Approximating the least hypervolume contributor: NP-hard in general, but fast in practice

Karl Bringmann[1] and Tobias Friedrich[2]

[1] Universität des Saarlandes, Saarbrücken, Germany
[2] International Computer Science Institute, Berkeley, CA, USA

Abstract. The hypervolume indicator is an increasingly popular set measure to compare the quality of two Pareto sets. The basic ingredient of most hypervolume indicator based optimization algorithms is the calculation of the hypervolume contribution of single solutions regarding a Pareto set. We show that exact calculation of the hypervolume contribution is **#P**-hard while its approximation is **NP**-hard. The same holds for the calculation of the minimal contribution. We also prove that it is **NP**-hard to decide whether a solution has the least hypervolume contribution. Even deciding whether the contribution of a solution is at most $(1+\varepsilon)$ times the minimal contribution is **NP**-hard. This implies that it is neither possible to efficiently find the least contributing solution (unless $\mathbf{P} = \mathbf{NP}$) nor to approximate it (unless $\mathbf{NP} = \mathbf{BPP}$).
Nevertheless, in the second part of the paper we present a very fast approximation algorithm for this problem. We prove that for arbitrarily given $\varepsilon, \delta > 0$ it calculates a solution with contribution at most $(1 + \varepsilon)$ times the minimal contribution with probability at least $(1-\delta)$. Though it cannot run in polynomial time for all instances, it performs extremely fast on various benchmark datasets. The algorithm solves very large problem instances which are intractable for exact algorithms (e.g., 10000 solutions in 100 dimensions) within a few seconds.

1 Introduction

Multi-objective optimization deals with the task of optimizing several objective functions at the same time. As these functions are often conflicting, we cannot aim for a single optimal solution but for a set of Pareto optimal solutions. Unfortunately, the Pareto set frequently grows exponentially in the problem size. In this case, it is not possible to compute the whole front efficiently and the goal is to compute a good approximation of the Pareto front.

There are many indicators to measure the quality of a Pareto set, but there is only one widely used that is strictly Pareto compliant [22], namely the hypervolume indicator. Strictly Pareto compliant means that given two Pareto sets A and B the indicator values A higher than B if the Pareto set A dominates the Pareto set B. The hypervolume (HYP) measures the volume of the dominated portion of the objective space. It was first proposed and employed for multi-objective optimization by Zitzler and Thiele [20].

It has become very popular recently and several algorithms have been developed to calculate it. The first one was the Hypervolume by Slicing Objectives (HSO) algorithm which was suggested independently by Zitzler [19] and Knowles [9]. To improve its runtime on practical instances, various speed up heuristics of HSO have been suggested [16, 18]. The currently best asymptotic runtime of $O(n \log n + n^{d/2})$ is obtained by Beume and Rudolph [2] by an adaption of Overmars and Yap's algorithm [11] for Klee's Measure Problem [8].

From a geometric perspective, the hypervolume indicator is just measuring the volume of the union of a certain kind of boxes in $\mathbb{R}^d_{\geq 0}$, namely of boxes which share the reference point[1] as a common point. We will use the terms point and box interchangeably for solutions as the dominated volume of a point defines a box and vice versa. Given a set M of n points in $\mathbb{R}^d$, we define the hypervolume of M to be

$$\mathrm{HYP}(M) := \textsc{vol}\left(\bigcup_{(x_1,\ldots,x_d)\in M} [0, x_1] \times \ldots \times [0, x_d] \right)$$

In [4] the authors have proven that it is **#P**-hard[2] in the number of dimension to calculate HYP precisely. Therefore, all hypervolume algorithms must have an exponential runtime in the number of objectives (unless $\mathbf{P} = \mathbf{NP}$). Without the widely accepted assumption $\mathbf{P} \neq \mathbf{NP}$, the only known lower bound for any d is $\Omega(n \log n)$ [3]. Note that the worst-case combinatorial complexity (i.e., the number of faces of all dimensions on the boundary of the union) of $\Theta(n^d)$ does not imply any bounds on the computational complexity.

Though the **#P**-hardness of HYP dashes the hope for an exact subexponential algorithm, there are a few estimation algorithms [1, 4] for approximating the hypervolume based on Monte Carlo sampling. However, the only approximation algorithm with proven bounds is presented in [4]. There, the authors describe an FPRAS for HYP which gives an ε-approximation of the hypervolume with probability $(1 - \delta)$ in time $\mathcal{O}(\log(1/\delta)\, nd/\varepsilon^2)$.

New complexity results

We will now describe a few problems related to the calculation of the hypervolume indicator and state our results. For this, observe that calculating the hypervolume itself is actually not necessary in a hypervolume-based evolutionary multi-objective optimizer as the algorithm actually only has to find a box with the minimal contribution to the hypervolume.

The *contribution* of a box $x \in M$ to the hypervolume of a set M of boxes is the volume dominated by x and no other element of M. We define the contribution

[1] Without loss of generality we assume the reference point to be 0^d.

[2] **#P** is the analog of **NP** for counting problems. For details see either the original paper by Valiant [15] or the standard textbook on computational complexity by Papadimitriou [12].

$\mathrm{CON}(M, x)$ of x to be

$$\mathrm{CON}(M, x) := \mathrm{HYP}(M) - \mathrm{HYP}(M \setminus x).$$

In Section 2 we show that this problem is **#P**-hard to solve exactly. Furthermore, approximating CON by a factor of $2^{d^{1-\varepsilon}}$ is **NP**-hard for any $\varepsilon > 0$. Hence, CON is not approximable. Note that this is no contradiction to the above-mentioned FPRAS for HYPas an approximation of HYP yields no approximation of CON.

As a hypervolume-based optimizer is only interested in the box with the *minimal contribution*, we also consider the following problem. Given a set M of n boxes in $\mathbb{R}^d$, find the least contribution of any box in M, that is,

$$\mathrm{MINCON}(M) := \min_{x \in M} \mathrm{CON}(M, x).$$

The reduction in Section 2 shows that MINCON is **#P**-hard and not approximable, even if we know the box which is the least contributor.

Both mentioned problems can be used to find the box contributing the least hypervolume, but their hardness does not imply hardness of the problem itself, which we are trying to solve, namely calculating *which box has the least contribution*. Therefore we also examine the following problem. Given a set M of n boxes in $\mathbb{R}^d$, we want to find a box with the least contribution in M, that is,

$$\mathrm{LC}(M) := \operatorname*{argmin}_{x \in M} \mathrm{CON}(M, x).$$

If there are multiple boxes with the same (minimal) contribution, we are, of course, satisfied with any of them. In Section 2 we prove that this problem is **NP**-hard to decide.

However, for practical purposes it most often suffices to solve a relaxed version of the above problem. That is, we just need to find a box which contributes not much more than the minimal contribution, meaning that is is only a $(1 + \varepsilon)$ factor away. If we then throw out such a box, we have an error of at most ε. We will call this ε-LC(M) as it is an "approximation" of the problem LC. Given a set M of n boxes in $\mathbb{R}^d$ and $\varepsilon > 0$, we want to find a box with contribution at most $(1 + \varepsilon)$ times the minimal contribution of any box in M, that is,

$$\mathrm{CON}(M, \varepsilon\text{-LC}(M)) \leq (1 + \varepsilon)\, \mathrm{MINCON}(M).$$

The final result of Section 2 is the **NP**-hardness of ε-LC. This shows, that there is no way of computing the least contributor efficiently, and even no way to approximate it.

New approximation algorithm

In Section 3 we will give a "practical" algorithm for determining a small contributor. Technically speaking, it solves the following problem we call ε-δ-LC(M):

Given a set M of n boxes in $\mathbb{R}^d$, $\varepsilon > 0$ and $\delta > 0$, with probability at least $1 - \delta$ find a box with contribution at most $(1 + \varepsilon)\,\mathrm{MINCON}(M)$.

$$\Pr[\mathrm{CON}(M, \varepsilon\text{-}\delta\text{-}\mathrm{LC}(M)) \leq (1 + \varepsilon)\,\mathrm{MINCON}(M)] \geq 1 - \delta.$$

As we will be able to choose δ arbitrarily, solving this problem is of high practical interest. By the **NP**-hardness of ε-LC there is no way of solving ε-δ-LC efficiently, unless **NP** = **BPP**. This means, our algorithm cannot run in polynomial time for all instances. Its runtime depends on some hardness measure H (cf. Section 3.2), which is an intrinsic property of the given input, but generally unbounded, i.e., not bounded by some function in n and d.

However, in Section 4 we show that our algorithm is practically very fast on various benchmark datasets, even for dimensions completely intractable for exact algorithms like $d = 100$ for which we can solve instances with $n = 10000$ points within seconds. This implies a huge shift in the practical usability of the hypervolume indicator.

2 Hardness of approximation

In this section we first show hardness of approximating MINCON, which we will use afterwards to show hardness of LC and ε-LC. We will reduce #MON-CNF to MINCON, which is the problem of counting the number of satisfying assignments of a Boolean formula in conjunctive normal form in which all variables are unnegated. While the problem of deciding satisfiability of such formula is trivial, counting the number of satisfying assignments is **#P**-hard and even approximating it by a factor of $2^{d^{1-\varepsilon}}$ for any $\varepsilon > 0$ is **NP**-hard, where d is the number of variables (see Roth [14] for a proof).

Theorem 1. MINCON *is* **#P***-hard and approximating it by a factor of* $2^{d^{1-\varepsilon}}$ *is* **NP***-hard for any* $\varepsilon > 0$.

Proof. To show the theorem, we reduce #MON-CNF to MINCON. Let $\square(a_1, \ldots, a_d)$ denote a box $[0, a_1] \times \ldots \times [0, a_d]$. Let $f = \bigwedge_{k=1}^{n} \bigvee_{i \in C_k} x_i$ be a monotone Boolean formula given in CNF with $C_k \subseteq [d] := \{1, \ldots, d\}$, for $k \in [n]$, d the number of variables, n the number of clauses. First, we construct a box $A_k = \square(a_1^k, \ldots, a_d^k, 2^d + 2) \subseteq \mathbb{R}^{d+1}$ for each clause C_k with one vertex at the origin and the opposite vertex at $(a_1^k, \ldots, a_d^k, 2^d + 2)$, where we set

$$a_i^k = \begin{cases} 1, & \text{if} \quad i \in C_k \\ 2, & \text{otherwise} \end{cases}, \quad i \in [d].$$

Additionally, we need a box $B = \square(2, \ldots, 2, 1) \subseteq \mathbb{R}^{d+1}$ and set $M = \{A_1, \ldots, A_n, B\}$. Since we can assume without loss of generality that no clause is dominated by another, meaning $C_i \not\subseteq C_j$ for every $i \neq j$, every box A_k uniquely overlaps a region $[x_1, x_1+1] \times \ldots \times [x_d, x_d+1] \times [1, 2^d+2]$ with $x_i \in \{0, 1\}$, $i \in [d]$,

so that the contribution of every box A_k is greater than 2^d and the contribution of B is at most 2^d, so that B is indeed the least contributor.

Observe that the contribution of B to $\mathrm{HYP}(M)$ can be written as a union of boxes of the form $B_{x_1,\dots,x_d} = [x_1, x_1+1] \times \cdots \times [x_d, x_d+1] \times [0,1]$ with $x_i \in \{0,1\}, i \in [d]$. Moreover, $B_{x_1,\dots,x_d}$ is not a subset of the contribution of B to $\mathrm{HYP}(M)$ iff it is a subset of $\bigcup_{k=1}^{n} A_k$ iff it is a subset of some A_k iff we have $a_i^k \geq x_i + 1$ for $i \in [d]$ iff $a_i^k = 2$ for all i with $x_i = 1$ iff $i \notin C_k$ for all i with $x_i = 1$ iff $(x_1, \dots, x_d)$ satisfies $\bigwedge_{i \in C_k} \neg x_i$ for some k iff $(x_1, \dots, x_d)$ satisfies the negated formula $\bar{f} = \bigvee_{k=1}^{n} \bigwedge_{i \in C_k} \neg x_i$. This implies that $B_{x_1,\dots,x_d}$ is a subset of the contribution of B iff $(x_1, \dots, x_d)$ satisfies f. Hence, since $\mathrm{VOL}(B_{x_1,\dots,x_d}) = 1$, we have $\mathrm{MINCON}(M) = \mathrm{CON}(M, B) = |\{(x_1, \dots, x_d) \in \{0,1\}^d \mid (x_1, \dots, x_d) \text{ satisfies } f\}|$. Thus a polynomial time algorithm solving $\mathrm{MINCON}(M)$ would result in a polynomial time algorithm for #MON-CNF, which proves the claim. □

Note that the reduction from above implies that MINCON is **#P**-hard and **NP**-hard to approximate even if the least contributor is known. Moreover, since we constructed boxes with integer coordinates in $[0, 2^d + 2]$ a number of $b = \mathcal{O}(d^2 n)$ bits suffices to represent all $d+1$ coordinates of the $n+1$ constructed points. Hence, MINCON is hard even if all coordinates are integral. We define as input size $b + n + d$, where b is the number of bits in the input. We will use this result in the next proof. Also note that the same hardness for CON follows immediately, as it is hard to compute $\mathrm{CON}(M, B)$ as constructed above.

By reducing MINCON to LC, one can now show **NP**-hardness of LC. We skip this proof and directly prove **NP**-hardness of ε-LC by using the hardness of approximating MINCON in the following theorem.

Theorem 2. *ε-LC is **NP**-hard for any constant ε. More precisely, it is **NP**-hard for $(1+\varepsilon)$ bounded from above by $2^{d^{1-c}-1}$ for some $c > 0$.*

Proof. We reduce MINCON to ε-LC. Let M be a set of n boxes in $\mathbb{R}^d$, i.e., a problem instance of MINCON represented by a number of b bits, so that the input size is $b + n + d$.

As discussed above, we can assume that the coordinates are integral. We can further assume that $d \geq 2$ as MINCON is trivial for $d = 1$. The minimal contribution of M might be 0, but this occurs if and only if one box in M dominates another. As the latter can be checked in polynomial time, we can without loss of generality also assume that $\mathrm{MINCON}(M) > 0$.

Now, let V be the volume of the bounding box of all the boxes in M, i.e., the product of all maximal coordinates in the d dimensions. We know that V is an integer with $1 \leq V \leq 2^b$, as there are only b bits in the input.

We now define a slightly modified set of boxes:

$$
\begin{aligned}
A &= \{\square(a_1 + 2V, a_2, \dots, a_d) \mid \square(a_1, \dots, a_d) \in M\},\\
B &= \square(2V, \dots, 2V),\\
C_\lambda &= \square(1, \dots, 1, 2V + \lambda),\\
M_\lambda &= A \cup \{B\} \cup \{C_\lambda\}.
\end{aligned}
$$

The boxes in A are the boxes of M, but shifted along the x_1-axis. By definition, $a_i \leq V$, $i \in [d]$ for all $\square(a_1, \ldots, a_d) \in M$. The contribution to $\mathrm{HYP}(M_\lambda)$ of a box in A is the same as the contribution to $\mathrm{HYP}(M)$ of the corresponding box in M as the additional part is overlapped by the "blocking" box B. Also note that the contribution of a box in A is less or equal than V.

The box B uniquely overlaps at least the space $[V, 2V] \times \ldots \times [V, 2V]$ (as every coordinate of a point in M is less than equal to V) which has volume at least V. Hence, B is never the least contributor of M_λ. The box C_λ then has a contribution of $\textsc{vol}([0,1] \times \ldots \times [0,1] \times [2V, 2V + \lambda]) = \lambda$, so that C_λ is a least contributor iff λ is less than or equal to the minimal contribution of any box in A to $\mathrm{HYP}(M_\lambda)$ which holds iff we have $\lambda \leq \mathrm{MINCON}(M)$.

Since we can decide, whether C_λ is the least contributor, by one call to $\mathrm{LC}(M_\lambda)$, we can do kind of a binary search on λ. As we are interested in a multiplicative approximation, we search for $\kappa := \log_2(\lambda)$ to be the largest value less than equal to $\log_2(\mathrm{MINCON}(M))$, where κ now is an integer in the range $[0, b]$. As we can only answer ε-LC-queries we cannot do exact binary search. But we can still follow its lines, recurring on the left half of the current interval, if for the median value κ_m we get $\varepsilon\text{-LC}(M_{\lambda_\mathrm{m}}) = C_{\lambda_\mathrm{m}}$, where $\lambda_\mathrm{m} = 2^{\kappa_\mathrm{m}}$, and on the right half, if we get any other result.

The incorrectness of ε-LC may misguide our search, but since we have $\mathrm{CON}(M, \varepsilon\text{-LC}(M)) \leq (1+\varepsilon)\,\mathrm{MINCON}(M)$ it can give a wrong answer (i.e., not the least contributor) only if we have $(1+\varepsilon)^{-1}\mathrm{MINCON}(M) \leq 2^\kappa \leq (1+\varepsilon)\,\mathrm{MINCON}(M)$. Outside of this interval our search goes perfectly well. Thus, after the binary search, i.e, after at most $\lceil \log_2(b) \rceil$ many calls to ε-LC, we end up at a value κ which is either inside the above interval (in which case we are satisfied) or the largest integer smaller than $\log_2((1+\varepsilon)^{-1}\mathrm{MINCON}(M))$ or the smallest integer greater than $\log_2((1+\varepsilon)\,\mathrm{MINCON}(M))$. Hence, we have $\kappa \leq \log_2((1+\varepsilon)\,\mathrm{MINCON}(M)) + 1$ implying $\lambda = 2^\kappa \leq 2(1+\varepsilon)\,\mathrm{MINCON}(M)$. Analogously, we get $\lambda = 2^\kappa \geq \mathrm{MINCON}(M)/(2(1+\varepsilon))$. Therefore after $\mathcal{O}(\log(b))$ many calls to ε-LC we get a $2\,(1+\varepsilon)$ approximation of $\mathrm{MINCON}(M)$. Since this is **NP**-hard for $2\,(1+\varepsilon)$ bounded from above by $2^{d^{1-c}}$ for some $c > 0$, we showed **NP**-hardness of ε-LC in this case. Note that this includes any constant ε. □

The **NP**-hardness of ε-LC not only implies **NP**-hardness of LC, but also the non-existence of an efficient algorithm for ε-δ-LC unless $\mathbf{NP} = \mathbf{BPP}$. The above proof also gives a very good intuition about the problem ε-LC: As we can approximate the minimal contribution by a small number of calls to ε-LC, there cannot be a much faster way to solve ε-LC but to approximate the contributions – approximating at least the least contribution can be only a factor of $\mathcal{O}(\log(b))$ slower than solving ε-LC. This motivates the algorithm we present in the next section, which tries to approximate the contributions of the various boxes.

3 Practical approximation algorithm

The last section ruled out the possibility of a worst case efficient algorithm for computing or approximating the least contributor. Nevertheless, we are now presenting an algorithm $\mathcal{A}$ that is "safe" and has a good practical runtime, but no polynomial worst case runtime (as this is not possible). By "safe" we mean that it provably solves ε-δ-LC, i.e., it holds that

$$\Pr[\mathrm{CON}(M, \mathcal{A}(M, \varepsilon, \delta)) \leq (1+\varepsilon)\,\mathrm{MINCON}(M)] \geq 1 - \delta.$$

As the algorithm is going to approximate the contributions, we cannot avoid ε and solve LC directly, as with no $(1+\Delta)$-approximation, for any $\Delta > 0$ we can decide whether two contributions are equal or just nearly equal (and in the latter case which one is greater). We consider an ε around 10^{-2} or 10^{-3} as sufficient for typical instances. This implies for most instances that we return the correct result as there are no two small contributions which are only a $(1+\varepsilon)$-factor apart. For the remaining cases we return at least a box which has contribution at most $(1+\varepsilon)$ times the minimal contribution, which means we make an "error" of ε.

Additionally, the algorithm is going to be a randomized Monte Carlo algorithm, which is why we need the δ and do not always return the correct result. However, we will be able to set $\delta = 10^{-6}$ or even $\delta = 10^{-12}$ without increasing the runtime overly. In the following we will describe algorithm $\mathcal{A}$, prove its correctness and describe its runtime.

3.1 The algorithm $\mathcal{A}$

Our algorithm works as follows. For each box A it determines the minimal bounding box of the space that is uniquely overlapped by the box. To do so we start with the box A itself. Then we iterate over all other boxes B. If B dominates A in all but one dimension, then we can cut the bounding box in the non-dominated dimension. This can be realized in $\mathcal{O}(dn^2)$.

Having the bounding box BB_A of the contribution of A we start to sample randomly in it. For each random point we determine if it is uniquely dominated by A. If we checked $\textsc{noSamples}(A)$ random points and $\textsc{noSuccSamples}(A)$ of them were uniquely dominated by A, then the contribution of A is about $\widetilde{V}_A := \frac{\textsc{noSuccSamples}(A)}{\textsc{noSamples}(A)}\textsc{vol}(\mathrm{BB}_A)$, where $\textsc{vol}(\mathrm{BB}_A)$ denotes the volume of the bounding box of the contribution of A. Additionally, we can give an estimate of the deviation of $\widetilde{V}_A$ from V_A, the correct contribution of A (i.e., $V_A = \mathrm{CON}(M, A)$): Using Chernoff's inequality we get that for

$$\Delta(A) := \sqrt{\frac{\log(2n/\delta)}{2\textsc{noSamples}(A)}}\textsc{vol}(\mathrm{BB}_A) \tag{1}$$

the probability that V_A deviates from $\widetilde{V}_A$ by more than $\Delta(A)$ is small enough.

We would like to sample in the bounding boxes in parallel such that every $\widetilde{V}_A$ deviates about the same Δ. We do this by initializing Δ arbitrarily (e.g., $\Delta = 1$) and then in every iteration decrease Δ by some factor (e.g., $\frac{1}{2}$) and sample in each bounding box until we have $\Delta(A) \leq \Delta$. If we then have at any point two boxes A and B with

$$\widetilde{V}_A - \Delta(A) > \widetilde{V}_B + \Delta(B) \tag{2}$$

we can with good probability assume that A is not a least contributor as we would need to have $\widetilde{V}_A - V_A > \Delta(A)$ or $V_B - \widetilde{V}_B > \Delta(B)$ for A having a less contribution than B (which is necessary for A being the least contributor). Hence, whenever such a situation occurs we can delete A from our race, meaning that we do not have to sample in its bounding box anymore. Note that we never have to compare two arbitrary boxes, but only a box A to the currently smallest box $\widetilde{LC}$, i.e., the box with $\widetilde{V}_{\widetilde{LC}}$ minimal.

We can run this race, deleting boxes if their contribution is clearly too much by the above selection equation until either there is just one box left, in which case we have found the least contributor, or until we have reached a point where we have approximated all contributions well enough. Given an abortion criterion ε we can just return $\widetilde{LC}$ (the box with currently smallest approximated contribution) when we have

$$0 < \frac{\widetilde{V}_{\widetilde{LC}} + \Delta(\widetilde{LC})}{\widetilde{V}_A - \Delta(A)} \leq 1 + \varepsilon,$$

for any box $A \neq \widetilde{LC}$ still in the race. If this equation holds, then we can be quite sure that any box has contribution at least $\frac{1}{1+\varepsilon} V_{\widetilde{LC}}$, and, similarly, all other boxes that are still in the race, too. So, after all, we have solved ε-δ-LC.

Due to space limitations, the proof that the described algorithm is indeed correct has been removed. It can be found in Section 3.3 of [5].

3.2 Runtime

As discussed above, our algorithm needs a runtime of at least $\Omega(dn^2)$. This seems to be the true runtime on many practical instances (cf. Section 4). However, by Theorem 2 we cannot hope for a matching upper bound. In this section we present an upper bound on the runtime depending on some characteristics of the input.

For an upper bound, observe that we have to approximate each box A up to $\Delta = (V_A - \text{MINCON}(M))/4$ to be able to delete it with high probability: At this point, $\widetilde{V}_A \geq V_A - \Delta$ and $\widetilde{V}_B \leq V_B + \Delta$, for B a least contributor, so that $\widetilde{V}_A - \widetilde{V}_B \geq 2\Delta$ with probability at least $1 - \delta/n$. Similarly, we can show that the expected value of Δ where we delete box A is $\Omega(V_A - \text{MINCON}(M))$. By equation (1) we observe that we need a number of

$$\frac{\log(2n/\delta)\text{VOL}(\text{BB}_A)^2}{2\Omega(V_A - \text{MINCON}(M))^2} = \mathcal{O}\left(\frac{\log(n/\delta)\text{VOL}(\text{BB}_A)^2}{(V_A - \text{MINCON}(M))^2}\right)$$

Algorithm 1 $\mathcal{A}(M, \varepsilon, \delta)$ solves ε-δ-LC(M) for a set M of n boxes in $\mathbb{R}^d$ and $\varepsilon, \delta > 0$, i.e., it determines a box $x \in M$ s.t. $\Pr[\mathrm{CON}(M, x) \leq (1 + \varepsilon)\,\mathrm{MINCON}(M)] \geq 1 - \delta$.

```
determine the bounding boxes BB_A for all A ∈ M
initialize noSamples(A) = noSuccSamples(A) = 0 for all A ∈ M
initialize Δ
set S := M
repeat
  set Δ := Δ/2
  for all A ∈ S do
    repeat
      sample a random point in BB_A
      increase noSamples(A) and possibly noSuccSamples(A)
      update Ṽ_A and Δ(A)
    until Δ(A) ≤ Δ
  od
  set L̃C := argmin{Ṽ_A | A ∈ S}
  for all A ∈ S do
    if Ṽ_A − Δ(A) > Ṽ_L̃C + Δ(L̃C) then
      S := S\{A}
    od
  od
until |S| = 1 or 0 < (Ṽ_L̃C + Δ(L̃C)) / (Ṽ_A − Δ(A)) ≤ 1 + ε for any L̃C ≠ A ∈ S
return L̃C
```

samples to delete box A on average. For the least contributor LC, we need $\mathcal{O}\Big(\frac{\log(n/\delta)\,\mathrm{VOL}(\mathrm{BB}_{LC})^2}{(\text{sec-min}(V) - \mathrm{MINCON}(M))^2}\Big)$ many samples until we have finally deleted all other boxes, where sec-min(V) denotes the second smallest contribution of any box in M. Since each sample takes runtime $\mathcal{O}(dn)$ and everything besides the sampling takes much less runtime, we get an overall runtime of $\mathcal{O}(dn\,(n + \log(n/\delta)\,\mathrm{H}))$, where

$$\mathrm{H} = \frac{\mathrm{VOL}(\mathrm{BB}_{LC})^2}{(\text{sec-min}(V) - \mathrm{MINCON}(M))^2} + \sum_{LC \neq A \in S} \frac{\mathrm{VOL}(\mathrm{BB}_A)^2}{(V_A - \mathrm{MINCON}(M))^2}$$

is a certain measure of hardness of the input. This value is unbounded and can even be undefined if there is no unique least contributor. In this case our abortion criterion comes into play: With probability $(1 - \delta)$ after approximating every contribution up to $\Delta = \frac{\varepsilon}{4+2\varepsilon}\mathrm{MINCON}(M)$ we have $\widetilde{V}_{LC} \leq V_{LC} + \Delta$, thus $\widetilde{V}_{\widetilde{LC}} \leq V_{LC} + \Delta$, and $\widetilde{V}_A \geq V_{LC} - \Delta$ for every other box A still in the race. Then we conclude

$$\frac{\widetilde{V}_{\widetilde{LC}} + \Delta(\widetilde{LC})}{\widetilde{V}_A - \Delta(A)} \leq \frac{V_{LC} + 2\Delta}{V_{LC} - 2\Delta} = \frac{1 + 2\frac{\varepsilon}{4+2\varepsilon}}{1 - 2\frac{\varepsilon}{4+2\varepsilon}} = 1 + \varepsilon$$

for every box $\widetilde{LC} \neq A \in S$. Hence, the above defined value for Δ suffices to enforce abortion. Since we get this Δ after $\text{noSamples}(A) = \frac{\log(2n/\delta)\text{vol}(\text{BB}_A)^2}{2(\frac{\varepsilon}{4+2\varepsilon}\text{MINCON}(M))^2}$ samples, this yields another upper bound for the overall number of samples, a still unbounded but always finite value:

$$\mathcal{O}\left(\frac{\log(n/\delta)}{\varepsilon^2 \text{MINCON}(M)^2} \sum_{A \in M} \text{vol}(\text{BB}_A)^2\right)$$

However, for the random testcases that we consider in Section 4 the above defined hardness H is a more realistic measure of runtime as there are never two identical contributions and not too many equally small contributions. There one observes values for H that roughly lie in the interval $[n, 10n]$.

4 Experimental analysis

To demonstrate the performance of the described approximation algorithm for the hypervolume contribution, we have implemented it and measured its performance on different datasets. To yield a practically relevant algorithm, we have implemented several heuristical improvements which are described in detail in Section 3.4 of [5]. The most important for the correct interpretation of the experiments is that we use a classical exact algorithm for small n and d. We now first describe the used benchmark datasets and then our results.

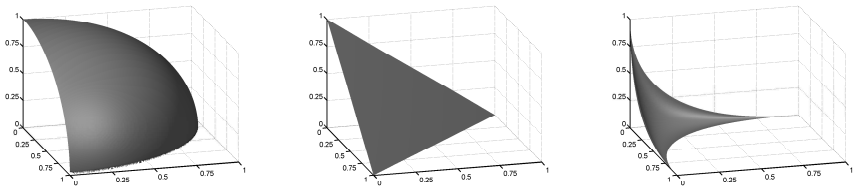
4.1 Datasets

We used five different fronts similar to the DTLZ test suite [7]. As we do not want to compare the hypervolume algorithms for point distributions specific to different optimizers like NSGA-II [6] or SPEA2 [21], we have sampled the points from different surfaces randomly. This allows full scalability of the datasets in the number of points and the number of dimensions.

To define the datasets, we use random variables with two different distributions. Simple uniformly distributed random variables are provided by the build-in random number generator rand() of C++. To get random variables with a Gaussian distribution, we used the polar form of the Box-Muller transformation as described in [13].

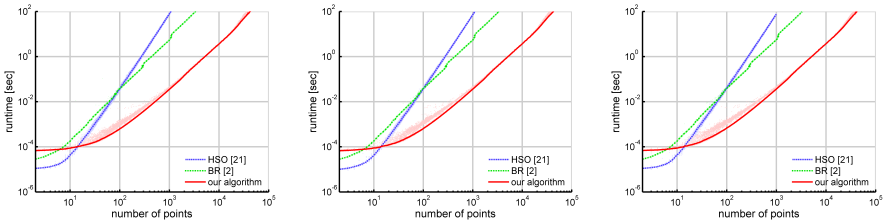
Linear dataset: The first dataset consists of points $(x_1, x_2, \ldots, x_d) \in [0,1]^d$ with $\sum_{i=1}^{d} x_i = 1$. They are obtained by generating d Gaussian random variables y_1, y_2, ..., y_d and then using the normalized points

$$(x_1, x_2, \ldots, x_n) := \frac{(|y_1|, |y_2|, \ldots, |y_n|)}{|y_1| + |y_2| + \ldots + |y_d|}.$$



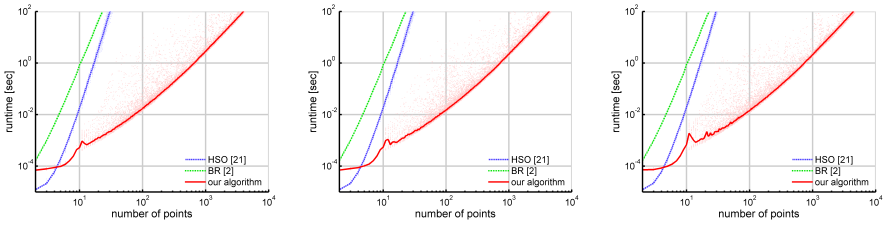
(a) Spherical dataset. (b) Linear dataset. (c) Concave dataset.

Fig. 1. Visualization of the first three datasets.



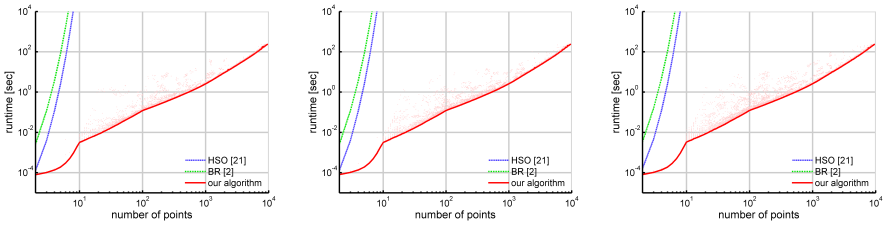
(a) Spherical dataset. (b) Linear dataset. (c) Concave dataset.

Fig. 2. Experimental results for $d = 3$.



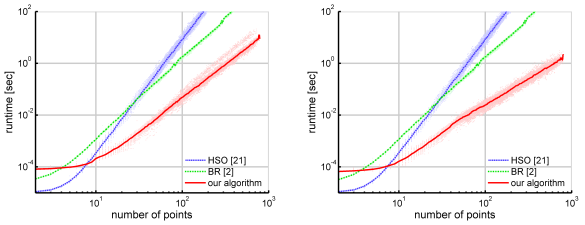
(a) Spherical dataset. (b) Linear dataset. (c) Concave dataset.

Fig. 3. Experimental results for $d = 10$.



(a) Spherical dataset. (b) Linear dataset. (c) Concave dataset.

Fig. 4. Experimental results for $d = 100$.



(a) Random dataset 1. (b) Random dataset 2.

Fig. 5. Experimental results for random datasets with $d = 5$.

Spherical dataset: To obtain uniformly distributed points $(x_1, x_2, \ldots, x_d) \in [0,1]^d$ with $\sum_{i=1}^{d} x_i^2 = 1$ we follow the method of Muller [10]. That is, we generate d Gaussian random variables y_1, y_2, ..., y_d and take the points

$$(x_1, x_2, \ldots, x_n) := \frac{(|y_1|, |y_2|, \ldots, |y_n|)}{\sqrt{y_1^2 + y_2^2 + \ldots + y_d^2}}.$$

Concave dataset: Analogously to the spherical dataset we choose points $(x_1, x_2, \ldots, x_d) \in [0,1]^d$ with $\sum_{i=1}^{d} \sqrt{x_i} = 1$. For this, we generate again d Gaussian random variables y_1, y_2, ..., y_d and use the points

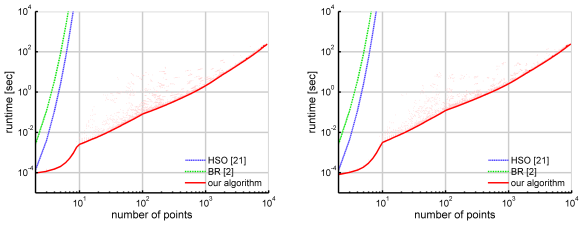
$$(x_1, x_2, \ldots, x_n) := \frac{(|y_1|, |y_2|, \ldots, |y_n|)}{(\sqrt{|y_1|} + \sqrt{|y_2|} + \ldots + \sqrt{|y_d|})^2}.$$

For $d = 3$, the surface of the dataset is shown in Figure 1. Additionally to random points lying on a lower-dimensional surface, we have also examined the following two datasets with points sampled from the actual space similar to the random dataset examined by While et al. [17].

Random dataset 1: We first draw n uniformly distributed points from $[0,1]^d$ and then replace all dominated points by new random points until we have a set of n nondominated points.

Random dataset 2: Very similar to the previous dataset, we choose random points until there are no dominated points. The only difference is that this time the points are not drawn uniformly, but Gaussian distributed in $\mathbb{R}^d$ with mean 1.

Note that the last two datasets are far from being uniformly distributed. The points of the first set all have at least one coordinate very close to 1 while the points of the second set all have at least one coordinate which is significantly above the mean value. This makes their computation for many points (e.g., $n \geq 100$) in small dimensions (e.g., $d \leq 5$) computationally very expensive as it becomes more and more unlikely to sample a nondominated point.



(a) Random dataset 1. (b) Random dataset 2.

Fig. 6. Experimental results for random datasets with $d = 100$.

4.2 Comparison

We have implemented our algorithm in C++ and compared it with the available implementations of HSO by Eckart Zitzler [19] and BR by Nicola Beume [2]. We did not add any further heuristics to both exact algorithms as all published heuristics do not improve the *asymptotic* runtime and even a speedup of a few magnitudes does not change the picture significantly.

It would be better to compare our approximation algorithm with other approximation algorithms instead of exact algorithms. However, the only other published approximation algorithm seems to be [1], which is not publicly available yet. Another reason is that all available optimization algorithms based on the hypervolume indicator use exact calculations and hence our speedup is carried over to them.

All experiments were run on a cluster of 100 machines with two 2.4 GHz AMD Opteron processors, operating in 32-bit mode, running Linux. For our approximation algorithm we used the parameters $\delta = 10^{-6}$ and $\varepsilon = 10^{-2}$. The code used is available upon request and will be distributed from the homepage of the second author.

Figure 2-6 show double-logarithmic plots of the runtime for different datasets and number of dimensions. The shown values are the median of 100 runs each. To illustrate the occurring deviations below and above the median, we also plotted all measured runtimes as ligther single points in the back. As both axes are scaled logarithmically, also the examined problem sizes are distributed logarithmically. That is, we only calculated Pareto sets of size n if $n \in \{\lfloor \exp(k/100) \rfloor \mid k \in \mathbb{N}\}$. We examined dimensions $d = 3, 10, 100$ for the first three datasets and $d = 5, 100$ for the last two datasets.

Independent of the number of solutions and dimension, we always observed that, unless $n \leq 10$, our algorithm outperformed HSO and BR substantially. On the used machines this means that only if the calculation time was insignificant (say, below 10^{-4} seconds), the exact algorithm could compete. On the other hand, the *much* lower median of our algorithm also comes with a much higher empirical standard deviation and interquartile range. In fact, we observed that the upper quartile can be up to five times slower than the median (for the especially degenerated random dataset 1). The highest ratio observed between

the maximum runtime and the average runtime is 66 (again for the random dataset 1). This behavior is represented in the plots by the spread of lighter datapoints in the back of the median. However, there are not too many outliers and even their runtime outperforms HSO and BR. The non-monotonicity of our algorithm around $n = 10$ for $d = 10$ is caused by the approximate for the runtimes of the exact algorithms.

For larger dimensions the advantage of our approximation algorithm becomes tremendous. For $d = 100$ we observed that within 100 seconds our algorithm could solve all problems with less than 6000 solutions while HSO an BR could not solve any problem for a population of 6 solutions in the same time. For example for 7 solutions on the 100-dimensional linear front, HSO needed 13 minutes, BR 7 hours while our algorithm terminated within 0.5 milliseconds.

5 Conclusions

We have proven that most natural questions about the hypervolume contribution which are relevant for evolutionary multi-objective optimizers are not only computationally hard to decide, but also hard to approximate. On the other hand, we have presented a new approximation algorithm which works extremely fast for all tested practical instances. It can solve efficiently large high-dimensional instances ($d \geq 10$, $n \geq 100$) which are intractable for all previous exact algorithms and heuristics.

It would be very interesting to compare the algorithms on further datasets. We believe that only when two solutions have contributions of very close value, our algorithm slows down. For practical instances this should not matter as it simply occurs too rarely – but this conjecture should be substantiated by some broader experimental study in the future.

Bibliography

[1] J. Bader and E. Zitzler. HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. TIK Report 286, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich, 2008.

[2] N. Beume and G. Rudolph. Faster S-metric calculation by considering dominated hypervolume as Klee's measure problem. In *Proc. Second International Conference on Computational Intelligence (IASTED '06)*, pp. 233–238, 2006.

[3] N. Beume, C. Fonseca, M. López-Ibáñez, L. Paquete, and J. Vahrenhold. On the complexity of computing the hypervolume indicator. Technical report CI-235/07, Technical University of Dortmund, 2007.

[4] K. Bringmann and T. Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. In *Proc. 19th International Symposium on Algorithms and Computation (ISAAC '08)*, Vol. 5369 of *Lecture Notes in Computer Science*, pp. 436–447, Gold Coast, Australia, 2008. Springer. Extended version available from http://arxiv.org/abs/0809.0835.

[5] K. Bringmann and T. Friedrich. Approximating the least hypervolume contributor: NP-hard in general, but fast in practice, 2008. Extended version available from http://arxiv.org/abs/0812.2636.

[6] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proc. 6th International Conference Parallel Problem Solving from Nature (PPSN VI)*, Vol. 1917 of *Lecture Notes in Computer Science*, pp. 849–858. Springer, 2000.

[7] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proc. IEEE Congress on Evolutionary Computation (CEC '02)*, pp. 825–830, 2002.

[8] V. Klee. Can the measure of $\bigcup[a_i, b_i]$ be computed in less than $O(n \log n)$ steps? *American Mathematical Monthly*, 84:284–285, 1977.

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

[10] M. E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Commun. ACM*, 2:19–20, 1959.

[11] M. H. Overmars and C.-K. Yap. New upper bounds in Klee's measure problem. *SIAM J. Comput.*, 20:1034–1045, 1991. Announced at *29th Annual Symposium on Foundations of Computer Science (FOCS '88)*.

[12] C. M. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, Reading, MA, 1994.

[13] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, second edition, 1992.

[14] D. Roth. On the hardness of approximate reasoning. *Artif. Intell.*, 82:273–302, 1996.

[15] L. G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[16] R. L. While, L. Bradstreet, L. Barone, and P. Hingston. Heuristics for optimizing the calculation of hypervolume for multi-objective optimization problems. In *Proc. IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 2225–2232, 2005.

[17] R. L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *IEEE Trans. Evolutionary Computation*, 10:29–38, 2006.

[18] X. Zhou, N. Mao, W. Li, and C. Sun. A fast algorithm for computing the contribution of a point to the hypervolume. In *Proc. Third International Conference on Natural Computation (ICNC '07)*, Vol. 4, pp. 415–420, 2007.

[19] E. Zitzler. Hypervolume metric calculation, 2001. Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/hypervol.c.

[20] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evolutionary Computation*, 3:257–271, 1999. Announced at *5th International Conference Parallel Problem Solving from Nature (PPSN V)*.

[21] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K. Giannakoglou et al., editors, *Proc. Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pp. 95–100. International Center for Numerical Methods in Engineering (CIMNE), 2002.

[22] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Trans. Evolutionary Computation*, 7:117–132, 2003.