Applying the WFG Algorithm to Calculate Incremental Hypervolumes

Lyndon While
School of Computer Science & Software Engineering
The University of Western Australia
Perth, Australia 6009
Email: lyndon@csse.uwa.edu.au

Lucas Bradstreet
School of Computer Science & Software Engineering
The University of Western Australia
Perth, Australia 6009
Email: lucas@csse.uwa.edu.au

***Abstract*—Hypervolume is increasingly being used in-line in multi-objective evolutionary algorithms, either to promote diversity, or in an archiving mechanism, or in the selection process. The usual requirement is to determine which point in a set contributes least to the hypervolume of the set, so that that point can be discarded. We describe a new exact algorithm IWFG for performing this calculation that combines two important features from other recent algorithms: the bounding trick from WFG, that accelerates calculations by generating lots of dominated points; and the best-first queuing mechanism from IHSO, that eliminates much of the calculation for most of the points in a set. Empirical results show that IWFG is significantly faster than IHSO on much experimental data in five or more objectives.**

I. Introduction

Hypervolume[1] (the S-metric[2] or the Lebesgue measure[3]) is a popular metric for comparing the performance of multi-objective optimisers. The hypervolume of a set of solutions is 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's main drawback is that it is expensive to calculate in larger numbers of dimensions.

Hypervolume is also increasingly used in-line in multi-objective 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, so that that point can be discarded. Calculating hypervolume in-line obviously puts much more stress on the performance issue.

The fastest algorithm yet known for calculating hypervolume as a metric is the recently-described WFG algorithm[11]. WFG splits the volume to be calculated into a series of point-wise exclusive hypervolumes, each calculated using a technique described in [12], [13]. 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, which makes WFG much faster (in five or more objectives) than any other published algorithm for calculating hypervolume exactly.

Given that WFG is based on a fast calculation for exclusive hypervolume, it would seem that applying it to in-line incremental calculations should be easy. However when determining the least-contributing point in a set, one crucial trick is to process as little as possible of each point: ideally, if the smallest point has volume v, for each other point one would calculate only enough volume to prove that it is not the smallest, i.e. $v + \epsilon$. All other calculations can be eliminated. This principle is exemplified in the best-first priority queuing scheme used in IHSO[14], the fastest algorithm yet known for determining least-contributing points. In most algorithms deferring these calculations is easy, because the volume associated with each point increases monotonically during the run, so we can just focus on processing the currently-smallest points(s) until one of them is fully evaluated. However because WFG calculates hypervolumes as *differences* between volumes, deferring partial calculations is much more complicated, and this principle breaks down.

The principal contribution of this paper is a new algorithm IWFG for determining the least-contributing point in a set. IWFG modifies WFG by applying one or more IHSO-style slicing operations at the highest levels of the calculation, thus dividing the volume of each point into a sum of smaller exclusive hypervolumes that can each be calculated very quickly using the bounding trick from WFG, and whose calculation can be deferred as appropriate. Experimental results show that IWFG is significantly faster than IHSO in much experimental data in five or more objectives, although our current implementation has some granularity issues that affect its performance on some types of data.

The rest of the paper is structured as follows. Section II describes the necessary background material in multi-objective optimisation and hypervolume, and Section III describes the relevant details of WFG and IHSO. Section IV describes the new incremental algorithm IWFG, and Section V describes an experimental comparison of IWFG and IHSO. Section VI concludes the paper and suggests some future work.

II. Background Material

A. Multi-objective Optimisation

In a multi-objective optimisation problem, we aim to find the set of optimal trade-off 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 characterised 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 multi-objective 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 [15].

B. Hypervolume

Given a set of solutions returned by a multi-objective optimiser, the question arises how well it approximates the Pareto optimal set. One metric used widely for comparing sets of solutions is their *hypervolume*[1], [2], [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 multi-objective optimisers, 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, i.e.

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

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, i.e.

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

All of these concepts are illustrated in Fig. 1.

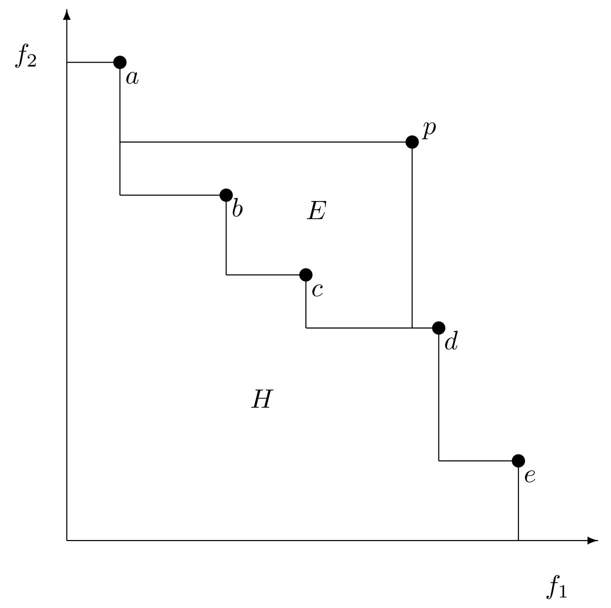


Fig. 1: Maximising in both objectives relative to the origin, the hypervolume of $\{a, b, c, d, e\}$ is the solid-bordered shape labeled H, the exclusive hypervolume of p relative to $\{a, b, c, d, e\}$ is the shape labeled E, and the inclusive hypervolume of any point is the rectangle bounded by that point and the origin. (Adapted from [11].)

C. Previous Algorithms for Calculating Exact Hypervolumes

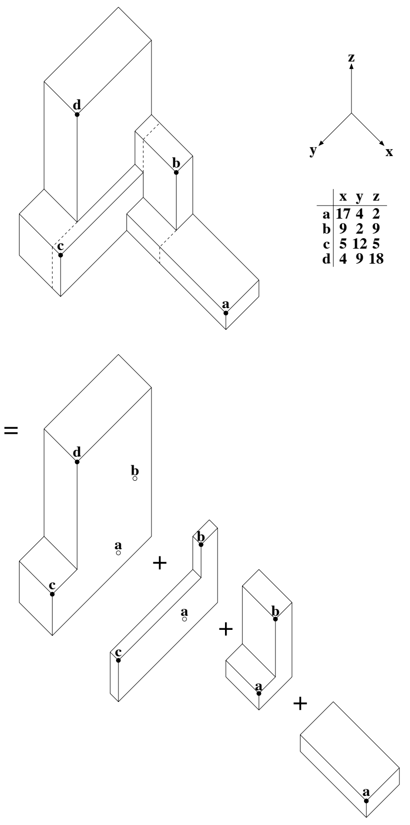
Several algorithms have been proposed for calculating hypervolumes and exclusive hypervolumes exactly.

The inclusion-exclusion algorithm for calculating the size of a set union has been adapted for hypervolume calculation[16], but its complexity is $O(n2^m)$ for m points in n objectives, so it is unusable in practice.

LebMeasure[17] 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. LebMeasure also has been shown to be very slow[18].

HSO (Hypervolume by Slicing Objectives)[19], [20], [18] 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. Fig. 2 illustrates the principle. HSO's worst-case complexity is $O(m^{n-1})$[18], but good heuristics have been described for re-ordering objectives[22] that deliver much better performance for typical data. FPL (Fonseca, Paquete, López-Ibáñez)[23] is a highly-optimised version of HSO: its principal interesting feature is a recent optimal algorithm for the 3D case[24].

IHSO (Incremental HSO)[14] is a version of HSO customised for incremental calculations. It uses various ideas to reduce the cost of calculating exclusive hypervolumes, and it uses a "best-first" queuing mechanism to determine very efficiently the least-contributing point in a set. IHSO is discussed in more detail in Section III. IIHSO (Iterated



	x	y	z
a	17	4	2
b	9	2	9
c	5	12	5
d	4	9	18

Fig. 2: One step in HSO for calculating $Hyp(\{a, b, c, d\})$. Objective x is processed, leaving four slices in y and z, each with a constant thickness in x. Marks indicate points, and unfilled marks indicate points which are dominated in y and z. (Reproduced from [21].)

IHSO)[21] combines IHSO with the point-wise processing of LebMeasure, again with heuristics to optimise the performance for a given data set.

HOY (Hypervolume by Overmars and Yap)[25] 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. HOY's worst-case complexity ($O(m \log m + m^{n/2})$) is by far the best of any algorithm, but its performance on realistically-sized fronts is poor[27], [21], [11].

The fastest algorithm yet known for calculating exact hypervolumes is WFG (Walking Fish Group)[11], which we discuss in Section III.

An obvious alternative to calculating hypervolume exactly is to use an approximation algorithm (for example [28], [29], [27], [30]). Using such algorithms introduces a trade-off between precision and performance, and much improvement has been reported recently on understanding this trade-off[31], [32], [33]. However, we do not compare with approximation algorithms here.

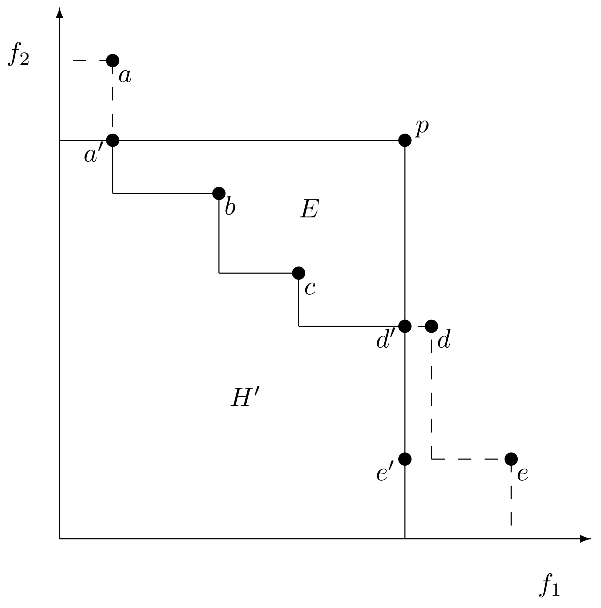


Fig. 3: Maximising in both objectives, the exclusive hypervolume of p relative to $\{a, b, c, d, e\}$ (i.e. E) = the inclusive hypervolume of p (i.e. the rectangle with p at the top-right corner) minus the hypervolume of $\{a', b, c, d', e'\}$ (i.e. H'). Clearly e' is dominated by d' and can be discarded. (Reproduced from [11].)

III. WFG and IHSO

WFG and IHSO form the basis of our new algorithm IWFG, so here we discuss their salient features in more detail.

A. WFG

WFG[11] is the fastest known algorithm for calculating exactly the hypervolume of a set of points. It combines the point-wise processing used in LebMeasure[17] with a new technique for calculating the exclusive hypervolume of a point[12], [13]. The exclusive hypervolume of a point p relative to a set S is the difference between the inclusive hypervolume of p and the hypervolume of S after each point has been limited to be no better than p in any objective. Fig. 3 illustrates the principle.

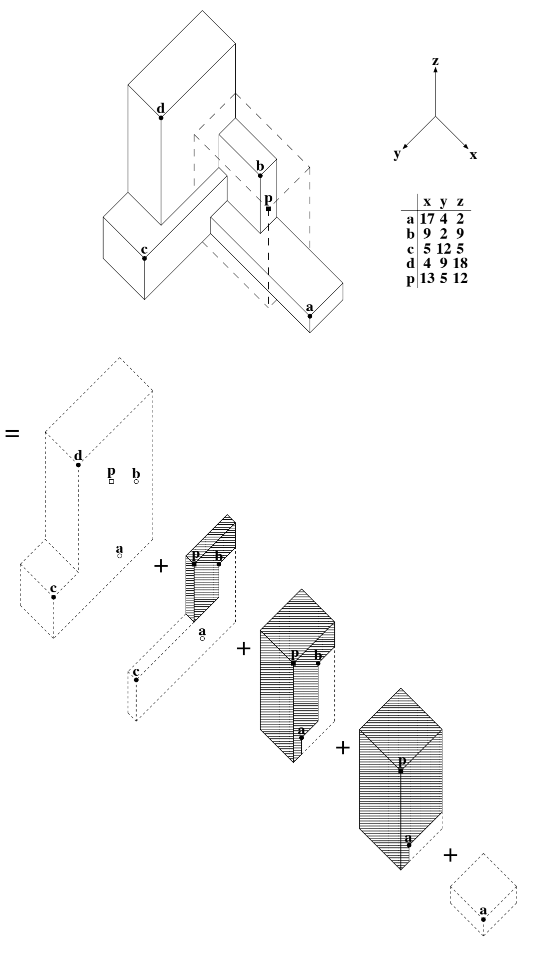
Experiments show that this limiting operation tends to produce a lot of points that are dominated within the new set, so subsequent calculations are very fast, and WFG easily outperforms all other exact hypervolume algorithms[11]. Fig. 4 gives pseudo-code for WFG.

B. IHSO

IHSO[14] is the fastest known algorithm for identifying the least-contributing point in a set. IHSO uses the slice-by-slice technique used in HSO, combined with heuristics to efficiently order the objectives and various techniques to eliminate slices where possible. In particular, slices "above" a point p and

```
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. 4: Pseudo-code for WFG. n is the number of objectives. (Reproduced from [11].)



	x	y	z
a	17	4	2
b	9	2	9
c	5	12	5
d	4	9	18
p	13	5	12

Fig. 5: One step in IHSO for calculating $ExcHyp(p, \{a, b, c, d\})$. Objective x is processed, leaving three non-empty slices (and two empty slices) in y and z, each with a constant thickness in x. The volume to be calculated is indicated by the dashed box on the main picture and by the shading on the slices. Marks indicate points, and unfilled marks indicate points which are dominated in y and z by one of $\{a, b, c, d\}$. (Reproduced from [21].)

slices where p is dominated in the remaining objectives both contribute nothing to p's exclusive hypervolume. Fig. 5 illustrates the principle.

But central to the performance of IHSO is a priority queuing scheme that tries to minimise the amount of work that must be done on each point. The volume dominated by each point is divided into small chunks, and only the currently-smallest point is processed at any given time. IHSO terminates when the currently-smallest point has been completely processed. Fig. 6 outlines the scheme. The granularity of the evaluation

```
Evaluate each point a bit
Identify the smallest point
while the smallest point is not completed
  Evaluate the smallest point a bit more
  Identify the new smallest point
return the smallest point
```

Fig. 6: Outline of the best-first queuing scheme in IHSO. (Reproduced from [14].)

used in IHSO has been set empirically so that the minimum amount of work is required to identify the least contributor.

IV. IWFG: A Fast Algorithm for Calculating Incremental Hypervolumes

Given how fast WFG is, we would like to use the bounding trick illustrated in Fig. 3 in a scheme to identify the least-contributing point in a set. However, there is a problem with combining the bounding trick with the priority queuing scheme used in IHSO: because the trick involves subtracting one volume from another, we cannot simply defer either calculation. The queuing scheme is based on the assumption that the volume associated with each point increases monotonically as it is processed, otherwise the designation of a "currently-smallest point" has no reliable meaning.

In order to defer calculations, we need an algorithm that does one or more initial steps that divide up the space into separate volumes to be summed, then we can defer the calculation of some of those volumes in the style of IHSO. There are several possible ways in which we could divide up the space, but given the success of IHSO, one obvious suggestion is an algorithm that

- does a single IHSO-style slicing step at the top-level,
- calculates the highest relevant slice for each point immediately, to initialise the queue,
- defers all other slices to minimise the amount of work done for each point,
- and processes all slices needed using WFG, to maximise performance.

Fig. 7 gives pseudo-code for this algorithm, which we shall call IWFG. Note in particular that the identity of the currently-smallest point `sm` will likely change between iterations of the `while` loop in Fig. 7. Doing only one slicing step causes granularity problems for this simple version of IWFG: however conceptually it is easy to solve these problems simply by adding more slicing steps.

```
iwfg(pl):
  sort the points worsening in any objective
  for each point p in pl:
    s = the highest slice relevant to p
    partial[p] = exclhv(p, s)
  sm = the point with the smallest partial
  while sm has not been completely processed
    s = the next slice for sm
    partial [sm] += exclhv(sm, s)
    sm = the point with the smallest partial
  return sm
```

Fig. 7: Pseudo-code for IWFG. Other functions are defined in Fig. 4.

V. Experimental Performance

We performed a series of experiments to investigate the performance of IWFG, and to compare its performance with IHSO. We used two types of data.

- Randomly-generated fronts, initialised by generating points with random values x, $0.1 \leq x \leq 10$, in all objectives. In order to guarantee mutual non-domination, we initialised $S = \phi$ and added each point $\overline{x}$ to S only if $\{\overline{x}\} \cup S$ would be mutually-non-dominating. We ran these fronts as maximisation problems with a reference point of 0^*.
- The discontinuous, spherical, and linear fronts from the DTLZ test suite[34]. (We omit the degenerate front because it runs extremely quickly with all reasonable algorithms.) 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 ran these fronts as minimisation problems with a reference point of 20^*.

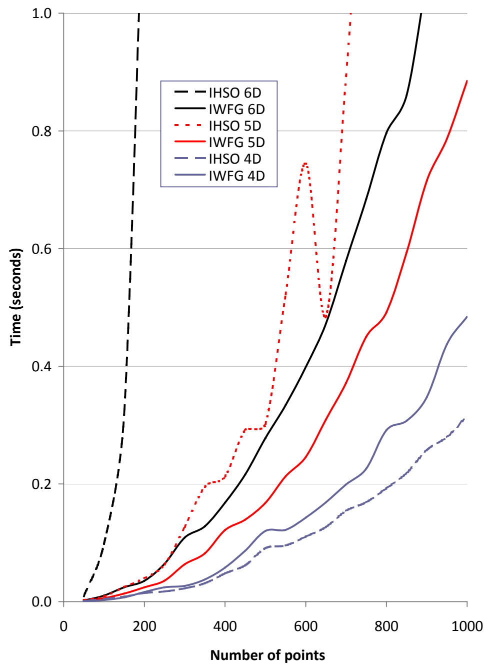
The data used in the experiments is available on request.

In each run, each algorithm takes a set of points and determines which point has the smallest exclusive hypervolume relative to the rest of the set.

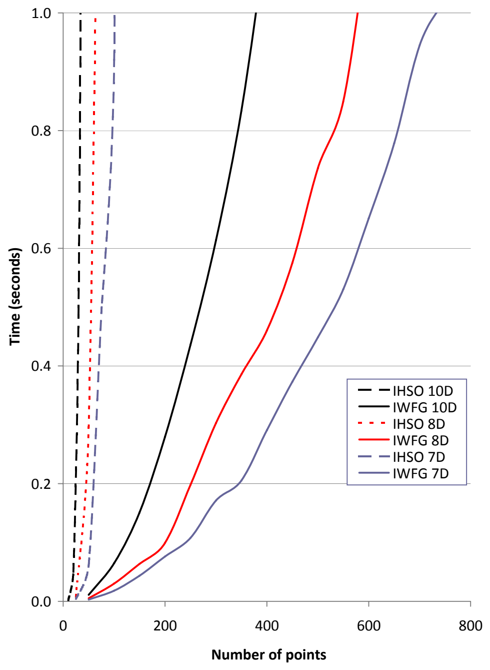
All timings were performed on a MacBook Pro 2010 with a dual core 2.3GHz i7 processor and 8GB of RAM, running Mac OSX 10.7.2. All programs were written in C and compiled with gcc v4.2.1 (LLVM) –O3 –arch x86_64.

Figs. 8–11 plot the performance of IWFG and IHSO on each of the front-types. We observe the following.

- For the linear, spherical, and discontinuous front-types, IHSO and IWFG perform similarly in 4–5D, but IWFG beats IHSO by a large margin in higher dimensions. Note that the linear front is really a special case of the spherical front, so it is not surprising that they behave similarly. Also note that we do not expect IWFG to outperform IHSO at 4D, because IIHSO outperforms WFG at 4D[11].
- For the randomly-generated front-type, the performance is a complete contrast: IWFG outperforms IHSO somewhat in 4–5D, but is well-beaten in higher dimensions. We believe this is due to the exceptional performance of IHSO on this data, and to the granularity issues with our current implementation of IWFG: IHSO can eliminate

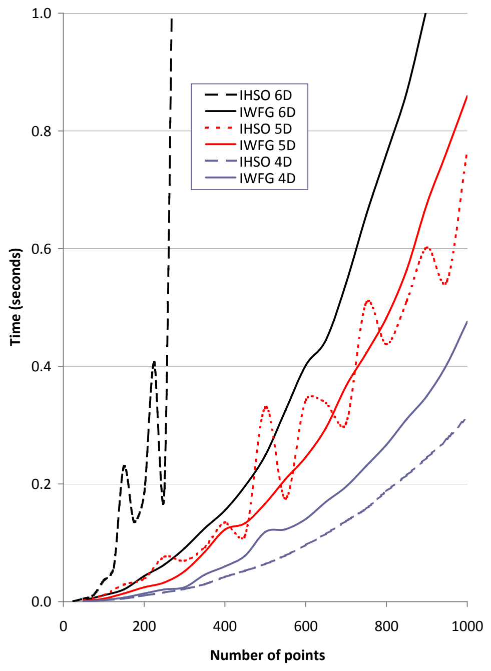


(a) 4–6 objectives.

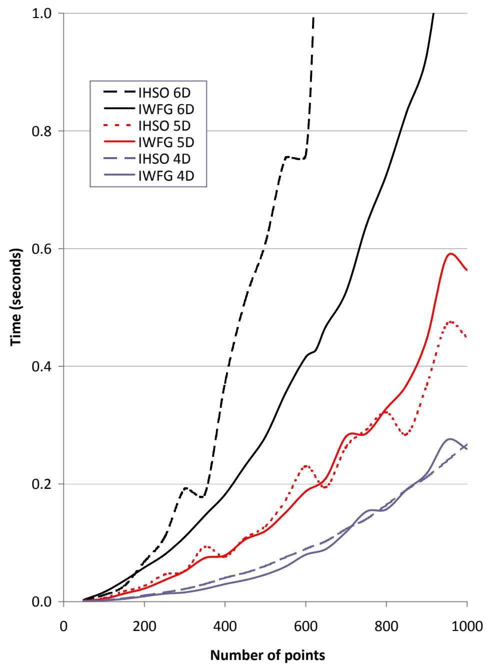


(b) 7–10 objectives.

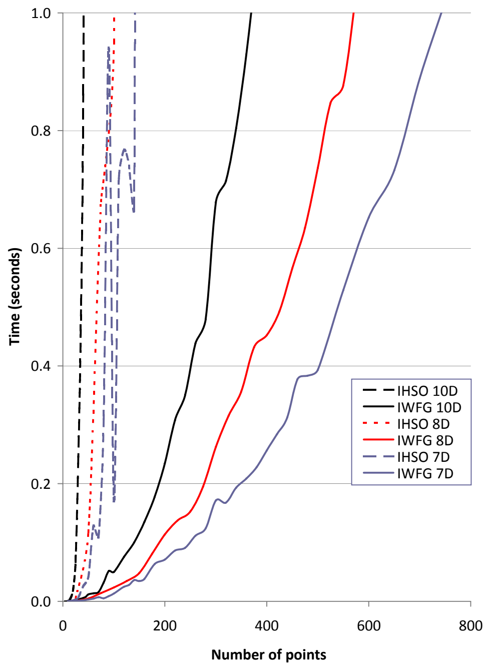
Fig. 8: Comparison of the performance of IWFG and IHSO on the DTLZ linear front. Each line plots the average processing time for twenty distinct fronts.



(a) 4–6 objectives.

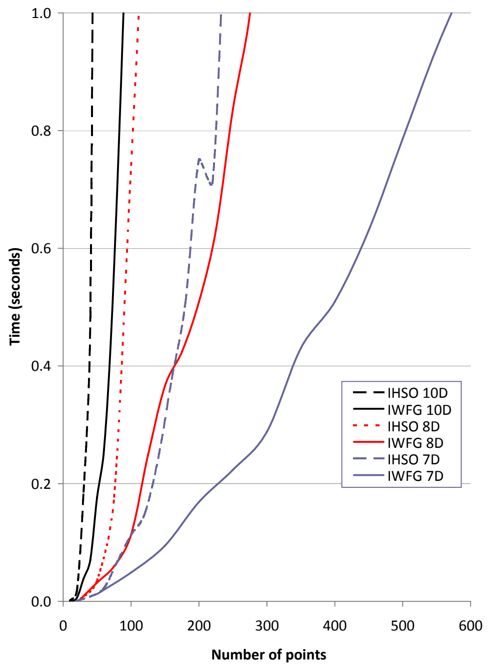


(a) 4–6 objectives.



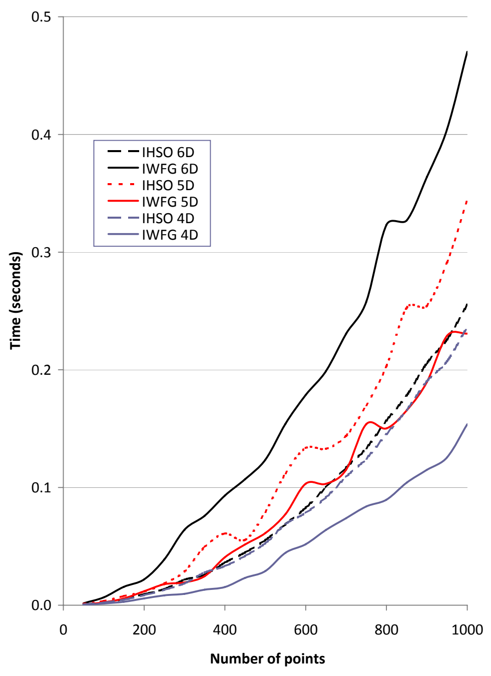
(b) 7–10 objectives.

Fig. 9: Comparison of the performance of IWFG and IHSO on the DTLZ spherical front. Each line plots the average processing time for twenty distinct fronts.

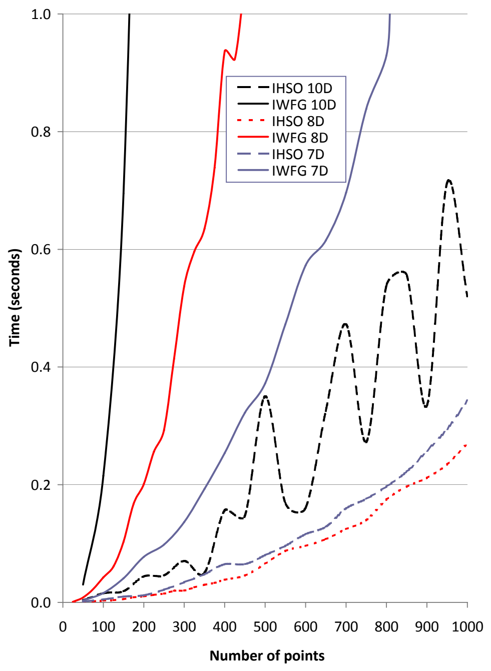


(b) 7–10 objectives.

Fig. 10: Comparison of the performance of IWFG and IHSO on the DTLZ discontinuous front. Each line plots the average processing time for twenty distinct fronts.



(a) 4–6 objectives.



(b) 7–10 objectives.

Fig. 11: Comparison of the performance of IWFG and IHSO on the randomly-generated fronts. Each line plots the average processing time for twenty distinct fronts. Note the different time scale on the first plot.

much more work because it is able to decompose each point's volume into more, smaller chunks. The granularity issues with IWFG are most easily illustrated by observing that the point which is worst in the slicing objective is always fully evaluated.

- The performance of IWFG is generally more predictable than that of IHSO, which may have value in some contexts.
- We as yet have no heuristics to optimise the performance of IWFG, whereas IHSO has been highly-optimised[14]. For example, initial experiments in choosing the slicing objective for IWFG indicate that this might deliver a performance improvement of 5–30%, depending on the front-type.

Table I shows the approximate sizes of fronts that IWFG can process in one second, for each front-type. The table shows that IWFG can process substantial fronts up to 8-10D in all of the types of data tested here.

TABLE I: Sizes of fronts that IWFG can process in one second. n is the number of objectives.

n	Linear	Spherical	Discontinuous	Random
4	$>$ 1,000	$>$ 1,000	$>$ 1,000	$>$ 1,000
5	$>$ 1,000	$>$ 1,000	$>$ 1,000	$>$ 1,000
6	880	900	910	$>$ 1,000
7	730	740	570	810
8	580	570	280	440
10	380	370	90	160

VI. Conclusion

We have described a new algorithm IWFG for determining which point in a set contributes least to the hypervolume of the set. IWFG modifies the WFG algorithm by applying one or more IHSO-style slicing operations at the highest levels of the calculation, thus dividing the volume of each point into a sum of smaller exclusive hypervolumes that can each be calculated very quickly using the bounding trick from WFG. Decomposing the volume of a point in this way allows us defer most of the processing of points which are unlikely to be the smallest.

We have presented empirical results which show that IWFG is significantly faster than IHSO in much experimental data in five or more objectives, although the current implementation has some granularity issues that mean it is slower on randomly-generated fronts.

We plan to continue the development of IWFG by incorporating multiple nested IHSO-style steps where appropriate. This should improve the granularity of the point evaluations wrt the best-first queue, and reduce the amount of work that the algorithm has to do overall. We also plan to investigate the design of heuristics to improve the performance of both IWFG and WFG.

Acknowledgment

We thank Simon Huband for generating the raw DTLZ data.

References

[1] R. Purshouse, "On the evolutionary optimisation of many objectives," Ph.D. dissertation, The University of Sheffield, United Kingdom, 2003.

[2] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications," Ph.D. dissertation, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland, 1999.

[3] M. Laumanns, E. Zitzler, and L. Thiele, "A unified model for multi-objective evolutionary algorithms with elitism," in *Congress on Evolutionary Computation*, R. Eberhart, Ed. IEEE, 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 Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, April 2003.

[5] M. Fleischer, "The measure of Pareto optima: Applications to multi-objective metaheuristics," Institute for Systems Research, University of Maryland, Technical Report ISR TR 2002-32, 2002.

[6] S. Huband, P. Hingston, L. While, and L. Barone, "An evolution strategy with probabilistic mutation for multi-objective optimization," in *Congress on Evolutionary Computation*, H. Abbass and B. Verma, Eds. IEEE, 2003, pp. 2284–2291.

[7] J. Knowles and D. Corne, "Properties of an adaptive archiving algorithm for storing nondominated vectors," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 100–116, April 2003.

[8] J. Knowles, D. Corne, and M. Fleischer, "Bounded archiving using the Lebesgue measure," in *Congress on Evolutionary Computation*, H. Abbass and B. Verma, Eds. IEEE, 2003, pp. 2490–2497.

[9] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Parallel Problem Solving from Nature VIII*, ser. Lecture Notes on Computer Science, X. Yao, E. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervos, J. A. Bullinaria, J. Rowe, P. Tino, A. Kaban, and H.-P. Schwefel, Eds., vol. 3242. Springer-Verlag, 2004, pp. 832–842.

[10] M. Emmerich, N. Beume, and B. Naujoks, "An EMO algorithm using the hypervolume measure as selection criterion," in *Evolutionary Multi-objective Optimisation*, ser. Lecture Notes on Computer Science, C. A. Coello Coello, A. H. Aguirre, and E. Zitzler, Eds., vol. 3410. Springer-Verlag, 2005, pp. 62–76.

[11] L. While, L. Bradstreet, and L. Barone, "A fast way of calculating exact hypervolumes," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 86–95, February 2012.

[12] K. Bringmann and T. Friedrich, "Approximating the least hypervolume contributor: NP-hard in general, but fast in practice," *CoRR*, vol. abs/0812.2636, 2008.

[13] L. Bradstreet, L. While, and L. Barone, "A new way of calculating exact exclusive hypervolumes," The University of Western Australia, School of Computer Science & Software Engineering, Technical Report UWA-CSSE-09-002, 2009.

[14] ——, "A fast incremental hypervolume algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 714–723, December 2008.

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

[16] J. Wu and S. Azarm, "Metrics for quality assessment of a multiobjective design optimization solution set," *Journal of Mechanical Design*, vol. 123, pp. 18–25, 2001.

[17] M. Fleischer, "The measure of Pareto optima: Applications to multi-objective metaheuristics," in *Evolutionary Multi-objective Optimisation*, ser. Lecture Notes on Computer Science, C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, Eds., vol. 2632. Springer-Verlag, 2003, pp. 519–533.

[18] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 29–38, February 2006.

[19] E. Zitzler, "Hypervolume metric calculation," 2001. [Online]. Available: *ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/hypervol.c*

[20] J. Knowles, "Local-search and hybrid evolutionary algorithms for Pareto optimisation," Ph.D. dissertation, The University of Reading, United Kingdom, 2002.

[21] L. Bradstreet, L. While, and L. Barone, "A faster many-objective hypervolume algorithm using iterated incremental calculations," in *Congress on Evolutionary Computation*, H. Ishibuchi, Ed. IEEE, 2010, pp. 179–186.

[22] L. While, L. Bradstreet, L. Barone, and P. Hingston, "Heuristics for optimising the calculation of hypervolume for multi-objective optimisation problems," in *Congress on Evolutionary Computation*, B. McKay, Ed. IEEE, 2005, pp. 2225–2232.

[23] C. M. Fonseca, L. Paquete, and M. López-Ibáñez, "An improved dimension-sweep algorithm for the hypervolume indicator," in *Congress on Evolutionary Computation*, C. L. P. Chen, Ed. IEEE, 2006, pp. 3973–3979.

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

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

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

[27] K. Bringmann and T. Friedrich, "Approximating the least hypervolume contributor: NP-hard in general, but fast in practice," in *Evolutionary Multi-objective Optimisation*, ser. Lecture Notes on Computer Science, M. Ehrgott, C. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, Eds., vol. 5467. Springer-Verlag, 2009, pp. 6–20.

[28] R. Everson, J. Fieldsend, and S. Singh, "Full elite sets for multi-objective optimisation," in *Proceedings of the 5th International Conference on Adaptive Computing in Design and Manufacture*, 2002, pp. 87–100.

[29] J. Bader, K. Deb, and E. Zitzler, "Faster hypervolume-based search using Monte Carlo sampling," in *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems*, ser. Lecture Notes in Economics and Mathematical Systems, M. Ehrgott, B. Naujoks, T. Stewart, and J. Wallenius, Eds., vol. 634. Springer-Verlag, 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 *Congress on Evolutionary Computation*, P. Haddow, Ed. IEEE, 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 *Foundations of Genetic Algorithms*, T. Jansen, I. Garibay, R. Wiegand, and A. Wu, Eds. ACM, 2009, pp. 87–102.

[32] K. Bringmann and T. Friedrich, "Don't be Greedy when Calculating Hypervolume Contributions," in *Foundations of Genetic Algorithms*, T. Jansen, I. Garibay, R. Wiegand, and A. Wu, Eds. ACM, 2009, pp. 103–112.

[33] T. Friedrich, C. Horoba, and F. Neumann, "Multiplicative approximations and the hypervolume indicator," in *Genetic and Evolutionary Computation Conference*, G. Raidl, Ed. ACM, 2009, pp. 571–578.

[34] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Congress on Evolutionary Computation*, R. Eberhart, Ed. IEEE, 2002, pp. 825–830.