Efficient Real-Time Hypervolume Estimation with Monotonically Reducing Error

Jonathan E. Fieldsend
University of Exeter
Department of Computer Science
EX4 4QF, UK
J.E.Fieldsend@exeter.ac.uk

ABSTRACT

The hypervolume (or S-metric) is a widely used quality measure employed in the assessment of multi- and many-objective evolutionary algorithms. It is also directly integrated as a component in the selection mechanism of some popular optimisers. Exact hypervolume calculation becomes prohibitively expensive in real-time applications as the number of objectives increases and/or the approximation set grows. As such, Monte Carlo (MC) sampling is often used to estimate its value rather than exactly calculating it. This estimation is inevitably subject to error. As standard with Monte Carlo approaches, the standard error decreases with the square root of the number of MC samples. We propose a number of real-time hypervolume estimation methods for unconstrained archives — principally for use in real-time convergence analysis. Furthermore, we show how the number of domination comparisons can be considerably reduced by exploiting incremental properties of the approximated Pareto front. In these methods the estimation error monotonically decreases over time for (i) a capped budget of samples per algorithm generation and (ii) a fixed budget of dedicated computation time per optimiser generation for new MC samples. Results are provided using an illustrative worst-case scenario with rapid archive growth, demonstrating the orders-of-magnitude of speed-up possible.

CCS CONCEPTS

• **Applied computing** → **Multi-criterion optimization and decision-making**; • **General and reference** → **Estimation**; **Performance**; • **Mathematics of computing** → **Evolutionary algorithms**.

KEYWORDS

Monte Carlo, hypervolume, real-time statistics, real-time analysis

ACM Reference Format:
Jonathan E. Fieldsend. 2019. Efficient Real-Time Hypervolume Estimation with Monotonically Reducing Error. In *Genetic and Evolutionary Computation Conference (GECCO '19), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3321707.3321730

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6111-8/19/07…$15.00
https://doi.org/10.1145/3321707.3321730

1 INTRODUCTION

In evolutionary multi-objective optimisation (optimising problems with two-three objectives) and many-objective optimisation (optimising problems with four or more objectives), the task is to find a *good* approximation set of solutions to the optimal Pareto set for the problem at hand. That is, the set of best possible trade-off solutions. Defining what a 'good' approximation set is for a multi-objective problem is itself a multi-faceted problem. This is why a range of indicators are often employed [16]. Nevertheless, one of the most regularly used quality measures is the *hypervolume* indicator. This measure calculates the (hyper)volume of the objective domain lying between the image an approximation set under the objective function and a reference point. It has the distinction among quality indicators on approximation sets of being Pareto compliant. As detailed in [25]:

(1) If one approximation completely dominates another approximation, it will have the greater hypervolume.
(2) The approximation set that maximises the hypervolume for a particular problem will represent all Pareto-optimal objective vectors.[1]

These properties underpin its popularity. However, its calculation can be relatively expensive compared to other quality measures.

In this work we detail computationally efficient methods for hypervolume *estimation* when maintaining an unconstrained Pareto set approximation, A (often referred to as an archive). This is mainly used during an optimisation process, to enable the use of the hypervolume in real-time for e.g. algorithm convergence and stopping decisions, even when the number of objectives and/or $|A|$ is large. By utilising information from previous time steps this approximation is shown to monotonically improve over time. This results in an orders-of-magnitude improvement in the standard error of the approximation, compared to using an equivalent fixed budget of Monte Carlo sample comparisons without exploiting the sample history. Additionally, we show empirically that multiple orders-of-magnitude computation speed up can be achieved by exploiting the incremental update nature of an approximation set.

This work proceeds as follows. In Section 2 we briefly discuss approaches for the calculation and estimation of the hypervolume, and in Section 3 we detail the need to measure the hypervolume in real-time. Section 4 introduces a method for efficient hypervolume estimation exploiting the estimation history in previous time-steps, along with some empirical results with a fixed budget of samples per time step. Section 5 introduces a procedure for dynamically

[1] Note, it may not contain correspondingly all Pareto optimal solutions, if there are duplicate objective vector mappings from the design space.

changing the number of samples given a minimum CPU time threshold, and provides an empirical assessment of its behaviour. The paper concludes with Section 6, which summarises the advances made in this work, and also identifies potential routes for even further efficiency gains, exploiting recent data structure research on maintaining non-dominated sets, and on efficient non-dominated sorting and updating.

2 COMPUTING THE HYPERVOLUME MEASURE

Given k objectives to minimise (without loss of generality), associated with a given solution (or design) $\mathbf{x}$, solutions may exist for which performance in one objective cannot be improved without reducing performance in at least one other.[2] Such solutions are said to be *Pareto optimal*. The *Pareto set* is the set of all such Pareto optimal solutions, whose image in objective space is known as the *Pareto front*. Identifying such solutions relies on Pareto *dominance*. A solution $\mathbf{x}$ is said to dominate another $\mathbf{x}'$ solution, iff it is better or equal in all objectives under the k-dimensional objective vector function $\mathbf{f}$, and better in at least one. This can be denoted as $\mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}')$. A solution weakly dominates another if it is better or equal on all objectives, denoted as $\mathbf{f}(\mathbf{x}) \preceq \mathbf{f}(\mathbf{x}')$. Such relationships may be extended to sets. With a slight abuse of the objective function notation, $\mathbf{f}(X) \prec \mathbf{f}(X')$ if each member of X' is dominated by at least one member of X, and $\mathbf{f}(X) \preceq \mathbf{f}(X')$ if each member of X' is weakly-dominated by at least one member of X.

2.1 Exact calculation

A number of works have developed methods for the exact calculation of the hypervolume dominated by an approximation set A, given a (dominated) reference point $\mathbf{u}$. Recent work in this area includes: [2, 5, 9, 12, 17, 20, 24]. These exact methods inevitably involve recursive slicing of the hypervolume to calculate the parts dominated by A.

The current state-of-the-art for worst case complexity when $k \geq 4$ is Chan's algorithm [5], which has a best worst case complexity of $O(|A|^{\frac{k}{3}}\texttt{polylog}|A|)$, although there appears to be no implementation currently available to assess its behaviour in practise [17]. In practical terms the algorithms of While et al. [24], Russo and Francisco [20] and the non-incremental version of Lacour et al. [17] are some of the most efficient for $k \geq 4$, but their worst case complexities are respectively $O(|A|^k)$, $O(2^{k(|A|-1)})$ and $O(|A|^{\lfloor \frac{k-1}{2}+1 \rfloor})$. Recent work by Jaszkiewicz has further developing the algorithm of [20] and has a worst case time complexity of $O(k^{|A|-1})$.

Although there have been notable improvements in exact hypervolume calculation in the last decade, it is still the case that for high k and/or $|A|$ its calculation can be prohibitively expensive as a real-time cost. This has resulted in researchers using *approximate* approaches (including recent work on hybridised approaches [22]).

2.2 Estimation

As the exact calculation of the hypervolume scales poorly with the number of objectives, it is often infeasible to calculate exactly, and instead it can be approximated via Monte Carlo (MC) sampling (as used in e.g. [1, 4, 8]).

In this case, the hyperrectangle to be sampled requires the definition of the reference point $\mathbf{u}$ (a vector of maximal values) as well as the definition of the lower bounds, which will form the components of $\mathbf{l}$. The hyperrectangle is then $[\mathbf{l}, \mathbf{u}]$. Once these two vectors are provided, the axis-parallel region bounded by them can be sampled. The hypervolume is estimated by counting the proportion of samples in this region weakly dominated by A. More formally, given n samples, drawn uniformly from $\mathbb{R}^k$ within the hyperrectangle defined by $[\mathbf{l}, \mathbf{u}]$, stored in Y:

$$Y = \{\mathbf{y} \sim \mathcal{U}([\mathbf{l}, \mathbf{u}])\}_{i=1}^{n}, \tag{1}$$

the estimated hypervolume is the proportion of samples weakly dominated:

$$H_{est} = \frac{1}{n}\sum_{i=1}^{n} I(\mathbf{f}(A) \preceq \{Y_i\}), \tag{2}$$

where Y_i is the ith objective vector sample in Y. The function $I()$ returns 1 if the argument is true, 0 otherwise. As standard with MC approaches, the error of this estimate is proportional to $1/\sqrt{n}$, so a reduction in error by a factor of 10 requires and increase in samples by a factor of 100. Note, the standard error is not dependent on k, making this approach especially attractive when k is large (many-objective problems).

3 NEED FOR REAL-TIME HYPERVOLUME

The use of the hypervolume measure can occur at a number of different stages when employing a multi-objective evolutionary algorithm (MOEA). It is often used to compare the (estimated) Pareto front qualities between algorithms, and the variation in performance over multiple runs of a single algorithm. As this assessment occurs *after* the optimisation process we consider this not to be a real-time use of the measure. However, there are a number of situations where the hypervolume is (or can) be employed in real-time (i.e. *during* an optimisation process):

(1) as part of a selection mechanism employed *within* an algorithm during an optimisation run (of which HypE [1] and SMS-EMOA [3] are prominent examples);
(2) incorporated in an automatic stopping criteria (i.e. as a convergence measure);
(3) as a means of real-time progress analysis of the optimiser by the practitioner.

Where $|A|$ is large and/or many objectives are employed, only MC estimation of the hypervolume may be reasonably employed for such real-time applications.

We focus on the use of unconstrained archives here. If the archive is constrained, its quality will often deteriorate. This was first shown theoretically in [11] and empirically in [8]. The impact of constrained archives on hypervolume, leading to non-monotonic improvement in an algorithm run when the reference point was altered during search, was shown in e.g. [14]. Most recently the detrimental impact on hypervolume of eight popular constrained archiving approaches was investigated in [19]. As such, using a constrained archive for convergence analysis and progress analysis, and/or varying $\mathbf{u}$ and $\mathbf{l}$ during such analyses, can be problematic.

[2] Depending on the problem, $\mathbf{x}$ may be a p-dimensional vector of real values, a permutation, a vector of discrete values, or indeed a combination of types.

4 EFFICIENT HYPERVOLUME ESTIMATION OVER TIME

As we mentioned in [7], efficiency gains in MC estimation of the hypervolume over time may be obtained by retaining a memory of those samples non-dominated at the previous time step (though we did not provide a formal description of such a process in [7]). We detail a number of improvements and extensions to this idea in the following sections.

Consider that we draw n MC samples Y^1 at the first time step (or algorithm generation), $t = 1$ (see (1)). We compare Y^1 to the archive at that time step, A^1, and can calculate H^1_{est}, as in (2). A common approach at the next time step is to draw a new set of n samples, to create Y^2, and compute H^2_{est} on these. When A is constructed as an unconstrained archive (as an active archive used in the search, or a passive archive to trace progress [23]), those MC samples dominated by A^t will *always* be dominated at A^{t+1} (and later time steps). This is because A^{t+1} contains all non-dominated solutions from $A^t \cup P^{t+1}$ (where P^{t+1} is the population of new solutions generated by the optimisation algorithm at generation $t+1$). Therefore, if we observe $H^t_{est} > H^{t+1}_{est}$ this can only be due to estimation error (as an *actual* degradation of hypervolume is not possible with an unconstrained A).

As time progresses we can construct a H^t_{est} whose expected error monotonically reduces with each time step, even given a fixed budget of MC samples to compare to, by exploiting the information of samples dominated in previous time steps.

Let us define Y^t_{dom} as the MC sampled points *dominated* by the archive at time t. At $t = 1$, $Y^1_{dom} \subseteq Y^1$, so at $t = 2$, $m = |Y^1_{dom}|$ samples have already been identified as being dominated by A^2. Correspondingly we may define Y^t_{nd} as the subset of Y^t which contains all samples *not dominated* by A^{t-1}, i.e. $Y^t \setminus Y^t_{dom}$. We may therefore construct Y^2 as:

$$Y^2 = Y^1_{nd} \cup \{\mathbf{y} \sim \mathcal{U}([\mathbf{l}, \mathbf{u}])\}^m_{i=1}, \tag{3}$$

which ensures $|Y^2| = n$. The hypervolume at $t = 2$ may be estimated as:

$$H^2_{est} = \frac{1}{n+m}\left(m + \sum_{i=1}^{n} I\left(\mathbf{f}(A^2) \preceq \{Y^2_i\}\right)\right). \tag{4}$$

We can do this without biasing our search as Y^2 contains unbiased samples from the hyperrectangle defined by $\mathbf{l}$ and $\mathbf{u}$, plus the proportion of Y^1 that was not dominated at $t = 1$, which is balanced by the m weighting term for the corresponding dominated portion of Y^1. As Y^1 was itself a set of unbiased samples, H^2_{est} is now also an unbiased estimate, but its accuracy is based on $n + m$ samples rather than n samples, although it only requires comparison of A^2 to n samples at time step 2.

We may generalise (4) for arbitrary t. Let us denote by m^t the number of samples which were *first* dominated at time step t. We denote by M^t the sum of m^t including time t, i.e. $M^t = \sum^t_{\tau=1} m^\tau$. We may construct Y^{t+1} as:

$$Y^{t+1} = Y^t_{nd} \cup \{\mathbf{y} \sim \mathcal{U}([\mathbf{l}, \mathbf{u}])\}^{m^t}_{i=1}. \tag{5}$$

Also, as:

$$M^{t+1} = M^t + \sum_{i=1}^{n} I\left(\mathbf{f}(A^{t+1}) \preceq \{Y^{t+1}_i\}\right) \tag{6}$$

the hypervolume at $t+1$ may be estimated as:

$$H^{t+1}_{est} = \frac{M^{t+1}}{|Y^{t+1}_{nd}| + M^{t+1}}. \tag{7}$$

This procedure may be made even more efficient given the set properties. The size of the subset of the MC samples Y^{t+1} which are not dominated at the previous time step, $|Y^t_{nd}|$, is $n - m^t$. As $\mathbf{f}(A_t)$ fails to dominate any members of Y^t_{nd}, it is wasteful to compute $I(\mathbf{f}(A^{t+1}) \preceq \{Y^t_{nd,i}\})$, as many elements of A^{t+1} are likely to be the same as in A^t. Instead, only $I(\mathbf{f}(A^{t+1} \setminus A^t) \preceq \{Y^t_{nd,i}\})$ need be calculated. Indeed, if $A^{t+1} = A^t$ then no comparisons against the members of Y^t_{nd} are required *at all* at $t+1$. In this further refinement the (estimated) hypervolume at $t+1$, M^{t+1} may be calculated as

$$M^{t+1} = M^t + \sum_{i=1}^{m^t} I\left(\mathbf{f}(A^{t+1}) \preceq \{\mathbf{y} \sim \mathcal{U}([\mathbf{l}, \mathbf{u}])\}\right) + \sum_{i=1}^{n-m^t} I\left(\mathbf{f}(A^{t+1} \setminus A^t) \preceq \{Y^t_{nd,i}\}\right) \tag{8}$$

At most n samples are compared against elements of the archive at $t+1$. Furthermore, only m^t new MC samples are drawn at each time step. As such, this is a *capped* sampling approach.

We now compare the empirical performance and accuracy of the approaches detailed in (6)–(8), along with a baseline which takes n samples each time step and does not exploit past sample history.

4.1 Empirical assessment of incremental updating

To illustrate the benefit of the proposed methods we employ a simple (1+1)-Evolution Strategy. These results are therefore for an algorithm which proposes and evaluates only a single new design at each iteration. The update procedure outlined is however generalisable to population-based approaches.

The optimiser is based on the PAES approach of [15], however rather than using a gridded constrained archive, an unconstrained archive is used, which is stored in a Dominance Decision Tree for efficient comparison [21]. The parent has a single design variable mutated with Gaussian noise, with width 0.1, and rejection sampling is used on mutations which lead to boundary violations. If the child is not weakly dominated by A the child replaces the parent at the next generation.

We run this optimiser for 100,000 iterations on the DTLZ2 test problem [6]. We choose this problem for illustration as there is likely to be a rapid growth in $|A|$ for the optimiser, as the Pareto set of this problem may be traversed by small changes to a single design variable. Therefore it is a good 'stress test' assessment of a worst case excessive archive growth. We use the recommended solution lengths for a the number of objectives used, specifically for $k = \{3, 5, 10, 20\}$ we correspondingly use $|\mathbf{x}| = \{12, 14, 19, 29\}$. We use $\mathbf{l} = \mathbf{0}$ and $\mathbf{u} = \mathbf{2}$ for the hypervolume estimation.

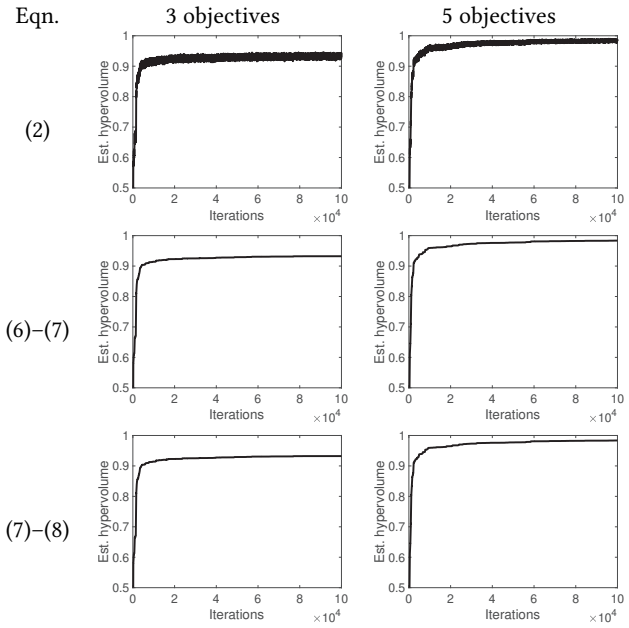

Figure 1: Estimated hypervolume with generation on an indicative run. DTLZ2 for $k = 3$ and $k = 5$, three different estimation approaches.

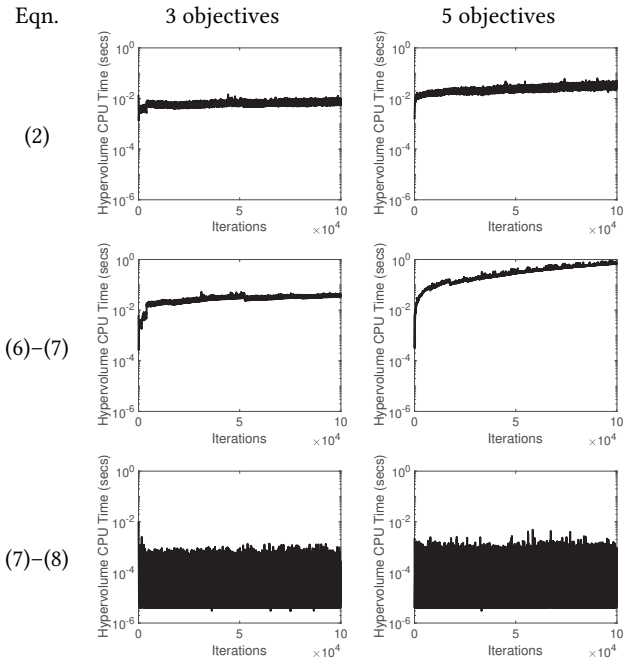

Figure 2: Estimated hypervolume CPU time per generation on an indicative run. DTLZ2 for $k = 3$ and $k = 5$, three different estimation approaches.

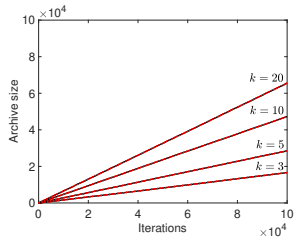

Figure 3: Archive size with generation, DTLZ2, using (7)–(8).

Rather than using a high-performance computing cluster or high-performance desktop, all experiments were performed on a mid-2010 MacBook Pro laptop. This was equipped with a 2.66 GHz Intel Core 2 Duo processor, and 8 GB 1067 MHz DDR3 ram. The results therefore demonstrate where good real-time performance can be achieved even with relatively modest computational resources.

First, we run the three different hypervolume estimation routines detailed in (2), (6)–(7) and (7)–(8) for single indicative runs of $k = 3$ and $k = 5$. We use the same seed across runs, so the progression of A^t is identical for each run for a particular k. Figure 1 shows the estimated hypervolume per generation for each approach. The variation in estimated hypervolume between generations using (2) is clear in the top panels — highlighting that relying soley on $n = 5000$ MC samples per generation is insufficient to provide consistent results. The middle and bottom rows are identical, as both use the same sample sets. Furthermore, the hypervolume increases in a stable fashion, with variation rapidly diminishing due to the exploitation of information in previous samples. Figure 2 shows the corresponding CPU time for hypervolume estimation each generation. There are a few interesting aspects to note here. Although (2) and (6)–(7) compare the same number of samples to A^t at each generation, (6)–(7) progressively has a worse time cost, approaching five times worse by $t = 100000$. This is because, unlike in (2), the samples compared at time steps are *biased* in (6)–(7). Samples in Y^t include those accrued in Y_{nd}^{t-1} which are not dominated at previous time steps. A larger proportion of these are likely not to be dominated at t compared to an equivalent number of unbiased samples from the domain. As non-dominated samples tend to incur a larger comparison cost to A^t (as they require comparison to multiple elements of A^t before they are identified as continuing to be non-dominated, even with efficient data structures), the overall time cost is higher for the same number of samples. However, by the same measure (7)–(8) is orders-of-magnitude faster than (2) and (6)–(7) as the Y_{nd}^{t-1} samples in Y^t are only compared to the new archive entrant, which is substantially cheaper than comparing to the entire archive.

We next repeat our experiments 50 times, choosing a different random seed for the Monte Carlo samples, but the *same* random seed for the algorithm. This enables us to isolate variability due to the hyervolume estimation process itself, rather than due to the stochastic nature of the optimiser, and due to the time costs involved, we focus on the approach in (7)–(8).

Figure 3 shows how the archive grows in size over time on DTLZ2 as the number of objectives is increased. This is linear with iteration, and the gradient increases from roughly 0.17 for $k = 3$ through to over 0.6 for $k = 20$. Figure 4 shows the estimated hypervolume across the 50 runs. The mean, minimum and maximum are

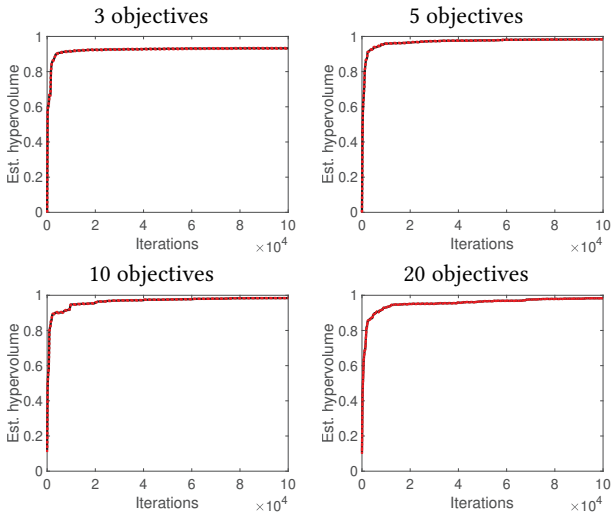

Figure 4: Estimated hypervolume with generation, DTLZ2, using (7)–(8). Mean over 50 runs in black, maximum and minimum in red.

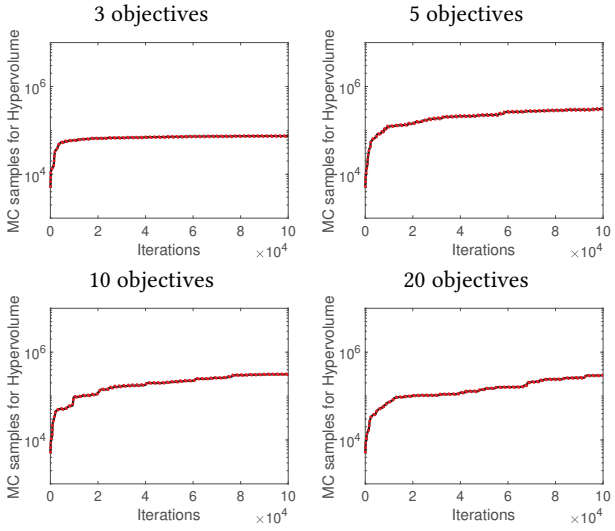

Figure 5: Cumulative number of MC samples with generation, DTLZ2, using (7)–(8). Mean over 50 runs in black, maximum and minimum in red.

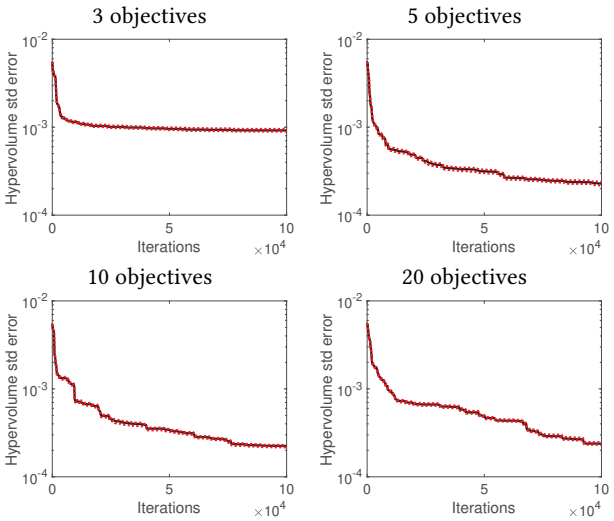

Figure 6: Estimated hypervolume standard error with generation, DTLZ2, using (7)–(8). Mean over 50 runs in black, maximum and minimum in red.

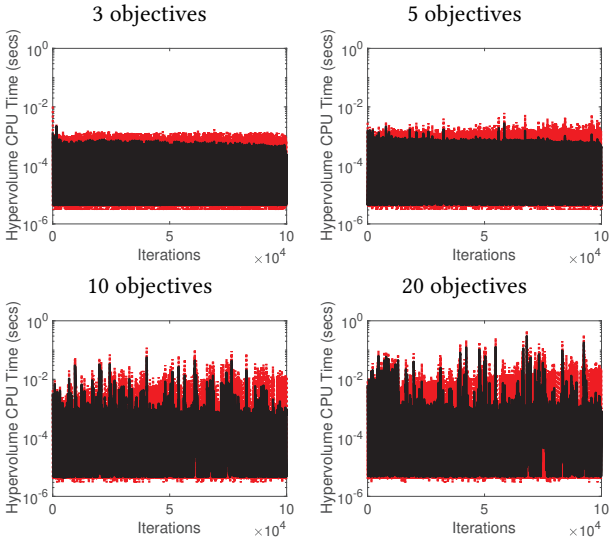

Figure 7: Estimated hypervolume CPU time per generation, DTLZ2, using (7)–(8). Mean over 50 runs in black, maximum and minimum in red.

plotted, however they are rapidly indistinguishable at this plotting resolution. Figure 5 shows how the number of MC samples grows over time and Figure 6 gives the corresponding standard error of these estimates, which can bee seen to monotonically decrease, and is consistent across runs.

Figure 7 shows the CPU time dedicated to the hypervolume estimation per generation. As can be seen, the timings are quite volatile, and are influenced by the degree to which comparisons are being made to either (i) A^t (new MC samples), or (ii) to the new entrant to A^t alone (i.e. when comparing to samples in Y_{nd}^{t-1}), at a particular time step. The vast majority of hypervolume estimate updates are seen to take below 10^{-2} seconds. The mean CPU time per iteration across time steps required for each of the problem sizes was 0.065ms for $k = 3$, 0.17ms for $k = 5$, 0.76ms for $k = 10$ and 1.9ms for $k = 20$. Figure 8 shows the (cumulative) mean number of MC samples compared to the archive as a whole (red) and to the single new archive entrant (black), exploiting the incremental nature of the archive update. For all objective dimensions there was over an order of magnitude fewer comparisons to the archive

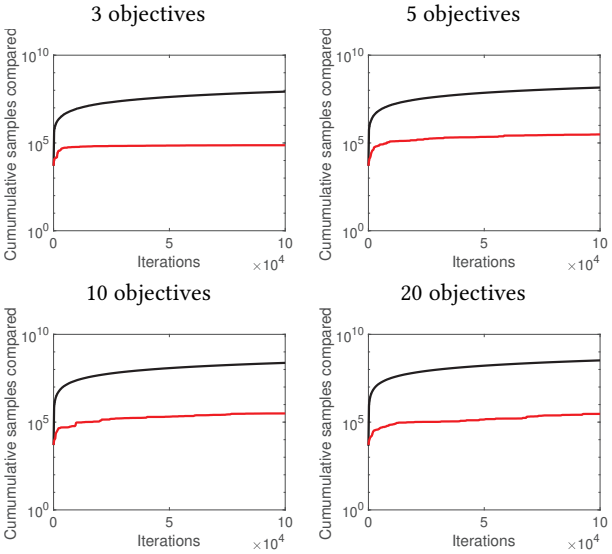

Figure 8: Cumulative number of sample comparisons to A (mean in red), and to a single new A element (mean in black). 50 runs on DTLZ2, using (7)–(8).

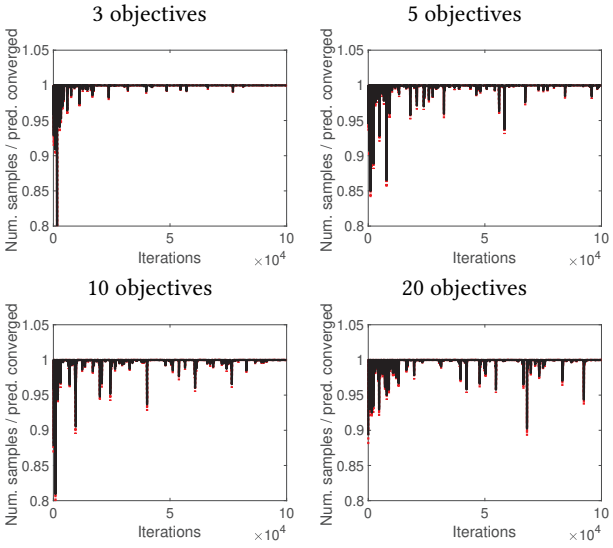

Figure 9: Convergence to $(M^t + |Y_{nd}^{t-1}|)/(n/(1 - H_{est}^t)) = 1$ (mean in black, maximum and minimum in red). 50 runs on DTLZ2.

compared to a single new entrant, which, given the archive growth shown in Figure 3 demonstrates the significant computational gains over the approach outlined in [7] (also shown earlier in Figure 2).

5 DYNAMIC SAMPLE SIZES

Statistically, the effect of the process described in (8) is for the *total* number of MC samples, $M^t + |Y_{nd}^{t-1}|$, to converge such that eventually n samples will be non-dominated by the set of solutions in the archive. This effect occurs because if the membership of A does not change for any prolonged period of time $m^t \rightarrow 0$, and $|Y_{nd}^t| \rightarrow n$, none of which are compared at t as $A^t \setminus A^{t-1} = \emptyset$. The converged value of $M^t + |Y_{nd}^{t-1}|$ is therefore $n/(1 - H_{est}^t)$. Figure 9 shows $(M^t + |Y_{nd}^{t-1}|)/(n/(1 - H_{est}^t))$, illustrating clear and rapid convergence to the predicted value each time the archive updates.

In the case where $A^{t+1} \setminus A^t = \emptyset$, i.e. when an archive is unchanged between time steps, no MC samples will be compared to $\mathbf{f}(A^{t+1})$ *at all* if $|Y_{nd}^{t-1}| = n$. Therefore, not only will the hypervolume estimate fix on a value, there will be no improvement in the accuracy of this value as time progresses. This stagnation behaviour may be viewed as undesirable.

In response, we now investigate a mechanism for incremental hypervolume estimation over time, which attempts to bracket the computational budget per time step dedicated to the hypervolume estimation. It therefore dynamically varies its sample size and number of comparisons per time step given this budget. This requires some additional book-keeping, but maintains the monotonically improving property of the hypervolume estimation previously described. It has the added advantage that, in situations where the archive is unchanged, the fidelity of the estimated hypervolume will *always* improve.

5.1 A ceiling on total hypervolume CPU time

We conducted initial investigations on limiting the total time dedicated to hypervolume estimation at each time step to a maximum threshold (i.e. the combined time spent comparing to both $|Y_{nd}^t|$ and new MC samples could not exceed a given value). This required maintaining a list of archive elements yet to be fully compared to Y_{nd}^t, and how far through Y_{nd}^t each had been compared. However, it became apparent such an approach rapidly led to processed archive member backlogs and stale hypervolume estimates when the time cost of comparing to all members of Y_{nd}^t would exceed the time threshold. This was only alleviated when substantial periods of static archive membership occurred, however this typically was not sufficient to clear the backlog. Given space limitations we do not provide an in depth analysis here, but highlight to the reader the inherent issues with such an approach.

5.2 A ceiling on hypervolume CPU applied after comparison to Y_{nd}^t completes

The second approach we considered was applying CPU time cost ceiling *after* $\mathbf{f}(A^{t+1} \setminus A^t)$ is compared to Y_{nd}^t. While a predefined time threshold κ is not reached new MC samples are repeatedly taken (the clock is started prior to any comparison of $\mathbf{f}(A^{t+1} \setminus A^t)$ to Y_{nd}^t). This ensures that H_{est}^{t+1} accurately reflects the hypervolume dominated by A^{t+1}, given the number of samples accrued. However when $A^{t+1} \setminus A^t \neq \emptyset$ and $|Y_{nd}^t|$ is large then the computational time expended on comparison to $|Y_{nd}^t|$ may be larger than κ, and therefore there is no guaranteed upper bound on *overall* hypervolume CPU cost per time step in this approach. The procedure is outlined in Algorithm 1

We set $\kappa = 0.1$ms. This resulted in a mean CPU time (over 50 runs) per iteration required for each of the problem sizes of 0.86ms

Algorithm 1 Dynamic (minimum time) hypervolume estimation.

Require: κ ▷ Minimum time to dedicate to each estimation
$P^0 \leftarrow$ `initialise_state()` ▷ Get initial optimiser population
$A^0 \leftarrow \emptyset$ ▷ Initial empty archive
$M \leftarrow 0$ ▷ Number of dominated samples
$Y_{nd} \leftarrow \emptyset$ ▷ Non-dominated set of Monte Carlo samples
$t \leftarrow 1$
$L \leftarrow \emptyset$ ▷ List of new archive members to compare to samples
$I \leftarrow \emptyset$ ▷ Indices to track processing of samples to lest members
while stopping condition(s) not met **do**
  $start\_time \leftarrow$ `get_time()` ▷ Get current time
  $domed\_samples \leftarrow 0$ ▷ To track number dominated at t
  $P^t \leftarrow$ `increment_optimiser`(P^{t-1}, A^{t-1}) ▷ Step optimiser
  $A^t \leftarrow$ `nondom`$(P^t \cup A^{t-1})$ ▷ Update approximate Pareto set
  $A_{new} \leftarrow A^t \setminus A^{t-1}$
  if $A_{new} \neq \emptyset$ **then** ▷ Archive has new member(s)
    for $i \leftarrow |Y_{nd}|, \ldots, 1$ **do**
      if $\mathbf{f}(A_{new}) \preceq \{Y_{nd,i}\}$ **then**
        $domed\_samples \leftarrow domed\_samples + 1$
        $Y_{nd} \leftarrow Y_{nd} \setminus \{Y_{nd,i}\}$
      end if
    end for
  end if
  while `get_time()` $- start\_time < \kappa$ **do**
    $\mathbf{s}$ = `monte_carlo_sample()` ▷ New sample
    if $\mathbf{f}(A^t) \not\preceq \{\mathbf{s}\}$ **then**
      $Y_{nd} \leftarrow Y_{nd} \cup \{\mathbf{s}\}$ ▷ Add non-dominated sample
    else
      $domed\_samples \leftarrow domed\_samples + 1$
    end if
  end while ▷ Out of time
  $M \leftarrow M + domed\_samples$
  $H_{est} \leftarrow \frac{M}{|Y_{nd}|+M}$
  $t \leftarrow t + 1$
end while

for $k = 3$, 0.46ms for $k = 5$, 0.86ms for $k = 10$ and 0.82ms for $k = 20$. Figure 10 shows how the hypervolume approximation changes over time. Figure 11 shows how the number of MC samples grows for each of the problem variants. Note the number drops as the number of objective increases, as the computational cost per dominance comparison rises with k. This effect is not seen in the corresponding Figure 5 where the sample number is budgeted rather than the computational cost. Figure 12 shows how the standard error progresses over time, note the descent is much more smooth than the corresponding set of results in Figure 6, due to the $\approx \kappa$ CPU time spent on new MC samples when $A^t \setminus A^{t+1} = \emptyset$. Figure 13 shows the CPU time spent at each time step calculating the estimated hypervolume. The lower bound of the $\kappa = 0.1$ms is clear. It is interesting to note that for these particular runs, with archives reaching over 60,000 members, and accruing over 1,500,000 MC samples, the CPU cost per time step for hypervolume estimation is clearly bracketed between 0.1–10ms per time step. For completeness, Figure 14 shows the (cumulative) number of times a sample is compared to the entire archive, or to just the most recent update to the archive. As with

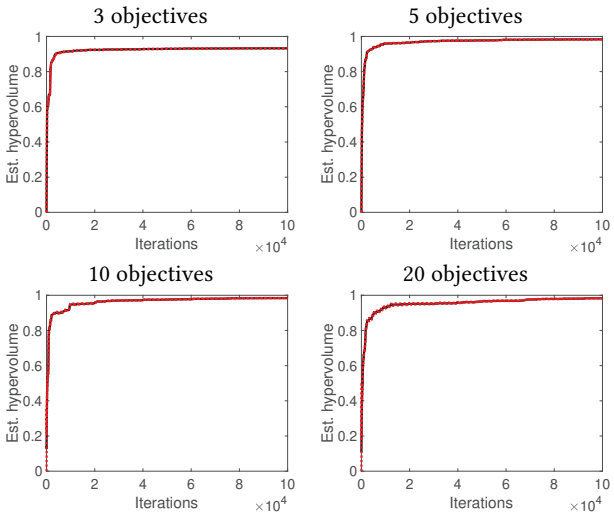

Figure 10: Estimated hypervolume with generation, DTLZ2. Mean over 50 runs in black, maximum and minimum in red.

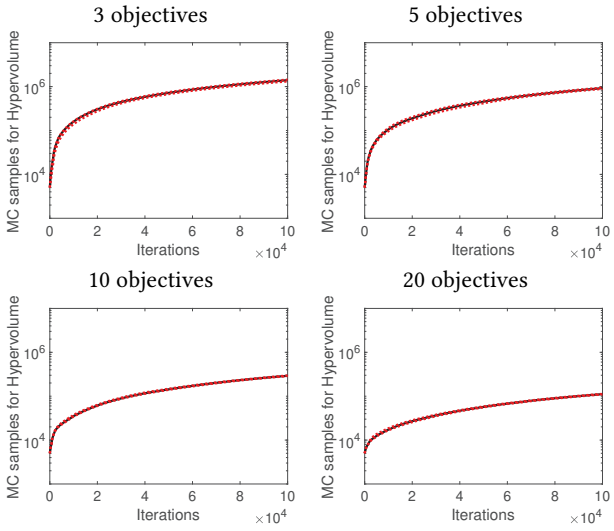

Figure 11: Cumulative number of MC samples with generation, DTLZ2. Mean over 50 runs in black, maximum and minimum in red.

the previous approach, there are a couple of orders-of-magnitude fewer of the former compared to the latter.

Java implementations of all the approaches detailed here are available at https://github.com/fieldsend.

6 DISCUSSION

We have detailed a new approach for incrementally updating the estimated hypervolume when maintaining an unconstrained Pareto approximation set of solutions during multi- and many-objective optimisation. This approach is shown to have monotonically reducing approximation error. We detail methods which provide multiple

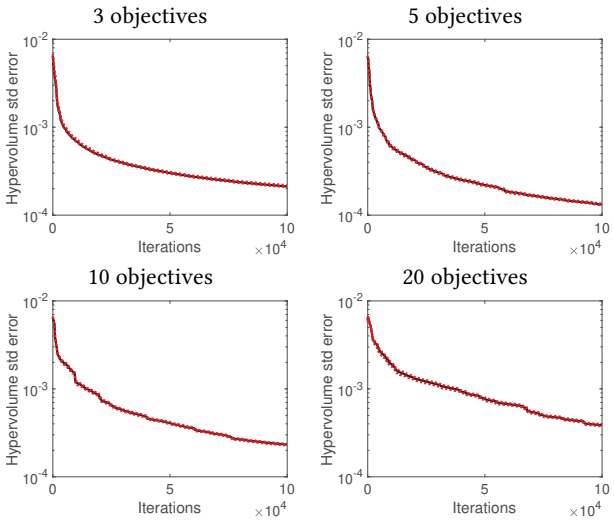

Figure 12: Estimated hypervolume standard error with generation, DTLZ2. Mean over 50 runs in black, maximum and minimum in red.

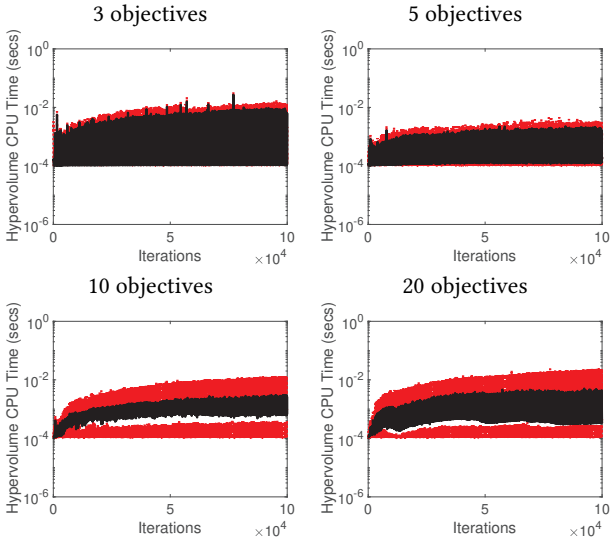

Figure 13: Estimated hypervolume CPU time per generation, DTLZ2. Mean over 50 runs in black, maximum and minimum in red.

orders-of-magnitude speed up using such an approach. We show results with two different formulations, either bounding the number of samples compared to each generation, or fixing a time cost limit on the number of *new* MC samples to be compared to. We provide an illustration where the approximation set grows to contain over 60k members, and the hypervolume accuracy is based on 1.5M samples, but the hypervolume estimation is still roughly 1ms per algorithm generation, on a modest laptop computer.

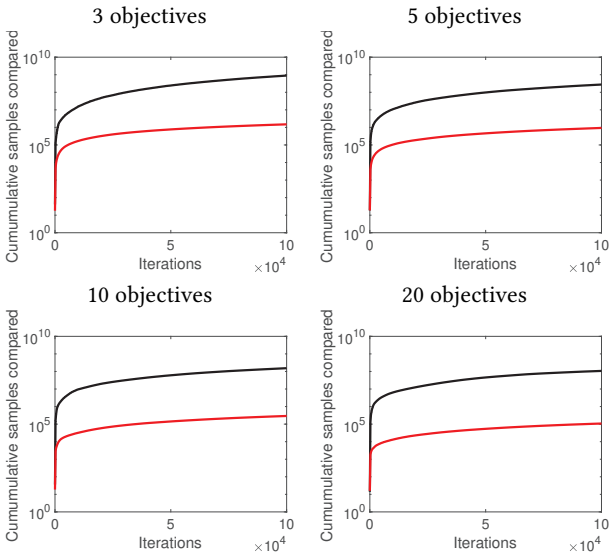

Figure 14: Cumulative number of sample comparisons to A (mean in red), and to a single new A element (mean in black). 50 runs on DTLZ2.

Although we limit our empirical work here to run-time and convergence analysis of an algorithm, we note that, subject to the upper and lower bounds of the sampled domain being fixed *a priori*, it may also be employed effectively within selection mechanisms of algorithms to increase hypervolume estimation fidelity (assuming an unconstrained archive is kept).

The Dominance Decision Tree data structure [21] has proved effective for storing the approximation set for weak-dominance queries in the hypervolume estimation. Nevertheless, recently developed data structures for this task may speed these calculations even further (e.g. [10, 13]), and we look forward to integrating these as alternative storage choices for the archive into our approach.

A potentially even greater efficiency gain may be achievable by storing and updating Y^t_{nd} in a sorted fashion. Currently all of Y^t_{nd} is compared to a new archive element, however, if Y^t_{nd} were sorted into non-dominated fronts with inverted objectives, then an attractive approach would be to process each front in turn (from the back) and stop processing once a front has been reached in Y^t_{nd} where no members have been dominated (as by construction, no members of any fronts ahead of this can possibly be dominated). Recent work has shown how to update a set of non-dominated fronts efficiently in a steady state fashion [18], which may be amenable for use in the context of this work also. However, maintaining such a sorted structure has itself a computation cost, and the trade-off between this additional cost and the computational gain of processing fewer elements of Y^t_{nd} per archive update will be interesting to explore.

ACKNOWLEDGMENTS

This work was supported by Innovate UK [grant number 104400] and the Engineering and Physical Sciences Research Council [grant number EP/N017846/1]. The author would like to sincerely thank the reviewers for their helpful and constructive comments.

REFERENCES

[1] Johannes Bader and Eckart Zitzler. 2011. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary computation* 19, 1 (2011), 45–76.

[2] Nicola Beume, Carlos M Fonseca, Manuel López-Ibáñez, Luís Paquete, and Jan Vahrenhold. 2009. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation* 13, 5 (2009), 1075–1082.

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

[4] Karl Bringmann, Tobias Friedrich, Christian Igel, and Thomas Voß. 2013. Speeding up many-objective optimization by Monte Carlo approximations. *Artificial Intelligence* 204 (2013), 22–29.

[5] Timothy M Chan. 2013. Klee's measure problem made easy. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 410–419.

[6] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. 2005. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary multiobjective optimization*. Springer, 105–145.

[7] Jonathan E Fieldsend. 2017. University Staff Teaching Allocation: Formulating and Optimising a Many-Objective Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 1097–1104.

[8] Jonathan E Fieldsend, Richard M Everson, and Sameer Singh. 2003. Using unconstrained elite archives for multi-objective optimization. *IEEE Transactions on Evolutionary Computation* 7, 3 (2003), 305–323.

[9] Carlos M Fonseca, Luís Paquete, and Manuel López-Ibánez. 2006. An improved dimension-sweep algorithm for the hypervolume indicator. In *2006 IEEE international conference on evolutionary computation*. IEEE, 1157–1163.

[10] Tobias Glasmachers. 2017. A Fast Incremental BSP Tree Archive for Non-dominated Points. In *EMO 2017 (LNCS)*, H. Trautmann et al. (Ed.), Vol. 10173. Springer, 252–266.

[11] Thomas Hanne. 1999. On the convergence of multiobjective evolutionary algorithms. *European Journal of Operational Research* 117, 3 (1999), 553–564.

[12] Andrzej Jaszkiewicz. 2018. Improved Quick Hypervolume Algorithm. *Computers & Operations Research* 90, C (Feb. 2018), 72–83.

[13] Andrzej Jaszkiewicz and Thibaut Lust. 2018. ND-Tree-Based Update: A Fast Algorithm for the Dynamic Nondominance Problem. *IEEE Transactions on Evolutionary Computation* 22, 5 (2018), 778–791.

[14] Leonard Judt, Olaf Mersmann, and Boris Naujoks. 2013. Non-monotonicity of Observed Hypervolume in 1-Greedy S-Metric Selection. *Journal of Multi-Criteria Decision Analysis* 20, 5-6 (2013), 277–290.

[15] Joshua Knowles and David Corne. 1999. The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, Vol. 1. IEEE, 98–105.

[16] Joshua D Knowles, Lothar Thiele, and Eckart Zitzler. 2006. *A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers*. 214. Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland. revised version.

[17] Renaud Lacour, Kathrin Klamroth, and Carlos M Fonseca. 2017. A box decomposition algorithm to compute the hypervolume indicator. *Computers & Operations Research* 79 (2017), 347–360.

[18] Ke Li, Kalyanmoy Deb, Qingfu Zhang, and Qiang Zhang. 2017. Efficient nondomination level update method for steady-state evolutionary multiobjective optimization. *IEEE Transactions on Cybernetics* 47, 9 (2017), 2838–2849.

[19] Miqing Li and Xin Yao. 2019. An Empirical Investigation of the Optimality and Monotonicity Properties of Multiobjective Archiving Methods. *Proceedings of Evolutionary Multi-Criterion Optimization (EMO)* (2019).

[20] Luís MS Russo and Alexandre P Francisco. 2014. Quick hypervolume. *IEEE Transactions on Evolutionary Computation* 18, 4 (2014), 481–502.

[21] Oliver Schütze. 2003. A New Data Structure for the Nondominance Problem in Multi-objective Optimization. In *International Conference on Evolutionary Multi-Criterion Optimization, EMO 2003 (LNCS)*, Vol. 2632. Springer, 509–518.

[22] Weisen Tang, Hailin Liu, and Lei Chen. 2017. A Fast Approximate Hypervolume Calculation Method by a Novel Decomposition Strategy. In *International Conference on Intelligent Computing*. Springer, 14–25.

[23] David A Van Veldhuizen and Gary B Lamont. 2000. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary computation* 8, 2 (2000), 125–147.

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

[25] Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. 2007. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 862–876.