Greedy Hypervolume Subset Selection in Low Dimensions[*]

Andreia P. Guerreiro apg@dei.uc.pt
CISUC, Department of Informatics Engineering, University of Coimbra, Pólo II,
P-3030 290 Coimbra, Portugal

Carlos M. Fonseca cmfonsec@dei.uc.pt
CISUC, Department of Informatics Engineering, University of Coimbra, Pólo II,
P-3030 290 Coimbra, Portugal

Luís Paquete paquete@dei.uc.pt
CISUC, Department of Informatics Engineering, University of Coimbra, Pólo II,
P-3030 290 Coimbra, Portugal

Abstract

Given a non-dominated point set $\mathrm{X} \subset \mathbb{R}^d$ of size n and a suitable reference point $r \in \mathbb{R}^d$, the Hypervolume Subset Selection Problem (HSSP) consists of finding a subset of size $k \leq n$ that maximizes the hypervolume indicator. It arises in connection with multiobjective selection and archiving strategies, as well as Pareto-front approximation post-processing for visualization and/or interaction with a decision maker. Efficient algorithms to solve the HSSP are available only for the 2-dimensional case, achieving a time complexity of $O(n(k + \log n))$. In contrast, the best upper bound available for $d > 2$ is $O(n^{d/2} \log n + n^{n-k})$. Since the hypervolume indicator is a monotone submodular function, the HSSP can be approximated to a factor of $(1 - 1/e)$ using a greedy strategy. In this paper, greedy $O(n(k + \log n))$-time algorithms for the HSSP in 2 and 3 dimensions are proposed, matching the complexity of current exact algorithms for the 2-dimensional case, and considerably improving upon recent complexity results for this approximation problem.

Keywords

Hypervolume Indicator, Multiobjetive Optimization, Subset Selection, Monotone Submodular Function, Greedy Algorithm

1 Introduction

Multiobjective optimization consists of finding solutions that minimize, without loss of generality, a vector of $d > 1$ objective functions, $f(x) = (f_1(x), \ldots, f_d(x))$. Each solution x is an element of a decision space, and is mapped onto a point in a d-dimensional objective space. Due to conflicting objectives, there is usually no ideal solution to a

[*] This paper is an extended version of a GECCO 2015 paper (Guerreiro et al., 2015) providing a more thorough analysis of the algorithms proposed, both with respect to time complexity and to experimental results. Additionally, improved illustrative examples are used in the description of the main algorithm.

multiobjective optimization problem, but multiple Pareto-optimal, or efficient, solutions (Ehrgott, 2005). In this paper, only the points in objective space are considered.

Let u and v be two points in objective space. Point u is said to weakly dominate point v, denoted by $u \leq v$, if $u_i \leq v_i$ for all $i = 1, \ldots, d$, and to dominate v if, in addition, $u \neq v$. If neither $u \leq v$ nor $v \leq u$, then u and v are said to be incomparable, or mutually non-dominated. A point is said to be non-dominated if no other point dominates it, and the set of all non-dominated points is called the Pareto front (Ehrgott, 2005; Deb, 2001).

Quality indicators map a set of n (mutually non-dominated) points onto a real value, and are used to evaluate the quality of discrete Pareto-front approximations. Due to its seemingly unique properties (Auger et al., 2009), the hypervolume indicator (Zitzler and Thiele, 1998; Knowles et al., 2003) is used extensively, both in performance studies and in multiobjective selection and archiving strategies. Unfortunately, the hypervolume indicator becomes computationally demanding as the number of objective dimensions grows. Its time complexity is $\Theta(n \log n)$ in up to 3 dimensions (Beume et al., 2009), but, for general d, the best upper bound known to date is $O(n^{d/3} \operatorname{polylog} n)$ (Chan, 2013). So far, the fastest algorithms in practice (HV4D, Guerreiro et al. (2012), for $d = 4$, and WFG, While et al. (2012), for $d > 4$) are asymptotically slower.

The problem of selecting k out of n points that maximize the hypervolume indicator commonly arises in multiobjective selection and archiving. This is known as the Hypervolume Subset Selection Problem (HSSP) (Bader and Zitzler, 2011), and algorithms to solve it exactly are available only for 2 dimensions, with $O(n(k + \log n))$ (Bringmann et al., 2014) and $O(k(n-k) + n \log n)$ (Kuhn et al., 2016) time complexity. For $d > 2$, the equivalent problem of determining a subset of $n - k$ points that contribute the least hypervolume to the original set can be solved in $O(n^{\frac{d}{2}} \log n + n^{n-k})$ (Bringmann and Friedrich, 2010).

Since the hypervolume indicator is a monotone submodular function (Ulrich and Thiele, 2012), and the HSSP consists of maximizing it subject to a cardinality constraint, an $(1 - 1/e)$-approximation to the hypervolume of an optimal subset may be obtained with an (incremental) greedy strategy (Nemhauser et al., 1978). Based on this result, Friedrich and Neumann (2014) show that a particular EMO algorithm (GSEMO) obtains such an approximation in an expected $O(n^2(k + \log n))$ number of steps, for any number of dimensions. However, their analysis does not take into account the time required to compute the hypervolume of the subsets evaluated in the process.

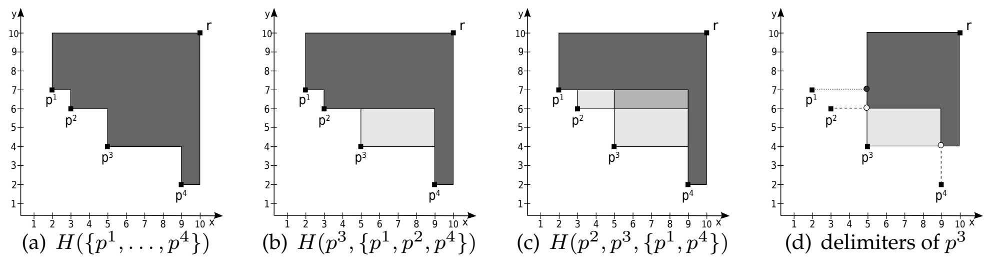
Greedy heuristics for the HSSP were also considered by Bradstreet et al. (2007), who studied both incremental and decremental strategies. In the incremental case, k points are greedily selected one at a time so as to maximize the increase in hypervolume at each step. In contrast, in the decremental case, $n - k$ points are removed one at a time from the original set of n points so as to minimize the loss of hypervolume in each step. The decremental case was also considered by Bader and Zitzler (2011), and was analyzed by Bringmann and Friedrich (2010), who showed that the decremental greedy solution may be very far from optimal, considering the volume left out by the $n - k > 1$ points discarded. In other words, the ratio between the volumes left out in the greedy solution and in the optimal solution may be very large.

As noted also by Bradstreet et al. (2007), both incremental and decremental greedy solutions for the HSSP can be easily computed using existing algorithms to compute hypervolume contributions (e.g. Emmerich and Fonseca (2011); Bringmann and Friedrich (2010)) or even just the hypervolume indicator (e.g. Guerreiro et al. (2012); While et al. (2012)). For example, for $d = 3$, a decremental greedy algorithm with $O((n-k)n \log n)$-time complexity is obtained using the EF algorithm (Emmerich and Fonseca, 2011) to compute all contributions in $O(n \log n)$ time. For the incremental greedy approach, an $O(nk^2)$-time complexity is achieved by iterating over the procedure used in the HV4D (Guerreiro et al., 2012) algorithm to compute single 3-dimensional contributions in linear time.

Directly solving the HSSP, even if only approximately, provides an interesting means of implementing selection in Evolutionary Multiobjective Optimization Algorithms (EMOAs). Indeed, EMOAs frequently work as a combination of generation and archiving strategies, where k solutions are maintained in the parental population (the archive), and a number, m, of offspring are generated from the parental population at each generation. Then, the next parental population is obtained by selecting a new set of k individuals from the $n = k + m$ parent and offspring individuals available, often using the hypervolume indicator as the selection criterion (Knowles et al., 2003; Beume et al., 2007; Bader and Zitzler, 2011). Furthermore, since discarded solutions cannot be recovered unless they are generated again by the genetic operators, assessing the approximation quality of the intermediate and/or final populations with respect to the quality of the best subset that may be selected from all solutions evaluated up to the corresponding generation becomes of interest (Bringmann et al., 2011). The incremental greedy approximation to the HSSP allows lower bounds on the quality of such optimal subsets to be determined, and may also be used to select a (possibly) better set of solutions than the final population for further consideration by a Decision Maker.

In this paper, incremental greedy algorithms for the HSSP in 2 and 3 dimensions are proposed, providing a $(1-1/e)$-approximation to the optimal subset. Rather than simply iterating over existing algorithms to compute hypervolume contributions (or the hypervolume indicator) in order to determine the greatest hypervolume contributor at each step, the algorithms proposed here exploit the incremental nature of the greedy approach, and efficiently update only those contributions that are changed by the selection of a new point at each iteration. In particular, in 3 dimensions, partially overlapping regions are specifically considered, and computing the volume of the same common sub-region more than once is avoided. Careful analysis shows that both algorithms proposed have $O(n(k + \log n))$ time complexity, which improves upon the $O(n^2)$-time bound previously reported for 3 dimensions (Guerreiro et al., 2015), and reveals the suitability of that algorithm for post-processing analysis, where n may be very large, but k is typically constant and relatively small.

The next section introduces relevant definitions, and reviews an existing approach to the computation of 3-dimensional hypervolume contributions in linear time. In Section 3, the general strategy is outlined first, and the simpler 2-dimensional case is considered before the main algorithm proposed is described in detail and illustrative examples are given. Experimental results are presented in Section 4, and are followed by some concluding remarks.



(a) $H(\{p^1,\dots,p^4\})$ (b) $H(p^3,\{p^1,p^2,p^4\})$ (c) $H(p^2,p^3,\{p^1,p^4\})$ (d) delimiters of p^3

Figure 1: Two-dimensional examples: (a) hypervolume indicator (dark gray region), (b) hypervolume contribution (light gray region), (c) joint hypervolume contribution (mid gray region), and (d) delimiters of p^3 (p^2 and p^4), as well as dominated point $(p^3 \vee p^1)$, represented by a filled circle, and non-dominated points $(p^3 \vee p^2)$ and $(p^3 \vee p^4)$, elements of J, represented by hollow circles.

2 Preliminaries

The following definitions are useful in stating the problem of interest and explaining the algorithm proposed. Illustrative examples are given in Figures 1(a)–(d).

Definition 1 ***(Hypervolume Indicator)*** *Given a point set* $\mathrm{S} \subset \mathbb{R}^d$ *and a reference point* $r \in \mathbb{R}^d$*, the hypervolume indicator of* S *is the measure of the region weakly dominated by* S *and bounded above by* r*, i.e.*

$$H(\mathrm{S}) = \lambda(\{q \in \mathbb{R}^d \mid \exists p \in \mathrm{S} : p \leq q \wedge q \leq r\})$$

where λ *denotes the Lebesgue measure.*

A fixed reference point is assumed throughout the paper.

Definition 2 ***(Hypervolume Contribution)*** *Given a point* $p \in \mathbb{R}^d$*, a point set* $\mathrm{S} \subset \mathbb{R}^d$*, and a reference point* $r \in \mathbb{R}^d$*, the (hypervolume) contribution of* p *to* S *is:*

$$H(p, \mathrm{S}) = H(\mathrm{S} \cup \{p\}) - H(\mathrm{S})$$

In some cases, such as when determining the decrease in the contribution of a given point $p \in \mathbb{R}^d$ to a set $\mathrm{S} \subset \mathbb{R}^d$ due to the addition of another point $q \in \mathbb{R}^d$ to S, it is also useful to consider the contribution dominated simultaneously and exclusively by two points:

Definition 3 ***(Joint Hypervolume Contribution)*** *The joint hypervolume contribution of* $p, q \in \mathbb{R}^d$ *to* $\mathrm{S} \subset \mathbb{R}^d$ *is:*

$$H(p, q, \mathrm{S}) = H(\mathrm{S} \cup \{p \vee q\}) - H(\mathrm{S})$$

where $\vee$ *denotes the* join*, or component-wise maximum between two points.*

Moreover, the contribution of a point p to a set S is bounded above by certain points $q \in \mathrm{S}$ that shall be referred to as *delimiters*, and are defined as follows:

Definition 4 (Delimiter) *Given a point set* $\mathrm{S} \subset \mathbb{R}^d$ *and a point* $p \in \mathbb{R}^d$*, let* $\mathrm{J} = \mathrm{nondominated}(\{(p \vee q) \mid q \in \mathrm{S}\})$*. Then,* $q \in \mathrm{S}$ *is called a (weak) delimiter of the contribution of* p *to* S *iff* $(p \vee q) \in \mathrm{J}$*. If, in addition,* $H(p, q, \mathrm{S}\backslash\{q\}) > 0$*, then* q *is also a strong delimiter of the contribution of* p *to* S.

where $\mathrm{nondominated}(\mathrm{X}) = \{q \in \mathrm{X} \mid t \leq q \Rightarrow t = q, \forall t \in \mathrm{X}\}$ denotes the set of nondominated points in $\mathrm{X} \subset \mathbb{R}^d$. Note that J is the smallest set of points weakly dominated by p that delimits its contribution to S, that is, $H(p, \mathrm{S}) = H(p, \mathrm{J})$. Consequently, all $q \in \mathrm{J}$ are such that $H(p, q, \mathrm{J}\backslash\{q\}) > 0$, and J is such that $H(p, \mathrm{S}) = H(p, \{\}) - H(\mathrm{J})$. For example, in Figure 1(d), the contribution of p^3 is delimited only by p^2 and p^4, where both p^2 and p^4 are strong delimiters. Non-strong delimiters can only exist when S contains points with repeated coordinates. In the same example, if there were a fifth point, p^5, such that $p^5_x = p^4_x$ and $p^5_y < p^4_y$, then both p^4 and p^5 would be weak delimiters but not strong delimiters. This means that, in practice, only one in such a group of delimiters is needed to bound the contribution of p. If one of them is deleted, then the contribution of p is kept unchanged, whereas it increases if all are deleted. Finally, the HSSP problem (Bader and Zitzler, 2011) is formally defined here as:

Problem 1 ***(Hypervolume Subset Selection Problem (HSSP))*** *Given a point set* $\mathrm{S} \subset \mathbb{R}^d$ *and an integer* $k \in \{0, 1, \ldots, |\mathrm{S}|\}$*, find a subset* $\mathrm{A} \subseteq \mathrm{S}$ *such that* $|\mathrm{A}| \leq k$ *and:*

$$H(\mathrm{A}) = \max_{\substack{\mathrm{B} \subseteq \mathrm{S} \\ |\mathrm{B}| \leq k}} H(\mathrm{B})$$

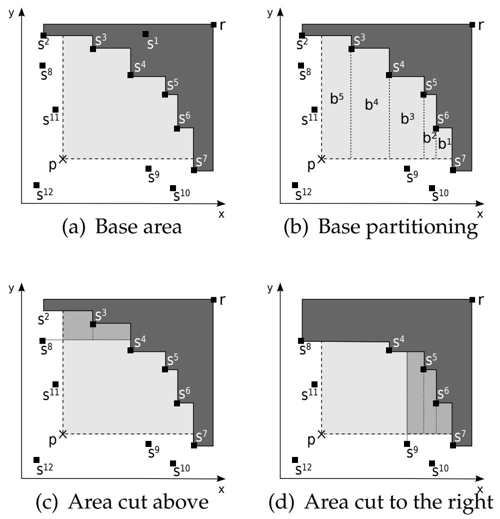
In the example of Figure 1(a), where $n = 4$, subset $\{p^1, p^3\}$ is the (single) optimal solution of the HSSP with $k = 2$. Any other subset of $\{p^1, \ldots, p^4\}$ with at most two points has a hypervolume indicator value lower than $H(\{p^1, p^3\}) = 39$.

2.1 Computing hypervolume contributions

The efficient computation of hypervolume contributions in 3 dimensions is an important aspect of the greedy algorithm for the HSSP developed in this work. In HV4D (Guerreiro et al., 2012), an algorithm for hypervolume computation in 4 dimensions, the 3-dimensional contribution of each new point visited in the main loop is computed in linear time. This is achieved using a dimension-sweep approach based on another algorithm by Emmerich and Fonseca (2011) for the problem of computing all contributions in 3 dimensions. A simplified version of that approach, which does not explicitly use a box-division of the contribution, is described next using Figure 2 for illustration. It will be referred to as IHV3D. Moreover, let p_x, p_y and p_z denote the x, y and z coordinate of a point $p \in \mathbb{R}^3$ in an (x, y, z)-space, respectively.

Given a point $p \in \mathbb{R}^3$ and a set $\mathrm{S} \subset \mathbb{R}^3$ of n points, the contribution $H(p, \mathrm{S})$ is computed in IHV3D by sweeping the points $q \in \mathrm{S}$ such that $q_z > p_z$ in ascending order of the z coordinate, and partitioning the 3-dimensional contribution in horizontal slices. The contribution of p is the sum of the volumes of all slices. The volume of a slice is the area of the base of that slice multiplied by its height. The height of a slice is the absolute difference between the two consecutive points defining that slice. The base is delimited by the projection onto the (x, y)-plane of the first point defining that slice and the points below it in z. Thus, S is split into two sets, $\mathrm{S}^1 = \{q \in \mathrm{S} \mid q_z \leq p_z\}$ and $\mathrm{S}^2 = \{q \in \mathrm{S} \mid q_z > p_z\}$. In addition, a set of points whose projections on the (x, y)-plane delimit the area exclusively dominated by p in each iteration, S', is maintained. This set of mutually non-dominated points is initialized with such points in S^1 to represent the base of the first slice.

Both the splitting of S and the initialization of S' are performed in linear time. In the example of Figure 2(a), $\mathrm{S}^1 = \{s^1, \ldots, s^7\}$, $\mathrm{S}^2 = \{s^8, \ldots, s^{12}\}$ and $\mathrm{S}' = \{s^2, \ldots, s^7\}$.


y
x
r
p
s^1 s^2 s^3 s^4 s^5 s^6 s^7 s^8 s^9 s^{10} s^{11} s^{12}
b^1 b^2 b^3 b^4 b^5

(a) Base area (b) Base partitioning

(c) Area cut above (d) Area cut to the right

Figure 2: Example of the computation of the contribution $H(p, \mathrm{S})$, where $\mathrm{S} = \{s^1, \ldots, s^{12}\}$. It is assumed that $s^1_z < \cdots < s^7_z < p_z < s^8_z < \cdots < s^{12}_z$.

Note that at most two points in S' are not dominated by p on the (x, y)-plane, one above and to the left, and another below and to the right (s^2 and s^7 in the example).

The area of the base of the first slice is computed by adding up the areas of the non-overlapping rectangles into which the base is partitioned (see Figure 2(b)) as the points in S' are visited in ascending order of y. Then, the points in S^2 are visited in ascending order of z. For each new point, the volume of the current slice is computed and the area of its base is updated to obtain the base area of the next slice. In the example, the first point visited is s^8. Therefore, the area of the base of the bottom slice is multiplied by $s^8_z - p_z$. Then, the base area is updated by subtracting the area of the region that is now dominated also by s^8 (see Figure 2(c)). This area is computed by visiting the points in S' that are dominated by s^8 on the (x, y)-plane. In the example, these are points s^2 and s^3, which are subsequently replaced in S' by s^8. Hence, S' becomes $\mathrm{S}' = \{s^8, s^4, \ldots, s^7\}$. The procedure for s^9 is similar (see Figure 2(d)). Visited points that do not dominate part of the region dominated by p are skipped (e.g. s^{10}).

The algorithm continues until a point in S^2 that dominates p on the (x, y)-plane is found, s^{12} in the example. The volume of the last slice is computed by multiplying the current base area by $(s^{12}_z - s^{11}_z)$. In IHV3D, all sets are implemented as sorted lists, and sentinels are used to ensure that limiting points such as s^2, s^7 and s^{12} always exist. IHV3D has an amortized $O(n)$ time complexity because each point in S is visited once when it is added to S' and a second time when it is removed from S', and all operations on S' are performed in constant time.

3 Greedy HSS Algorithm

In this section, a general greedy algorithm (Bradstreet et al., 2007) for the HSSP is explained. Subsequently, specialized algorithms for 2 and 3 dimensions are proposed.

Algorithm 1 gHSS(X, k, r)

```
Require: $\mathrm{X} \subset \mathbb{R}^d$, $k \in \mathbb{N}^+$, $r \in \mathbb{R}^d$
 1: $\mathrm{S} \leftarrow \{\}$
 2: for all $q \in \mathrm{X}$ do
 3:     $q.c \leftarrow H(q, \mathrm{S})$
 4: for $i = 1..k$ do
 5:     $p \leftarrow \arg\max_{q \in \mathrm{X}}\{q.c\}$
 6:     $\mathrm{X} \leftarrow \mathrm{X} \backslash \{p\}$
 7:     for all $q \in \mathrm{X}$ do
 8:         $q.c \leftarrow q.c - H(p, q, \mathrm{S})$
 9:     $\mathrm{S} \leftarrow \mathrm{S} \cup \{p\}$
10: return $\mathrm{S}$
```

3.1 General Case

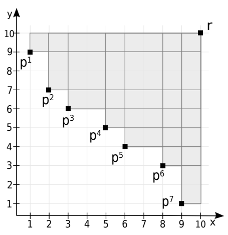
Given a non-dominated point set $\mathrm{X} \subset \mathbb{R}^d$, where $|\mathrm{X}| = n$, in the greedy algorithm for the HSSP (Problem 1), $k \leq n$ points from X are chosen and stored in S, one at a time, always selecting the point that contributes the most hypervolume to the set of points already chosen. For that reason, in gHSS (Algorithm 1), the contribution of each point $q \in \mathrm{X}$ to the initially empty set S (line 3) is computed first and is stored in $q.c$. Afterwards, the point p in X with maximal contribution is picked (line 5). Then, the contribution of the remaining points in X to $\mathrm{S} \cup \{p\}$ is updated (line 8), i.e., the portion of the contribution of each $q \in \mathrm{X}$ that is dominated by p is removed. Finally, p is moved from X to S (lines 6 and 9). Lines 5 to 9 are repeated until S contains k points.

Note that ties may occur, i.e., at some point, more than one point may have the (same) highest contribution. In such cases, it is correct to choose any of the tied points. However, ties that are solved differently, will possibly result in different subsequent intermediate and final greedy solutions, both with respect to the set of points selected and, thus, to the corresponding hypervolume indicator value.

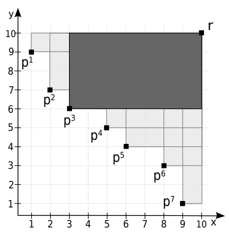
It is clear from Definition 3 that, in the absence of an efficient algorithm to compute the Joint Hypervolume Contribution (line 8), an algorithm for the Hypervolume Indicator can be used to determine this quantity. Furthermore, the whole contribution $H(q, \mathrm{S} \cup \{p\})$ could be simply recomputed in line 8. Updating the contributions of all points p in a set X to the corresponding sets $\mathrm{X} \backslash \{p\}$ under single-point changes to X can already be performed efficiently in 2 dimensions (Hupkens and Emmerich, 2013). However, in the incremental greedy algorithm, the contributions of points in X have to be updated w.r.t. a set S and not to X itself. Thus, the algorithm proposed by Hupkens and Emmerich (2013) cannot be used directly by the greedy algorithm. The same is true for algorithms to compute the contributions of all points p in X to $\mathrm{X} \backslash \{p\}$ (Emmerich and Fonseca, 2011; Bringmann and Friedrich, 2010). In contrast, the procedure explained in Subsection 2.1 to compute a single 3-dimensional contribution in linear time could be adapted to this case, resulting in a $O(k^2 n)$-time algorithm.

3.1.1 Example

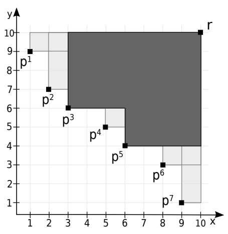
Figure 3 shows an example of how the greedy algorithm can be applied to a set of non-dominated points $\mathrm{X} = \{p^1, \ldots, p^7\}$ in two-dimensional space for k up to 7. Figure 3



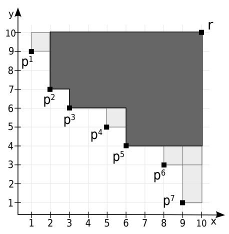
(a) $\mathrm{S} = \{\}$
$\mathrm{X} = \{p^1, \ldots, p^7\}$



(b) $\mathrm{S} = \{p^3\}$
$\mathrm{X} = \{p^1, p^2, p^4, \ldots, p^7\}$



(c) $\mathrm{S} = \{p^3, p^5\}$
$\mathrm{X} = \{p^1, p^2, p^4, p^6, p^7\}$



(d) $\mathrm{S} = \{p^3, p^5, p^2\}$
$\mathrm{X} = \{p^1, p^4, p^6, p^7\}$

		|S|						
	$H(p, \mathrm{S})$	0	1	2	3	4	5	6
p	p^1	9	2	2	1	**1**	-	-
	p^2	24	3	**3**	-	-	-	-
	p^3	**28**	-	-	-	-	-	-
	p^4	25	5	1	1	1	**1**	-
	p^5	24	**8**	-	-	-	-	-
	p^6	14	6	2	2	1	1	**1**
	p^7	9	5	3	**3**	-	-	-

(e) Contributions with respect to S

Figure 3: Example of the incremental greedy algorithm (gHSS2D) for the HSSP in 2 dimensions.

illustrates which points are selected by the greedy algorithm and in which order. Figures 3(a), 3(b), 3(c) and 3(d) show the evolution of sets X and S in the first 4 iterations, and depict the corresponding contributions of the points in X to S. The table in Figure 3(e) shows the values of these contributions for every iteration of the algorithm. Each column corresponds to an iteration of the algorithm, and is labeled with the size of the set of selected points, |S|. Each row shows the contribution of a point in X as set S grows. Consequently, column i shows the contribution to S of each point not yet selected at iteration i. The values in bold indicate which point from X is selected in each iteration. Note that column i shows both the points in the greedy solution for $k = i$ (the dashed cells) and an intermediate solution for $k > i$. Thus, any particular case where $k < n$ leads to performing just the first k steps of the example.

The first step of the algorithm (line 3 in Algorithm 1) is to compute the contribution of every point $p \in \mathrm{X}$ to $\mathrm{S} = \{\}$, which corresponds to the first column in the table. Then, in iteration 1 of the for loop in line 4, since p^3 contributes the most to $\mathrm{S} = \{\}$, it is selected and moved from X to S (lines 6 and 9). The contributions of the points remaining in X are updated (line 8) to account for the addition of p^3 to S. For example, the contribution of p^1 is updated by subtracting the area of its contribution that becomes dominated by p^3 (the area between $(3, 9)$ and r). Formally, $H(p^1, \{p^3\}) = H(p^1, \{\}) - H(p^1, p^3, \{\}) = 9 - 7 = 2$. The same calculations are performed for $p^2, p^4, \ldots, p^7$. After the contribution update step, the contributions of every point still in X to $\mathrm{S} = \{p^3\}$ are known as shown in the second column of the table. In iteration 2, p^5 is the point that contributes the most to $\mathrm{S} = \{p^3\}$, and so it is selected and moved from X to S. Note that, in such a case, the contributions of p^1 and p^2 remain the same with the addition of p^5 to S, and so only those of p^4, p^6 and p^7 have to be updated. The algorithm repeats the select and update steps until k points are selected. In Figure 3, the greedy solution for $k \leq n$ is formed by the first k points of the sequence: $p^3, p^5, p^2, p^7, p^1, p^4, p^6$. For a given $k < n$, the greedy solution contains the points that, in the column k, correspond to dashed cells. For example, for $k = 3$, the greedy solution can be seen in the fourth column, which corresponds to $|\mathrm{S}| = 3$, and is $\mathrm{S} = \{p^2, p^3, p^5\}$.

The following subsections show how $H(p, q, \mathrm{S})$ can be efficiently computed in the 2- and 3-dimensional cases.

3.2 2-dimensional Case

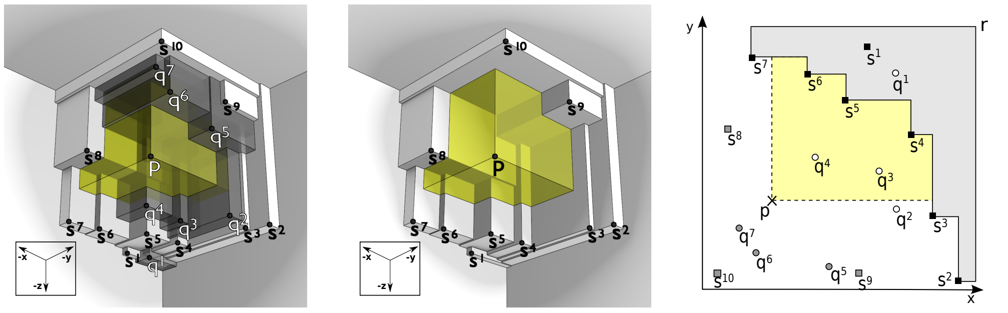
The algorithm proposed in this subsection (gHSS2D) deals with the particular case of gHSS in 2 dimensions. In gHSS2D, a simple procedure can be used to update the contributions of the unchosen points in line 8 of Algorithm 1. Assume that points in $\mathrm{X} \cup \mathrm{S}$ are kept sorted in ascending order of the y-coordinate (and in descending order of the x-coordinate). Therefore, every point $q \in \mathrm{X} \cup \mathrm{S}$ needs to keep the information of the next and of the previous point in $\mathrm{X} \cup \mathrm{S}$ which is determined once by sorting X in a pre-processing step.

Because a contribution is represented by a rectangle, every $q \in \mathrm{X}$ stores its contribution ($q.c$) and also the corresponding upper bound. All upper bounds are initially set to the reference point. Let t^1 and t^2 be the closest points in S to the left and to the right of p, respectively. Then, for each point p chosen, the points to its left are visited, in ascending order of coordinate y, until t^1 is reached. Similarly, the points to its right are visited until t^2 is reached. Every point $q \in \mathrm{X}$ between t^1 and p has the upper bound previously set to (t^2_x, t^1_y) and therefore, the quantity $H(p, q, \mathrm{S})$ is $(t^2_x - p_x) \times (t^1_y - q_y)$, and the upper bound is set to (p_x, t^1_y). Points $q \in \mathrm{X}$ between p and t^2 have $H(p, q, \mathrm{S}) = (t^2_x - q_x) \times (t^1_y - p_y)$ and their upper bound is set to (t^2_x, p_y).

The time complexity of gHSS2D is $\Theta(n(k + \log n))$ because the initial sorting costs $\Theta(n \log n)$-time and, for each point p chosen, up to n contributions have to be computed, where each contribution is computed in constant time. A worst case example is the set of n points $\{(-i, -2^{n-i+1} + 1) \mid i \in \{1, \ldots, n\}\}$ with $r = (0, 0)$. Note that gHSS2D and the exact algorithms (Bringmann et al., 2014; Kuhn et al., 2016) have similar time complexities. However, the greedy version should be easier to implement and, because it is very simple and uses simple data structures, it is very fast in practice.

3.3 3-dimensional Case

The algorithm that deals with the particular case of gHSS in 3 dimensions (gHSS3D) is presented in this subsection. This subsection starts by defining the data structures,



(a) 3D example (b) $H(\mathrm{S})$ and $H(p, \mathrm{S})$ (c) 2D projection with $H(\mathrm{S}^*)$ and $H(p^*, \mathrm{S}^*)$ at $z = p_z$

Figure 4: Example where $\mathrm{X} = \{q^1, \ldots, q^7\}$, $\mathrm{S} = \{s^1, \ldots, s^{10}\}$ and p is the last point removed from X and to be added to S. (a) shows X, S and p, (b) shows how much volume p will add to S (transparent region). (c) shows a cut at $z = p_z$ and the 2-dimensional projection of all points in (a). In (c), points in X are represented with circles, and points in S are represented with squares. The squares and circles painted gray have higher z coordinate than p and the remaining ones have lower z coordinate than p. As can be seen in (a), when $\mathrm{X} \cup \mathrm{S} \cup \{p\}$ is sorted in ascending order of the z coordinate, the following sequence is obtained: $q^1, s^1, \ldots, s^7, q^2, \ldots, q^4, p, s^8, q^5, q^6, s^9, q^7, s^{10}$.

some notation and procedures used by the algorithm. Then, the algorithm itself and, in particular, the update of the contribution of the points in X in line 8 of Algorithm 1 are detailed. This subsection finishes with a discussion of the time complexity of gHSS3D.

The main aspect of gHSS3D is how the contributions of points in X are updated. For an easier understanding of how those are performed in gHSS3D, the problem depicted in Figure 4 will be used as an example. The Figure shows an intermediate iteration of Algorithm 1, where some points have already been moved from X to S. Note that Figure 4 does not intend to illustrate an actual choice of points by gHSS3D, and that the update procedure presented here is independent of the choice of points moved from X to S. The only assumptions about X and S are that they are disjoint sets and $\mathrm{X} \cup \mathrm{S}$ is a non-dominated point set. In Figure 4(a), the contribution to S of every point in X is shown. The transparent (yellow) volume in Figure 4(b) shows more clearly the volume that p adds to S. Any point in X whose contribution to S lies partially in that yellow region has to have that volume removed from its contribution, as it becomes dominated. 2D projections as the one in Figure 4(c) will be used further in this paper.

3.3.1 Data structures and procedures

In gHSS3D, doubly linked lists are used to maintain the sets of points sorted, and sentinels ensure that there is always a point in the limiting conditions. Algorithm 2 keeps both sets X and S sorted in ascending order of all coordinates. Each point $q \in \mathrm{X}$ keeps some information associated to it, such as area ($q.a$), volume ($q.v$), height ($q.z$) and contribution ($q.c$). The first three values are temporary values that are used to compute the volume $H(p, q, \mathrm{S})$ which will be subtracted from $q.c$, the contribution of q. The value $q.z$ indicates the value of the third coordinate up to which volume $q.v$ has been updated, and $q.a$ keeps the area dominated at height $q.z$. p^* and S^* will denote the projections of p and S onto the (x, y)-plane, respectively.

Algorithm 2 gHSS3D(X, k, r)

```
Require: $\mathrm{X} \subset \mathbb{R}^d$, $k \in \mathbb{N}^+$, $r \in \mathbb{R}^d$
1: X is sorted in *ascending* order of all dimensions
2: $\mathrm{S} \leftarrow \{(r_x, -\infty, -\infty), (-\infty, r_y, -\infty), (-\infty, -\infty, r_z)\}$
3: for all $q \in \mathrm{X}$ do
4:     $q.c \leftarrow (r_x - p_x) \times (r_y - p_y) \times (r_z - p_z)$
5: for $i = 1..k$ do
6:     initialize $q.v$ to 0 for all $q \in \mathrm{X}$
7:     $p \leftarrow \arg\max_{q \in \mathrm{X}} \{q.c\}$
8:     $\mathrm{X} \leftarrow \mathrm{X} \backslash \{p\}$
9:     $\mathrm{X}' \leftarrow \mathrm{X}$
10:    for all $o \in \{(x, y, z), (z, x, y), (y, z, x)\}$ do
11:        $(x, y, z) \leftarrow o$ // Change coordinate order
12:        $\mathrm{X}^1 \leftarrow \{q \in \mathrm{X}' \mid p^* \leq q^* \wedge p_z \geq q_z\}$
13:        $\mathrm{X}^2 \leftarrow \{q \in \mathrm{X}' \mid p^* \geq q^* \wedge p_z \leq q_z\}$
14:        $\mathrm{S}^1, \mathrm{S}^2 \leftarrow$ split$_z(\mathrm{S}, p_z)$
15:        jointContributions1$(p, \mathrm{S}^1, \mathrm{S}^2, \mathrm{X}^1, r)$
16:        jointContributions2$(p, \mathrm{S}^1, \mathrm{S}^2, \mathrm{X}^2, r)$
17:        $\mathrm{X}' \leftarrow \mathrm{X}' \backslash (\mathrm{X}^1 \cup \mathrm{X}^2)$
18:    for all $q \in \mathrm{X}$ do
19:        $q.c \leftarrow q.c - q.v$
20:    $\mathrm{S} \leftarrow \mathrm{S} \cup \{p\}$
21: return S
```

Given $\mathrm{X}, \mathrm{S} \subset \mathbb{R}^3$ represented by sorted lists and the points $p, q \in \mathbb{R}^3$ and $h \in \mathbb{R}$, the following procedures are available.

next$_y(p, \mathrm{S})$ The point following p in S with respect to coordinate y, for $p \in \mathrm{S}$.

head$_y(\mathrm{S})$ The point $q \in \mathrm{S}$ with the least q_y.

min$_y(p, \mathrm{S})$ The point $q \in \mathrm{S}$ with the least $q_y > p_y$ such that $q_x \leq p_x$.

addFirst$_y(p, \mathrm{S})$ Add point p to S where p becomes head$_y(\mathrm{S})$.

minima(S) Return the points that are not dominated on the (x, y)-plane , i.e., $\{q \in \mathrm{S} \mid \nexists t \in \mathrm{S} : t^* \leq q^*, t \neq q\}$.

split$_y(\mathrm{S}, h)$ Given $h \in \mathbb{R}$, split S in two sets, S^1 and S^2 such that $\mathrm{S}^1 = \{q \in \mathrm{S} \mid q_y \leq h\}$ and $\mathrm{S}^2 = \{q \in \mathrm{S} \mid q_y > h\}$ and return S^1 and S^2.

area(p, S) The quantity $H(p^*, \mathrm{S}^*)$.

updateVolume(X, h) Given $h \in \mathbb{R}$, for each point $q \in \mathrm{X}$, update its volume, save it in $q.v$ and update $q.z$, i.e., compute $q.v \leftarrow q.v + q.a \times (h - q_z)$ and set $q.z \leftarrow h$.

initializeBases$(\mathrm{X}, p, \mathrm{S})$ For each point $q \in \mathrm{X}$, compute $H(p^*, q^*, \mathrm{S}^*)$ and save it in $q.a$.

updateAreas$(p, \mathrm{X}, \mathrm{S})$ For each $q \in \mathrm{X}$ compute $q.a \leftarrow q.a - H(p^*, q^*, \mathrm{S}^*)$.

Procedures next, head, min, addFirst and split are also available for coordinates x and z and, apart from split and min, they all run in constant time. split_y has a cost of $O(|\mathrm{S}^2|)$ because it is the cost of finding the break point by sweeping points in descending order of coordinate y. The remaining procedures have linear cost w.r.t. the total size of the input sets, i.e., either $O(|\mathrm{X}|)$, $O(|\mathrm{S}|)$ or $O(|\mathrm{S}|+|\mathrm{X}|)$. Both initializeBases and updateAreas will be explained in more detail in Subsection 3.3.3. All procedures that modify or return a subset of a given sorted set guarantee that the returned sets are also sorted according to the coordinate used for sweeping the points. These procedures may also guarantee that those points are sorted according to other coordinates, if needed. More information about those procedures will be given in the next subsections. Note that, if a set of non-dominated points on the (x, y)-plane is sorted in ascending order of one coordinate, then it is also sorted according to the other coordinate, but in descending order.

3.3.2 Main loop of gHSS3D

gHSS3D follows the same working principle as gHSS, but, instead of updating the contributions of points in X one by one (lines 7 and 8 in Algorithm 1), the (x, y, z)-space is divided into 8 octants with a common vertex at p, and the contributions of the points in each pair of opposite octants are updated at the same time (lines 9 to 17 in Algorithm 2).

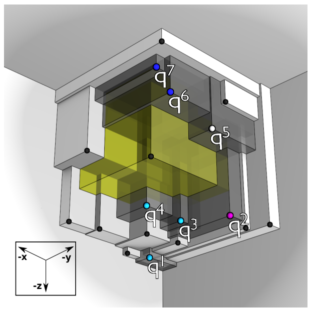
The two octants corresponding to the region that dominates p and the region dominated by p are ignored because they do not contain any points. The remaining three pairs of octants are all updated in the same way, except that a different coordinate order is considered for each pair. The order is set in such a way that a different dimension is used as the z-coordinate in each case (lines 10-17 of Algorithm 2).

Given a coordinate order $o \in \{(x, y, z), (z, x, y), (y, z, x)\}$, the two octants considered when order o is selected are those that contain sets $\mathrm{X}^1 \subseteq \mathrm{X}$ and $\mathrm{X}^2 \subseteq \mathrm{X}$, defined as follows. X^1 contains the points in X that are dominated by p in the first two dimensions of o, but are equal to or better (i.e., lower) than p in the third dimension. X^2 contains the points in X that dominate p in the first two dimensions of o but are equal to or worse (i.e., higher) than p in the third dimension. Figure 5 shows how the set X from the example in Figure 4 is split into octants, and shows X^1 and X^2 according to the objective order considered. In Figure 5(a), points are assigned colors, each associated with a single octant. Figures 5(b), 5(c) and 5(d) show the corresponding sets X^1 and X^2 according to objective order (x, y, z), (y, z, x) and (z, x, y), respectively. Note that considering a different objective order is equivalent to rotating the space.

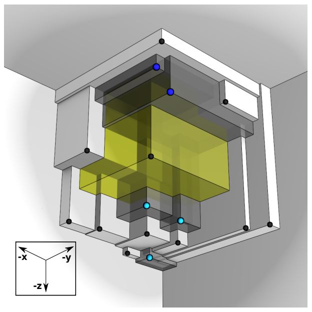
Lines 9 and 17 of Algorithm 2 guarantee that no point in X is updated more than once in case there are points with repeated coordinates, i.e., points on the boundary between two octants. The computation of $q.v = H(p, q, \mathrm{S})$ for $q \in \mathrm{X}^1$ and $q \in \mathrm{X}^2$ is detailed in Algorithms 4 and 3, respectively.

3.3.3 Updating contributions of points in X

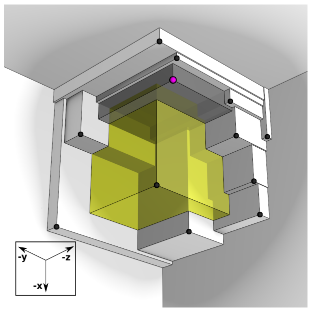
Consider the case where the order considered is $o = (x, y, z)$. In the example given, $\mathrm{X}^1 = \{q^1, q^3, q^4\}$ and $\mathrm{X}^2 = \{q^6, q^7\}$, as depicted in Figure 5(b). The update procedure for the remaining coordinate orders is similar (Figures 5(c) and 5(d)). Figure 6 shows (in red) the joint contributions of p with each point in X^1 (Figures 6(a) to 6(c)) and with each point in X^2 (Figures 6(d) and 6(e)), which have to be computed and removed.



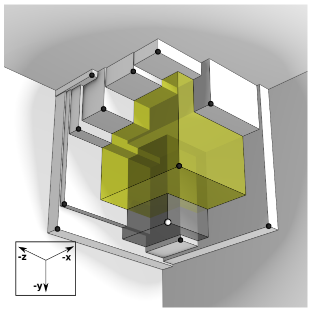
(a) Base example



(b) $o = (x, y, z)$,
$\mathrm{X}^1 = \{q^1, q^3, q^4\}$,
$\mathrm{X}^2 = \{q^6, q^7\}$



(c) $o = (y, z, x)$, $\mathrm{X}^1 = \{\}$,
$\mathrm{X}^2 = \{q^2\}$



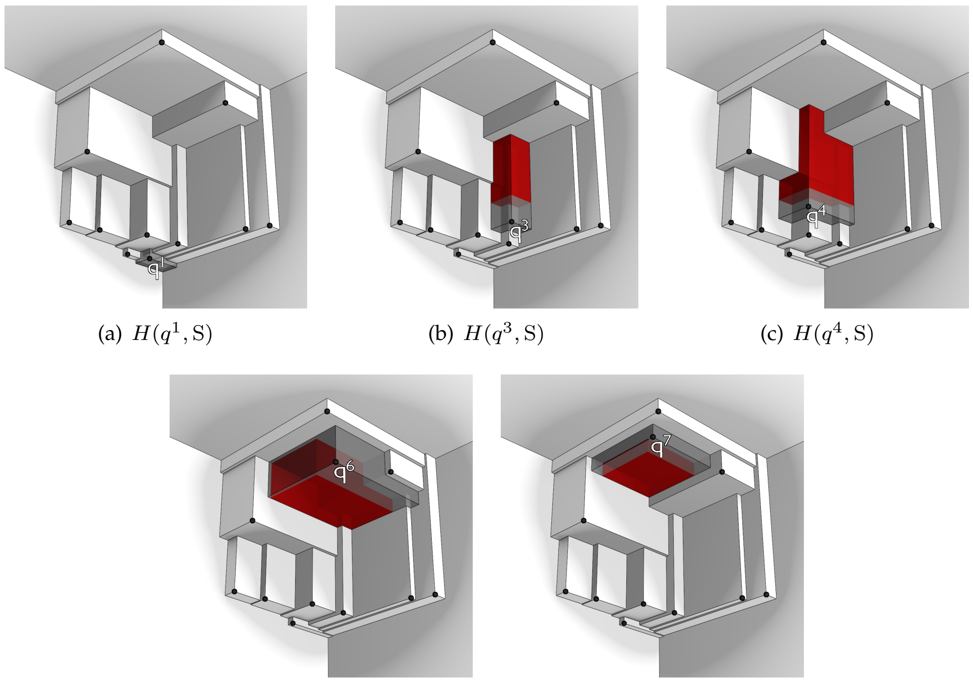
(d) $o = (z, x, y)$, $\mathrm{X}^1 = \{q^5\}$,
$\mathrm{X}^2 = \{\}$

Figure 5: The subproblems X^1 and X^2 for the three objective orders considered, $o = \{(x, y, z), (y, z, x), (z, x, y)\}$.

Note that, in the case of q^1, there is no joint contribution with p. Furthermore, note that some of the joint contributions (partially) overlap, for example, those of p with q^3 and of p with q^4.

The update procedure will be explained first for X^2 and then for X^1. In both cases, the algorithms are built upon IHV3D (explained in Section 2). Therefore, Algorithm 2 splits S into S^1 and S^2. In this case only, function split takes $O(n)$-time as it has to guarantee that S^1 and S^2 are sorted according to all dimensions. Set S' is also maintained along the execution of both Algorithms 4 and 3, and is initialized as the subset of S^1 that delimits the area dominated by p. In the example, we have that $\mathrm{S}^1 = \{s^1, \ldots, s^7\}$, $\mathrm{S}^2 = \{s^8, \ldots, s^{10}\}$ and $\mathrm{S}' = \{s^3, \ldots, s^7\}$ (initial set).

In the case of X^2 ($\mathrm{X}^2 = \{q^6, q^7\}$), all points dominate p on the (x, y)-plane, and have higher z-coordinate. Therefore, if X^2 is sorted in ascending order of z, then, given any $q \in \mathrm{X}^2$ and its previous point u in X^2, the joint contribution of q with p is equal to the joint contribution of u with p above the value q_z of coordinate z. Moreover, the contribution of q when it is $\mathrm{head}_z(\mathrm{X}^2)$ is equal to the contribution of p above the value q_z of coordinate z. Figure 7(a) shows the joint contributions of p with each point in $\mathrm{X}^2 = \{q^6, q^7\}$. Note that the joint contribution of q^7 with p is equal to the joint contribution of q^6 with p for $z \geq q^7_z$. The joint contribution of q^6 with p is equal to the contribution of p for $z \geq q^6_z$. Thus, if the volume between $z = u_z$ and $z = q_z$



(a) $H(q^1, \mathrm{S})$ (b) $H(q^3, \mathrm{S})$ (c) $H(q^4, \mathrm{S})$

(d) $H(q^6, \mathrm{S})$ (e) $H(q^7, \mathrm{S})$

Figure 6: Individual contributions to S (dark gray plus red region) of the points in $\mathrm{X}^1 = \{q^1, q^3, q^4\}$ and in $\mathrm{X}^2 = \{q^6, q^7\}$. The region of each contribution that is dominated by p is in red.

is associated only to u, then the joint contribution of every $q \in \mathrm{X}^2$ can be calculated by accumulating the volume associated to each point while visiting X^2 in descending order of z. Hence, in Algorithm 3, the previous point $u \in \mathrm{X}^2$ of each $q \in \mathrm{X}^2$ is stored in $q.prev$.

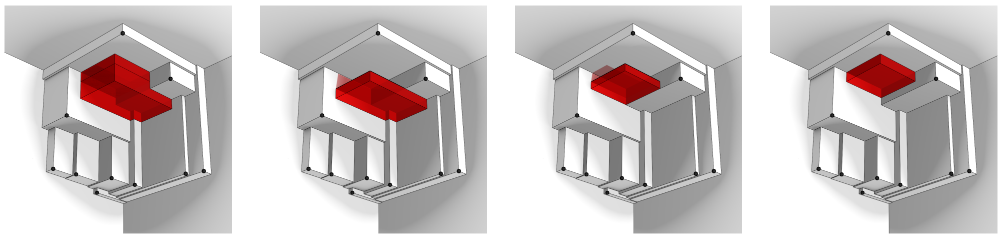
Algorithm 3 behaves just like IHV3D, i.e., in the algorithm, S' is used to maintain the points delimiting the area of p at height $z = q_z$, and points in S^2 are visited in ascending order of z in order to update the area of p and to compute the slice volume (lines 8 to 15). Additionally, X^2 is swept along with S^2 in ascending order of the z-coordinate by merging the two lists (line 4). The last point visited from X^2 is recorded as u (except for first initialization of u, which is a sentinel), and its volume (line 16) and area (line 17) are updated when the area of p ($p.a$) is updated. Whenever the current point $q \in \mathrm{X}^2 \cup \mathrm{S}^2$ belongs to X^2, the volume simultaneously dominated by u and p bounded below by u_z becomes bounded above by q_z (line 19) and stays associated to (and only to) u. The contribution of p above q_z becomes associated to q (lines 20 and 21) and q is set as the new u (line 23). In that way, each part of the contribution p (a consecutive set of slices) is associated to a single point of X^2. When the first point from S^2 that dominates p on the (x, y)-plane is found (s^{10}), the volume of the last point from X^2 is updated (line 25). Figures 7(b) and 7(c) show the two slices (whose volume is) associated to q^6, and Figure 7(d) shows the only slice associated to q^7. Lastly, $q \in \mathrm{X}^2$ are visited again but in descending order of the z-coordinate, and their volumes are

Algorithm 3 jointContributions2$(p, \mathrm{S}^1, \mathrm{S}^2, \mathrm{X}^2, r)$

```
1:  $\mathrm{S}' \leftarrow \{q \in \mathsf{minima}(\mathrm{S}^1) \mid p^* \leq q^*\} \cup \{\min_x(\mathrm{S}^1), \min_y(\mathrm{S}^1)\}$
2:  $p.a \leftarrow \mathsf{area}(p, \mathrm{S}')$
3:  $u \leftarrow p$
4:  $\mathrm{L} \leftarrow \mathrm{X}^2 \cup \mathrm{S}^2$
5:  $q \leftarrow \mathsf{head}_z(\mathrm{L})$
6:  **while** $q \in \mathrm{X}^2$ **or** $q^* \nleq p^*$ **do**
7:      **if** $q \in \mathrm{S}^2$ **then**
8:          **if** $q_x \leq p_x$ **then**
9:              $\mathrm{S}', \mathrm{T} \leftarrow \mathsf{split}_y(\mathrm{S}', q_y)$
10:             $p.a \leftarrow p.a - \mathsf{area}(p \vee q, \mathrm{T} \cup \{\mathsf{head}_x(\mathrm{S}')\})$
11:             $\mathsf{addFirst}_x(q, \mathrm{S}')$
12:         **else**
13:             $\mathrm{S}', \mathrm{T} \leftarrow \mathsf{split}_x(\mathrm{S}', q_x)$
14:             $p.a \leftarrow p.a - \mathsf{area}(p \vee q, \mathrm{T} \cup \{\mathsf{head}_y(\mathrm{S}')\})$
15:             $\mathsf{addFirst}_y(q, \mathrm{S}')$
16:         $\mathsf{updateVolume}(\{u\}, q_z)$
17:         $u.a \leftarrow p.a$
18:     **else**
19:         $\mathsf{updateVolume}(\{u\}, q_z)$
20:         $q.a \leftarrow p.a$
21:         $q.z \leftarrow q_z$
22:         $q.prev \leftarrow u$
23:         $u \leftarrow q$
24:     $q \leftarrow \mathsf{next}_z(q, \mathrm{L})$
25: $\mathsf{updateVolume}(\{u\}, q_z)$
26: $vol \leftarrow 0$
27: **while** $u \neq p$ **do**
28:     $vol \leftarrow vol + u.v$
29:     $u.v \leftarrow vol$
30:     $u \leftarrow u.prev$
```

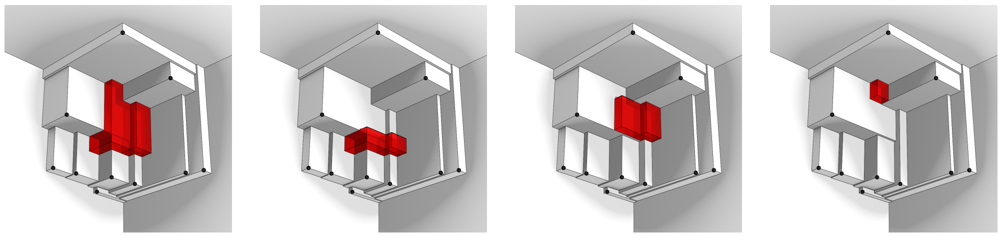
accumulated and added to the next visited points (lines 27 to 30). In the example, the volume associated to q^6 is stored in vol, and corresponds to its joint contribution with p. Then the volume associated to q^7 is added to vol, corresponding now to the joint contribution of q^7 and p.

In the case of points $q \in \mathrm{X}^1$, not every point has its contribution reduced with the addition of p to S, as it is the case of q^1 in Figure 6(a). In Algorithm 4, a set $\mathrm{X}' \subseteq \mathrm{X}^1$ is used to keep the points of X^1 whose joint contributions with p are still being computed. X' is initialized in linear time with the points in X^1 that need to be updated, i.e., those points such that $q^* \in \mathrm{X}^{1*}$ is not dominated by S'^*. Figure 8(a) shows the joint contributions of p with every point in the initial set $\mathrm{X}' = \{q^3, q^4\}$, which partially overlap. Figures 8(b) to 8(d) show how these joint contributions are split into slices by the algorithm. The computation of $H(p, q, \mathrm{S})$ for each $q \in \mathrm{X}'$ is performed in two main steps, the computation of the base area of the volume $H(q \vee p, \mathrm{S}')$ (line 3) and the area updates (lines 13 and 21). Figures 9 and 10 will be used as examples of these steps, respectively. Note that the base areas of the volumes $H(q \vee p, \mathrm{S}')$ correspond to the



(a) $H(p, q^6, \mathrm{S})$ and $H(p, q^7, \mathrm{S})$ (b) Slice 1 (c) Slice 2 (d) Slice 3

Figure 7: The (partially overlapping) joint contributions of every point in $\mathrm{X}^2 = \{q^6, q^7\}$ with p and their division into slices.



(a) $H(p, q^3, \mathrm{S})$ and $H(p, q^4, \mathrm{S})$ (b) Slice 1 (c) Slice 2 (d) Slice 3

Figure 8: The (partially overlapping) joint contributions of every point in $\{q^3, q^4\} \subset \mathrm{X}^1$ with p and their division into slices.

base areas in the red volume in Figure 8(b), while the base areas of the red volumes of Figures 8(c) and 8(b) correspond to the subsequently updated areas.

Figure 9(a) corresponds to Figure 8(b) in the cut plane at $z = p_z$ and shows, also in red, the base areas of q^3 and q^4 to be computed, which overlap partially. The base areas are initialized in linear time by computing $H(p^*, \mathrm{S}'^*) - H(p^*, \mathrm{S}'^* \cup \{q^*\})$, for each $q \in \mathrm{X}'$. The areas associated to points in X' are initialized with $H(p^*, \mathrm{S}'^*)$, which was previously computed in linear time in line 2 of Algorithm 3. The value $H(p^*, \mathrm{S}'^* \cup \{q^*\})$ is computed in two sweeps. Points $q \in \mathrm{S}' \cup \mathrm{X}'$ are first visited in ascending order of x and the area dominated by p between p_x and each q_x is computed in a cumulative way. When a point from X' is visited, the area accumulated so far is subtracted from the area associated with that point. In Figures 9(b) and 9(c), the striped area is the accumulated area that is subtracted from $q^4.a$ and $q^3.a$, respectively. The remaining area to be subtracted, i.e., the area between p_y and q_y to the right of p_x, is computed in an analogous way by sweeping points in $\mathrm{S}' \cup \mathrm{X}'$ in ascending order of the y-coordinate. Note that, in the second sweep, it is also necessary to add to $q.a$ the area bounded below by p and above by q, for each $q \in \mathrm{X}'$, as it was subtracted twice.

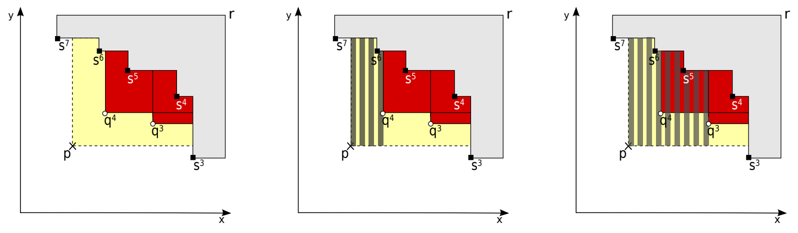
Figure 10 shows the update of the areas of points in X' (lines 13 and 21). The volumes are updated in lines 12 and 20. This corresponds to the cases where a point in S' cuts above the area dominated by p (lines 7 to 14 and s^8 in Figures 10(a) and 10(b)) or to the right (line 16 to 22 and s^9 in Figure 10(c) and 10(d)). Points $q \in \mathrm{S}^2$ are visited in ascending order of z-coordinate, as in IHV3D. Looking at the case where $q = s^8$,

Algorithm 4 jointContributions1$(p, \mathrm{S}^1, \mathrm{S}^2, \mathrm{X}^1, r)$

```
1: $\mathrm{S}' \leftarrow \{q \in \mathsf{minima}(\mathrm{S}^1) \mid p^* \leq q^*\} \cup \{\min_x(\mathrm{S}^1), \min_y(\mathrm{S}^1)\}$
2: $\mathrm{X}' \leftarrow \{q \in \mathrm{X}^1 \mid p^* \leq q^*$ **and** $\nexists t \in \mathrm{S}' : t^* \leq q^*\}$
3: initializeBases$(\mathrm{X}', p, \mathrm{S}')$
4: $\mathrm{L} \leftarrow \mathrm{S}^2$
5: $q \leftarrow$ head$_z(\mathrm{L})$
6: **while** $q^* \nleq p^*$ **do**
7:     **if** $q_x \leq p_x$ **then**
8:         $\mathrm{S}', \mathrm{T} \leftarrow$ split$_y(\mathrm{S}', q_y)$
9:         $t \leftarrow$ head$_x(\mathrm{S}')$
10:        $\mathrm{X}', \mathrm{A} \leftarrow$ split$_y(\mathrm{X}', q_y)$
11:        $\mathrm{B} \leftarrow \{u \in \mathrm{X}' \mid u_x < t_x\}$
12:        updateVolume$(\mathrm{A} \cup \mathrm{B}, q_z)$
13:        updateAreas$_y(p \vee q, \mathrm{B}, \mathrm{T} \cup \{$head$_x(\mathrm{S}')\})$
14:        addFirst$_x(q, \mathrm{S}')$
15:    **else**
16:        $\mathrm{S}', \mathrm{T} \leftarrow$ split$_x(\mathrm{S}', q_x)$
17:        $t \leftarrow$ head$_y(\mathrm{S}')$
18:        $\mathrm{X}', \mathrm{A} \leftarrow$ split$_x(\mathrm{X}', q_x)$
19:        $\mathrm{B} \leftarrow \{u \in \mathrm{X}' \mid u_y < t_y\}$
20:        updateVolume$(\mathrm{A} \cup \mathrm{B}, q_z)$
21:        updateAreas$_y(p \vee q, \mathrm{B}, \mathrm{T} \cup \{$head$_y(\mathrm{S}')\})$
22:        addFirst$_y(q, \mathrm{S}')$
23:    $q \leftarrow$ next$_z(q, \mathrm{L})$
```

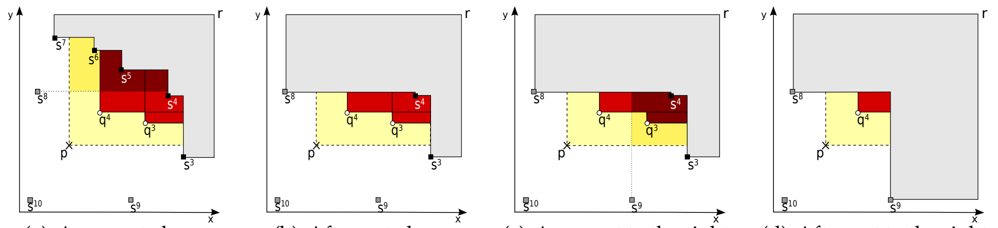
let T be the set of points in S' with higher y-coordinate than s^8_y ($\mathrm{T} = \{s^5, \ldots, s^7\}$). In Algorithm 4, set T is first removed from S' ($\mathrm{S}' = \{s^3, \ldots, s^7\}$), in $O(|\mathrm{T}|)$-time, through function split. Thus, S' becomes $\mathrm{S}' = \{s^3, s^4\}$). T is kept sorted according to the y-coordinate. Similarly, set A is removed from X' ($\mathrm{X}' = \{q^3, q^4\}$) in $O(|\mathrm{A}|)$-time, also through split (line 10). Set A contains the points that are dominated by q^* on the (x, y)-plane, i.e., those that have no more contribution above q_z ($\mathrm{A} = \{\}$). Finally, the areas associated with the points in X' to the left of the point that delimits q at right ($t = s^4$) have to be updated. Therefore, those points are stored in B (line 11, $\mathrm{B} = \{q^3, q^4\}$). The volumes of points in $\mathrm{B} \cup \mathrm{A}$ are updated in $O(|\mathrm{A}| + |\mathrm{B}|)$-time (line 12). Then, in procedure updateAreas (line 13), points $u \in \mathrm{B} \cup \mathrm{T}$ are visited in descending order of coordinate x. For each u visited, the area dominated by $p \vee q$ to the right of u is accumulated. Whenever the visited point u belongs to B, the area computed so far is subtracted from $u.a$. Therefore, the update of the areas of points in B costs $O(|\mathrm{B}| + |\mathrm{T}|)$. At last, q is added to S' ($\mathrm{S}' = \{s^8, s^4, s^3\}$). The updated areas (and the updated X' and S') are depicted in Figure 9(b), which corresponds to Figure 8(c) at the cut plane $z = s^8_z$.

When $q_y \leq p_y$, the procedure is analogous. Note that, in the example of Figure 10(c), $q = s^9$ and $\mathrm{A} = \{q^3\}$. Therefore, q^3 is removed from X' and its volume is updated. X' becomes $\{q^4\}$, then $t = s^8$, $\mathrm{B} = \{q^4\}$, and, at the end of that iteration, $\mathrm{S}' = \{s^8, s^9\}$ and $\mathrm{X}' = \{q^4\}$. The updated areas (and the updated X' and S') are depicted in Figure 10(d), which corresponds to Figure 8(d) at the cut plane $z = s^9_z$. Algorithm 4 terminates when the first point q that dominates p on the (x, y)-plane is found (s^{10}).



(a) Areas of q^3 and q^4 (b) Update $q^4.a$ (c) Update $q^3.a$

Figure 9: Example of how the base area of each points in $\mathrm{X}' = \{q^3, q^4\}$ is computed in initializeBases$(\mathrm{X}', p, \mathrm{S}')$.



(a) Areas cut above (b) After cut above (c) Areas cut to the right (d) After cut to the right

Figure 10: Example of how points from S^2 are used in Algorithm 4 to update the areas of points in X'.

3.3.4 Time complexity

In Algorithm 2 (gHSS3D), points are sorted in $O(n \log n)$-time and their contributions are initialized in linear time. Then, the contributions of points in X are updated k times by sweeping those points considering three different coordinate orders. For each coordinate order, the points of two disjoint sets are updated, X^1 and X^2 in Algorithms 4 and 3, respectively. Lines 11 to 14 and 17 have linear cost, while Algorithm 3 (line 16) has amortized linear time. Algorithm 4 (line 15) has amortized $O(nk)$-time, but the amortization is along the entire execution of Algorithm 2.

Algorithm 3 inherits the $O(n)$ time complexity of IHV3D. As in IHV3D, the base area associated with p is computed in linear time (line 2). Afterwards, points in S are only visited once when they are added to S' (line 1, 11 or 15), and once again if they are removed from S' (lines 9 and 13) and are used to update the area of p (lines 10 and 14). Points in X^2 are used in constant time operations in the first while loop (lines 16 to 25), and are all visited once again, to update their volumes (lines 27 to 30).

The time complexity of Algorithm 4 is analyzed by considering all calls (a maximum of $k \leq n$) from Algorithm 2. The analysis shows that some steps amortize to $O(n)$-time for each call of Algorithm 4, while others are worse than linear in a single call, but result in $O(nk)$-time across the k possible calls. In the worst scenario of the latter case (a single call), each point in the final set S (the greedy solution) may be compared against all other points a constant number of times.

In Algorithm 4, it is guaranteed that each point is visited only if its area or volume has to be updated or if it delimits the contribution of the area being computed.

The operations performed to maintain S' (lines 8, 16, 14 and 22) are the same as in Algorithm 3, so these operations amortize to linear time. As with S', points from X^1 are added once to X' (line 2) and are removed once from X' (line 10 or 18). Therefore, maintaining X' also amortizes to linear time. It remains to explain how many times points in $\mathrm{B} \subset \mathrm{X}'$ are visited. Constructing B has a cost $O(|\mathrm{B}|)$, and procedure updateAreas has a cost of $O(|\mathrm{B}| + |\mathrm{T}|)$. Since the points in T were previously removed from S', they are used by updateAreas only once. However, in the worst case, B is equal to X' in every iteration of Algorithm 4, and points are never removed from X'. This means that procedure updateAreas has a cost of $O(n)$, leading to $O(kn)$-time complexity over $|\mathrm{S}^2| \leq k$ iterations.

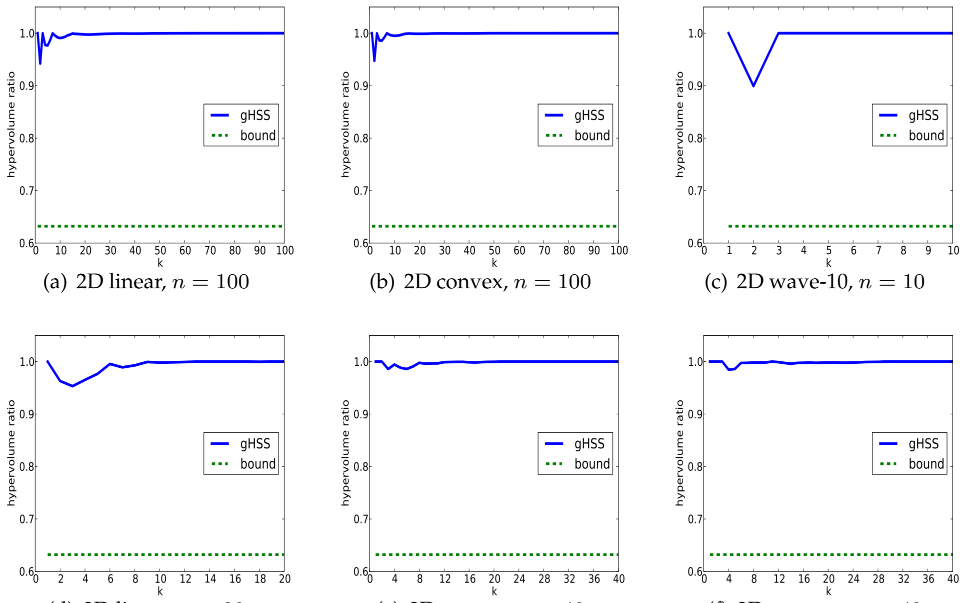
Basically, the role of updateAreas is to compare each point in X with points in S. However, in this procedure, each point in X is compared against each point in S once throughout the execution of gHSS3D. As $|\mathrm{S}| \leq k$ always holds, all executions of updateAreas, and of updateVolume, amortize to $O(nk)$. This can be checked by noting that a point $u \in \mathrm{X}^1 \subseteq \mathrm{X}$ is only compared to a point $s \in \mathrm{S}^2 \subset \mathrm{S}$ if s delimits the contribution of u. These points are compared only because the region where s delimits the contribution of u is dominated by p. Therefore, after p is added to S, the point s will not delimit the contribution of u anymore. This is because $p \leq (u \vee s)$ and $p_z < s_z$, which means that u has no contribution to $\mathrm{S} \cup \{p\}$ above the value p_z of coordinate z ($H(u, s, (\mathrm{S} \cup \{p\})\backslash\{s\}) = 0$). Moreover, note that $(p \vee u) < (s \vee u)$, and, thus, according to Definition 4 i.e., the definition of delimiter, s is not a delimiter of the contribution of u to $\mathrm{S} \cup \{p\}$. Consequently, u will not be compared again with s. In the example of Figure 10, neither q^3 nor q^4 will be compared against s^8 or s^9 in any future execution of procedure updateAreas. In the examples of Figures 6(b) and 6(c) it can be seen that, after removing the part of the point contribution in red, the remaining volume, in dark gray, is delimited by neither s^8 nor s^9. Therefore, Algorithm 2 has amortized $O(nk)$-time complexity.

Although gHSS3D has $O(n(k + \log n))$-time complexity, if it is used repeatedly for a fixed k and a growing set of n solutions by iteratively adding new small sets of $m \ll n$ solutions, then these sets may be previously sorted in $O(m \log m)$, and inserted in the data structures (linked lists) in $O(n + m)$ time. Therefore, the set of k solutions selected can be updated in $O((n + m)k + m \log m)$ time in such a scenario.

4 Results

The proposed algorithms, gHSS2D and gHSS3D, were implemented in C and compiled with gcc version 4.7.2. All tests were run on an Intel(R) Core(TM) i7-3612QM CPU @ 2.10GHz with 6 MB of cache and 8 GB of RAM.

The first results (see Figure 11) concern approximation quality. The results produced by gHSS2D were compared against those obtained with the exact algorithm by Bringmann and Friedrich (2010), HypSSP. Regarding gHSS3D, optimal results were obtained by adapting the HSSP Integer Linear Programming (ILP) formulation by Kuhn et al. (2016) to the 3-dimensional case, and solving it with the GNU Linear Programming Toolkit. Figure 11 shows the ratio between the hypervolume indicator values of the approximate and exact solutions for various sets, for growing values of k. Spherical concave fronts consisted of points randomly located on the positive quadrant/octant of a unit circle/sphere with center at the origin of the coordinate axes.



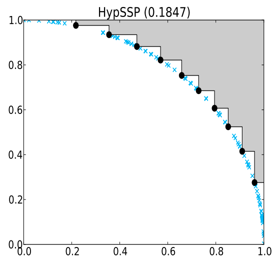
(a) 2D linear, $n = 100$ (b) 2D convex, $n = 100$ (c) 2D wave-10, $n = 10$

(d) 3D linear, $n = 20$ (e) 3D convex, $n = 40$ (f) 3D concave, $n = 40$

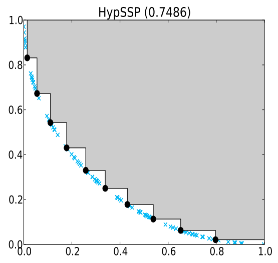
Figure 11: The ratio between the hypervolume indicator of the approximate and the exact solutions for a fixed set size n and variable subset size k.

Spherical convex fronts consisted of points randomly located on the negative quadrant/octant of a unit circle/sphere, but with center at $(1, 1)$ or $(1, 1, 1)$, as appropriate. Linear fronts were such that $0 \leq x_i \leq 1$ for $i = 1, 2$ and $i = 1, 2, 3$, respectively, and $\sum_i x_i = 1$ in both cases. The reference point was set to $(1, 1)$ and $(1, 1, 1)$ for 2 and 3 dimensions, respectively. Moreover, in two-dimensions, a "wave-w" data set with both concave and convex regions was considered, where $w \in [1, \dots, 5, 10]$ is the number of convex regions. This data set represents a front defined by a cosine function over the diagonal between $(0, 1)$ and $(1, 0)$ on the positive quadrant and with magnitude of $0.2/w$.

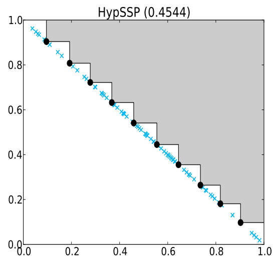
The hypervolume ratio between the greedy and the optimal solution was always found to be greater than 0.89 (observed for the 2-dimensional "wave-10" data set in Figure 11(c)), which is much better than the theoretical bound of $(1 - 1/e) \simeq 0.63212$. Moreover, the approximation ratio was generally lower for smaller values of k, with $k = 2$ being the subset size that most often led to the lowest approximation ratio. For $k \geq n/2$, the ratio was found to be either 1 (an optimum was found) or very close to 1. Figures 12 and 13 show the solutions selected by the greedy and the exact algorithms for different fronts, for the 2- and the 3-dimensional case, respectively. One can observe that, as expected, the greedy solutions are not optimal in general, but they are still quite similar to the optimal ones, and the corresponding hypervolume indicator values are very close to optimal. It is worth noting that, in the cases depicted in Figures 12 and 13, the corresponding hypervolume ratios are indeed much better than the theoretical bound. For example, for the concave case in Figures 12(a) and 12(d), the ratio between the greedy solution (by gHSSD2D) and the optimal solution (by HypSSP) is $\frac{0.1827}{0.1847} = 0.9892$. Similarly, for the 3-dimensional convex case in Figures 13(e) and 13(h),



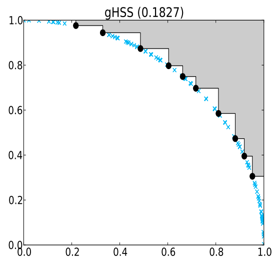
(a) concave



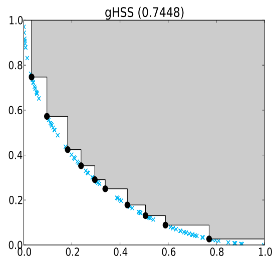
(b) convex



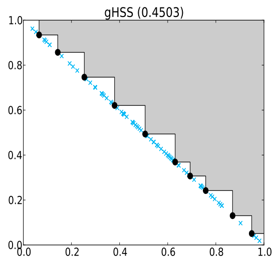
(c) linear



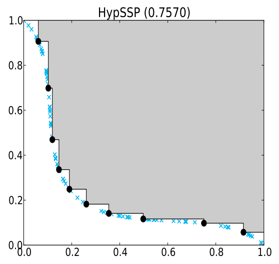
(d) concave



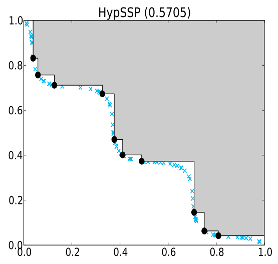
(e) convex



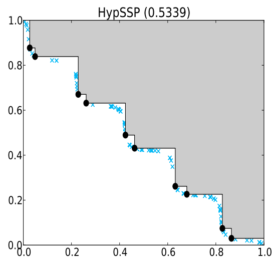
(f) linear



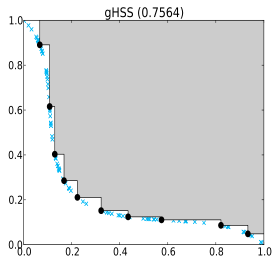
(g) wave-1



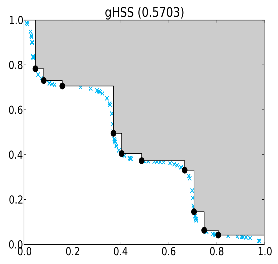
(h) wave-3



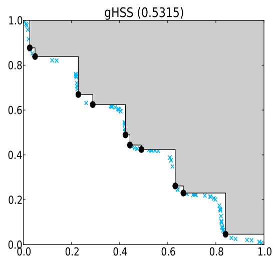
(i) wave-5



(j) wave-1

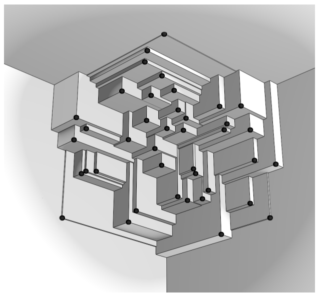


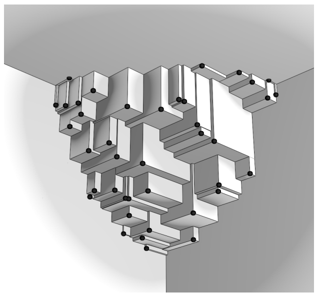
(k) wave-3

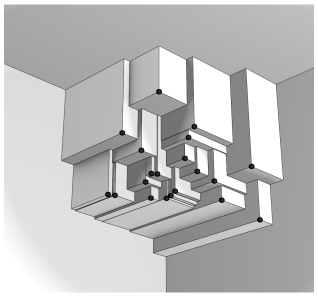


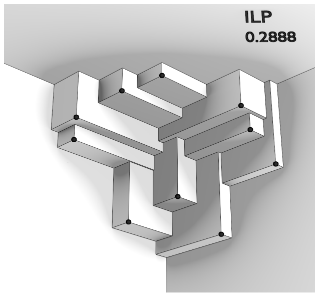
(l) wave-5

Figure 12: Greedy and optimal subsets produced by gHSS2D and HypSSP, respectively, for 2-dimensional concave, convex and linear fronts (from left to right, the top two rows), and for mixed convex/concave fronts (bottom two rows). The selected solutions are represented as black dots, and the region dominated by them is shaded gray. The corresponding area is given in the title of each plot. The remaining solutions (blue crosses, $\times$) are discarded. The data corresponds to cases where $n = 100$ and $k = 10$.

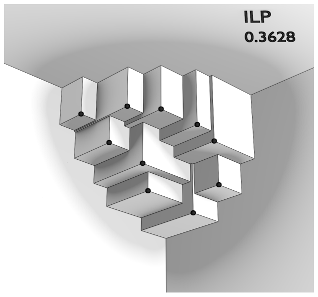

(a) concave, n=40


(b) convex, n=40

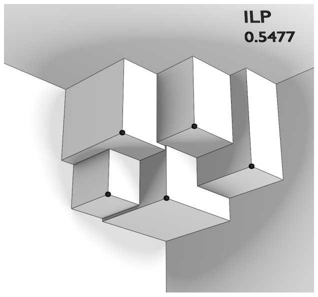

(c) linear, n=20



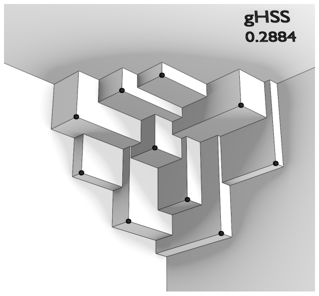
(d) concave, n=40, k=10



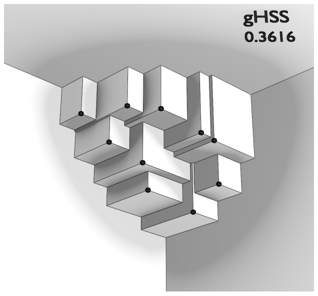
(e) convex, n=40, k=10



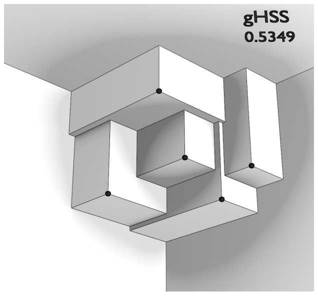
(f) linear, n=20, k=5



(g) concave, n=40, k=10

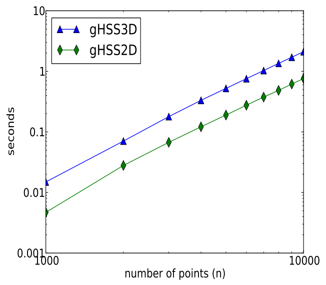


(h) convex, n=40, k=10

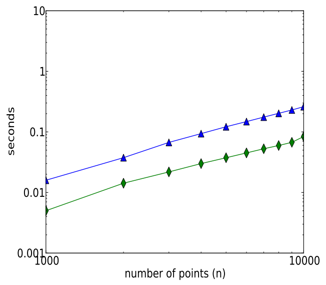


(i) linear, n=20, k=5

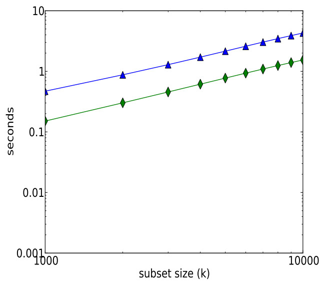
Figure 13: The optimal and the greedy subsets produced by ILP and gHSS3D, respectively, for 3-dimensional concave, convex and linear fronts (from left to right). The top row shows the n solutions given (black dots) and the corresponding dominated region. The middle and bottom rows show the subsets of solutions selected by ILP and gHSS, respectively. The corresponding hypervolume indicator values are displayed in the upper-right corners.



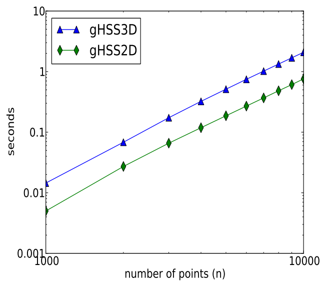
(a) linear, $k = n/2$



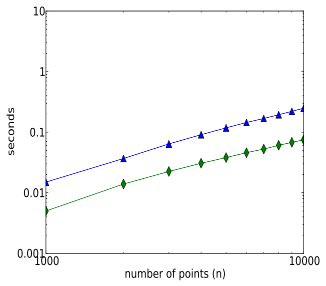
(b) linear, $k = 500$



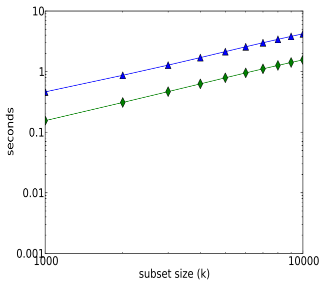
(c) linear, $n = 10^4$



(d) convex, $k = n/2$



(e) convex, $k = 500$



(f) convex, $n = 10^4$

Figure 14: Runtime of gHSS2D and gHSS3D for the linear and the convex data sets for: varying n and k (first column); varying n and a fixed k (second column); and a fixed n and varying k (third column).

the ratio between the greedy solution (by gHSSD3D) and the optimal solution (by ILP) is $\frac{0.3616}{0.3628} = 0.9967$.

The second set of experiments concerned runtime. Figure 14 shows the behavior of gHSS2D and gHSS3D for different settings of n and k. The average runtimes of 13 runs on the linear and convex fronts with the characteristics described earlier are depicted (concave fronts produced very similar results, and thus the corresponding plots are not shown). Figure 14 clearly shows the quadratic growth of the runtime of both algorithms for growing set size n and $k = n/2$. Still, selecting $k = 5000$ out of $n = 10000$ points was performed in less than 1 second in 2 dimensions, and in about 2 seconds in the 3-dimensional case.

The second column of plots (Figures 14(b) and 14(e)) shows the runtime for growing set size n and fixed $k = 500$. Although runtimes appear to grow slightly faster than linearly for small n, for $n > 3000$, the runtime growth of both gHSS2D and gHSS3D is essentially linear. The same can be observed for growing subset size k and fixed $n = 10000$, in the third column of plots (Figures 14(c) and 14(f)). The observed runtimes are in agreement with their time complexity of $O(n(k + \log n))$.

5 Concluding Remarks

In this paper, incremental greedy algorithms for the HSSP in two and three dimensions were proposed, providing a $(1 - 1/e)$-approximation to the optimal solution in each case. The two-dimensional version, gHSS2D, has a worst-case time complexity of $\Theta(n(k+\log n))$, which is similar to the complexity of exact algorithms for the same problem (Bringmann et al., 2014; Kuhn et al., 2016). However, it does have the advantage of being very simple to implement and very fast in practice. The $O(n(k + \log n))$ time complexity of the three-dimensional version, on the other hand, is considerably lower than that of the corresponding exact algorithms, which are $O(n^{\frac{3}{2}} \log n + n^{n-k})$ (Bringmann and Friedrich, 2010) and $\Omega(n^3)$ due to the $\Omega(n^3)$ constraints in the ILP formulation (Kuhn et al., 2016). It is also better than other incremental greedy approaches based on iterating over existing algorithms to compute hypervolume indicator/contributions. For example, only $O(k^2 n)$ would be achieved by adapting HV4D (Guerreiro et al., 2015). In practice, gHSS3D was at most 3 times slower than gHSS2D on the data sets used for testing.

Regarding the quality of the approximation, the results obtained experimentally were much better than the guaranteed approximation factor, staying within at least 0.89 of the optimal values, in comparison to the theoretical $(1 - 1/e) \simeq 0.63212$. Therefore, gHSS3D should provide an interesting alternative to much more computationally expensive exact algorithms for hypervolume-based multiobjective selection and/or archiving in the three-objective case.

Finally, the proposed algorithms are suited for post-processing evaluation of EMOAs (fixed k and variable n scenario), providing bounds on the hypervolume of the best subset of k solutions among all solutions generated by the EMOA up to given points in its execution. In this case, such bounds could be updated in $O((n + m)k + m \log m)$ when m new solutions are added to an existing set of n previously processed solutions.

Acknowledgments

This work was supported by national funds through the Portuguese Foundation for Science and Technology (FCT), by the European Regional Development Fund (FEDER) through COMPETE 2020 – Operational Program for Competitiveness and Internationalization (POCI), and by the iCIS project (CENTRO-07-ST24-FEDER-002003), which was co-financed by QREN in the scope of the Mais Centro Program and FEDER. A. P. Guerreiro acknowledges FCT for Ph.D. studentship SFHR/BD/77725/2011, co-funded by the European Social Fund and by the State Budget of the Portuguese Ministry of Education and Science in the scope of NSRF–HPOP–Type 4.1–Advanced Training.

References

Auger, A., Bader, J., Brockhoff, D., and Zitzler, E. (2009). Theory of the hypervolume indicator: Optimal μ-distributions and the choice of the reference point. In *Foundations of Genetic Algorithms (FOGA 2009)*, pages 87–102, New York, NY, USA. ACM.

Bader, J. and Zitzler, E. (2011). HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76.

Beume, N., Fonseca, C., López-Ibáñez, M., Paquete, L., and Vahrenhold, J. (2009). On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13(5):1075–1082.

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

Bradstreet, L., While, L., and Barone, L. (2007). Incrementally maximising hypervolume for selection in multi-objective evolutionary algorithms. In *2007 IEEE Congress on Evolutionary Computation*, pages 3203–3210.

Bringmann, K. and Friedrich, T. (2010). An efficient algorithm for computing hypervolume contributions. *Evolutionary Computation*, 18(3):383–402.

Bringmann, K., Friedrich, T., and Klitzke, P. (2014). Two-dimensional subset selection for hypervolume and epsilon-indicator. In *Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 589–596, New York, NY, USA. ACM.

Bringmann, K., Friedrich, T., Neumann, F., and Wagner, M. (2011). Approximation-guided evolutionary multi-objective optimization. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1198–1203. AAAI Press.

Chan, T. M. (2013). Klee's measure problem made easy. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 410–419, Los Alamitos, CA, USA. IEEE Computer Society.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA.

Ehrgott, M. (2005). *Multicriteria Optimization*. Springer, second edition.

Emmerich, M. T. M. and Fonseca, C. M. (2011). Computing hypervolume contributions in low dimensions: Asymptotically optimal algorithm and complexity results. In Takahashi, R. H. C., Deb, K., Wanner, E. F., and Greco, S., editors, *EMO 2011*, volume 6576 of *LNCS*, pages 121–135. Springer, Berlin, Heidelberg.

Friedrich, T. and Neumann, F. (2014). Maximizing submodular functions under matroid constraints by multi-objective evolutionary algorithms. In *Parallel Problem Solving from Nature (PPSN) XIII*, volume 8672 of *LNCS*, pages 922–931. Springer International Publishing.

Guerreiro, A. P., Fonseca, C. M., and Emmerich, M. T. M. (2012). A fast dimension-sweep algorithm for the hypervolume indicator in four dimensions. In *Canadian Conference on Computational Geometry (CCCG) 2012*, pages 77–82.

Guerreiro, A. P., Fonseca, C. M., and Paquete, L. (2015). Greedy hypervolume subset selection in the three-objective case. In *Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 671–678. ACM.

Hupkens, I. and Emmerich, M. (2013). Logarithmic-time updates in SMS-EMOA and hypervolume-based archiving. In Emmerich, M., Deutz, A., Schuetze, O., Bäck, T., Tantar, E., Tantar, A.-A., Moral, D. P., Legrand, P., Bouvry, P., and Coello, A. C., editors, *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary*

Computation IV, volume 227 of *Advances in Intelligent Systems and Computing*, pages 155–169. Springer, Heidelberg".

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

Kuhn, T., Fonseca, C. M., Paquete, L., Ruzika, S., Duarte, M. M., and Figueira, J. R. (2016). Hypervolume subset selection in two dimensions: Formulations and algorithms. *Evolutionary Computation*, 24(3):411–425.

Nemhauser, G., Wolsey, L., and Fisher, M. (1978). An analysis of approximations for maximizing submodular set functions–I. *Mathematical Programming*, 14(1):265–294.

Ulrich, T. and Thiele, L. (2012). Bounding the effectiveness of hypervolume-based $(\mu + \lambda)$-archiving algorithms. In *Learning and Intelligent Optimization*, volume 7219 of *LNCS*, pages 235–249. Springer.

While, L., Bradstreet, L., and Barone, L. (2012). A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation*, 16(1):86–95.

Zitzler, E. and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms – A comparative case study. In *Parallel Problem Solving from Nature (PPSN) V*, volume 1498 of *LNCS*, pages 292–301. Springer.