Parameterized Average-Case Complexity of the Hypervolume Indicator

Karl Bringmann
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Tobias Friedrich
Friedrich-Schiller-Universität
Jena, Germany

ABSTRACT

The hypervolume indicator (HYP) is a popular measure for the quality of a set of n solutions in $\mathbb{R}^d$. We discuss its asymptotic worst-case runtimes and several lower bounds depending on different complexity-theoretic assumptions. Assuming that P $\neq$ NP, there is no algorithm with runtime $\mathrm{poly}(n, d)$. Assuming the exponential time hypothesis, there is no algorithm with runtime $n^{o(d)}$. In contrast to these worst-case lower bounds, we study the average-case complexity of HYP for points distributed i.i.d. at random on a d-dimensional simplex. We present a general framework which translates any algorithm for HYP with worst-case runtime $n^{f(d)}$ to an algorithm with worst-case runtime $n^{f(d)+1}$ and fixed-parameter-tractable (FPT) average-case runtime. This can be used to show that HYP can be solved in expected time $\mathcal{O}(d^{d^2/2}\, n + d\, n^2)$, which implies that HYP is FPT on average while it is W[1]-hard in the worst-case. For constant dimension d this gives an algorithm for HYP with runtime $\mathcal{O}(n^2)$ on average.

This is the first result proving that HYP is asymptotically easier in the average case. It gives a theoretical explanation why most HYP algorithms perform much better on average than their theoretical worst-case runtime predicts.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]:
Analysis of Algorithms and Problem Complexity

Keywords

Multiobjective Optimization, Theory,
Performance Measures, Selection

1. INTRODUCTION

Many real-world optimization problems consider multi-objective fitness functions, where several objectives are to be maximized (or minimized) at the same time [17]. The aim is to find a set of good trade-off solutions constituting the Pareto front. The quality of such a set of solutions is measured by an indicator function that maps a set of solutions to a real value. The currently most popular indicator is the hypervolume (HYP), which measures the volume of the union of the regions in the objective space that are dominated by the set of solutions. All hypervolume-based algorithms like the multi-objective covariance matrix adaptation evolution strategy (MO-CMA-ES) [23], the SMS-EMOA [5], and the hypervolume-based variant of the indicator-based evolutionary algorithm (IBEA) [42] aim at maximizing the hypervolume. This aim is known to help reaching a good approximation of the Pareto front. More precisely, it is known that the worst-case approximation factor obtained by any hypervolume-maximal set of fixed size μ is asymptotically equivalent to the best worst-case approximation factor achievable by any set of size μ [14]. Moreover, the hypervolume is the only known indicator that, given two sets A and B, values A higher than B if A dominates B [43].

It is known that the problem of computing the hypervolume is #P-hard [11], which implies that the hypervolume cannot be computed exactly in time polynomial in the number of objectives d unless P $=$ NP. Calculating an approximation is much easier: A simple Monte-Carlo algorithm yields an FPRAS (fully polynomial-time randomized approximation scheme) which gives an ε-approximation of the hypervolume of n points in time $\mathcal{O}(dn/\varepsilon^2)$ [11]. There are also several approximation algorithms for determining the least hypervolume contributor [1, 13, 25]. In contrast to this, we want to study the complexity of *exact* hypervolume computation in more detail.

Worst-case upper bounds. There are many algorithms for computing HYP with proven upper bounds on their worst-case runtime. The oldest ones stem from *Klee's Measure Problem* (KMP). This classical question of computational geometry asks for the measure of the union of n axis-parallel boxes in the d-dimensional space $\mathbb{R}^d$ [26]. HYP is a special case of KMP where all boxes share a common corner. Around 1980 it was shown that KMP can be solved in time $\mathcal{O}(n^{d-1})$ [36], which was ten years later improved by Overmars and Yap [33] to $\mathcal{O}(n^{d/2}\log n)$. It was claimed in [4] that the algorithm of Overmars and Yap can be improved to $\mathcal{O}(n \log n + n^{d/2})$ for HYP, but this was later withdrawn in [3]. The only asymptotic improvement of Overmars and Yap's algorithm is the algorithm of Chan [15], with runtime $n^{d/2}\, 2^{\mathcal{O}(\log^* n)}$, where $\log^*$ denotes the iterated logarithm. While determining a solution with minimal hypervolume contribution seems to need $\Theta(n)$ hypervolume computations, this is can be done in time $\mathcal{O}(n^{d/2}\log n)$ [12].

It was recently shown that any algorithm for KMP on (unit) cubes yields an algorithm for HYP with the same asymptotic runtime [10]. This implies that the algorithm of Bringmann [9] for KMP on cubes also gives an algorithm with runtime $\mathcal{O}(n^{(d+2)/3})$ for HYP. Another recent improvement for HYP is the algorithm of Yıldız and Suri [40] with runtime $\mathcal{O}(n^{(d-1)/2}\log n)$. There are also several algorithms for specific dimensions. However, compared to the two last mentioned algorithms of Bringmann [9] and Yıldız and Suri [40], only the algorithm of Fonseca et al. [20] gives an improvement for $d = 3$ with a runtime of $\mathcal{O}(n \log n)$. Overall, the currently best known asymptotic runtimes for HYP are summarized in the following Table 1.

Dimension	Runtime	Reference
$d = 1$	$\Theta(n \log n)$	[26]
$d = 2$	$\Theta(n \log n)$	[2, 33]
$d = 3$	$\Theta(n \log n)$	[20]
$d = 4$	$\mathcal{O}(n^{1.5} \log n)$	[40]
$d = 5$	$\mathcal{O}(n^{2} \log n)$	[40]
$d = 6$	$\mathcal{O}(n^{2.5} \log n)$	[40]
$d \geqslant 7$	$\mathcal{O}(n^{(d+2)/3})$	[9]

Table 1: Best known asymptotic worst-case runtimes for calculating the hypervolume indicator (HYP) depending on the number of points n and the number of dimensions d.

Computational hardness. The only known unconditional lower bound for the computational complexity of HYP for any d is $\Omega(n \log n)$ [6]. It is also known that HYP is #P-hard [11], which implies that it cannot be solved exactly in time $\mathcal{O}(\text{poly}(n, d))$ unless P $=$ NP. This does not rule out that there could be an algorithm with runtime say $\mathcal{O}(d^d n)$. For studying the dependence between n and d, we use parameterized complexity theory [19, 32], which is a generalization of classical one-dimensional complexity theory. This has been shown beneficial for the analysis of evolutionary algorithms in [28, 35]. We use the dimension d as a parameter. Then a problem is called fixed-parameter-tractable (FPT) if there is an algorithm with runtime $\mathcal{O}(f(d)\ \text{poly}(n))$ for an arbitrary function f. The parameterized analog of NP is the class W[1]. The textbook example for a W[1]-complete problem is the clique problem, which asks for the existence of a clique of size d in an n-vertex graph.

It is known that KMP is W[1]-hard [15] by a reduction from the clique problem to KMP. As there is a parameterized reduction from KMP to HYP [10], also HYP is W[1]-hard. This implies that there is no algorithm for HYP with runtime $f(d)\,\text{poly}(n)$ for any function f unless FPT = W[1]. In order to obtain better lower bounds, we need stronger complexity assumptions. The exponential time hypothesis (ETH) [24, 29] states that 3-SAT cannot be solved in subexponential time in the worst case, which implies P $\neq$ NP. ETH is generally believed and commonly used to prove stronger lower bounds. Assuming ETH, there is no $f(d)\, n^{o(d)}$ time algorithm for the clique problem [16]. Therefore, under the same assumption there is no algorithm for HYP with runtime $n^{o(d)}$.

For even more precise lower bounds, we have to look more closely at the reductions of Chan [15] and Bringmann [10]. Chan [15] showed that if KMP can be solved in time $T(n, d)$, then we can decide the clique problem in time $\mathcal{O}(T(\mathcal{O}(n^2), d))$. Bringmann [10] showed that if HYP can be solved in $T(n, d)$, then KMP can be solved in time $\mathcal{O}(T(n, 2d))$. The best time bound for the clique problem is $\mathcal{O}(n^{\omega\lceil d/3\rceil + (d \bmod 3)})$ [31] using fast matrix multiplication, where $\omega < 2.3727$ [39]. For large d this gives a runtime of $\mathcal{O}(n^{0.7909\, d})$ for clique. Above reduction implies that an algorithm with runtime $\mathcal{O}(n^{0.19\, d})$ for HYP would also improve upon the very classical clique problem. On the other hand, without algebraic techniques like Strassen or Williams [39], the best time bound for a purely combinatorial algorithm for the clique problem is $\mathcal{O}(n^d)$ (ignoring logarithmic factors). Hence, without algebraic techniques, solving HYP in time $o(n^{\lfloor d/2\rfloor/2})$ would also imply a significant complexity-theoretic breakthrough and is therefore unlikely. The following Table 2 gives a summary of the discussed lower bounds.

Lower Bound	Assumption
$\Omega(n \log n)$	none
$\omega(\text{poly}(n, d))$	P $\neq$ NP
$\omega(f(d)\,\text{poly}(n))\ \forall f$	FPT $\neq$ W[1]
$n^{\Omega(d)}$	exponential time hypothesis (ETH)
$\Omega(n^{0.19d})$	no improvement on clique problem
$\Omega(n^{0.24d})$	no combinatorial improvement on clique problem

Table 2: Lower bounds for the runtime of calculating the hypervolume indicator (HYP) depending on the number of points n and the number of dimensions d, ordered by the strength of the necessary complexity-theoretic assumption.

Heuristics. Additionally to the aforementioned algorithms with theoretically best known worst-case runtimes, there are various heuristic improvements with and without worst-case guarantees. A popular algorithm is Hypervolume by slicing objectives (HSO) which was suggested independently by Zitzler [41] and Knowles [27]. Its worst-case complexity is $\mathcal{O}(n^{d-1})$. Another experimentally very competitive algorithm is the WFG algorithm of While et al. [37] with worst-case complexity of $\mathcal{O}(2^n)$. While the algorithm of Overmars and Yap [33] has a much better worst-case complexity, it performs slower in experiments than many heuristics [37], especially on random inputs. We want to explain this difference by studying the average-case complexity of HYP.

New Results: Average-case. Most experimental comparisons of hypervolume algorithms use (at least as part of their dataset) randomly-generated fronts (e.g. [7, 8, 13, 20, 37, 38]). As these empirical observations do not always match with the theoretical worst-case behavior, we want to study rigorously the average-case complexity of the hypervolume indicator. As the average-case behavior depends on the probability distribution of the given point set, we first have to model the random input.

ASSUMPTION 1. *Our average-case model assumes that all n points are chosen independently and uniformly (i.i.d.) at random on a d-dimensional simplex Δ_d.*

This assumption is commonly used in experimental comparisons of HYP algorithms (see references above). Moreover, it is also motivated by experimental comparisons of multi-objective evolutionary algorithms (MOEAs) based on the hypervolume like the multi-objective covariance matrix adaptation evolution strategy (MO-CMA-ES) [23], the SMS-EMOA [5], and variants of the indicator-based evolutionary algorithm (IBEA) [42]. The reason why studying the computational complexity of computing the hypervolume of points on the d-dimensional simplex is interesting not only for HYP as an indicator, but also for HYP-based MOEAs, is that the commonly used benchmark function DTLZ1 [18] constitutes of many layers of d-dimensional simplices.

Recall from Table 2 that HYP is W[1]-hard, which implies that there is no algorithm with FPT runtime in the worst case (unless FPT = W[1]). Even stronger, assuming the exponential time hypothesis, there is no algorithm with runtime $n^{o(d)}$ in the worst case.

In stark contrast to this, we show that HYP is FPT on average. Given any algorithm with a (reasonable) worst-case runtime, we can translate it to an algorithm with similar worst-case runtime and FPT average-case runtime with the following theorem.

THEOREM 1. *Let A be an algorithm which computes HYP in worst-case time $\mathcal{O}(n^{f(d)})$ for an arbitrary function f. Then we can construct an algorithm B which computes HYP in worst-case time $\mathcal{O}(n^{f(d)+1})$ and average-case time $\mathcal{O}(g(d)\, n + d\, n^2))$ with*

$$g(d) := \lceil f(d) \rceil^{2f(d)+1} d^{(d-1)f(d)+1}.$$

The rest of the papers presents the proof of this theorem. Using Theorem 1, we can now choose any algorithm from Table 1 and get an algorithm with FPT average-case runtime. Plugging in the algorithm of Yıldız and Suri [40] with $f(d) = (d-1)/2$ yields the following simple corollary that shows that on average HYP can be computed exactly in FPT runtime.

THEOREM 2. *HYP can be solved on average in time $\mathcal{O}(d^{d^2/2}\, n + d\, n^2)$ for $d \geqslant 3$.*

Note that for any constant dimension d, Theorem 2 gives an algorithm with average-case runtime $\mathcal{O}(n^2)$ for HYP. Theorem 2 also shows that while HYP is W[1]-hard in the worst case, its computational complexity drops to FPT in the average case. This is the first result proving that HYP is asymptotically easier in the average case. It gives a theoretical explanation why many HYP algorithms (especially WFG [37]) perform much better on average than their theoretical worst-case runtime predicts.

We want to point out that the study of parameterized average-case complexity has been initiated by Müller [30]. What comes closest to our result is that the worst-case W[1]-hard problem clique has proven to be FPT on average for Erdős-Rényi random graphs [21] and inhomogeneous random graphs [22].

2. PRELIMINARIES

For a point $x \in \mathbb{R}^d$ we write x_i for its i-th component. For $x, y \in \mathbb{R}^d$ we say that x is dominated by y, or $x \preceq y$, if $x_i \leqslant y_i$ for all dimensions i. We write $x < y$ if $x_i < y_i$ for all i.

For a finite point set $P \subset \mathbb{R}^d$ and a reference point $r \in \mathbb{R}^d$ the hypervolume $\mathrm{HYP}(P)$ is defined as

$$\mathrm{HYP}(P) = \mathrm{VOL}(\{y \in \mathbb{R}^d \mid \exists x \in P \colon r \preceq y \preceq x\}),$$

where VOL is the usual Lebesgue measure. In this paper we set $r = (0, \dots, 0)$ for convenience (as translations do not change the hypervolume) and assume that $P \subset \mathbb{R}^d_{\geqslant 0}$. Moreover, we define the contribution of a point $x \in \mathbb{R}^d_{\geqslant 0}$ to a point set $P \subset \mathbb{R}^d_{\geqslant 0}$ (with $x \notin P$) as

$$\mathrm{CON}_P(x) := \mathrm{HYP}(P \cup \{x\}) - \mathrm{HYP}(P).$$

For $x \in \mathbb{R}^d_{\geqslant 0}$ we define the spanned box B_x as

$$B_x := [0, x_1] \times \ldots \times [0, x_d].$$

Note that x and B_x are equivalent objects, and many of the above notions can easily be expressed in terms of B_x: We have $x \preceq y$ iff $B_x \subseteq B_y$, and we have $\mathrm{HYP}(P) = \mathrm{VOL}(\bigcup_{x \in P} B_x)$. Furthermore, for $x, y \in \mathbb{R}^d_{\geqslant 0}$ we define $x \cap y$ as the point in $\mathbb{R}^d_{\geqslant 0}$ with i-th coordinate

$$(x \cap y)_i = \min\{x_i, y_i\}.$$

Then $B_{x \cap y} = B_x \cap B_y$, which explains the notation "$x \cap y$".

In this paper we consider hypervolume computation in the average case, when the point set P consists of n uniformly random points from the standard simplex

$$\Delta = \Delta_d := \Big\{(x_1, \dots, x_d) \in \mathbb{R}^d_{\geqslant 0} \mid \sum_i x_i = 1\Big\}.$$

3. THE ALGORITHM

In this section we describe algorithm B which is constructed in Theorem 1. The general layout of algorithm B is as follows. Given a point set P, for each point $x \in P$ we compute its contribution $\mathrm{CON}_{P \setminus \{x\}}(x)$ and then delete x from P. Summing up all these contributions, we get the overall hypervolume of P. More precisely, if $P = \{x^{(1)}, \dots, x^{(n)}\}$, and letting $P_i := \{x^{(i+1)}, \dots, x^{(n)}\}$, we sum up $\mathrm{CON}_{P_i}(x^{(i)})$ for $i = 1, \dots, n$.

Pseudocode 1: Overview of Algorithm B

```
Input: $P = \{x^{(1)}, \dots, x^{(n)}\}$
1 Let $P_i := \{x^{(i+1)}, \dots, x^{(n)}\}$
2 $H \leftarrow 0$
3 for $i \leftarrow 1$ to $n$ do
4     $H \leftarrow H + \mathrm{CON}_{P_i}(x^{(i)})$.
5 Return $H$
```

It remains to give an algorithm that, given a point set P and a point $x \notin P$, computes the contribution $\mathrm{CON}_P(x)$. For this part of algorithm B a running example is depicted in Figure 1. Observe that the domain contributed by x to P is contained in the box $B_x = [0, x_1] \times \ldots \times [0, x_d]$. Recall that $x \cap y$ is defined as the point with i-th coordinate $\min\{x_i, y_i\}$. We do not change the contribution $\mathrm{CON}_P(x)$ by replacing any point $y \in P$ by $y \cap x$, since inside B_x the points y and $y \cap x$ dominate the same set of points. In other words, considering the point set

$$P \cap x := \{y \cap x \mid y \in P\},$$

we have

$$\mathrm{CON}_P(x) = \mathrm{CON}_{P \cap x}(x),$$

see Figure 1b.

We note that the set $P \cap x$ can, in general, contain dominated points, even if P does not. In fact, we will show that the expected number of non-dominated points in $P \cap x$ is bounded by $d^{\mathcal{O}(d)}$ for random points on the simplex, which can be much smaller than n.

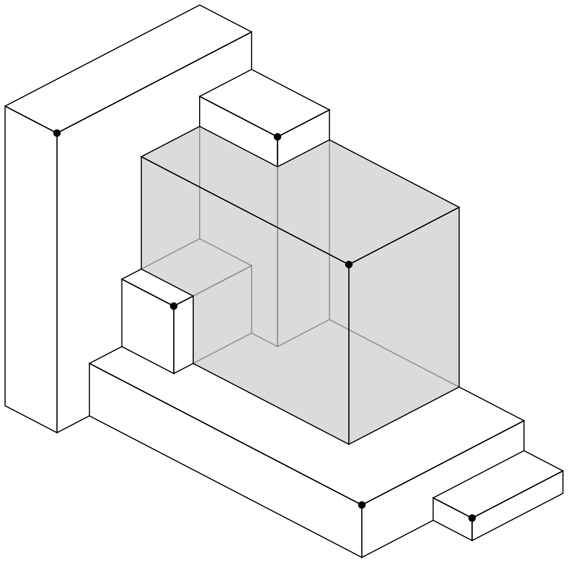
Without loss of generality we can assume that x is not dominated by any point in P; this is certainly true for points on the simplex and otherwise we can first delete all dominated points. Consider the points that are larger than x in all but one dimension,

$$Q_i = Q_i(x, P) := \{y \in P \mid y_j \geqslant x_j \text{ for all } j \neq i\},$$

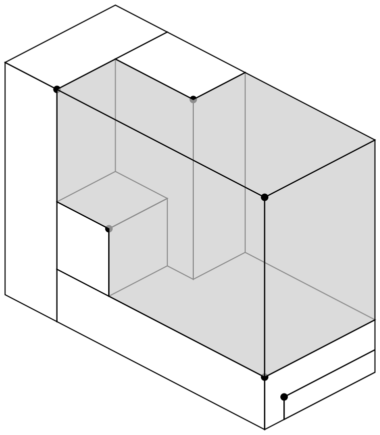
and define

$$z_i = z_i(x, P) := \max\{y_i \mid y \in Q_i\},$$

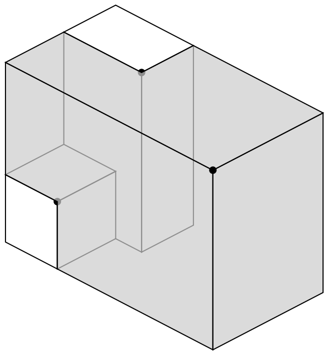
or $z_i = 0$, if this set is empty. This defines a bounding box $B(z, x) = [z_1, x_1] \times \ldots \times [z_d, x_d]$, which contains the domain that is contributed by x to P (since every point in the space $B_x \setminus B(z, x)$ is dominated by one of the points defining z). In particular, any point $y \in P$ with $y_i < z_i$ for some i is dominated in $P \cap x$, as $y \cap x$ is contained in $B_x \setminus B(z, x)$. This allows to simplify the problem as follows. First, we can translate every point by $-z$, so that z is translated to $(0, \dots, 0)$, while the contribution of x (translated to $x - z$) is still contained in $\mathbb{R}^d_{\geqslant 0}$. After that, we can delete



(a) Algorithm B computes the contribution of a point x (shaded box) to the remaining point set P (white boxes).



(b) This figure shows x (shaded box) and $P \cap x$ (white boxes). Note that the contribution of x to $P \cap x$ is the same as its contribution to P. Also note that $P \cap x$ has dominated points, while P did not.



(c) This shows the constructed subproblem $I_{x,P}$ (white boxes) and point x (shaded box). Only two boxes remain that influence the contribution of x.

Figure 1: For computing a contribution $\text{CON}_P(x)$, algorithm B first simplifies the instance in two steps as shown in this figure.

every point that now has a negative or zero coordinate, as such a point does not influence the hypervolume in $\mathbb{R}^d_{\geqslant 0}$. This yields the instance

$$I = I_{x,P} := \{(y \cap x) - z \mid y \in P, y > z(x,P)\},$$

where "$>$" means greater in *all* coordinates, see Figure 1c. By the above arguments, we have $\text{CON}_I(x - z) = \text{CON}_P(x)$. Note that $x - z$ dominates any point in I, so that we have

$$\begin{aligned}\text{CON}_I(x - z) &= \text{HYP}(I \cup \{x - z\}) - \text{HYP}(I) \\ &= \text{HYP}(\{x - z\}) - \text{HYP}(I).\end{aligned}$$

Since $\text{HYP}(\{x - z\}) = \prod_{i=1}^d (x_i - z_i)$ is trivial, we only have to solve the subinstance $\text{HYP}(I_{x,P})$, which we do using any algorithm A with worst-case runtime $\mathcal{O}(n^{f(d)})$. The overall algorithm for computing a contribution is shown in Pseudocode 2.

Pseudocode 2: Computing a contribution $\text{CON}_P(x)$

```
  Input: $x \in \mathbb{R}^d_{\geqslant 0}$, $P \subset \mathbb{R}^d_{\geqslant 0}$
1 Compute the lower coordinates of the bounding box:
2 $z_i = \max\{y_i \mid y \in P, y_j \geqslant x_j$ for all $j \neq i\}$ (or 0, if this set
  is empty).
3 Compute the subinstance
  $I_{x,P} = \{(y \cap x) - z \mid y \in P, y > z\}$
4 Compute $\text{HYP}(I_{x,P})$ using algorithm A
5 $\text{HYP}(\{x - z\}) = \prod_{i=1}^d (x_i - z_i)$
6 Return $\text{HYP}(\{x - z\}) - \text{HYP}(I_{x,P})$.
```

We remark that there are two differences between algorithm B and the WFG algorithm: First, the latter directly computes the non-dominated points in $P \cap x$, while algorithm B computes an approximation of this using the bounding box $B(z, x)$. Second, WFG solves each subproblem recursively, while algorithm B resorts to a worst-case algorithm A for the subproblems.

4. RUNTIME ANALYSIS

First note that the worst-case runtime of algorithm B is at most $\mathcal{O}(n^{f(d)+1})$, since it uses n calls to algorithm A that has worst-case runtime $\mathcal{O}(n^{f(d)})$.

For the average-case runtime, observe that one can check in time $\Theta(d)$ whether a point $y \in P$ belongs to any of the sets Q_i, and even to which one. Thus, the point z can be computed by a linear scan over all points in P, and the set $I_{x,P}$ can be computed by a second linear scan. Over all $i = 1, \ldots, n$, this sums up to a runtime of $\Theta(dn^2)$, explaining the second term of Theorem 1. The remaining runtime stems from algorithm A computing the n subinstances $I_i := I_{x^{(i)},P_i}$. Recall that the runtime of algorithm A is $\mathcal{O}(n^{f(d)})$. Then clearly the runtime of the overall algorithm is

$$\mathcal{O}(dn^2 + \sum_{i=1}^{n} |I_i|^{f(d)}). \tag{1}$$

Hence, we can bound the expected runtime of algorithm B by investigating $\text{Ex}[|I_{x,P}|^{f(d)}]$ for random points on a simplex, which we do in the following lemma.

LEMMA 3. *Let $k \in \mathbb{R}_{\geqslant 0}$, let $x, x^{(1)}, \ldots, x^{(n)}$ be uniformly i.i.d. points on the simplex Δ_d, and set $P := \{x^{(1)}, \ldots, x^{(n)}\}$. Then we have*

$$\text{Ex}[|I_{x,P}|^k] \leqslant 2\lceil k \rceil^{2k+1} d^{k(d-1)+1}.$$

Plugging this into equation (1) immediately yields Theorem 1. Note that for $f(d) = \mathcal{O}(d)$ the bound in the above lemma can be simplified to $d^{\mathcal{O}(d^2)}$. In the next section we prove Lemma 3.

5. PROOF OF LEMMA 3

In this section, let $x, x^{(1)}, \dots, x^{(n)}$ be uniformly i.i.d. points on the simplex $\Delta = \Delta_d$ and set $P = \{x^{(1)}, \dots, x^{(n)}\}$.

We split the proof of Lemma 3 as follows. First, we reduce the case of real numbers k to the case of integral k. After that, in Section 5.2 we prove Lemma 3 for k being an integer.

5.1 Real Exponents

Let $k \in \mathbb{R}_{\geqslant 0}$ and $K := \lceil k \rceil$. Assume that we know that Lemma 3 holds for K, i.e.,

$$\mathrm{Ex}[|I_{x,P}|^K] \leqslant 2K^{2K+1} d^{K(d-1)+1}. \tag{2}$$

By Jensen's inequality we know that for a positive random variable X and $0 \leqslant \alpha \leqslant 1$ we have

$$\mathrm{Ex}[X^\alpha] \leqslant \mathrm{Ex}[X]^\alpha.$$

Using this with $X = |I_{x,P}|^K$ and $\alpha = k/K$ yields

$$\begin{aligned}\mathrm{Ex}[|I_{x,P}|^k] &\leqslant \mathrm{Ex}[|I_{x,P}|^K]^{k/K} \\ &\overset{(2)}{\leqslant} (2K^{2K+1} d^{K(d-1)+1})^{k/K} \\ &\leqslant 2K^{2k+1} d^{k(d-1)+1},\end{aligned}$$

which proves Lemma 3 for k, as $K = \lceil k \rceil$. Hence, we reduced the proof of Lemma 3 to the case of $k \in \mathbb{N}$.

5.2 Integral Exponents

In the remainder of this section we can assume that $k \in \mathbb{N}$. Algorithm B first computes the lower end point $z = z(x, P)$ of the bounding box containing x's contribution. Note that instead of sampling $x, x^{(1)}, \dots, x^{(n)}$ in this order, we can generate (x, P) by first sampling x uniformly from the simplex, then sampling z with the right distribution conditioned on x, and then sample the remaining points $x^{(1)}, \dots, x^{(n)}$ conditioned on x and z. In order to analyze $|I_{x,P}|$ in this way, we first have to understand the distribution of z conditioned on x, and then the distribution of $|I_{x,P}|$ conditioned on x and z. Together, this yields an expression for $|I_{x,P}|$ that we can further analyze.

We first proof three lemmas that lay the foundations for the above approach.

LEMMA 4. *Let $x' \in \Delta$. Conditioned on $x = x'$, we have for any $z' \in B_x$*

$$\Pr[z \preceq z' \mid x = x'] = \Big(1 - \sum_{i=1}^{d} (x'_i - z'_i)^{d-1}\Big)^n.$$

Proof. Consider for $1 \leqslant i \leqslant d$ the set

$$S_i := \{y \in \Delta \mid y_i > z'_i \text{ and } y_j \geqslant x'_j \text{ for all } j \neq i\},$$

depicted as the dark shaded regions in Figure 2. Recall the definition of z_i as $\max\{y_i \mid y \in P, y_j \geqslant x_j \text{ for all } j \neq i\}$ or, if this set is empty, $z_i = 0$. The event $z \preceq z'$ means that for each dimension i we have $z_i \leqslant z'_i$, i.e., that there is no point $y \in P$ in S_i. Note in Figure 2 that these regions are smaller simplices inside Δ. Indeed, consider for $y \in \Delta$ the point $\hat{y}$ defined by $\hat{y}_i := y_i - z'_i$ and

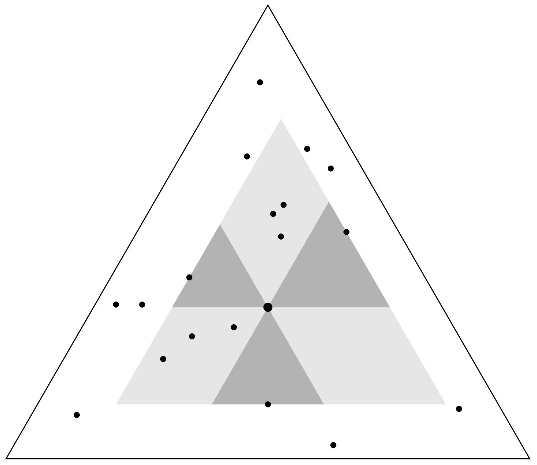


Figure 2: This figure shows the simplex Δ_3, a point x (the center point) and a point set P (all other points). Our algorithm to compute $\mathrm{CON}_P(x)$ first constructs the point z. The points defining z are the ones incident to the dark shaded regions. Note that the dark shaded regions are free of points (by definition of z). The points that end up in the subproblem $I_{x,P}$ are the points contained in the light shaded regions.

$\hat{y}_j := y_j - x'_j$ for $j \neq i$. Then y is contained in S_i iff $\hat{y}$ lies in the set

$$\Big\{(a_1 \dots, a_d) \in \mathbb{R}^d_{\geqslant 0} \;\Big|\; \sum_j a_j = 1 - z'_i - \sum_{j \neq i} x'_j, a_i \neq 0\Big\}.$$

Note that for a uniformly random point in Δ the probability of falling in the above set is the same as the probability of falling in the following simplex, as $y_i = z'_i$ (or $a_i = 0$) happens with probability 0:

$$(1 - z'_i - \sum_{j \neq i} x'_j)\Delta = (x'_i - z'_i)\Delta,$$

i.e., the simplex with sidelength $x'_i - z'_i$. Note that the relative volume of $(x'_i - z'_i)\Delta$ in Δ is $(x'_i - z'_i)^{d-1}$ (since Δ is a $(d-1)$-dimensional object). Thus, the probability of a uniformly random point on the simplex to fall into one of the sets S_i is

$$\sum_{i=1}^{d} (x'_i - z'_i)^{d-1},$$

and the probability of n points not to fall into one of the S_i is

$$\Big(1 - \sum_{i=1}^{d} (x'_i - z'_i)^{d-1}\Big)^n.$$

Since this is equal to the probability of the event $z \preceq z'$ (conditioned on $x = x'$), this finishes the proof. $\square$

Now that we know the distribution of z, let x and z be fixed. Then we can show that $|I_{x,P}|$ is stochastically dominated by a binomial random variable $\mathrm{Bin}(n, p)$ for some probability p, $|I_{x,P}| \preceq \mathrm{Bin}(n, p)$, as shown by the following lemma.

LEMMA 5. *Let $x' \in \Delta, z' \in B_{x'}$. Conditioned on $x = x'$ and $z = z'$, we have*

$$|I_{x,P}| \preceq \mathrm{Bin}(n, p),$$

where

$$p = \Big(\sum_{i=1}^{d} x_i' - z_i'\Big)^{d-1}.$$

In particular, for $k \in \mathbb{N}$ we have

$$\mathrm{Ex}[|I_{x,P}|^k \mid x = x', z = z'] \leqslant \mathrm{Ex}[\mathrm{Bin}(n,p)^k].$$

Proof. Conditioned on $x = x'$ and $z = z'$ we want to estimate the distribution of $|I_{x,P}|$, the number of points in P that strictly dominate z. Note that by specifying z we have already fixed up to d points, namely the ones that define the z_i. There are less than n points left that may be counted in $|I_{x,P}|$. By definition of z, none of these points may be contained in one of the sets

$$S_i := \{y \in \Delta \mid y_i > z_i' \text{ and } y_j \geqslant x_j' \text{ for all } j \neq i\},$$

$1 \leqslant i \leqslant d$, depicted as the dark shaded regions in Figure 2. Thus, they are uniformly i.i.d. over the set $\Delta \setminus \bigcup_{i=1}^d S_i$. Such a point belongs to $I_{x,P}$ only if it is contained in the set

$$T := \{y \in \Delta \mid y_i \geqslant z_i \text{ for all } 1 \leqslant i \leqslant d\}.$$

Note that T is a simplex of sidelength $1 - \sum_j z_j$ inside Δ. Indeed, a point y is in T iff $y - z$ is in the simplex

$$\Big\{(a_1,\dots,a_d) \in \mathbb{R}^d_{\geqslant 0} \;\Big|\; \sum_j a_j = 1 - \sum_j z_j\Big\}.$$

Since such a simplex is a $(d-1)$-dimensional object, the relative volume of T in Δ is $(1 - \sum_j z_j)^{d-1}$. Similarly, one can show that the relative volume of S_i in Δ is $(x_i - z_i)^{d-1}$. Thus, the probability of a uniform point in $\Delta \setminus \bigcup_{i=1}^d S_i$ to lie in $T \setminus \bigcup_{i=1}^d S_i$ is

$$\frac{(1 - \sum_i z_i)^{d-1} - \sum_i (x_i - z_i)^{d-1}}{1 - \sum_i (x_i - z_i)^{d-1}}.$$

Using $\frac{b-a}{c-a} \leqslant \frac{b}{c}$ for $a \leqslant b \leqslant c$, we can bound this probability by

$$\leqslant \Big(1 - \sum_i z_i\Big)^{d-1} = \Big(\sum_i x_i - z_i\Big)^{d-1} = p.$$

Since the points are i.i.d., we get that $|I_{x,P}|$ is stochastically dominated by $\mathrm{Bin}(n,p)$.

Since both random variables are positive, the second claim follows directly from the stochastic dominance. □

Furthermore, we have to bound the moments of binomial random variables.

LEMMA 6. *For any $n, k \in \mathbb{N}$ and $p \in [0,1]$ we have*

$$\mathrm{Ex}[\mathrm{Bin}(n,p)^k] \leqslant k^{k+1}(1 + (np)^k).$$

Proof. We can write $\mathrm{Bin}(n,p)$ as a sum of n i.i.d. Bernoulli random variables X_i with $\Pr[X_i = 1] = p$,

$$\mathrm{Bin}(n,p) = \sum_{i=1}^n X_i.$$

This yields

$$\mathrm{Ex}[\mathrm{Bin}(n,p)^k] = \mathrm{Ex}\Big[\Big(\sum_{i=1}^n X_i\Big)^k\Big] = \sum_{j_1,\dots,j_k \in [n]} \mathrm{Ex}[X_{j_1} \cdots X_{j_k}].$$

For a particular summand $\mathrm{Ex}[X_{j_1} \cdots X_{j_k}]$, let $i_1 < \ldots < i_\ell$ be the different values taken by $j_1,\dots,j_k$, i.e., $\{i_1,\dots,i_\ell\} = \{j_1,\dots,j_k\}$. Since the X_i are indicator random variables, and since they are i.i.d., we have

$$\mathrm{Ex}[X_{j_1} \cdots X_{j_k}] = \mathrm{Ex}[X_{i_1} \cdots X_{i_\ell}] = \mathrm{Ex}[X_1 \cdots X_\ell] = p^\ell.$$

Now, how many summands $\mathrm{Ex}[X_{j_1} \cdots X_{j_k}]$ reduce to p^ℓ as above? There are $\binom{n}{l} \leqslant n^\ell$ ways to choose $i_1,\dots,i_\ell$. We roughly bound the number of ways to distribute $i_1,\dots,i_\ell$ over $j_1,\dots,j_k$ by $\ell^k \leqslant k^k$. This yields

$$\begin{aligned}\mathrm{Ex}[\mathrm{Bin}(n,p)^k] &\leqslant \sum_{\ell=1}^k k^k (np)^\ell \\ &\leqslant k^{k+1}\max\{1,(np)^k\} \leqslant k^{k+1}(1+(np)^k). \; \square\end{aligned}$$

Recall that we want to analyze $\mathrm{Ex}[|I_{x,P}|^k]$ by first choosing x uniformly at random in the simplex, then sampling z according to the right distribution conditioned on x, and analyze $|I_{x,P}|$ conditioned on x and z. In other words, we want to compute

$$\begin{aligned}\mathrm{Ex}[|I_{x,P}|^k] = \int_\Delta \Pr[x = x'] \int_{B_{x'}} &\Pr[z = z' \mid x = x'] \\ &\cdot \mathrm{Ex}[|I_{x,P}|^k \mid x = x', z = z'] dz' dx'.\end{aligned}$$

Lemma 5 allows to bound

$$\mathrm{Ex}[|I_{x,P}|^k \mid x = x', z = z'] \leqslant \mathrm{Ex}[\mathrm{Bin}(n,p)^k \mid x = x', z = z'],$$

where

$$p := \Big(\sum_i x_i' - z_i'\Big)^{d-1}. \tag{3}$$

Applying Lemma 6, this can be simplified to

$$\mathrm{Ex}[|I_{x,P}|^k \mid x = x', z = z'] \leqslant k^{k+1}(1 + (np)^k).$$

Note that the summand $k^k \cdot 1$ on the above right hand side simply contributes k^{k+1} to the overall integral. The remainder is

$$k^{k+1} n^k \int_\Delta \Pr[x = x'] \int_{B_{x'}} \Pr[z = z' \mid x = x'] \cdot p^k dz' dx',$$

where p is dependent on x and z as given by equation (3). We bound

$$\begin{aligned}p^k &\leqslant d^{k(d-1)} \max_i \{(x_i' - z_i')^{d-1}\}^k \\ &\leqslant d^{k(d-1)} \sum_i (x_i' - z_i')^{k(d-1)},\end{aligned}$$

and get, conditioned on $x = x'$,

$$\begin{aligned}&\int_{B_{x'}} \Pr[z = z' \mid x = x'] p^k dz' \\ &\leqslant d^{k(d-1)} \sum_{i=1}^d \int_{B_{x'}} \Pr[z = z' \mid x = x'] (x_i' - z_i')^{k(d-1)} dz'.\end{aligned}$$

Since the inner part of this integral is independent of z_j, $j \neq i$, this simplifies to

$$d^{k(d-1)} \sum_{i=1}^d \int_0^{x_i'} \Pr[z_i = z_i' \mid x = x'] (x_i' - z_i')^{k(d-1)} dz_i'.$$

We can calculate this term, since we know the distribution of z (and, thus, z_i) by Lemma 4.

LEMMA 7. *Let $x' \in \Delta$. Conditioned on $x = x'$, we have*

$$\int_0^{x'_i} \Pr[z_i = z'_i \mid x = x'](x'_i - z'_i)^{k(d-1)} dz'_i \leqslant \frac{k^k}{n^k}.$$

This allows to simplify

$$\int_{B_{x'}} \Pr[z = z' \mid x = x'] p^k dz' \leqslant d^{k(d-1)} \cdot d\frac{k^k}{n^k}.$$

In total, this yields the bound

$$\begin{aligned}\mathrm{Ex}[|I_{x,P}|^k] &\leqslant k^{k+1}\left(1 + n^k \int_\Delta \Pr[x = x'] d^{k(d-1)} d\frac{k^k}{n^k} dx'\right)\\ &\leqslant 2k^{2k+1} d^{k(d-1)+1} = d^{\mathcal{O}(d^2)},\end{aligned}$$

finishing the proof of Lemma 3.

Proof of Lemma 7. Partial integration yields

$$\begin{aligned}&\int_0^{x'_i} \Pr[z_i = z'_i \mid x = x'](x'_i - z'_i)^{k(d-1)} dz'_i\\ &= \Big[\Pr[z_i \leqslant z'_i \mid x = x'](x'_i - z'_i)^{k(d-1)}\Big]_0^{x'_i}\\ &\quad + k(d-1)\int_0^{x'_i} \Pr[z_i \leqslant z'_i \mid x = x'](x'_i - z'_i)^{k(d-1)-1} dz'_i.\end{aligned}$$

Note that the first summand of this is at most 0, since plugging in x'_i for z'_i it evaluates to 0 and plugging in 0 it evaluates to a non-negative value. On the other hand, by Lemma 4 the second summand equals

$$k(d-1)\int_0^{x'_i} (1 - (x'_i - z'_i)^{d-1})^n (x'_i - z'_i)^{k(d-1)-1} dz'_i.$$

Substituting $u := (x'_i - z'_i)^{d-1}$ with $\frac{du}{dz'_i} = -(d-1)(x'_i - z'_i)^{d-2}$, this is equal to

$$\begin{aligned}&k\int_0^{x'^{d-1}_i} (1-u)^n u^{k-1} du\\ &\leqslant k\int_0^1 (1-u)^n u^{k-1} du.\end{aligned}$$

The right hand side is known, as the integrand is a Bernstein polynomial, and evaluates to

$$k\frac{n!(k-1)!}{(n+k)!} \leqslant \frac{k^k}{n^k},$$

finishing the proof. □

6. CONCLUSION

We have shown that HYP can be solved on average in time $\mathcal{O}(d^{d^2/2}\, n + d\, n^2)$ while $n^{\Omega(d)}$ is necessary in the worst-case (assuming the exponential time hypothesis). This proves an exponential gap between the average and worst-case complexity of HYP. As the algorithmic framework presented in Section 3 is very similar to the WFG algorithm [37] (cf. the discussion at the end of Section 3), this gives a theoretical explanation why its empirical runtime on random instances is much better than its worst-case bound.

Traditionally the best asymptotic runtime was obtained by the algorithm of Overmars and Yap [33]. However, this algorithm deviates significantly from the two step structure of an Algorithm B as described in Theorem 1 and Pseudocode 1. In fact, the algorithm of Overmars and Yap might not only have runtime $\Omega(n^{d/2}\log(n))$ in the worst-case, but also on average. This explains why various hypervolume heuristics were able to outperform the algorithm of Overmars and Yap on random instances. Though Theorem 2 presents an algorithm with average-case runtime $\mathcal{O}(n^2)$ for all constant d, the involved constants can get very large. We do not expect to get much tighter bounds for Algorithm B (cf. Pseudocode 1) to improve Theorem 1. However, other algorithmic approaches might yield smaller constants like $2^{\mathcal{O}(d)}$ or $\log^{\mathcal{O}(d)} n$, or possibly even a polynomial time average-case algorithm. Russo and Francisco [34] recently published a draft where they announce an algorithm with average-case runtime $\mathcal{O}(dn^{1.1}\log^d(n))$. Unfortunately, neither a description of the algorithm nor a proof is currently available. It would be interesting whether there are algorithms which avoid the expensive test for domination and reach an average case runtime of $\mathcal{O}(n)$ for any fixed d. We hope that such an algorithm can be found that is fast, both practically and theoretically in the average case.

References

[1] J. Bader, K. Deb, and E. Zitzler. Faster hypervolume-based search using Monte Carlo sampling. In *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems (MCDM '10)*, Vol. 636 of *Lecture Notes in Economics and Mathematical Systems*, pp. 313–326. Springer-Verlag, 2010.

[2] J. L. Bentley. Algorithms for Klee's rectangle problems, 1977. Department of Computer Science, Carnegie Mellon University, Unpublished notes.

[3] N. Beume. S-Metric calculation by considering dominated hypervolume as Klee's measure problem. *Evolutionary Computation*, 17:477–492, 2009.

[4] 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.

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

[6] N. Beume, C. M. Fonseca, M. López-Ibáñez, L. Paquete, and J. Vahrenhold. On the complexity of computing the hypervolume indicator. *IEEE Trans. Evolutionary Computation*, 13:1075–1082, 2009.

[7] L. Bradstreet, R. L. While, and L. Barone. A fast incremental hypervolume algorithm. *IEEE Trans. Evolutionary Computation*, 12:714–723, 2008.

[8] L. Bradstreet, L. While, and L. Barone. A fast many-objective hypervolume algorithm using iterated incremental calculations. In *Proc. Congress on Evolutionary Computation (CEC '10)*, pp. 1–8, 2010.

[9] K. Bringmann. An improved algorithm for Klee's measure problem on fat boxes. *Computational Geometry: Theory and Applications*, 45:225–233, 2012.

[10] K. Bringmann. Bringing order to special cases of Klee's measure problem, 2013. Draft available at arXiv:1301.7154.

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

[12] K. Bringmann and T. Friedrich. An efficient algorithm for computing hypervolume contributions. *Evolutionary Computation*, 18:383–402, 2010.

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

[14] K. Bringmann and T. Friedrich. Approximation quality of the hypervolume indicator. *Artificial Intelligence*, 195: 265–290, 2013.

[15] T. M. Chan. A (slightly) faster algorithm for Klee's measure problem. *Computational Geometry: Theory and Applications*, 43:243 – 250, 2010.

[16] J. Chen, X. Huang, I. A. Kanj, and G. Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72:1346 – 1367, 2006.

[17] R. Chiong, T. Weise, and Z. Michalewicz, editors. *Variants of Evolutionary Algorithms for Real-World Applications*. Springer, 2012.

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

[19] R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[20] C. M. Fonseca, L. Paquete, and M. López-Ibáñez. An improved dimension-sweep algorithm for the hypervolume indicator. In *Proc. IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 1157–1163, 2006.

[21] N. Fountoulakis, T. Friedrich, and D. Hermelin. On the average-case complexity of parameterized clique. Unpublished manuscript.

[22] T. Friedrich and A. Krohmer. Parameterized clique on scale-free networks. In *Proc. 23rd International Symposium on Algorithms and Computation (ISAAC '12)*, Vol. 7676 of *LNCS*, pp. 659–668. Springer, 2012.

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

[24] R. Impagliazzo and R. Paturi. The complexity of k-SAT. In *Proc. 14th IEEE Conference on Computational Complexity (CCC)*, pp. 237–240, 1999.

[25] H. Ishibuchi, N. Tsukamoto, Y. Sakane, and Y. Nojima. Indicator-based evolutionary algorithm with hypervolume approximation by achievement scalarizing functions. In *Proc. 12th Annual Conference on Genetic and Evolutionary Computation Conference (GECCO '10)*, pp. 527–534. ACM Press, 2010.

[26] 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.

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

[28] S. Kratsch and F. Neumann. Fixed-parameter evolutionary algorithms and the vertex cover problem. In *Proc. 11th Annual Conference on Genetic and Evolutionary Computation (GECCO '09)*, pp. 293–300. ACM Press, 2009.

[29] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

[30] M. Müller. *Parameterized Randomization*. PhD thesis, Albert-Ludwigs-Universität Freiburg im Breisgau, 2008.

[31] J. Nešetřil and S. Poljak. On the complexity of the subgraph problem. *Commen. Math. Univ. Carol.*, 26:415–419, 1985.

[32] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[33] M. H. Overmars and C.-K. Yap. New upper bounds in Klee's measure problem. *SIAM J. Comput.*, 20:1034–1045, 1991.

[34] L. M. S. Russo and A. P. Francisco. Quick hypervolume, 2012. Draft available at arXiv:1207.4598.

[35] A. M. Sutton and F. Neumann. A parameterized runtime analysis of evolutionary algorithms for the euclidean traveling salesperson problem. In *Proc. 26th AAAI Conference on Artificial Intelligence (AAAI)*, 2012.

[36] J. van Leeuwen and D. Wood. The measure problem for rectangular ranges in d-space. *J. Algorithms*, 2:282–300, 1981.

[37] L. While, L. Bradstreet, and L. Barone. A fast way of calculating exact hypervolumes. *IEEE Trans. Evolutionary Computation*, 16:86 –95, 2012.

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

[39] V. V. Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proc. 44th Symposium on Theory of Computing (STOC '12)*, pp. 887–898. ACM Press, 2012.

[40] H. Yıldız and S. Suri. On Klee's measure problem for grounded boxes. In *Proc. ACM Symposium on Computational Geometry (SoCG '12)*, pp. 111–120, 2012.

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

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

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