Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 569346, 18 pages
http://dx.doi.org/10.1155/2014/569346

Research Article

Visualizing Exact and Approximated 3D Empirical Attainment Functions

Tea Tušar and Bogdan Filipič

Department of Intelligent Systems, Jožef Stefan Institute and Jožef Stefan International Postgraduate School, Jamova cesta 39, 1000 Ljubljana, Slovenia

Correspondence should be addressed to Tea Tušar; tea.tusar@ijs.si

Received 4 June 2014; Accepted 10 August 2014; Published 17 September 2014

Academic Editor: Chunlin Chen

Most real-world engineering optimization problems are inherently multiobjective, for example, searching for trade-off solutions of high quality and low cost. As no single optimal solution exists for such problems, multiobjective optimizers provide sets of optimal (or near-optimal) trade-off solutions to choose from. The empirical attainment function (EAF) is a powerful tool that can be used to analyze and compare the performance of these optimizers. While the visualization of EAFs is rather straightforward in two objectives, the three-objective case presents a great challenge as we need to visualize a large number of 3D cuboids. This paper addresses the visualization of exact as well as approximated 3D EAF values and differences in these values provided by two competing multiobjective optimizers. We show that the exact EAFs can be visualized using slicing and maximum intensity projection (MIP), while the approximated EAFs can be visualized using slicing, MIP, and direct volume rendering. In addition, the paper demonstrates the use of the proposed visualization techniques on a steel casting optimization problem.

1. Introduction

When solving engineering optimization problems it is usually not enough to find the best solution with regard to single measure of quality, but other objectives, such as other measures of quality, robustness of the solution or its cost, can be important too. While traditional approaches were used to combine all these heterogeneous objectives into a single one and solve the resulting singleobjective optimization problem, in the last two decades a lot of research has been directed into developing algorithms able to tackle these problems in their true multiobjective form. Most notably, multiobjective evolutionary algorithms (MOEAs) have proven to be effective in solving such problems [1, 2].

A MOEA provides a set of trade-off solutions where no solution from the set is better than any other in all objectives. This is called an *approximation set* and MOEAs typically return a different approximation set in every run. If the performance of a MOEA needs to be analyzed or compared to the performance of another algorithm, several different measures can be adopted [3]. Nevertheless, visualization of approximation sets can give an important insight into the properties of the algorithms (or the problem at hand) and is often used in this regard. While simple scatter plots can visualize 2D and 3D approximation sets (approximation sets of higher dimensions that require more sophisticated methods to be visualized will not be discussed in this paper—the interested reader is referred to [4] for a comprehensive review of such methods), scatter plots of multiple approximation sets can become too cluttered and loose their interpretation potential. The empirical attainment function (EAF), which assigns to each vector in the objective space a value signifying how often the vector was attained by the given approximation sets, can be used instead [5]. Visualization of 2D EAFs entails plotting of rectangles in different colors (or shades of gray) corresponding to EAF values or differences in EAF values and is rather straightforward to do.

This paper focuses on the visualization of EAFs in the 3D case, which is more demanding than the 2D case since a large number of *cuboids* or rather unions of cuboids need to be first computed and then visualized. Building on some previous work [6, 7], we present how the visualization of exact EAFs cuboids can be done using slicing (showing the 2D images

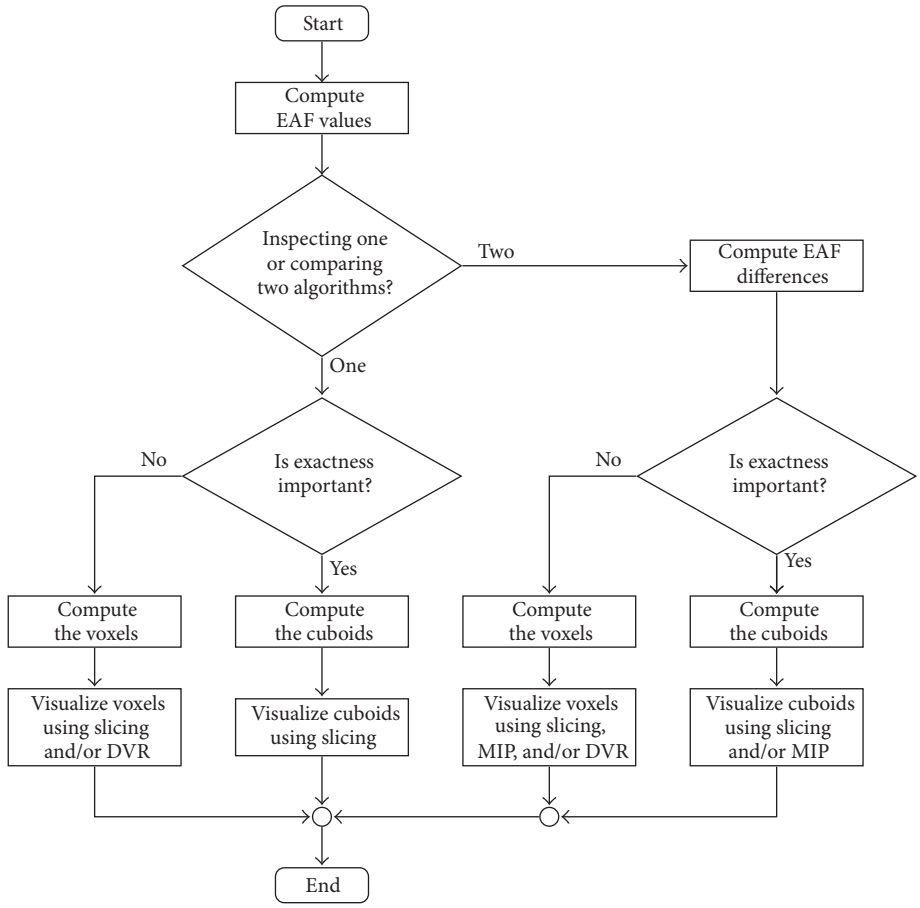


Figure 1: Flowchart of EAF visualization depending on the preferences.

obtained by slicing the cuboids at different angles) and maximum intensity projection (MIP) [8]. However, if exactness is not crucial, the 3D objective space can be approximated using a grid of *voxels*, that is, values in a 3D grid. The paper introduces the idea of visualizing approximated EAFs using slicing, MIP, and direct volume rendering (DVR) [9]. A flowchart explaining the connections among these methods is shown in Figure 1. Finally, we demonstrate the use of the proposed methods on steel casting optimization problem.

The paper is organized as follows. Section 2 details the background and related work, while Section 3 introduces the benchmark approximation sets that are used in the rest of the paper. Visualization of exact and approximated EAFs is presented in Sections 4 and 5, respectively, and summarized in Section 6. The steel casting use case is depicted in Section 7. The paper concludes with final remarks in Section 8.

2. Background and Related Work

In addition to recalling some relations and concepts often used in multiobjective optimization, this section presents formal definitions of some new terms related to the EAF, such as *attainment anchors* and *opposites*. It also reviews the existing methods for visualization of EAFs and concludes with the presentation of slicing, MIP, and DVR.

2.1. Multiobjective Optimization Concepts. The multiobjective optimization problem consists of finding the optimum of a function:

$$\mathbf{f} : X \longrightarrow F,$$
$$\mathbf{f} : (x_1, \ldots, x_n) \longmapsto (f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)), \tag{1}$$

where X is an n-dimensional decision space and F is an m-dimensional objective space ($m \geq 2$). Each solution $\mathbf{x} \in X$ is called a decision vector, while the corresponding element $\mathbf{z} = \mathbf{f}(\mathbf{x}) \in F$ is an objective vector. Without loss of generality we assume that $F \subseteq \mathbb{R}^m$ and all objectives $f_i : X \to \mathbb{R}$ are to be minimized.

As this paper deals with visualization in the objective space, which can be viewed rather independently from the decision space, the following definitions are confined to the objective space.

Definition 1 (Pareto dominance of vectors). The objective vector $\mathbf{z}^A = (z_1^A, \ldots, z_m^A) \in F$ *dominates* the objective vector $\mathbf{z}^B = (z_1^B, \ldots, z_m^B) \in F$; that is, $\mathbf{z}^A \prec \mathbf{z}^B$, if

$$z_i^A \leq z_i^B \quad \forall i \in \{1, \ldots, m\},$$
$$\text{there exists a } j \in \{1, \ldots, m\} \quad \text{for which } z_j^A < z_j^B. \tag{2}$$

Definition 2 (weak Pareto dominance of vectors). The objective vector $\mathbf{z}^A = (z_1^A, \ldots, z_m^A) \in F$ *weakly dominates* the objective vector $\mathbf{z}^B = (z_1^B, \ldots, z_m^B) \in F$; that is, $\mathbf{z}^A \preceq \mathbf{z}^B$, if

$$z_i^A \leq z_i^B \quad \forall i \in \{1, \ldots, m\}. \tag{3}$$

Definition 3 (strict Pareto dominance of vectors). The objective vector $\mathbf{z}^A = (z_1^A, \ldots, z_m^A) \in F$ *strictly dominates* the objective vector $\mathbf{z}^B = (z_1^B, \ldots, z_m^B) \in F$; that is, $\mathbf{z}^A \ll \mathbf{z}^B$, if

$$z_i^A < z_i^B \quad \forall i \in \{1, \ldots, m\}. \tag{4}$$

Definition 4 (incomparability of vectors). The objective vectors $\mathbf{z}^A = (z_1^A, \ldots, z_m^A) \in F$ and $\mathbf{z}^B = (z_1^B, \ldots, z_m^B) \in F$ are *incomparable*; that is, $\mathbf{z}^A \parallel \mathbf{z}^B$, if

$$\mathbf{z}^A \not\preceq \mathbf{z}^B, \qquad \mathbf{z}^B \not\preceq \mathbf{z}^A. \tag{5}$$

Definition 5 (approximation set). A set of objective vectors $Z \subseteq F$ is called an *approximation set* if $\mathbf{z}^A \parallel \mathbf{z}^B$ for any two objective vectors $\mathbf{z}^A, \mathbf{z}^B \in Z$.

Definition 6 (weak Pareto dominance of approximation sets). The approximation set $Z^A \subseteq F$ *weakly dominates* the approximation set $Z^B \subseteq F$; that is, $Z^A \preceq Z^B$, if every $\mathbf{z}^B \in Z^B$ is weakly dominated by at least one $\mathbf{z}^A \in Z^A$.

Definition 7 (ideal point). The *ideal point* $\mathbf{z}^* = (z_1^*, \ldots, z_m^*)$ represents the minimum possible value in each objective (typically, it cannot be achieved):

$$z_i^* = \min\{f_i(\mathbf{x}); \mathbf{x} \in X\} \quad \forall i \in \{1, \ldots, m\}. \tag{6}$$

Definition 8 (minimal element, minimal set). Let $Z \subseteq F$ be a set of objective vectors. An objective vector $\mathbf{m} \in Z$ is a *minimal element* of Z if

$$\forall \mathbf{z} \in Z, \quad \mathbf{z} \preceq \mathbf{m} \Longrightarrow \mathbf{z} = \mathbf{m}. \tag{7}$$

All minimal elements of Z constitute the *minimal set* of Z denoted in this paper by $Z^{\min}$.

Maximal elements and the maximal set $Z^{\max}$ can be defined dually.

Definition 9 (Pareto front). The set of Pareto optimal objective vectors known as the *Pareto front* can be formally defined as the minimal set of $\mathbf{f}(X)$:

$$P_{\mathbf{f}} = \mathbf{f}(X)^{\min}. \tag{8}$$

Definition 10 (Edgeworth-Pareto hull). Let $Z \subseteq F$ be an approximation set. The Edgeworth-Pareto hull (EPH) of Z includes all objective vectors weakly dominated by Z:

$$Z_{\text{EPH}} = \left\{\mathbf{z} \in F; \text{exists } \mathbf{z}' \in Z \text{ so that } \mathbf{z}' \preceq \mathbf{z}\right\}. \tag{9}$$

2.2. Empirical Attainment Function. Based on the multiobjective concept of goal-attainment [5] we say that an objective vector is *attained* when it is weakly dominated by an approximation set, that is, when it is contained in the set's EPH. The surface delimiting the attained vectors from the ones that have not been attained by an approximation set is called the *attainment surface* and its minimal elements are called the *attainment anchors*.

Definition 11 (set boundary). The boundary of a set $Z \subseteq F$ is the intersection of the closure of Z, $\overline{Z}$, with the closure of its complement, $\overline{(F \setminus Z)}$:

$$\partial Z = \overline{Z} \cap \overline{(F \setminus Z)}. \tag{10}$$

Definition 12 (attainment surface, attainment anchors). Let $Z \subseteq F$ be an approximation set. The *attainment surface* of Z is the boundary of the EPH of Z:

$$S_Z = \partial Z_{\text{EPH}}. \tag{11}$$

The minimal elements of S_Z are called *attainment anchors* and are equal to the original approximation set Z:

$$A_Z = S_Z^{\min} = Z. \tag{12}$$

Now, imagine that an optimization algorithm is run r times, producing r approximation sets. Then, each objective vector can be attained between 0 and r times.

Definition 13 (empirical attainment function). For algorithm A the empirical attainment function of the objective vector $\mathbf{z} \in Z \subseteq F$ represents the frequency of attaining $\mathbf{z}$ by its approximation sets $Z_1^A, \ldots, Z_r^A$:

$$\alpha_r^{\text{A}}(\mathbf{z}) = \alpha\left(Z_1^{\text{A}}, \ldots, Z_r^{\text{A}}; \mathbf{z}\right) = \frac{1}{r}\sum_{i=1}^{r} \mathbf{I}\left(Z_i^{\text{A}} \preceq \{\mathbf{z}\}\right), \tag{13}$$

where $\mathbf{I}$ is the indicator function, defined as

$$\mathbf{I}(b) = \begin{cases} 1, & \text{if } b \text{ is true,} \\ 0, & \text{otherwise,} \end{cases} \tag{14}$$

and $\preceq$ is the weak Pareto dominance relation between sets.

This means that for algorithm A with approximation sets $Z_1^A, \ldots, Z_r^A$ an EAF value from the set of frequencies $\{0, 1/r, 2/r, \ldots, 1\}$ can be assigned to every vector in the objective space. Of course, in practice the EAF cannot be computed for every objective vector and only attainment surfaces with a constant EAF value (and their corresponding attainment anchors) are of interest. They are called k%-*attainment surfaces* (also *summary attainment surfaces*) in order to distinguish them from the "ordinary" attainment surfaces of single approximation sets. Vectors in the k%-attainment surface are the tightest objective vectors that have been attained in at least k% of the runs.

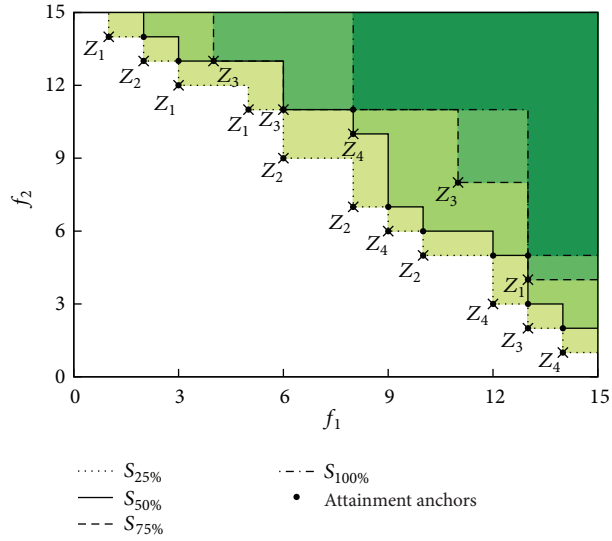


FIGURE 2: Four approximation sets in a two-objective optimization problem, their summary attainment surfaces, and all attainment anchors. Colors denote areas with equal EAF values (darker colors correspond to larger values).

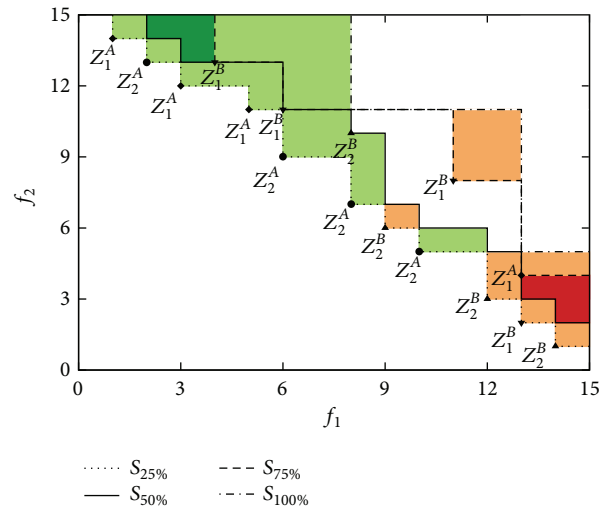


FIGURE 3: Four approximation sets in a two-objective optimization problem: two produced by algorithm A and two by algorithm B. Colored areas show where algorithm A outperformed algorithm B (green hues) and algorithm B outperformed algorithm A (red hues).

Definition 14 (summary attainment surfaces). Let $Z_1, \ldots, Z_r$ be r approximation sets of algorithm A and $\alpha_r^A(\mathbf{z})$ its EAF. Let the t/r-superlevel set of α_r^A be defined as

$$L_{t/r}^{+}\left(\alpha_r^{\mathrm{A}}\right) = \left\{\mathbf{z} \in F; \alpha_r^{\mathrm{A}}(\mathbf{z}) \geq t/r\right\} \quad \forall t \in \{1, \ldots, r\}. \tag{15}$$

Then the *summary* (or t/r-) *attainment surface* $S_{t/r}$ is equal to the boundary of $L_{t/r}^{+}(\alpha_r^{\mathrm{A}})$:

$$S_{t/r} = \partial L_{t/r}^{+}\left(\alpha_r^{\mathrm{A}}\right) \quad \forall t \in \{1, \ldots, r\}. \tag{16}$$

We will use $A_{t/r}$ to denote the attainment anchors of $S_{t/r}$:

$$A_{t/r} = S_{t/r}^{\min} \quad \forall t \in \{1, \ldots, r\}. \tag{17}$$

Note how the definition of an attainment surface coincides with the definition of a summary attainment surface if $r = 1$. In general, the number of all attainment anchors (including duplicates), $n_A = \sum_{t=1}^{r} |A_{t/r}|$, is much larger than the number of vectors contained in the approximation sets, $n_Z = \sum_{t=1}^{r} |Z_t|$ (in the context of the EAF computation [10], vectors from the approximation sets and attainment anchors were called *input* and *output points*, resp.). It has been shown in [10] that $n_A \in O(r^2 n_Z)$ for three-objective optimization problems.

We illustrate these concepts in the two-objective case. Figure 2 shows an example of four approximation sets $Z_1, \ldots, Z_4$ (each consisted of four objective vectors indicated with crosses) and their summary attainment surfaces. Dots denote all attainment anchors and colors are used to emphasize areas with equal EAF values (darker colors correspond to higher values).

If two algorithms A and B are run r times each, they can be compared by computing and visualizing their EAF values α_r^A and α_r^B separately. However, there exists a straightforward way to directly compare the algorithms by computing *EAF differences*, that is, differences in their EAF values [11].

Definition 15 (EAF differences). Assume that algorithms A and B are run r times each. The EAF differences between algorithms A and B are defined for each objective vector $\mathbf{z} \in Z$ as

$$\delta_r^{A-B}(\mathbf{z}) = \alpha_r^A(\mathbf{z}) - \alpha_r^B(\mathbf{z}). \tag{18}$$

Note that EAF differences need to be computed for all attainment anchors of the combined $2r$ approximation sets. Defined in this way, the differences can adopt values from the set $\{-1, -(r-1)/r, \ldots, 0, 1/r, \ldots, 1\}$. Positive EAF differences correspond to areas in the objective space where the algorithm A outperforms the algorithm B, while negative EAF differences denote areas in the objective space where the algorithm B outperforms the algorithm A. Naturally, where the differences are zero, both algorithms attain the area equally well.

Let us focus on the example from Figure 3, where algorithms A and B solving a two-objective optimization problem produced two approximation sets each. The lines show the overall summary attainment surfaces. The colored areas emphasize areas of the objective space where algorithm A outperformed algorithm B (green hues) and areas where algorithm B outperformed algorithm A (red hues). We can readily notice that algorithm A is more successful in optimizing the first objective, while algorithm B attains better lower values of the second objective. This small example nevertheless shows that visualization of 2D EAF differences can be very helpful when analyzing and comparing the results of two algorithms.

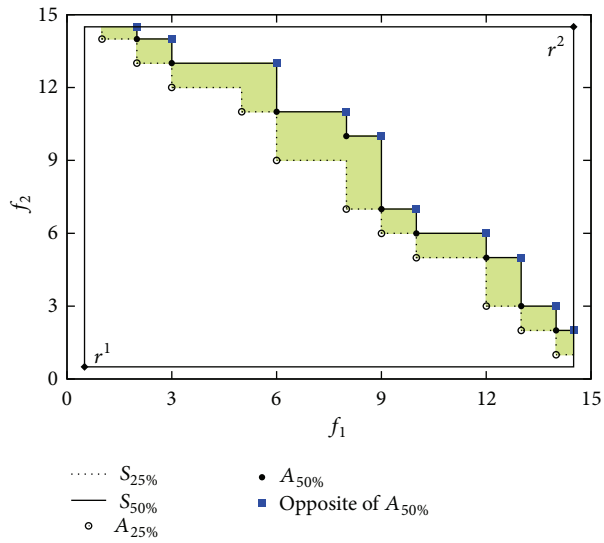


FIGURE 4: The area between the 2D 25%- and 50%-attainment surfaces from the example in Figure 2. Vectors from the opposite of the attainment anchors of the 50%-attainment surface are denoted by squares.

We can safely assume that we are interested in a limited portion of the objective space, for example, the one enclosed by reference vectors $\mathbf{r}^1$ and $\mathbf{r}^2$, where $\mathbf{r}^1 \ll \mathbf{r}^2$. Then, areas with equal EAF values/differences correspond to unions of rectangles in 2D and unions of cuboids in 3D [7]. If we allow overlapping, then each rectangle or cuboid between the t/r- and $(t+1)/r$-attainment surfaces, where $t \in \{1, \ldots, r-1\}$, can be defined by two vertices: one from the t/r-attainment surface and the other from the $(t+1)/r$-attainment surface (Figure 4 shows 2D areas between 25%- and 50%-attainment surfaces). While the lower vertices correspond to the attainment anchors of the t/r-attainment surface, the upper vertices are not the attainment anchors of the $(t+1)/r$-attainment surface but their *opposite*. The opposite can be defined for an arbitrary approximation set as follows.

Definition 16 (approximation set opposite). Let reference vectors $\mathbf{r}^1$ and $\mathbf{r}^2$, where $\mathbf{r}^1 \ll \mathbf{r}^2$, define the boundaries of the observed objective space

$$R = \left\{\mathbf{z} \in F; z_i \in \left[r_i^1, r_i^2\right] \ \forall i \in \{1, \ldots, m\}\right\}. \quad (19)$$

Let $Z \subseteq R$ be an approximation set enclosed in this space with the attainment surface S_Z. The *opposite* O_Z^R of the approximation set Z within the observed objective space R is the maximal set of $S_Z \cap R$:

$$O_Z^R = \left(S_Z \cap R\right)^{\max}. \quad (20)$$

Finding the opposites of attainment anchors in 2D is rather trivial [7]. The computation of opposites (and the corresponding cuboids) for the 3D case is more demanding and is presented in detail in Section 4.1.

Finally, let us recall the formal definition of the *hypervolume indicator* [12], which is often used to measure the quality of approximation sets.

Definition 17 (hypervolume indicator). Let again reference vectors $\mathbf{r}^1$ and $\mathbf{r}^2$, where $\mathbf{r}^1 \ll \mathbf{r}^2$, define the boundaries of the observed objective space

$$R = \left\{\mathbf{z} \in F; z_i \in \left[r_i^1, r_i^2\right] \ \forall i \in \{1, \ldots, m\}\right\}. \quad (21)$$

The *hypervolume indicator* I_H of an approximation set $Z \subseteq R$ can be formulated via the empirical attainment function of Z as

$$I_H(Z) = \int_{\mathbf{z} \in R} \alpha(Z; \mathbf{z})\, d\mathbf{z}. \quad (22)$$

2.3. Visualization of EAFs. 2D EAFs are most often visualized through *best*, *median*, and *worst* summary attainment surfaces corresponding to $1/r$%, ~50%, and 100%-attainment surfaces, respectively. The first attempt to visualize 3D summary attainment surfaces is documented in [13]. At the time, an efficient algorithm for computing 3D attainment anchors was not yet available, so separate summary attainment surfaces were approximated by discretizing the objective space using a grid. Assume the t/r-attainment surface needs to be visualized. The algorithm examines the objective space for each objective separately. First, it finds the intersections between all approximation sets and the lines stemming from the 2D grid of the remaining two objectives. Then, the intersections on each of these lines are counted and if they amount to t, a marker is drawn at the corresponding height. Combining these markers by considering all objectives yields informative visualization of the chosen 3D summary attainment surface. Years later, an efficient algorithm for computing attainment anchors and their EAF values for the 3D case was finally provided [10, 14]. The attainment function tools from [14] were used to compute all EAF values in this paper.

While [10] offers an estimation of the number of attainment anchors also for the general mD case, where $m > 3$, no algorithm for computing 4D EAFs exists yet. As a consequence, we do not discuss visualization of EAFs beyond 3D.

Recently, the idea of visualizing EAF differences in addition to EAF values was introduced in [11]. When visually comparing algorithms A and B, four plots can be produced, each corresponding to one of the values of α^A, α^B, δ^{A-B}, and δ^{B-A}. All attainment anchors are plotted using gray-shaded dots, where the shade corresponds to the value of the chosen function (α^A, α^B, δ^{A-B}, or δ^{B-A}). On top of this, the best, median, and worst overall summary attainment surfaces are drawn. Examples from [11, 15] show that this kind of visualization can be very useful in exploratory analysis.

Building on this work we have performed some initial research on visualizing 3D EAFs using slicing and maximum intensity projection in [6, 7]. These two methods, followed by direct volume rendering, are introduced next.

2.4. Slicing. This is a simple method where spatial data is sliced using a plane. Then, the intersection of the plane with the data is visualized in 2D. In its most general form, slicing can be applied to any kind of spatial data; that is, the data does not have to be represented by voxels.

While the plane used in slicing is most often axis-aligned, the nature of multiobjective optimization problems yields the need for a plane that always contains one axis and the origin of the objective space (which usually corresponds to the ideal point) and intersects all summary attainment surfaces. Therefore (similarly to the *prosection plane* used in [4, 16]), the slicing plane cuts the objective space at an angle $\varphi \in (0^\circ, 90^\circ)$.

While not used for visualizing attainment surfaces, the interactive decision maps (IDMs) are related to slicing using axis-aligned planes [17]. They visualize a number of axis-aligned sampling surfaces of the EPH and use different colors to represent each slice. As the surface of Z_{EPH} is exactly the attainment surface of Z, it should be rather straightforward to apply this method for visualizing attainment surfaces.

2.5. Maximum Intensity Projection. Maximum intensity projection (MIP) is a volume rendering method for spatial data represented by voxels. The method inspects voxels in direction of parallel rays traced from the viewpoint to the projection plane and takes the maximum value encountered in the voxels along a ray as the projection value for the ray.

The method, originally called maximum activity projection (MAP) [8], was proposed for 3D image rendering in nuclear medicine and tested in tomographic studies. It was later accepted not only in medical imaging, but also in scientific data visualization in general.

The advantages of MIP are its simplicity and efficiency and the ability of achieving high contrast, which arises from the fact that maximum voxel values are projected. On the other hand, as a limitation, the resulting projections lack the sense of depth of the original data (see [18, 19] for attempts to remedy this issue). Moreover, the viewer cannot distinguish between left and right and front and back. As an improvement, animations are usually provided, consisting of a sequence of MIP renderings at slightly different viewpoints, which results in the illusion of rotation.

2.6. Direct Volume Rendering. Direct volume rendering (DVR) [20] is another volume rendering method based on the voxel representation of data that is often used for visualization in medicine and science. It can generate very appealing results by employing advanced shading techniques and can achieve real-time interactive rendering. DVR is able to produce images of volumetric data without the need to explicitly extract geometry or surfaces (hence the name *direct*).

First, each voxel value is assigned color and opacity (the RGBA value) by means of a chosen *transfer function*. The transfer function can be a simple ramp, a piecewise linear function, or an arbitrary table (the problem of specifying a good transfer function is a research area on its own [21, 22]). Then, one of the following techniques is used to project the RGBA values to the screen: ray casting (for only ray casting is used in this paper, any mention of DVR can be understood to refer to ray-casting DVR from now on), splatting, shear-warp factorization, or texture-based volume rendering. In ray casting [9], viewing rays are traced from the viewpoint through the volume, accumulating color and opacity values at each sample position along the ray. The final RGB values shown in the image are computed from the accumulated RGBA values using the volume rendering integral [23].

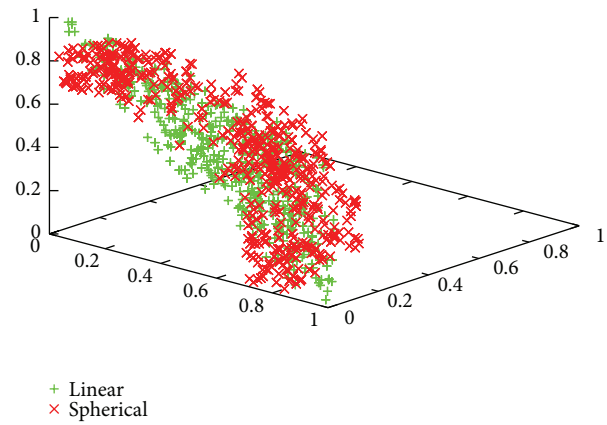


Figure 5: The linear and spherical approximation sets used in all visualization examples in Sections 4 and 5.

3. Benchmark Approximation Sets

In order to show the outcome of different visualization methods, we need some approximation sets to visualize. The two sets of approximation sets used in this paper are not products of optimization algorithms but rather benchmark approximation sets with known properties used for visualization purposes [7]. They are shown in Figure 5.

The first set consists of five *linear* approximation sets with a uniform distribution of vectors that satisfy the constraint

$$\sum_{i=1}^{3} z_i = c_L. \tag{23}$$

In order to simulate the behavior of a stochastic algorithm, the values c_L follow the normal distribution $N(1, 0.05)$. The second set contains five *spherical* approximation sets with a nonuniform distribution of vectors where only few vectors are located in the middle of the approximation set, while most of them are near its corners. The vectors from the spherical approximation sets satisfy the constraint

$$\sum_{i=1}^{3} z_i^2 = c_S^2, \tag{24}$$

where $c_S \sim N(0.75, 0.05)$. Each individual approximation set contains 100 objective vectors.

The values of c_L and c_S were chosen so that the sets are intertwined; in one region, the linear approximation sets dominate the spherical ones, while in others, the spherical sets dominate the linear ones. This assures that there will always be visible EAF differences between the two sets of approximation sets.

Input: Sets of attainment anchors $A_1, \ldots, A_r$ and a reference vector $\mathbf{r}^2$, for which $A_1 \preceq \cdots \preceq A_r$ and $\mathbf{z} \ll \mathbf{r}^2$ for all $\mathbf{z} \in A_i, i = 1, \ldots, r$
Output: The set of cuboids C
$A_{r+1} \leftarrow \{\mathbf{r}^2\}$;
$\mathbf{r}^1 \leftarrow \mathbf{z}^*$;
foreach adjacent pair of sets A_i and $A_{i+1}, i \in \{1, \ldots, r\}$ **do**
  $O \leftarrow \text{opposite}(A_{i+1}, \mathbf{r}^1, \mathbf{r}^2)$;
  foreach $\mathbf{z} \in A_i$ **do**
    $O' \leftarrow \{\mathbf{o} \in O; \mathbf{z} \ll \mathbf{o}\}$;
    foreach $\mathbf{o} \in O'$ **do**
      $C \leftarrow C \cup \text{cuboid}(\mathbf{z}, \mathbf{o}, \text{value}(\mathbf{z}))$;
    end
  end
end

Algorithm 1: Computing the cuboids.

Input: Approximation set A and reference vectors $\mathbf{r}^1$ and $\mathbf{r}^2$, for which $\mathbf{r}^1 \preceq \mathbf{z} \ll \mathbf{r}^2$ for all $\mathbf{z} \in A_i, i = 1, \ldots, r$
Output: The opposite O
$O \leftarrow \{\mathbf{r}^2\}$;
foreach $\mathbf{z} \in A$ **do**
  $O' \leftarrow \{\mathbf{o} \in O; \mathbf{z} \ll \mathbf{o}\}$;
  $O \leftarrow O - O'$;
  foreach $\mathbf{o} \in O'$ **do**
    $\mathbf{o}^{n_1} \leftarrow (z_1, o_2, o_3)$;
    $\mathbf{o}^{n_2} \leftarrow (o_1, z_2, o_3)$;
    $\mathbf{o}^{n_3} \leftarrow (o_1, o_2, z_3)$;
    for $i \leftarrow 1$ **to** 3 **do**
      if $\neg$ redundant $(\mathbf{o}^{n_i}, \mathbf{r}^1, O)$ **then**
        $O \leftarrow O \cup \{\mathbf{o}^{n_i}\}$;
      end
    end
  end
end

Algorithm 2: Computing the opposite of a 3D approximation set.

We will use α_5^L and α_5^S to denote EAF values of the linear and spherical approximation sets, respectively. In addition, δ_5^{L-S} and δ_5^{S-L} will denote differences between those values.

4. Visualizing Exact EAFs

Exact EAFs can be represented as unions of cuboids with values corresponding to either EAF values of one algorithm or EAF differences between two algorithms. This section will first show the procedure for computing the cuboids from the given attainment surfaces and then present visualization of these cuboids using slicing and MIP.

4.1. Computing the Cuboids. The algorithm for computing EAFs [10] takes as input a series of approximation sets $Z_1, \ldots, Z_r$ and returns as output a series of sets containing attainment anchors $A_1, \ldots, A_r$ that define the corresponding summary attainment surfaces and for which the following holds: $A_i \preceq A_{i+1}$ for $i \in \{1, 2, \ldots, r-1\}$. From these sets of attainment anchors and a reference vector $\mathbf{r}^2$ that limits the observed objective space, the cuboids can be computed. We propose a very simple algorithm for this purpose that uses the first set of attainment anchors and the opposite of the second set of attainment anchors to compute overlapping cuboids (see Algorithm 1).

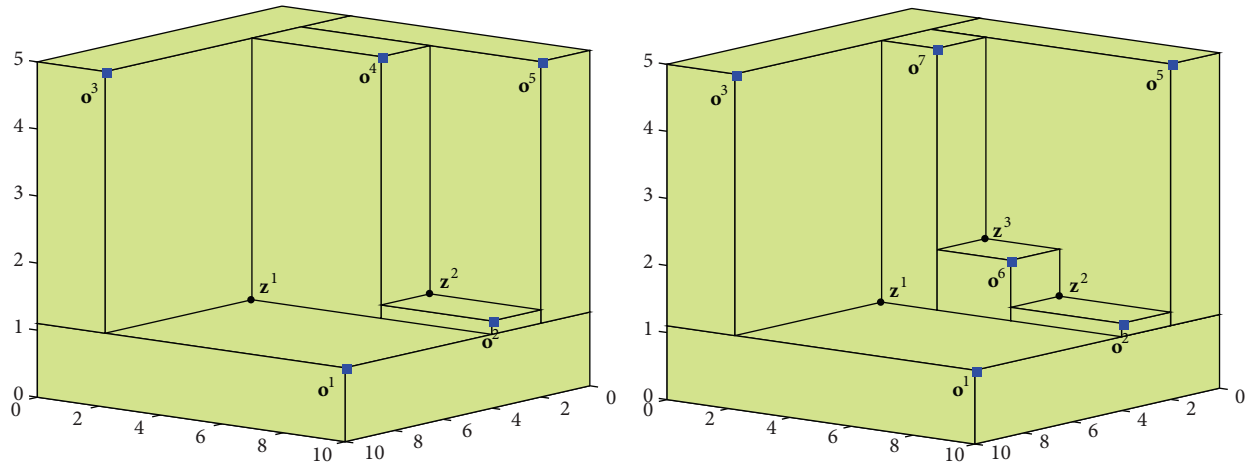
First, a set containing only the given reference vector $\mathbf{r}^2$ is added at the end of the input set of attainment anchors, so that any cuboids ranging from the last set of attainment anchors to $\mathbf{r}^2$ can also be computed. Then, the reference vector $\mathbf{r}^1$ needed for computation of the opposites is set to be equal to the ideal point. Next, the main loop iterates through all adjacent pairs of sets A_i and A_{i+1} and computes the opposite of A_{i+1} using Algorithm 2. For each attainment anchor $\mathbf{z} \in A_i$ all dominated vectors from the opposite are stored in O'. Each pair $(\mathbf{z}, \mathbf{o})$, where $\mathbf{o} \in O'$, represents the two vertices required to define the cuboid. In addition, the cuboid is given a value: either the EAF value or the EAF difference in $\mathbf{z}$.

Note that there might be multiple cuboids stemming from the same attainment anchor (see the analog example of multiple rectangles in Figure 4)—we call this a *union of cuboids*. This is not a problem since all such overlapping cuboids have the same value (if they did not have the same value, another attainment anchor would have already split the cuboid). The result of Algorithm 1 is a set of cuboids (or unions of cuboids).

Next, we show with the help of Figure 6 how to find the opposite of a 3D approximation set. Imagine a single cuboid defined by the reference vectors $\mathbf{r}^1$ and $\mathbf{r}^2$. The opposite is initialized to $\{\mathbf{r}^2\}$. Every time a vector from Z is "cut into" the existing cuboid, the vectors strictly dominated by it are removed from the opposite and new vectors are added to the opposite. Assume we have already performed this step for vectors $\mathbf{z}^1$ and $\mathbf{z}^2$ (see Figure 6(a)) and vector $\mathbf{z}^3$ is next in line (see Figure 6(b)). First, we delete from the opposite all vectors that are strictly dominated by the current vector $\mathbf{z}$ (this means that in our example we delete vector $\mathbf{o}^4$ but not $\mathbf{o}^5$). Next, for every deleted vector $\mathbf{o}$ we create three new vectors in the following way:

$$\begin{aligned} \mathbf{o}^{n_1} &= (z_1, o_2, o_3), \\ \mathbf{o}^{n_2} &= (o_1, z_2, o_3), \\ \mathbf{o}^{n_3} &= (o_1, o_2, z_3). \end{aligned} \tag{25}$$

Finally, each of these vectors is added to the opposite if it is not redundant. This means that it has to contribute to the opposite (it cannot be coplanar with $\mathbf{r}^1$ or collinear with any of the vectors from the existing opposite). In our example, we add to the opposite vectors $\mathbf{o}^6$ and $\mathbf{o}^7$ but not $\mathbf{o}^8 = (2, 6.5, 5)$ (not pictured in the figure) as it is collinear with $\mathbf{o}^5$ and thus redundant. When these steps have been taken for every vector from Z, the resulting set O represents the opposite of Z. See Algorithm 2 for the algorithmic notation of the described procedure.



(a) Opposite of $\{\mathbf{z}^1, \mathbf{z}^2\}$ (b) Opposite of $\{\mathbf{z}^1, \mathbf{z}^2, \mathbf{z}^3\}$

FIGURE 6: Two steps in the construction of the opposite of the approximation set Z.

As we cannot visualize the entire (infinite) objective space, Algorithms 1 and 2 use two reference vectors $\mathbf{r}^1$ and $\mathbf{r}^2$ to limit it. In order to preserve all information, they need to be set in such a way that $\mathbf{r}^1 \preceq \mathbf{z} \ll \mathbf{r}^2$ for all attainment anchors $\mathbf{z}$. While the ideal point is the most sensible choice for $\mathbf{r}^1$, we can choose any objective vector dominated by all attainment anchors for $\mathbf{r}^2$.

The presented approach for computing the cuboids is very simple and easy to implement but comes at high computational cost. Computing the cuboids between two sets of n attainment anchors has the worst-case computational complexity of $O(n^3)$, which means that all cuboids between r pairs of attainment surfaces can be computed in $O(rn^3)$ time. However, in practice, the loop **foreach** $\mathbf{o} \in O'$ **do** in Algorithm 2 is executed only a few times for each $\mathbf{z} \in A$, meaning that in practice the computational complexity is quadratic in the attainment anchors set size (the check for redundancy still takes linear time).

Another possible approach would be to deal only with nonoverlapping cuboids, which is somewhat similar to the problem of computing the hypervolume indicator in 3D. In future work we wish to investigate if a more efficient algorithm for our problem could be found by, for example, modifying the space-sweep algorithm from [24], which computes the hypervolume indicator in the 3D case with the computational complexity of only $O(n \log n)$. If this would be possible, the total cost of computing the cuboids would be $O(rn \log n)$. However, time complexity is not the only important aspect in the computation of cuboids. We would also like to study which method would yield a lower number of cuboids as this considerably affects the tractability of their visualization.

4.2. Visualizing Cuboids Using Slicing. In this paper, the slicing plane is aligned with one of the three axes (assume for now that this is f_3), includes the origin $\mathbf{r}^1$, and cuts through the underlying plane $(f_1 f_2)$ at some angle $\varphi \in (0°, 90°)$. In this way, each slice always captures all attainment surfaces.

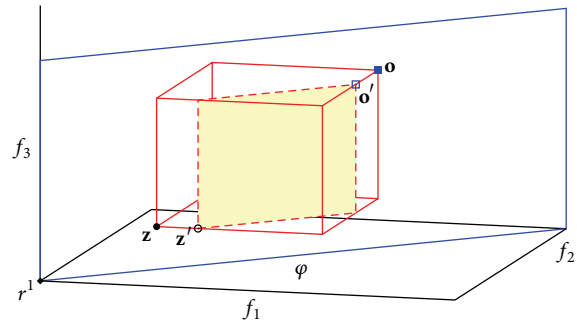


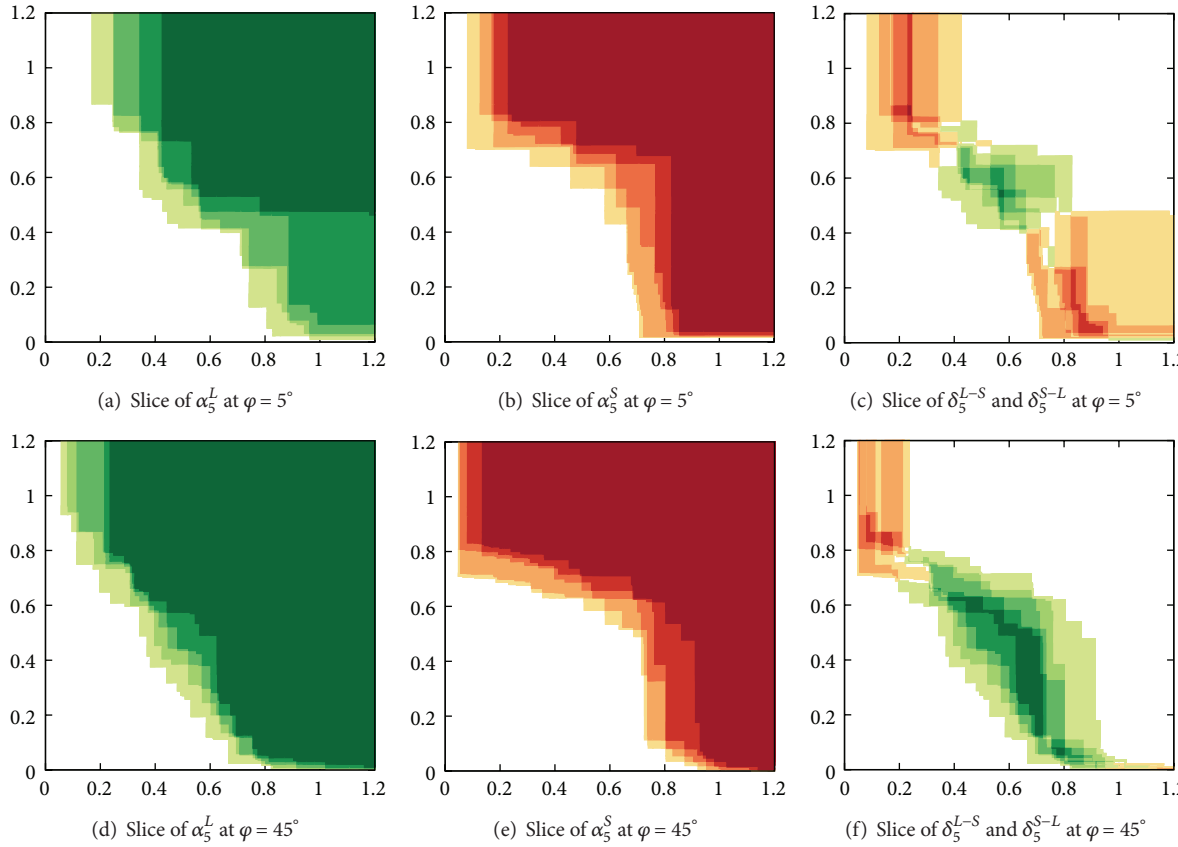
FIGURE 7: A cuboid sliced using the plane aligned with f_3 that slices the $f_1 f_2$ plane under angle. The cuboid vertices $\mathbf{z}$ and $\mathbf{o}$ are projected to 3D rectangle vertexes $\mathbf{z}'$ and $\mathbf{o}'$, respectively.

Slicing through the objective space containing a large number of cuboids means that only those cuboids that intersect the slicing plane are visualized. When a cuboid is sliced, the result is a rectangle in 3D (see the example in Figure 7). Two steps are needed to compute this rectangle in 2D. First, we need to calculate the projected cuboid vertices $\mathbf{z}'$ and $\mathbf{o}'$ yielding the rectangle in 3D, and second, we need to rotate the vertices by angle $-\varphi$ to get $\mathbf{z}''$ and $\mathbf{o}''$. Depending on the angles

$$\varphi_{\mathbf{z}} = \arctan\left(\frac{z_2 - r_2^1}{z_1 - r_1^1}\right), \qquad \varphi_{\mathbf{o}} = \arctan\left(\frac{o_2 - r_2^1}{o_1 - r_1^1}\right), \tag{26}$$

this is done in the following way:

$$\mathbf{z}' = \begin{cases} \left(r_1^1 + \dfrac{z_2 - r_2^1}{\tan\varphi}, z_2, z_3\right), & \text{if } \varphi_{\mathbf{z}} \geq \varphi, \\ \left(z_1, r_2^1 + \left(z_1 - r_1^1\right)\tan\varphi, z_3\right), & \text{if } \varphi_{\mathbf{z}} < \varphi, \end{cases}$$



(a) Slice of α_5^L at $\varphi = 5^\circ$ (b) Slice of α_5^S at $\varphi = 5^\circ$ (c) Slice of δ_5^{L-S} and δ_5^{S-L} at $\varphi = 5^\circ$

(d) Slice of α_5^L at $\varphi = 45^\circ$ (e) Slice of α_5^S at $\varphi = 45^\circ$ (f) Slice of δ_5^{L-S} and δ_5^{S-L} at $\varphi = 45^\circ$

FIGURE 8: Slices of the exact 3D EAF values and differences for the benchmark approximation sets under two angles. Darker colors represent higher EAF values/differences.

$$\mathbf{o}' = \begin{cases} \left(o_1, r_2^1 + \left(o_1 - r_1^1\right)\tan\varphi, o_3\right), & \text{if } \varphi_{\mathbf{o}} \geq \varphi, \\ \left(r_1^1 + \dfrac{o_2 - r_2^1}{\tan\varphi}, o_2, o_3\right), & \text{if } \varphi_{\mathbf{o}} < \varphi, \end{cases}$$

$$\mathbf{z}'' = \begin{cases} \left(r_1^1 + \dfrac{z_2 - r_2^1}{\sin\varphi}, z_3\right), & \text{if } \varphi_{\mathbf{z}} \geq \varphi, \\ \left(r_1^1 + \dfrac{z_1 - r_1^1}{\cos\varphi}, z_3\right), & \text{if } \varphi_{\mathbf{z}} < \varphi, \end{cases}$$

$$\mathbf{o}'' = \begin{cases} \left(r_1^1 + \dfrac{o_1 - r_1^1}{\cos\varphi}, o_3\right), & \text{if } \varphi_{\mathbf{o}} \geq \varphi, \\ \left(r_1^1 + \dfrac{o_2 - r_2^1}{\sin\varphi}, o_3\right), & \text{if } \varphi_{\mathbf{o}} < \varphi. \end{cases} \tag{27}$$

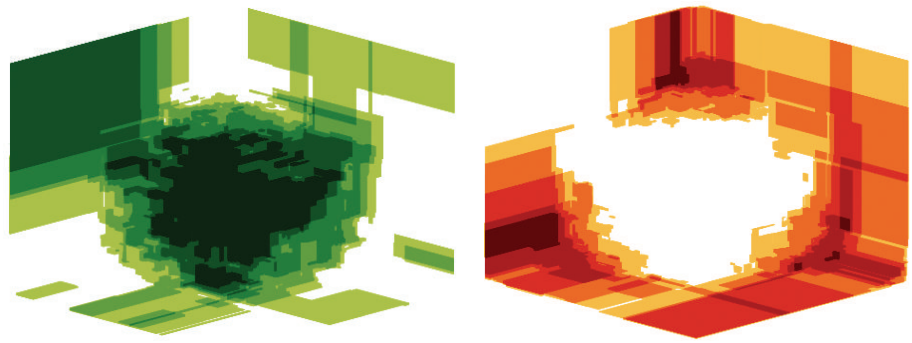
When slicing through a union of cuboids stemming from the same vertex $\mathbf{z}$, some of the projected vertices $\mathbf{o}''$ can be redundant. In order to decrease the number of total rectangles to plot, it is therefore sensible to keep only nonredundant vertices $\mathbf{o}''$ (those that are nondominated with regard to the inverted weak dominance relation $\succeq$).

The cuboids can be visualized by slicing them at different angles, thus showing different parts of the objective space. We present visualization using slicing of our benchmark approximation sets at angles $\varphi = 5^\circ$ and $\varphi = 45^\circ$. The EAF values α_5^L and α_5^S are shown separately, while the EAF differences δ_5^{L-S} and δ_5^{S-L} are presented on the same plot (see Figure 8).

From the plots of EAF values it is easy to distinguish linear sets (green hues) from the spherical ones (red hues) as the shape of the sets is readily visible. We can also see that the two best spherical approximation sets are better than the remaining three by a solid margin. While these plots provide a lot of information by themselves, the comparison between the sets is best visualized using EAF differences. They nicely show regions in the objective space where linear sets outperform the spherical ones and vice versa.

Note that in order to fully explore the entire objective space, slicing planes at several different angles should be performed. Also, one might want to consider slicing planes aligned to the axis f_1 or f_2, too. However, as our benchmark approximation sets are rather symmetrical, there is no need to do this here.

4.3. Visualizing Cuboids Using MIP. As explained in Section 2.5, MIP shows only the maximum value encountered on rays from the viewpoint to the projection plane. This



(a) MIP of δ_5^{L-S} (b) MIP of δ_5^{S-L}

Figure 9: MIP of exact 3D EAF differences between the benchmark approximation sets.

means that it is not sensible to use this method for visualizing EAF values as most of the objective space has the maximum value and the resulting visualization would not be very informative. However, MIP seems a good choice for visualizing EAF differences where the highest values are of most interest as they represent the largest differences between the algorithms.

MIP could be used to visualize cuboids of different values by sorting the cuboids so that those with higher values would be put on top of those with lower values. However in general, the 3D plotting tools capable of visualizing cuboids render them in a sequence that tries to maintain some notion of depth (cuboids near the viewpoint are shown in front of the cuboids further away) and do not allow for custom sorting of the cuboids according to their values. Therefore, this sorting can only be done after the viewpoint has been set and the cuboids are already projected to 2D. At that time we can choose to visualize them in the order of ascending values, which (although a bit cumbersome) effectively achieves the MIP visualization of the cuboids.

Another difficulty in using MIP for visualizing exact EAF differences is that we need to visualize a large number of cuboids, which can be challenging to do. While many of them are completely covered by cuboids with larger values and could therefore be spared without altering the final image, removal of such cuboids is not trivial as it depends also on the position of the viewpoint.

Nevertheless, we present visualization of exact EAF differences using MIP in Figure 9. The differences δ_5^{L-S} and δ_5^{S-L} need to be visualized separately in order to avoid overlapping of cuboids. The MIP visualizations are very informative; we can easily see which parts of the objective space are better attained by the linear approximation sets and which by the spherical approximation sets. As is typical with MIP, while we are able to "see through the cuboids," we loose the sense of depth. Although it is inevitable to lose some information when projecting 3D structures onto 2D, this can be amended by combining two visualization techniques: MIP and slicing. Together they give a good idea of the 3D "cloud" of cuboids.

5. Visualizing Approximated EAFs

If we wish to visually compare the outcome of two algorithms and are not interested in the details of such a visualization, we can use approximated instead of exact EAFs. Approximation means that the objective space is discretized into a grid of voxels (similarly to what was proposed in [13]). This section first presents how such discretization is performed and then illustrates visualization of approximated EAFs using slicing, MIP, and DVR.

5.1. Discretization into Voxels. A voxel is a single point on a regularly spaced 3D grid of $v_1 \times v_2 \times v_3$ voxels. Recall that, based on the reference vectors $\mathbf{r}^1$ and $\mathbf{r}^2$, the 3D observed objective space R is defined as

$$R = \left\{\mathbf{z} \in F; z_i \in \left[r_i^1, r_i^2\right] \quad \forall i \in \{1,2,3\}\right\}. \tag{28}$$

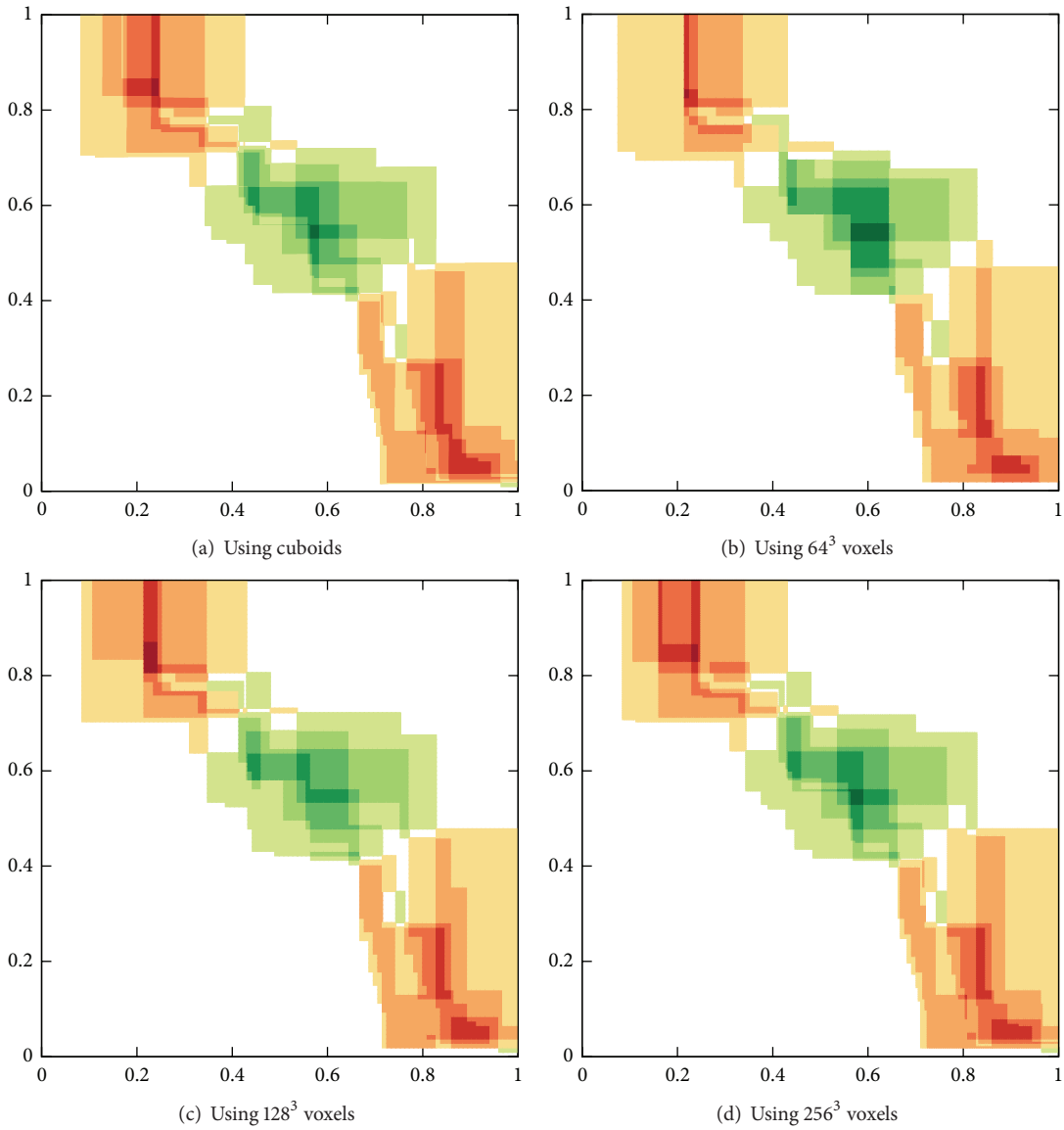
If R is a cube, $v_1 = v_2 = v_3$; otherwise, some care must be taken to ensure that the grid of voxels is truly regular. Either R must be normalized prior to approximation or the number of voxels in each dimension v_i must be set to be proportional to $r_i^2 - r_i^1$ for all $i \in \{1,2,3\}$. Note that a voxel represents only a single point not a volume. How this missing information is reconstructed depends on the visualization method.

From the observed objective space R the grid of $v_1 \times v_2 \times v_3$ voxels is constructed so that the voxel at grid position (k_1, k_2, k_3) has the following coordinates:

$$\left(\frac{\left(r_1^2 - r_1^1\right)(2k_1 - 1)}{2v_1}, \frac{\left(r_2^2 - r_2^1\right)(2k_2 - 1)}{2v_2}, \frac{\left(r_3^2 - r_3^1\right)(2k_3 - 1)}{2v_3}\right), \tag{29}$$

where $k_i \in \{1, \ldots, v_i\}$ for $i \in \{1,2,3\}$.

Computing voxel values when the cuboids have already been computed is rather straightforward. Iterating over all cuboids, the voxels that are "covered" by the cuboid adopt its value. If we are not interested in visualizing the exact EAFs (and therefore have not computed the cuboids), we can compute the voxel values directly from the set of approximation sets. The value of each voxel is set to either EAF value or EAF difference by counting how many approximation sets weakly dominate it.



(a) Using cuboids

(b) Using 64^3 voxels

(c) Using 128^3 voxels

(d) Using 256^3 voxels

FIGURE 10: Slices of exact and approximated δ_5^{L-S} and δ_5^{S-L} at angle $\varphi = 5^\circ$ showing the approximation error.

5.2. Visualizing Voxels Using Slicing. For visualization using slicing we assume that the whole volume of the voxel has the same value as its center. Therefore, slicing of voxels is done in the same way as slicing of cuboids (see Section 4.2). In order to avoid showing plots that are very similar to those presented in Figure 8, Figure 10 shows only the slices of δ_5^{L-S} and δ_5^{S-L} at angle $\varphi = 5^\circ$ produced using different discretizations.

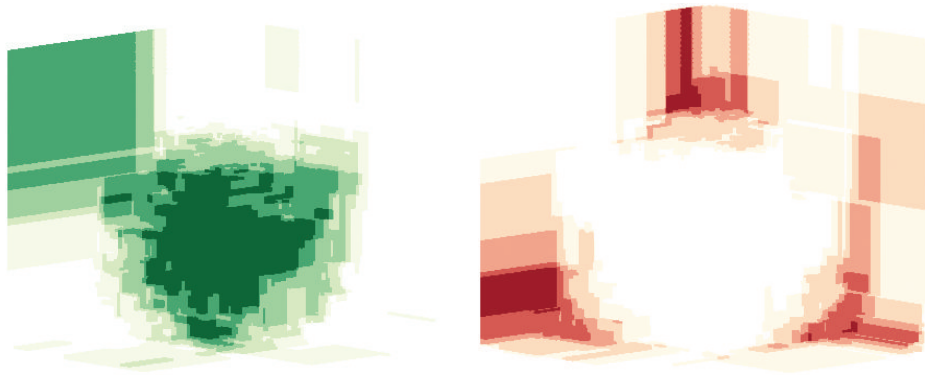
We can see that by refining the discretization we get results that are increasingly similar to the exact EAFs. As we believe that for the purpose of this study the discretization into 128^3 voxels suffices, we will use this discretization in the remainder of the paper.

5.3. Visualizing Voxels Using MIP. Visualizing voxels using MIP is trivial with a tool supporting such visualization, as there are no additional parameters to set. We use a volume rendering engine called Voreen [25, 26] to produce the MIP images from Figure 11 and all DVR images.

Figure 11 shows the MIP for δ_5^{L-S} and δ_5^{S-L} and we can see that although these are approximations of the images presented in Figure 9, the results are quite similar.

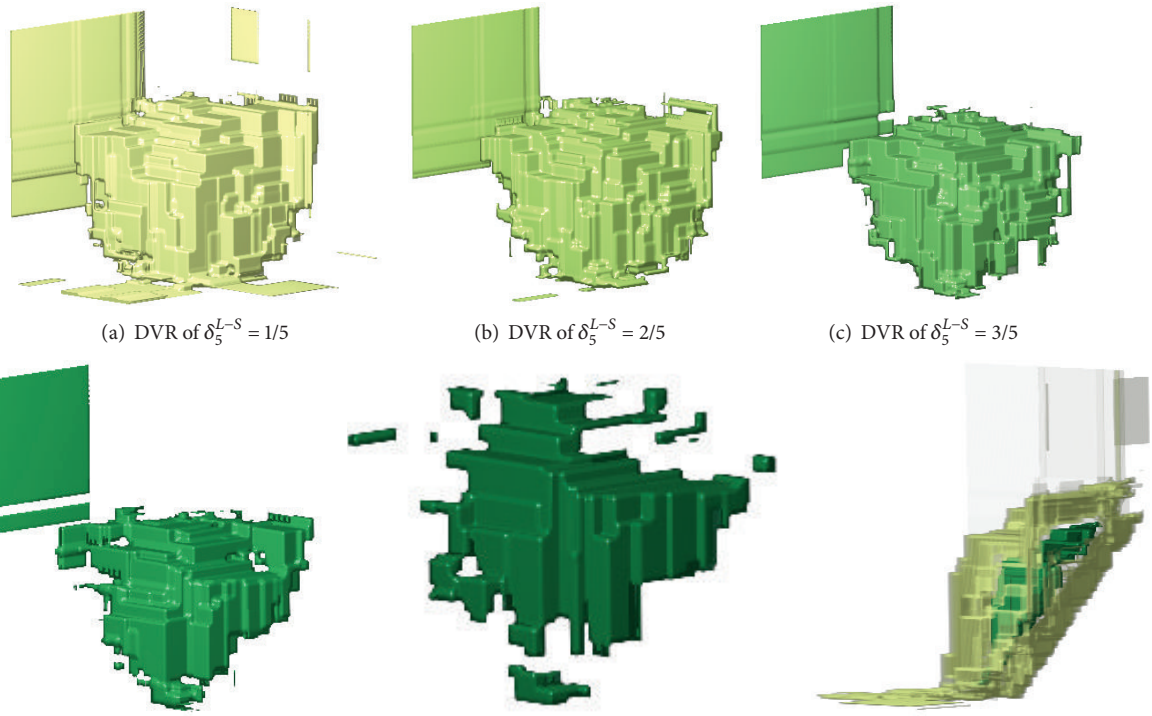
5.4. Visualizing Voxels Using DVR. Visualization using DVR is a bit trickier as we need to define the transfer function that assigns color and opacity to each voxel value. In our case, voxel values are discrete as they equal either the EAF values or the EAF differences, which makes this task easier.

Figures 12 and 13 show visualization using DVR of the EAF differences between the two benchmark approximation sets δ_5^{L-S} and δ_5^{S-L}, respectively. The first five plots in both figures show the voxels with a single value from $\{1/5, \ldots, 5/5\}$. The transfer functions used to obtain these plots are simple piecewise linear functions that set the opacity of voxels of the desired value to 1 and the opacity of all other voxels to 0. In this way, we are able to visualize each



(a) MIP of δ_5^{L-S} (b) MIP of δ_5^{S-L}

Figure 11: MIP of approximated 3D EAF differences between the benchmark approximation sets.



(a) DVR of $\delta_5^{L-S} = 1/5$ (b) DVR of $\delta_5^{L-S} = 2/5$ (c) DVR of $\delta_5^{L-S} = 3/5$

(d) DVR of $\delta_5^{L-S} = 4/5$ (e) DVR of $\delta_5^{L-S} = 5/5$ (f) DVR of $\delta_5^{L-S} \in \{1/5, 5/5\}$

Figure 12: DVR of approximated 3D EAF differences between the linear and spherical benchmark approximation sets δ_5^{L-S}.

of the values separately, which is reflected in the nice and informative plots in Figures 12 and 13.

The biggest advantage of DVR is that, by setting the transfer function "the right way," it is possible to see inside the visualized volume. This was attempted on the plots shown in Figures 12(f) and 13(f), which show completely opaque voxels with value 5/5 and almost transparent voxels with value 1/5 (both plots are rotated to give a nicer view of inner voxels).

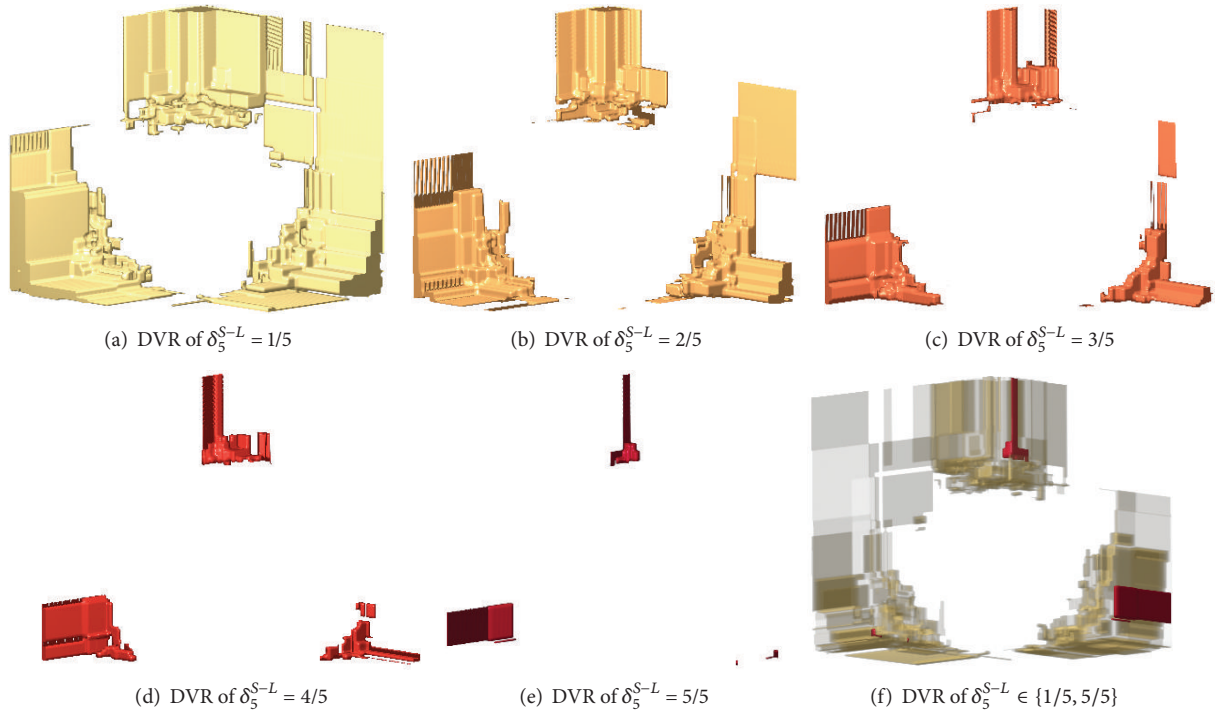
In contrast to MIP, DVR can be used for visualizing EAF values, too. Figure 14 shows an example of such a visualization for the spherical approximation sets, where only the values $\delta_5^S \in \{1/5, 4/5\}$ are shown (opacity is set to 0.3 for these two values and to 0 for all other values).

6. Discussion

Let us summarize the properties of all presented visualization methods (see also Table 1). Slicing is a rather simple method that enables visualization of exact EAF values and differences. Its biggest advantage is its accuracy; it enables a detailed analysis of the approximation sets in the same way that can be done for 2D EAFs. Its biggest drawback is that it is able to visualize only one slice at a time. Generally, the approximation sets are not as symmetrical as in our benchmark case; therefore, multiple slices are needed to sufficiently inspect the EAFs. Also, note that the meaning of the angle φ changes if the observed objective space has different ranges

Table 1: Summarized properties of the presented visualization methods.

	Slicing	MIP	DVR
Exact EAF values	+ Enables detailed analysis of the approximation sets – Visualizes only one slice at a time	– Not sensible to use since usually a large portion of the objective space has the maximum EAF value	– Not possible to use
Exact EAF differences	+ Enables detailed analysis of the approximation sets + Capable of simultaneously visualizing positive and negative differences without overlapping – Visualizes only one slice at a time	+ No parameters to set + Shows all values in a single image – No sense of depth – Feasible only for a limited number of cuboids (impractical)	– Not possible to use
Approximated EAF values	– No need to use it on the approximation as it works well for exact EAFs	– Not sensible to use since usually a large portion of the objective space has the maximum EAF value	+ Nice visualizations + Enables looking through the volume and preserves the sense of depth – Requires definition of the transfer function
Approximated EAF differences	– No need to use it on the approximation as it works well for exact EAFs	+ No parameters to set + Shows all values in a single image – No sense of depth	+ Nice visualizations + Enables looking through the volume and preserves the sense of depth – Requires definition of the transfer function



(a) DVR of $\delta_5^{S-L} = 1/5$ (b) DVR of $\delta_5^{S-L} = 2/5$ (c) DVR of $\delta_5^{S-L} = 3/5$

(d) DVR of $\delta_5^{S-L} = 4/5$ (e) DVR of $\delta_5^{S-L} = 5/5$ (f) DVR of $\delta_5^{S-L} \in \{1/5, 5/5\}$

Figure 13: DVR of approximated 3D EAF differences between the spherical and linear benchmark approximation sets δ_5^{S-L}.

(if R is not a cube, it is not cut exactly in half by the plane at angle $\varphi = 45^\circ$). While slicing can be used also for visualizing approximated EAFs, there is no need to do so as it works well for exact EAFs.

It is not reasonable to use MIP for visualizing EAF values since usually a large portion of the objective space has the maximum EAF value and such plots are not very informative. While MIP can be used for visualizing exact EAF differences,

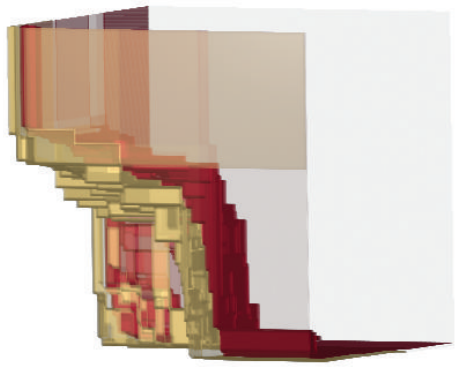


Figure 14: DVR of approximated 3D EAF values of the spherical benchmark approximation sets for $\alpha_5^S \in \{1/5, 4/5\}$.

it works much better on the approximated case, for which it was conceived. Visualizing 3D cuboids is rather impractical (especially for a very large number of cuboids) and requires sorting of cuboids with regard to their value in order to produce MIP image. However, MIP can easily be used to visualize approximated EAF differences as it has no parameters to set. Its biggest advantage is that it combines all values in a single image giving precedence to largest differences (which are the most important when visualizing EAF differences), while its the biggest disadvantage is the lost sense of depth.

Finally, DVR can be used to visualize approximated EAFs. It produces nice and informative visualizations of both the EAF values and differences. With an insightful setting of the transfer function it is even possible to "look through" the cloud of cuboids. The need to define the transfer function is at the same time the biggest disadvantage of DVR as it might be demanding for a user not familiar with the method. Nevertheless, it is much easier to find the right transfer function for EAFs than, for example, for some medical data, as EAFs take only discrete values.

7. Steel Casting Use Case

This section shows how the described visualization methods can be used on a real-world optimization problem with three objectives from a previous study [27].

7.1. The Steel Casting Problem. Continuous casting of steel is a complex metallurgical process where molten steel is cooled and shaped into semimanufactures of desired dimensions. Cooling is done using water in the primary and secondary cooling subsystems. Primary cooling is performed in the mold, while secondary cooling comprehends wreath cooling at the exist from the mold, and spray cooling of the strand.

The goal is to set the parameters of this process (casting speed, mold outlet coolant temperature, and wreath and spray coolant flows (see Table 2)) in such a way that the quality of the cast steel is as high as possible. The quality of steel is defined by the following three objectives: distance from desired *metallurgical length* (the length of the liquid core in the strand), distance from desired *shell thickness* (thickness of the solid shell at mold exit), and distance from desired *surface temperature* at a predefined point in the strand. Table 3 details the variables defining these optimization objectives. Their bounds and desired values were determined by domain experts.

Table 2: Steel casting process parameters.

Parameter	Lower bound	Upper bound	Disc. step
Casting speed (m/min)	1.5	2.0	0.01
Mold outlet coolant temp. (°C)	33	35	1
Wreath coolant flow (m^3/h)	10	40	5
Spray coolant flow (m^3/h)	25	65	5

Table 3: Variables defining the optimization objectives.

Variable	Lower bound	Upper bound	Desired value
Metallurgical length (m)	10	11	10
Shell thickness (mm)	11	15	13
Surface temperature (°C)	1115	1130	1122.5

If a parameter setting yields values outside the boundary constraints presented in Table 3, it is deemed infeasible. The objectives of the feasible solutions are computed by

$$f_k = \left| y_k - y_k^* \right| \quad \forall i \in 1, 2, 3, \tag{30}$$

where y_k is the value of the observed variable and y_k^* is its desired value.

As real-world experimentation with parameter settings is expensive and time-consuming and could also be dangerous, we have simulated it using a numerical model of steel casting based on a meshless technique for diffusive heat transport [28]. One simulation of the steel casting process takes approximately 2 minutes on a standard desktop computer.

Two instances of this optimization problem were studied: *discrete* and *continuous*. The discrete problem instance was set by domain experts as presented in Table 2 in such a way that all possible solutions (9639 in total) could be explored. While the discrete problem instance enables the rough exploration of the objective space, the continuous problem instance where any value within the variable bounds could be chosen is the one we wish to solve.

7.2. Results of Optimization Algorithms. The discrete problem instance was solved using the exhaustive search (ES) algorithm. These solutions form the Pareto front of the discrete problem instance and we wanted to see whether an algorithm tackling the continuous problem instance could come close to this Pareto front.

We chose the algorithm DEMO (differential evolution for multiobjective optimization) [29], which is a MOEA algorithm that uses differential evolution [30], to explore the decision space for the continuous problem instance. DEMO was run five times, each time exploring 3200 solutions. More detailed information on the employed experimental setup can be found in [27].

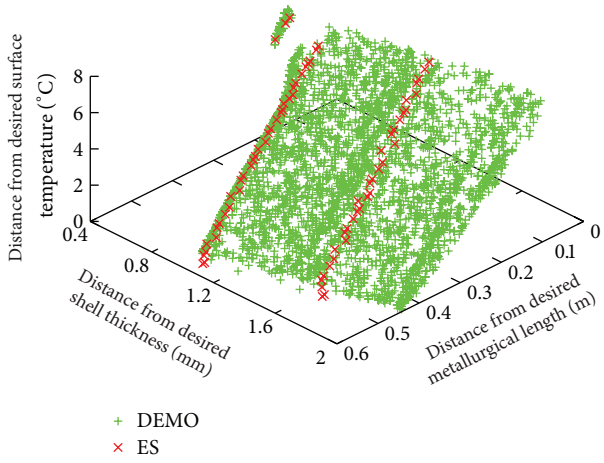


FIGURE 15: The best solutions found by ES on the discrete problem instance and by DEMO on the continuous problem instance (all five runs shown).

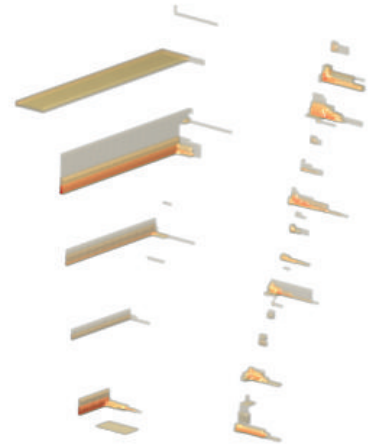
The majority of the explored solutions were infeasible. For example, out of 9639 solutions found by ES only 1242 were feasible and of those only 72 were mutually nondominated. DEMO was solving an extended problem and therefore found more nondominated solutions: on average, 645 per run (although their number varied significantly over different runs).

First, we visualize all final (feasible and nondominated) solutions found by both algorithms in Figure 15. We can see that ES found two distinct subsets of solutions. A more detailed analysis of results revealed that they correspond to two of the three mold outlet coolant temperatures (the remaining one always produces infeasible solutions). Since DEMO was not bound by this discretization, it was able to find solutions also around these subsets. Both algorithms found a few solutions with low distances from desired shell thickness that are rather "detached" from the rest. They actually lie on a larger disconnected region of the Pareto front of which just this minor part is feasible.

This visualization is able to show that DEMO is able to reach the Pareto front of the discrete problem and is therefore a good choice for solving this problem. However, there are two aspects we are interested in that are hard to infer from this visualization alone (they could, of course, be computed from the solutions). First, we wish to see whether different runs of DEMO produce similar results. Visualizing all five approximation sets together with different markers or visualizing one approximation set at a time does not provide a good means for comparison. Second, although the approximation sets found by DEMO look linear, they are in fact slightly convex making it very hard to see (even by rotating the objective space) whether results by ES are in fact dominated by those by DEMO. We will try to find the answers to these questions using the proposed visualization methods.

In order to use the proposed visualization methods to visualize the results of ES and DEMO, we need to address two issues. The first issue is the uneven ranges of the attainment surfaces, which are important when computing voxels.

(a) DVR of $\delta_5^{\text{DEMO-ES}}$

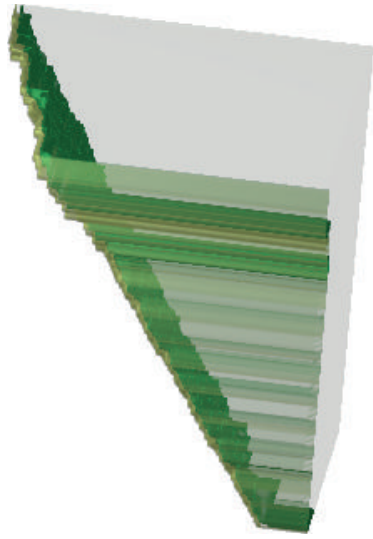


(b) DVR of $\delta_5^{\text{ES-DEMO}}$

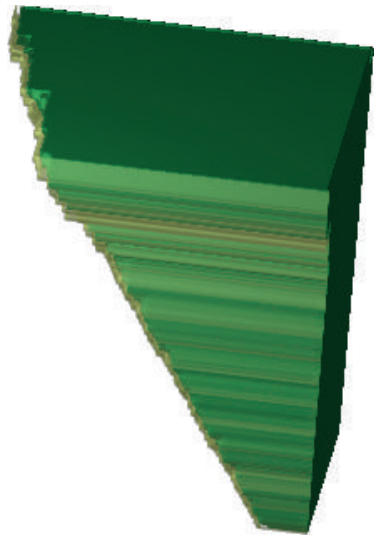
FIGURE 16: DVR of approximated 3D EAF differences between the two algorithms for all values in $\{1/5, \ldots, 5/5\}$.

As we can infer from Table 3, the observed objective space is equal to $[0, 1] \times [0, 2] \times [0, 7.5]$. Moreover, the feasible results found by the two algorithms actually lie in an even smaller cuboid contained in $[0, 0.6] \times [0.5, 2] \times [0, 7.5]$. If the number of voxels was proportional to the objective ranges, the first two objectives would be discretized too roughly. Therefore, we choose to normalize the objective vectors with regard to $[0, 0.6] \times [0.5, 2] \times [0, 7.5]$. The second issue is the uneven number of runs of both algorithms (ES was run once). To leave the meaning of the EAFs differences intact, we copy the results by ES to get five runs in total. This seems a reasonable approach as the deterministic ES would actually produce five equal results if it was run five times.

Now, visualization methods as described in Sections 4 and 5 can be used on these results. Figure 16 presents the DVR of approximated EAF differences between the two algorithms, $\delta_5^{\text{DEMO-ES}}$ and $\delta_5^{\text{ES-DEMO}}$, which gives us a general idea of their performance. However, this does not answer our two questions. To check the repeatability of DEMO, we need to inspect the difference between the best and worst

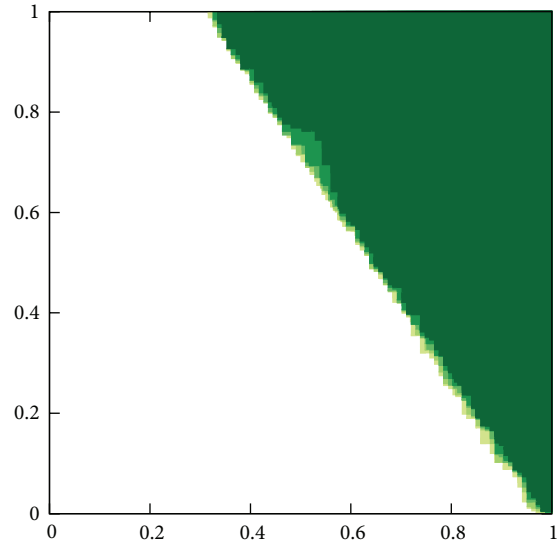


(a) DVR of $\alpha_5^{\text{DEMO}} \in \{1/5, 4/5\}$

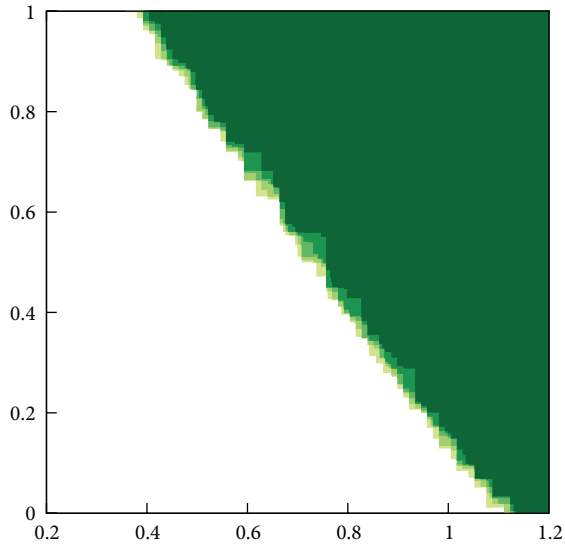


(b) DVR of $\alpha_5^{\text{DEMO}} \in \{1/5, 5/5\}$

Figure 17: DVR of approximated 3D EAF values of DEM.



(a) Slice of α_5^{DEMO} at $\varphi = 25^\circ$



(b) Slice of α_5^{DEMO} at $\varphi = 45^\circ$

Figure 18: Slices of exact 3D EAF values of DEMO at two angles.

summary attainment surfaces. The narrow layer between $\alpha_5^{\text{DEMO}} = 1/5$ and $\alpha_5^{\text{DEMO}} = 4/5$ as well as $\alpha_5^{\text{DEMO}} = 1/5$ and $\alpha_5^{\text{DEMO}} = 5/5$ (see Figure 17) suggests that although DEMO's approximation sets were of considerably different cardinality, they attain a similar portion of the objective space. This is further confirmed by a more detailed inspection using slicing at angles $\varphi = 25^\circ$ and 45° (see Figure 18), which shows that only a small border of the attained objective space is attained less than five times (denoted by light green hues).

Finally, MIP can be used to check whether DEMO ever finds better solutions than those found by ES. If this was not the case, each solution found by ES would have exactly one union of cuboids (albeit small) for which $\delta_5^{\text{ES-DEMO}} = 5/5$. As accuracy is important in this case (small cuboids can be omitted if the discretization into voxels is not fine enough), we use MIP on exact EAF differences. Figure 19 clearly shows that although $\delta_5^{\text{ES-DEMO}} = 5/5$ for some solutions, this does not hold for all of them, meaning that DEMO was actually able to find solutions that dominate those by ES. This was of course possible only because DEMO was solving an extended instance of the problem.

8. Conclusions

When comparing the results of multiobjective optimization algorithms it is important to be able to visualize them as this can provide new information regarding the algorithms or the given problem. If the algorithms are stochastic, the EAF can be used to describe how well the algorithms attain the objective space with their multiple approximation sets. Visualization of EAFs is rather straightforward in 2D but presents a challenge in 3D as multiple cuboids need to be visualized.

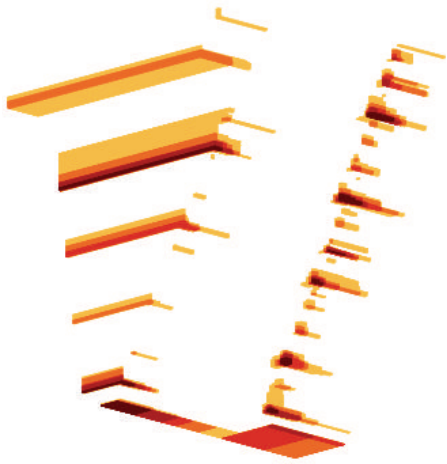


Figure 19: MIP of exact 3D EAF differences between ES and DEMO, $\delta_5^{\text{ES-DEMO}}$.

This paper presented how these cuboids can be computed and visualized using slicing and MIP. If accuracy of visualization is not crucial, the EAF values and differences can be approximated by discretizing the objective space into a grid of voxels. In this way, visualization can be performed using slicing, MIP, and DVR. We have shown how all these methods perform on two sets of benchmark approximation sets and discussed their advantages and disadvantages.

In addition, we demonstrated the use of slicing, MIP, and DVR on a real-world optimization problem solved by exhaustive search and MOEA algorithm. We have shown that these powerful visualization methods are able to give a new insight regarding the performance of the algorithms that cannot be otherwise seen using solely "standard" visualization of approximation sets.

In the future, we wish to find a more efficient way to compute the cuboids either by improving the provided algorithm or by adjusting existing efficient algorithms for hypervolume calculation to suit our needs.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors are grateful to Robert Vertnik for making the casting process simulator available and Miha Mlakar for providing the results of DEMO and ES. The work presented in this paper was carried out as part of research Program P2-0209 and research Projects L2-3651 and J2-4120, all funded by the Slovenian Research Agency.

References

[1] K. Deb, *Multi-Objective Optimization Using Evolution ary Algorithms*, John Wiley & Sons, Chichester, UK, 2001.

[2] K. Deb, "Advances in evolutionary multi-objective optimization," in *Search Based Software Engineering*, G. Fraser and J. T. de Souza, Eds., vol. 7515 of *Lecture Notes in Computer Science*, pp. 1–26, Springer, New York, NY, USA, 2012.

[3] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[4] T. Tusar and B. Filipic, "Visualization of Pareto front approximations in evolutionary multiobjective optimization: a critical review and the prosection method," *IEEE Transactions on Evolutionary Computation*, 2014.

[5] V. D. G. da Fonseca, C. M. Fonseca, and A. O. Hall, "Inferential performance assessment of stochastic optimisers and the attainment function," in *Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization (EMO '01)*, vol. 1993 of *Lecture Notes in Computer Science*, pp. 213–225, Springer, 2001.

[6] T. Tušar and B. Filipič, "An approach to visualizing the 3D empirical attainment function," in *Proceeding of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*, pp. 1367–1372, New York, NY, USA, July 2013.

[7] T. Tušar and B. Filipic, "Initial experiments in visualization of empirical attainment function differences using maximum intensity projection," in *Proceedings of the 16th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '14)*, pp. 1099–1106, Vancouver, Canada, 2014.

[8] J. W. Wallis, T. R. Miller, C. A. Lerner, and E. C. Kleerup, "Three-dimensional display in nuclear medicine," *IEEE Transactions on Medical Imaging*, vol. 8, no. 4, pp. 297–303, 1989.

[9] M. Levoy, "Display of surfaces from volume data," *IEEE Computer Graphics and Applications*, vol. 8, no. 3, pp. 29–37, 1988.

[10] C. M. Fonseca, A. P. Guerreiro, M. Lopez-Ibanez, and L. Paquete, "On the computation of the empirical attainment function," in *Proceedings of the 6th International Conference on Evolutionary Multi-Criterion Optimization (EMO '11)*, vol. 6576 of *Lecture Notes in Computer Science*, pp. 106–120, Springer, New York, NY, USA, 2011.

[11] M. Lopez-Ibanez, L. Paquete, and T. Stiitzle, "Exploratory analysis of stochastic local search algorithms in biobjective optimization," in *Experimental Methods for the Analysis of Optimization Algorithms*, T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, Eds., pp. 209–222, Springer, 2010.

[12] E. Zitzler, D. Brockhoff, and L. Thiele, "The hypervolume indicator revisited: on the design of Pareto-compliant indicators via weighted integration," in *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization (EMO '07)*, vol. 4403 of *Lecture Notes in Computer Science*, pp. 862–876, Springer, 2001.

[13] J. Knowles, "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 (ISDA '05)*, pp. 552–557, September 2005.

[14] "Attainment function tools," http://eden.dei.uc.pt/~cmfonsec/software.html.

[15] M. López-Ibáñez and T. Stützle, "An experimental analysis of design choices of multi-objective ant colony optimization algorithms," *Swarm Intelligence*, vol. 6, no. 3, pp. 207–232, 2012.

[16] T. Tušar and B. Filipič, "Visualizing 4D approximation sets of multiobjective optimizers with prosections," in *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference (GECCO '11)*, pp. 737–744, July 2011.

[17] A. V. Lotov and K. Miettinen, "Visualizing the Pareto frontier," in *Multiobjective Optimization*, J. Branke, K. Deb, K. Miettinen, and R. Slowinski, Eds., vol. 5252 of *Lecture Notes in Computer Science*, pp. 213–243, Springer, 2008.

[18] W. Heidrich, M. McCool, and J. Stevens, "Interactive maximum projection volume rendering," in *Proceedings of the IEEE Conference on Visualization*, pp. 11–18, 1995.

[19] J. Diaz and P. Vazquez, "Depth-enhanced maximum intensity projection," in *Proceedings of the 8th IEEE/EG International Conference on Volume Graphics (VG '10)*, pp. 93–100, 2010.

[20] K. Engel, M. Hadwiger, J. M. Kniss, C. Rezk-Salama, and D. Weiskopf, *Real-time Volume Graphics*, A. K. Peters, Natick, Mass, USA, 2006.

[21] C. P. Botha and F. H. Post, "New technique for transfer function specification in direct volume rendering using real-time visual feedback," in *Proceeding of the Medical Imaging: Visualization, Image-Guided Procedures and Display*, pp. 349–356, San Diego, Calif, USA, February 2002.

[22] P. Sereda, A. Vilanova, and F. A. Gerritsen, "Automating transfer function design for volume rendering using hierarchical clustering of material boundaries," in *Proceedings of the 8th Joint Eurographics/IEEE VGTC Conference on Visualization (EUROVIS '06)*, pp. 243–250, 2006.

[23] N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, 1995.

[24] N. Beume, C. M. Fonseca, M. López-Ibáñez, L. Paquete, and J. Vahrenhold, "On the complexity of computing the hypervolume indicator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1075–1082, 2009.

[25] J. Meyer-Spradow, T. Ropinski, J. Mensmann, and K. Hinrichs, "Voreen: a rapid-prototyping environment for ray-casting-based volume visualizations," *IEEE Computer Graphics and Applications*, vol. 29, no. 6, pp. 6–13, 2009.

[26] "Voreen, volume rendering engine," http://voreen.uni-muenster.de/.

[27] M. Mlakar, T. Tušar, and B. Filipič, "Discrete vs. continuous multiobjective optimization of continuous casting of steel," in *Proceeding of the 14th International Conference on Genetic and Evolutionary Computation (GECCO '12)*, pp. 587–590, New York, NY, US, July 2012.

[28] R. Vertnik and B. Šarler, "Simulation of continuous casting of steel by a meshless technique," *International Journal of Cast Metals Research*, vol. 22, no. 1–4, pp. 311–313, 2009.

[29] T. Robic and B. Filipic, "DEMO: differential evolution for multiobjective optimization," in *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization*, vol. 3410 of *Lecture Notes in Computer Science*, pp. 520–533, Springer, 2005.

[30] K. V. Price and R. Storn, "Differential evolution—a simple evolution strategy for fast optimization," *Dr. Dobb's Journal*, vol. 22, no. 4, pp. 18–24, 1997.