A Simple Recursive Algorithm for calculating Expected Hypervolume Improvement

Alistair Shilton, Santu Rana, Sunil Kumar Gupta, Svetha Venkatesh
Center for Pattern Recognition and Data Analytics, Deakin University, Geelong, Australia
{ashilton,santu.rana,sunil.gupta,svetha.venkatesh}@deakin.edu.au

Abstract

In multi-objective optimisation, expected hypervolume improvement is a popular metric for assessing the merit of candidate solutions to guide the optimisation process. However the computational cost of calculating the EHI can become prohibitive, particularly as the number of objective functions increases. In this paper we present a new recursive algorithm for calculating the EHI. We show that the algorithm is simple to implement and significantly faster than alternative methods.

1 Introduction

In multi-objective Pareto optimisation the expected hypervolume improvement (EHI) is a popular metric for measuring the merit of candidate solutions to guide the optimisation process [11, 15, 10]. However the computational cost of calculating the EHI remains a significant hurdle when applying such approaches [12, 15], particularly as the number of objectives (and hence the dimensionality of the hypervolume) becomes larger. In particular while heavily optimised algorithms are available for calculating EHI for up to 3 dimensions (for example [7]) the more general case remains computationally challenging.

Most approaches to exactly calculating the EHI tend to be cell-based [3, 2, 1, 7]: the space is divided into cells based on the set of vectors defining (dominating) the hypervolume, the contribution of each cell calculated, and the result is the sum of all such contributions. Cell-based approaches can be somewhat complex to implement in the general (n-dimensional) case - for example, in the IRS algorithm [7] when calculating the contribution of a cell one must calculate correction factors by enumerating all subsets of the axis $\{x, y, z, \ldots\}$ and calculating the dominated (projected) hypervolume for each cross-section so defined. Our aim in the present paper is to provide an algorithm for calculating the EHI that is both fast (computationally efficient) and simple to implement. To achieve this we eschew the standard cell-based approach and instead base our method on the Hypervolume by Slicing Objectives algorithm (HSO, [14]), which is a recursive algorithm for calculating the hypervolume (not the EHI) of a dominated set. The result is a fast and easy to implement recursive algorithm for calculating EHI.

1.1 Notation

The integers modulo $n \in \mathbb{Z}^+$ are denoted $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ ($\mathbb{Z}_0 = \emptyset$). Column vectors are written $\mathbf{a}, \mathbf{b}, \ldots \in \mathbb{A}^n$ ($n \in \mathbb{Z}^+$), with elements denoted $a_i, b_i, \ldots \; \forall i \in \mathbb{Z}_n$ (indices start from 0, C-style). Following Matlab, $n : m = [n, n+1, \ldots, m]^{\mathrm{T}}$ for $n \geq m \in \mathbb{Z}$, and if $\mathbf{a} \in \mathbb{A}^n$ and $\mathbf{i} \in \mathbb{Z}_n^p$ then $\mathbf{a}_\mathbf{i} = [a_{i_0}, a_{i_1}, \ldots, a_{i_{p-1}}]^{\mathrm{T}} \in \mathbb{A}^p$.

2 Background

For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ we say that $\mathbf{x}$ is dominates $\mathbf{y}$, written $\mathbf{x} \succeq \mathbf{y}$, if $x_i \geq y_i \; \forall i \in \mathbb{Z}_n$; and that $\mathbf{x}$ strongly dominates $\mathbf{y}$, written $\mathbf{x} \succ \mathbf{y}$, if $\mathbf{x} \succeq \mathbf{y}$ and $\mathbf{x} \neq \mathbf{y}$. For a finite set $\mathbb{X} \subset \mathbb{R}^n$ we say that $\mathbb{X}$ dominates $\mathbf{y}$, written $\mathbb{X} \succeq \mathbf{y}$, if $\exists \mathbf{x} \in \mathbb{X} : \mathbf{x} \succeq \mathbf{y}$; and similarly $\mathbb{X} \succ \mathbf{y}$ if $\exists \mathbf{x} \in \mathbb{X} : \mathbf{x} \succ \mathbf{y}$. For a finite set

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

$\mathbb{X} = \{\mathbf{x}^0, \mathbf{x}^1, \ldots, \in \mathbb{R}^n\}$ we define $\mathrm{dom}(\mathbb{X})$ to be the set of non-dominated points in $\mathbb{X}$:

$$\mathrm{dom}\left(\mathbb{X}\right) = \left\{ \mathbf{x}^i \in \mathbb{X} \middle| \nexists j \neq i : \mathbf{x}^j \succeq \mathbf{x}^i \right\}$$

We denote the Lebesgue measure (hypervolume) of $\mathbb{Y} \subset \mathbb{R}^n$ by $\mathrm{Vol}\left(\mathbb{Y}\right)$. The $\mathcal{S}$-metric [16] (hypervolume [6, 9], Lebesgue measure [8, 5]) of a finite set $\mathbb{X} \subset (\mathbb{R}^+)^n$ is defined to be:

$$\mathcal{S}_n\left(\mathbb{X}\right) = \mathrm{Vol}\left(\mathbf{y} \in \mathbb{R}^n \middle| \mathbb{X} \succeq \mathbf{y} \succeq \mathbf{0}\right), \mathcal{S}_n\left(\emptyset\right) = 0$$

Where we note that $\mathcal{S}_n\left(\mathbb{X}\right) = \mathcal{S}_n\left(\mathrm{dom}\left(\mathbb{X}\right)\right)$ and $\mathcal{S}_n\left(\{\mathbf{y}\}\right) = \prod_{i \in \mathbb{Z}_n} y_i$.

Given a finite set $\mathbb{X} \subset (\mathbb{R}^+)^n$ and an additional vector $\mathbf{y} \in (\mathbb{R}^+)^n$ the exclusive hypervolume [13] of $\mathbf{y}$ relative to underlying set $\mathbb{X}$, denoted $\Delta\mathcal{S}_n\left(\mathbb{X}\middle|\mathbf{y}\right)$, is defined as the change in $\mathcal{S}$-metric induced by adding the additional vector $\mathbf{y}$ to the set $\mathbb{X}$:

$$\Delta\mathcal{S}_n\left(\mathbb{X}\middle|\mathbf{y}\right) = \mathcal{S}_n\left(\mathbb{X} \cup \{\mathbf{y}\}\right) - \mathcal{S}_n\left(\mathbb{X}\right) \geq 0$$

If $\mathbf{y}$ is drawn from some distribution $\mathcal{P}$ the expected hypervolume improvement (EHI), denoted $\Delta\mathcal{S}_n\left(\mathbb{X}\middle|\mathcal{P}\right)$, is the expected exclusive hypervolume given $\mathbf{y} \sim \mathcal{P}$:

$$\Delta\mathcal{S}_n\left(\mathbb{X}\middle|\mathcal{P}\right) = \mathrm{E}\left[\Delta\mathcal{S}_n\left(\mathbb{X}\middle|\mathbf{y}\right)\middle|\mathbf{y} \sim \mathcal{P}\right] \geq 0$$

The expected hypervolume improvement is commonly used as a measure of merit of a proposed solution [11, 15, 10].

3 Hypervolume and the HSO Algorithm

The Hypervolume by Slicing Objectives algorithm (HSO, [14]) algorithm is a recursive algorithm for calculating $\mathcal{S}_n\left(\mathbb{X}\right)$, where $\mathbb{X} = \left\{\mathbf{x}^0, \mathbf{x}^1, \ldots \mathbf{x}^{M-1} \in (\mathbb{R}^+)^n\right\}$. Our first step is to re-express HSO in the form of a recursive equation which will form the basis of our calculation of expected hypervolume improvement (EHI). The HSO algorithm calculates $\mathcal{S}_n\left(\mathbb{X}\right)$ as follows:

1. Prune (optional): remove all dominated points from $\mathbb{X}$ - that is, $\mathbb{X} \to \mathrm{dom}\left(\mathbb{X}\right)$.
2. Sort: sort the points from smallest to largest based on axis 0, so $x_0^0 \leq x_0^1 \leq \ldots$, and divide the hypervolume into *slices* as shown in figure 2.
3. Recurse: the total volume is the sum of the hypervolumes of all slices, where the volume of each slice is the *length* of that slice (that is, $x_0^j - x_0^{j-1}$) multiplied by the hypervolume of the *collapsed slice*, which is obtained by excluding objective (axis) 0 as shown in figure 2. The base-case is $n = 1$ (1-dimensional), where the hypervolume is simply the length.

The prune and sort steps for our EHI calculation algorithm will be the same as the HSO algorithm. It is convenient to define the following:

- Let m be the number of slices along axis 0 induced by $\mathrm{dom}\left(\mathbb{X}\right) = \{\mathbf{y}, \mathbf{z}, \ldots\}$ - that is, m is the number of distinct values in the set $\{y_0, z_0, \ldots\}$.
- For each slice $j \in \mathbb{Z}_m$ define $\mathbf{i}^j \in \mathbb{Z}_M^{r_j}$, $r_j \in \mathbb{Z}^+$, so that $\{\mathbf{x}^k | k = i_0^j, i_1^j, \ldots i_{r_j-1}^j\}$ contains *all* vectors in $\mathrm{dom}\left(\mathbb{X}\right)$ lying on the upper boundary of slice j while the slices themselves are sorted:

$$\begin{array}{ll} x_0^{i_0^0} < x_0^{i_0^1} < \ldots < x_0^{i_0^{m-1}} & \text{(sorting)} \\ x_0^{i_0^j} = x_0^{i_1^j} = \ldots = x^{i^j_{r_j-1}} \forall j \in \mathbb{Z}_m & \text{(boundary)} \end{array}$$

- For each slice $j \in \mathbb{Z}_m$ define $\mathbb{X}_j \in (\mathbb{R}^+)^{n-1}$ to dominate collapsed slice j (see figure 2):

$$\mathbb{X}_j = \left\{ \mathbf{x}_{1:n-1}^{i_0^k}, \mathbf{x}_{1:n-1}^{i_1^k}, \ldots, \mathbf{x}_{1:n-1}^{i_{r_k-1}^k} \middle| k \in \mathbb{Z}_m \backslash \mathbb{Z}_j \right\}$$

  where we have used Matlab notation $\mathbf{x}_{1:n-1} = [x_1, x_2, \ldots, x_{n-1}]^{\mathrm{T}} \in \mathbb{R}^{n-1}$ and recall that indices are $0, 1, \ldots, n-1$ (start from 0, C style).

- For each slice $j \in \mathbb{Z}_m$ define the lower and upper bounds $l_j = x_0^{i_0^{j-1}}$ ($l_0 = 0$) and $u_j = x_0^{i_0^j}$ on axis 0; and the length $L_j = u_j - l_j$ of slice $j \in \mathbb{Z}_m$ ($L_m = 0$ for convenience).

Thus the hypervolume of collapsed slice $j \in \mathbb{Z}_m$ is $L_j\, \mathcal{S}_{n-1}\left(\mathbb{X}_j\right)$, and we may write the formula for calculating the $\mathcal{S}$-metric, as implemented by the HSO algorithm, in recursive form:

$$\mathcal{S}_n\left(\mathbb{X}\right) = \left\{ \begin{array}{ll} \sum_{j \in \mathbb{Z}_m} L_j\, \mathcal{S}_{n-1}\left(\mathbb{X}_j\right) & \text{if } n > 1 \\ L_0 & \text{if } n = 1 \end{array} \right. \tag{1}$$

which is straight-forward to code. Each step in the recursion reduces the dimensionality of the problem by 1, eventually terminating at the trivial 1-dimensional base-case.

4 Exclusive Hypervolume

Given a set $\mathbb{X} = \{\mathbf{x}^0, \mathbf{x}^1, \ldots \in (\mathbb{R}^+)^n\}$ and an additional vector $\mathbf{y} \in (\mathbb{R}^+)^n$ the exclusive hypervolume is defined as the change in hypervolume induced by adding $\mathbf{y}$ to $\mathbb{X}$ [13] - that is, $\Delta\mathcal{S}_n(\mathbb{X}|\,\mathbf{y}) = \mathcal{S}_n(\mathbb{X} \cup \{\mathbf{y}\}) - \mathcal{S}_n(\mathbb{X}) \geq 0$. For notational convenience we define $u_m = \infty$, $L_m = 0$, $\mathbb{X}_m = \emptyset$. Let $p \in \mathbb{Z}_{m+1}$ be the slice in which the additional vector $\mathbf{y}$ lies - that is, $p = p \in \mathbb{Z}_{m+1} | l_p \leq y_0 < u_p$, where $p = m$ corresponds to y_0 being not less than $x_0\ \forall \mathbf{x} \in \mathbb{X}$. From (1) it may be seen that $\forall n > 1$:

$$\begin{aligned} \mathcal{S}_n(\mathbb{X} \cup \{\mathbf{y}\}) \ = \ & \textstyle\sum_{j \in \mathbb{Z}_p} L_j\, \mathcal{S}_{n-1}(\mathbb{X}_j \cup \{\mathbf{y}_{1:n-1}\}) + (y_0 - l_p)\, \mathcal{S}_{n-1}(\mathbb{X}_p \cup \{\mathbf{y}_{1:n-1}\}) \\ & \left\{ + (u_p - y_0)\, \mathcal{S}_{n-1}(\mathbb{X}_p) + \textstyle\sum_{j=p}^{m} L_j\, \mathcal{S}_{n-1}(\mathbb{X}_j) \ \text{ if } p < m \right\} \end{aligned}$$

It follows that:

$$\Delta\mathcal{S}_n(\mathbb{X}|\,\mathbf{y}) = \begin{cases} \sum_{j \in \mathbb{Z}_{p+1}} \widehat{L}_j\, \Delta\mathcal{S}_{n-1}(\mathbb{X}_j|\,\mathbf{y}_{1:n-1}) & \text{if } n > 1 \\ \max(0, y_0 - L_0) & \text{if } n = 1 \end{cases} \tag{2}$$

where $\widehat{L}_j = L_j\ \forall j \in \mathbb{Z}_p$ and $\widehat{L}_p = y_0 - l_p$. The base-case ($n = 1$) here is the exclusive hypervolume (change in hypervolume) in 1-dimensional space, which is just the change in dominated length. The exclusive hypervolume may be calculated directly using a slight variant of the HSO algorithm. The prune and sort steps remain unchanged, and the position p may be trivially calculated. The structure of the recursive equation is also much the same, except that the range of summation is different ($j \in \mathbb{Z}_{p+1}$ rather than $\mathbb{Z}_m$), the length L_j has been replaced by $\widehat{L}_j$, and the base case differs.

5 Expected Hypervolume Improvement

In multi-objective optimisation we often wish to calculate the expected hypervolume improvement. This is similar the the exclusive hypervolume, except that $\mathbf{y}$ is treated as a random variable rather than a known quantity. Suppose $\mathbf{y} \sim \mathcal{P}$, where $\mathcal{P}$ is some distribution, such that the scalar components $y_0, y_1, \ldots$ are drawn from independent distributions $y_i \sim \mathcal{P}_i\ \forall i \in \mathbb{Z}_n$ (so $\mathcal{P} = \mathcal{P}_0 \otimes \mathcal{P}_1 \otimes \ldots \otimes \mathcal{P}_{n-1}$). Further assume that $\forall i \in \mathbb{Z}_n$ the distributions $\mathcal{P}_i$ are continuous distributions with densities $f_i(z)$. The expected hypervolume improvement is:

$$\begin{aligned} \Delta\mathcal{S}_n(\mathbb{X}|\,\mathcal{P}) \ &= \ \mathrm{E}\left[\Delta\mathcal{S}_n(\mathbb{X}|\,\mathbf{y})|\,\mathbf{y} \sim \mathcal{P}\right] \\ &= \ \textstyle\int_{z_0=0}^{\infty} \ldots \int_{z_{n-1}=0}^{\infty} \Delta\mathcal{S}_n(\mathbb{X}|\,\mathbf{z})\, f_0(z_0) \ldots f_{n-1}(z_{n-1})\, dz_0 \ldots dz_{n-1} \\ &= \ \textstyle\sum_{p \in \mathbb{Z}_{m+1}} \Delta\mathcal{S}_{n;p}(\mathbb{X}, \mathcal{P}) \ \text{ if } n > 1 \end{aligned}$$

where, recalling that $l_0 = 0$, $u_m = \infty$ and $u_j = l_{j+1}$, we have defined:

$$\Delta\mathcal{S}_{n;p}(\mathbb{X}|\,\mathcal{P}) \ = \ \textstyle\int_{z_0=l_p}^{u_p} \int_{z_1=0}^{\infty} \ldots \int_{z_{n-1}=0}^{\infty} \Delta\mathcal{S}_n(\mathbb{X}|\,\mathbf{z})\, f_0(z_0)\, f_1(z_1) \ldots dz_0 \ldots dz_{n-1}$$

Using (2) it follows that:

$$\Delta\mathcal{S}_{n;p}(\mathbb{X}|\mathcal{P}) = \textstyle\int_{l_p}^{u_p} (z - l_p) f_0(z) dz\, \Delta\mathcal{S}_{n-1}(\mathbb{X}_p|\mathcal{P}_{1:n-1}) + \sum_{j \in \mathbb{Z}_p} L_j \int_{l_p}^{u_p} f_0(z) dz\, \Delta\mathcal{S}_{n-1}(\mathbb{X}_j|\mathcal{P}_{1:n-1})$$

where we have slightly abused Matlab notation by denoting $\mathcal{P}_{1:n-1} = \mathcal{P}_1 \otimes \mathcal{P}_2 \otimes \ldots \otimes \mathcal{P}_{n-1}$. Combining these results, and recalling that $L_m = 0$, $\mathbb{X}_m = \emptyset$, it may be seen that:

$$\Delta\mathcal{S}_n(\mathbb{X}|\,\mathcal{P}) = \begin{cases} \sum_{j \in \mathbb{Z}_{m+1}} \widetilde{L}_j\, \Delta\mathcal{S}_{n-1}(\mathbb{X}_j|\,\mathcal{P}_{1:n-1}) & \text{if } n > 1 \\ \widetilde{L}_m & \text{if } n = 1 \end{cases} \tag{3}$$

where $\widetilde{L}_j = \int_{l_j}^{u_j} (z - l_j)\, f_0(z)\, dz + L_j \int_{u_j}^{\infty} f_0(z)\, dz\ \forall j \in \mathbb{Z}_{m+1}$. This recursive equation allows us to calculate EHI using a simple variant of the HSO algorithm. There are two distinguishing features here that differ from the HSO algorithm, namely that the range of summation is $j \in \mathbb{Z}_{m+1}$ (as opposed to $j \in \mathbb{Z}_m$ in (1)); and the lengths L_j used in the HSO algorithm must be replaced by the distribution-dependent scaling factors $\widetilde{L}_j$. We call this variant of the HSO algorithm for calculating EHI the ΔHSO algorithm.

Depending on the distribution the density-dependent scaling factors $\widetilde{L}_j$ may be derived in

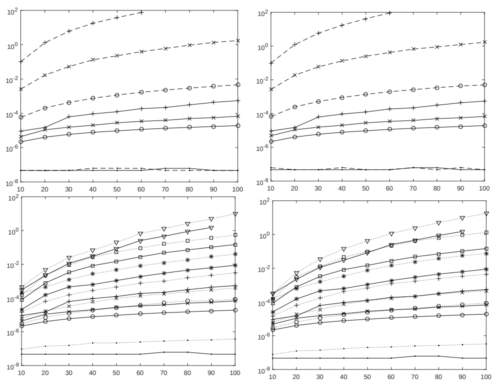

Figure 1: Average EHI computation times (seconds) versus dataset size. `ConvexSpherical` on left, `ConcaveSpherical` on right, $\Delta\text{HSO}_{\text{fast}}$ versus IRS (dashed line) on top; $\Delta\text{HSO}_{\text{fast}}$ versus ΔHSO (dotted line) on bottom. Key: $\cdot$ ($n=1$), $\circ$ ($n=2$), $\times$ ($n=3$), $+$ ($n=4$), $*$ ($n=5$), $\square$ ($n=6$), ∇ ($n=7$).

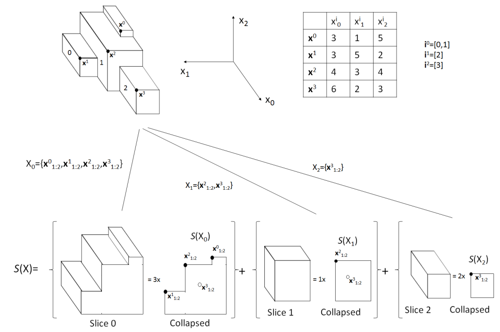

Figure 2: HSO algorithm operational example. Points are sorted according to axis x_0 to obtain the index vectors $\mathbf{i}^0$, $\mathbf{i}^1$, $\mathbf{i}^2$. The block (top left) is then divided into 3 slices (bottom) that are then collapsed by removing axis x_0. The hypervolume is the sum of the slice length (on x_0 axis) multiplied by the hypervolume of the collapsed slice - i.e. $\mathcal{S}_3(\mathbb{X}) = 3\,\mathcal{S}_2(\mathbb{X}_0) + 1\,\mathcal{S}_2(\mathbb{X}_1) + 2\,\mathcal{S}_2(\mathbb{X}_2)$.

closed-form. For example, suppose $\mathcal{P}_i = \mathcal{N}\left(\mu_i, \sigma_i^2\right)\ \forall i \in \mathbb{Z}_n$. It follows that $f_i(z) = (2\sigma_i^2\pi)^{-1/2}\exp(-(z-\mu_i)^2/(2\sigma_i^2))$, and it is not difficult to show that:

$$\int_m^M f_i(z)\,dz = \frac{1}{2}\left(\operatorname{erf}\left(\frac{M-\mu_i}{\sqrt{2\sigma_i^2}}\right) - \operatorname{erf}\left(\frac{m-\mu_i}{\sqrt{2\sigma_i^2}}\right)\right)$$

$$\int_m^M (z-m) f_i(z)\,dz = \frac{m-\mu_i}{2}\left(\operatorname{erf}\left(\frac{m-\mu_i}{\sqrt{2\sigma_i^2}}\right) - \operatorname{erf}\left(\frac{M-\mu_i}{\sqrt{2\sigma_i^2}}\right)\right) + \frac{\sigma_i}{\sqrt{2\pi}}\left(\exp\left(-\frac{(m-\mu_i)^2}{2\sigma_i^2}\right) - \exp\left(-\frac{(M-\mu_i)^2}{2\sigma_i^2}\right)\right)$$

and hence:

$$\widetilde{L}_j = \sigma_0 e\left(\frac{l_j-\mu_0}{\sqrt{2\sigma^2}}\right) - \sigma_0 e\left(\frac{u_j-\mu_0}{\sqrt{2\sigma^2}}\right)\ \forall j \in \mathbb{Z}_m, \quad \widetilde{L}_m = \sigma_0 e\left(\frac{l_m-\mu_0}{\sqrt{2\sigma^2}}\right) \tag{4}$$

where $e(z) = \frac{z}{\sqrt{2}}(\operatorname{erf}(z) - 1) + \frac{1}{\sqrt{2\pi}}\exp\left(-z^2\right)$.

6 Experimental Validation

In our experiments we have compared three algorithms: ΔHSO (our algorithm), $\Delta\text{HSO}_{\text{fast}}$ (an optimised version of our algorithm that (a) retains a cache of calculated $E_{ij} = \sigma_j e((x_j^i - \mu_j)/(\sqrt{2}\sigma_j))$ values and (b) pre-calculates index vectors $\mathbf{i}^j$ etc that depend only on $\mathbb{X}$) and IRS (the algorithm described in described [7]). All simulations were written in C++. We have used the `ConvexSpherical` and `ConcaveSpherical` datasets from [4] with $1 \leq n \leq 9$.

We have validated our method by comparing the output of ΔHSO with IRS for all experiments.[1] Figure 1 shows the average computation time (computed over 1000 sequential evaluations each) for a single evaluation of EHI for ΔHSO, $\Delta\text{HSO}_{\text{fast}}$ and IRS algorithms. As can be seen from these results ΔHSO is significantly faster than IRS, and moreover $\Delta\text{HSO}_{\text{fast}}$ is faster than ΔHSO.

7 Conclusion

We have presented a simple recursive algorithm (ΔHSO) for calculating the expected hypervolume improvement (EHI) using a variant of the recursive hypervolume slicing optimisation (HSO) algorithm used for calculating hypervolume (not EHI). In contrast to most cell-based approaches, our method is recursive and therefore very easy to implement. We have shown that the computational cost of our approach in practise is better than that of comparable cell-based algorithms (IRS). We have also presented an optimised form of ΔHSO, called $\Delta\text{HSO}_{\text{fast}}$, and studied the relative merits of $\Delta\text{HSO}_{\text{fast}}$ and ΔHSO.

[1] For validation of results we also implemented a simplified version of [1]. As we implemented only a simplified version of this algorithm (using binary cells rather than WFG generated ones) we have not reported timings as they are not representative of the full version.

References

[1] Ivo Couckuyt, Dirk Deschrijver, and Tom Dhaene. Fast calculation of multiobjective probability of improvement and expected improvement criteria for pareto optimization. *Journal of Global Optimization*, 60(3):575–594, 2014.

[2] Michael Emmerich and Jan-Willem Klinkenberg. The computation of the expected improvement in dominated hypervolume of pareto front approximations. Technical report, Leiden University, 2008.

[3] Michael T. M. Emmerich, André H. Deutz, and Jan Willem Klinkenberg. Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *Proceedings of the 2011 IEEE Congress on Evolutionary Computation (CEC)*, pages 2147–2154, 2011.

[4] Michael T. M. Emmerich and Carlos M. Fonseca. *Computing Hypervolume Contributions in Low Dimensions: Asymptotically Optimal Algorithm and Complexity Results*, pages 121–135. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[5] Mark Fleischer. The measure of pareto optima: Applications to multi-objective metaheuristics. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 519–533, 2000.

[6] Simon Huband, Phil Hingston, Lyndon While, and Luigi Barone. An evolution strategy with probabilistic mutation for multi-objective optimisation. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 4, pages 2284–2291, 2003.

[7] Iris Hupkens, André Deutz, Kaifeng Yang, and Michael Emmerich. Faster exact algorithms for computing expected hypervolume improvement. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 65–79. Springer, 2015.

[8] Marco Laumanns, Eckart Zitzler, and Lothar Thiele. A unified model for multi-objective evolutionary algorithms with elitism. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 1, pages 46–53, 2000.

[9] Robin Charles Purshouse. *On the Evolutionary Optimisation of Many Objectives*. PhD thesis, University of Sheffield, 2003.

[10] Koji Shimoyama, Shinkyu Jeong, and Shigeru Obayashi. Kriging-surrogate-based optimization considering expected hypervolume improvement in non-constrained many-objective test problems. In *Proceedings of 2013 IEEE Congress on Evolutionary Computation*, 2013.

[11] Ofer M. Shir, Michael Emmerich, Thomas Back, and Marc J. J. Vrakking. The application of evolutionary multi-criteria optimization to dynamic molecular aligment. In *Proceedings of 2007 IEEE Congress on Evolutionary Computation*, 2007.

[12] Tobias Wagner, Michael Emmerich, André Deutz, and Wolfgang Ponweiser. On expected-improvement criteria for model-based multi-objective optimization. In *Proceedings of the 2010 International Conference on Parallel Problem Solving from Nature*, pages 718–727, 2010.

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

[14] Lyndon While, Philip Hingston, Luigi Barone, and Simon Huband. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, 2006.

[15] Martin Zaefferer, Thomax Bartz-Beielstein, Boris Naujoks, Tobias Wagner, and Michael Emmerich. A case study on multi-criteria optimization of an event detection software under limited budgets. In *Proceedings of the 2013 International Conference on Evolutionary Multi-Criterion Optimization*, pages 756–770. Springer, 2013.

[16] Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology Zurich, 1999.