A Fast Way of Calculating Exact Hypervolumes

Lyndon While, *Senior Member, IEEE,* Lucas Bradstreet, *Member, IEEE,* and Luigi Barone

***Abstract*—We describe a new algorithm WFG for calculating hypervolume exactly. WFG is based on the recently-described observation that the exclusive hypervolume of a point p relative to a set S is equal to the difference between the inclusive hypervolume of p and the hypervolume of S with each point limited by the objective values in p. WFG applies this technique iteratively over a set to calculate its hypervolume. Experiments show that WFG is substantially faster (in five or more objectives) than all previously-described algorithms that calculate hypervolume exactly.**

***Index Terms*—Diversity, evolutionary computation, hypervolume, multiobjective optimization, performance metrics.**

I. Introduction

HYPERVOLUME [1], also known as the S-metric [2] or the Lebesgue measure [3], is a popular metric for comparing the performance of multiobjective optimizers. The hypervolume of a set of solutions measures the size of the portion of objective space that is dominated by those solutions collectively. Hypervolume captures in one scalar both the closeness of the solutions to the optimal set and their spread across objective space. Hypervolume also has nicer mathematical properties than other metrics [4], [5]; however, it is sensitive to the relative scaling of the objectives, and to the presence or absence of extremal points. Hypervolume is also increasingly used in-line in multiobjective evolutionary algorithms, either to promote diversity [6], or as part of an archiving mechanism [7], [8], or as part of the selection process [9], [10]. Here, the requirement is usually to determine which point in a set contributes least to the hypervolume of the set.

The principal problem with hypervolume is that it is expensive to calculate. Several algorithms have been proposed; significant recent developments include:

1) the HOY algorithm [11], that has by far the best worst-case complexity of any algorithm ($O(m \log m + m^{n/2})$, where m is the number of points and n is the number of objectives);
2) a provably optimal $O(m \log m)$ algorithm [12] for the 3-D case;
3) the IIHSO algorithm [13], currently the fastest algorithm on much typical benchmark data in many objectives.

We describe a new approach to calculating hypervolume exactly, based on a recently-described technique [14], [15] for calculating exclusive hypervolumes. To calculate the exclusive hypervolume of a point p relative to a set S, replace each point q in S with a point that dominates the intersection of q's and p's volumes, then the resulting set dominates a subset of p's inclusive hypervolume, and p's exclusive hypervolume is simply the difference between them. This modification of S often leads to a large proportion of its points becoming dominated, leading to a very fast calculation overall. Applied iteratively with only minor optimizations, this bounding technique gives a new and very simple algorithm WFG that is far faster (in five or more objectives) than any previously-published algorithm for calculating hypervolumes exactly.

The remainder of this paper is structured as follows. Section II describes the necessary background material in multiobjective optimization and hypervolume, including a brief description of previous exact hypervolume algorithms. Section III describes the new technique for calculating exclusive hypervolumes, including some analysis of its benefits. Section IV describes the new algorithm WFG, some ways in which WFG can be optimized, an experimental comparison of WFG with other recent hypervolume algorithms, and an analysis of WFG's complexity. Section V concludes this paper and suggests some future work, including a discussion of using the bounding technique in-line in evolutionary algorithms.

II. Background Material

A. Multiobjective Optimization

In a multiobjective optimization problem, we aim to find the set of optimal tradeoff solutions known as the Pareto optimal set. Pareto optimality is defined with respect to the concept of non-domination between points in objective space. Given two objective vectors $\overline{x}$ and $\overline{y}$, $\overline{x}$ *dominates* $\overline{y}$ iff $\overline{x}$ is at least as good as $\overline{y}$ in all objectives, and better in at least one. A vector $\overline{x}$ is *non-dominated* with respect to a set of solutions X iff there is no vector in X that dominates $\overline{x}$. X is a *non-dominated set* iff all vectors in X are mutually non-dominating. Such a set of objective vectors is sometimes called a *non-dominated front*.

A vector $\overline{x}$ is Pareto optimal iff $\overline{x}$ is non-dominated with respect to the set of all possible vectors. Pareto optimal vectors are characterized by the fact that improvement in any one objective means worsening at least one other objective. The Pareto optimal set is the set of all possible Pareto optimal vectors. The goal in a multiobjective problem is to find the Pareto optimal set, although for continuous problems a representative subset will usually suffice.

Precise definitions of these terms can be found in [16].

B. Hypervolume

Given a set of solutions returned by a multiobjective optimizer, the question arises how well it approximates the Pareto optimal set. One metric used widely for comparing sets of

Manuscript received March 18, 2010; revised June 27, 2010; accepted August 29, 2010. Date of publication May 12, 2011; date of current version January 31, 2012.

The authors are with the School of Computer Science and Software Engineering, University of Western Australia, Perth, WA 6009, Australia (e-mail: lyndon@csse.uwa.edu.au; lucas@csse.uwa.edu.au; luigi@csse.uwa.edu.au).

Digital Object Identifier 10.1109/TEVC.2010.2077298

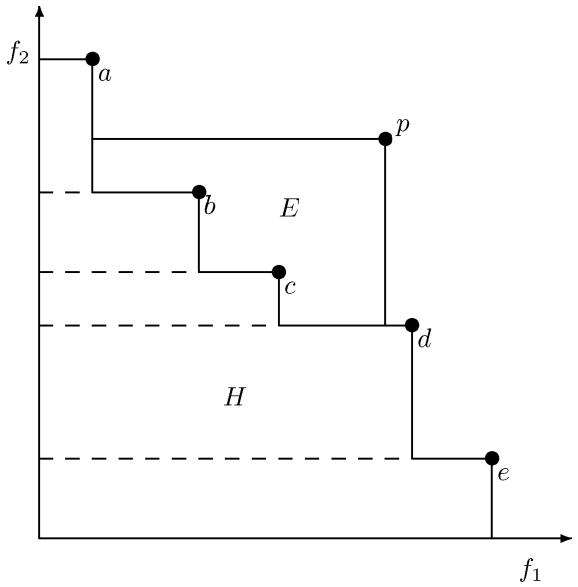


Fig. 1. Maximizing in both objectives relative to the origin, the hypervolume of $\{a, b, c, d, e\}$ is the solid-bordered shape labeled H, the exclusive hypervolume of p relative to $\{a, b, c, d, e\}$ is the shape labeled E, and the inclusive hypervolume of any point is the rectangle bounded by that point and the origin. The dashed lines indicate the sub-parts of H used in (4).

solutions is their *hypervolume* [1]–[3]. The hypervolume of a set S is the size of the part of objective space that is dominated collectively by the solutions in S. The hypervolume of a set is measured relative to a reference point, usually the anti-optimal point or "worst possible" point in space. (We do not address here the problem of choosing a reference point, if the anti-optimal point is not known or does not exist: one suggestion is to take, in each objective, the worst value from any of the sets being compared.) If a set S has a greater hypervolume than a set S', S is taken to be a better set of solutions than S'.

The *exclusive hypervolume* of a point p relative to an *underlying set* S is the size of the part of objective space that is dominated by p but is not dominated by any member of S. Exclusive hypervolume is used widely in multiobjective optimizers, either to promote diversity [6], or as part of an archiving mechanism [7], [8], or as part of the selection process [9], [10]. The requirement is usually to determine which point in a set contributes least to the hypervolume of the set. Exclusive hypervolume can be defined in terms of hypervolume, that is

$$ExcHyp(p, S) = Hyp(S \cup \{p\}) - Hyp(S). \quad (1)$$

We shall also use the term *inclusive hypervolume* of a point p to mean the size of the part of objective space dominated by p alone, that is

$$IncHyp(p) = Hyp(\{p\}). \quad (2)$$

All of these concepts are illustrated in Fig. 1.

C. Previous Algorithms for Calculating Exact Hypervolumes

Several algorithms have been proposed for calculating hypervolumes and exclusive hypervolumes exactly.

The inclusion-exclusion algorithm for calculating the size of a set union has been adapted for hypervolume calculation [17], but its complexity is $O(n2^m)$ as it examines every subset of the set of points, so it is unusable in practice.

LebMeasure [18] processes the points one at a time, calculating the exclusive hypervolume dominated by one point relative to the rest of the set, then discarding that point and processing the others in turn (this idea is discussed further in Section II-D). LebMeasure was originally believed to have polynomial complexity, but it has since been proved to be exponential [19] and even when optimized it is very slow [20].

Hypervolume by slicing objectives (HSO) [20]–[22] processes the objectives one at a time. It slices the nD-hypervolume into separate $n-1$D-hypervolumes through the values in one of the objectives, then it calculates the hypervolume of each slice and sums these values. HSO's worst-case complexity is $O(m^{n-1})$ [20], but good heuristics have been described for re-ordering objectives [23] that deliver much better performance for typical data.

Fonseca, Paquete, López-Ibáñez (FPL) [24] optimizes HSO further, mainly through the use of an advanced data structure to optimize repeated domination checks and the recalculation of partial hypervolumes. It also incorporates a recent optimal algorithm for the 3-D case [12], that works by maintaining a sorted 2-D front in a balanced tree structure as it descends in the third objective. The key to the 3-D algorithm is that this front can be updated in logarithmic time.

Incremental HSO (IHSO) [25] is a version of HSO customized for incremental calculations. It uses various ideas and heuristics to reduce the cost of calculating exclusive hypervolumes (discussed more fully in Section III), and it uses a novel "best-first" queuing mechanism to determine very efficiently the least-contributing point in a set.

Iterated IHSO (IIHSO) [13] combines the efficient incremental calculations of IHSO with the point-wise processing of LebMeasure, again with heuristics to optimize the performance for a given data set. The resulting algorithm is the fastest yet known on much typical benchmark data in many objectives.

Hypervolume by Overmars and Yap (HOY) [11] adapts an algorithm for solving Klee's measure problem [26] to hypervolume calculation. Objective space is divided into regions, each one containing the points that overlap that region. Regions are broken up recursively until the hypervolume within each region can be measured trivially. HOY's worst-case complexity $[O(m \log m + m^{n/2})]$ is by far the best of any algorithm, but experiments reported in [27] and [13] and in this paper show that its performance on realistically-sized fronts is not great.

An obvious alternative to calculating hypervolume exactly is to use an approximation algorithm (for example, [27]–[30]). Using such algorithms introduces a tradeoff between precision and performance, and much improvement has been reported recently on understanding this tradeoff [31]–[33]. However, we do not compare with approximation algorithms here.

D. Calculating Hypervolume Point-Wise

LebMeasure [18] introduced the idea of calculating hypervolume as a summation of exclusive hypervolumes, that is

$$Hyp(\{p_1, \ldots, p_m\}) = \sum_{i=1}^{m} ExcHyp(p_i, \{p_{i+1}, \ldots, p_m\}). \quad (3)$$

Considering Fig. 1 again, this corresponds to calculating the hypervolume of $\{a, b, c, d, e\}$ by summing the regions delineated by the dashed lines, that is

$$\begin{aligned} H = {} & ExcHyp(a, \{b, c, d, e\}) + \\ & ExcHyp(b, \{c, d, e\}) + \\ & ExcHyp(c, \{d, e\}) + \\ & ExcHyp(d, \{e\}) + \\ & ExcHyp(e, \{\}). \end{aligned} \tag{4}$$

This technique is used in IIHSO [13], and we use it again in our new algorithm WFG, described in Section IV.

III. A New Method for Calculating Exact Exclusive Hypervolumes

Equation (1) provides a clear definition for exclusive hypervolume, but it does not provide an efficient mechanism for its calculation: it requires two separate hypervolume calculations. IHSO [25] gets around this by calculating the differences between the two hypervolumes directly, using two main tricks.

1) Slices above the *contributing point* p (i.e., slices with better values than p in the current objective) are discarded: they contain no hypervolume that is dominated by p, so clearly they add nothing to the hypervolume dominated exclusively by p.
2) If, in a given slice, p is dominated in the remaining (unprocessed) objectives, clearly it dominates no more exclusive hypervolume in that slice, or in any lower slices. Those slices too are discarded.

Bringmann and Friedrich [14] and Bradstreet *et al.* [15] described a new way of using the first of these tricks preemptively rather than on-the-fly, by modifying the points in the underlying set. Each point is replaced with one whose value in each objective is limited to be no better than the contributing point. Fig. 2 illustrates the principle.

For the set $\{a, b, c, d, e\}$, each of their objective values is replaced with the smaller of that value and the corresponding value from the contributing point p. The effect is that the hypervolume dominated by the modified underlying set is a subset of the inclusive hypervolume of the contributing point, and a simple subtraction returns the exclusive hypervolume.

This calculation is defined in

$$ExcHyp(p, S) = Hyp(\{p\}) - Hyp(S') \tag{5}$$

where

$$S' = \{limit(s, p) | s \in S\} \tag{6}$$

$$\begin{aligned} & limit(< s_1, \ldots, s_n >, < p_1, \ldots, p_n >) \\ & \quad =< worse(s_1, p_1), \ldots, worse(s_n, p_n) > . \end{aligned} \tag{7}$$

Note that any point in S' that is dominated by some other point in S' (for example, e' in Fig. 2) has no more relevance to the result, and it can be discarded before any further calculation is performed. We can see how crucial this step is to the efficiency of these calculations by plotting the percentage of dominated points in S' for various types of data. Fig. 4 shows that the great majority of sets in 4–10D lose over 50% of their points after being limited by *just one contributing point*, and that most sets lose over 80% of their points. And the optimizations described in Section IV-B increase these percentages significantly.

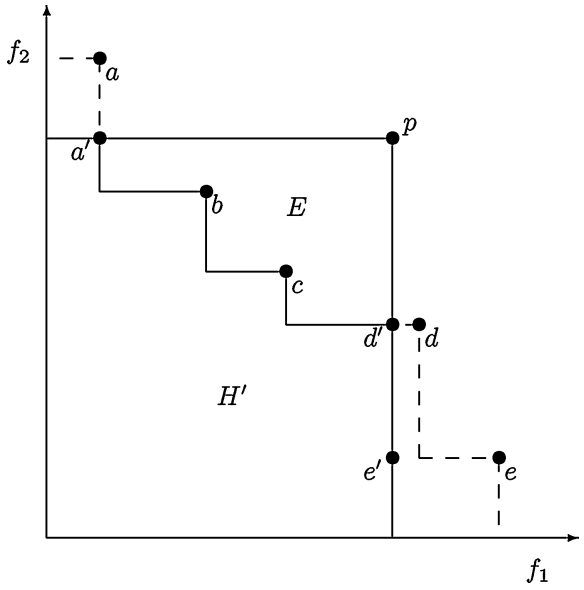


Fig. 2. Maximizing in both objectives, the exclusive hypervolume of p relative to $\{a, b, c, d, e\}$ (i.e., E) = the inclusive hypervolume of p (i.e., the rectangle with p at the top-right corner) minus the hypervolume of $\{a', b, c, d', e'\}$ (i.e., H'). Clearly, e' is dominated by d' and can be discarded.

m	m	m	m	1
$m-1$	$m-1$	$m-1$	1	2
$m-2$	$m-2$	1	2	3
$m-3$	1	2	3	4
$m-4$	2	3	4	5
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	$m-3$	$m-2$	$m-1$	m

Fig. 3. Pathological example for the bounding technique. This pattern describes sets of m points in five objectives, all being maximized and all (except the first) increasing monotonically after the first four points. The pattern can be generalized for other numbers of objectives.

Of course not all data has so many points dominated. The worst case appears to be data of the form shown in Fig. 3, for which no points are dominated in the largest $n-2$ underlying sets.

However note that this does not immediately imply that this data will be processed slowly overall, just that the largest underlying sets will be processed slowly.

Bringmann and Friedrich [14] made the observation that if the contributing point is poor in several objectives, then more of the points in the underlying set will tend to be lost. We use this as a basis for optimizing the performance of WFG in Section IV-B.

IV. WFG: A Fast Algorithm for Calculating Exact Hypervolumes

A. Basic Algorithm

The basic WFG algorithm is a mutually recursive combination of the point-wise calculation from Section II-D,

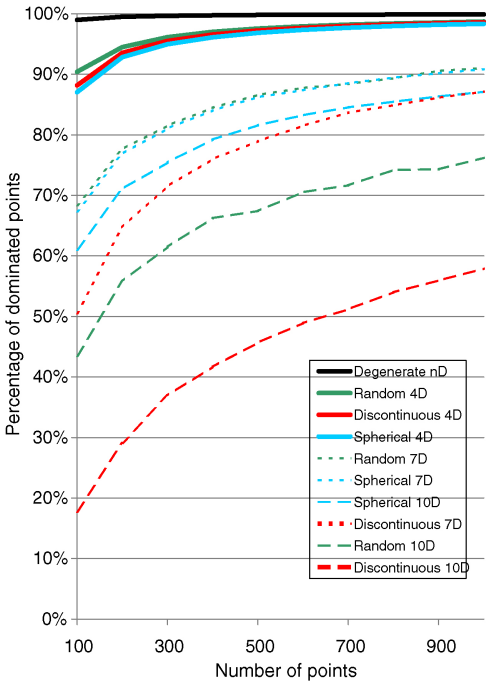


Fig. 4. Percentage of dominated points in a front after it has been limited by one of its points. Each datum is the average of 20 distinct fronts, where for each point in each front we calculate the number of dominated points after the front is limited by that point, then we average those figures. The data is described in Section IV-C.

```
wfg(pl):
  return sum {exclhv(pl, k) | k in {1 .. |pl|}}

exclhv(pl, k):
  return inclhv(pl[k]) - wfg(nds(limitset(pl, k)))

inclhv(p):
  return product {|p[j] - refPoint[j]| | j in {1 .. n}}

limitset(pl, k):
  for i = 1 to |pl| - k
    for j = 1 to n
      ql[i][j] = worse(pl[k][j], pl[k+i][j])
  return ql

nds(pl) returns the non-dominated subset of pl
```

Fig. 5. Pseudo-code for WFG. n is the number of objectives.

and the technique from Section III for calculating exclusive hypervolumes. The hypervolume of a set of points is calculated as a sum of exclusive hypervolumes, and each exclusive hypervolume is calculated by limiting the underlying set with the contributing point p, and subtracting the hypervolume of the modified set from the inclusive hypervolume of p. The result is a kind of inclusion-exclusion algorithm that uses domination to reduce the number and sizes of sets that have to be examined.

The base case for the basic algorithm is when it is applied to an empty set. Fig. 5 gives pseudo-code for WFG.

B. Optimizations

There are two obvious optimizations that can be applied to WFG.

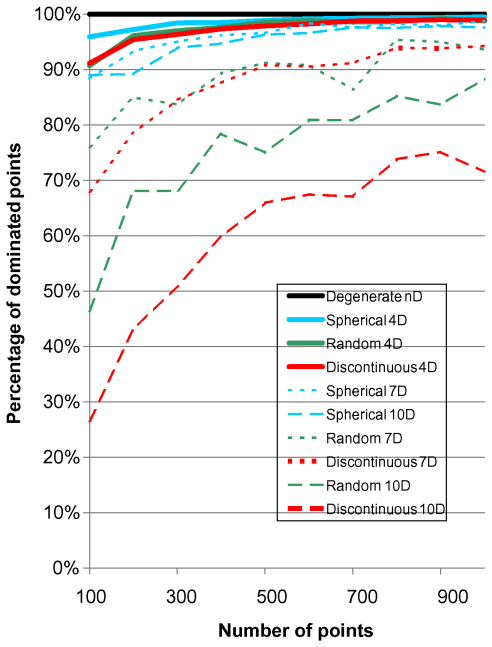


Fig. 6. Percentage of dominated points in a front after it has been limited by the worst point in the last objective. Each datum is the average of 20 distinct fronts. The degenerate line is 100% everywhere. The data is described in Section IV-C.

1) *Sorting the Points:* WFG calculates the exclusive hypervolume of every point in a set, but note that each point is processed relative to a different-sized underlying set. Specifically, the last point is processed relative to the empty set, the penultimate point relative to a set with one element, etc, up to the first point, which is processed relative to a set with $m - 1$ elements.

In addition, contributing points with worse objective values will limit the points in their underlying set more, so they will tend to generate a higher proportion of dominated points and consequently a smaller set in the recursive call.

Putting these two features together tells us that we should process contributing points with better objective values relative to smaller underlying sets. This means that the larger underlying sets will tend to have more dominated points that can be discarded, and the calculation will be faster overall.

The simplest way to implement this optimization is to sort the points so that they are improving monotonically in one of the objectives. Early experiments suggest that more-complicated sorting orders can generate even more dominated points, but sorting in a single objective is cheap, it is effective (see Fig. 6 and compare with Fig. 4), and it allows us to incorporate a further significant optimization.

2) *Slicing the Points:* Once the points are sorted in an objective, we can optimize WFG further by also slicing the hypervolume in that objective, as in HSO. This makes subsequent processing of the points cheaper (as they are smaller), and also it allows us to incorporate a new base case when the points are reduced to two objectives. This is a fast simple case with complexity $O(m)$ when the points are already sorted.

The alternative base case for a slicing algorithm is when the points are reduced to three objectives, when we can use the optimal 3-D algorithm of Beume *et al.* [12]. This algorithm has complexity $O(m \log m)$, but of course it saves one level of recursion in the main algorithm.

Other optimizations are clearly possible, the most obvious being a heuristic to select which objective should be used to sort and slice the points. We discuss some options briefly in Section V.

C. Experimental Comparison

We performed a series of experiments to investigate the performance of WFG and its optimizations, and to compare its performance with other recent hypervolume algorithms. We used three types of data.

1) Randomly-generated fronts, initialized by generating points with random values x, $0.1 \leq x \leq 10$, in all objectives. In order to guarantee mutual non-domination, we initialized $S = \phi$ and added each point $\overline{x}$ to S only if $\{\overline{x}\} \cup S$ would be mutually-non-dominating.
2) The discontinuous, spherical, and degenerate fronts from the DTLZ test suite [34]. For each front, we generated mathematically a representative set of 10 000 points from the known Pareto optimal set: then to form a front of a given size, we sampled this set randomly. We omit the linear front from DTLZ because it gives very similar performance to the spherical front.
3) The "worst-case data" for the bounding technique, from Fig. 3.

The data used in the experiments is available from wfg.csse.uwa.edu.au/hypervolume, and the code is available as follows.

1) HOY: ls11-www.cs.uni-dortmund.de/people/beume/publications/hoy.cpp.
2) FPL: iridia.ulb.ac.be/~manuel/hyper-volume(version 1.2).
3) IIHSO and WFG: wfg.csse.uwa.edu.au/hyper-volume.

All timings were performed on an Intel 2.4 GHz Core 2 Duo processor with 2 GB of RAM, running Ubuntu Linux 8.04.3. All algorithms were compiled with `gcc/g++ -O3 -funroll-loops -march=nocona` (version 4.2.4).

1) *Comparing the Optimizations:* Fig. 7 reports experiments on fronts of various sizes and types comparing the performance of WFG using the optimizations discussed in Section IV-B. The graphs show clearly that simply sorting the points so that they are improving in one objective provides a substantial speedup for WFG: slicing the points to either a 2-D or 3-D base case provides an additional speedup that is relatively small by comparison.

2) *Comparing with Other Algorithms:* Figs. 8–12 report experiments on fronts of various sizes and types comparing the performance of optimized WFG with FPL [24], HOY [11], and IIHSO [13]. FPL was optimized by using the MWW heuristic [23] (adapted for FPL's complexity model) to reorder the objectives, if there are more than four. HOY was run as is: no-one has yet published any useful optimizations for this algorithm.

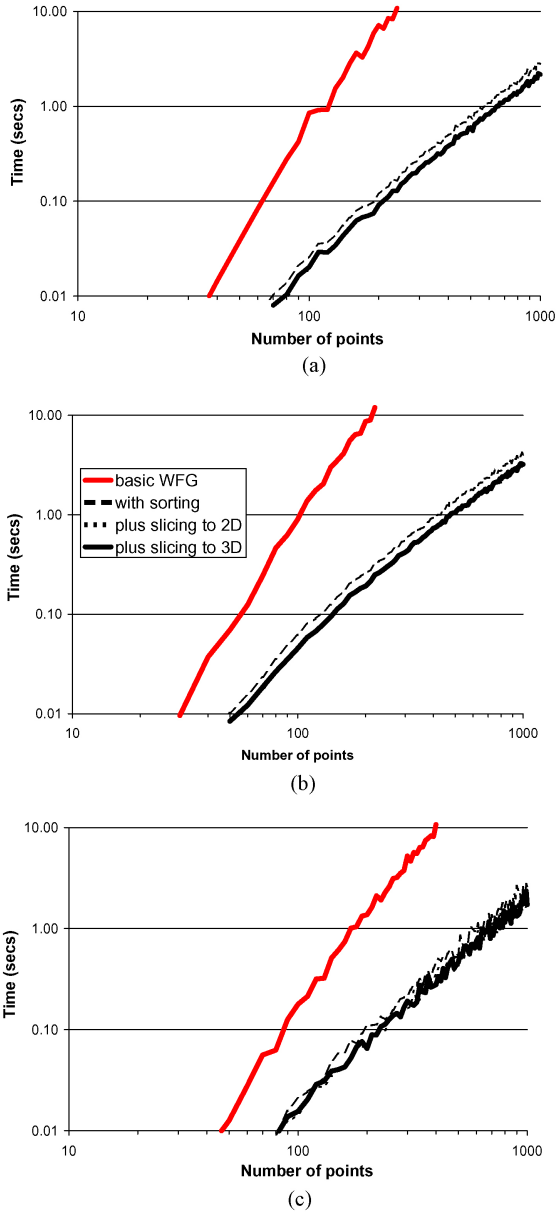


Fig. 7. Comparison of the performance of WFG using various optimizations. Each datum is the average processing time for 20 distinct fronts in 7 objectives. The legend on the middle graph applies for all three. (a) Random fronts in 7 objectives. (b) Discontinuous fronts in 7 objectives. (c) Spherical fronts in 7 objectives. In 7(a) and (b), the two "slicing" lines almost coincide.

The graphs show clearly that WFG outperforms by a significant margin all of the previous algorithms in almost all of the experiments, especially at higher numbers of objectives. The only exception is that IIHSO beats WFG for the 4-D data in Fig. 8.

3) *Performance on the Degenerate Front:* Fig. 13 reports experiments on the degenerate front-type.

Most modern hypervolume algorithms can process the degenerate front-type in polynomial time, and this is confirmed

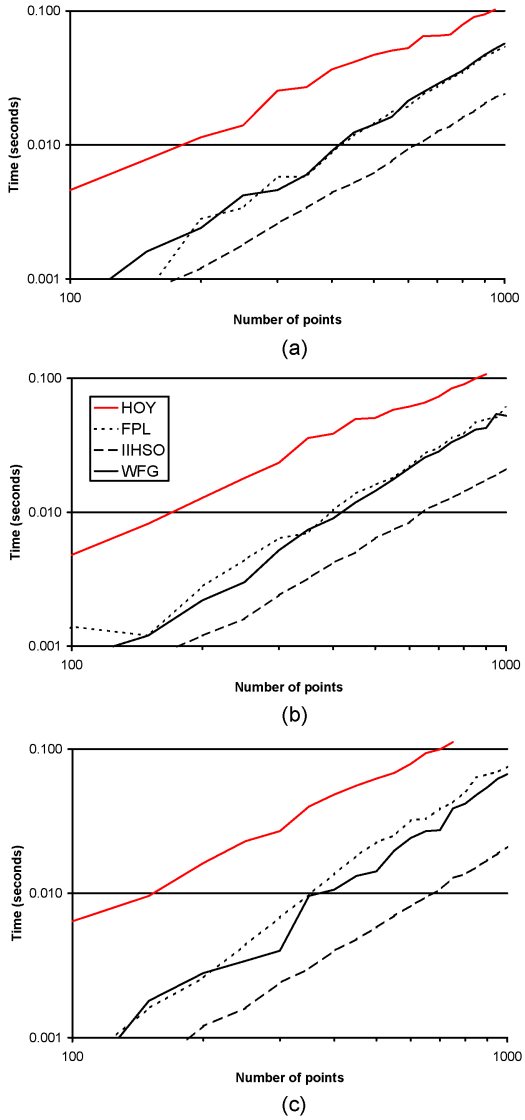


Fig. 8. Comparison of the performance of WFG with FPL, HOY, and IIHSO in four objectives. Each datum is the average processing time for 20 distinct fronts. The legend on the middle graph applies for all three. (a) Random fronts in four objectives. (b) Discontinuous fronts in four objectives. (c) Spherical fronts in four objectives.

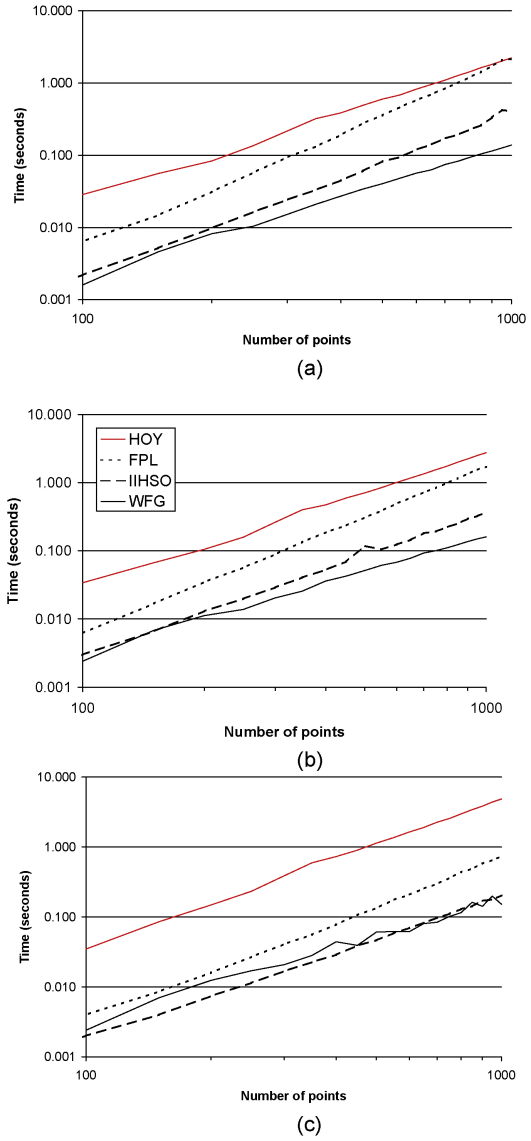


Fig. 9. Comparison of the performance of WFG with FPL, HOY, and IIHSO in five objectives. Each datum is the average processing time for 20 distinct fronts. The legend on the middle graph applies for all three. (a) Random fronts in five objectives. (b) Discontinuous fronts in five objectives. (c) Spherical fronts in five objectives.

by these results. However, degenerate appears to be the worst-case data for HOY: it can process only around 10–15 points in 13-D in a reasonable time frame.

4) *Performance on the "Worst-Case Data":* Fig. 14 reports experiments on the "worst-case data" for the bounding technique, from Fig. 3.

Even though the largest $n-2$ sets will have no dominated points for this front in each iteration, it is clear that the data poses no problem for the algorithm as a whole. We have yet to identify any data that troubles WFG.

5) *Overall Usability of WFG:* Table I shows the sizes of fronts that WFG can process in ten seconds, for each front-type. The table shows that WFG can process substantial fronts in all types up to 11 objectives, and probably a lot more objectives in spherical.

TABLE I
SIZES OF FRONTS THAT WFG CAN PROCESS IN TEN SECONDS

n	Random	Discontinuous	Spherical
3	>10 000	>10 000	>10 000
4	>10 000	>10 000	>10 000
5	>10 000	9600	7800
6	5800	5600	4400
7	2300	2000	2100
8	900	600	1300
9	450	300	1000
10	235	150	700
11	185	105	575
12	130	85	475
13	100	70	450

n is the number of objectives.

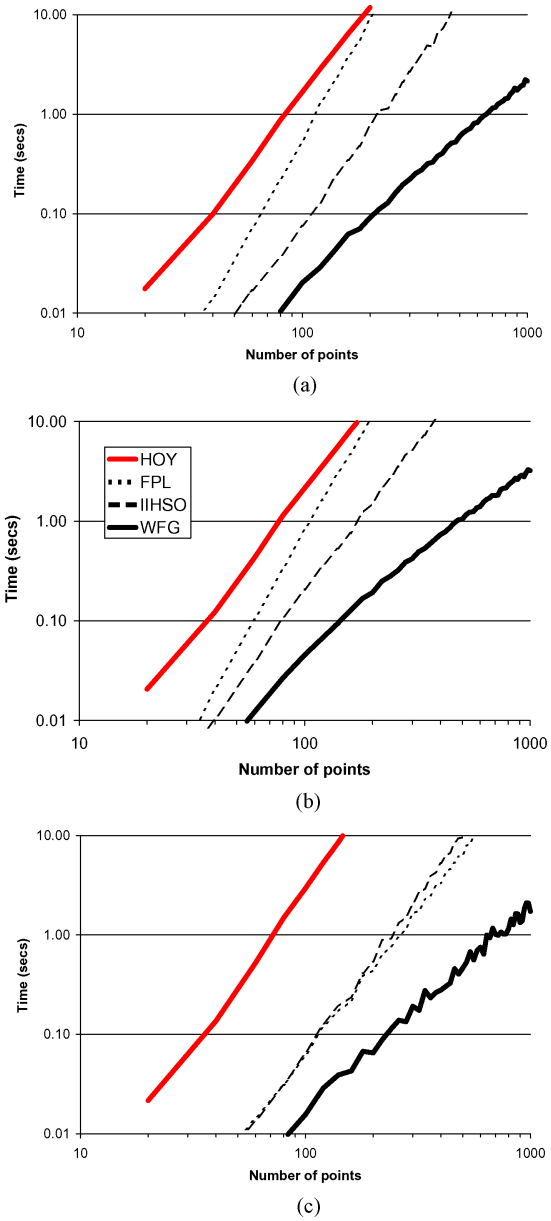


Fig. 10. Comparison of the performance of WFG with FPL, HOY, and IIHSO in seven objectives. Each datum is the average processing time for 20 distinct fronts. The legend on the middle graph applies for all three. (a) Random fronts in seven objectives. (b) Discontinuous fronts in seven objectives. (c) Spherical fronts in seven objectives.

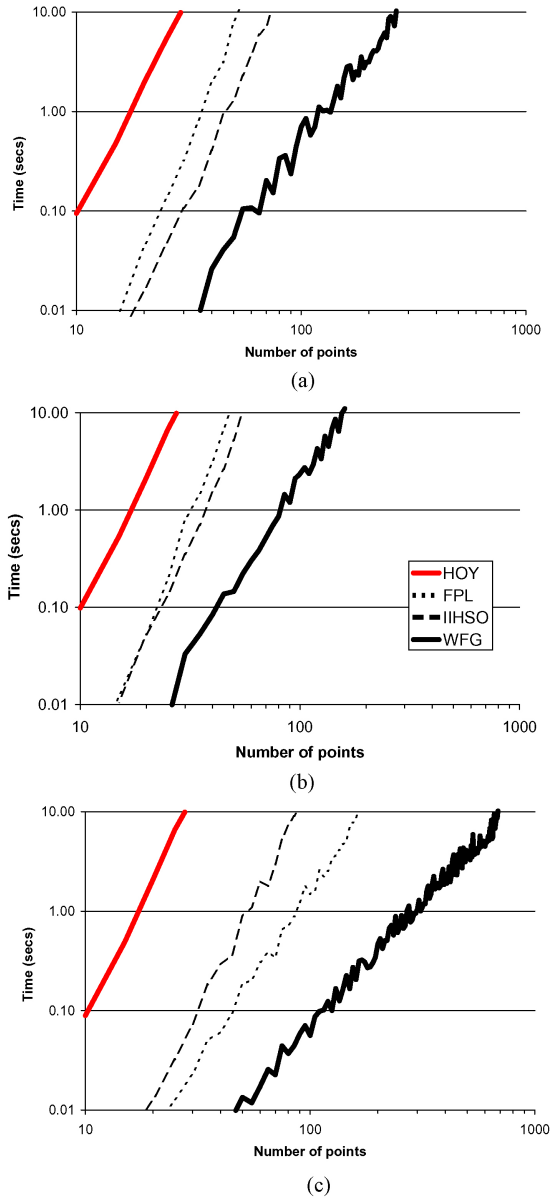


Fig. 11. Comparison of the performance of WFG with FPL, HOY, and IIHSO in ten objectives. Each datum is the average processing time for 20 distinct fronts. The legend on the middle graph applies for all three. (a) Random fronts in ten objectives. (b) Discontinuous fronts in ten objectives. (c) Spherical fronts in ten objectives.

D. Complexity

The complexity of basic WFG for m points in n objectives can be modeled by the following recurrence relations:

$$f(m,n) = \sum_{i=0}^{m-1} g(i,n) \tag{8}$$

$$g(m,n) = 1 + f(m',n). \tag{9}$$

Equation (8) returns the number of inclusive hypervolumes that are calculated in an application of WFG: it models the cost as a sum of exclusive hypervolumes over smaller sets. m' in (9) represents the size of the non-dominated set after it has been limited by one point.

Under the worst-case assumption that no points are ever dominated, i.e., $m' = m$, we can easily prove that

$$f(m,n) = 2^m - 1 \tag{10}$$

i.e., that WFG is exponential in the number of points in the worst case.

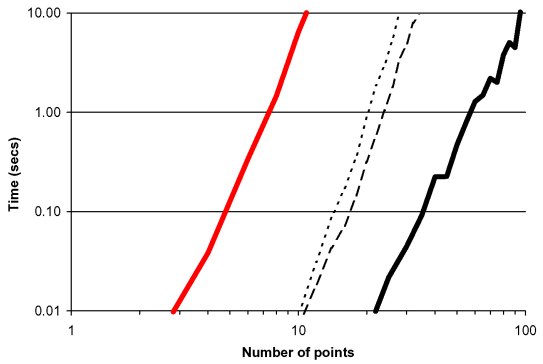
Proof: The proof is by induction on m.

Base case: $m = 0$

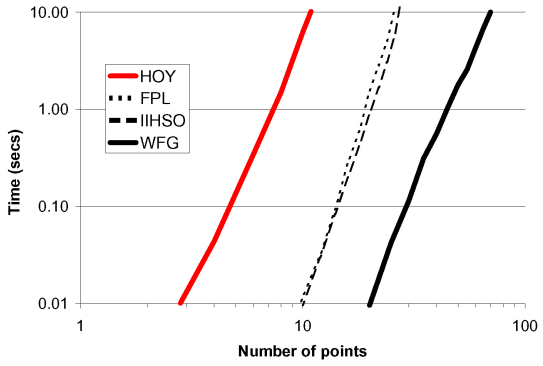
$$\sum_{i=0}^{0-1} g(i,n) = 0 = 2^0 - 1.$$

Inductive step: suppose (10) holds for $m = j$, that is

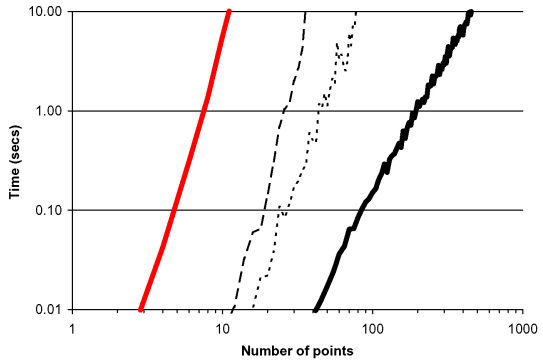
$$\sum_{i=0}^{j-1} g(i,n) = 2^j - 1. \tag{11}$$



(a)



(b)



(c)

Fig. 12. Comparison of the performance of WFG with FPL, HOY, and IIHSO in 13 objectives. Each datum is the average processing time for 20 distinct fronts. The legend on the middle graph applies for all three. (a) Random fronts in 13 objectives. (b) Discontinuous fronts in 13 objectives. (c) Spherical fronts in 13 objectives.

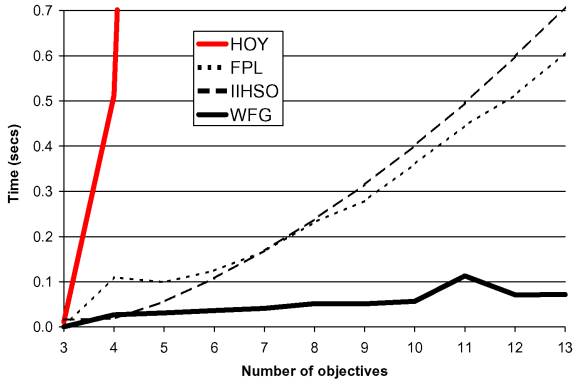


Fig. 13. Comparison of the performance of WFG with FPL and IIHSO on the degenerate front-type. Each datum is the average processing time for 20 distinct fronts, each with 1000 points.

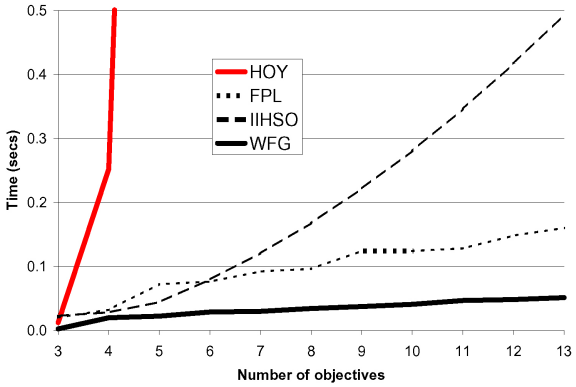


Fig. 14. Comparison of the performance of WFG with FPL, HOY, and IIHSO on the data from Fig. 3. Each front has 1000 points.

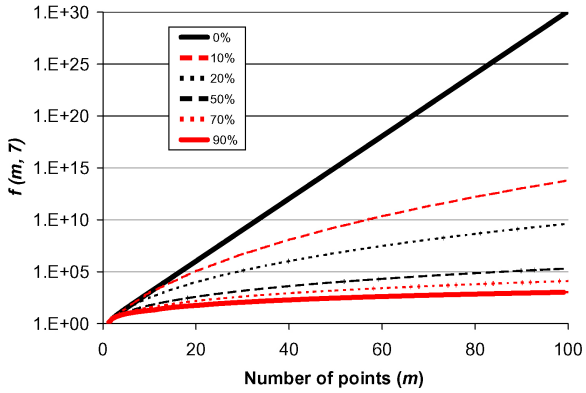


Fig. 15. Plot of the complexity of WFG in seven objectives, for various consistent percentages of dominated points. 0% corresponds to the theoretical worst case for the algorithm.

Then

$$\begin{aligned}
\sum_{i=0}^{j+1-1} g(i,n) &= \sum_{i=0}^{j} g(i,n) \\
&= g(j,n) + \sum_{i=0}^{j-1} g(i,n) \\
&= 1 + f(j,n) + \sum_{i=0}^{j-1} g(i,n) \text{ by (9)} \\
&= 1 + \sum_{i=0}^{j-1} g(i,n) + \sum_{i=0}^{j-1} g(i,n) \text{ by (8)} \\
&= 1 + 2\sum_{i=0}^{j-1} g(i,n) \\
&= 1 + 2(2^j - 1) \text{ by (11)} \\
&= 2^{j+1} - 1. \qquad \blacksquare
\end{aligned}$$

It is clear from the experimental results in Section IV-C that the real performance of WFG is totally unrelated to this worst-case complexity. It is instructive to plot (8) for various values of m'/m, to see the effect that the proportion of dominated points has on the complexity of WFG. Fig. 15 shows that even relatively small percentages of dominated points make a huge difference to the algorithm's performance. Fig. 6 shows what percentages might be realized for typical real-world data.

We do not include here complexity coefficients for WFG, because of the difficulty in applying the worst-case model

when it is clearly not appropriate. However the performance data in Section IV-C demonstrate strongly that the algorithm is a significant development.

V. Conclusion

We have described a new algorithm WFG that can calculate exact hypervolumes in five or more objectives far faster than any previously-described algorithm. WFG is based on the recently-described observation that the exclusive hypervolume of a point p relative to a set S is equal to the difference between the inclusive hypervolume of p and the hypervolume of S with each point limited by the objective values in p. Limiting the points in S in this way leads to many points being dominated, which lead to very fast calculations. WFG applied this technique iteratively over a set to calculate its hypervolume. We also described some minor but effective optimizations for WFG.

Future work on WFG will include investigating further optimization by:

1) using heuristics to select a good objective for sorting and slicing;
2) using different sorting rules to generate smaller underlying sets;
3) re-organizing the domination calculations in WFG.

Another obvious idea is to use the bounding technique directly to calculate exclusive hypervolumes. However, the most common use of exclusive hypervolume is to identify the point from a set that contributes the least to the hypervolume of the set. We do not need to know the exclusive hypervolume of each point—once we know that point p is bigger than point q, we ignore p—so a crucial part of this process is that we calculate as little as possible about each point. This is easy using HSO, which calculates the exclusive hypervolume of a point as a series of additions. However, WFG uses both additions and subtractions, so breaking up the calculation is more complicated, and we have not yet found a decomposition that gives good results. This also is future work.

Acknowledgment

The authors thank S. Huband for generating the raw DTLZ data.

References

[1] R. Purshouse, "On the evolutionary optimization of many objectives," Ph.D. dissertation, Dept. Automatic Control Syst. Eng., Univ. Sheffield, Sheffield, U.K., 2003.

[2] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications," Ph.D. dissertation, Comput. Eng. Netw. Lab., Swiss Federal Instit. Technol. (ETH), Zurich, Switzerland, 1999.

[3] M. Laumanns, E. Zitzler, and L. Thiele, "A unified model for multi-objective evolutionary algorithms with elitism," in *Proc. Congr. Evol. Comput.*, 2000, pp. 46–53.

[4] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.

[5] M. Fleischer, "The measure of Pareto optima: Applications to multiobjective metaheuristics," Instit. Syst. Res., Univ. Maryland, College Park, Tech. Rep. ISR TR 2002-32, 2002.

[6] S. Huband, P. Hingston, L. While, and L. Barone, "An evolution strategy with probabilistic mutation for multiobjective optimization," in *Proc. Congr. Evol. Comput.*, 2003, pp. 2284–2291.

[7] J. Knowles and D. Corne, "Properties of an adaptive archiving algorithm for storing nondominated vectors," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 100–116, Apr. 2003.

[8] J. Knowles, D. Corne, and M. Fleischer, "Bounded archiving using the Lebesgue measure," in *Proc. Congr. Evol. Comput.*, 2003, pp. 2490–2497.

[9] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Proc. 8th Parallel Problem Solving Nature*, LNCS 3242. 2004, pp. 832–842.

[10] M. Emmerich, N. Beume, and B. Naujoks, "An EMO algorithm using the hypervolume measure as selection criterion," in *Proc. Evol. Multi-objective Optimization*, LNCS 3410. 2005, pp. 62–76.

[11] N. Beume, "S-metric calculation by considering dominated hypervolume as Klee's measure problem," *Evol. Comput.*, vol. 17, no. 4, pp. 477–492, 2009.

[12] N. Beume, C. M. Fonseca, M. López-Ibáñez, L. Paquete, and J. Vahrenhold, "On the complexity of computing the hypervolume indicator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1075–1082, Oct. 2009.

[13] L. Bradstreet, L. While, and L. Barone, "A faster many-objective hypervolume algorithm using iterated incremental calculations," in *Proc. Congr. Evol. Comput.*, 2010, pp. 179–186.

[14] K. Bringmann and T. Friedrich, "Approximating the least hypervolume contributor: NP-hard in general, but fast in practice," in *Proc. 5th Int. Conf. Evol. Multi-Criterion Optimization*, 2009, pp. 6–20.

[15] L. Bradstreet, L. While, and L. Barone, "A new way of calculating exact exclusive hypervolumes," School Comput. Sci. Softw. Eng., Univ. Western Australia, Perth, Australia, Tech. Rep. UWA-CSSE-09-002, 2009.

[16] T. Bäck, D. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. Bristol, U.K.: Institute of Physics Publishing and Oxford University Press, 1997.

[17] J. Wu and S. Azarm, "Metrics for quality assessment of a multiobjective design optimization solution set," *J. Mech. Des.*, vol. 123, no. 1, pp. 18–25, 2001.

[18] M. Fleischer, "The measure of Pareto optima: Applications to multi-objective metaheuristics," in *Proc. Evol. Multiobjective Optimization*, LNCS 2632. 2003, pp. 519–533.

[19] L. While, "A new analysis of the LEB measure algorithm for calculating hypervolume," in *Proc. Evol. Multiobjective Optimization*, LNCS 3410. 2005, pp. 326–340.

[20] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 29–38, Feb. 2006.

[21] E. Zitzler. (2001). *Hypervolume Metric Calculation* [Online]. Available: ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/hypervol.c

[22] J. Knowles, "Local-search and hybrid evolutionary algorithms for Pareto optimization," Ph.D. dissertation, Dept. Comput. Sci., Univ. Reading, Reading, U.K., 2002.

[23] L. While, L. Bradstreet, L. Barone, and P. Hingston, "Heuristics for optimizing the calculation of hypervolume for multiobjective optimization problems," in *Proc. Congr. Evol. Comput.*, 2005, pp. 2225–2232.

[24] C. M. Fonseca, L. Paquete, and M. López-Ibáñez, "An improved dimension-sweep algorithm for the hypervolume indicator," in *Proc. Congr. Evol. Comput.*, 2006, pp. 3973–3979.

[25] L. Bradstreet, L. While, and L. Barone, "A fast incremental hypervolume algorithm," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 714–723, Dec. 2008.

[26] M. H. Overmars and C.-K. Yap, "New upper bounds in Klee's measure problem," *SIAM J. Comput.*, vol. 20, no. 6, pp. 1034–1045, Dec. 1991.

[27] K. Bringmann and T. Friedrich, "Approximating the least hypervolume contributor: NP-hard in general, but fast in practice," in *Proc. Evol. Multiobjective Optimization*, LNCS 5467. 2009, pp. 6–20.

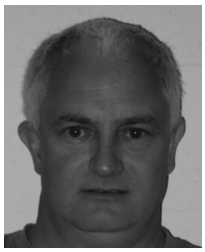
[28] R. Everson, J. Fieldsend, and S. Singh, "Full elite sets for multiobjective optimization," in *Proc. 5th Int. Conf. Adaptive Comput. Design Manuf.*, 2002, pp. 87–100.

[29] J. Bader, K. Deb, and E. Zitzler, "Faster hypervolume-based search using Monte Carlo sampling," in *Proc. Multiple Criteria Decision Making Sustainable Energy Transp. Syst.*, LNCS 634. 2010, pp. 313–326.

[30] H. Ishibuchi, N. Tsukamoto, Y. Sakane, and Y. Nojima, "Hypervolume approximation using achievement scalarizing functions for evolutionary many-objective optimization," in *Proc. Congr. Evol. Comput.*, 2009, pp. 530–537.

[31] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Theory of the hypervolume indicator: Optimal μ-distributions and the choice of the

reference point," in *Proc. Workshop Found. Genet. Algorithms*, 2009, pp. 87–102.
[32] K. Bringmann and T. Friedrich, "Don't be greedy when calculating hypervolume contributions," in *Proc. Workshop Found. Genet. Algorithms*, 2009, pp. 103–112.
[33] T. Friedrich, C. Horoba, and F. Neumann, "Multiplicative approximations and the hypervolume indicator," in *Proc. Genet. Evol. Comput. Conf.*, 2009, pp. 571–578.
[34] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multiobjective optimization test problems," in *Proc. Congr. Evol. Comput.*, 2002, pp. 825–830.



Lyndon While (SM'04) received the B.S. (Eng.) and Ph.D. degrees from the Imperial College of Science and Technology, London, U.K., in 1985 and 1988, respectively.

He is currently an Associate Professor with the School of Computer Science and Software Engineering, University of Western Australia, Perth, Australia. His current research interests include evolutionary algorithms, multiobjective optimization, and the semantics and implementation of functional programming languages.

Lucas Bradstreet (M'10) received the Bachelors degree in computer and mathematical sciences from the University of Western Australia, Perth, Australia, in 2004. He recently received the Ph.D. at the School of Computer Science and Software Engineering, University of Western Australia.

His current research interests include multiobjective evolutionary algorithms, and metrics and benchmarks for assessing their performance.

Luigi Barone received the B.S. and Ph.D. degrees from the University of Western Australia, Perth, Australia, in 1994 and 2004, respectively.

He is currently a Lecturer with the School of Computer Science and Software Engineering, University of Western Australia. His current research interests include evolutionary algorithms and their use for optimization and opponent modeling, and the modeling of biological systems.