationship between $H(A,R)$ and $H(S,A,R)$



r
$H_4(c,A,R) = H(\{a,b,c,d\},A,R)$
$H_3(c,A,R) = H(\{a,b,c\},A,R)$
$+H(\{b,c,d\},A,R)$
$H_2(c,A,R) = H(\{b,c\},A,R)$
$+H(\{c,d\},A,R)$
$H_1(c,A,R) = H(\{c\},A,R)$
$f(a)$
$f(b)$
$f(c)$
$f(d)$

(b) The relationship between $H(S,A,R)$ and $H_i(a,A,R)$

Figure 1. Illustration of the notions of $H(A,R)$, $H(S,A,R)$, and $H_i(a,A,R)$ in the objective space for a Pareto set approximation $A = \{a,b,c,d\}$ and reference set $R = \{r\}$.

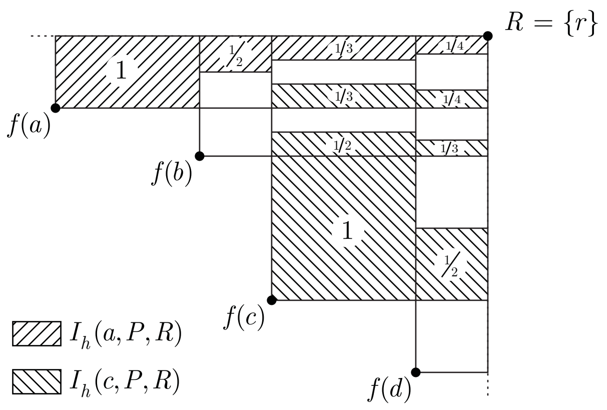


Figure 2. Illustration of the basic fitness assignment scheme where the fitness F_a of a solution a is set to $F_a = I_h(a,P,R)$.

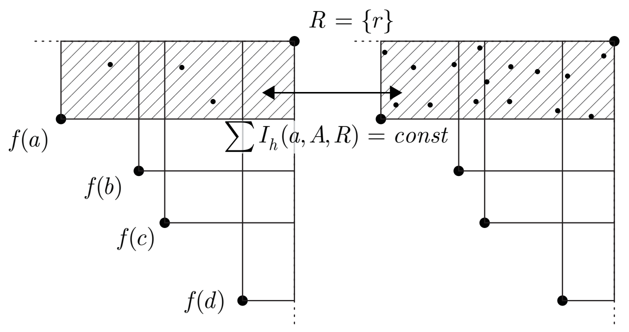


Figure 3. Shows for an example population the selection probabilities for the population members (left). The sizes of the points correlate with the corresponding selection probabilities. As one can see on the right, the overall selection probability for the shaded area does not change when dominated solutions are added to the population.

account, we here propose the following scheme to aggregate the hypervolume contributions into a single scalar value.

Definition III.2. *Let $A \in \Psi$ and $R \subset Z$. Then the function I_h with*

$$I_h(a,A,R) := \sum_{i=1}^{|A|} \frac{1}{i} \lambda(H_i(a,A,R)) \tag{7}$$

gives for each solution $a \in A$ the hypervolume that can be attributed to a with regard to the overall hypervolume $I_H(A,R)$.

The motivation behind this definition is simple: the hypervolume contribution of each partition $H(S,A,R)$ is shared equally among the dominating solutions $s \in S$. That means the portion of Z solely weakly dominated by a specific solution a is fully attributed to a, the portion of Z that a weakly dominates together with another solution b is attributed half to a and so forth—the principle is illustrated in Fig. 2. Thereby, the overall hypervolume is distributed among the distinct solutions according to their hypervolume contributions as the following theorem shows (the proof can be found in the appendix). Note that this scheme does not require that the solutions of the considered Pareto set approximation A are mutually non-dominating; it applies to nondominated and dominated solutions alike.

Theorem III.3. *Let $A \in \Psi$ and $R \subset Z$. Then it holds*

$$I_H(A,R) = \sum_{a \in A} I_h(a,A,R) \tag{8}$$

This aggregation method has some desirable properties that make it well suited to mating selection where the fitness F_a of a population member $a \in P$ is $F_a = I_h(a,P,R)$ and the corresponding selection probability p_a equals $F_a / I_H(P,R)$. As Fig. 3 demonstrates, the accumulated selection probability remains the same for any subspace $H(\{a\},R)$ with $a \in P$, independently of how many individuals $b \in P$ are mapped to $H(\{a\},R)$ and how the individuals are located within $H(\{a\},R)$. This can be formally stated in the next theorem; the proof can again be found in the appendix.

Theorem III.4. *Let $A \in \Psi$ and $R \subset Z$. For every $a \in A$ and all multisets $B_1, B_2 \in \Psi$ with $\{a\} \preccurlyeq B_1$ and $\{a\} \preccurlyeq B_2$ holds*

$$\sum_{b_1 \in \{a\} \cup B_1} I_h(b_1, \{a\} \cup B_1, R) = \sum_{b_2 \in \{a\} \cup B_2} I_h(b_2, \{a\} \cup B_2, R) \tag{9}$$

Since the selection probability per subspace is constant as long as the overall hypervolume value does not change, adding dominated solutions to the population leads to a redistribution of the selection probabilities and thereby implements a natural

Table I
COMPARISON OF THREE FITNESS ASSIGNMENT SCHEMES: (1) CONSTANT FITNESS, (2) NONDOMINATED SORTING PLUS $\lambda(H_1(a,P,R))$, AND (3) THE PROPOSED METHOD. EACH VALUE GIVES THE PERCENTAGE OF CASES WHERE THE METHOD ASSOCIATED WITH THAT ROW YIELDS A HIGHER HYPERVOLUME VALUE THAN THE METHOD ASSOCIATED WITH THE CORRESPONDING COLUMN.

versus	constant (1)	standard (2)	new (3)
constant (1)	-	44%	28%
standard (2)	56%	-	37%
new (3)	72%	63%	-

niching mechanism. Another advantage of this fitness assignment scheme is that it takes all hypervolume contributions $H_i(a,P,R)$ for $1 \leq i \leq |P|$ into account. As will be discussed in Section IV, this allows to more accurately estimate the ranking of the individuals according to their fitness values when using Monte Carlo simulation.

In order to study the usefulness of this fitness assignment strategy, we consider the following experiment. A standard evolutionary algorithm implementing pure nondominated sorting fitness is applied to a selected test function (biobjective WFG1 [21] using the setting as described in Section VI) and run for 100 generations. Then, mating selection is carried out on the resulting population, i.e., the individuals are reevaluated using the fitness scheme under consideration and offspring is generated employing binary tournament selection with replacement and corresponding variation operators. The hypervolume of the (multi)set of offspring is taken as an indicator for the effectiveness of the fitness assignment scheme. By comparing the resulting hypervolume values for different strategies (constant fitness leading to uniform selection, nondominated sorting plus $\lambda(H_1(a,P,R))$, and the proposed fitness according to Def. III.2) and for 100 repetitions of this experiment, we can investigate the influence of the fitness assignment strategy on the mating selection process.

The Quade test, a modification of Friedman's test which has more power when comparing few treatments [10], reveals that there are significant differences in the quality of the generated offspring populations at a signficance level of 0.01 (test statistics: $T_3 = 12.2$). Performing post-hoc pairwise comparisons following [10] using the same significance level as in the Quade test provides evidence that the proposed fitness strategy can be advantageous over the other two strategies, cf. Table I; in the considered setting, the hypervolume values achieved are significantly better. Comparing the standard hypervolume-based fitness with constant fitness, the former outperforms the latter significantly. Nevertheless, also the required computation resources need to be taken into account. That means in practice that the advantage over uniform selection may diminish when fitness computation becomes expensive. This aspect will be investigated in Section VI.

Next, we will extend and generalize the fitness assignment scheme with regard to the environmental selection phase.

B. Extended Scheme for Environmental Selection

In the context of hypervolume-based multiobjective search, environmental selection can be formulated in terms of the hypervolume subset selection problem (HSSP).

Definition III.5. *Let $A \in \Psi$, $R \subset Z$, and $k \in \{0, 1, \ldots, |A|\}$. The hypervolume subset selection problem (HSSP) is defined as the problem of finding a subset $A' \subseteq A$ with $|A'| = |A| - k$ such that the overall hypervolume loss is minimum, i.e.,*

$$I_H(A', R) = \max_{\substack{A'' \subseteq A \\ |A''| = |A| - k}} I_H(A'', R) \qquad (10)$$

Here, we assume that parents and offspring have been merged into a single population P which then needs to be truncated by removing k solutions. Since dominated solutions in the population do not affect the overall hypervolume, they can be deleted first; therefore, we assume in the following that all solutions in P are incomparable[3] or indifferent[4] to each other.

If $k = 1$, then HSSP can be solved exactly by removing that solution a from the population P with the lowest value $\lambda(H_1(a,P,R))$; this is the principle implemented in most hypervolume-based MOEAs which consider one offspring per generation, e.g., [25], [15], [23]. However, it has been recently shown that exchanging only one solution in the population like in steady state MOEAs ($k = 1$) may lead to premature convergence to a local optimum in the hypervolume landscape [46]. This problem can be avoided when generating at least as many offspring as parents are available, i.e., $k \geq |P|/2$.

For arbitrary values of k, dynamic programming can be used to solve HSSP in a biobjective setting; in the presence of three or more objectives, it is an open problem whether HSSP becomes NP-hard. In practice, a greedy heuristic is employed to obtain an approximation [41], [9]: all solutions are evaluated with respect to their usefulness and the l least important solutions are removed where l is a prespecified parameter. Most popular are the following two approaches:

1) **Iterative ($l = 1$):** The greedy heuristics is applied k times in a row; each time, the worst solution is removed and afterwards the remaining solutions are re-evaluated.
2) **One shot ($l = k$):** The greedy heuristics is only applied once; the solutions are evaluated and the k worst solutions are removed in one step.

Best results are usually obtained using the iterative approach, as the re-evaluation increases the quality of the generated approximation. In contrast, the one-shot approach substantially reduces the computation effort, but the quality of the resulting subset is lower. In the context of density-based MOEAs, the first approach is for instance used in SPEA2, while the second is employed in NSGA-II.

The key issue with respect to the above greedy strategy is how to evaluate the usefulness of a solution. The scheme

[3]Two solutions $a, b \in X$ are called *incomparable* if and only if neither weakly dominates the other one, i.e., $a \not\preceq b$ and $b \not\preceq a$

[4]Two solutions $a, b \in X$ are called *indifferent* if and only if both weakly dominate other one, i.e., $a \preceq b$ and $b \preceq a$

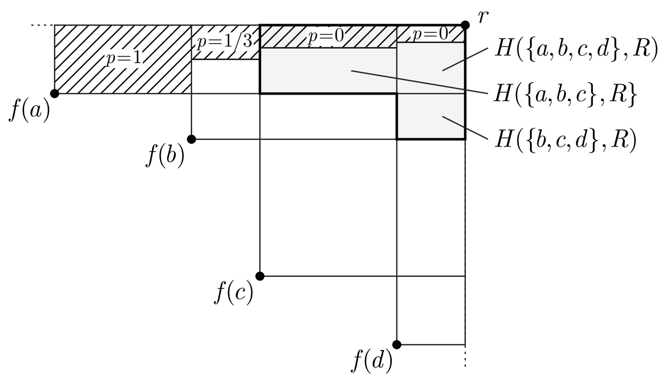


Figure 4. The figure is based on the previous example with $A = \{a,b,c,d\}, R = \{r\}$ and shows (i) which portion of the objective space remains dominated if any two solutions are removed from A (shaded area), and (ii) the probabilities p that a particular area that can be attributed to $a \in A$ is lost if a is removed from A together with any other solution in A.

presented in Def. III.2 has the drawback that portions of the objective space are taken into account that for sure will not change. Consider, for instance, a population with four solutions as shown in Fig. 4; when two solutions need to be removed ($k = 2$), then the subspaces $H(\{a,b,c\},P,R)$, $H(\{b,c,d\},P,R)$, and $H(\{a,b,c,d\},P,R)$ remain weakly dominated independently of which solutions are deleted. This observation led to the idea of considering the expected loss in hypervolume that can be attributed to a particular solution when exactly k solutions are removed. In detail, we consider for each $a \in P$ the average hypervolume loss over all subsets $S \subseteq P$ that contain a and $k-1$ further solutions; this value can be easily computed by slightly extending the scheme from Def. III.2 as follows.

Definition III.6. *Let $A \in \Psi$, $R \subset Z$, and $k \in \{0,1,\ldots,|A|\}$. Then the function I_h^k with*

$$I_h^k(a,A,R) := \frac{1}{|\mathcal{S}|}\sum_{S\in\mathcal{S}}\left[\sum_{\substack{T\subseteq S\\ a\in T}}\frac{1}{|T|}\lambda\big(H(T,A,R)\big)\right] \tag{11}$$

where $\mathcal{S} = \{S \subseteq A\,;\, a \in S \wedge |S| = k\}$ contains all subsets of A that include a and have cardinality k gives for each solution $a \in A$ the expected hypervolume loss that can be attributed to a when a and $k-1$ uniformly randomly chosen solutions from A are removed from A.

Notice that $I_h^1(a,A,R) = \lambda(H_1(a,A,R))$ and $I_h^{|A|}(a,A,R) = I_h(a,A,R)$, i.e., this modified scheme can be regarded as a generalization of the scheme presented in Def. III.2 and the commonly used fitness assignment strategy for hypervolume-based search [25], [15], [23], [1]. The next theorem shows how to calculate $I_h^k(a,A,R)$ without averaging over all subsets $S \in \mathcal{S}$; the proof can be found in the appendix.

Theorem III.7. *Let $A \in \Psi$, $R \subset Z$, and $k \in \{0,1,\ldots,|A|\}$. Then it holds*

$$I_h^k(a,A,R) = \sum_{i=1}^{k}\frac{\alpha_i}{i}\lambda(H_i(a,A,R)) \tag{12}$$

where

$$\alpha_i := \prod_{j=1}^{i-1}\frac{k-j}{|A|-j} \tag{13}$$

Next, we will study the effectiveness of $I_h^k(a,A,R)$ for approximating the optimal HSSP solution. To this end, we assume that for the iterative greedy strategy ($l = 1$) in the first round the values $I_h^k(a,A,R)$ are considered, in the second round the values $I_h^{k-1}(a,A,R)$, and so forth; each time an individual assigned the lowest value is selected for removal. For the one-step greedy method ($l = k$), only the $I_h^k(a,A,R)$ values are considered.

Table II provides a comparison of the different techniques for $100,000$ randomly chosen Pareto set approximations $A \in \Psi$ containing ten incomparable solutions, where the ten points are randomly distributed on a three dimensional unit simplex, i.e., we consider a three objective scenario. The parameter k was set to 5, so that half of the solutions needed to be removed. The relatively small numbers were chosen to allow to compute the optimal subsets by enumeration. Thereby, the maximum hypervolume values achievable could be determined.

The comparison reveals that the new fitness assignment scheme is in the considered scenario more effective in approximating HSSP than the standard scheme. The mean relative distance (see Table II) to the optimal solution is about 60% smaller than the distance achieved using I_h^1 in the iterative case and about 44% smaller in the one shot case. Furthermore, the optimum was found much more often in comparison to the standard fitness: 34% more often for the iterative approach and 497% in the one shot scenario.

Finally, note that the proposed evaluation function I_h^k will be combined with nondominated sorting for environmental selection, cf. Section V, similarly to [15], [23], [9], [40], [1]. One reason is computation time: with nondominated sorting the worst dominated solutions can be removed quickly without invoking the hypervolume calculation algorithm; this advantage mainly applies to low-dimensional problems and to the early stage of the search process. Another reason is that the full benefits of the scheme proposed in Def. III.6 can be

Table II
COMPARISON OF GREEDY STRATEGIES FOR THE HSSP (ITERATIVE VS. ONE SHOT) USING THE NEW (I_h^k) AND THE STANDARD HYPERVOLUME FITNESS (I_h^1); AS A REFERENCE, PURELY RANDOM DELETIONS ARE CONSIDERED AS WELL. THE FIRST COLUMN GIVES THE PORTION OF CASES AN OPTIMAL SUBSET WAS GENERATED; THE SECOND COLUMN PROVIDES THE AVERAGE DIFFERENCE IN HYPERVOLUME BETWEEN OPTIMAL AND GENERATED SUBSET. THE LAST TWO COLUMNS REFLECT THE DIRECT COMPARISONS BETWEEN THE TWO FITNESS SCHEMES FOR EACH GREEDY APPROACH (ITERATIVE, ONE SHOT) SEPARATELY; THEY GIVE THE PERCENTAGES OF CASES WHERE THE CORRESPONDING METHOD WAS BETTER THAN OR EQUAL TO THE OTHER ONE.

greedy strategy	optimum found	distance	better	equal
iterative with I_h^k	59.8 %	1.09 10^{-3}	30.3 %	66.5 %
iterative with I_h^1	44.5 %	2.59 10^{-3}	3.17 %	66.5 %
one shot with I_h^k	16.9 %	39.3 10^{-3}	65.2 %	23.7 %
one shot with I_h^1	3.4 %	69.6 10^{-3}	11.1 %	23.7 %
uniformly random	0.381 %	257 10^{-3}		

exploited when the Pareto set approximation A under consideration only contains incomparable and indifferent solutions; otherwise, it cannot be guaranteed that nondominated solutions are preferred over dominated ones.

C. Exact Calculation of I_h^k

In this subsection, we tackle the question of how to calculate the fitness values for a given population $P \in \Psi$. We present an algorithm that determines the values $I_h^k(a, P, R)$ for all elements $a \in P$ and a fixed k—in the case of mating selection k equals $|P|$, in the case of environmental selection k gives the number of solutions to be removed from P. It operates according to the 'hypervolume by slicing objectives' principle [39], [26], [36], but differs from existing methods in that it allows (i) to consider a set R of reference points and (ii) to compute all fitness values, e.g., the $I_h^1(a, P, R)$ values for $k = 1$, in parallel for any number of objectives instead of subsequently as in [4]. Although it looks at all partitions $H(S, P, R)$ with $S \subseteq P$ explicitly, the worst-case runtime complexity is not affected by this; it is of order $\mathcal{O}(|P|^n + n|P| \log |P|)$ assuming that sorting of the solutions in all dimensions is carried out as a preprocessing step. Clearly, this is only feasible for a low number of objectives, and the next section discusses how the fitness values can be estimated using Monte Carlo methods.

Details of the procedure are given by Algorithms 1 and 2. Algorithm 1 just provides the top level call to the recursive function *doSlicing* and returns a fitness assignment $\mathcal{F}$, a multiset containing for each $a \in P$ a corresponding pair (a, v) where v is the fitness value. Note that n at Line 3 denotes the number of objectives. Algorithm 2 recursively cuts the dominated space into hyperrectangles and returns a (partial) fitness assignment $\mathcal{F}'$. At each recursion level, a scan is performed along a specific objective—given by i—with u^* representing the current scan position. The vector $(z_1, \ldots, z_n)$ contains for all dimensions the scan positions, and at each invocation of *doSlicing* solutions (more precisely: their objective vectors) and reference points are filtered out according to these scan positions (Lines 3 and 4) where also dominated solutions may be selected in contrast to [39], [26], [36]. Furthermore, the partial volume V is updated before recursively invoking Algorithm 2 based on the distance to the next scan position. At the lowest recursion level ($i = 0$), the variable V gives the hypervolume of the partition $H(A, P, R)$, i.e., $V = \lambda(H(A, P, R))$ where A stands for the remaining solutions fulfilling the bounds given by the vector $(z_1, \ldots, z_n)$—UP contains the objective vectors corresponding to A, cf. Line 3. Since the fitness according to Def. III.6 is additive with respect to the partitions, for each $a \in A$ the partial fitness value v can be updated by adding $\frac{\alpha_{|UP|}}{|UP|}V$. Note that the population is a multiset, i.e., it may contain indifferent solutions or even duplicates; therefore, all the other sets in the algorithms are multisets.

The following example illustrates the working principle of the hypervolume computation.

Example III.8. *Consider the three-objective scenario depicted in Fig. 5 where the population contains four solutions a, b, c, d the objective vectors of which are $f(a) = (-10, -3, -2), f(b) = (-8, -1, -8), f(c) = (-6, -8, -10), f(d) = (-4, -5, -11)$ and the reference set includes two points $r = (-2, 0, 0), s = (0, -3, -4)$. Furthermore, let the parameter k be 2.*

In the first call of doSlicing, it holds $i = 3$ and U contains all objective vectors associated with the population and all reference points. The following representation shows U with

Algorithm 1 Hypervolume-based Fitness Value Computation

```
Require: population $P \in \Psi$, reference set $R \subseteq Z$, fitness
    parameter $k \in \mathbb{N}$
1: procedure computeHypervolume($P$, $R$, $k$)
2:     $\mathcal{F} \leftarrow \bigcup_{a \in P} \{(a, 0)\}$
3:     return doSlicing($\mathcal{F}$,$R$,$k$,$n$,1,$(\infty, \infty, \ldots, \infty)$);
4: end procedure
```

Algorithm 2 Recursive Objective Space Partitioning

```
Require: current fitness assignment $\mathcal{F}$, reference set $R \subseteq Z$,
    fitness parameter $k \in \mathbb{N}$, recursion level $i$, partial volume
    $V \in \mathbb{R}$, scan positions $(z_1, \ldots, z_n) \in \mathbb{R}^n$
1: procedure doSlicing($\mathcal{F}$, $R$, $k$, $i$, $V$, $(z_1, \ldots, z_n)$)
2:     /* filter out relevant solutions and reference points */
3:     $UP \leftarrow \bigcup_{(a,v) \in \mathcal{F},\ \forall i < j \le n:\ f_j(a) \le z_j} \{f(a)\}$
4:     $UR \leftarrow \bigcup_{(r_1,\ldots,r_n) \in R,\ \forall i < j \le n:\ r_j \ge z_j} \{(r_1, \ldots, r_n)\}$
5:     if $i = 0 \wedge UR \neq \emptyset$ then
6:         /* end of recursion reached */
7:         $\alpha \leftarrow \prod_{j=1}^{|UP|-1} (k - j)/(|\mathcal{F}| - j)$
8:         /* update hypervolumes of filtered solutions */
9:         $\mathcal{F}' \leftarrow \emptyset$
10:        for all $(a, v) \in \mathcal{F}$ do
11:            if $\forall 1 \le j \le n : f_j(a) \le z_j$ then
12:                $\mathcal{F}' \leftarrow \mathcal{F}' \cup \{(a, v + \frac{\alpha}{|UP|} V)\}$
13:            else
14:                $\mathcal{F}' \leftarrow \mathcal{F}' \cup \{(a, v)\}$
15:            end if
16:        end for
17:    else if $i > 0$ then
18:        /* recursion continues */
19:        $\mathcal{F}' \leftarrow \mathcal{F}$
20:        $U \leftarrow UP \cup UR$
21:        /* scan current dimension in ascending order */
22:        while $U \neq \emptyset$ do
23:            $u^* \leftarrow \min_{(u_1,\ldots,u_n) \in U} u_i$
24:            $U' \leftarrow \{(u_1, \ldots, u_n) \in U \mid u_i > u^*\}$
25:            if $U' \neq \emptyset$ then
26:                $V' = V \cdot \left((\min_{(u'_1,\ldots,u'_n) \in U'} u'_i) - u^*\right)$
27:                $\mathcal{F}' \leftarrow$ doSlicing($\mathcal{F}'$, $R$, $k$, $i - 1$, $V'$,
28:                        $(z_1, \ldots, z_{i-1}, u^*, z_{i+1}, \ldots, z_n)$ )
29:            end if
30:            $U = U'$
31:        end while
32:    end if
33:    return $\mathcal{F}'$
34: end procedure
```

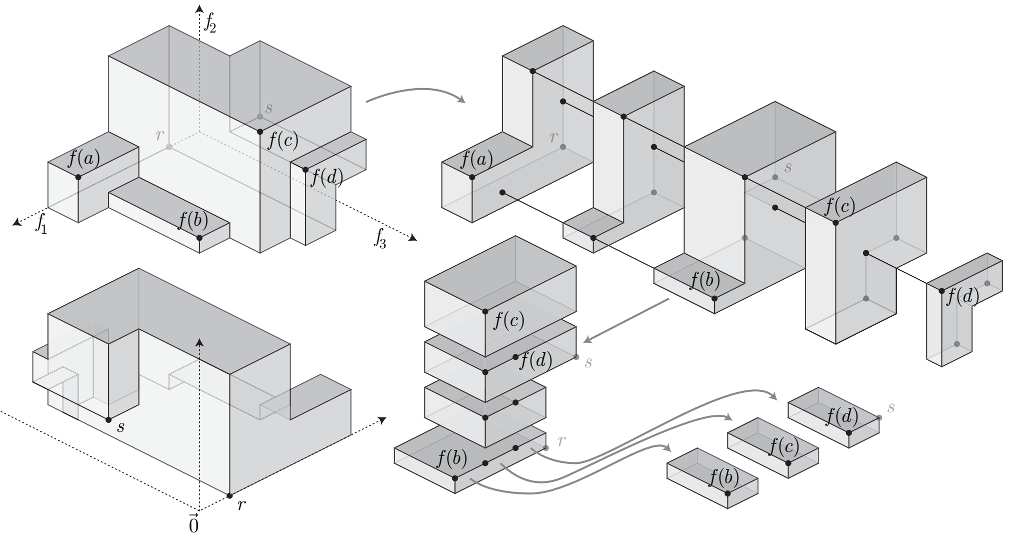


Figure 5. Illustration of the principle underlying Algorithm 2 where one looks from $(-\infty, -\infty, -\infty)$ on the front except for the lower left picture where one looks from $(\infty, -\infty, \infty)$ to the origin. First, the dominated polytope is cut along the third dimension leading to five slices, which are again cut along the second dimension and finally along the first dimension. In contrast to existing 'Hypervolume by Slicing Objectives' algorithms, also dominated points are carried along.

its elements sorted in ascending order according to their third vector components:

$$U = \begin{array}{rl} f(d): & (-4,-5,-11)\downarrow \\ f(c): & (-6,-8,-10) \\ f(b): & (-8,-1,-8) \\ s: & (-0,-3,-4) \\ f(a): & (-10,-3,-2) \\ r: & (-2,0,0) \end{array} \quad (14)$$

Hence, in the first two iterations of the loop beginning at Line 22 the variable u^* is assigned to $f_3(d) = -11$ resp. $u^* = f_3(c) = -10$. Within the third iteration, U is reduced to $\{f(a), f(b), r, s\}$ which yields $u^* = f_3(b) = -8$ and in turn $V' = 1 \cdot (-4 - (-8)) = 4$ with the current vector of scan positions being $(z_1, z_2, z_3) = (\infty, \infty, -8)$; these values are passed to the next recursion level $i = 2$ where U is initialized at Line 20 as follows (this time sorted according to the second dimension):

$$U = \begin{array}{rl} f(c): & (-6,-8,-10)\downarrow \\ f(d): & (-4,-5,-11) \\ s: & (0,-3,-4) \\ f(b): & (-8,-1,-8) \\ r: & (-2,0,0) \end{array} \quad (15)$$

Now, after three iterations of the loop at Line 22 with $u^* = f_2(c) = -8$, $u^* = f_2(d) = -5$, and $u^* = s_2 = -3$, respectively, U is reduced in the fourth iteration to $\{f(b), r\}$ and u^* is set to $f_2(b) = -1$. As a result, $V' = 1 \cdot 4 \cdot (0 - (-1)) = 4$ and $(z_1, z_2, z_3) = (\infty, -1, -8)$ which are the parameters for the next recursive invocation of doSlicing where U is set to:

$$U = \begin{array}{rl} f(b): & (-8,-1,-8)\downarrow \\ f(c): & (-6,-8,-10) \\ f(d): & (-4,-5,-11) \\ r: & (-2,0,0) \end{array} \quad (16)$$

At this recursion level with $i = 1$, in the second iteration it holds $u^* = f_1(c) = -6$ and $V' = 1 \cdot 4 \cdot 1 \cdot (-4 - (-6)) = 8$. When calling doSlicing at this stage, the last recursion level is reached ($i = 0$): First, α is computed based on the population size $N = 4$, the number of individuals dominating the hyperrectangle ($|UP| = 2$), and the fitness parameter $k = 2$, which yields $\alpha = 1/3$; then for b and c, the fitness values are increased by adding $\alpha \cdot V/|UP| = 4/3$.

Applying this procedure to all slices at a particular recursion level identifies all hyperrectangles which constitute the portion of the objective space enclosed by the population and the reference set.

IV. Estimating Hypervolume Contributions Using Carlo Simulation

As outlined above, the computation of the proposed hypervolume-based fitness scheme is that expensive that only problems with at maximum four or five objectives are tractable within reasonable time limits. However, in the context of randomized search heuristics one may argue that the exact fitness values are not crucial and approximated values may be sufficient; furthermore, if using pure rank-based selection schemes, then only the resulting order of the individuals matters. These considerations lead to the idea of estimating the hypervolume contributions by means of Monte Carlo simulation.

The basic principle is described and investigated in the following subsection, while more advanced sampling strategies that automatically adjust the number of samples to be drawn are discussed in the second part of this section.

A. Basic Concept

To approximate the fitness values according to Definition III.6, we need to estimate the Lebesgue measures of the domains $H_i(a, P, R)$ where $P \in \Psi$ is the population. Since these domains are all integrable, their Lebesgue measure can be approximated by means of Monte Carlo simulation.

For this purpose, a sampling space $S \subseteq Z$ has to be defined with the following properties: (i) the hypervolume of S can easily be computed, (ii) samples from the space S can be generated fast, and (iii) S is a superset of the domains $H_i(a, P, R)$ the hypervolumes of which one would like to approximate. The latter condition is met by setting $S = H(P, R)$, but since it is hard both to calculate the Lebesgue measure of this sampling space and to draw samples from it, we propose using the axis-aligned minimum bounding box containing the $H_i(a, P, R)$ subspaces instead, i.e.:

$$S := \{(z_1, \dots, z_n) \in Z \,|\, \forall 1 \le i \le n : l_i \le z_i \le u_i\} \quad (17)$$

where

$$\begin{aligned} l_i &:= \min\nolimits_{a \in P} f_i(a) \\ u_i &:= \max\nolimits_{(r_1,\dots,r_n) \in R} r_i \end{aligned} \quad (18)$$

for $1 \le i \le n$. Hence, the volume V of the sampling space S is given by $V = \prod_{i=1}^{n} \max\{0, u_i - l_i\}$.

Now given S, sampling is carried out by selecting M objective vectors $s_1, \dots, s_M$ from S uniformly at random. For each s_j it is checked whether it lies in any partition $H_i(a, P, R)$ for $1 \le i \le k$ and $a \in P$. This can be determined in two steps: first, it is verified that s_j is 'below' the reference set R, i.e., it exists $r \in R$ that is dominated by s_j; second, it is verified that the multiset A of those population members dominating s_j is not empty. If both conditions are fulfilled, then we know that—given A—the sampling point s_j lies in all partitions $H_i(a, P, R)$ where $i = |A|$ and $a \in A$. This situation will be denoted as a *hit* regarding the ith partition of a. If any of the above two conditions is not fulfilled, then we call s_j a *miss*. Let $X_j^{(i,a)}$ denote the corresponding random variable that is equal to 1 in case of a hit of s_j regarding the ith partition of a and 0 otherwise.

Based on the M sampling points, we obtain an estimate for $\lambda(H_i(a, P, R))$ by simply counting the number of hits and multiplying the hit ratio with the volume of the sampling box:

$$\hat{\lambda}\big(H_i(a, P, R)\big) = \frac{\sum_{j=1}^{M} X_j^{(i,a))}}{M} \cdot V \quad (19)$$

This value approaches the exact value $\lambda(H_i(a, P, R))$ with increasing M by the law of large numbers. Due to the linearity of the expectation operator, the fitness scheme according to Eq. (11) can be approximated by replacing the Lebesgue measure with the respective estimates given by Eq. (19):

$$\hat{I}_h^k(a, P, R) = \sum_{i=1}^{k} \frac{\alpha_i}{i} \cdot \left(\frac{\sum_{j=1}^{M} X_j^{(i,a))}}{M} V \right) \quad (20)$$

Algorithm 3 Hypervolume-based Fitness Value Estimation

```
Require: population $P \in \Psi$, reference set $R \subseteq Z$, fitness
    parameter $k \in \mathbb{N}$, number of sampling points $M \in \mathbb{N}$
 1: procedure estimateHypervolume($P$, $R$, $k$, $M$)
 2:     /* determine sampling box $S$ */
 3:     for $i \leftarrow 1, n$ do
 4:         $l_i = \min_{a \in P} f_i(a)$
 5:         $u_i = \max_{(r_1,\dots,r_n) \in R} r_i$
 6:     end for
 7:     $S \leftarrow [l_1, u_1] \times \cdots \times [l_n, u_n]$
 8:     $V \leftarrow \prod_{i=1}^{n} \max\{0, (u_i - l_i)\}$
 9:     /* reset fitness assignment */
10:     $\mathcal{F} \leftarrow \bigcup_{a \in P} \{(a, 0)\}$
11:     /* perform sampling */
12:     for $j \leftarrow 1, M$ do
13:         choose $s \in S$ uniformly at random
14:         if $\exists r \in R:\ s \le r$ then
15:             $UP \leftarrow \bigcup_{a \in P,\ f(a) \le s} \{f(a)\}$
16:             if $|UP| \le k$ then
17:                 /* hit in a relevant partition */
18:                 $\alpha \leftarrow \prod_{l=1}^{|UP|-1} \frac{k-l}{|P|-l}$
19:                 /* update hypervolume estimates */
20:                 $\mathcal{F}' \leftarrow \emptyset$
21:                 for all $(a, v) \in \mathcal{F}$ do
22:                     if $f(a) \le s$ then
23:                         $\mathcal{F}' \leftarrow \mathcal{F}' \cup \{(a, v + \frac{\alpha}{|UP|} \cdot \frac{V}{M})\}$
24:                     else
25:                         $\mathcal{F}' \leftarrow \mathcal{F}' \cup \{(a, v)\}$
26:                     end if
27:                 end for
28:                 $\mathcal{F} \leftarrow \mathcal{F}'$
29:             end if
30:         end if
31:     end for
32:     return $\mathcal{F}$
33: end procedure
```

The details of estimation procedure are described by Algorithm 3 which returns a fitness assignment, i.e., for each $a \in P$ the corresponding hypervolume estimate $\hat{I}_h^k(a, P, R)$. It will be later used by the evolutionary algorithm presented in Section V. Note that the partitions $H_i(a, P, R)$ with $i > k$ do not need to be considered for the fitness calculation as they do not contribute to the I_h^k values that we would like to estimate, cf. Def. III.6.

In order to study how closely the sample size M and the accuracy of the estimates is related, a simple experiment was carried out: ten imaginary individuals $a \in A$ were generated, the objective vectors $f(a)$ of which are uniformly distributed at random on a three dimensional unit simplex, similarly to the experiments presented in Table II. These individuals were then ranked on the one hand according to the estimates $\hat{I}_h^{|A|}$ and on the other hand with respect to the exact values $I_h^{|A|}$. The closer the former ranking is to the latter ranking, the higher is the accuracy of the estimation procedure given by Algorithm 3. To quantify the differences between the two

Table III
ACCURACY OF THE RANKING OF 10 INDIVIDUALS ACCORDING TO $\hat{I}_h^{10}$ (20) IN COMPARISON TO I_h^{10} FOR DIFFERENT SAMPLE SIZES. THE PERCENTAGES REPRESENT THE NUMBER OF PAIRS OF INDIVIDUALS RANKED CORRECTLY.

number of samples M	ranking accuracy
10^1	56.0%
10^2	74.1%
10^3	89.9%
10^4	96.9%
10^5	99.2%
10^6	99.8%
10^7	100.0 %

rankings, we calculated the percentage of all pairs (i, j) with $1 \leq i < j \leq |A|$ where the individuals at the ith position and the jth position in the ranking according to $I_h^{|A|}$ have the same order in the ranking according to $\hat{I}_h^{|A|}$, see [31]. The experiment was repeated for different numbers of sampling points as shown in Table III. The experimental results indicate that 10.000 samples are necessary to achieve an error below 5% and that 10.000.000 sampling point are sufficient in this setting to obtain the exact ranking.

B. Adaptive Sampling

Seeing the close relationship between sample size and accuracy, one may ask whether M can be adjusted automatically on the basis of confidence intervals. That means sampling is stopped as soon as the statistical confidence in the estimated fitness values reaches a prespecified level. The hope is that thereby the estimation is less sensitive to the choice of M and that the number of drawn samples can be reduced.

Using the normal approximation of the distribution of an estimate $\hat{I}_h^k(a, A, R)$, we can state there is a probability of L, that the true value $I_h^k(a, A, R)$ is in

$$I_h^k(a, A, R) \in \hat{I}_h^k(a, A, R) \pm z_{1/2+L/2} \sqrt{\widehat{\operatorname{Var}}\big(\hat{I}_h^k(a, A, R)\big)} \tag{21}$$

where z_β denotes the β-percentile of a standard normal distribution and $\widehat{\operatorname{Var}}\big(\hat{I}_h^k(a, A, R)\big)$ denotes the estimated variance of the estimate according to (30), see Appendix B for details.

Based on this confidence interval, we can derive a lower bound for the probability C_w, that the individual with the smallest estimated contribution $\hat{I}_h^k(a, A, R)$ contributes least to the hypervolume (see Appendix B1a). Let $A = \{a_1, \ldots, a_{|A|}\}$ with $a_1 \leq a_i \; \forall i \in \{1, \ldots, |A|\}$, then

$$C_w \geq 1 - (1 - L)/2 - \sum_{i=2}^{|A|} P\big(I_h^k(a_i, A, R) > B\big) \tag{22}$$

where B denotes the upper end of confidence interval (21) for confidence level $(L + 1)/2$. Similiarly, a lower bound for the confidence C_r, that the ranking of individuals is correct follows as

$$\begin{aligned} C_r \geq 1 - &\sum_{i=1}^{|A|-1} P(I_h^k(a_i, A, R) < B_i) \\ &- \sum_{i=2}^{|A|} P(I_h^k(a_i, A, R) > B_{i-1}) \end{aligned} \tag{23}$$

(see Appendix B1c for details, including the meaning of B_i).

The lower bound (22) can be used to limit the number of samples used to estimate $I_h^k(a, A, R)$ in the environmental selection step, and (23) to limit the number of samples in the mating selection. Algorithm 4 gives an adaptive sampling procedure—based on the confidence levels—that resembles Algorithm 3. In contrast to the latter, the elements $(a, v, (h_1, \ldots, h_{|P|}))$ of $\mathcal{F}$ take a third component that records the number of hits h_i per partition $H_i(a, A, R)$, which are needed to calculate the confidence level.

In the following we investigate to what extend the adaptive adjustment of sampling size reduces the total number of samples needed. For this purpose, assume from a Pareto-front approximation A, some individuals are removed one by one. In each removal step, the confidence level C_w is calculated after a certain number of samples Θ are drawn, here set to $t = 100$. If C_w exceeds a user defined level L, then the adaptive sampling stops. Otherwise it continues, periodically recalculating the confidence level after Θ samples until either the confidence exceeds L or the maximum number of samples M_{max} is reached.

We consider the following scenario: 20 individuals $a \in A$ are generated whose objective values are set uniformly at random on a 3 dimensional unit simplex. From these 20 individuals, 10 are removed one by one based on the smallest indicator value $I_h^k(a, A, R)$ with $R = (2, 2, 2)$ and $k = |A| - 10$. The resultant set of 10 individuals A_{ref} acts as reference to assess the quality of the two sampling strategies.

The same procedure is then carried out for the estimated indicator $\hat{I}_h^k(a, A, R)$ using a constant number of samples M as well as using an adaptive number of samples according to Algorithm 4. In the latter case, the maximum number of samples M_{max} is set to M and the desired confidence level L is set to 0.99. The quality of the resulting set $A_{adaptive}$ is assessed by counting the percentage of individuals which are in $A_{adaptive}$ but not in A_{ref}, i.e., $|A_{ref} \cap A_{adaptive}|/|A_{ref}|$. In the same way the percentage is calculated for the set $A_{constant}$ that results from applying the constant sampling strategy.

Figure 6 shows the percentages obtained for both the constant and adaptive sampling approach. For small maximum number of samples M_{max}, the adaptive sampling algorithm hardly ever reaches the desired confidence level L. Hence, both the number of samples and ranking accuracy is similar to the constant approach. However, as the maximum sample size increases, the more often L is reached. In this case, the number of samples is only half the number needed by the algorithm based on the constant sampling strategy while the accuracy of the estimate is only marginaly affected. Since the confidence level is set to a relatively high value, the accuracy is only affected very slightly. This indicates that using adaptive sampling might be beneficial to speed up sampling when the number of samples is large.

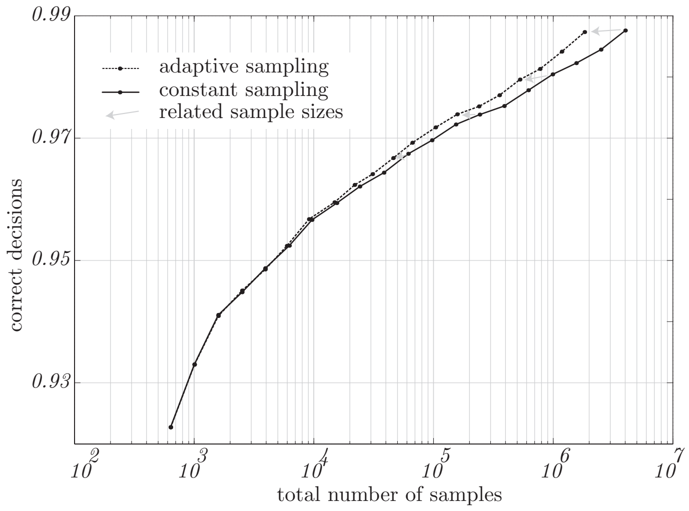


Figure 6. Ranking accuracy with increasing (maximum) number of samples for the constant and adaptive sampling strategy. As the grey arrows indicate, the latter uses less samples than the constant approach if the sample size is large enough. On the other hand, the two approaches differ only slightly when few samples are used.

V. HypE: Hypervolume Estimation Algorithm for Multiobjective Optimization

In this section, we describe an evolutionary algorithm named HypE (Hypervolume Estimation Algorithm for Multiobjective Optimization) which is based on the fitness assignment schemes presented in the previous sections. When the number of objectives is small (≤ 3), the hypervolume values I_h^k are computed exactly using Algorithm 1, otherwise they are estimated based on Algorithm 3.

The main loop of HypE is given by Algorithm 5. It reflects a standard evolutionary algorithm and consists of the successive application of mating selection (Algorithm 6), variation, and environmental selection (Algorithm 7). As to mating selection, binary tournament selection is proposed here, although any other selection scheme could be used as well. The procedure *variation* encapsulates the application of mutation and recombination operators to generate N offspring. Finally, environmental selection aims at selecting the most promising N solutions from the multiset-union of parent population and offspring; more precisely, it creates a new population by carrying out the following two steps:

1) First, the union of parents and offspring is divided into disjoint partitions using the principle of nondominated sorting [20], [13], also known as dominance depth. Starting with the lowest dominance depth level, the partitions are moved one by one to the new population as long as the first partition is reached that cannot be transfered completely. This corresponds to the scheme used in most hypervolume-based multiobjective optimizers [15], [23], [9].

Algorithm 4 Hypervolume-based Fitness Value Estimation With Adaptive Sampling

```
Require: population $P \in \Psi$, reference set $R \subseteq Z$, fitness
    parameter $k \in \mathbb{N}$, maximum number of sampling points
    $M_{max} \in \mathbb{N}$, desired confidence $L$, sampling interval $\Theta$
 1: procedure estimateHypervolumeAS($P$, $R$, $k$, $M_{max}$, $L$)
 2:     /* determine sampling box $S$ */
 3:     for $i \leftarrow 1, n$ do
 4:         $l_i = \min_{a \in P} f_i(a)$
 5:         $u_i = \max_{(r_1,\ldots,r_n) \in R}\ r_i$
 6:     end for
 7:     $S \leftarrow [l_1, u_1] \times \cdots \times [l_n, u_n]$
 8:     $V \leftarrow \prod_{i=1}^{n} \max\{0, (u_i - l_i)\}$
 9:     /* reset fitness assignement
10:         third component: number of hits per $H_i(a, P, R)$ */
11:     $\mathcal{F} \leftarrow \bigcup_{a \in P} \{(a, 0, (0, \ldots, 0))\}$
12:     /* compute $\alpha$-vector according to Equation (13) */
13:     $\alpha \leftarrow (0, \ldots, 0)$
14:     for $i \leftarrow 1, k$ do
15:         $\alpha_i \leftarrow \prod_{l=1}^{k-1} \frac{k-l}{|A|-l}$
16:     end for
17:     /* perform sampling */
18:     $C \leftarrow 0$ /* either $C_w$ (env. sel.) or $C_r$ (mat. sel) */
19:     $j \leftarrow 1$
20:     while $j \leq M_{max}$ and $C < L$ do
21:         choose $s \in S$ uniformly at random
22:         if $\exists r \in R:\ s \leq r$ then
23:             $\mathcal{F}' \leftarrow \emptyset$
24:             /* update estimates and hit counts */
25:             for all $(a, v, (h_1, \ldots, h_{|P|})) \in \mathcal{F}$ do
26:                 if $\forall 1 \leq j \leq n : f_j(a) \leq s_j$ then
27:                     $(h'_1, \ldots, h'_{|P|}) \leftarrow (h_1, \ldots, h_{|P|})$
28:                     $h'_{|A|} \leftarrow h'_{|A|} + 1$
29:                     $\mathcal{F}' \leftarrow \mathcal{F}' \cup \{(a, v + \frac{\alpha_A}{|A|} \cdot \frac{V}{M},$
30:                                         $(h'_1, \ldots, h'_{|P|}))\}$
31:                 else
32:                     $\mathcal{F}' \leftarrow \mathcal{F}' \cup \{(a, v, (h_1, \ldots, h_{|P|}))\}$
33:                 end if
34:             end for
35:             $\mathcal{F} \leftarrow \mathcal{F}'$
36:         end if
37:         /* Recalculate confidence level reached */
38:         if  $\bmod (j, \Theta) = 0$ then
39:             $C \leftarrow$ confidence according to (22) or (23)
40:         end if
41:     end while
42:     return $\mathcal{F}$
43: end procedure
```

2) The partition that only fits partially into the new population is then processed using the method presented in Section III-B. In each step, the fitness values for the partition under consideration are computed and the individual with the worst fitness is removed—if multiple individuals share the same minimal fitness, then one of them is selected uniformly at random. This procedure

Algorithm 5 HypE Main Loop

```
Require: reference set $R \subseteq Z$, population size $N \in \mathbb{N}$, number of generations $g_{\max}$, number of sampling points $M \in \mathbb{N}$
1: initialize population $P$ by selecting $N$ solutions from $X$ uniformly at random
2: $g \leftarrow 0$
3: while $g \leq g_{\max}$ do
4:     $P' \leftarrow$ matingSelection$(P, R, N, M)$
5:     $P'' \leftarrow$ variation$(P', N)$
6:     $P \leftarrow$ environmentalSelection$(P \cup P'', R, N, M)$
7:     $g \leftarrow g + 1$
8: end while
```

is repeated until the partition has been reduced to the desired size, i.e., until it fits into the remaining slots left in the new population.

Concerning the fitness assignment, the number of objectives determines whether the exact or the estimated I_h^k values are considered. If less than four objectives are involved, we recommend to employ Algorithm 1, otherwise to use Algorithm 3. The latter works with a fixed number of sampling points to estimate the hypervolume values I_h^k, regardless of the confidence of the decision to be made; hence, the variance of the estimates does not need to be calculated and it is sufficient to update for each sample drawn an array storing the fitness values of the population members.

Instead of Algorithm 3, one may also apply the adaptive sampling routine described in Algorithm 4 when estimating the hypervolume contributions. To this end, the variance of the estimates is calculated after a certain number of initial samples and from this, the confidence level is determined. If this lies below a user-defined level, then the sampling process continues. Since this process can last arbitrarily long (the difference between the hypervolume contributions of two solutions can be arbitrarily small), an upper bound for the maximal number of samples M_{max} has to be defined. If this number is reached, a decision is made based on the current estimates regardless of the confidence level. The main advantage of this adaptive procedure is that it is robust with respect to the choice of the sample size M. Its main disadvantage is the need to store for each population member the number of hits in all domains $H_i(a, P, R)$, which slows down the sampling considerably, as will be shown in Section VI-C.

VI. Experiments

This section serves two goals: (i) to investigate the influence of specific algorithmic concepts (fitness, sample size, adaptive sampling) on the performance of HypE, and (ii) to study the effectiveness of HypE in comparison to existing MOEAs. A difficulty that arises in this context is how to statistically compare the quality of Pareto-set approximations with respect to the hypervolume indicator when a large number of objectives ($n \geq 5$) is considered. In this case, exact computation of the hypervolume becomes infeasible; to this end, we propose

Algorithm 6 HypEMating Selection

```
Require: population $P \in \Psi$, reference set $R \subseteq Z$, number of offspring $N \in \mathbb{N}$, number of sampling points $M \in \mathbb{N}$
1: procedure matingSelection($P$,$R$,$N$,$M$)
2:     if $n \leq 3$ then
3:         $\mathcal{F} \leftarrow$ computeHypervolume$(P, R, N)$
4:     else
5:         $\mathcal{F} \leftarrow$ estimateHypervolume$(P, R, N, M)$
6:     end if
7:     $Q \leftarrow \emptyset$
8:     while $|Q| < N$ do
9:         choose $(a, v_a), (b, v_b) \in \mathcal{F}$ uniformly at random
10:        if $v_a > v_b$ then
11:            $Q \leftarrow Q \cup \{a\}$
12:        else
13:            $Q \leftarrow Q \cup \{b\}$
14:        end if
15:    end while
16:    return $Q$
17: end procedure
```

Monte Carlo sampling using appropriate statistical tools as detailed below.

A. Experimental Setup

HypE is implemented within the PISA framework [6] and tested in two versions: the first (HypE) uses fitness-based mating selection as described in Algorithm 6, while the second (HypE*) employs a uniform mating selection scheme where all individuals have the same probability of being chosen for reproduction. Unless stated otherwise, for sampling the number of sampling points is fixed to $M = 10,000$ and $M_{max} = 20,000$ respectively, both kept constant during a run.

HypE and HypE* are compared to three popular MOEAs, namely NSGA-II [13], SPEA2 [42], and IBEA (in combination with the ε-indicator) [41]. Since these algorithms are not designed to optimize the hypervolume, it cannot be expected that they perform particularly well when measuring the quality of the approximation in terms of the hypervolume indicator. Nevertheless, they serve as an important reference as they are considerably faster than hypervolume-based search algorithms and therefore can execute a substantially larger number of generations when keeping the available computation time fixed. On the other hand, dedicated hypervolume-based methods are included in the comparisons. The algorithms proposed in [15], [23], [9] use the same fitness assignment scheme which can be mimicked by means of a HypE variant that only uses the I_h^1 values for fitness assignment, i.e., k is set to 1, and employs the routine for exact hypervolume calculation (Algorithm 1). We will refer to this approach as RHV (regular hypervolume-based algorithm)—the acronym RHV* stands for the variant that uses uniform selection for mating. However, we do not provide comparisons to the original implementations of [15], [23], [9] because the focus is on the fitness assignment principles and

Algorithm 7 HypE Environmental Selection

```
Require: population $P \in \Psi$, reference set $R \subseteq Z$, number of
    offspring $N \in \mathbb{N}$, number of sampling points $M \in \mathbb{N}$
 1: procedure environmentalSelection($P$,$R$,$N$,$M$)
 2:     $P' \leftarrow P$ /* remaining population members */
 3:     $Q \leftarrow \emptyset$ /* new population */
 4:     $Q' \leftarrow \emptyset$ /* current nondominated set */
 5:     /* iteratively copy nondominated sets to $Q$ */
 6:     repeat
 7:         $Q \leftarrow Q \cup Q'$
 8:         /* determine current nondominated set in $P'$ */
 9:         $Q', P'' \leftarrow \emptyset$
10:         for all $a \in P'$ do
11:             if $\forall b \in P' : b \preceq a \Rightarrow a \preceq b$ then
12:                 $Q' \leftarrow Q' \cup \{a\}$
13:             else
14:                 $P'' \leftarrow P'' \cup \{a\}$
15:             end if
16:         end for
17:         $P' \leftarrow P''$
18:     until $|Q| + |Q'| \geq N \vee P' = \emptyset$
19:     /* truncate last non-fitting nondominated set $Q'$ */
20:     $k = |Q| + |Q'| - N$
21:     while $k > 0$ do
22:         if $n \leq 3$ then
23:             $\mathcal{F} \leftarrow$ computeHypervolume($Q', R, k$)
24:         else
25:             $\mathcal{F} \leftarrow$ estimateHypervolume($Q', R, k, M$)
26:         end if
27:         /* remove worst solution from $Q'$ */
28:         $Q' \leftarrow \emptyset$
29:         removed $\leftarrow$ false
30:         for all $(a, v) \in \mathcal{F}$ do
31:             if removed = true $\vee$ $v \neq \min_{(a,v) \in \mathcal{F}}\{v\}$ then
32:                 $Q' \leftarrow Q' \cup \{a\}$
33:             else
34:                 removed $\leftarrow$ true
35:             end if
36:         end for
37:         $k \leftarrow k - 1$
38:     end while
39:     $Q \leftarrow Q \cup Q'$
40:     return $Q$
41: end procedure
```

not on specific data structures for fast hypervolume calculation as in [15] or specific variation operators as in [23]. Furthermore, we consider the sampling-based optimizer proposed in [1], here denoted as SHV (sampling-based hypervolume-oriented algorithm); it more or less corresponds to RHV with adaptive sampling. Finally, to study the influence of the nondominated sorting we also include a simple HypE variant named RS (random selection) where all individuals are assigned the same constant fitness value. Thereby, the selection pressure is only maintained by the nondominated sorting carried out during the environmental selection phase.

As basis for the comparisons, the DTLZ [14], the WFG [21], and the knapsack [45] test problem suites are considered since they allow the number of objectives to be scaled arbitrarily—here, ranging from 2 to 50 objectives. For the DTLZ problem, the number of decision variables is set to 300, while for the WFG problems individual values are used, see Table IV. As to the knapsack problem, we used 400 items which where modified with mutation probability 1 by one-bit mutation and by one-point crossover with probability 0.5. For each benchmark function, 30 runs are carried out per algorithm using a population size of $N = 50$ and a maximum number $g_{\max} = 200$ of generations (unless the computation time is fixed). The individuals are represented by real vectors, where a polynomial distribution is used for mutation and the SBX-20 operator for recombination [12]. The recombination and mutation probabilities are set according to [14].

B. Statistical Comparison Methodology

The quality of the Pareto-set approximations are assessed using the hypervolume indicator, where for less than 6 objectives the indicator values are calculated exactly and otherwise approximated by Monte Carlo sampling as described in [2]. When sampling is used, uncertainty of measurement is introduced which can be expressed by the standard deviation of the sampled value $u(\hat{I}_H(A,R)) = I_H(A,R)\sqrt{p(1-p)/n}$, where p denotes the hit probability of the sampling process and $\hat{I}_H$ the hypervolume estimate. Unless otherwise noted, $1,000,000$ samples are used per Pareto-set approximation. For a typical hit probability between 10% to 90% observed, this leads to a very small uncertainty below 10^{-3} in relation to I_H. Therefore, it is highly unlikely that the uncertainty will influence the statistical test applyied to the hypervolume estimates and if it does nonetheless, the statistical tests become over-conservative. Hence, we do not consider uncertainty in the following tests.

Let A_i with $1 \leq i \leq l$ denote the algorithms to be compared. For each algorithm A_i, the same number r of independent runs are carried out for 200 generations. For formal reason, the null hypothesis that all algorithms are equally well suited to approximate the Pareto-opimal set is investigated first, using the Kruskal-Wallis test at a significance level of $\alpha = 0.01$ [10]. This hypothesis could be rejected in all test cases described below. Thereafter, for all pairs of algorithms the difference in median of the hypervolume is compared.

To test the difference for significance, the Conover-Inman procedure is applied with the same α level as in the Kruskal-Wallis test [10]. Let $\delta_{i,j}$ be 1, if A_i turns out to be significantly better than A_j and 0 otherwise. Based on $\delta_{i,j}$, for each algorithm A_i the performance index $P(A_i)$ is determined as follows:

$$P(A_i) = \sum_{\substack{j=1 \\ j \neq i}}^{l} \delta_{i,j} \qquad (24)$$

This value reveals how many other algorithms are better than the corresponding algorithm on the specific test case. The smaller the index, the better the algorithm; an index of

Table IV
NUMBER OF DECISION VARIABLES AND THEIR DECOMPOSITION INTO POSITION AND DISTANCE VARIABLES AS USED FOR THE WFG TEST FUNCTIONS DEPENDING ON THE NUMBER OF OBJECTIVES.

	Objective Space Dimensions (n)						
	2d	3d	5d	7d	10d	25d	50d
distance parameters	20	20	42	58	50	76	150
position parameters	4	4	8	12	9	24	49
decision variables	24	24	50	70	59	100	199

zero means that no other algorithm generated significantly better Pareto-set approximations in terms of the hypervolume indicator.

C. Results

In the following, we discuss the experimental results grouped according to the foci of the investigations.

1) Constant Versus Adaptive Sampling: First, the issue of adaptive sampling is examined. We compare HypE with its counterpart that uses the adaptive sampling strategy described in Algorithm 4 in both environmental selection (employing Equation (22)) and mating selection (emplyoing Equation (23)) with confidence level 0.1 and maximal number of samples of $M = 20,000$. As mentioned above, standard HypE is run with a sample size of $M = 10,000$ and a constant number of 200 generations.

The experiments indicate that for the 17 test problems each instantiated with 5, 7, 10, 25, and 50 objectives, the adaptive strategy beats the constant sampling method in 12 cases, which is vice versa in 10 cases better than adaptive sampling. In the remaining 63 instances, the two versions of HypE do not differ significantly in terms of the hypervolumes achieved.

While both strategy show about the same performance, it is expected that adaptive sampling needs less computation time because fewer samples are used. However, this is not the case: the adaptive sampling takes between 2.4 and 5.2 times longer than its counterpart, the difference increasing with the number of objectives. Two main reasons can be mentioned to explain the additional computation time:

- The confidence levels are seldom reached. This means the maximum number of samples are drawn with the overhead of calculating the variance of the points from time to time. Figure 7 shows the average number of samples drawn for the DTLZ 2 test problem with 5 objectives. While at the initial stage of the run about 50% of the maximum number of samples are used, the percentage steadily increases to over 95% after 100 generations.
- Calculating the variances requires to take track of the hits in each partition $H_i(a, A, R)$ for every population member a. This slows down the sampling process considerably.

These observations lead to the conclusion that the benefits of adaptive sampling are mainly compensated by the additional computational overhead. For this reason, only HypE with a constant sample size is considered in the following comparisons. Future research may be necessary to investigate how adaptivity could be integrated more efficiently. For instance, as Figure 7 indicates, one could use adaptive sampling in an early stage of the evolutionary run, say in the first twenty generations, and then switch to constant sampling. Furthermore, adaptive sampling might pay off when the user defined confidence L is close to 1 and large number of samples need to be drawn.

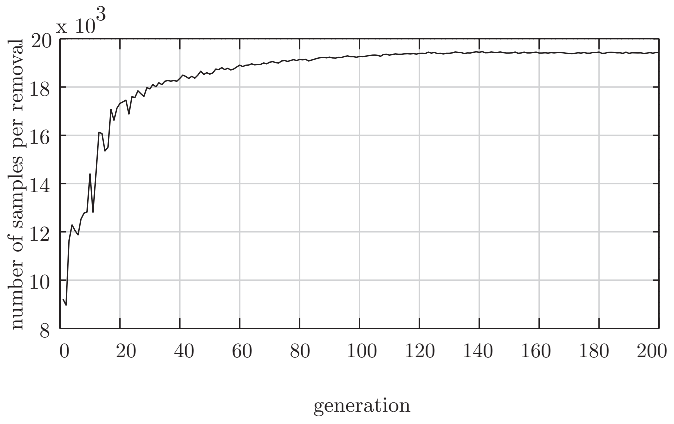


Figure 7. Average number of samples used per removal for adaptive HypE with significance level 0.05 and a maximum of $20,000$ samples. The testproblem is DTLZ 2 with 5 objectives and the average is based on 30 runs.

2) Exact Hypervolume Computation Versus Sampling: Next, we compare HypE with RHV—due to the large computation effort caused by the exact hypervolume calculation only on a single test problem, namely DTLZ2 with 2, 3, 4, and 5 objectives. Both HypE and HypE* are run with exact fitness calculation (Algorithm 1) as well as with the estimation procedure (Algorithm 3); the former variants are marked with a trailing 'e', while the latter variants are marked with a trailing '-s'. All algorithms run for 200 generations, per algorithm 30 runs were performed.

Figure 8 shows the hypervolume values normalized for each test problem instance separately. As one may expect, HypE beats HypE*. Moreover, fitness-based mating selection is beneficial to both HypE and RHV. The two best variants, HypE-e and RHV, reach about the same hypervolume values, independently of the number of objectives. Although HypE reaches a better hypervolume median for all four number of objectives, the difference is never significant [5]. . Hence,

[5]According to the Kruskal-Wallis test described in Section VI-B with confidence level 0.01.

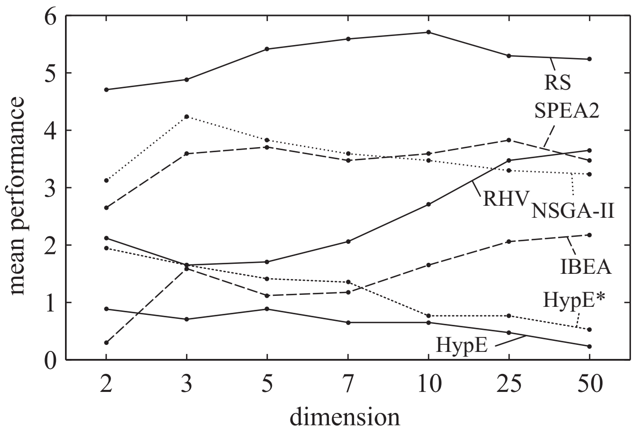


Figure 9. Mean performance score over all test problems for different number of objectives. The smaller the score, the better the Pareto-set approximation in terms of hypervolume.

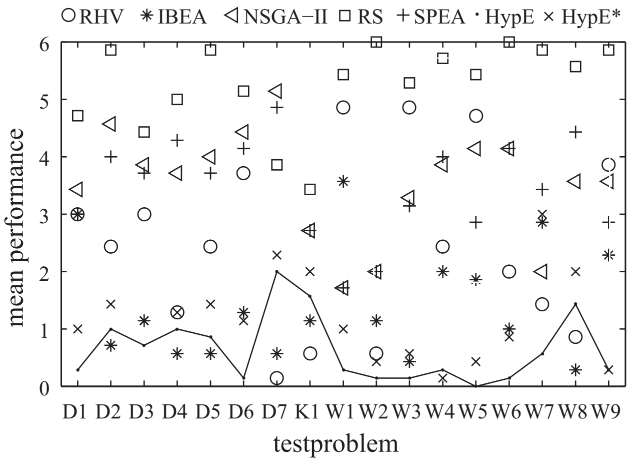


Figure 10. Mean performance score over all dimensions for different test problems, namely DTLZ (Dx), WFG (Wx) and knapsack (K1). The values of HypE+ are connected by a solid line to easier assess the score.

HypE can be considered an adequate alternative to the regular hypervolume algorithms; the main advantage though becomes evident when the respective fitness measures need to be estimated, see below.

3) HypE Versus Other MOEAs: Now we compare HypE and HypE*, both using a constant number of samples, to other multiobjective evolutionary algorithms. Table V on pages 23–25 shows the performance score and mean hypervolume on the 17 test problems mentioned in the *Experimental Setup* Section. Except on few testproblems HypE is better than HypE*. HypE reaches the best performance score overall. Summing up all performance scores, HypE yields the best total (76), followed by HypE* (143), IBEA (171) and the method proposed in [1] (295). SPEA2 and NSGA-II reach almost the same score (413 and 421 respectively), clearly outperforming the random selection (626).

In order to better visualize the performance index, we show to Figures where the index is summarized for different test-problems and number of objectives respectively. Figure 9 shows the average performance over all testproblems for different number of objectives. Except for two objective problems, HypE yields the best score, increasing its lead in higher dimensions. The version using uniform mating selection, HypE*, is outperformed by IBEA for two to seven objectives and only thereafter reaches a similar score as HypE. This indicates, that using non-uniform mating selection is particularly advantageous for small number of objectives.

Next we look at the performance score for the individual test-problems. Figure 10 shows the average index over all number of objectives. For DTLZ2, 4, 5 and 7, knapsack and WFG8, IBEA outperforms HypE, for DTLZ7 and knapsack, SHV as well is better than HypE. On the remaining 11 testproblems, HypE reaches the best mean performance.

Note that the above comparison is carried out for the case all algorithms run for the same number of generations and HypE needs longer execution time, e.g., in comparison to SPEA2 or NSGA-II. We therefore investigate in the following, whether NSGA-II and SPEA2 will not overtake HypE given a constant amount of time. Figure 11 shows the hypervolume of the Pareto-set approximations over time for HypE using the exact fitness values as well as the estimated values for different samples sizes M. Although only the results on WFG9 are shown, the same experiments were repeated on DTLZ2, DTLZ7, WFG3 and WFG6 and provided similar outcomes. Even though SPEA2, NSGA-II and even IBEA are able to process twice as many generations as the exact HypE, they do not reach its hypervolume. In the three dimensional example used, HypE can be run sufficiently fast without approximating the fitness values. Nevertheless, the sampled version is used as well to show the dependency of the execution time and quality on the number of samples M. Via M, the execution time of HypE can be traded off against the quality of the Pareto-set approximation. The fewer samples are used, the more the behavior of HypE resembles random selection. On the other hand by increasing M, the quality of exact calculation can be achieved, increasing the execution time, though. For example, with $M = 1,000$, HypE is able to carry out nearly the same number of generations as SPEA2 or NSGA-II, but the Pareto-set is just as good as when $100,000$ samples are used, producing only a fifteenth the number of generations. In the example given, $M = 10,000$ represents the best compromise, but the number of samples should be increased in two cases: (i) the fitness evaluation takes more time. This will affect the faster algorithm much more and increasing the number of samples will influence the execution time much less. (ii) More generations are used. In this case, HypE using more samples might overtake the faster versions with fewer samples, since those are more vulnerable to stagnation.

VII. Conclusions

This paper proposes HypE (Hypervolume Estimation Algorithm for Multiobjective Optimization), a novel hypervolume-based multiobjective evolutionary algorithm that can be applied to problems with arbitrary numbers of objective functions. It incorporates a new fitness assignment scheme based

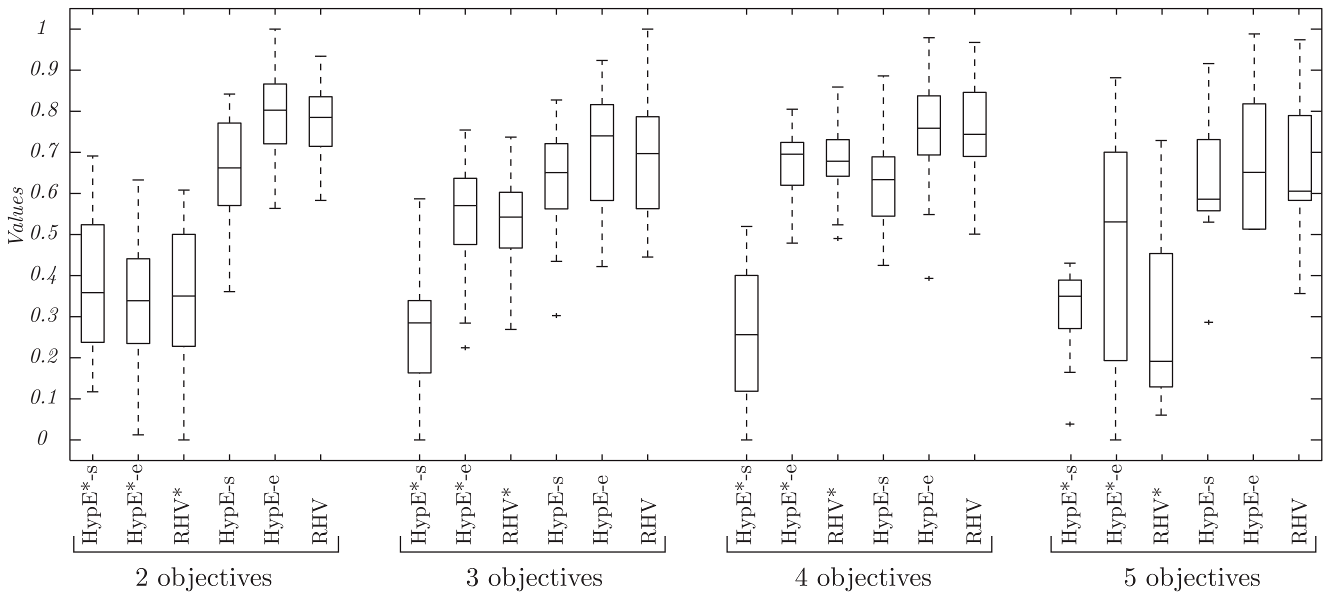


Figure 8. Comparison of the hypervolume indicator values for different variants of HypE and the regular hypervolume algorithm (RHV) on DTLZ2 with 2, 3, 4, and 5 objectives. For presentation reasons, the hypervolume values are normalized to the minimal and maximal values observed per problem instance.

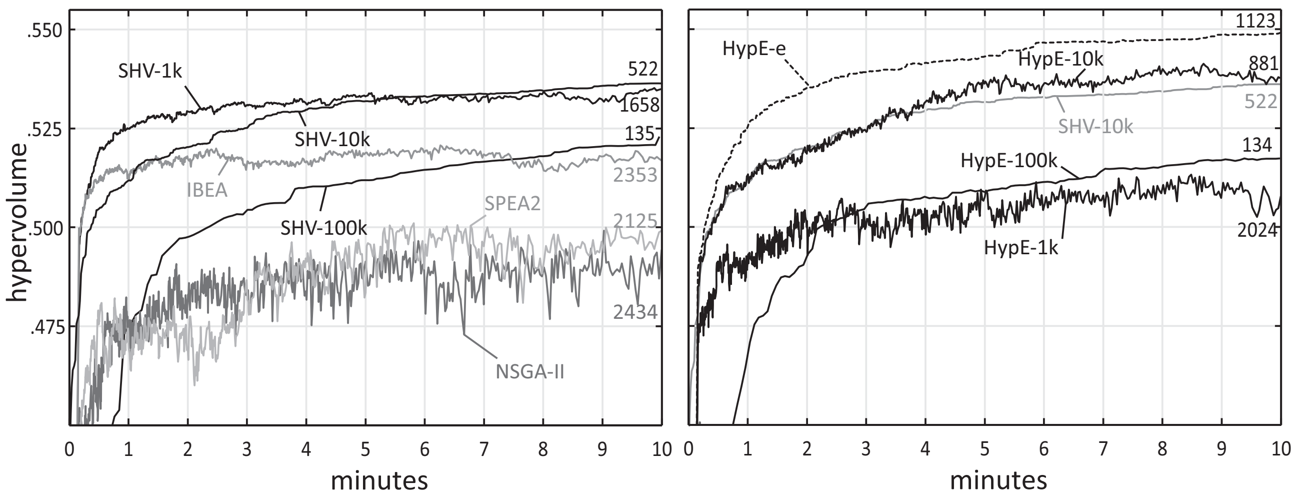


Figure 11. Hypervolume process over ten minutes of HypE+ for different samples sizes x in thousands (Hy-xk) as well as using the exact values (Hy-e). The test problem is WFG9 for three objectives. HypE is compared to the algorithms presented in Section VI, where the results are split in two figures with identical axis for the sake of clarity. The numbers at the left border of the figures indicate the total number of generations.

on the Lebesgue measure, where this measure can be both exactly calculated and estimated by means of Monte Carlo sampling. The latter allows to trade-off fitness accuracy versus the overall computing time budget which renders hypervolume-based search possible also for many-objective problems, in contrast to [15], [23], [9]. HypE is available for download at http://www.tik.ee.ethz.ch/sop/download/supplementary/hype/.

HypE was compared to various state-of-the-art MOEAs with regard to the hypervolume indicator values of the generated Pareto-set approximations—on the DTLZ [14], the WFG [21], and the knapsack [45] test problem suites. The simulations results indicate that HypE is a highly competitive multiobjective search algorithm; in the considered setting the Pareto front approximations obtained by HypE reached the best hypervolume value in 6 out of 7 cases averaged over all testproblems.

A promising direction of future research is the development of advanced adaptive sampling strategies that exploit the available computing resources most effectively, such as increasing the number of samples towards the end of the evolutionary run.

Acknowledgements

Johannes Bader has been supported by the Indo-Swiss Joint Research Program under grant IT14.

References

[1] J. Bader, K. Deb, and E. Zitzler. Faster Hypervolume-based Search using Monte Carlo Sampling. In *Conference on Multiple Criteria Decision Making (MCDM 2008)*. Springer, 2008. to appear.

[2] J. Bader and E. Zitzler. HypE: Fast Hypervolume-Based Multiobjective Search Using Monte Carlo Sampling. TIK Report 286, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich, April 2006.

[3] N. Beume, C. M. Fonseca, M. Lopez-Ibanez, L. Paquete, and J. Vahrenhold. On the Complexity of Computing the Hypervolume Indicator. Technical Report CI-235/07, University of Dortmund, December 2007.

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

[5] N. Beume and G. Rudolph. Faster S-Metric Calculation by Considering Dominated Hypervolume as Klee's Measure Problem. Technical Report CI-216/06, Sonderforschungsbereich 531 Computational Intelligence, Universität Dortmund, 2006. shorter version published at IASTED International Conference on Computational Intelligence (CI 2006).

[6] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA—A Platform and Programming Language Independent Interface for Search Algorithms. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, volume 2632 of *LNCS*, pages 494–508, Berlin, 2003. Springer.

[7] L. Bradstreet, L. Barone, and L. While. Maximising Hypervolume for Selection in Multi-objective Evolutionary Algorithms. In *2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, pages 6208–6215, Vancouver, BC, Canada, 2006. IEEE.

[8] K. Bringmann and T. Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. In *Proc. of the 19th International Symposium on Algorithms and Computation (ISAAC 2008)*, Berlin, Germany, 2008. Springer.

[9] D. Brockhoff and E. Zitzler. Improving Hypervolume-based Multiobjective Evolutionary Algorithms by Using Objective Reduction Methods. In *Congress on Evolutionary Computation (CEC 2007)*, pages 2086–2093. IEEE Press, 2007.

[10] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley, 3 edition, 1999.

[11] J. Cooper and T. Friedrich. The Hypervolume Indicator and Klee's Measure Problem. Submitted to ICALP 2008.

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

[13] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In M. Schoenauer et al., editors, *Conference on Parallel Problem Solving from Nature (PPSN VI)*, volume 1917 of *LNCS*, pages 849–858. Springer, 2000.

[14] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable Test Problems for Evolutionary Multi-Objective Optimization. In A. Abraham, R. Jain, and R. Goldberg, editors, *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, chapter 6, pages 105–145. Springer, 2005.

[15] M. Emmerich, N. Beume, and B. Naujoks. An EMO Algorithm Using the Hypervolume Measure as Selection Criterion. In *Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, volume 3410 of *LNCS*, pages 62–76. Springer, 2005.

[16] M. Emmerich, A. Deutz, and N. Beume. Gradient-Based/Evolutionary Relay Hybrid for Computing Pareto Front Approximations Maximizing the S-Metric. In *Hybrid Metaheuristics*, pages 140–156. Springer, 2007.

[17] R. Everson, J. Fieldsend, and S. Singh. Full Elite-Sets for Multiobjective Optimisation. In I.C. Parmee, editor, *Proceedings of the fifth international conference on adaptive computing in design and manufacture (ADCM 2002)*, pages 343–354, London, UK, 2002. Springer.

[18] M. Fleischer. The measure of Pareto optima. Applications to multi-objective metaheuristics. In C. M. Fonseca et al., editors, *Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, volume 2632 of *LNCS*, pages 519–533, Faro, Portugal, 2003. Springer.

[19] C. M. Fonseca, L. Paquete, and M. López-Ibáñez. An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator. In *Congress on Evolutionary Computation (CEC 2006)*, pages 1157–1163, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC Canada, 2006. IEEE Press.

[20] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.

[21] S. Huband, P. Hingston, L. Barone, and L. While. A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

[22] S. Huband, P. Hingston, L. White, and L. Barone. An Evolution Strategy with Probabilistic Mutation for Multi-Objective Optimisation. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, volume 3, pages 2284–2291, Canberra, Australia, 2003. IEEE Press.

[23] C. Igel, N. Hansen, and S. Roth. Covariance Matrix Adaptation for Multi-objective Optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

[24] J. Knowles and D. Corne. On Metrics for Comparing Non-Dominated Sets. In *Congress on Evolutionary Computation (CEC 2002)*, pages 711–716, Piscataway, NJ, 2002. IEEE Press.

[25] J. Knowles and D. Corne. Properties of an Adaptive Archiving Algorithm for Storing Nondominated Vectors. *IEEE Transactions on Evolutionary Computation*, 7(2):100–116, 2003.

[26] J. D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD thesis, University of Reading, 2002.

[27] M. Laumanns, G. Rudolph, and H.-P. Schwefel. Approximating the Pareto Set: Concepts, Diversity Issues, and Performance Assessment. Technical Report CI-7299, University of Dortmund, 1999.

[28] S. Mostaghim, J. Branke, and H. Schmeck. Multi-objective Particle Swarm Optimization on Computer Grids. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO 2007)*, pages 869–875, New York, NY, USA, 2007. ACM.

[29] M. Nicolini. A Two-Level Evolutionary Approach to Multi-criterion Optimization of Water Supply Systems. In *Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, volume 3410 of *LNCS*, pages 736–751. Springer, 2005.

[30] S. Obayashi, K. K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors. *Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)*, volume 4403 of *LNCS*, Berlin, Germany, 2007. Springer.

[31] J. Scharnow, K. Tinnefeld, and I. Wegener. The Analysis of Evolutionary Algorithms on Sorting and Shortest Paths Problems. *Journal of Mathematical Modelling and Algorithms*, 3(4):349–366, 2004. Online Date Tuesday, December 28, 2004.

[32] J. R. Swisher and S. H. Jacobson. A survey of ranking, selection, and multiple comparison procedures for discrete-event simulation. In *WSC '99: Proceedings of the 31st conference on Winter simulation*, pages 492–501, New York, NY, USA, 1999. ACM.

[33] D. A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Graduate School of Engineering, Air Force Institute of Technology, Air University, 1999.

[34] T. Wagner, N. Beume, and B. Naujoks. Pareto-, Aggregation-, and Indicator-based Methods in Many-objective Optimization. In S. Obayashi et al., editors, *Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)*, volume 4403 of *LNCS*, pages 742–756, Berlin Heidelberg, Germany, 2007. Springer. extended version published as internal report of Sonderforschungsbereich 531 Computational Intelligence CI-217/06, Universität Dortmund, September 2006.

[35] L. While. A New Analysis of the LebMeasure Algorithm for Calculating Hypervolume. In *Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, volume 3410 of *LNCS*, pages 326–340, Guanajuato, México, 2005. Springer.

[36] L. While, P. Hingston, L. Barone, and S. Huband. A Faster Algorithm for Calculating Hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, 2006.

[37] Q. Yang and S. Ding. Novel Algorithm to Calculate Hypervolume Indicator of Pareto Approximation Set. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues, Third International Conference on Intelligent Computing (ICIC 2007)*, volume 2, pages 235–244, 2007.

[38] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH Zurich, Switzerland, 1999.

[39] E. Zitzler. Hypervolume metric calculation. ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/hypervol.c, 2001.

[40] E. Zitzler, D. Brockhoff, and L. Thiele. The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration. In S. Obayashi et al., editors, *Conference on Evolutionary*

Multi-Criterion Optimization (EMO 2007), volume 4403 of *LNCS*, pages 862–876, Berlin, 2007. Springer.

[41] E. Zitzler and S. Künzli. Indicator-Based Selection in Multiobjective Search. In *Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *LNCS*, pages 832–842. Springer, 2004.

[42] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In K.C. Giannakoglou et al., editors, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pages 95–100. International Center for Numerical Methods in Engineering (CIMNE), 2002.

[43] E. Zitzler and L. Thiele. An Evolutionary Approach for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Networks Laboratory, ETH Zurich, May 1998.

[44] E. Zitzler and L. Thiele. Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In *Conference on Parallel Problem Solving from Nature (PPSN V)*, pages 292–301, Amsterdam, 1998.

[45] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[46] E. Zitzler, L. Thiele, and J. Bader. On Set-Based Multiobjective Optimization. Technical Report 300, Computer Engineering and Networks Laboratory, ETH Zurich, February 2008.

[47] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

Appendix

A. Proofs for Section III

Proof of Theorem III.3, page 4: According to 5 it holds:

$$\begin{aligned} I_H(A,R) &= \lambda\big(H(A,R)\big) \\ &= \lambda\Big(\dot{\bigcup_{S\subseteq A}} H(S,A,R)\Big) \ . \end{aligned}$$

By dividing the subsets into groups of equal size, one obtains

$$= \lambda\Big(\dot{\bigcup_{1\le i\le |A|}} \dot{\bigcup_{\substack{S\subseteq A\\|S|=i}}} H(S,A,R)\Big)$$

which can be rewritten as

$$= \sum_{i=1}^{|A|} \lambda\Big(\bigcup_{\substack{S\subseteq A\\|S|=i}} H(S,A,R)\Big)$$

because the inner unions are all disjoint. Now, for each subset of size i we count the Lebesque measure once for each element and then divide by $1/i$:

$$= \sum_{i=1}^{|A|} \frac{1}{i} \sum_{a\in A} \lambda\Big(\bigcup_{\substack{S\subseteq A\\|S|=i\\a\in S}} H(S,A,R)\Big) \ .$$

Changing the order of the sums results in

$$= \sum_{a\in A} \sum_{i=1}^{|A|} \frac{1}{i} \lambda\Big(\bigcup_{\substack{S\subseteq A\\|S|=i\\a\in S}} H(S,A,R)\Big)$$

and using Definition III.2 we obtain

$$= \sum_{a\in A} I_h(a,A,R)$$

which concludes the proof. ■

Proof of Theorem III.4, page 4: From Theorem III.3 we know that

$$\sum_{b_1\in\{a\}\cup B_1} I_h(b_1,\{a\}\cup B_1,R) = I_H(\{a\}\cup B_1,R)$$

which—following Definition III.1—equals

$$= \lambda\big(H(\{a\}\cup B_1,R)\big) \ .$$

Since $\{a\} \preccurlyeq B_1$, it holds $H(b,R) \subseteq H(A,R)$ for all $b \in B_1$ and therefore the above formula can be simplified to

$$= \lambda\big(H(\{a\},R)\big)$$

The same holds analogically for the right-hand side of the equation in Theorem III.4 which proves the claim. ■

Proof of Theorem III.7, page 4: Definition III.6 states that

$$I_h^k(a,A,R) = \frac{1}{|\mathcal{S}|} \sum_{S\in\mathcal{S}} \left[\sum_{\substack{T\subseteq S\\a\in T}} \frac{1}{|T|} \lambda\big(H(T,A,R)\big) \right]$$

where $\mathcal{S}$ denotes the set of subsets of A, that contain k elements, one of which is individual a, i.e., $\mathcal{S} = \{S \subseteq A; a \in S \wedge |S| = k\}$. Inserting the definition of $\mathcal{S}$ leads to

$$= \frac{1}{|\mathcal{S}|} \sum_{\substack{S\in A\\|S|=k\\a\in S}} \sum_{\substack{T\subseteq S\\a\in T}} \frac{1}{|T|} \lambda\big(H(T,A,R)\big) \tag{25}$$

To combine the two summations of the previous equation, let $o(T)$ denote the number of times the summand $\frac{1}{|T|}\lambda\big(H(T,A,R)\big)$ is added for the same set T

$$= \frac{1}{|\mathcal{S}|} \sum_{\substack{T\subseteq A\\a\in T}} o(T) \frac{1}{|T|} \lambda\big(H(T,A,R)\big)$$

splitting up into summation over subsets of equal size gives

$$= \frac{1}{|\mathcal{S}|} \sum_{i=1}^{k} \frac{1}{i} \sum_{\substack{T\subseteq A\\|T|=i\\a\in T}} o(T) \lambda\big(H(T,A,R)\big)$$

For symmetry reasons, each subset T with cardinality $|T| = i$ has the same number of occurences $o(T) =: o_i$

$$= \frac{1}{|\mathcal{S}|} \sum_{i=1}^{k} \frac{o_i}{i} \sum_{\substack{T\subseteq A\\|T|=i\\a\in T}} \lambda\big(H(T,A,R)\big)$$

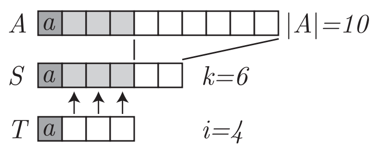


Figure 12. T is a subset of S, which in turn is a subset of A; all three sets contain a. Given one particular T of size i there exist $\binom{|A|-i}{k-i}$ subsets $S \subseteq A$ of size k which are a superset of T.

since all $H(T,A,R)$ in the sum are disjoint according to (5)

$$= \frac{1}{|\mathcal{S}|}\sum_{i=1}^{k}\frac{o_i}{i}\lambda\Big(\bigcup_{\substack{T\subseteq A\\|T|=i\\a\in T}} H(T,A,R)\Big)$$

and according to Equation (6)

$$= \sum_{i=1}^{k}\frac{o_i}{i\cdot|\mathcal{S}|}\lambda\big(H_i(a,A,R)\big)$$

After having transformed the original equation, we determine the number o_i, i.e., the number of times the term $\frac{1}{|T|}\lambda\big(H(T,A,R)\big)$ appears in (25). The term is added once every time the corresponding set T is a subset of S. Hence, $o(T)$ with $|T|=i$ corresponds to the number of sets S that are a superset of T. As depicted in Figure A, T defines i elements of S and the remaining $k-i$ elements can be chosen freely from the $|A|-i$ elements in A that are not yet in S.

Therefore, there exist $\binom{|A|-i}{k-i}$ subsets $S\in\mathcal{S}$ that contain one particular T with $|T|=i$ and $a\in T$. Therefore, $o(T)=o_i=\binom{|A|-i}{k-i}$. Likewise, the total number of sets S is $|\mathcal{S}|=\binom{|A|-1}{k-1}$.

Hence

$$\begin{aligned}\frac{o_i}{|\mathcal{S}|} &= \frac{\binom{|A|-i}{k-i}}{\binom{|A|-1}{k-1}}\\ &= \frac{(|A|-i)!(k-1)!((|A|-1)-(k-1))!}{(k-i)!((|A|-i)-(k-i))!(|A|-1)!}\\ &= \frac{(|A|-i)!(k-1)!}{(|A|-1)!(k-i)!}\\ &= \frac{(k-1)(k-2)\cdots(k-(i-1))}{(|A|-1)(|A|-2)\cdots(|A|-(i-1)}\\ &= \prod_{j=1}^{i-1}\frac{k-j}{|A|-j}\\ &= \alpha_i\end{aligned}$$

Therefore

$$I_h^k(a,A,R) = \sum_{i=1}^{k}\frac{\alpha_i}{i}\lambda\big(H_i(a,A,R)\big)$$

which concludes the proof. ■

B. Derivation of Confidence Intervals used in IV-B

Here, we consider confidence intervals for $\hat{I}_h^k(a,A,R)$ in order to adaptively determine the number of sampling points to be drawn in order to reach a user predefined confidence level. However, there are also other uses: for instance, to indicate the quality when reporting the values to a decision maker or carrying out statistical tests; or to adapt the selection probabilites for fitness proportionate selection meaning that high-fitness individuals are penalized whenever the fitness value comes along with a high uncertainty.

The confidence intervals can be derived by determining the distribution of a sampled value. To this end, a new discrete random variable Z is introduced with the property $\Pr(Z=\alpha_i/i)=p_i$ where p_i denotes the probability of a hit in the ith partition, i.e., a hit in $H_i(a,A,R)$; p_i corresponds to the ratio between the volume of the particular partition and the volume of the sampling space. Formula (20) can now be rewritten as a sum of M random variables Z_i distributed independently identically as Z

$$\hat{I}_h^k(a,A,R) = \frac{V}{M}\sum_{s=1}^{M}Z_i \tag{26}$$

According to the central limit theorem, this estimate is approximately normally distributed with mean $I_h^k(a,A,R$ and the following variance:

$$\mathrm{Var}\big(\hat{I}_h^k(a,A,R)\big) = \frac{V^2}{M^2}\Big(\sum_{i=1}^{k}\frac{\alpha_i^2}{i^2}\mathrm{Var}(P_i)+ \tag{27}$$

$$2\sum_{i=1}^{k}\sum_{j=i+1}^{k}\frac{\alpha_i\alpha_j}{ij}\mathrm{Cov}(P_i,P_j)\Big) \tag{28}$$

where P_i denotes the number of hits in the ith partition. The variance of P_i follows from the binomial distribution as $\mathrm{Var}(P_i)=Mp_i(1-p_i)$ and the covariance of P_i and P_j is

$$\mathrm{Cov}(P_i,P_j) = \sum_{f=1}^{M}\sum_{g=1}^{M}\mathrm{Cov}\big(X_f^{(i)},X_g^{(j)}\big) = -Mp_ip_j \tag{29}$$

because $\mathrm{Cov}(X_f^{(i)},X_g^{(j)})=-p_ip_j$ if $i=j$ and 0 otherwise. Since the probabilities p_i are unknown, the estimates $\hat{p}_i=P_i/M$ are used instead. This of course presupposes counting the number of hits in a particular partition. The estimated variance according to Eq. (28) becomes

$$\widehat{\mathrm{Var}}\big(\hat{I}_h^k(a,A,R)\big) = \frac{V^2}{M}\Big(\sum_{i=1}^{k} \frac{\alpha_i^2}{i^2}P_i(1-P_i) - 2\sum_{i=1}^{k}\sum_{j=i+1}^{k}\frac{\alpha_i\alpha_j}{ij}P_iP_j\Big) \tag{30}$$

The normal approximation with the mean and variance just derived can now be used to give the confidence interval for a particular estimate, i.e.,

$$I_h^k(a,A,R) \in \hat{I}_h^k(a,A,R) \pm z_{1-p_{err}/2}\sqrt{\widehat{\mathrm{Var}}\big(\hat{I}_h^k(a,A,R)\big)} \tag{31}$$

where z_β denotes the β-percentile of a standard normal distribution and p_{err} the probability of error.

1) Ranking and Selection: In order to use the confidence interval of Equation 31 for adaptive sampling, we will distinguish three situations—depending on how the fitness estimates $\hat{I}_h^k(a, A, R)$ of a population A are used:

- The individual of A with the smallest value needs to be identified for removal: This occurs when applying the iterative greedy strategy for environmental selection.
- The subset A' of A compromising the k individuals with the smallest fitness values should be determined for removal: This occurs when applying the one-shot greedy strategy for environmental selection.
- The population members $a \in A$ should be ranked according to their $I_h^k(a, A, R)$ values: This is the case when using rank-based mating selection, e.g., tournament selection.

In all three scenarios, one is interested in the probability that the selection and ranking respectively is done correctly, i.e., corresponds to the one that would result when using the exact values $I_h^k(a, A, R)$. This probability is derived in the following which then can be used as a stopping criterion for the sampling process.

Consider first the simple case of deciding between the estimate of two individuals $a, b \in A$ whether either is smaller. Let the indicator values be $v_1 := I_h^k(a, A, R)$ and $v_2 := I_h^k(b, A, R)$ and their estimates be $\hat{v}_1 := \hat{I}_h^k(a, A, R)$ and $\hat{v}_2 := \hat{I}_h^k(b, A, R)$ with $\hat{v}_1 < \hat{v}_2$, hence individual a is selected. Furthermore, let $\delta := v_1 - v_2$ denote difference between the indicator values and $\hat{\delta} := \hat{v}_1 - \hat{v}_2$ the difference between the estimates. By examining the distribution of δ given $\hat{\delta}$, the probability of correct selection C_p corresponds to the probability $P(\delta < 0 | \hat{\delta})$.

The difference δ is approximately normally distributed around $\hat{v}_1 - \hat{v}_2$ due to the same arguments used above. Since the same samples are used to estimate v_1 and v_2, the corresponding estimates are not independent—the variance of the difference is $\mathrm{Var}(\delta) \approx \sigma_{\hat{v}_1}^2 + \sigma_{\hat{v}_2}^2 - 2\sigma_{\hat{v}_1}\sigma_{\hat{v}_2}\rho_{\hat{v}_1,\hat{v}_2}$. The first two summands can be determined according to (30), the correlation ρ is more difficult to determine as it can vary greatly as Fig. 13 reveals. The correlation can be estimated iteratively, but this implies updating ρ for all pairs of individuals after each sample drawn. For two reasons this is impractical: Firstly, keeping track of all correlations requires a lot of memory and processing time compared to the sampling process itself and will significantly slow down the estimation process; secondly, the exact error probability is not crucial, and an approximation of it or a lower bound is adequate for the intended applications.

Therefore, we propose using an upper bound for the correct probability, based solely on the variance of the estimates. If the exact value v_1 is smaller than a value B and v_2 is bigger than B, then clearly $v_1 < v_2$. We can therefore give a lower

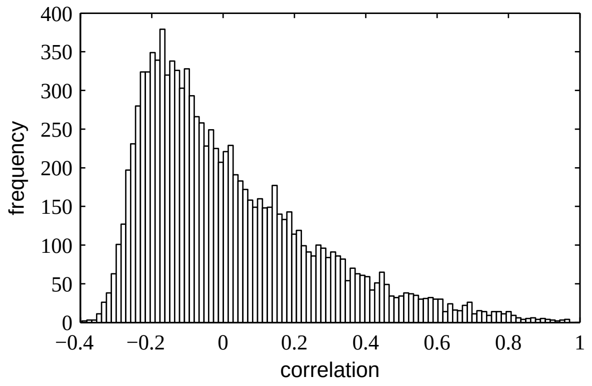


Figure 13. Correlation between fitness estimates of different individuals of a given Pareto-set approximation introduced in Table II. The closer to points are, the higher the correlation between their estimates becomes. Contrariwise, the estimates can be negatively correlated for points which lie far apart.

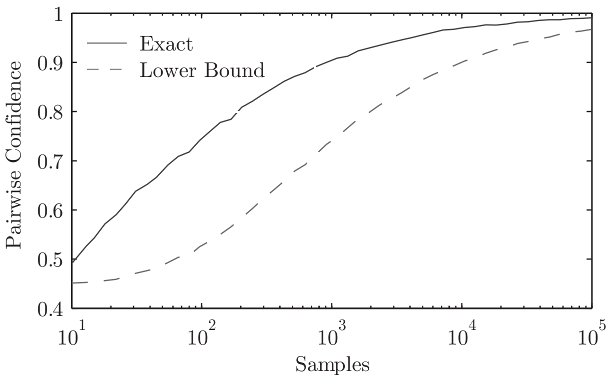


Figure 14. Shows the confidence interval for a comparison between two randomly selected individuals of a Pareto set approximation as in Table II. The solid line shows the exact confidence interval calculated based on the correlation information. The dashed course represents the lower bound according to (33).

bound for ordering the two individuals correctly by

$$P(v_1 < v_2 | \hat{v}_1 < \hat{v}_2) \geq P(v_1 < B | \hat{v}_1) \cdot P(v_2 > B | v_1 < B, \hat{v}_2) \tag{32}$$

where $B = \big(\hat{v}_1 + \hat{v}_2\big)/2$. (32) in turn can be lower bounded, which finally leads to

$$P(v_1 < v_2 | \hat{v}_1 < \hat{v}_2) \geq 1 - P(v_1 > B | \hat{v}_1) - P(v_2 < B | \hat{v}_2) \tag{33}$$

In (33) all probabilities involved are pairwise independent and hence can be determined using the normal distribution with the variance given in Eq. (30). Figure B1 shows the estimated confidence according to (33) in comparison to the exact confidence level.

After these general considerations, we can now provide specific lower bounds for the three situations mentioned above.

a) Selecting the Worst Individual: There exist many procedures in the field of ranking and selection [32]. However, they often assume that the estimates are independent or at

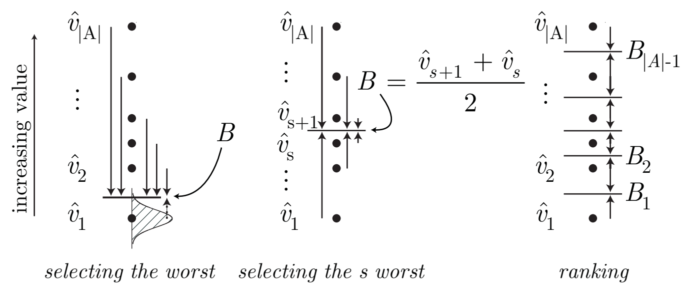


Figure 15. Shows the threshold for values for the three cases discussed in Section B1: When selecting the worst point, the threshold is determined by its estimated variance (left illustration); when the s worst points need to be selected (middle) or the points have to be ranked, then the threshold value(s) lie halfway between two estimates.

least that their correlation matrix has special properties like sphericity. Unfortunately, these assumptions are not met in our case. Moreover, determining the necesarry correlations between estimates slows down the sampling process considerably.

We therefore propose to use a rough lower bound for the confidence C_w. Though it is a quite conservative bound, the sampling speed is in a much minor degree affected compared to tighter approximations which consider the correlation. The idea is again to determine a threshold to which the estimates are compared.

Let $v_i = I_h^k(a_i, A, R)$ denote the indicator values for different individuals $a_i \in A$ with $1 \leq i \leq |A|$ and $\hat{v}_i$ the sampled estimates thereof. Let $\hat{v}_1 < \hat{v}_2 \cdots < \hat{I}_{|A|}$. The confidence C_w of correctly selecting the worst individuals is $P(\bigcap_{i=2}^{|A|} v_i > v_1 | \hat{v}_1, \ldots, \hat{v}_{|A|})$.

Left hand side of Figure 15 illustrates the procedure to determine a lower bound for C_w. Firstly, we calculate the confidence interval of the worst estimate $\hat{v}_1$ according to (31) at the level $(L+1)/2$, where L denotes the user defined confidence level for the selection problem. Let the threshold B then be the upper endpoint of this interval. Hence, v_1 will be smaller than B approximately with probability $(L+1)/2$ given $\hat{v}_1$.

For the remaining estimates $\hat{v}_i$ with $i > 1$ we then compute the probability $P(v_i \geq B | \hat{v}_i)$, that their exact value v_i is bigger than the threshold B by using once again the normal approximation (shown as arrows in Fig. 15). Clearly, the bigger $\hat{v}_i$, the bigger the probability becomes.

Based on L and $P(v_i \geq B | \hat{v}_i, i \geq 2)$, the probability of obtaining a correct selection outcome is then calculated analogically to (32) (all following probabilities are under the condition of known $\hat{v}_i$, which is omitted to facilitate readability):

$$C_w = P\left(\bigcap_{i=2}^{|A|} v_i > v_1\right) \tag{34}$$

$$\geq P(v_1 < B) \cdot P(v_2 \geq B | v_1 < B) \cdots \tag{35}$$

$$P\left(v_{|A|} \geq B | v_1 < B, \bigcap_{i=2}^{|A|-1} v_i \geq B\right) \tag{36}$$

which in turn can be lower bounded by

$$C_w \geq 1 - (1-L)/2 - \sum_{i=2}^{|A|} P(v_i > B) \tag{37}$$

In the latter approximation, all factors $P(v_i > B)$ can be determined using the normal approximation and the variance derived in (28), not involving any correlation between different estimates.

b) Selecting the s Worst Individuals: Let again $\hat{v}_1 < \hat{v}_2 \leq \cdots \leq \hat{v}_{|A|}$ denote the estimates of $v_i = I_h^k(a_i, A, R)$, with $a_i \in A$. In this section we consider the problem of not only removing the worst point, but the set $\hat{A}_w = \{a_1, \ldots, a_s\}$ of those s individuals with the worst estimated indicator values. On this problem, the confidence C_s shall be determined that the set $\hat{A}_w$ corresponds to the set A_w resulting form ranking the individuals according to the exact indicator values v_i. Hence, C_s is equal to the probability $P(\hat{A}_w = A_w)$.

The middle part of Figure 15 illustrates how a lower bound for the probability can be obtained. This time, the threshold B is set to lie between the best estimate still to be removed $\hat{v}_s$ and the worst estimate which remains in the set $\hat{v}_{s+1}$. Hence, $B = (\hat{v}_{s+1} + \hat{v}_s)/2$. A lower bound for the confidence of correctly selecting the set A_w is obtained by multiplying the probabilities, that v_i is smaller than B for all individuals of $\hat{A}_w$ and bigger than B for all $a_i \in A \backslash \hat{A}_w$:

$$C_s = P_c(\hat{A}_w = A_w) \tag{38}$$

$$\geq \prod_{i=1}^{s} P\Big(v_i < B | \bigcap_{j=1}^{i-1} v_j < B\Big) \cdot \tag{39}$$

$$\prod_{i=s+1}^{|A|} P\Big(v_i > B | \bigcap_{j=1}^{s} v_j < B, \bigcap_{k=1}^{i-1} v_k > B\Big) \tag{40}$$

which can be further simplified by the lower bounded

$$\geq 1 - \sum_{i=1}^{s} P(v_i > B) - \sum_{i=s+1}^{|A|} P(v_i < B) \tag{41}$$

c) Ranking Individuals: Given a ranking of estimated indicator values $\hat{v}_1 < \hat{v}_2 \cdots < \hat{v}_{|A|}$, where $\hat{v}_i := \hat{I}_h^k(a_i, A, R)$ as before. We are interested in finding the confidence C_r, that this ranking represents the ranking of the exact values v_i. To this end, we introduce multiple thresholds B_i with $1 \leq i \leq |A|-1$ between each pair $\hat{v}_i$ and $\hat{v}_{i+1}$ of consecutive estimates (see right hand side of Figure 15). Hence, $B_i = (\hat{v}_i + \hat{v}_{i+1})/2$.

Using B_i, a lower bound is

$$C_r = P(\forall i \leq j : \; v_i < v_j) \tag{42}$$

$$\geq P(v_1 < B_1) \prod_{i=2}^{|A|-1} \left(P(v_i > B_i) \cdot P(v_i < B_{i+1})\right) \tag{43}$$

$$\cdot P(v_{|A|} > B_{|A|-1}) \tag{44}$$

which is again lower bounded by

$$\geq 1 - \sum_{i=1}^{|A|-1} P(v_i < B_i) - \sum_{i=2}^{|A|} P(v_i > B_{i-1}) \tag{45}$$

Table V: Comparison of HypE to different MOEAs with respect to the hypervolume indicator. The first number represents the performance score P, which stands for the number of participants significantly dominating the selected algorithm. The number in brackets denote the hypervolume value, normalized to the minimum and maximum value observed on the test problem.

	Problem	SHV	IBEA	NSGA-II	RS	SPEA2	HypE	HypE*
2 objectives	DTLZ 1	3 *(0.286)*	0 *(0.667)*	2 *(0.441)*	3 *(0.306)*	3 *(0.343)*	1 *(0.545)*	3 *(0.279)*
	DTLZ 2	2 *(0.438)*	0 *(0.871)*	5 *(0.306)*	5 *(0.278)*	2 *(0.431)*	1 *(0.682)*	4 *(0.362)*
	DTLZ 3	6 *(0.265)*	0 *(0.759)*	1 *(0.596)*	3 *(0.452)*	1 *(0.578)*	3 *(0.454)*	2 *(0.483)*
	DTLZ 4	1 *(0.848)*	0 *(0.928)*	3 *(0.732)*	3 *(0.834)*	2 *(0.769)*	1 *(0.779)*	3 *(0.711)*
	DTLZ 5	2 *(0.489)*	0 *(0.931)*	5 *(0.361)*	6 *(0.279)*	2 *(0.463)*	1 *(0.724)*	4 *(0.428)*
	DTLZ 6	2 *(0.670)*	0 *(0.914)*	5 *(0.326)*	4 *(0.388)*	6 *(0.229)*	1 *(0.856)*	2 *(0.659)*
	DTLZ 7	0 *(0.945)*	1 *(0.898)*	6 *(0.739)*	2 *(0.818)*	4 *(0.817)*	2 *(0.853)*	1 *(0.876)*
	Knapsack	2 *(0.523)*	0 *(0.631)*	0 *(0.603)*	3 *(0.493)*	0 *(0.574)*	0 *(0.633)*	0 *(0.630)*
	WFG 1	4 *(0.567)*	0 *(0.949)*	1 *(0.792)*	6 *(0.160)*	1 *(0.776)*	2 *(0.744)*	4 *(0.557)*
	WFG 2	1 *(0.987)*	4 *(0.962)*	3 *(0.974)*	6 *(0.702)*	4 *(0.969)*	0 *(0.990)*	0 *(0.989)*
	WFG 3	2 *(0.994)*	0 *(0.997)*	4 *(0.991)*	6 *(0.559)*	4 *(0.990)*	0 *(0.997)*	2 *(0.994)*
	WFG 4	0 *(0.964)*	0 *(0.969)*	4 *(0.891)*	6 *(0.314)*	4 *(0.898)*	0 *(0.968)*	0 *(0.963)*
	WFG 5	3 *(0.994)*	0 *(0.997)*	5 *(0.992)*	6 *(0.402)*	2 *(0.995)*	0 *(0.998)*	2 *(0.995)*
	WFG 6	2 *(0.945)*	0 *(0.975)*	4 *(0.932)*	6 *(0.418)*	4 *(0.930)*	1 *(0.955)*	2 *(0.942)*
	WFG 7	3 *(0.929)*	0 *(0.988)*	1 *(0.946)*	6 *(0.294)*	2 *(0.939)*	1 *(0.947)*	4 *(0.920)*
	WFG 8	3 *(0.431)*	0 *(0.675)*	1 *(0.536)*	3 *(0.367)*	1 *(0.514)*	0 *(0.683)*	1 *(0.549)*
	WFG 9	1 *(0.920)*	0 *(0.939)*	4 *(0.891)*	6 *(0.313)*	4 *(0.878)*	1 *(0.924)*	0 *(0.931)*
3 objectives	DTLZ 1	3 *(0.313)*	1 *(0.505)*	6 *(0.168)*	0 *(0.607)*	5 *(0.275)*	1 *(0.395)*	3 *(0.336)*
	DTLZ 2	2 *(0.995)*	0 *(0.998)*	5 *(0.683)*	6 *(0.491)*	4 *(0.888)*	1 *(0.996)*	3 *(0.994)*
	DTLZ 3	3 *(0.210)*	1 *(0.495)*	3 *(0.179)*	0 *(0.679)*	3 *(0.216)*	2 *(0.398)*	3 *(0.196)*
	DTLZ 4	1 *(0.945)*	0 *(0.989)*	3 *(0.777)*	3 *(0.774)*	2 *(0.860)*	0 *(0.987)*	2 *(0.922)*
	DTLZ 5	1 *(0.991)*	0 *(0.994)*	5 *(0.696)*	6 *(0.374)*	4 *(0.882)*	2 *(0.990)*	3 *(0.989)*
	DTLZ 6	2 *(0.971)*	0 *(0.990)*	6 *(0.151)*	5 *(0.237)*	4 *(0.266)*	0 *(0.991)*	3 *(0.967)*
	DTLZ 7	0 *(0.993)*	1 *(0.987)*	6 *(0.633)*	4 *(0.794)*	5 *(0.722)*	3 *(0.970)*	2 *(0.980)*
	Knapsack	2 *(0.441)*	0 *(0.544)*	1 *(0.462)*	6 *(0.322)*	1 *(0.441)*	0 *(0.550)*	0 *(0.473)*
	WFG 1	4 *(0.792)*	3 *(0.811)*	3 *(0.827)*	6 *(0.207)*	1 *(0.881)*	0 *(0.985)*	1 *(0.894)*
	WFG 2	0 *(0.556)*	3 *(0.475)*	3 *(0.406)*	6 *(0.261)*	2 *(0.441)*	0 *(0.446)*	0 *(0.372)*
	WFG 3	2 *(0.995)*	3 *(0.981)*	4 *(0.966)*	6 *(0.689)*	4 *(0.966)*	0 *(0.999)*	1 *(0.998)*
	WFG 4	0 *(0.978)*	3 *(0.955)*	5 *(0.708)*	6 *(0.220)*	4 *(0.740)*	1 *(0.975)*	0 *(0.979)*
	WFG 5	2 *(0.988)*	3 *(0.952)*	4 *(0.884)*	6 *(0.343)*	5 *(0.877)*	0 *(0.991)*	0 *(0.991)*
	WFG 6	2 *(0.959)*	2 *(0.955)*	4 *(0.914)*	6 *(0.415)*	5 *(0.879)*	0 *(0.987)*	1 *(0.981)*
	WFG 7	1 *(0.965)*	3 *(0.950)*	5 *(0.770)*	6 *(0.183)*	4 *(0.858)*	0 *(0.988)*	2 *(0.958)*
	WFG 8	2 *(0.887)*	0 *(0.922)*	4 *(0.842)*	6 *(0.301)*	5 *(0.780)*	0 *(0.906)*	3 *(0.870)*
	WFG 9	1 *(0.954)*	3 *(0.914)*	5 *(0.735)*	6 *(0.283)*	4 *(0.766)*	0 *(0.972)*	1 *(0.956)*

continued on next page

continued from previous page

	Problem	SHV	IBEA	NSGA-II	RS	SPEA2	HypE	HypE*
5 objectives	DTLZ 1	2 *(0.927)*	3 *(0.905)*	5 *(0.831)*	6 *(0.548)*	4 *(0.869)*	0 *(0.968)*	1 *(0.961)*
	DTLZ 2	1 *(0.998)*	0 *(0.999)*	4 *(0.808)*	6 *(0.324)*	5 *(0.795)*	2 *(0.998)*	3 *(0.998)*
	DTLZ 3	2 *(0.754)*	1 *(0.786)*	6 *(0.365)*	4 *(0.529)*	4 *(0.520)*	0 *(0.824)*	1 *(0.768)*
	DTLZ 4	1 *(0.997)*	0 *(0.998)*	4 *(0.749)*	5 *(0.558)*	6 *(0.537)*	2 *(0.992)*	2 *(0.992)*
	DTLZ 5	0 *(0.997)*	0 *(0.998)*	4 *(0.854)*	6 *(0.403)*	5 *(0.841)*	2 *(0.996)*	2 *(0.995)*
	DTLZ 6	3 *(0.964)*	1 *(0.979)*	5 *(0.428)*	6 *(0.311)*	4 *(0.597)*	0 *(0.988)*	1 *(0.977)*
	DTLZ 7	0 *(0.988)*	0 *(0.986)*	6 *(0.478)*	4 *(0.672)*	5 *(0.569)*	2 *(0.868)*	2 *(0.862)*
	Knapsack	0 *(0.676)*	0 *(0.862)*	2 *(0.163)*	2 *(0.235)*	1 *(0.369)*	2 *(0.242)*	2 *(0.256)*
	WFG 1	4 *(0.766)*	5 *(0.703)*	2 *(0.832)*	6 *(0.291)*	2 *(0.820)*	0 *(0.973)*	1 *(0.951)*
	WFG 2	0 *(0.671)*	0 *(0.533)*	0 *(0.644)*	6 *(0.351)*	0 *(0.624)*	0 *(0.557)*	3 *(0.503)*
	WFG 3	6 *(0.339)*	0 *(0.974)*	3 *(0.946)*	5 *(0.760)*	4 *(0.932)*	0 *(0.977)*	0 *(0.971)*
	WFG 4	0 *(0.965)*	3 *(0.894)*	5 *(0.711)*	6 *(0.241)*	4 *(0.741)*	1 *(0.948)*	1 *(0.949)*
	WFG 5	5 *(0.754)*	1 *(0.971)*	4 *(0.892)*	6 *(0.303)*	3 *(0.911)*	0 *(0.978)*	1 *(0.975)*
	WFG 6	0 *(0.953)*	0 *(0.949)*	4 *(0.913)*	6 *(0.392)*	5 *(0.872)*	1 *(0.948)*	2 *(0.940)*
	WFG 7	0 *(0.921)*	1 *(0.822)*	2 *(0.774)*	6 *(0.157)*	4 *(0.745)*	2 *(0.784)*	5 *(0.700)*
	WFG 8	0 *(0.847)*	0 *(0.856)*	4 *(0.685)*	6 *(0.309)*	5 *(0.588)*	2 *(0.825)*	3 *(0.809)*
	WFG 9	5 *(0.496)*	2 *(0.720)*	4 *(0.645)*	6 *(0.138)*	3 *(0.667)*	0 *(0.937)*	0 *(0.956)*
7 objectives	DTLZ 1	2 *(0.962)*	2 *(0.960)*	5 *(0.950)*	6 *(0.563)*	2 *(0.961)*	0 *(0.995)*	0 *(0.995)*
	DTLZ 2	3 *(0.998)*	0 *(1.000)*	5 *(0.808)*	6 *(0.340)*	4 *(0.850)*	1 *(0.999)*	1 *(0.999)*
	DTLZ 3	1 *(0.951)*	1 *(0.958)*	5 *(0.589)*	6 *(0.438)*	4 *(0.723)*	0 *(0.973)*	1 *(0.952)*
	DTLZ 4	1 *(0.999)*	0 *(1.000)*	4 *(0.902)*	6 *(0.569)*	5 *(0.814)*	2 *(0.999)*	2 *(0.999)*
	DTLZ 5	1 *(0.997)*	0 *(0.997)*	4 *(0.888)*	6 *(0.502)*	4 *(0.899)*	0 *(0.997)*	1 *(0.997)*
	DTLZ 6	3 *(0.954)*	2 *(0.983)*	5 *(0.635)*	6 *(0.397)*	4 *(0.756)*	0 *(0.993)*	1 *(0.988)*
	DTLZ 7	0 *(0.981)*	1 *(0.958)*	5 *(0.348)*	4 *(0.559)*	5 *(0.352)*	2 *(0.877)*	2 *(0.870)*
	Knapsack	0 *(0.745)*	0 *(0.768)*	2 *(0.235)*	2 *(0.226)*	2 *(0.272)*	2 *(0.276)*	4 *(0.212)*
	WFG 1	4 *(0.647)*	5 *(0.649)*	2 *(0.814)*	6 *(0.189)*	2 *(0.812)*	0 *(0.956)*	1 *(0.937)*
	WFG 2	0 *(0.632)*	0 *(0.747)*	1 *(0.409)*	5 *(0.155)*	0 *(0.837)*	0 *(0.528)*	0 *(0.630)*
	WFG 3	6 *(0.105)*	2 *(0.975)*	3 *(0.961)*	5 *(0.709)*	4 *(0.958)*	0 *(0.983)*	0 *(0.982)*
	WFG 4	3 *(0.888)*	2 *(0.919)*	4 *(0.688)*	6 *(0.200)*	4 *(0.694)*	0 *(0.956)*	0 *(0.952)*
	WFG 5	6 *(0.042)*	2 *(0.982)*	4 *(0.905)*	5 *(0.406)*	3 *(0.938)*	0 *(0.986)*	0 *(0.987)*
	WFG 6	0 *(0.978)*	0 *(0.967)*	4 *(0.940)*	6 *(0.453)*	5 *(0.921)*	0 *(0.974)*	3 *(0.967)*
	WFG 7	1 *(0.688)*	3 *(0.657)*	0 *(0.813)*	6 *(0.207)*	3 *(0.658)*	1 *(0.713)*	5 *(0.606)*
	WFG 8	0 *(0.933)*	1 *(0.905)*	4 *(0.709)*	6 *(0.366)*	5 *(0.537)*	2 *(0.863)*	2 *(0.874)*
	WFG 9	5 *(0.385)*	2 *(0.681)*	3 *(0.679)*	6 *(0.119)*	3 *(0.683)*	0 *(0.928)*	0 *(0.943)*
10 objectives	DTLZ 1	3 *(0.981)*	5 *(0.971)*	4 *(0.986)*	6 *(0.590)*	2 *(0.990)*	0 *(0.999)*	0 *(0.999)*
	DTLZ 2	3 *(0.999)*	2 *(1.000)*	5 *(0.825)*	6 *(0.290)*	4 *(0.868)*	0 *(1.000)*	0 *(1.000)*
	DTLZ 3	3 *(0.951)*	1 *(0.990)*	5 *(0.676)*	6 *(0.358)*	4 *(0.750)*	0 *(0.994)*	1 *(0.990)*
	DTLZ 4	2 *(1.000)*	0 *(1.000)*	4 *(0.988)*	6 *(0.560)*	5 *(0.960)*	1 *(1.000)*	0 *(1.000)*
	DTLZ 5	3 *(0.951)*	0 *(0.998)*	4 *(0.899)*	6 *(0.471)*	4 *(0.892)*	0 *(0.998)*	1 *(0.997)*
	DTLZ 6	4 *(0.497)*	2 *(0.987)*	4 *(0.706)*	6 *(0.276)*	3 *(0.769)*	0 *(0.994)*	1 *(0.992)*
	DTLZ 7	0 *(0.986)*	1 *(0.831)*	4 *(0.137)*	6 *(0.057)*	4 *(0.166)*	2 *(0.744)*	1 *(0.781)*
	Knapsack	0 *(0.568)*	0 *(0.529)*	2 *(0.149)*	4 *(0.119)*	2 *(0.173)*	5 *(0.068)*	5 *(0.060)*
	WFG 1	6 *(0.402)*	4 *(0.843)*	2 *(0.932)*	5 *(0.562)*	2 *(0.937)*	0 *(0.977)*	0 *(0.975)*
	WFG 2	0 *(0.971)*	0 *(0.988)*	0 *(0.978)*	5 *(0.020)*	2 *(0.962)*	0 *(0.981)*	1 *(0.966)*
	WFG 3	6 *(0.088)*	1 *(0.973)*	3 *(0.947)*	5 *(0.792)*	4 *(0.933)*	0 *(0.980)*	1 *(0.976)*
	WFG 4	3 *(0.698)*	2 *(0.896)*	3 *(0.708)*	6 *(0.207)*	5 *(0.669)*	0 *(0.950)*	0 *(0.955)*
	WFG 5	6 *(0.014)*	2 *(0.979)*	4 *(0.832)*	5 *(0.365)*	3 *(0.913)*	0 *(0.987)*	0 *(0.989)*
	WFG 6	3 *(0.934)*	1 *(0.949)*	4 *(0.896)*	6 *(0.449)*	5 *(0.865)*	0 *(0.959)*	1 *(0.949)*
	WFG 7	1 *(0.686)*	4 *(0.464)*	1 *(0.604)*	6 *(0.077)*	4 *(0.473)*	0 *(0.683)*	3 *(0.548)*
	WFG 8	0 *(0.956)*	1 *(0.903)*	4 *(0.689)*	6 *(0.221)*	5 *(0.438)*	2 *(0.883)*	2 *(0.875)*
	WFG 9	5 *(0.222)*	3 *(0.584)*	3 *(0.644)*	6 *(0.109)*	2 *(0.676)*	1 *(0.893)*	0 *(0.925)*

continued on next page

continued from previous page

	Problem	SHV	IBEA	NSGA-II	RS	SPEA2	HypE	HypE*
25 objectives	DTLZ 1	4 *(0.994)*	5 *(0.987)*	2 *(1.000)*	6 *(0.657)*	3 *(0.998)*	0 *(1.000)*	0 *(1.000)*
	DTLZ 2	3 *(0.999)*	2 *(1.000)*	4 *(0.965)*	6 *(0.301)*	5 *(0.882)*	0 *(1.000)*	0 *(1.000)*
	DTLZ 3	3 *(0.967)*	2 *(0.999)*	4 *(0.930)*	6 *(0.455)*	5 *(0.827)*	0 *(0.999)*	0 *(1.000)*
	DTLZ 4	3 *(1.000)*	2 *(1.000)*	4 *(1.000)*	6 *(0.546)*	5 *(0.991)*	0 *(1.000)*	0 *(1.000)*
	DTLZ 5	5 *(0.781)*	2 *(0.996)*	3 *(0.949)*	6 *(0.457)*	4 *(0.808)*	0 *(0.999)*	1 *(0.999)*
	DTLZ 6	6 *(0.286)*	2 *(0.993)*	3 *(0.957)*	5 *(0.412)*	4 *(0.830)*	0 *(0.999)*	0 *(0.998)*
	DTLZ 7	0 *(0.973)*	0 *(0.966)*	3 *(0.856)*	2 *(0.893)*	4 *(0.671)*	2 *(0.889)*	3 *(0.825)*
	Knapsack	0 *(0.000)*	4 *(0.000)*	5 *(0.000)*	3 *(0.000)*	6 *(0.000)*	1 *(0.000)*	2 *(0.000)*
	WFG 1	6 *(0.183)*	4 *(0.930)*	0 *(0.971)*	5 *(0.815)*	3 *(0.965)*	0 *(0.972)*	0 *(0.973)*
	WFG 2	0 *(0.951)*	0 *(0.951)*	2 *(0.935)*	6 *(0.072)*	2 *(0.933)*	2 *(0.934)*	2 *(0.928)*
	WFG 3	6 *(0.037)*	0 *(0.983)*	3 *(0.965)*	5 *(0.758)*	3 *(0.963)*	1 *(0.974)*	1 *(0.977)*
	WFG 4	6 *(0.063)*	2 *(0.890)*	3 *(0.541)*	5 *(0.170)*	4 *(0.432)*	0 *(0.941)*	0 *(0.945)*
	WFG 5	6 *(0.003)*	3 *(0.832)*	4 *(0.796)*	5 *(0.227)*	2 *(0.915)*	0 *(0.989)*	0 *(0.989)*
	WFG 6	3 *(0.932)*	0 *(0.959)*	5 *(0.913)*	6 *(0.579)*	3 *(0.926)*	0 *(0.961)*	0 *(0.962)*
	WFG 7	3 *(0.286)*	4 *(0.183)*	2 *(0.386)*	6 *(0.081)*	4 *(0.185)*	0 *(0.707)*	1 *(0.479)*
	WFG 8	0 *(0.924)*	0 *(0.909)*	4 *(0.517)*	6 *(0.189)*	5 *(0.305)*	2 *(0.817)*	3 *(0.792)*
	WFG 9	5 *(0.118)*	3 *(0.531)*	3 *(0.580)*	5 *(0.133)*	2 *(0.681)*	0 *(0.893)*	1 *(0.848)*
50 objectives	DTLZ 1	4 *(0.992)*	5 *(0.985)*	2 *(1.000)*	6 *(0.566)*	3 *(0.999)*	1 *(1.000)*	0 *(1.000)*
	DTLZ 2	3 *(1.000)*	2 *(1.000)*	4 *(0.998)*	6 *(0.375)*	5 *(0.917)*	0 *(1.000)*	0 *(1.000)*
	DTLZ 3	3 *(0.984)*	2 *(1.000)*	3 *(0.988)*	6 *(0.518)*	5 *(0.891)*	0 *(1.000)*	0 *(1.000)*
	DTLZ 4	2 *(1.000)*	2 *(1.000)*	4 *(1.000)*	6 *(0.517)*	5 *(0.999)*	0 *(1.000)*	0 *(1.000)*
	DTLZ 5	5 *(0.477)*	2 *(0.996)*	3 *(0.954)*	5 *(0.425)*	4 *(0.752)*	0 *(0.999)*	0 *(0.999)*
	DTLZ 6	6 *(0.112)*	2 *(0.995)*	3 *(0.979)*	5 *(0.399)*	4 *(0.839)*	0 *(0.998)*	1 *(0.998)*
	DTLZ 7	1 *(0.767)*	0 *(0.966)*	5 *(0.233)*	4 *(0.254)*	6 *(0.020)*	2 *(0.684)*	3 *(0.675)*
	Knapsack	0 *(0.000)*	4 *(0.000)*	5 *(0.000)*	3 *(0.000)*	6 *(0.000)*	1 *(0.000)*	2 *(0.000)*
	WFG 1	6 *(0.210)*	4 *(0.869)*	2 *(0.962)*	4 *(0.823)*	2 *(0.961)*	0 *(0.971)*	0 *(0.970)*
	WFG 2	3 *(0.538)*	0 *(0.962)*	0 *(0.959)*	6 *(0.076)*	0 *(0.952)*	2 *(0.945)*	3 *(0.943)*
	WFG 3	6 *(0.059)*	0 *(0.981)*	2 *(0.972)*	5 *(0.731)*	2 *(0.973)*	0 *(0.976)*	0 *(0.979)*
	WFG 4	6 *(0.011)*	2 *(0.783)*	3 *(0.268)*	5 *(0.118)*	3 *(0.258)*	0 *(0.944)*	1 *(0.908)*
	WFG 5	6 *(0.003)*	2 *(0.940)*	4 *(0.789)*	5 *(0.416)*	3 *(0.913)*	1 *(0.987)*	0 *(0.989)*
	WFG 6	4 *(0.933)*	2 *(0.963)*	4 *(0.941)*	6 *(0.663)*	2 *(0.961)*	0 *(0.974)*	0 *(0.976)*
	WFG 7	1 *(0.312)*	5 *(0.026)*	3 *(0.208)*	5 *(0.022)*	4 *(0.034)*	0 *(0.581)*	1 *(0.378)*
	WFG 8	1 *(0.669)*	0 *(0.913)*	4 *(0.341)*	6 *(0.147)*	5 *(0.233)*	1 *(0.602)*	2 *(0.579)*
	WFG 9	5 *(0.250)*	3 *(0.597)*	3 *(0.559)*	6 *(0.166)*	2 *(0.727)*	0 *(0.907)*	0 *(0.903)*