On the Computation of the Empirical Attainment Function

Carlos M. Fonseca[1,2,3], Andreia P. Guerreiro[4],
Manuel López-Ibáñez[5], and Luís Paquete[6]

[1] Department of Informatics Engineering, University of Coimbra,
Pólo II, 3030-290 Coimbra, Portugal
cmfonsec@dei.uc.pt
[2] CEG-IST, Instituto Superior Técnico, Technical University of Lisbon
Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal
[3] DEEI, Faculty of Science and Technology, University of Algarve
Campus de Gambelas, 8005-139 FARO, Portugal
[4] Instituto Superior Técnico, Technical University of Lisbon
Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal
andreia.guerreiro@ist.utl.pt
[5] IRIDIA, Université Libre de Bruxelles (ULB)
Av. F. Roosevelt 50, CP 194/6, 1050 Brussels, Belgium
manuel.lopez-ibanez@ulb.ac.be
[6] CISUC, Department of Informatics Engineering, University of Coimbra,
Pólo II, 3030-290 Coimbra, Portugal
paquete@dei.uc.pt

Abstract. The attainment function provides a description of the location of the distribution of a random non-dominated point set. This function can be estimated from experimental data via its empirical counterpart, the empirical attainment function (EAF). However, computation of the EAF in more than two dimensions is a non-trivial task. In this article, the problem of computing the empirical attainment function is formalised, and upper and lower bounds on the corresponding number of output points are presented. In addition, efficient algorithms for the two and three-dimensional cases are proposed, and their time complexities are related to lower bounds derived for each case.

Keywords: Empirical attainment function, algorithms, multiobjective optimiser performance, estimation.

1 Introduction

The development of new stochastic optimisers, and comparison thereof, depends on the ability to assess their performance in some way. However, assessing the performance of stochastic *multiobjective* optimisers, such as multiobjective evolutionary algorithms, is considered difficult for two main reasons. On the one hand, theoretical convergence results are often unavailable for such optimisers, or are too weak to be indicative of their performance in practice. On the other

R.H.C. Takahashi et al. (Eds.): EMO 2011, LNCS 6576, pp. 106–120, 2011.

hand, the analysis of experimental data is more challenging than in the single-objective case, due to the set nature of multiobjective optimisation outcomes.

Initial ideas on the performance assessment of stochastic multiobjective optimisers were put forward early in Evolutionary Multiobjective Optimisation history, based on the notion of attainment surfaces [2]. Those ideas were subsequently formalised in terms of the so-called *attainment function*, and links to existing results in random set theory were established [5]. As a mean-like, first-order moment measure of the statistical distribution of multiobjective optimisation outcomes, the attainment function provides a description of their *location* in objective space. More importantly, the attainment function may be estimated from experimental data using its empirical counterpart, the *empirical attainment function* (EAF). Thus, the performance of a stochastic multiobjective optimiser on a given optimisation problem, understood in terms of location of the corresponding outcome distribution, may be assessed by observing the outcomes of several independent optimisation runs and computing the corresponding EAF. Empirical comparisons may then be performed either visually or by formulating and testing appropriate statistical hypotheses.

Despite initial interest in the attainment function, much greater attention has been devoted in the literature to performance indicators, especially the hypervolume indicator [10], for which a body of theoretical and experimental results is currently available. In the meantime, the development of the attainment function approach has advanced slowly, and has focused mainly on theoretical aspects [4]. Due to computational difficulties, the EAF has been used mostly for visualisation purposes, in two dimensions [8].

In this article, the computation of the empirical attainment function is considered. The empirical attainment function and related concepts are introduced in Section 2, and the EAF computation problem is formalised in Section 3. In Section 4, bounds on the number of points to be computed are derived for two, three, and more dimensions. Efficient algorithms to compute the EAF in two and three dimensions are presented in Section 5, together with their computational complexities and some lower bounds on the complexity of the problem. The paper concludes with a discussion of the main contributions presented and directions for further work.

2 Background

When applied to an optimisation problem involving $d \geq 2$ objectives, stochastic multiobjective optimisers such as multiobjective evolutionary algorithms produce sets of solutions whose images in the d-dimensional objective space approximate the Pareto-optimal front of the problem in some sense. The quality of this approximation is usually considered to depend solely on the images in objective space, so that the outcome of an optimisation run may be seen as a set of points in $\mathbb{R}^d$. Typically, this is a subset of the *non-dominated*, or *minimal*, objective vectors evaluated during the run, since only such points effectively contribute to the quality of the approximation.

Definition 1 (Minima). *Given a set of points* $\mathrm{X} = \{x_1, \ldots, x_m \in \mathbb{R}^d\}$*, the set of minima of* X *under the component-wise order is the set*

$$\min \mathrm{X} = \{x \in \mathrm{X} : \forall y \in \mathrm{X}, y \leq x \Rightarrow y = x\} \tag{1}$$

Definition 2 (Non-dominated point set). *A set of point* X *such that* $\min \mathrm{X} = \mathrm{X}$ *is called a set of minima, or a non-dominated point set.*

In practice, the actual outcome sets produced for the same problem vary from optimisation run to optimisation run, due to the stochastic nature of the optimisers, and may be seen as *realisations* of a random non-dominated point set, or RNP set [3]. Optimiser performance may then be studied through the distribution of such a random set. In particular, the *attainment function* provides information about this distribution with respect to *location* [5,4], and is defined as the probability of an outcome set X attaining an arbitrary point $z \in \mathbb{R}^d$, i.e., the probability of $\exists x \in \mathrm{X} : x \leq z$. The symbol $\trianglelefteq$ is used to denote attainment of a point by a set, e.g., $\mathrm{X} \trianglelefteq z$.

The attainment function may be estimated from experimental data through its empirical version:

Definition 3 (Empirical attainment function). *Let* $\mathbf{I}\{\cdot\} \colon \mathbb{R}^d \longmapsto \{0, 1\}$ *denote the indicator function, and let* $\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_n$ *be non-dominated point set realisations drawn independently from some RNP set distribution. The empirical attainment function (EAF) is the discrete function* $\alpha_n \colon \mathbb{R}^d \longmapsto [0, 1]$*, where*

$$\alpha_n(z) = \alpha_n(\mathrm{X}_1, \ldots, \mathrm{X}_n; z) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{I}\{\mathrm{X}_i \trianglelefteq z\} \tag{2}$$

This definition clearly describes how to evaluate the EAF at a given point $z \in \mathbb{R}^d$, but, in practice, it is also necessary to decide at which points the EAF should be evaluated. For visualisation purposes [8], for example, the boundaries of the regions of the objective space where the EAF takes a constant value are usually of interest. These boundaries were originally referred to as *%-attainment surfaces* [2], and may be understood as the family of tightest goal vectors individually attained by a given percentage of the optimisation runs considered. They have also been described as summary attainment surfaces [6]. Visualisation of the EAF, and of differences between EAFs associated to different optimisers, is often used as an exploratory data analysis tool to obtain additional insight into optimiser performance [6,8].

To produce a graphical representation of the EAF, the desired summary attainment surfaces must be computed in some way. The above description suggests that a summary attainment surface may be represented by a set of minima, and this set can be easily computed in two dimensions as discussed later in Section 5. Yet, no formal definition of the corresponding computation problem, or algorithm to solve it in more than two dimensions, have been presented to date. To overcome this difficulty, Knowles [6] proposed a plotting method based on the computation of the intersections between a number of axis-aligned sampling lines and the summary attainment surface of interest. These intersections

provide a grid-like sampling of the surface which is fast to compute, and leads to plots that are easy to interpret, at least in three dimensions. However, this sampling provides only an approximate description of the desired attainment surface while potentially involving a much larger number of points than the exact description adopted in this work.

3 The EAF Computation Problem

Summary attainment surfaces, as described in the previous section, may be understood as the lower boundary of the corresponding EAF superlevel sets. Formally, denote the t/n-superlevel set of $\alpha_n(z)$, $t = 1, \ldots, n$, by:

$$\mathrm{V}_{t/n} = \{z \in \mathbb{R}^d : \alpha_n(z) \geq t/n\} \tag{3}$$

and the corresponding set of minima, which Proposition 1 will show to be finite although $\mathrm{V}_{t/n}$ is infinite, by $\mathrm{L}_t = \min \mathrm{V}_{t/n}$. Since $\alpha_n(z)$ is a coordinate-wise non-decreasing function, $\mathrm{V}_{t/n}$ is equal to the upper set of L_t, i.e.:

$$\mathrm{V}_{t/n} = \{z \in \mathbb{R}^d : \mathrm{L}_t \trianglelefteq z\} \tag{4}$$

Thus, the EAF computation problem may be defined as:

Problem 1 (EAF computation). Given an input sequence of non-empty non-dominated point sets:

$$\mathrm{S} = (\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_n) \tag{5}$$

containing

$$m = \sum_{i=1}^{n} m_i, \quad m_i = |\mathrm{X}_i| \tag{6}$$

input points, find the output sequence

$$\mathrm{R} = (\mathrm{L}_1, \mathrm{L}_2, \ldots, \mathrm{L}_n) \tag{7}$$

where L_t, $t = 1, \ldots, n$, denotes the set of minima of the t/n-superlevel set, $\mathrm{V}_{t/n}$, of $\alpha_n(\mathrm{X}_1, \ldots, \mathrm{X}_n; z)$. The total number of output points is

$$\ell = \sum_{t=1}^{n} \ell_t, \quad \ell_t = |\mathrm{L}_t| \tag{8}$$

It remains to show how the output sets $\mathrm{L}_1, \ldots, \mathrm{L}_n$ relate to the input sequence, S. For each $t = 1, \ldots, n$, the auxiliary set

$$\mathrm{J}_t = \left\{ \bigvee_{i=1}^{t} z_i : (z_1, \ldots, z_t) \in \prod_{i=1}^{t} \mathrm{X}_{j_i},\ j_i \in \{1, \ldots, n\} \wedge (a < b \Leftrightarrow j_a < j_b) \right\} \tag{9}$$

is defined, where $\bigvee_{i=1}^{t} z_i$ denotes the *join* (component-wise maximum, or least upper bound) of points $z_1, \ldots, z_t \in \mathbb{R}^d$, $\prod_{i=1}^{t} \mathrm{X}_{j_i}$ denotes the Cartesian product of sets $\mathrm{X}_{j_1}, \ldots, \mathrm{X}_{j_t}$, and $(\mathrm{X}_{j_1}, \ldots, \mathrm{X}_{j_t})$ is any length-t subsequence of S. Then, the following holds true:

Proposition 1. L_t *is finite and equal to the set of minima of* J_t, $t = 1, \ldots, n$.

Proof.

1. $\mathrm{J}_t \subseteq \mathrm{V}_{t/n}$, since all elements of J_t are attained by at least t distinct input sets by construction.
2. Each minimum of $\mathrm{V}_{t/n}$ must be the least upper bound (join) of t points from distinct input sets, which is in J_t by construction. Therefore, $\min \mathrm{V}_{t/n} \subseteq \mathrm{J}_t$.
3. Together, 1. and 2. imply that $\min \mathrm{V}_{t/n} \subseteq \min \mathrm{J}_t$.
4. $\min \mathrm{J}_t \subseteq \min \mathrm{V}_{t/n}$. Assume that a minimum of J_t is not a minimum of $\mathrm{V}_{t/n}$. Then, there must be a minimum of $\mathrm{V}_{t/n}$ which dominates it. Given 3., this minimum of $\mathrm{V}_{t/n}$ must be in J_t as well, which gives rise to a contradiction.
5. Together, 3. and 4. imply that $\min \mathrm{V}_{t/n} = \min \mathrm{J}_t$. Since J_t is finite, so is L_t.

4 The Size of the Output Sets

As the complexity of any algorithm for the EAF computation problem will necessarily be bounded below by the size of the output sets $\mathrm{L}_1, \ldots, \mathrm{L}_n$, the maximum size of these sets is of interest.

4.1 The Two-Objective Case

When $d = 2$, an upper bound for the total number of output points, ℓ, can be easily obtained by noting that all output sets L_t, $t = 1, \ldots, n$, are non-dominated point sets, which implies that all points in L_t must have distinct coordinates. Therefore, the cardinality $\ell_t = |\mathrm{L}_t|$ is bounded above by the total number of input points, m. In other words, $\ell_t \in O(m)$, which leads to $\ell \in O(nm)$.

Furthermore, the above bound can be shown to be tight by means of an example. Given two positive integers n and m such that $m \geq n$, consider the input sequence:

$$\mathrm{S} = (\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_n) \tag{10}$$

with

$$\mathrm{X}_i = \big\{(j, m - j + 1) : (j - i) \bmod n = 0, j \in \{1, \ldots, m\}\big\} \tag{11}$$

for all $i = 1, \ldots, n$. Then, $\ell_t = m - t + 1$ and $\ell = n(2m - n + 1)/2$. Since $m \geq n$, $\ell \in \Omega(nm)$, and the following proposition holds true:

Proposition 2. *In the 2-dimensional EAF computation problem,* $\ell \in \Theta(nm)$.

4.2 The Three-Objective Case

To establish an upper bound on the maximum total number of output points when $d = 3$, assume, without loss of generality, that all input points have distinct coordinates.[1] Recall that output points are the join of points from different input

[1] If this is not the case, add sufficiently small perturbations to the common coordinate values in the input so that all coordinates become distinct. Note that the number of minimal elements of J_t, $t = 1, \ldots, n$, may increase as a result, but cannot decrease.

sets, and that all output sets L_t, $t = 1, \ldots, n$, are non-dominated point sets by definition. In particular, L_1 is the set of minima of $\mathrm{J}_1 = \bigcup_{i=1}^{n} \mathrm{X}_i$, which implies that $\ell_1 \in O(m)$.

Since the number of objectives is $d = 3$, points in each output set L_t must differ from each other in at least two coordinates, or one would dominate the other. Thus, for each input point $p = (p^x, p^y, p^z) \in \mathrm{J}_1$, there may be at most two distinct points $q = (q^x, q^y, q^z)$ in each output set L_t such that $(q^x, q^z) = (p^x, p^z)$ and $q^y \geq p^y$, or $(q^y, q^z) = (p^y, p^z)$ and $q^x \geq p^x$. Note that this does not exclude the case where $q = p$, nor does it imply that such a point $q \in \mathrm{L}_t$ exists.

Moreover, for each point $q = (p^x, q^y, p^z) \in \mathrm{L}_t$ considered above, there may be at most one point $r = (r^x, r^y, r^z)$ in each L_j, $t < j \leq n$, such that $(r^x, r^y) = (p^x, q^y)$ and $r^z > p^z$, and similarly for each $q = (q^x, p^y, p^z) \in \mathrm{L}_t$. Thus, each input point p may be associated with $O(n)$ output points of the form q above and with $O(n^2)$ output points of the form r.

Finally, every output point must be of one of these two types, q or r. This is clearly true for every output point that is either an input point as well, or is dominated by some input point while differing from it only in the value of a single, x or y, coordinate, which corresponds to type q. In all other cases, the join of the input point(s) defining the x and y coordinates of a given output point will differ from that output point in the z coordinate value only, which must now be equal to that of (one of) the other input point(s) considered. This join must be a minimum of J_j for some $j < t$, or the original output point would not be a minimum of J_t, either. Therefore, type r applies.

Since there are m input points in total, the following holds true:

Proposition 3. *In the 3-dimensional EAF computation problem,* $\ell \in O\left(n^2 m\right)$.

4.3 More Than Three Objectives

As shown above, the maximum number of output points, ℓ, grows at most linearly with the total number of input points, m, when the number of objectives is two or three, and the number of input sets, n, is constant. Unfortunately, this result does not carry over to more than three objectives. As an example, consider, for given positive integers m_1 and m_2, that:

$$\mathrm{S} = (\mathrm{X}_1, \mathrm{X}_2) \tag{12}$$

$$\mathrm{X}_1 = \{(j_1, m_1 - j_1 + 1, 0, 0) : j_1 = 1, \ldots, m_1\} \tag{13}$$

$$\mathrm{X}_2 = \{(0, 0, j_2, m_2 - j_2 + 1) : j_2 = 1, \ldots, m_2\} \tag{14}$$

Then,

$$\mathrm{J}_1 = \mathrm{X}_1 \cup \mathrm{X}_2 \tag{15}$$

$$\mathrm{J}_2 = \{(j_1, m_1 - j_1 + 1, j_2, m_2 - j_2 + 1) : j_1 = 1, \ldots, m_1,\ j_2 = 1, \ldots, m_2\} \tag{16}$$

Since, in this case, both J_1 and J_2 are non-dominated point sets, $(\mathrm{L}_1, \mathrm{L}_2) = (\mathrm{J}_1, \mathrm{J}_2)$, and

$$|\mathrm{L}_1| = m_1 + m_2 = m \tag{17}$$

$$|\mathrm{L}_2| = m_1 m_2 \tag{18}$$

Algorithm 1. EAF computation

1: **for** $t = 1$ **to** n **do**
2: compute J_t from $\mathrm{X}_1, \ldots, \mathrm{X}_n$
3: $\mathrm{L}_t \leftarrow \mathsf{minima}(\mathrm{J}_t)$

Setting $m_1 = m_2 = m/2$, the total number of output points is $\ell = m + m^2/4$, which establishes a lower bound of $\Omega(m^2)$ in the four-objective case.

The above lower bound can be shown to apply also to higher numbers of objectives (through dimension embedding), but tighter bounds for increasing values of d can be achieved by extending the proposed construction. Assuming that d is constant and even, consider $n = d/2$ input sets:

$$\mathrm{X}_1 = \{(j_1, m_1 - j_1 + 1, 0, 0, \ldots, 0, 0) \in \mathbb{R}^d : j_1 = 1, \ldots, m_1\} \tag{19}$$

$$\mathrm{X}_2 = \{(0, 0, j_2, m_2 - j_2 + 1, 0, 0, \ldots, 0, 0) \in \mathbb{R}^d : j_2 = 1, \ldots, m_2\} \tag{20}$$

$$\vdots$$

$$\mathrm{X}_i = \{(0, 0, \ldots, 0, 0, j_i, m_i - j_i + 1, 0, 0, \ldots, 0, 0) \in \mathbb{R}^d : j_i = 1, \ldots, m_i\} \tag{21}$$

$$\vdots$$

$$\mathrm{X}_n = \{(0, 0, \ldots, 0, 0, j_n, m_n - j_n + 1) \in \mathbb{R}^d : j_n = 1, \ldots, m_n\} \tag{22}$$

of equal size $m_1 = \cdots = m_n = m/n$, and focus on the cardinality of $|\mathrm{L}_n| = |\mathrm{J}_n| = (m/n)^n = (m/n)^{d/2}$. Then, the following general result may be stated:

Proposition 4. *In the d-dimensional EAF computation problem, the maximum total number of output points is* $\Omega\left(m^{\lfloor d/2 \rfloor}\right)$.

5 Time Complexity and Algorithms

The cardinality lower bounds derived in the previous section provide trivial lower bounds on the time complexity of the EAF computation problem. Additionally, the known lower bound of $O(n \log n)$ on the complexity of finding the minima (or, alternatively, the maxima) of a point set [9, Theorem 4.8] applies to EAF computation as well, since $\mathrm{L}_1 = \min \mathrm{J}_1$. Formally:

Proposition 5. *In the comparison-tree model, any algorithm that solves the EAF computation problem in d dimensions requires time* $\Omega\left(m \log m + m^{\lfloor d/2 \rfloor}\right)$. *The time required when* $d = 2$ *is* $\Omega\left(m \log m + nm\right)$.

The design of algorithms for the EAF computation problem is approached here by noting that $\mathrm{L}_t = \min \mathrm{J}_t$, as stated in Proposition 1, which immediately suggests dividing the problem into two main computation steps, as outlined in Algorithm 1. The disadvantage of such a brute-force approach is that $|\mathrm{J}_t|$ grows exponentially in t, leading to overall exponential runtime growth in n, even in two or three dimensions.

Algorithm 2. Minima of a set of points

Input: X // a set of points in $\mathbb{R}^d$
Output: L_1 // the set of minima of X
1: $m \leftarrow |X|$
2: Q is a queue containing X sorted in *ascending* order of coordinate d
3: $L_1 \leftarrow \varnothing$
4: $L_1^* \leftarrow \varnothing$
5: **while** $Q \neq \varnothing$ **do**
6: $p \leftarrow \mathsf{pop}(Q)$
7: p^* is the projection of p onto the first $d-1$ dimensions
8: **if** $L_1^* \not\preceq p^*$ **then**
9: $L_1^* \leftarrow \mathsf{minima}(L_1^* \cup \{p^*\})$
10: $L_1 \leftarrow L_1 \cup \{p\}$
11: **return** L_1

A better alternative consists of alternating between J_t computation steps and L_t computation steps, while avoiding generating points in J_t which would be dominated by those already in L_t. Such an approach is consistent with the well-known dimension-sweep paradigm [9, p. 10f] of computational geometry, and the EAF algorithms developed in this work are based on existing dimension-sweep algorithms for minima [7].

Consider the computation of L_1, which, as pointed out earlier, consists of determining the minima of all input points, regardless of the input set to which each point actually belongs. A general solution to this problem [9, p. 160] is outlined in Algorithm 2, under the common assumption that all input points are distinct, and have distinct coordinate values.The algorithm starts from an empty output set L_1, and visits input points in ascending order of their last coordinate, i.e., it sweeps X along the last dimension. Clearly, a newly visited point cannot dominate previously visited points, but it will be dominated by earlier points whenever this is true with respect to the first $d-1$ coordinates. Therefore, it suffices to check the projected point p^* against a set, L_1^*, of minimal projections in order to decide whether or not p itself is a minimum. Due to this dimensionality reduction, efficient dimension-sweep algorithms can be obtained for the minima problem in two and three dimensions by specialising the dominance check and update steps (lines 8–9) in each case.

In practice, input coordinate values may not be all distinct, and adjustments to dimension-sweep algorithms in their basic form may be required, which can often be made without compromising the worst-case complexity of the original versions of the algorithms. In the following, no distinct-coordinate assumptions are made, and repeated coordinate values and/or non-disjoint input sets, $X_1, \ldots, X_n$, are explicitly allowed in the input to the EAF algorithms presented.

5.1 The Two-Objective Case

The general approach of Algorithm 2 becomes particularly simple when $d = 2$. In that case, p^* is a scalar, and L_1^* may contain at most one element. Therefore,

Algorithm 3. EAF computation in two dimensions

Input: $\mathrm{S} = (\mathrm{X}_1, \ldots, \mathrm{X}_n)$ // a sequence of non-dominated point sets
Output: $\mathrm{R} = (\mathrm{L}_1, \ldots, \mathrm{L}_n)$ // a sequence of non-dominated point sets
1: $\mathrm{X} \leftarrow \biguplus_{i=1}^{n} \mathrm{X}_i$ // multiset sum, duplicate points are allowed
2: $m \leftarrow \sum_{i=1}^{n} |\mathrm{X}_i|$
3: X^x is X sorted in *descending* order of the x coordinate
4: X^y is X sorted in *ascending* order of the y coordinate
5: **for** $t = 1$ **to** n **do**
6: $\quad \mathrm{L}_t \leftarrow \varnothing$
7: $\quad \mathrm{Q}^x \leftarrow \mathrm{X}^x$ // initialise queue
8: $\quad \mathrm{Q}^y \leftarrow \mathrm{X}^y$ // initialise queue
9: $\quad \mathrm{A} \leftarrow \varnothing$
10: $\quad level \leftarrow 0$
11: $\quad p \leftarrow (\infty, -\infty)$
12: $\quad$ **while** $\mathrm{Q}^x \neq \varnothing$ **and** $\mathrm{Q}^y \neq \varnothing$ **do**
13: $\quad\quad$ **repeat**
14: $\quad\quad\quad q \leftarrow \mathsf{pop}(\mathrm{Q}^y)$
15: $\quad\quad\quad$ **if** $q^x < p^x$ **then**
16: $\quad\quad\quad\quad$ **if** $\mathsf{input\_set}(q) \notin \mathrm{A}$ **then**
17: $\quad\quad\quad\quad\quad level \leftarrow level + 1$
18: $\quad\quad\quad\quad \mathrm{A} \leftarrow \mathrm{A} \uplus \{\mathsf{input\_set}(q)\}$ // multiset sum
19: $\quad\quad$ **until** $\mathrm{Q}^y = \varnothing$ **or** $(level \geq t$ **and** $q^y \neq \mathsf{top}(\mathrm{Q}^y)^y)$
20: $\quad\quad$ **if** $level \geq t$ **then**
21: $\quad\quad\quad$ **repeat**
22: $\quad\quad\quad\quad p \leftarrow \mathsf{pop}(\mathrm{Q}^x)$
23: $\quad\quad\quad\quad$ **if** $p^y \leq q^y$ **then** // $(p^x, q^y) \in \mathrm{J}_t$
24: $\quad\quad\quad\quad\quad \mathrm{A} \leftarrow \mathrm{A} \setminus \{\mathsf{input\_set}(p)\}$ // multiset difference
25: $\quad\quad\quad\quad\quad$ **if** $\mathsf{input\_set}(p) \notin \mathrm{A}$ **then**
26: $\quad\quad\quad\quad\quad\quad level \leftarrow level - 1$
27: $\quad\quad\quad$ **until** $level < t$ **and** $(\mathrm{Q}^x = \varnothing$ **or** $p^x \neq \mathsf{top}(\mathrm{Q}^x)^x)$
28: $\quad\quad\quad \mathrm{L}_t \leftarrow \mathrm{L}_t \cup \{(p^x, q^y)\}$
29: **return** $(\mathrm{L}_1, \ldots, \mathrm{L}_n)$

each iteration of the while loop can be performed in $O(1)$ time, and the algorithm runs in asymptotically optimal, $O(m \log m)$ time due to sorting in line 2.

Extending this approach to the full EAF computation problem in two dimensions is not only possible, but a C implementation of such an algorithm was contributed by the first author to the PISA platform [1] in 2005. A somewhat simpler, but functionally-equivalent, algorithm to compute the EAF in two dimensions is presented as Algorithm 3.

Input sets are merged, pre-sorted according to each coordinate, and stored into two queues, Q^x and Q^y. The following operations can be performed in constant time: $\mathsf{top}(\mathrm{Q})$ returns the element at the top of a queue Q; $\mathsf{pop}(\mathrm{Q})$ retrieves the element at the top and removes it from Q; $\mathsf{input\_set}(p)$ returns the index of the input set containing p.

Each output set L_t is computed independently from the others. For each $t = 1, \ldots, n$, and starting from an empty L_t, input points are visited in ascending

order of their y coordinates until points from at least t different input sets have been visited (lines 13–19). The last point visited, q, establishes the smallest value of the y coordinate of any point in J_t and, thus, of any of its minima. A second sweep is then made in descending order of x-coordinate values (lines 21–27). For each point p thus visited, if it is such that $p^y \leq q^y$, then (p^x, q^y) must be an element of J_t. The number of points from each input set which dominate (p^x, q^y) is tracked using multiset A and the variable *level*.

Note that, as long as the second repeat-until loop has not exited, the number of input sets that attain (p^x, q^y) must be at least t. Also, p and q must be either the same point or belong to distinct input sets. Otherwise, the first repeat-until loop would have exited before q was reached. In addition, if the current point p is the only one from its input set to dominate (p^x, q^y), then (p^x, q^y) is actually a minimum of J_t and, consequently, an element of L_t.

This process is iterated by alternating between sweeps along the y and x dimensions until either queue is exhausted. A new minimum of J_t is obtained after each iteration of the outer while loop (line 12–28), with the possible exception of the last one. As a result, the exponential growth associated with the full enumeration of J_t is avoided because, once a minimum of J_t is found, no further elements of J_t dominated by that minimum are ever generated. Repeated coordinate values are handled by the conditions underlined in lines 19 and 27.

Regarding algorithmic complexity, sorting the input requires $O(m \log m)$ time, and each output set L_t is generated in $O(m)$ time. Since the number of output sets is n, the overall time complexity of the algorithm is $O(m \log m + nm)$, which matches the corresponding lower bound stated in Proposition 5. Algorithm 3 is, therefore, asymptotically optimal.

5.2 The Three-Objective Case

Asymptotically optimal algorithms for minima in three dimensions can also be obtained from Algorithm 2. Since L_1^* is now a set of minima in two dimensions, it admits a total order, and may be organised as a height-balanced binary search tree on either of the first two coordinates. This allows the dominance check in line 8 to be performed in $O(\log m)$ time and the L_1^* update in line 9 to be performed in amortised $O(\log m)$ time. Indeed, each update consists of a search operation and at most one insertion, both of which have complexity $O(\log m)$ time, plus a variable number of removals[2] which, in total, cannot exceed m. Since each removal can also be performed in $O(\log m)$ time, the overall complexity is $O(m \log m)$, and the algorithm is asymptotically optimal [7].

Extending this approach to the EAF computation problem in three dimensions is much less straightforward than it was in the two-dimensional case. A discussion of the main aspects of the solution proposed as Algorithm 4 follows.

Data structures. Since all input sets $\mathrm{X}_1, \ldots, \mathrm{X}_n$ and output sets $\mathrm{L}_1, \ldots, \mathrm{L}_n$ are non-dominated point sets, $2n$ data structures based on a height-balanced binary search tree, as in the minima algorithm described above, are used to manage

[2] Each point to be removed may be found in constant time if the tree is threaded.

Algorithm 4. EAF computation in three dimensions

Input: $\mathrm{S} = (\mathrm{X}_1, \ldots, \mathrm{X}_n)$ // a sequence of non-dominated point sets
Output: $\mathrm{R} = (\mathrm{L}_1, \ldots, \mathrm{L}_n)$ // a sequence of non-dominated point sets
1: $\mathrm{X} = \biguplus_{i=1}^{n} \mathrm{X}_i$ // multiset sum, duplicate points are allowed
2: $m \leftarrow \sum_{i=1}^{n} |\mathrm{X}_i|$
3: Q is X sorted in *ascending* order of the z coordinate
4: $\mathrm{L}_t \leftarrow \varnothing$, $t = 1, \ldots, n$
5: $\mathrm{L}_t^* \leftarrow \{(-\infty, \infty, -\infty), (\infty, -\infty, -\infty)\}$, $t = 1, \ldots, n$ // Sentinels
6: $\mathrm{X}_i^* \leftarrow \{(-\infty, \infty, -\infty), (\infty, -\infty, -\infty)\}$, $i = 1, \ldots, n$ // Sentinels
7: $p \leftarrow \mathsf{pop}(\mathrm{Q})$
8: $j \leftarrow \mathsf{input\_set}(p)$
9: $\mathsf{insert}(p, \mathrm{X}_j^*)$
10: $\mathsf{insert}(p, \mathrm{L}_1^*)$
11: $\mathrm{A} \leftarrow \{j\}$
12: $t_{\max} \leftarrow 1$
13: **while** $\mathrm{Q} \neq \varnothing$ **do**
14: $\quad p \leftarrow \mathsf{pop}(\mathrm{Q})$
15: $\quad j \leftarrow \mathsf{input\_set}(p)$
16: $\quad q \leftarrow \mathsf{floor}^x(p, \mathrm{X}_j^*)$
17: $\quad$ **if** $p^y < q^y$ **then** // always true if X_j is indeed a non-dominated point set
18: $\quad\quad t \leftarrow t_{\max}$
19: $\quad\quad t_{\min} \leftarrow 1$
20: $\quad\quad$ **while** $t \geq t_{\min}$ **do**
21: $\quad\quad\quad r \leftarrow \mathsf{floor}^x(p, \mathrm{L}_t^*)$
22: $\quad\quad\quad$ **if** $r^y \leq p^y$ **then**
23: $\quad\quad\quad\quad t_{\min} \leftarrow t + 1$
24: $\quad\quad\quad$ **else if** $r^y < q^y$ **then**
25: $\quad\quad\quad\quad s_t \leftarrow (p^x, r^y, p^z)$
26: $\quad\quad\quad$ **else**
27: $\quad\quad\quad\quad s_t \leftarrow \mathsf{lower}^y(q, \mathrm{L}_t^*)$
28: $\quad\quad\quad t \leftarrow t - 1$
29: $\quad\quad$ **repeat**
30: $\quad\quad\quad q \leftarrow \mathsf{higher}^x(q, \mathrm{X}_j^*)$
31: $\quad\quad\quad b \leftarrow \max(p^y, q^y)$
32: $\quad\quad\quad$ **for** $t = t_{\max}$ **down to** $t_{\min}$ **do**
33: $\quad\quad\quad\quad$ **while** $s_t^y \geq b$ **and** $(s_t^y > b$ **or** $b > p^y)$ **do**
34: $\quad\quad\quad\quad\quad$ **if** $s_t^x \geq q^x$ **then**
35: $\quad\quad\quad\quad\quad\quad s_t \leftarrow \mathsf{lower}^y(q, \mathrm{L}_t^*)$
36: $\quad\quad\quad\quad\quad$ **else**
37: $\quad\quad\quad\quad\quad\quad$ submit (s_t^x, s_t^y, p^z) to L_{t+1}^*
38: $\quad\quad\quad\quad\quad\quad s_t \leftarrow \mathsf{higher}^x(s_t, \mathrm{L}_t^*)$
39: $\quad\quad$ **until** $q^y \leq p^y$
40: $\quad\quad$ **for** $t = t_{\max}$ **down to** $t_{\min}$ **do**
41: $\quad\quad\quad$ **if** $s_t^x < q^x$ **then**
42: $\quad\quad\quad\quad$ submit (s_t^x, p^y, p^z) to L_{t+1}^*
43: $\quad\quad$ submit p to X_j^*
44: $\quad\quad$ submit p to $\mathrm{L}_{t_{\min}}^*$
45: $\quad$ **if** $j \notin \mathrm{A}$ **then**
46: $\quad\quad \mathrm{A} \leftarrow \mathrm{A} \cup \{j\}$
47: $\quad\quad t_{\max} \leftarrow \min(t_{\max} + 1, n - 1)$
48: $\mathrm{L}_t \leftarrow \mathrm{L}_t \cup (\mathrm{L}_t^* \setminus \{(-\infty, \infty, -\infty), (\infty, -\infty, -\infty)\})$, $t = 1, \ldots, n$
49: **return** $(\mathrm{L}_1, \ldots, \mathrm{L}_n)$

them. Points from each set are organised in the corresponding data structure with respect to their projection onto the xy-plane, which conveniently allows both x and y coordinates to be used as search key as long as the projections of all points stored are non-dominated. Insertion, removal and the following search operations can be performed in logarithmic time on such a data structure, X^*:

floor$^x(p, \mathrm{X}^*)$ The point $q \in \mathrm{X}^*$ with the greatest $q^x \leq p^x$
lower$^x(p, \mathrm{X}^*)$ The point $q \in \mathrm{X}^*$ with the greatest $q^x < p^x$
ceiling$^x(p, \mathrm{X}^*)$ The point $q \in \mathrm{X}^*$ with the least $q^x \geq p^x$
higher$^x(p, \mathrm{X}^*)$ The point $q \in \mathrm{X}^*$ with the least $q^x > p^x$

The corresponding operations with respect to the y coordinate are available as well, and have the same complexity.

Algorithm description. Output sets are computed by sweeping along the z dimension and searching for the different types of output points identified in Section 4. As a consequence, all output sets are computed concurrently, instead of independently from each other.

For simplicity, begin by assuming that no repeated z-coordinate values appear in the input, a restriction which will be lifted later. Before entering the main loop in line 13, the input data is queued in ascending order of the z coordinate, and individual data structures X_i^* and L_t^*, $i, t = 1, \ldots, n$, are created and initialised with sentinels (lines 1–6), so that all search operations are guaranteed to return a point. Then, the first point retrieved from the queue is inserted into the corresponding X_j^* and into L_1^* (lines 7–10), as it must be a minimal element of J_1. Set A is used to track which input sets have been visited so far.

In the main loop, each new input point dequeued is checked to ascertain that it is not dominated by other points in its input set (lines 14–17). Then, for each non-empty $\mathrm{L}_{n-1}, \mathrm{L}_{n-2}, \ldots, \mathrm{L}_1$, new output points are generated as follows:

1. For each input point $p \in \mathrm{X}$, element of some input set X_j, $j \in \{1, \ldots, n\}$, an output point $r = (r^x, r^y, r^z) \in \mathrm{L}_t$ is sought, such that $\mathrm{X}_j \not\trianglelefteq r$, $(p^x, p^z) \geq (r^x, r^z)$ and $p^y < r^y$. This is depicted as point 1 in Fig. 1. Then, $s = (p^x, r^y, p^z)$ must be an element of J_{t+1}, as it is attained by one more input set than r. In addition, if s^y is required to be minimal, s will be an element of L_{t+1}. Such points, r, if they exist, are identified in the first inner loop of the algorithm (lines 20–28), and the corresponding output point, s, depicted as point A in the figure, is generated in line 25. For convenience, it is output only later, in line 37.
2. For each input point $p \in \mathrm{X}$, element of some input set X_j, $j \in \{1, \ldots, n\}$, all output points $s = (s^x, s^y, s^z) \in \mathrm{L}_t$ such that $\mathrm{X}_j \not\trianglelefteq s$, $(p^x, p^y) < (s^x, s^y)$ and $p^z \geq s^z$ are sought. Then, (s^x, s^y, p^z) must also be an element of J_{t+1}. Since input points are processed in ascending order of z-coordinate values, (s^x, s^y, p^z) will be an element of L_{t+1}. Such points, if they exist, are determined in the second inner-loop (lines 29–39) and, eventually, also as the last point found in the first inner loop, in line 27. They are depicted as points 2, 3 and 6 in Fig. 1.

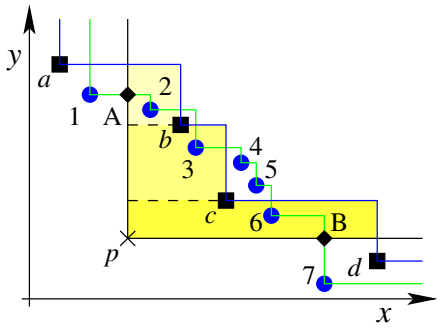

Fig. 1. Example, where $\mathrm{L}_t^* = \{1, ..., 7\}$, $\mathrm{X}_j^* = \{a, b, c, d\}$, and p is the new point in X_j

Algorithm 5. Submit u to L_t^*

```
1: v ← floor^x(u, L_t^*)
2: if u^y < v^y then
3:    for all w ∈ L_t^* : (u^x, u^y) ≤ (w^x, w^y) do
4:       if u^z > w^z then
5:          L_t ← L_t ∪ {w}
6:       remove(w, L_t^*)
7:    insert(u, L_t^*)
```

3. In the third inner loop (lines 40–42), output points analogous to those determined in the first loop, but with the roles of the x and y coordinates reversed, are computed (point B in the figure).
4. Finally, each input point p will itself be a member of the output set $\mathrm{L}_{t_{\min}}$, where $t_{\min}$ is the lowest index t such that L_t^* does not attain p in the current iteration of the outer loop. The value of $t_{\min}$ is determined in lines 22–23, and L_t is updated in line 44.

The L_t^* data structures are updated as detailed in Algorithm 5. Provided that the new point u is not dominated by the points currently in L_t^* (line 2), points w in L_t^* whose projections are dominated by u are removed (lines 3–6), and the new point u is inserted (line 7). Otherwise, u is simply discarded.

Repeated z-coordinate values in the input sets are handled by delaying the addition of points to L_t until they are removed from L_t^* as a result of the insertion of a new point (lines 4–5). This guarantees that output points generated in one iteration of the outer loop which become dominated in a subsequent iteration due to the processing of a second input point with the same z coordinate are not actually added to the output. Lines 4–5 do not apply to the updating of the X_j^* data structures, which is otherwise identical.

Complexity. A complete run of the algorithm involves m searches and (up to) m insertions into the X_j^* data structures, plus up to m search-removal pairs, to maintain the input data structures. Since the total number of input points is m,

the cost of these operations is bounded by $O(\log m)$, and the complexity due to them is $O(m \log m)$.

Maintaining the output data structures, L_t^*, requires $O(n^2 m \log(n^2 m)) = O(n^2 m \log m)$ time,[3] since there are $O(n^2 m)$ output points. In addition, $O(m)$ searches in the X_j^* data structures and $O(n^2 m)$ searches in the L_t^* data structures are performed in the body of the algorithm, including $O(nm)$ searches in L_t^* that may not lead to the generation of new output points. As this extra work is also done in $O(n^2 m \log m)$ time, the time complexity of Algorithm 4 is $O(n^2 m \log m)$, and the algorithm is asymptotically optimal as long as the number of input sets, n, is assumed to be constant (compare with Proposition 5). When $m \in O(n)$, the time complexity of the algorithm is only a logarithmic factor worse than the cardinality upper bound derived in Section 4, Proposition 3.

6 Concluding Remarks

In this work, the EAF computation problem has been formalised as the problem of computing a finite description of all of its superlevel sets (or the corresponding summary attainment surfaces) from experimental data. After proving that each component of the solution of this problem consists of the set of minima of a suitable auxiliary set constructed based on the input data, the size of the output sets which describe the EAF was shown to grow linearly with the number of input points only when $d = 2, 3$. Finally, efficient algorithms for the EAF in two and three dimensions were developed based on existing dimension-sweep algorithms for minima. The algorithm for $d = 3$ is asymptotically optimal when the number of input sets is fixed, whereas the algorithm for $d = 2$ is asymptotically optimal in the general case.

Extending the algorithms proposed here to more than three dimensions is the subject of further work, although quadratic complexity with respect to the number of input points will be unavoidable in the worst case. This is a direct consequence of the lower bound stated in Proposition 5, but it may still be possible to seek improved performance in non-worst-case scenarios, e.g., by developing output-sensitive EAF algorithms.

A C-language implementation of Algorithms 3 and 4 is available from the authors on `http://eden.dei.uc.pt/~cmfonsec/software.html`. As an example of practical runtime performance, computing the EAF of a set of 50 spherical, three-objective fronts with 240 points each took only 6 seconds on a single core of an AMD Opteron 2216 HE 2.4GHz processor. It is expected that these results will contribute to a more widespread use of the empirical attainment function as a performance assessment tool in Evolutionary Multiobjective Optimisation.

Acknowledgements. Andreia P. Guerreiro acknowledges financial support from CEG-IST through an Integration into Research Grant sponsored by the Portuguese Foundation for Science and Technology. Manuel López-Ibáñez acknowledges support from the FRFC project "*Méthodes de recherche hybrides pour la résolution de problèmes complexes*".

[3] Note that $n \leq m$.

References

1. Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E.: PISA – A platform and programming language independent interface for search algorithms. In: Fonseca, C.M., et al. (eds.) EMO 2003. LNCS, vol. 2632, pp. 494–508. Springer, Heidelberg (2003)
2. Fonseca, C.M., Fleming, P.J.: On the performance assessment and comparison of stochastic multiobjective optimizers. In: Voigt, H.M., et al. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 584–593. Springer, Heidelberg (1996)
3. Fonseca, C.M., Grunert da Fonseca, V., Paquete, L.: Exploring the performance of stochastic multiobjective optimisers with the second-order attainment function. In: Coello, C.C., et al. (eds.) EMO 2005. LNCS, vol. 3410, pp. 250–264. Springer, Heidelberg (2005)
4. Grunert da Fonseca, V., Fonseca, C.M.: The attainment-function approach to stochastic multiobjective optimizer assessment and comparison. In: Bartz-Beielstein, T., et al. (eds.) Experimental Methods for the Analysis of Optimization Algorithms. Springer, Berlin (2010)
5. Grunert da Fonseca, V., Fonseca, C.M., Hall, A.O.: Inferential performance assessment of stochastic optimisers and the attainment function. In: Zitzler, E., et al. (eds.) EMO 2001. LNCS, vol. 1993, pp. 213–225. Springer, Heidelberg (2001)
6. Knowles, J.D.: A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers. In: Proceedings of the 5th International Conference on Intelligent Systems Design and Applications, pp. 552–557 (2005)
7. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. Journal of the ACM 22(4), 469–476 (1975)
8. López-Ibáñez, M., Paquete, L., Stützle, T.: Exploratory analysis of stochastic local search algorithms in biobjective optimization. In: Bartz-Beielstein, T., et al. (eds.) Experimental Methods for the Analysis of Optimization Algorithms, pp. 209–222. Springer, Berlin (2010)
9. Preparata, F.P., Shamos, M.I.: Computational Geometry. An Introduction, 2nd edn. Springer, Berlin (1988)
10. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - A comparative case study. In: Eiben, A.E., et al. (eds.) PPSN V 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)