An efficient algorithm for computing hypervolume contributions*

Karl Bringmann
Universität des Saarlandes, Saarbrücken, Germany

Tobias Friedrich
Max-Planck-Institut für Informatik, Saarbrücken, Germany

Abstract

The hypervolume indicator serves as a sorting criterion in many recent multi-objective evolutionary algorithms (MOEAs). Typical algorithms remove the solution with the smallest loss with respect to the dominated hypervolume from the population. We present a new algorithm which determines for a population of size n with d objectives, a solution with minimal hypervolume contribution in time $\mathcal{O}(n^{d/2} \log n)$ for $d > 2$. This improves all previously published algorithms by a factor of n for all $d > 3$ and by a factor of $\sqrt{n}$ for $d = 3$.

We also analyze hypervolume indicator based optimization algorithms which remove $\lambda > 1$ solutions from a population of size $n = \mu + \lambda$. We show that there are populations such that the hypervolume contribution of iteratively chosen λ solutions is much larger than the hypervolume contribution of an optimal set of λ solutions. Selecting the optimal set of λ solutions implies calculating $\binom{n}{\mu}$ conventional hypervolume contributions, which is considered to be computationally too expensive. We present the first hypervolume algorithm which calculates directly the contribution of every set of λ solutions. This gives an additive term of $\binom{n}{\mu}$ in the runtime of the calculation instead of a multiplicative factor of $\binom{n}{\mu}$. More precisely, for a population of size n with d objectives, our algorithm can calculate a set of λ solutions with minimal hypervolume contribution in time $\mathcal{O}(n^{d/2} \log n + n^{\lambda})$ for $d > 2$. This improves all previously published algorithms by a factor of $n^{\min\{\lambda, d/2\}}$ for $d > 3$ and by a factor of n for $d = 3$.

1 Introduction

How to compare Pareto sets lies at the heart of research in multi-objective optimization. One measure that has been the subject of much recent study in evolutionary multi-objective optimization is the "hypervolume indicator" (HYP). It measures the volume of the dominated portion of the objective space. The hypervolume metric is of exceptional interest as it possesses the highly desirable feature of strict Pareto compliance [27]. That is, considering two Pareto sets A and B, the hypervolume indicator values A higher than B if the Pareto set A dominates the Pareto set B. This property makes it well suited for many-objective problems.

*A conference version of this article appeared under the title "Don't be greedy when calculating hypervolume contributions" [10] in the *Proceedings of the 10th ACM Foundations of Genetic Algorithms (FOGA 2009)*.

The hypervolume was first introduced for performance assessment in multi-objective optimization by Zitzler and Thiele [26]. Later on it was used to guide the search in various hypervolume-based evolutionary optimizers [6, 14, 15, 17, 25, 28]. Since then, several algorithms for calculating the hypervolume have been developed. The first one was the Hypervolume by Slicing Objectives (HSO) algorithm which was suggested independently by Zitzler [24] and Knowles [16]. To improve its runtime on practical instances, various speed up heuristics of HSO have been suggested [21, 23]. The currently best asymptotic runtime of $\mathcal{O}(n^{d/2} \log n)$ for $d \geqslant 3$ by Beume and Rudolph [4, 5] is based on Overmars and Yap's algorithm for Klee's Measure Problem [19]. There are also various algorithms for small dimensions [14, 18].

So far, the only known lower bound for any d is $\Omega(n \log n)$ [7]. Note that the worst-case combinatorial complexity (i.e., the number of faces of all dimensions on the boundary of the union of the boxes) of $\Theta(n^d)$ does not imply any lower bounds on the computational complexity. It has recently been shown by the authors [9] that the calculation of HYP is **#P**-hard, which implies that all hypervolume algorithms must have a super-polynomial runtime in the number of objectives or boxes unless $\mathbf{P} = \mathbf{NP}$. The paper [9] also presents an FPRAS (fully polynomial-time randomized approximation scheme) which gives an ε-approximation of the hypervolume indicator with probability $1 - \delta$ in time $\mathcal{O}(\log(1/\delta)\, nd/\varepsilon^2)$. Though this algorithm gives a very fast approximation in time (linear in n and d) for the hypervolume, it is important to note that this is not an approximation of the *contributing hypervolume*. Even the approximation of the latter is **NP**-hard as has recently been shown in [11].

Let us now look at the optimization problem. Given an arbitrary decision space $\mathcal{X}$, we want to maximize a function $f\colon \mathcal{X} \to \mathbb{R}^d_{\geqslant 0}$. A solution x is an element of the decision space $\mathcal{X}$, but we will typically identify it with the corresponding $f(x)$ in the objective space. A hypervolume-based algorithm maintains a population (set of solutions) $M \subseteq \mathbb{R}^d_{\geqslant 0}$ of size μ. Then the hypervolume is defined as

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

with $\mathrm{VOL}(\cdot)$ being the usual Lebesgue measure. Without loss of generality we assume the reference point to be 0^d. To avoid an unbounded population, the number of solutions in the population is usually fixed to a certain threshold and with every new Pareto optimal solution another one has to be removed. Most hypervolume based algorithms like SIBEA [28] or the generational MO-CMA-ES [15]* remove the solution $x \in M$ with the smallest contribution [6, 12, 28]

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

Throughout the paper, we will use $\mathrm{CON}(x) := \mathrm{CON}_M(x)$, if there is no ambiguity in the choice of M. The contribution can be seen as the measure of the space that is dominated by x, but no other point in M. A point $(x_1, \dots, x_d) \in \mathbb{R}^d$ dominates a point $(y_1, \dots, y_d) \in \mathbb{R}^d$, iff $x_i \geq y_i$ for all $i = 1, \dots, d$ and there is a $j \in \{1, \dots, d\}$ with $x_j > y_j$. The solution with minimal contribution can be calculated easily by μ hypervolume calculations.

*There are also steady-state variants of MO-CMA-ES [15, 20].

The problem is that often λ solutions should be removed at once. In this case one aims for a set $S \subseteq M$ of size λ, i.e., S is a *λ-set* or *λ-subset* of M, such that

$$\mathrm{CON}_M(S) := \mathrm{HYP}(M) - \mathrm{HYP}(M \setminus S)$$

is minimized. To calculate the optimal λ-set $S^\lambda_{\mathrm{opt}}(M)$ which has the smallest joint contribution with

$$S^\lambda_{\mathrm{opt}}(M) := \operatorname*{argmin}_{\substack{S \subseteq M \\ |S|=\lambda}} \mathrm{CON}_M(S) \tag{1}$$

requires $\binom{\mu+\lambda}{\mu}$ calculations of HYP$(\cdot)$ for $d > 2$[†]. In most settings, λ is much smaller than $n = \mu + \lambda$, that is, $\lambda \ll n$, and hence order n^λ calculations of HYP$(\cdot)$ are usually needed to obtain $S^\lambda_{\mathrm{opt}}(M)$. This is generally considered to be computationally too expensive [2, 6, 15, 28]. This is why all current hypervolume based optimization algorithms just calculate a greedy λ-set $S^\lambda_{\mathrm{greedy}}(M)$ by starting with $S^0_{\mathrm{greedy}}(M) := \emptyset$ and then iteratively setting

$$S^{\lambda+1}_{\mathrm{greedy}}(M) := S^\lambda_{\mathrm{greedy}}(M) \cup \left\{ \operatorname*{argmin}_{x \in M \setminus S^\lambda_{\mathrm{greedy}}(M)} \mathrm{CON}_{M \setminus S^\lambda_{\mathrm{greedy}}(M)}(x) \right\}. \tag{2}$$

This is computationally much cheaper as the number of calculations of HYP$(\cdot)$ is bounded by a small polynomial in n and λ and not exponential in λ, as for $S^\lambda_{\mathrm{opt}}(M)$. Throughout the paper, we will use $S^\lambda_{\mathrm{opt}} := S^\lambda_{\mathrm{opt}}(M)$ and $S^\lambda_{\mathrm{greedy}} := S^\lambda_{\mathrm{greedy}}(M)$ if there is no ambiguity in the choice of M.

In Section 2 we will show that $\mathrm{CON}(S^\lambda_{\mathrm{greedy}})/\mathrm{CON}(S^\lambda_{\mathrm{opt}})$ can theoretically be arbitrarily large. We also report on Pareto fronts with significant differences between both λ-sets in the DTLZ library [13].

This observation motivates the algorithm introduced in Section 3. It is an adaption of Overmars and Yap's algorithm which allows the direct computation of the contribution of every set of λ solutions. This avoids calculating $\binom{\mu+\lambda}{\mu}$ conventional hypervolume calculations. The basic idea of the new algorithm is it to maintain the volume of the contribution of every set of λ solutions during the calculation and to find the smallest of them afterwards. This second step causes overall an additive term of $\binom{\mu+\lambda}{\mu}$ in the runtime of the calculation instead of a multiplicative factor of $\binom{\mu+\lambda}{\mu}$ when using equation (1) directly. For a population of size $n = \mu + \lambda$ and $d > 2$, the algorithm calculates S^λ_{opt} in time $\mathcal{O}(n^{d/2} \log n + n^\lambda)$, which improves all previously published algorithms by a factor of $n^{\min\{\lambda, d/2\}}$.

This has two consequences. First, even if we remove only $\lambda = 1$ solutions in every step, this is a speed up by a factor of order n. This means we can determine the box with least contribution, i.e., the greedy solution, in time of order $n^{d/2} \log n$ instead of the usual $\mathcal{O}(n^{d/2+1} \log n)$ of Beume and Rudolph [4, 5]. Second, for $\lambda \leqslant d/2$ the asymptotic runtime is independent of λ. Therefore, using $\lambda = d/2$ instead of the commonly used $\lambda = 1$ (greedy) gives the same asymptotic runtime and yields the same or potentially even smaller contributions of the calculated λ-sets.

[†]For $d = 2$ this can be solved in time $\mathcal{O}(n^2)$ by dynamic programming [1, Alg. 1].

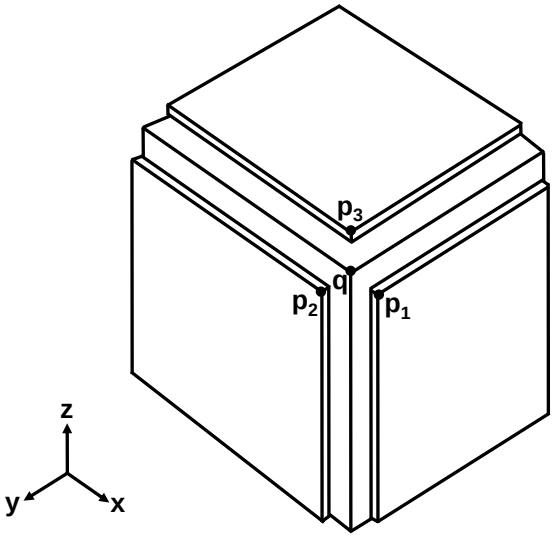

Figure 1: A three-dimensional example of a set $M \subseteq \mathbb{R}^3_{\geqslant 0}$ such that the greedy λ-set $S^\lambda_{\text{greedy}}(M)$ gives a much higher contribution than the optimal λ-set $S^\lambda_{\text{opt}}(M)$.

Note that we will assume throughout the paper, that d and λ are constant, i.e., n is the only growing parameter for the asymptotic analysis. This is no real restriction as both of them appear in the exponent of the resulting runtime and hence non-constant values for them would immediately make the algorithm inefficient.

2 The contributing hypervolume of greedy selection can be arbitrarily larger than the optimal one

For $\lambda = 1$, the greedy λ-set is optimal, i.e., $S^\lambda_{\text{greedy}} = S^\lambda_{\text{opt}}$. However, it is known that for $\lambda \geqslant 2$ the greedy algorithm is not able to construct the optimal solution set in general. For example Bradstreet et al. [8] present a three-dimensional example of $n = 6$ points where for $\lambda = 2$ the contribution of the greedy λ-set $S^\lambda_{\text{greedy}}$ is 12.5% larger than the optimal λ-set S^λ_{opt}. In this section we show that the λ-set $S^\lambda_{\text{greedy}}$ found by the greedy algorithm can have an arbitrarily larger contribution than the optimal λ-set S^λ_{opt} for all $\lambda \geqslant 2$. Let κ denote the ratio between the contribution of $S^\lambda_{\text{greedy}}$ and S^λ_{opt}. For many sets M, κ is either one or very close to one. Next, we prove that for given κ, dimension $d \geqslant 3$ and number of boxes $n > d$ there is a set of solutions M of size n such that the ratio between $\text{CON}_M(S^\lambda_{\text{greedy}})$ and $\text{CON}_M(S^\lambda_{\text{opt}})$ is larger than κ for all $2 \leq \lambda \leq d$. Additionally, we show that $\kappa > 1$ also holds for some fronts from the DTLZ library [13].

Lemma 1. *For all $\kappa \geqslant 1$, $d \geqslant 3$ and $n > d$ there is a set $M \subseteq \mathbb{R}^d_{\geqslant 0}$ with $|M| = n$ such that $\text{CON}_M(S^\lambda_{\text{greedy}})/\text{CON}_M(S^\lambda_{\text{opt}}) > \kappa$ for all $2 \leqslant \lambda \leqslant d$.*

Proof. We first assume $n = d + 1$. Let $\varepsilon := 1/(2\kappa d^2)$ and $\sigma := d^2\varepsilon^2$. We choose $M =$

$\{q, p_1, p_2, \ldots, p_d\}$ with

$$\begin{aligned} q &= (1+\varepsilon, 1+\varepsilon, \ldots, 1+\varepsilon), \\ p_1 &= (1+\varepsilon+\sigma, 1, 1, \ldots, 1), \\ p_2 &= (1, 1+\varepsilon+\sigma, 1, \ldots, 1), \\ &\cdots \\ p_d &= (1, 1, \ldots, 1, 1+\varepsilon+\sigma). \end{aligned}$$

A three-dimensional example is shown in Figure 1. As the space dominated by p_i but no other point in M is exactly

$$[0,1]^{i-1} \times [1+\varepsilon, 1+\varepsilon+\sigma] \times [0,1]^{d-i},$$

we have $\mathrm{CON}_M(p_i) = \sigma$. The point q dominates $[0, 1+\varepsilon]^d$, but $[0,1]^d$ and the d rectangular regions of the form $[0,1] \times \cdots \times [0,1] \times [1, 1+\varepsilon] \times [0,1] \times \cdots \times [0,1]$ are dominated by the other points in M, too, so that we have

$$\mathrm{CON}_M(q) = (1+\varepsilon)^d - 1^d - d \cdot \varepsilon = \sum_{i=2}^{d} \binom{d}{i} \varepsilon^i.$$

Similarly, we get

$$\begin{aligned} \mathrm{CON}_M(\{\underbrace{p_{i_1}, p_{i_2}, \ldots, p_{i_\lambda}}_{\lambda \text{ different } p_i\text{'s}}\}) &= \lambda\sigma, \\ \mathrm{CON}_M(\{q, \underbrace{p_{i_1}, \ldots, p_{i_{(\lambda-1)}}}_{\lambda-1 \text{ different } p_i\text{'s}}\}) &= \mathrm{CON}_M(q) + (\lambda-1)(\varepsilon+\sigma), \end{aligned}$$

where we used that after picking q every p_i dominates a portion of space with volume $\varepsilon + \sigma$. Since

$$\begin{aligned} \mathrm{CON}_M(q) &< \sum_{i=0}^{\infty} \binom{d}{i+2} \varepsilon^{i+2} < \sum_{i=0}^{\infty} \frac{d^{i+2}\varepsilon^{i+2}}{(i+2)!} = \varepsilon^2 d^2 \sum_{i=0}^{\infty} \frac{d^i \varepsilon^i}{(i+2)!} < \varepsilon^2 d^2 \sum_{i=0}^{\infty} \frac{1}{(i+2)!} \\ &= \varepsilon^2 d^2 (e-2) < \varepsilon^2 d^2 = \mathrm{CON}_M(p_i), \end{aligned}$$

the greedy algorithm chooses the λ-set

$$S^{\lambda}_{\text{greedy}} = \{q, \underbrace{p_{i_1}, p_{i_2}, \ldots, p_{i_{(\lambda-1)}}}_{(\lambda-1) \text{ different } p_i\text{'s}}\}$$

though the optimal λ-set is

$$S^{\lambda}_{\text{opt}} = \{\underbrace{p_{i_1}, p_{i_2}, \ldots, p_{i_\lambda}}_{\lambda \text{ different } p_i\text{'s}}\}.$$

Therefore for all $\lambda \leqslant d$,

$$\frac{\mathrm{CON}_M(S^{\lambda}_{\text{greedy}})}{\mathrm{CON}_M(S^{\lambda}_{\text{opt}})} = \frac{\mathrm{CON}_M(q) + (\lambda-1)(\varepsilon+\sigma)}{\lambda\sigma} > \frac{(\lambda-1)\varepsilon}{\lambda\sigma} \geqslant \frac{1}{2\,d^2\varepsilon} = \kappa.$$

This shows the claim for $n = d+1$. To prove the remaining case $n > d+1$, we take the set M from above, shift all points in M by 1 along the first dimension and add some extra boxes, each one contributing too much to be chosen by the greedy or the optimal algorithm. For this we define

$$\begin{aligned} M' =& \{(x_1+1, x_2, \dots, x_d) \mid (x_1, \dots, x_d) \in M\}, \\ B =& \textstyle\bigcup_{1 \le i < n-d} \{(1, C \cdot i, C \cdot (n-d-i), C, \dots, C)\}, \\ N =& M' \cup B, \end{aligned}$$

where $C > 2\,(1+\varepsilon+\sigma)^2$ is a sufficiently large number. As $|M| = d+1$, N contains exactly n boxes. Furthermore, each point in B uniquely dominates the rectangular region

$$[0,1] \times [C(i-1), Ci] \times [C(n-d-i-1), C(n-d-i)] \times [0,C]^{d-3},$$

when considering only points from B. Hence, we have $\mathrm{CON}_B(x) \ge C^{d-1}$ for all $x \in B$. The contribution of x in N can be smaller than the contribution of x in B. However, the contribution cannot decrease by more than the overall hypervolume of M' as every point lying in the one contributing space but not in the other one has to be dominated by a point in M'. Therefore $\mathrm{CON}_N(x) \ge \mathrm{CON}_B(x) - \mathrm{HYP}(M')$. We further know that the hypervolume of M' is bounded from above by $2\,(1+\varepsilon+\sigma)^d$ as each i-th coordinate of a point in M' is less than $1+\varepsilon+\sigma$ for $2 \le i \le d$ and at most $2+\varepsilon+\sigma < 2+2\varepsilon+2\sigma$ for $i = 1$. Hence,

$$\begin{aligned} \mathrm{CON}_N(x) \ge& C^{d-1} - 2\,(1+\varepsilon+\sigma)^d \\ >& 2^{d-1}(1+\varepsilon+\sigma)^{2d-2} - 2\,(1+\varepsilon+\sigma)^d \\ \ge& 4\,(1+\varepsilon+\sigma)^d - 2\,(1+\varepsilon+\sigma)^d \\ =& 2\,(1+\varepsilon+\sigma)^d \\ \ge& \mathrm{HYP}(M'). \end{aligned}$$

This implies that it is better to remove from N all elements in M' than to remove one element in B. Therefore none of S^λ_{opt} or $S^\lambda_{\text{greedy}}$ can contain a point in B for $2 \le \lambda \le d$. Moreover, the contribution of an element $x \in M$ to M is the same as the contribution of the corresponding element $x' \in M'$ to N, as the boxes in B cut away all additional dominated space (every box in B dominates $[0,1] \times [0, 1+\varepsilon+\sigma]^{d-1}$). Hence

$$\begin{aligned} \mathrm{CON}_N(S^\lambda_{\text{opt}}(N)) &= \mathrm{CON}_M(S^\lambda_{\text{opt}}(M)), \\ \mathrm{CON}_N(S^\lambda_{\text{greedy}}(N)) &= \mathrm{CON}_M(S^\lambda_{\text{greedy}}(M)) \end{aligned}$$

for $2 \le \lambda \le d$, which implies that their ratio is at least κ as shown for the case $n = d+1$. □

To validate that such differences indeed occur in real data sets, we have calculated the greedy and the optimal λ-set contribution for some populations on fronts from the DTLZ library [13]. To allow an easy verification of our results we used the populations generated by [22] available from http://www.wfg.csse.uwa.edu.au/hypervolume/. Given these populations of different sizes n, we calculated the optimal λ-set S^λ_{opt} by equation (1) and the greedy λ-set $S^\lambda_{\text{greedy}}$ by equation (2) for various λ. We observed relative differences of up to one third between calculating the contribution greedily or

Test case	d	n	λ	$\frac{\text{CON}(S^{\lambda}_{\text{greedy}})-\text{CON}(S^{\lambda}_{\text{opt}})}{\text{CON}(S^{\lambda}_{\text{opt}})}$
DTLZLinearShape.3d.front.50pts	3	50	5	4.23%
DTLZLinearShape.3d.front.10pts	3	10	9	5.76%
DTLZSphereShape.3d.front.50pts	3	50	6	4.19%
DTLZSphereShape.3d.front.50pts	3	50	7	8.57%
DTLZDiscontinuousShape.5d.front.20pts	5	20	8	2.60%
DTLZDegenerateShape.8d.front.10pts	8	10	3	11.31%
DTLZDegenerateShape.6d.front.10pts	6	10	3	32.64%

Table 1: Some examples of populations on fronts from the DTLZ library [13] where the contributions of the greedy λ-set $S^{\lambda}_{\text{greedy}}$ and the optimal λ-set S^{λ}_{opt} deviate.

optimally. Some representative numbers of larger deviations between both contributions are shown in Table 1. These examples show that for the examined populations the resulting hypervolume is larger (i.e., better as we consider maximization problems) if the λ-setis chosen optimally instead of greedily. This does not imply that the overall search process is slower with greedy selection, but still motivates to use the optimal selection if possible.

3 Algorithm

Consider a set S of boxes in $\mathbb{R}^d$, $n := |S|$. Throughout this chapter, we use the term λ-set for a subset $T \subseteq S$ with $|T| = \lambda$. We also say that T is a λ-subset of S. Similarly, we use $\lambda^{\leq}$-set for denoting any set $T \subseteq S$ with $|T| \leq \lambda$, or say that T is a $\lambda^{\leq}$-subset of S.

The optimal λ-set T^* of S is a λ-set with $\text{CON}_S(T^*)$ minimal among all λ-sets T. This set T^* is the set we would like to discard from our solution set S. The task of finding T^* can be accomplished by computing $\text{HYP}(S \setminus T)$ for all λ-sets T (and $T = \emptyset$), as $\text{CON}_S(T) = \text{HYP}(S) - \text{HYP}(S \setminus T)$. The main idea of our algorithm is that for doing this we do not have to compute these hypervolume measures independently, but can "parallelize" the execution of the algorithm the currently fastest hypervolume algorithm is based on: the algorithm of Overmars and Yap [19]. Therefore, we present their ideas in short and give a sketch of our changes afterwards.

The general framework of the algorithm is the same as the one of Bentley [3]: do a space sweep along dimension d stopping at each endpoint of a box in decreasing order and, inserting the new box, solve the dynamic $(d-1)$-dimensional measure problem. The latter is the same as the problem of computing the hypervolume, but the boxes are given one by one and we have to output the current hypervolume after each box. Bentley's original approach to this dynamic problem took $\mathcal{O}(n^{d-2} \log n)$. Those $(d-1)$-dimensional measures then have to be multiplied by the length of the interval in dimension d we overleaped and summed up to get the overall hypervolume of S. Confer [3] for details and correctness of this formula. Note that Bentley solved a more general problem than just computing the hypervolume. In our context we, other than Bentley, never have to delete boxes, as all boxes have the same lower d-th coordinate 0 (as they all share the origin as a joint corner).

For the tree approach of Overmars and Yap [19] to the dynamic problem, we need

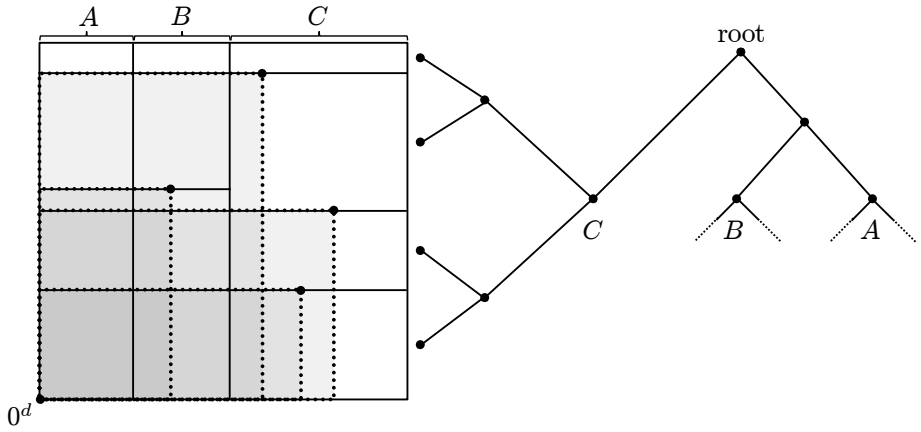

Figure 2: A two-dimensional partition of four boxes and the corresponding partition tree.

some more terminology. For a point $x \in \mathbb{R}^d$ we denote its i-th coordinate by x_i. In general, we will denote a d-dimensional object as d-object if we want to emphasize that it is an object lying in $\mathbb{R}^d$. We consider a d-box B to be a set $[0, b_1] \times \ldots \times [0, b_d]$, so that we can think of B also as the point $(b_1, \ldots, b_d) \in \mathbb{R}^d$. We will use this dualism often, speaking of boxes and points being the same. Moreover, we consider a $(d-1)$-region R, or just region for short, to be a set $[a_1, b_1] \times \ldots \times [a_{d-1}, b_{d-1}]$, i.e., a rectangular region in $\mathbb{R}^{d-1}$. For such a region we define R^* to be the d-region $R \times [0, U]$, where $U := \max\{x_d \mid x \in S\}$ is a fixed upper bound for the d-th coordinates of the points in S. Furthermore, we speak of the *projected* box B_π by dropping the d-th coordinate of the box B, i.e., $B_\pi = [0, b_1] \times \ldots \times [0, b_{d-1}]$. Also, for a set S of points (or boxes) in $\mathbb{R}^d$ we denote by S_π the set of all projected boxes $\{(x_1, \ldots, x_{d-1}) \mid (x_1, \ldots, x_d) \in S\}$.

Definition 1. A d-box B is said to *partially cover* a $(d-1)$-region R if the boundary of B_π intersects the interior of R. B is said to *(fully) cover* R if $R \subseteq B_\pi$.

We speak of the two i-boundaries of a box B being the two sets $\{x \in B \mid x_i = 0\}$ and $\{x \in B \mid x_i = b_i\}$. Additionally, we let the i-interval of a box or region to be its projection on the x_i-axis. Then we speak of a box B being an i-pile of the region R, if for each $1 \leq j \leq d-1, j \neq i$ the j-interval of R is fully contained in the j-interval of B. Less formally that means that within a region R all j-intervals of i-piles of R only differ for $j = i$. We consider a set S of d-boxes to form a trellis in the region R, if each box in S is an i-pile for some $1 \leq i \leq d-1$ in R. See Figure 3 for an illustration of a general trellis as defined by Overmars and Yap. At last, we will need a restricted hypervolume: For any region R and finite point set $T \subset \mathbb{R}^d$ we define $\text{HYP}_{R^*}(T) := \text{VOL}(R^* \cap \bigcup_{x \in T}[0, x_1] \times \ldots \times [0, x_d])$, which is the hypervolume dominated by T restricted to R^*.

To calculate the volume efficiently, Overmars and Yap [19] cleverly use a partitioning of the $(d-1)$-dimensional space by an orthogonal partition tree. There, each node u is associated to a (leaf-)region R_u. The root is associated to a region R_{root} with R^*_{root} being a bounding box for all the boxes in S. We will in this paper always assume R^*_{root} to be the box $\mathbf{BB} = [0, \mathbf{BB}_1] \times \ldots \times [0, \mathbf{BB}_d]$ with $\mathbf{BB}_i = \max\{x_i \mid x \in S\}$. Additionally, the associated region to each node splits up to the two children covering and intersection free, i.e., if $\ell(u)$ and $r(u)$ are the two children of u, then $R_{\ell(u)} \cup R_{r(u)} = R_u$ and $R_{\ell(u)} \cap R_{r(u)}$ has zero Lebesgue measure. In every leaf ℓ they require that any box in

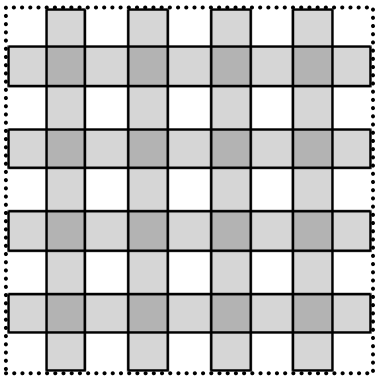

Figure 3: A 2-dimensional trellis for arbitrary boxes as in Overmars and Yap [19]. There, a trellis consists of long vertical (pairwise disjoint) rectangles superposed on long horizontal (pairwise disjoint) rectangles. The dotted rectangle around the trellis shows the corresponding region R. Note that such a configuration can only occur for the more general Klee's Measure Problem, but not for the hypervolume.

S (the problem instance) that partially covers R_ℓ is a pile for R_ℓ, so that the boxes in any leaf form a trellis. Figure 2 gives an example of a partition and the corresponding partition tree.

Overmars and Yap [19] show how to build such a tree with several nice properties. Among others, they prove (in their Lemma 4.2) the following three properties which will be useful in the remainder.

Lemma 2. *The partition tree built by the algorithm of Overmars and Yap has the following properties:*

- *The depth of the tree is* $\mathcal{O}(\log n)$.
- *Each projection of a box in* S *partially covers* $\mathcal{O}(n^{(d-2)/2})$ *leaf-regions.*
- *Each projection of a box in* S *partially covers* $\mathcal{O}(n^{(d-2)/2} \log n)$ *regions of inner nodes.*

Note, we build this tree only for the first $d-1$ dimensions, while Overmars and Yap solve the problem in d dimensions, which explains the difference in the statements.

For inserting or deleting a box in this tree, one only has to update the measure in each of the influenced regions. This can be done in constant time for internal nodes and in logarithmic time for the leafs, as we will see in the next but one section. Hence, this tree helps to efficiently determine the dynamic measure.

Streaming variant

Overmars and Yap [19] also present a streaming variant of their algorithm. It uses less space but elsewise performs the same operations as the tree variant, just in a different ordering. The tree variant can be seen as a sweep in "time" (being the d-th coordinate), where we insert a box into the tree when it is reached in time. We can rearrange this the following way: We traverse the tree, and for each leaf we sweep the "time" inserting all boxes that influence the current leaf. In other words, we do not perform every insertion

one by one on the whole tree structure, but look at the leaf-regions and perform all the insertions that will influence the region at hand. This rearrangement is possible as we know all insertion times beforehand. The benefit of the latter variant is that we do not have to explicitly store the whole tree structure: As Overmars and Yap managed to simulate the splitting of an inner node just by looking at the boxes that influence the associated region, we just need the tree structure implicitly, reducing the amount of storage to $\mathcal{O}(n)$. This variant fits better our purpose of a practical algorithm.

Trellises

What is left is how to deal with the leaf-regions of the tree. Overmars and Yap [19] saw that maintaining the measure of a projected trellis dynamically can be done in $\mathcal{O}(\log n)$: Consider a region R, i.e., a rectangle with side lengths $L_1, \dots, L_{d-1}$. Furthermore, consider the i-piles of this region: by projecting them onto dimension i, we can determine their measure by solving a 1-dimensional measure problem with overlapping intervals, which can be maintained in $\mathcal{O}(\log n)$ per update by an interval tree. This way, we get M_i, the measure of the 1-dimensional problem of the i-piles. Then we can compute the measure of the projected trellis easily as

$$\prod_{i=1}^{d-1} L_i - \prod_{i=1}^{d-1} (L_i - M_i)$$

as explained in [19].

Beume and Rudolph [4, 5] noticed that in the case of the hypervolume indicator the measures M_i can be maintained even in constant time, since we do not delete boxes and the interval overlapped by the i-piles is always of the form $[0, r]$ for some $r \in \mathbb{R}$, so that we just have to save the largest right end of such an interval, which can be updated in $\mathcal{O}(1)$.

Sketch of our algorithm

Roughly speaking, the algorithm of Overmars and Yap can be summarized as follows:

- By building the partition tree, compute for all leaf-regions R the hypervolume $\text{HYP}_{R^*}(S)$ of the space dominated by S restricted to R^*.
- Sum up these volumes.

The crucial observation is that the space dominated by S restricted to R^* as considered in the first step forms a trellis, whose hypervolume can be determined efficiently.

As sketched at the beginning of Section 3, we want to compute all the hypervolumes $\text{HYP}(S \setminus T)$ for λ-sets T in parallel. Observing that we can use the same partitioning tree for $S \setminus T$ as for S, we can come up with a simple adaption of the above method:

- By building the partition tree, compute for all leaf-regions R the hypervolume $\text{HYP}_{R^*}(S \setminus T)$ for all λ-subsets T of S.
- Sum up these volumes independently to get $\text{HYP}(S \setminus T)$ for each T.

One way to carry out the first step is described in the following. For this, we compute for every leaf-region R and all $\lambda^{\leq}$-sets U the following volumes M_U^R.

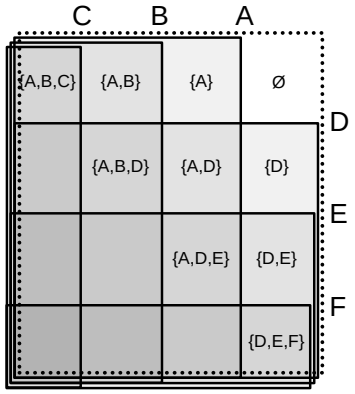

Figure 4: The first two dimensions of a 3-dimensional trellis consisting of the hypervolume-boxes defined by $S = \{A, B, C, D, E, F\}$. Here, a trellis consists of boxes that cover the region completely in each of the $(d-1)$ dimensions except one. The dotted rectangle indicates the corresponding region R.

Definition 2. *The volume M_U^R denotes the volume of the space in R^* that is* not *dominated by $S \setminus U$, but is dominated by every $S \setminus W$ for $W \subsetneq U$.*

This way every point in R^* not dominated by $S \setminus T$ is counted in exactly one of the measures M_U^R with $U \subseteq T$. We get

$$\text{HYP}_{R^*}(S \setminus T) = \text{VOL}(R^*) - \sum_{U \subseteq T} M_U^R. \tag{3}$$

Using this and postponing the summation for each T we can restate the method now as follows:

- Compute the measures M_U^R for all leaf-regions R and $\lambda^{\leq}$-sets U
- Sum up these measures to get $M_U := \sum_R M_U^R$ (the sum goes over all leaf-regions R)
- For each λ-set T compute $\text{HYP}(S \setminus T) = \text{VOL}(R^*_{\text{root}}) - \sum_{U \subseteq T} M_U$

The subtle point making this method superior to the naïve approach is that most of the M_U^R's are actually zero. For seeing this, let us take a closer look on how one would compute the values M_U^R: Inside R^* we can do a space sweep along dimension d, just as in the algorithm of Bentley [3], stopping at each of the d-th coordinates of the points in S in decreasing order. At the stop for $x \in S$, we have to insert the box x and compute a $(d-1)$-dimensional measure, namely the volume $M_U^{R,x}$:

Definition 3. *The volume $M_U^{R,x}$ denotes the volume of the $(d-1)$-dimensional space in R that is* not *dominated by $S_\pi^x \setminus U_\pi$, but is dominated by every $S_\pi^x \setminus W_\pi$ for $W \subsetneq U$, where $S^x = \{y \in S \mid y_d \geq x_d\}$ denotes the set of already inserted boxes.*

We multiply this measure by the covered distance $x_d - x_d^N$ in the d-th dimension, where x_d^N is the d-th coordinate of the next stop. This is summed up over all stops to get the measure M_U^R.

This way, we reduced the computation of M_U^R to $M_U^{R,x}$, which is a measure inside the first $d-1$ dimensions of a trellis, which have a fairly simple geometric structure, as depicted in Figure 4. In the picture we can see for each part, to which measure $M_U^{R,x}$ it corresponds. The set $\{A, D\}$ for example marks the space corresponding to $M_{\{A,D\}}^{R,x}$. There we can also verify equation (3) (reduced to $d-1$ dimensions): The hypervolume of the space dominated by all but the points A and D equals the total volume of R minus the volumes of the parts marked with $\{A, D\}$, $\{A\}$, $\{D\}$ and $\emptyset$, i.e., the subsets of $\{A, D\}$.

Moreover, observe that $M_U^{R,x}$ is non-zero only if U contains the largest ℓ_i i-piles in R for some ℓ_i, but no other i-pile, $i = 1, \dots, d-1$. In the example we may choose $\emptyset, \{A\}, \{A, B\}$ or $\{A, B, C\}$ being the 1-piles contained in U to get a non-zero $M_U^{R,x}$, but not, e.g., only $\{B\}$. As we need to compute M_U only for $\lambda^{\leq}$-sets U, we additionally condition on $|U| = \sum_{i=1}^{d-1} \ell_i \leq \lambda$. But then there are at most as many non-zero $M_U^{R,x}$'s, as there are $(d-1)$-tuples $(\ell_1, \dots, \ell_{d-1}) \in \mathbb{N}_0^{d-1}$ with $\sum_{i=1}^{d-1} \leq \lambda$, which is a constant number for d and λ being constant. This is why there is only a constant number of non-zero $M_U^{R,x}$'s for R and x fixed, which implies that there is only a small number of non-zero M_U^R's for R fixed. As we will see in the next section in detail, we can even determine those non-zero values quickly, even in the same asymptotic runtime as we need for the standard algorithm of Overmars and Yap. Computing the hypervolumes $\text{HYP}(S \setminus T)$ for all λ-sets T can then be accomplished by summing up all M_U with $U \subseteq T$, as pointed out above, from which we can compute $\text{CON}_S(T) = \text{HYP}(S) - \text{HYP}(S \setminus T)$, as we get $\text{HYP}(S) = \text{VOL}(R^*_{\text{root}}) - M_\emptyset$ for free, and thus quickly determine the optimal λ-set.

Details of the algorithm

COMPUTEMEASURES

One obvious arising problem concerns boxes that fully cover a region of some inner node of the tree. In such a case Overmars and Yap collapse the interval in dimension d, where the region is fully covered, into a single moment, memorizing the deleted volume, and recur. We may not do this, as the fully covering box may be in the set T we disregard, so that in $\text{HYP}(S \setminus T)$ the region is not fully covered. This is why we do not collapse any intervals, but have to pass the fully covering boxes to the recursive calls, so that we can deal with them in the leaf-nodes. Note that the runtime analysis of Overmars and Yap's algorithm does not rely at any point on collapsing intervals, which is why we get the same asymptotic runtime. It does, however, rely on the fact, that inside a leaf-node we spend time $\mathcal{O}(|S'| \log n)$ and in an inner node $\mathcal{O}(|S'|)$, where S' is the set of boxes in S that partially cover the region at hand. Hence, we may not pass *all* fully covering boxes to the recursive calls to be inside this time bound. Luckily, any measure M_U^R is zero, if the fully covering boxes of R contained in U are not the ℓ largest ones for some ℓ, i.e., have largest d-th coordinate among all fully covering boxes. This stems from the fact that for fully covering boxes $x \in U$, $y \notin U$ with d-th coordinates $x_d \leq y_d$ we have $\text{HYP}_{R^*}(S \setminus U) = \text{HYP}_{R^*}(S \setminus (U \setminus \{x\}))$, so that $M_U^R = 0$. This is why we need to pass only the up to $\lambda + 1$ largest covering boxes to the recursive calls. Since this is a constant number, we do not increase the runtime in an inner node or leaf-node asymptotically.

The streaming variant of [19] is essentially the same as the algorithm COMPUTEMEASURES (cf. Algorithm 1), only that we added the set COV containing the up to $\lambda+1$ largest covering boxes, i.e., out of the set C of boxes that fully cover the region R at hand (including covering boxes of any parent region) we save the $\min\{\lambda+1, |C|\}$ many ones with greatest d-th coordinate. This set is updated determining the set $U \subseteq S$ of boxes fully covering the region R, where S is the current set of boxes. Everything after determining the set COV$'$ is copied from Overmars and Yap: We proceed by computing the measures in a trellis, if the remaining boxes S' form one, and by splitting the region R into two regions R_1, R_2 and recursing, otherwise. For splitting we need the sets S'_1 and S'_2, where S'_1 is the set of all boxes in S' that have a 1- or 2- or ... or $(i-1)$-boundary in R and S'_2 is the set of boxes in S' that do not have such a boundary in R. For details of this splitting method confer [19]. Note that we never split a region along dimension d as is implied by the use of Overmars and Yaps splitting method and our definition of trellis (which considers only the first $d-1$ dimensions).

Algorithm 1 COMPUTEMEASURES$(R, S, i, \text{COV}, \lambda)$ computes the measures M_U of the $\lambda^{\leq}$-subsets U of the set of boxes S in the region $R^* \subseteq \mathbb{R}^d$, where i is the current splitting dimension and COV is a set containing the up to $\lambda+1$ largest covering boxes.

```
discard boxes in S not influencing R
determine the set U ⊆ S of boxes fully covering R
S' := S \ U
determine the new set COV' ⊆ COV ∪ U
if the boxes in S' form a trellis in R then
  COMPUTEMEASURESTRELLIS(R, S', COV', λ)
else
  determine the sets S'_1 and S'_2 (as defined on page 13)
  if S'_1 ≠ ∅ then
    split R into R_1, R_2 along the median i-boundary in S'_1
    COMPUTEMEASURES(R_1, S', i, COV', λ)
    COMPUTEMEASURES(R_2, S', i, COV', λ)
  else if S'_2 contains more than √n i-boundaries then
    split R into R_1, R_2 along the median i-boundary in S'_2
    COMPUTEMEASURES(R_1, S', i, COV', λ)
    COMPUTEMEASURES(R_2, S', i, COV', λ)
  else
    COMPUTEMEASURES(R, S', i+1, COV', λ)
  od
od
```

The procedure COMPUTEMEASURESTRELLIS will need the boxes S' to be sorted by d-th coordinate. This can be achieved easily by sorting the boxes before the first call of COMPUTEMEASURES and maintaining this ordering during all steps of COMPUTEMEASURES, without increasing the overall asymptotic runtime. Hence, we may assume in the following, that S' in the input of COMPUTEMEASURESTRELLIS is sorted.

Computing the set COV$'$ can be done in $\mathcal{O}(|S|)$, assuming λ to be constant. Hence, as long as we provide a COMPUTEMEASURESTRELLIS-function which runs in $\mathcal{O}(|S'| \log n)$ for the set of boxes S', which is exactly the same runtime as Overmars and Yap's, we do not increase the overall runtime of $\mathcal{O}(n^{d/2} \log n)$ of their algorithm.

For determining the needed storage consider the following trick of Overmars and Yap [19]: If we save the boxes in U we can reconstruct the old S by joining U and S'. Hence, we can send S' down the recursion, not copying it; we just have to reconstruct it at the end of the recursion call. This way, no box is saved at two places at any time, so that the overall space for the sets S and U is just $\mathcal{O}(n)$. Since the size of COV is λ and thus constant, we can save it normally, getting an additional space needed of $\mathcal{O}(\lambda \log n)$, as the recursion depth equals the depth of the partition tree which is $\mathcal{O}(\log n)$ by Lemma 2, so that we overall need a storage of $\mathcal{O}(n)$. Note that if we would not follow this trick we had a storage of $\mathcal{O}(n \log n)$, i.e., $\mathcal{O}(n)$ in each of the $\mathcal{O}(\log n)$ levels of the recursion.

COMPUTEMEASURESTRELLIS

To complete the description of COMPUTEMEASURES we have to provide the procedure COMPUTEMEASURESTRELLIS, which will report the measures M_U^R. These measure will then be summed up to get M_U from which we can directly compute the hypervolume dominated by any $S \setminus T$, for T a λ-subset of S, as sketched in the preceeding section. One way to compute these measures is given in Algorithm 2.

Algorithm 2 COMPUTEMEASURESTRELLIS$(R, S, \text{COV}, \lambda)$ computes the measures M_U^R for each $\lambda^{\leq}$-subset U of S, where the boxes in S restricted to $R^* \subseteq \mathbb{R}^d$ form a trellis and COV is a set containing the up to $\lambda + 1$ largest covering boxes.

discard boxes in S not influencing R
set $A_j^i :=$ **undef** $(1 \leq i \leq d-1, 1 \leq j \leq \lambda + 1)$
set $A_0^i := R$ $(1 \leq i \leq d-1)$
$S := S \cup \text{COV} \cup \{(0, \ldots, 0)\}$
$x_d^L := \mathbf{BB}_d$
initialize M_U^R's to 0
for all $x \in S$ ordered by decreasing x_d **do**
  for all $(k_1, \ldots, k_{d-1}) \in \mathbb{N}_0^{d-1}$ with $\sum_{i=1}^{d-1} k_i \leq \lambda$ and $A_{k_i}^i$ defined for all i **do**
    $U := \{A_j^i \mid 1 \leq i < d \text{ and } 1 \leq j \leq k_i\}$
    $M_U^{R,x} := \prod_{i=1}^{d-1}((A_{k_i}^i)_i - (A_{k_i+1}^i)_i)$
    $M_U^R := M_U^R + (x_d^L - x_d) \cdot M_U^{R,x}$
  od
  if x is a k-pile: update A_j^k $(1 \leq j \leq \lambda + 1)$
  $x_d^L := x_d$
od
for all U with non-zero M_U^R **do**
  $M_U := M_U + M_U^R$
od

There, at first, we remove all boxes from the set of boxes S that do not influence the current region R at all. The variable x_d^L is going to be the d-th coordinate of the last box inserted and initialized to $\mathbf{BB}_d$. We will maintain an ordered list of the up to $\lambda + 1$ largest i-piles of the $(d-1)$-region R for each i, i.e., the i-piles with greatest i-th coordinate. Those will be the boxes $A_j^i,\ 1 \leq i < d, 1 \leq j \leq \lambda + 1$, which are undefined initially and get updated every time we insert a box, so that A_1^i is the greatest i-pile, A_2^i the second greatest and so on. We use $(A)_i$ to denote the maximal i-th coordinate of a

point in a box A, i.e., A_i (viewed as a point). For simplicity, we define $(A)_i$ to be 0 if A is undefined and $A_0^i = R$ to be the region itself for each i.

Going further, we add COV to S, which we need, since each fully covering box is a pile of R, and those boxes are not already in S. Since we need the set S sorted according to d-th coordinate and S was sorted in the beginning, we have to insert the points in COV into S properly, which can be done in $\mathcal{O}(|S|)$ as $|\text{COV}| = \mathcal{O}(1)$. Additionally, we need to add the dummy point $(0, \ldots, 0)$ to S, as we want to sweep along the entire d-interval of R^*, i.e., we want to end at 0.

Now, we go through all the boxes in S ordered by d-th coordinate in decreasing order. For each point $x \in S$, we go through all the tuples $(k_1, \ldots, k_{d-1}) \in \mathbb{N}_0^{d-1}$ with $\sum_{i=1}^{d-1} k_i \leq \lambda$, but only those, for which $A_{k_i}^i$ is not undefined. Each such tuple corresponds to a set $U = \{A_j^i \mid 1 \leq i < d \text{ and } 1 \leq j \leq k_i\}$, where all occurring A_j^i are defined for the condition mentioned before. We then compute the $(d-1)$-dimensional measure not covered by $S^x \setminus U$, but by $S^x \setminus W$ for any $W \subsetneq U$, where $S^x = \{y \in S \mid y_d > x_d\}$ denotes the set of the already inserted boxes. This measure is $M_U^{R,x} = \prod_{i=1}^{d-1}((A_{k_i}^i)_i - (A_{k_i+1}^i)_i)$. It has to be multiplied by the length of the interval of the d-th coordinate we are currently regarding, which is $x_d^L - x_d$ (as x_d^L was the last d-th coordinate of an insertion). The resulting product has to be added to M_U^R. We implicitly initialize the measures M_U^R to 0. Also, we do not want to explicitly save each value M_U^R, as most of them are zero, but save only the non-zero ones. Both points can be achieved by using a dynamic hash-table, that contains U and M_U^R iff M_U^R is non-zero.

Afterwards, we determine the number $k, 1 \leq k < d$, for which x is a k-pile in R. If this number is not unique, which can only happen if the box fully covers R, assign an arbitrary $1 \leq k < d$. Then we update the largest k-piles $A_j^k, 1 \leq j \leq \lambda + 1$, i.e., we insert x at the correct position, shifting all smaller ones by one position.

In the end we report the computed measures M_U^R, i.e., we add them to M_U. Here, again, we will implicitly initializing each M_U with 0 (before the start of COMPUTEMEASURES) and save U and M_U in a dynamic hash-table for every non-zero M_U.

Considering the runtime we see that everything inside the main loop can be done in constant time: Since d and λ are considered to be constant, we have to update a constant number of boxes A_j^i. Furthermore, there are at most $(\lambda+1)^{d-1}$ many tuples $(k_1, \ldots, k_{d-1})$, since every entry lies between 0 and λ. Since we can view this as assigning at most λ many ones to $d-1$ many buckets, the number of tuples is also bounded from above by $\sum_{i=0}^{\lambda}(d-1)^i < (d-1)^{\lambda+1}$. All we do with such a tuple can be done in constant time for the same reason, which establishes, that COMPUTEMEASURESTRELLIS runs in $\mathcal{O}(|S|)$. Observe that this is even better than Overmars and Yaps runtime of $\mathcal{O}(|S| \log n)$, so that we definitely get their overall asymptotic runtime of $\mathcal{O}(n^{d/2} \log n)$.

Correctness

We now show that the above methods are indeed correct.

Lemma 3. *The measures M_U computed by* COMPUTEMEASURES *satisfy the following equa-*

tion for any $\lambda^{\leq}$-set T of S:

$$\text{HYP}(S \setminus T) = \text{VOL}(\boldsymbol{BB}) - \sum_{U \subseteq T} M_U$$

Proof. The described algorithm partitions the bounding box **BB** into a number of leaf-regions that contain trellises. Since we sum up over all of those regions, all we have to show is that for each region R for which we call COMPUTEMEASURESTRELLIS it holds that

$$\text{HYP}_{R^*}(S \setminus T) = \text{VOL}(R^*) - \sum_{U \subseteq T} M_U^R.$$

Now, inside R^* we sweep along the d-th dimension considering intervals $[x_d, x_d^L]$, where the boxes influencing the $(d-1)$-dimensional measures stay the same, and sum up weighted by $x_d^L - x_d$. Since we start with $x_d^L = \mathbf{BB}_d$ and end at the dummy point $(0, \dots, 0)$ with $x_d = 0$ our sweep indeed covers the interval $[0, \mathbf{BB}_d]$. Hence, the summation along dimension d is correct as long as it holds that at the stop for $x \in S$, with $S^x = \{y \in S \mid y_d > x_d \text{ and } y \text{ partially covers } R\}$ the set of already inserted boxes:

$$\text{HYP}_R(S_\pi^x \setminus T_\pi) = \text{VOL}(R) - \sum_{U \subseteq T} M_U^{R,x}. \tag{4}$$

Note that we are here dealing with $(d-1)$-dimensional volumes, which is why we used the projected set of boxes S_π^x and T_π.

Let T be of the form as in the pseudo code, i.e., $T = \{A_j^i \mid 1 \leq i < d \text{ and } 1 \leq j \leq k_i\}$ for some $(k_1, \dots, k_{d-1}) \in \mathbb{N}_0^{d-1}$ with $\sum_{i=1}^{d-1} k_i \leq \lambda$ and all $A_{k_i}^i$ defined. We compute (non-zero) measures $M_U^{R,x}$ only for subsets $U \subseteq T$ of the form $\{A_j^i \mid 1 \leq i < d \text{ and } 1 \leq j \leq \ell_i\}$ for some $(\ell_1, \dots, \ell_{d-1}) \in \mathbb{N}_0^{d-1}$ with $\ell_i \leq k_i$ for all i. Thus, we have:

$$\begin{aligned}
\sum_{U \subseteq T} M_U^{R,x} &= \sum_{\substack{(\ell_1,\dots,\ell_{d-1}) \in \mathbb{N}_0^{d-1} \\ \ell_i \leq k_i \text{ for all } i}} M^{R,x}_{\{A_j^i \mid 1 \leq i < d \text{ and } 1 \leq j \leq \ell_i\}} \\
&= \sum_{\substack{(\ell_1,\dots,\ell_{d-1}) \in \mathbb{N}_0^{d-1} \\ \ell_i \leq k_i \text{ for all } i}} \prod_{i=1}^{d-1} ((A_{\ell_i}^i)_i - (A_{\ell_i+1}^i)_i) \\
&= \prod_{i=1}^{d-1} \sum_{\ell_i=0}^{k_i} ((A_{\ell_i}^i)_i - (A_{\ell_i+1}^i)_i) \\
&= \prod_{i=1}^{d-1} ((A_0^i)_i - (A_{k_i+1}^i)_i) \\
&= \prod_{i=1}^{d-1} (R_i - (A_{k_i+1}^i)_i).
\end{aligned}$$

There, we denote by R_i the maximal i-th coordinate of a point in R. Observe that $S_\pi^x \setminus T_\pi$ is of a very simple form (restricted to R): It forms a trellis where the maximal i-pile is $A_{k_i+1}^i$. This means that the space inside R not overlapped by this trellis is a

rectangle with side lengths $R_i - (A^i_{k_i+1})_i$ for $i = 1, \ldots, d-1$, so that we established the equality

$$\sum_{U \subseteq T} M_U^{R,x} = \text{VOL}(R) - \text{HYP}_R(S^x_\pi \setminus T_\pi).$$

Note that the above argument also makes sense if $A^i_{k_i+1}$ is not defined, since then we have $(A^i_{k_i+1})_i = 0$. This gives us correctness for sets T of the aforementioned form.

If, on the other hand, T is not of the indicated form, then it has some maximal subset $T' \subseteq T$, which is of this form, i.e., $T' = \{A^i_j \mid 1 \leq i < d \text{ and } 1 \leq j \leq k_i\}$ for some $(k_1, \ldots, k_{d-1}) \in \mathbb{N}_0^{d-1}$. Since T' is maximal, either $A^i_{k_i+1}$ is not contained in T or it is not defined. In both cases every box in $T_\pi \setminus T'_\pi$ is included in some box in $S^x_\pi \setminus T'_\pi$, so that those boxes do not influence the measure in R, i.e., we have $\text{HYP}_R(S^x_\pi \setminus T_\pi) = \text{HYP}_R(S^x_\pi \setminus T'_\pi)$. Also, we will report a measure for a set $U \subseteq T$ only if $U \subseteq T'$ by construction, so that we have, using the former case:

$$\text{HYP}_R(S^x_\pi \setminus T_\pi) = \text{HYP}_R(S^x_\pi \setminus T'_\pi) = \text{VOL}(R) - \sum_{U \subseteq T'} M_U^{R,x} = \text{VOL}(R) - \sum_{U \subseteq T} M_U^{R,x}$$

This shows the desired equality. It also implies that any box which is not among the largest λ in one dimension does not contribute to our measures at all, which also makes clear why we only need the $\lambda + 1$ largest covering boxes in COV. □

Putting everything together

After we computed the measures M_U, we can compute the actual contribution of a λ-set T easily, using Lemma 3. We have:

$$\begin{aligned}\text{CON}_S(T) =& \text{HYP}(S) - \text{HYP}(S \setminus T)\\ =& (\text{VOL}(\mathbf{BB}) - M_\emptyset) - (\text{VOL}(\mathbf{BB}) - \sum_{U \subseteq T} M_U)\\ =& \sum_{\emptyset \neq U \subseteq T} M_U\end{aligned}$$

This confirms the following combined procedure:

Algorithm 3 COMPUTEOPTIMALSUBSET(S, λ) computes the optimal λ-subset T of the set of boxes S in $\mathbb{R}^d$.

```
initialize M_U's to 0
COMPUTEMEASURES(BB, S, 1, ∅, λ)
return argmin{Σ_{∅≠U⊆T} M_U | T ⊆ S, |T| = λ}
```

Since T has size at most λ, it has at most 2^λ subsets, which is a constant. Hence, given the measures M_U we can compute the contribution of all λ-sets in $\mathcal{O}(n^\lambda)$ (as their number is bounded by this), so that we get an overall runtime of $\mathcal{O}(n^{d/2} \log n + n^\lambda)$ for COMPUTEOPTIMALSUBSET. Its correctness follows directly from Lemma 3.

As mentioned in the preceding section, we need a big hash-table to store the non-zero measures M_U. Since there are $\mathcal{O}(n^\lambda)$ $\lambda^{\leq}$-subsets of a size-n set, there are $\mathcal{O}(n^\lambda)$ entries in the hash. On the other hand, by Lemma 2 each d-box partially covers $\mathcal{O}(n^{(d-2)/2})$ many leaf-regions. As COMPUTEMEASURESTRELLIS runs in $\mathcal{O}(|S'|)$, where S' is the set of boxes in S that partially cover the region R at hand, we report $\mathcal{O}(|S'|)$ non-zero measures M_U^R in each region. This way, we get an upper bound of $\mathcal{O}(n^{d/2})$ reported non-zero measures. Since there are at most this many entries in the hash-table, COMPUTEOPTIMALSUBSET needs a space of $\mathcal{O}(\min(n^{d/2}, n^\lambda))$, using a dynamically growing hash-table.

4 Discussion

We have presented an algorithm which calculates the optimal λ-set $S^\lambda_{\text{opt}}(M)$ of a population of size $n = |M|$ in time $\mathcal{O}(n^{d/2} \log n + n^\lambda)$ for $d > 2$. For $d > 3$ this improves all previously published algorithms by a factor of $n^{\min\{\lambda, d/2\}}$.

For small λ ($\lambda \leqslant d/2$) the algorithm gives an improvement in the runtime of the calculation of the hypervolume by a factor of order n^λ. Hence even for the greedy calculation of $S^1_{\text{opt}} = S^1_{\text{greedy}}$, we have a speed up by a factor of n.

For very large λ the algorithm might still be intractable. It is open whether this can be avoided. Our algorithm allows the calculation of S^λ_{opt} in the same time as S^1_{opt} if $\lambda \leqslant d/2$. We therefore suggest the following compromise between S^λ_{opt} and $S^\lambda_{\text{greedy}}$ for large λ:

$$\begin{aligned} S^\lambda_{\text{comp}}(M) &:= S^\lambda_{\text{opt}}(M) && \text{for all } \lambda \leqslant d/2, \\ S^{\lambda+d/2}_{\text{comp}}(M) &:= S^\lambda_{\text{comp}}(M) \cup S^{d/2}_{\text{opt}}(M \setminus S^\lambda_{\text{comp}}(M)) \\ & && \text{for all } \lambda > d/2. \end{aligned}$$

As $\text{CON}(S^\lambda_{\text{greedy}}) \geqslant \text{CON}(S^\lambda_{\text{comp}}) \geqslant \text{CON}(S^\lambda_{\text{opt}})$ for all λ, above improved greedy algorithm returns λ-sets with the same or maybe smaller contributions than the classical greedy algorithm within the same asymptotic runtime.

References

[1] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Investigating and exploiting the bias of the weighted hypervolume to articulate user preferences. In *Proc. 11th annual Conference on Genetic and Evolutionary Computation (GECCO '09)*, pages 563–570, 2009.

[2] J. Bader and E. Zitzler. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 2010+. To appear.

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

[4] N. Beume. S-Metric calculation by considering dominated hypervolume as Klee's measure problem. *Evolutionary Computation*, 17(4):477–492, 2009.

[5] 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)*, pages 233–238, 2006.

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

[7] 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(5):1075–1082, Oct. 2009.

[8] L. Bradstreet, L. Barone, and L. While. Maximising hypervolume for selection in multi-objective evolutionary algorithms. In *Proc. IEEE Congress on Evolutionary Computation (CEC '06)*, pages 6208–6215, 2006.

[9] 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)*, volume 5369 of *Lecture Notes of Computer Science*, pages 436–447, 2008.

[10] K. Bringmann and T. Friedrich. Don't be greedy when calculating hypervolume contributions. In *Proc. 10th International Workshop on Foundations of Genetic Algorithms (FOGA '09)*, pages 103–112, 2009.

[11] K. Bringmann and T. Friedrich. Approximating the least hypervolume contributor: NP-hard in general, but fast in practice. In *Proc. 5th International Conference on Evolutionary Multi-Criterion Optimization (EMO '09)*, pages 6–20, 2009.

[12] D. Brockhoff and E. Zitzler. Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. In *Proc. IEEE Congress on Evolutionary Computation (CEC '07)*, pages 2086–2093, 2007.

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

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

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

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

[17] J. D. Knowles, D. W. Corne, and M. Fleischer. Bounded archiving using the Lebesgue measure. In *Proc. IEEE Congress on Evolutionary Computation (CEC '03)*, volume 4, pages 2490–2497, Dec 2003.

[18] B. Naujoks, N. Beume, and M. T. M. Emmerich. Multi-objective optimisation using S-metric selection: application to three-dimensional solution spaces. In *Proc. IEEE Congress on Evolutionary Computation (CEC '05)*, pages 1282–1289, 2005.

[19] M. H. Overmars and C.-K. Yap. New upper bounds in Klee's measure problem. *SIAM J. Comput.*, 20(6):1034–1045, 1991.

[20] T. Suttorp, N. Hansen, and C. Igel. Efficient covariance matrix update for variable metric evolution strategies. *Mach. Learn.*, 75(2):167–197, 2009.

[21] 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)*, pages 2225–2232, 2005.

[22] R. L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *IEEE Trans. Evolutionary Computation*, 10(1):29–38, 2006.

[23] 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)*, volume 4, pages 415–420, 2007.

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

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

[26] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evolutionary Computation*, 3(4):257–271, 1999.

[27] 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(2):117–132, 2003.

[28] E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *Proc. Fourth International Conference on Evolutionary Multi-Criterion Optimization (EMO '07)*, volume 4403 of *Lecture Notes of Computer Science*, pages 862–876, 2007.