The hypervolume indicator for multi-objective optimisation: calculation and use

This thesis is

presented to the

Department of Computer Science & Software Engineering

for the degree of

Doctor of Philosophy

of

The University of Western Australia

By

Lucas Bradstreet

April 2011

Abstract

Multi-objective problems, requiring the optimisation of two or more conflicting criteria, abound in the real world. Multi-objective optimisers produce solution sets that represent the trade-offs between problem criteria. As a result of computational and space limitations, a multi-objective optimiser is often unable to retain all generated trade-off solutions and instead must endeavour to keep the solutions that best cover the trade-off front. Indicators which map these sets into unary values that can be easily compared are valuable and are used frequently in multi-objective performance assessment, or as a part of the selection operator of a multi-objective optimiser.

One indicator which incorporates many mathematical properties favourable for use in multi-objective optimisation is the hypervolume indicator. Hypervolume is the n-dimensional space that is "contained" by a set of points. It encapsulates in a single unary value a measure of the spread of the solutions along the Pareto front, as well as the closeness of the solutions to the Pareto-optimal front. However, hypervolume has one serious drawback: calculating hypervolume exactly is NP-hard and exponential in the number of objectives.

This thesis describes research into improving the performance of hypervolume calculation and techniques for its use.

One major contribution of this thesis is the introduction of two new calculation algorithms, IIHSO and WFG, which outperform existing hypervolume calculation algorithms on several forms of test data. The best performing of the two, WFG,

greatly outperforms all existing algorithms on tested data sets and represents a substantial improvement in the feasibility of hypervolume calculation. The thesis also introduces a number of objective reordering heuristics which improve the typical performance of several existing hypervolume algorithms that use a dimension sweep approach.

The use of hypervolume within multi-objective optimisers is a relatively new and developing area. It is not yet known how to best apply hypervolume as part of the selection criteria within a multi-objective optimiser. Furthermore, hypervolume's high cost can limit its use within optimisers that require many solutions to be evaluated. However, it has great promise for use in optimisation if the performance issue can be limited.

In these hypervolume-based selection and archiving schemes, it is typically necessary to determine the solution that contributes the least hypervolume to a set. This thesis introduces the IHSO algorithm which quickly determines the hypervolume contributed by a solution. It also describes heuristics designed to improve IHSO's typical performance. Additionally, it describes and analyses techniques that reduce the calculations necessary to find the solution which contributes the least hypervolume.

When using hypervolume as part of a selection or archiving process, one important goal is to reduce the size of a solution set while maximising the hypervolume of this set. Finding the subset with the optimal hypervolume using naive schemes, results in combinatorial explosion in the number of hypervolume calculations required. For this reason, this thesis introduces and compares several simple meta-heuristic schemes that attempt to maximise the hypervolume of selected subsets. A local search and two greedy schemes are proposed in this thesis as means of overcoming this problem.

Acknowledgements

I would like to express my respect and gratitude to my supervisors and friends Dr. Luigi Barone and Dr. Lyndon While. They introduced me to research as an honours student and I thank them for providing me with first class guidance and support ever since. Their ideas, insights and encouragement were invaluable to me through my PhD. Thanks for everything, especially encouraging me and sticking with me when the task felt far too large. Luigi, thanks for all your ideas and readings of this thesis.

I would also like to thank the School of Computer Science & Software Engineering at The University of Western Australia for providing me with the infrastructure to complete this work. I would also like to thank The University of Western Australia for the financial support provided to me in the form of an Australian Postgraduate Award and a Postgraduate Travel Grant. The Association for Computing Machinery (A.C.M.) also assisted financially, providing a travel scholarship to present and attend a conference overseas.

I would also like thank the staff and students of the School of Computer Science & Software Engineering for their friendship and for providing such an interesting work environment. I've really enjoyed my time at university and I would like to acknowledge the many postgrads for helping make the time enjoyable — especially Rasha, Valance, Daniel, Bobby, and Angel.

I would like to thank Katherine Gasmire and Dad for proof reading.

Finally, I would like to thank all of my friends and my family, for listening, help,

patience, and support. I have enjoyed these years and my life wouldn't be the same without you all. Special thanks go to Rami, Lara, Tom and Ling. Martin, thanks for being a good brother and for all the late night chats. Most importantly, Mum and Dad, thanks for all of the support and encouragement you have provided throughout my life.

Papers Included in This Thesis

Paper 1 (Refereed)

L. While, L. Bradstreet, L. Barone, and P. Hingston. Heuristics for Optimising the Calculation of Hypervolume for Multi-Objective Optimization Problems. In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, pages 2225–2232, Edinburgh, Scotland, September 2005. IEEE Press

Paper 2 (Refereed)

L. Bradstreet, L. Barone, and L. While. Maximising Hypervolume for Selection in Multi-objective Evolutionary Algorithms. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 6208–6215, Vancouver, BC, Canada, July 2006. IEEE Press

Paper 3 (Refereed)

L. Bradstreet, L. While, and L. Barone. A Fast Incremental Hypervolume Algorithm. *IEEE Transactions on Evolutionary Computation*, 12(6):714–723, December 2008

Paper 4 (Refereed)

L. Bradstreet, L. While, and L. Barone. Incrementally Maximising Hypervolume for Selection in Multi-Objective Evolutionary Algorithms. In *2007 IEEE Congress on Evolutionary Computation (CEC'2007)*, pages 3203–3210, Singapore, September 2007. IEEE Press

Paper 5 (Refereed)

L. Bradstreet, L. Barone, and L. While. Updating Exclusive Hypervolume Contributions Cheaply. In *2009 IEEE Congress on Evolutionary Computation (CEC'2009)*, pages 538–544, Thronheim, Norway, May 2009. IEEE Press

Paper 6 (Refereed)

L. Bradstreet, L. While, and L. Barone. A Fast Many-objective Hypervolume Algorithm using Iterated Incremental Calculations. In *2010 IEEE Congress on Evolutionary Computation (CEC'2010)*, pages 179–186, Barcelona, Spain, July 2010. IEEE Press

Paper 7 (Submitted)

L. While, L. Bradstreet, and L. Barone. A Fast Way of Calculating Exact Hypervolumes, 2010. To appear in *IEEE Transactions on Evolutionary Computation*

Contribution By Candidate To Included Papers

Paper 1 50% contribution.
Developed a high performance implementation of the algorithm, implemented several heuristics, and co-wrote the paper.

Paper 2 90% contribution.
Designed and implemented the algorithms and techniques, wrote and presented the paper at the Congress on Evolutionary Computation 2006.

Paper 3 70% contribution.
Co-designed the algorithm, heuristics, and application techniques. Implemented the algorithm and performed experiments. Co-wrote the paper.

Paper 4 90% contribution.
Designed and implemented the algorithms and techniques and wrote the paper.

Paper 5 90% contribution.

Designed and implemented the algorithms and techniques, wrote and presented the paper at the Congress on Evolutionary Computation 2009.

Paper 6 60% contribution.

Co-designed the algorithm, heuristics, and application techniques. Implemented the algorithm and performed experiments. Co-wrote and will present the paper at the Congress on Evolutionary Computation 2010.

Paper 7 50% contribution.

Contributed to the algorithm, implemented some code, and performed the experiments.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This submission is composed of seven papers co-authored by the PhD candidate. The included papers are the result of research ongoing from 2005. Five are conference papers accepted at the IEEE Congress on Evolutionary Computation (CEC) over various years. CEC is very competitive, with an acceptance rate of around 50%, and rated in the top tier of computer science conferences by the Computing Research and Education Association of Australasia (CORE). One paper was published in the IEEE Transactions on Evolutionary Computation (TEC), one of the top ranked journals in the Artificial Intelligence (AI) discipline. The final paper has been accepted for publication in TEC.

This thesis investigates the use and calculation of hypervolume, an indicator that is widely used in multi-objective optimisation. Section 1.1 introduces and motivates the importance of multi-objective optimisation problems and multi-objective optimisers (MOOs), including multi-objective evolutionary algorithms (MOEAs). Section 1.2 introduces indicators useful in multi-objective optimisation and ways in which indicators can be used. Section 1.3 introduces the hypervolume indicator in more detail. Finally, Section 1.4 discusses the contributions made by this thesis to the field of multi-objective optimisation.

1.1 Multi-objective Optimisation and Multi-objective Evolutionary Algorithms

A multi-objective optimisation problem is one in which potential solutions are assessed by their performance in more than one criterion, or objective [32, 33]. The result of running MOOs is a set of solutions, with each solution representing a trade-off between objectives. MOEAs are evolutionary algorithms (EAs) applied to multi-objective optimisation problems.

An example of an application where MOEAs excel can be found in Barone et al. [7] with their MOEA for the design of rock crushers. Rock crushers are used to extract raw materials from rocks and are subject to various tunable parameters. Barone et al.'s MOEA attempts to simultaneously optimise two objectives: maximising the capacity of the rock crushers while minimising the size of the crushed product. An inevitable trade-off is found; reducing the size of the crushed rocks leads to reduced capacity, and vice versa.

MOEAs produce a range of solutions, each representing a different trade-off in objectives which could not be anticipated prior to optimisation. As such, MOEAs allow their users to respond effectively to change (e.g. varying market conditions) without the need to rerun the optimisation algorithm. As it may take a considerable period of time to rerun an optimiser to respond to changing conditions, MOEAs represent a significant advantage for a problem solver who wishes to adapt quickly.

1.2 Multi-objective Indicators

MOEAs produce a solution set that represents discovered trade-offs. These trade-offs result in several issues which are not present in single objective optimisation. The solution sets produced by MOEAs are difficult to compare against one another as generally one set is not decidedly superior to another.

The inability to qualitatively compare optimisation results can be extremely limiting to multi-objective optimisation research. As a result, several unary indicators have been created to measure the quality of solution sets. These map a solution set in two or more objectives to a single value that can be used to easily compare the quality of solutions sets. All indicators have advantages and disadvantages, however good indicators factor in coverage, diversity, and relative goodness of the solution set [77].

1.3 The Hypervolume Indicator

One of the most popular indicators for MOOs is hypervolume, otherwise known as the S-metric [39, 60, 69] or Lebesgue measure [52].

If solutions are considered as points in objective space, hypervolume is the n-dimensional space that is contained by a solution set — i.e. the n-dimensional volume of the set relative to some reference point. Due to properties of the indicator [77], a set with a larger hypervolume is likely to present a better set of trade-offs than sets with lower hypervolume. Unfortunately, use of hypervolume has been greatly limited by the high computation cost of existing hypervolume calculation algorithms and application techniques [6, 30, 36].

1.3.1 Use of Hypervolume for Performance Assessment

Indicators are commonly used to compare solution sets produced by MOOs. Through use of these measures, one can gain a better understanding of how MOOs function, such as how changes in parameters, selection methods, and other techniques affect the optimisation process and the resulting quality of the solutions. Indicators are also used to compare MOOs directly. Indeed, many discussions of MOOs compare their indicator values with those of popular MOOs on benchmark problems using statistical tests over many runs. The PISA framework [12] simplifies this kind of analysis for popular indicators and is used widely for benchmarking MOOs (e.g. [6, 30, 61, 71]) and new test problems (e.g. [15]).

1.3.2 Use of Hypervolume in Selection

EAs operate via an iterative process emulating natural selection, adaptation, and genetic inheritance [4]. A key issue for MOEAs, a popular form of multi-objective optimiser, is that of mating selection.

MOEAs attempt to find the best possible approximation solution set that represent the trade-offs between conflicting problem criteria. In MOEAs, this occurs via a process of selection and reproduction. In single objective EAs, selection is relatively simple: rank the solutions by the objective of interest and select a proportion of the best solutions. In MOO, this process is more complicated. Many MOOs separate solutions into many distinct progressively worsening ranks, such that the solutions of a lower rank are superior to a solution in all higher ranks, and within each rank there exist no solutions that are superior to each other.

Unfortunately, populations are generally bounded in size and as such it is not always possible to keep an entire rank of solutions between generations. Therefore, the optimiser must choose a subset of these incomparable solutions to use in the next generation. To ensure diversity in the population, it is important that the chosen solutions are spread over the entire range of possible trade-offs. Additionally, as the

number of criteria increase, the number of potential solutions contained in each rank can increase exponentially [38, 61]. As such, better selection methods are required to improve the performance of these optimisers.

In multi-objective optimisation, indicators for performance assessment, such as hypervolume, are usually adaptable for selection. After all, if one set is better than another under a performance measure, the set should be more desirable during the course of the run of the EA. Therefore, when reducing the size of a set during selection, a MOEA could attempt to maximise an indicator over the reduced size solution set.

1.4 Contributions

The primary contribution of this thesis is in-depth research into hypervolume calculation algorithms and techniques for its use. Hypervolume's favourable properties have made it extremely popular in current multi-objective optimisation research. However, hypervolume use has been held back by algorithms that perform poorly [64] and recent research has proven exact hypervolume calculation to be NP-hard [21].

This thesis primarily focuses on improving the feasibility of hypervolume use in multi-objective optimisation.

These contributions fall into three key areas:

- The development of two faster hypervolume calculation algorithms, IIHSO and WFG, that can be used to calculate higher objective problems considered infeasible for calculation using previous hypervolume calculation algorithms. These algorithms are primarily intended for use in performance assessment.

- The development of the IHSO algorithm, a fast exclusive hypervolume contribution calculation algorithm (the individual hypervolume attributable to one

particular solution) and a search technique that minimises the computation required to find solutions that contribute the least hypervolume.

- The introduction and comparison of several hypervolume based set selection techniques that can be used within MOOs. These techniques aim to improve the selection operation within an EA by maximising the hypervolume of solution sets of bounded size.

1.5 Organisation of the Thesis

As conference papers often have tight page restrictions, a thesis submission by a collection of papers is inevitably different to a conventional thesis. Background material in conference papers is often condensed with only the most relevant and important concepts and ideas included in the paper. As such, this thesis is structured to include some of the material omitted from these papers.

Chapter 2 introduces background information explaining why the hypervolume indicator is important to multi-objective optimisation, discusses algorithms to calculate hypervolume and techniques for hypervolume's use. Existing hypervolume calculation algorithms are examined, including their limitations, and their performance cost. Issues relevant to the use of hypervolume within MOOs are also discussed.

Chapter 3 describes the published papers which make up this dissertation. These are presented in an order that preserves the logical separation of ideas, not necessarily in order of publication.

Chapter 4 presents the seven papers that comprise this submission.

Chapter 5 summarises the research discussed in the published papers, highlighting their significance and contribution to hypervolume use in multi-objective research. The implications of these contributions in the context of future uses of hypervolume are examined.

Chapter 2

Background

This chapter presents a review of background material not discussed in the included papers making up this thesis. The first section introduces multi-objective optimisation and discusses two key areas of research: performance assessment indicators which are used to compare multi-objective optimiser runs, and selection indicators which are used during optimisation to choose candidate solutions that drive optimisation toward high quality results. Hypervolume, a popular indicator, can be used for each of these purposes.

The second part of this chapter examines hypervolume calculation algorithms, with a primary focus on its use in performance assessment. The final section investigates the calculation and application of hypervolume as a selection indicator within MOOs.

2.1 Definitions

In a multi-objective optimisation problem, the goal is 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 or *Pareto optimal front* is the set of all possible Pareto optimal vectors.

Relations commonly used to compare these vectors include:

Pareto dominance	$A \prec B$
Weak Pareto dominance	$A \preceq B$
Comparable	$A \preceq B \| \| B \preceq A$
Incomparable	$\neg(A \preceq B \| \| B \preceq A)$
Indifferent	$A \preceq B \wedge B \preceq A$

Pareto dominance occurs when $\overline{x}$ is better than $\overline{y}$ in one objective and equal or better in the remaining objectives. Under weak Pareto dominance, $\overline{x}$ is not worse than $\overline{y}$ in all objectives. Under these definitions, a set is a Pareto optimal set if no solutions exist that weakly dominate solutions in the set. Precise definitions of these terms can be found in [4, 77].

Throughout this thesis, m is used to refer to the size of non-dominated fronts, and n for the number of dimensions (objectives) in vectors contained in the front.

2.2 Indicators

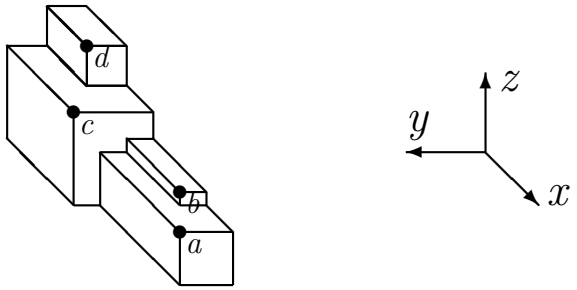
The ultimate goal in multi-objective optimisation is to find the Pareto optimal front for a problem. However, in practice, MOOs can only aim to find a representative subset. As this subset typically contains worse solutions than the Pareto optimal front, or does not contain every possible solution (of which there can be an innumerable number for continuous problems), there exists a need to measure the quality of

these fronts. Many *indicators*, also known as *metrics*, have been designed for this purpose. Measuring the quality of these approximation sets is a difficult problem and any evaluation should factor in the distance of the set from the Pareto optimal front and the spread of the solutions in the objective space [77].

2.3 The Hypervolume Indicator

The hypervolume indicator [60] or S-metric [69] has become widely used in recent years. Hypervolume is the n-dimensional space that is "contained" by an n-dimensional set of points. When applied to multi-objective optimisation, the n-dimensional objective values for solutions can be treated as points. That is, the hypervolume of a set is the total size of the space dominated by the solutions in the set (see Figure 1).

Hypervolume encapsulates in a single unary value a measure of the spread of the solutions along the Pareto front, as well as the distance of the set from the Pareto-optimal front. Additionally, it has several favourable mathematical properties. It was the first unary metric to detect when a set of solutions are not worse than another set for all solution pairings [77]. Additionally, it is maximised if, and only if, the set of solutions contains all Pareto optimal points [39].



	x	y	z
a	11	4	4
b	9	2	5
c	5	6	7
d	3	3	10

Figure 1: Example hypervolume in three dimensions. Points are marked by circles and labelled with letters. The hypervolume of the set is the volume of the space covered by points a–d. Figure reproduced from [67].

The hypervolume of a set is measured relative to a reference point, usually the anti-optimal point or "worst possible" point for the space. The choice of reference point is important as it can influence the conclusions resulting from the comparison of hypervolumes. While still an open problem, one suggestion is to take the worst known value in each objective and shift it by an appropriate amount [54].

2.4 Hypervolume Metric Algorithms

Though hypervolume has been found to have many favourable properties as a multi-objective indicator, its high computational complexity limits its use. Indeed, hypervolume calculation has been proven by Bringmann and Friedrich [23] to be #P-hard (analogous to NP-hard for counting problems) in the number of objectives and therefore no polynomial time algorithm exists unless $NP = P$. As a result of traditionally poor performance, hypervolume algorithms have been used primarily for performance assessment (as a metric). Typically, these are used to compare the performance of MOOs using indicator values calculated using resultant solution sets.

Many algorithms have been created to calculate hypervolume, each with a different worst-case complexity. This section introduces the main algorithms proposed to calculate hypervolume, and discusses their time complexity, in the process demonstrating the evolution of hypervolume calculation algorithms.

2.4.1 Hypervolume via Inclusion-Exclusion

Inclusion-Exclusion is the most obvious and easily understood method for calculating hypervolume [68]. The Inclusion-Exclusion hypervolume algorithm works by the inclusion-exclusion principle in combinatorial mathematics: the algorithm adds volumes of rectangular polytopes (n-dimensional rectangular volumes) dominated by each point individually, then subtracts volumes dominated by intersections of

pairs of points, then adds back in volumes dominated by intersections of three points, and so on.

Pseudo-code for this algorithm is shown in Figure 2. Unfortunately, while simple, this method has a time complexity of $O(n2^m)$ that makes it infeasible on all but the smallest sets (it is exponential in the number of points!).

```
InclusionExclusion(ps):
  volume = 0
  for each non-empty subset s of ps
    vol = vol + intersection(s) * (-1)^(|s|+1)
  return vol

intersection(ps):
  returns the volume of the largest rectangular
  polytope that is dominated by all members in ps
```

Figure 2: Pseudo-code for the inclusion-exclusion algorithm. Code reproduced from [67].

2.4.2 The LebMeasure Hypervolume Algorithm

The LebMeasure algorithm by Fleischer [39] is based on the observation that for any space covered by a set of non-dominated points, one can always identify a rectangular polytope that does not intersect with any other region which can be "lopped off". The hypervolume contributions of these lopped off regions are easily calculated. The hypervolume of the space dominated by these polytopes is added to a hypervolume accumulator, and new points are spawned to reflect the removal of this region. This process can then be repeated until the remaining polytopes no longer dominate any region of space.

Figure 3 demonstrates lopping and spawning in LebMeasure. Pseudo-code for this algorithm is shown in Figure 4.

LebMeasure was initially thought to have polynomial time performance, however it was later demonstrated empirically by While [64] to exhibit exponential time complexity in the number of objectives. While also proves the lower bound for

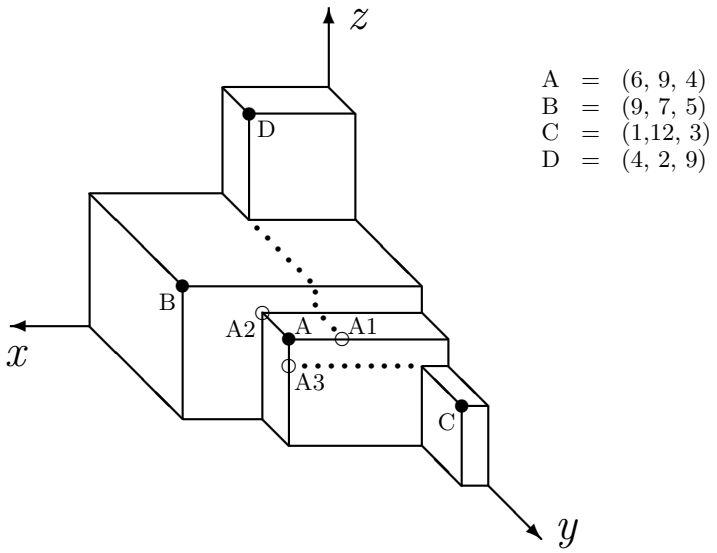


Figure 3: Spawning in LebMeasure. The blocks denote the hypervolume dominated by A, B, C, D in a maximisation problem relative to the origin. The filled circles represent the four points, and the empty circles represent the three potential spawns of A. Each spawn is generated by reducing one objective to the largest smaller value from the other points. It is clear that A2 is dominated by B, so A is replaced by A1 and A3. Picture and caption reproduced from [67].

LebMeasure's worst case complexity to be $O(2^{n-1})$, and thus exponential in the number of objectives.

2.4.3 The HSO Hypervolume Algorithm

Hypervolume by Slicing Objectives (HSO) is a hypervolume calculation algorithm that processes a front by processing one objective at a time, "slicing" along the chosen objective [50, 67, 70]. This is known as a dimension-sweep algorithm.

HSO is given with a front that is presorted in the first objective. Point values in this objective are used to create cross-sectional slices along this objective. When sweeping along an objective, each point in the list is visited in turn. A list of points is maintained which is sorted in the $n-1$th-objective, containing points that have been processed thus far, i.e. the points contributing to the current slice. As each slice is an $n-1$-objective hypervolume, its hypervolume is calculated recursively and multiplied by the depth of the slice (the difference between the current point

```
LebMeasure (ps):
  pl = a list containing the points from ps in some order,
       each paired with n, the number of objectives
  hypervolume = 0
  while pl is not empty
    (p, z) = head (pl)
    pl = tail (pl)
    a = oppositeCorner (p, pl)
    hypervolume = hypervolume + volBetween (p, a)
    ql = spawns (p, z, a, pl)
    prepend ql to pl
  return hypervolume

spawns (p, z, a, pl):
  return up to z spawn points that collectively
  dominate the remainder of ps exclusive hypervolume

oppositeCorner (p, pl):
  returns the point that bounds the largest rectangular polytope that is
  dominated exclusively by the point p relative to the points on pl

volBetween (p, q):
  return the hypervolume between the points p and q
```

Figure 4: Pseudo-code for the LebMeasure algorithm. Each point on the main stack is paired with the highest index that it can use to generate a non-dominated spawn. Code reproduced from [64] with slight modification.

value and the next point value). The point is then added to the $n-1$-objective slice, after removing any points that it dominates. This process repeats until every point in the list has been visited. This slicing process is depicted in Figure 5. Pseudo-code for HSO is shown in Figure 6.

While et al. [67] prove that HSO is exponential in the number of objectives (roughly $O(m^{n-1})$). This is a significant improvement compared to the inclusion-exclusion approach (recall, its time performance is exponential in the number of points), and While et al. show that HSO greatly outperforms LebMeasure for all tested DTLZ [35] and randomly generated front types. However, HSO still performs poorly for data sets with high dimensionality.

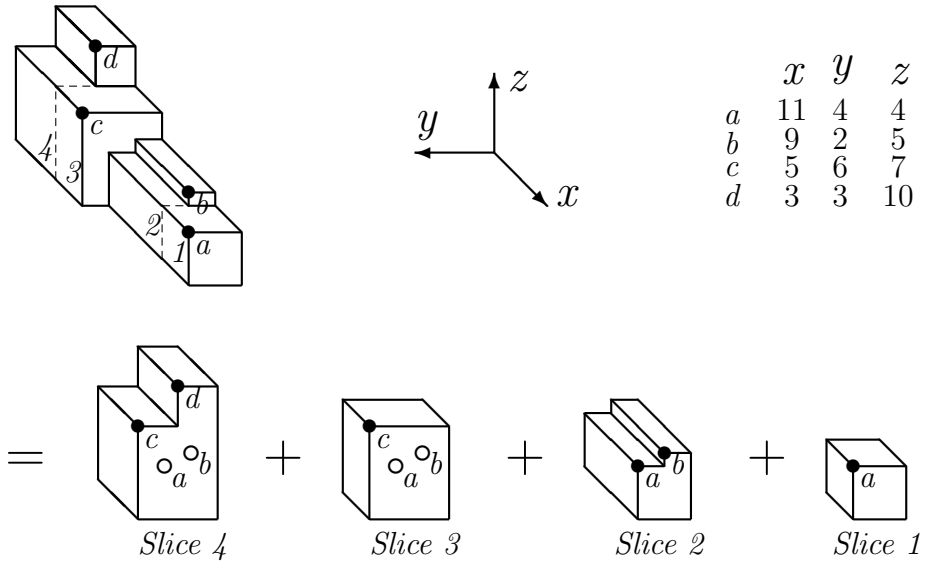


Figure 5: One step in HSO for the four 3D points shown. Objective x is processed, leaving four two-objective shapes in y and z. Points are marked by circles and labelled with letters: unfilled circles represent points that are dominated in y and z. Slices are labelled with numbers, and are separated on the main picture by dashed lines. Figure reproduced from [67].

```
hso (ps, n):
  if n is 2
    return hv2D (ps)
  ps2 = empty point list
  for each p in ps
    depth = distance from p to next point in ps
            or to the reference point if p is the final point
    insert (ps2, p, n-1)
    hypervolume = hypervolume + depth * hso (ps2, n-1)
  return hypervolume

insert (ps, p, n):
  insert p into list ps, maintaining sorting in objective n,
  and deleting all points in ps dominated by p

hv2D (ps):
  return the 2D hypervolume of ps
```

Figure 6: Pseudo-code for HSO. HSO assumes list *ps* is sorted in objective n. Code reproduced from [67].

2.4.4 Optimal 3D Hypervolume Algorithm

Paquete et al. [8, 58] describe an optimal algorithm for calculating hypervolume in three dimensions. It is a dimension sweep algorithm which uses the relationship between hypervolume computation and the process of finding the maxima of a point set [51]. The algorithm sweeps along a front sorted in one objective, maintaining an overall 2D area for the points considered thus far.

For each point, p, in the front, a height balanced binary tree is queried to determine the position of p in the remaining objectives. If p is dominated, it is discarded. If p dominates other points, they are deleted from the tree. If needed, the 2D area is then updated in constant time. The height from p to the next point down (i.e. the slice depth) is then multiplied by the area and the result added to the overall volume. Pseudo-code is shown in Figure 7.

```
Paquete3D (ps):
  initialise tree, sort ps in 3rd objective, set volume to 0
  p = head (ps)
  ps = tail (ps)
  area = p[0] * p[1]
  z = p
  for each p in ps
    search tree for point q to the right of p
    if p is not dominated
      increase volume by slice between z and p
      z = p
      for each point s in tree dominated by p
         remove s from tree
         decrease area by contribution of s
      increase area by contribution of p
      insert p in tree
  increase volume by area * z[2]
```

Figure 7: Pseudo-code for optimal 3D hypervolume algorithm by Paquete et al. Code derived from [8].

As each tree insertion and deletion has a cost of $\log m$ and there are at most m insertions and deletions, this algorithm has a worst case complexity of $O(m \log m)$, superior to the $O(m^2)$ worst case complexity of HSO in 3D.

2.4.5 The FPL Hypervolume Algorithm

The Fonseca Paquete López-Ibáñez (FPL) hypervolume algorithm is another dimension-sweep algorithm [40] which improves upon HSO in three principal ways:

1. It adds a new linked data structure which reduces the work required to maintain the fronts built iteratively by HSO. One necessary change that results from this structure is that dominated points must be retained, as points must be reinserted in the reverse order of their deletion. Therefore, dominated points are marked instead of deleted and are skipped over in lower objectives. This data structure improves performance by minimising the number of comparisons necessary to maintain the sorting within the $n-1$-dimensional slices.

2. It reuses previous calculations when a smaller dimensional slice has already been calculated. Hypervolumes are stored along with the current coordinate in the current objective. As these values become stale, bound values which keep track of reusable hypervolumes are updated whenever points are deleted or reinserted.

3. It uses the 3D algorithm by Paquete et al. [58] as a base case.

Fonseca et al. state that the worst-case complexity of FPL is $O(m^{n-2} \log m)$ [40]. The $\frac{m}{\log m}$ complexity improvement over HSO is due to the new 3D base case.

2.4.6 Hypervolume Using the Overmars and Yap Algorithm

The Hypervolume Overmars and Yap (HOY) algorithm by Beume and Rudolph [11] adapts an algorithm by Overmars and Yap [56] for a computational geometry problem that is similar to the hypervolume problem. The Klee's measure problem (KMP) [48] occurs in computational geometry: calculate the size of the union of a set of nD hyper-cuboids. The Overmars and Yap algorithm [56] solves the KMP in

$O(m \log m + m^{n/2} \log m)$ time. By converting each point to a hyper-cuboid anchored at the reference point, Beume and Rudolph adapt this algorithm to calculate the hypervolume of a front in $O(m \log m + m^{n/2})$ time. The $\log m$ improvement in complexity over the Overmars and Yap algorithm comes from an optimisation resulting from all hyper-cuboids being anchored at the same place.

HOY starts with the smallest rectangular polytope region anchored at the reference point which contains all of the points in the front. This region is then split recursively, creating two smaller regions at each step. A point p *intersects* a region r if p lies within r for the current splitting objective and in at least one already-processed objective. Conversely, p *non-intersects* r if p lies within r in the current splitting objective and in no already-processed objectives.

The region is split in the current objective at the median of the intersecting points if any exist; otherwise, it is split at the median of the non-intersecting points if enough exist; otherwise the splitting objective is increased. HOY recurses with each sub-region and the points that partially cover that sub-region. Every region is split until one of the following holds:

- The points completely cover the region.
- The points partially covering the region form a trellis, meaning that the hypervolume covers the region in all objectives bar one; now the size of the hypervolume in the region can be calculated quickly using the well-known inclusion-exclusion algorithm.

Figure 8 shows pseudo-code for HOY.

2.4.7 Hypervolume Approximation

Though recent exact hypervolume algorithms [11, 40, 65] have led to improved feasibility or better worst-case time complexities, hypervolume calculation will remain NP-hard and exponential in the number of objectives [23]. It is therefore likely that

```
HOY (ps, region, split):
  for each p in ps
    if the region is completely covered by p in d-1 objectives
      add volume of covered region
      remove points from p onwards from ps

  return if ps is empty

  if points in ps form a trellis over the region
    add volume of trellis
  else
    if at least one member of ps intersects the region
      split the region with the median of the list of intersecting points
      HOY (ps, region1, split)
      HOY (ps, region2, split)
    else if at least sqrt(N) members of ps non-intersects the region
      split the region with the median of the list of non-intersecting points
      HOY (ps, region1, split)
      HOY (ps, region2, split)
    else
      HOY (ps, region, split+1)

intersects (p, region, split):
  p intersects region if it is within region's bounds
  for split and one or more dimensions from 0 to split-1

non-intersects (p, region, split):
  p non-intersects region if it is within region's bounds
  for split and intersects (p, region, split) is false
```

Figure 8: Pseudo-code for the Beume and Rudolph's HOY algorithm. N is the size of the initial front. Code derived from [11, 22].

exact calculation will remain infeasible for problems with a great number of objectives. Hence, there is merit to approximating a set's hypervolume if performance can be significantly improved and accuracy is within tolerable limits. One approximation approach by Everson et al. [37] uses Monte Carlo sampling to approximate expensive hypervolume calculations, however this approach does not guarantee a bound on the error. A recent approach by Bringmann and Friedrich [23] has a polynomially bounded error and shows promise. Approximation using scalarising functions, has also been proposed [47].

While hypervolume approximation techniques are an important area of hypervolume research and use, this thesis primarily focuses on exact calculation algorithms. One reason is that it is desirable for hypervolume calculations to be accurate when hypervolume is used for performance assessment as optimiser comparisons require accurate conclusions.

Hypervolume approximation techniques require a trade-off between accuracy and performance. However, the accuracy of hypervolumes needed for different problem instances, and thus the necessary samples and runtime required, is relatively unknown prior to optimisation. It is thus difficult to know whether the benefits of such a trade-off is worthwhile ahead of time. Hypervolumes can range greatly depending on the data, reference point used, point normalisation, and other factors.

The improving performance of exact algorithms has greatly increased their feasibility for real world multi-objective problems e.g. FPL [40] can calculate hypervolumes for tested data in approximately 1.5–2 more objectives than HSO [67], and HSO has a similar additional advantage over LebMeasure [39]. New exact algorithms may continue to improve the feasibility of exact hypervolume calculation in this way. Indeed Chapter 3 shows that IIHSO and WFG further improve the feasibility of hypervolume calculation significantly. Additionally, many of the ideas and algorithms for exact hypervolume calculation may be adaptable for use in approximation techniques.

However, as mentioned above, it is still likely that hypervolume approximation techniques will remain necessary for problems in many objectives (although techniques such as objective reduction are an alternative).

2.5 Use of Hypervolume Within MOOs

While hypervolume has predominantly been used for performance assessment, recent research has investigated its use within MOOs. It is ideally suited for use in multi-objective optimisation, as it is one of the only indicators that both preserves dominance and guarantees that only the Pareto-optimal set results in the maximum potential hypervolume for a problem. As expected, indicators which result in the mating of solutions that best cover the Pareto front and drive optimisation closer to the Pareto optimal front are highly desirable within an MOEA [77].

The LebMeasure algorithm introduced the use of hypervolume for bounded archiving, maintaining an archive of solutions found by the optimisation algorithm during a run. However, a problem arises when this archive is bounded in size, as eventually the number of non-dominated points contained in the archive will overflow the size of the archive. At this point, an indicator is needed to decide which points should be discarded to maintain a high quality archive.

The Lebesgue Archiving HillClimber (LAHC) approach described by Knowles et al. [39] locally maximises the hypervolume of the archive by removing the point with the least exclusive hypervolume contribution whenever the archive is full and a new point is available. LebMeasure is used for this purpose, as it can directly determine the exclusive contribution of a single point relative to the overall set of points. Unfortunately, LebMeasure used this way still suffers from poor performance, as its computational complexity remains exponential in the number of objectives.

The LAHC archiving approach improves the chance that desirable individuals encountered during the search process remain when the optimisation process terminates. Although archiving schemes do not drive the optimisation process itself,

LAHC was a precursor to the use of hypervolume within the selection stage of MOEAs.

Most popular MOEAs employ a Pareto-based ranking scheme for mating selection. In the non-dominated ranking scheme by Goldberg [41], solutions are ranked higher (worse) than the highest ranked solution that dominates them. Equally ranked solutions form a non-dominated set.

Selection using the simple non-dominated ranking scheme above operates as follows:

1. Rank individuals into non-dominated fronts using Pareto dominance.

2. Starting from the best ranked front, iteratively append each front to the population until a front is too large to fit in its entirety.

3. If the previous front was too large, select a subset of the front by random selection or use of an indicator.

Many methods have been designed to select from equally ranked solutions, with some of the most popular being *crowding distance* (used in NSGA-II by Deb et al. [34] to measure the spread between solutions), *strength* (used in SPEA2 by Zitzler et al. [74]), and the *binary hypervolume* and *binary epsilon* indicators (used in IBEA by Zitzler and Künzli [73]). Some MOEAs, including the commonly used NSGA-II and SPEA2, perform quite well on two objective problems, however they often perform badly on problems in additional objectives [46, 61]. Therefore, better selection indicators are needed and hypervolume appears promising for this purpose as it incorporates aspects of both distance and spread.

Brockhoff et al. [25] provide a theoretical analysis of the selection process within hypervolume based MOEAs which iteratively remove individual solutions with the least contribution from fronts. Additionally, these type of MOEAs have been found to suffer from premature convergence [10, 76] which may lead to lower quality solutions to be produced.

2.5.1 SMS-EMOA MOEA

The Lebesgue Archiving Hillclimber [39] introduced the use of hypervolume to maintain a bounded-size solution archive. Emmerich et al. [36, 54] take this approach one step further in their SMS-EMOA MOEA by using hypervolume throughout optimisation as an indicator in selection. SMS-EMOA considers a single offspring per generation, known as a steady state EA.

In SMS-EMOA, each newly created solution is ranked and a solution is removed from the worst ranked front in order to maintain the population size. The solution that contributes the least to the hypervolume of the worst ranked front is discarded. Due to the expense of calculating hypervolume contributions, SMS-EMOA has only been applied to 2D and 3D problems but has had success for certain applications [53, 54].

By using a steady state EA, SMS-EMOA circumvents a quality consideration in hypervolume based MOEAs: how to maximise the hypervolume of a selected front subset when more than one solution must be discarded? Unfortunately, a steady state MOEA is not ideal for all problem types and suffers from the issues discussed in the previous section.

2.5.2 Weighted Hypervolume and SIBEA MOEA

Zitzler et al. [71] also investigate the use of hypervolume within an MOEA. Their approach uses weighted hypervolume to capture user preferences for desirable regions of the objective space. This is achieved by first defining hypervolume as an attainment function [42], and then formulating hypervolume as an integral over the product of the attainment function and a weight distribution function. The use of weighted hypervolume allows a user to modify the weight function to articulate preferences for regions of the objective space.

While these techniques may be desirable for performance assessment, they were

primarily created to be used within MOOs, allowing user preferences to guide optimisation toward desired regions of the objective space. Recent research by Auger et al. [1] has investigated approaches for articulating these preferences.

To test the use of weighted hypervolumes in optimisation experimentally, Zitzler et al. introduce a basic weighted hypervolume based MOEA. Unlike the steady state SMS-EMOA, the Simple Indicator-Based Optimization Algorithm (SIBEA) generates more than one offspring in each generation. Non-dominated selection is used via the Goldberg scheme [41] described in Section 2.5, with greedy hypervolume based selection performed inter-rank. The solution with the smallest weighted hypervolume contribution is removed from the best ranked unselected front until the reduced front can be added to the population.

Initial proof of principle experiments demonstrate that weighted integration used within SIBEA achieves the desired bias preferences in guiding optimisation. The relative performance of SIBEA compared to other MOEAs is not assessed, as SIBEA was only introduced as a platform to test the weighted hypervolume indicator.

2.5.3 Objective Reduction Approach Using Hypervolume

The complexity and real world performance of hypervolume calculation limits its use in problems with more than three objectives. Brockhoff and Zitzler [29, 30] introduce the use of objective reduction [24, 26–29, 31] into hypervolume based MOEAs. Objective reduction aims to omit the least important problem objectives from consideration. The performance of hypervolume based MOEAs could be greatly improved by objective reduction, as hypervolume calculation is exponential in the number of objectives.

However, the omission of objectives can affect the underlying dominance structure for a problem (i.e. dominance relations between solutions). Brockhoff and Zitzler find these relations to be affected in two key ways: comparable solutions can become indifferent, and incomparable solutions can become comparable or indifferent.

Objective reduction schemes should attempt to minimise these effects by finding a subset of objectives that best preserves the underlying dominance structure for the solution set.

Brockhoff and Zitzler introduce two methods to determine which objectives should be omitted:

1. k-EMOSS (Minimum Objective Subset of Size k with Minimum Error) — the problem of finding an objective subset of predefined size k that minimises error.

2. δ-MOSS (Minimum Objective Subset Problem) — the problem of finding the δ-minimum objective set. This is the objective subset with the smallest size while preserving the original dominance structure with a maximum error of δ.

For each of these methods, a measure of error is required. Brockhoff and Zitzler choose the additive ϵ indicator [77] for this purpose.

These techniques have been tested and incorporated into the SIBEA MOEA [71], which uses the hypervolume indicator for selection [30]. After dimensionality reduction is applied, SIBEA operates as described in Section 2.5.2, using hypervolume contributions to reduce the size of non-dominated fronts. Brockhoff and Zitzler find that SIBEA can be improved considerably using k-EMOSS objective reduction, however they find δ-MOSS objective reduction to be relatively ineffectual.

As existing exact hypervolume algorithms exhibit exponential time complexity in the number of objectives, objective reduction is a compelling approach to reducing the high cost of using hypervolume within a MOEA. While objective reduction techniques do not replace the need for faster hypervolume calculation algorithms, it is evident that they could greatly increase the use of hypervolume on otherwise infeasible problems.

2.5.4 Selection Using Hypervolume Approximation and the HypE MOEA

Approximation techniques are an alternative to objective reduction techniques for improving the performance of hypervolume based optimisation algorithms. Bader, Deb and Zitzler [5] have also developed a sampling method that approximates the rankings of individuals determined by the hypervolume indicator. Recall that rankings, rather than actual hypervolume values, are used for selection. Recently, this hypervolume ranking approximation technique has been incorporated, and improved upon, in the HypE (Hypervolume Estimation Algorithm) MOEA by Bader and Zitzler [6].

Whereas SIBEA uses a greedy scheme to remove the least contributing point at each step, HypE approximates the expected loss in contribution resulting from the removal of a point from a front, i.e. the approximate average contribution loss over all subset combinations of that size. Bader and Zitzler find that using the expected loss measure improves the quality of selections substantially compared to a greedy scheme that only considers the loss resulting from a single point. HypE uses a partition splitting and aggregation approach to avoid calculating the hypervolume of all subset combinations. Hypervolume contributions for points in each partition are considered to equally share the hypervolume for that partition whether they are non-dominated or not. Using this approach, it possible to calculate the expected contribution loss with a worst case complexity that remains $O(m^n + nm \log m)$.

When HypE is used on problems with more than three objectives, expected contribution losses are approximated using the hypervolume Monte Carlo technique by Bader, Deb, and Zitzler [5]. Initial results on the DTLZ [35] and WFG [43, 44] test problems and the Knapsack problem [75] are extremely promising, with HypE generally outperforming popular MOEAs, including an implementation which mimics SMS-EMOA.

2.5.5 Optimal Hypervolume λ-set Selection

MOOs using greedy subset selection techniques have been shown to suffer from non-convergence problems [10]. HypE [6] provides a qualitative improvement on greedy hypervolume selection schemes. However, it is still possible for HypE to make poor selections as its expected loss measure is only an estimation. Ultimately, MOOs would perform best if they could determine the optimal subset that maximises hypervolume.

Auger et al. [3] provide an analysis of optimal subset selections on several 2D ZDT [72] and DTLZ [35] problems. Additionally, they investigate how the choice of reference point affects optimal selections as well as extremal solutions. In order to include extremal points, Auger et al. recommend a reference point much larger than solution objective values.

Unfortunately, simple algorithms that find optimal subsets by evaluating the hypervolume of every subset combination, are almost always infeasible. Until very recently, no feasible solutions to this problem existed. Additionally, few experiments had even been performed to compare the performance of simple hypervolume selection schemes. However, two algorithms have recently been proposed which should considerably improve the worst-case time complexity of finding the λ-subset with the optimal hypervolume.

In an another paper, Auger et al. [2] describe an exact algorithm which solves the 2D hypervolume optimal subset selection problem in $O(m^3)$ time. Auger et al. use dynamic programming to build optimal subsets using subsets containing one fewer point calculated in previous iterations. This optimal subset selection method is incorporated into an MOEA using weighted hypervolume and shown to perform well on bi-objective problems [71].

Similarly, Bringmann and Friedrich [22] have introduced a technique which determines the optimal λ-subset selection. n-dimensions. This technique is implemented

within the HOY algorithm with subset contributions calculated in parallel. Calculating these contributions directly within HOY results in only an additive term of m^{λ}, where $m = \mu + \lambda$, to HOY's worst case complexity of $O(m \log m + m^{n/2})$. This is a huge improvement in feasibility compared to the multiplicative increase that results from the calculation of the hypervolumes of every subset combination using HOY.

Their algorithm operates like HOY, with a few key modifications. At the leaf-regions the volumes of all λ-subsets are computed. To do so, fully covering points must be passed to recursive calls as numerous subset volumes are computed. The hypervolumes of these subsets are calculated in the trellis calculation step and stored in a dynamic hash table. Finally, λ-subset hypervolumes are determined by combining their smaller subsets. These exclusively dominate space that no other subsets dominate.

Whereas HypE relies on an expected loss heuristic, which can result in inferior front selections, Bringmann and Friedrich's algorithm will always determine the optimal subset selection. However, when the number of possible subset combinations is large, finding the optimal subset is usually infeasible. Even moderately small numbers for m and λ will likely lead to an infeasible computational cost. This cost is in addition to the expensive $m^{n/2}$ cost of the HOY algorithm. As HOY exhibits reasonably poor performance in high numbers of objectives (see *Paper 6*), this technique is likely to be infeasible for all but fronts with small m, λ, and n. However, no performance figures have been published, so the actual performance of this technique is unknown.

To improve the feasibility of this approach, Bringmann and Friedrich discuss a compromise between a greedy selection scheme and calculating the optimal subset. This scheme should improve on the accuracy of a purely greedy scheme without the full cost of calculating the optimal subset. However, the actual performance of this hybrid remains unknown.

Chapter 3

Overview of Included Papers

This chapter presents an overview of the included papers that make up this thesis. Extra commentary and discussion about each included paper is given, along with a discussion of the context of the paper within the overall research themes contained in this thesis.

The hypervolume indicator is an interesting and fertile area due to its useful properties, increasingly common usage, varied applications, and the computational issues that prevent its wide-scale use for many-objective problems. The initial focus of this thesis was on improving hypervolume's use for performance assessment, and later on improving its use for selection in MOEAs.

The papers discussed here cover two main themes: improving the performance of hypervolume when used for performance assessment, and exploring issues relevant to using hypervolume for selection.

3.1 Experimental Test Data

All papers contained in this thesis compare algorithms using two forms of data:

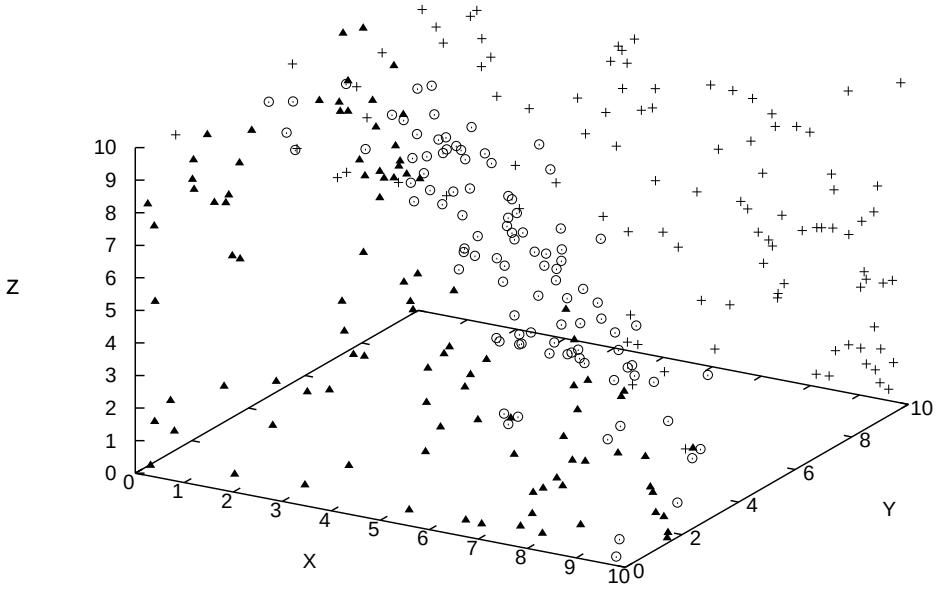
- Sampled from the linear (DTLZ1), spherical (DTLZ2), degenerate (DTLZ5), and discontinuous (DTLZ7) problems of the DTLZ test suite [35]. Properties of these test problems can be found in Deb et al. [35, 44]. For each front, a representative set of $10,000$ points from the known Pareto optimal set was generated mathematically. This set was then randomly sampled to form a front of the required size.

- Randomly-generated fronts, initialised by generating points with random values $0.1 \leq x \leq 10$ in all objectives. Mutual non-domination is guaranteed by initialising $S = \phi$ and adding each point $\overline{x}$ to S only if $\overline{x} \cup S$ is mutually-non-dominating.

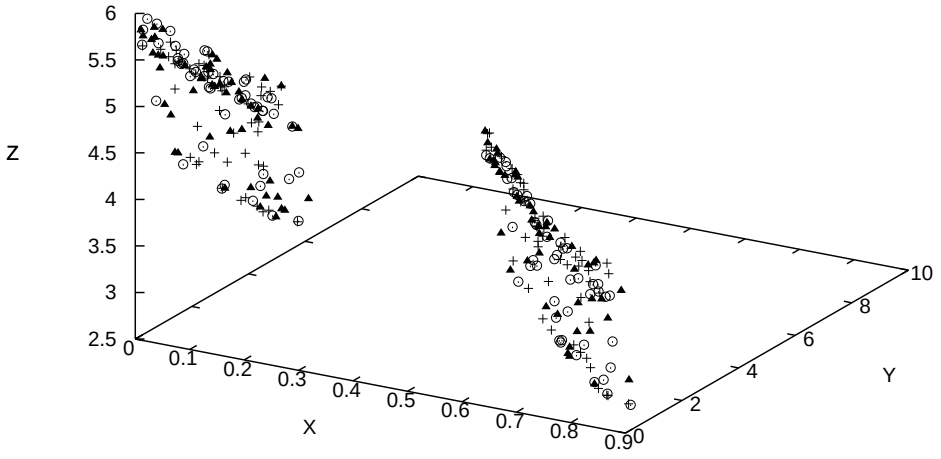
The 3D shape of these fronts is demonstrated in Figure 9. Experiments reported in this thesis were undertaken on various front sizes and dimensionalities. The test data used in experiments described in the published papers is available online [62].

Randomly generated fronts do not necessarily provide good approximations of real world problems. For example, as is evident Figure 9, these fronts exhibit knees near the optimal point that are atypical for most problems. As a result, random fronts are not ideal for testing hypervolume based selection techniques where selection quality is the primary consideration.

However, random fronts can be argued to have reasonable properties when used to measure hypervolume calculation performance. Random fronts have hypervolumes that distribute fairly uniformly. Furthermore, in most hypervolume calculation algorithms, performance varies greatly depending on the domination characteristics of points in less than n objectives and not the shape of the front. The number of hypervolume calculations required isn't dependent on front shape, all other things being equal. For example, in the FPL and HSO algorithms, dominated points are

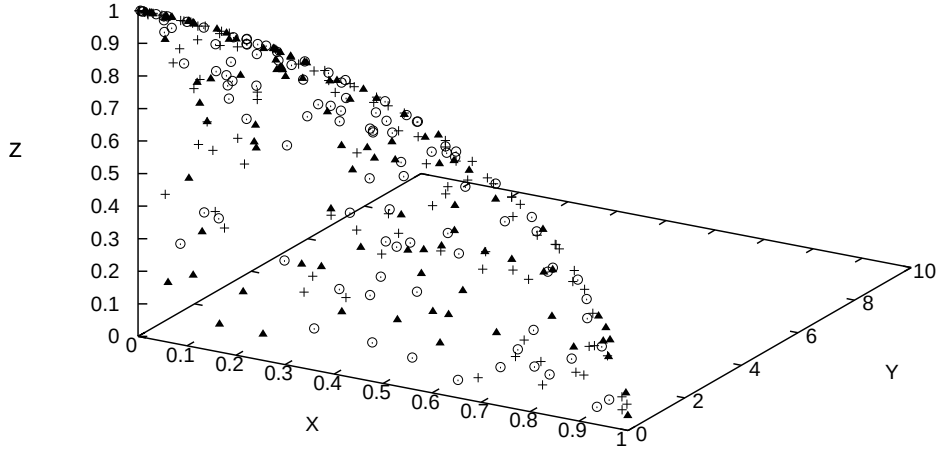


(a) Random fronts.

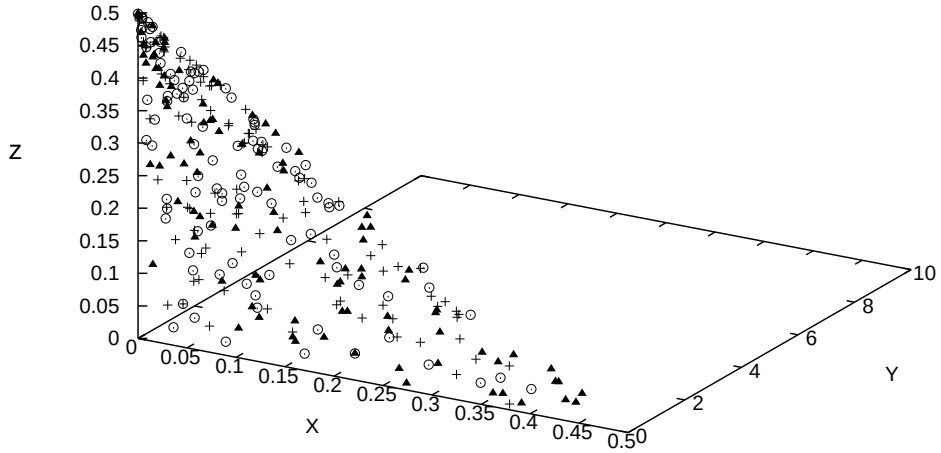


(b) Discontinuous fronts.

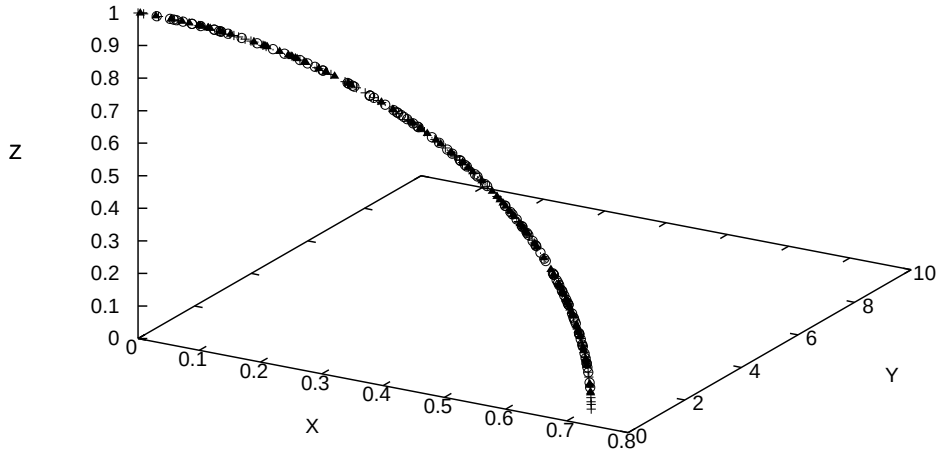
Figure 9: Three distinct 3D sample fronts for the five different data types used in the experimental comparisons of this dissertation.



(c) Spherical fronts.



(d) Linear fronts.



(e) Degenerate fronts.

Figure 9: Three distinct 3D sample fronts for the five different data types used in the experimental comparisons of this dissertation. (continued)

ignored or deleted, decreasing the number of points in lower dimensional slices, thus improving the performance on these slices. For a similar reason, in HOY the region splitting process may encounter covered or trellis regions more or less often.

To demonstrate the domination characteristics of the test data sets in this dissertation, the number of points dominated when various objectives are removed from fronts have been calculated. Twenty unique fronts are tested on all possible combinations of removed objectives. Table 1 shows the average minimum, average, and average maximum number of points dominated when 1 to 4 objectives are randomly removed from fronts containing data with six objectives (resulting in 2D to 5D data).

Table 1: Domination statistics: fronts in 6D, 1000 points, 20 unique fronts, all enumerated removed objectives.

		% points dominated		
data type	# obj removed	average min	average	average max
spherical	1	7.7	58.9	98.8
	2	59.8	88.7	99.6
	3	92.4	97.6	99.9
	4	98.6	99.4	99.9
linear	1	7.4	54.3	97.9
	2	61.1	87.3	99.2
	3	92.6	97.5	99.6
	4	98.8	99.4	99.9
degenerate	1	0.0	16.6	99.9
	2	0.0	33.3	99.9
	3	0.0	49.9	99.9
	4	0.0	66.6	99.9
discontinuous	1	41.1	50.7	79.1
	2	75.8	82.8	93.4
	3	92.9	95.4	98.4
	4	98.6	99.0	99.5
random	1	49.2	59.6	68.9
	2	83.8	86.3	89.8
	3	94.9	96.1	97.5
	4	98.6	99.1	99.5

Note that over 97% of points in the DTLZ spherical, linear, and degenerate fronts are dominated when a single best objective is removed. Conversely, certain other objectives lead to very few dominated points in these sets. The number of points dominated after removal of particular objectives for these sets can lead to very different performance characteristics for algorithms that filter dominated points. In contrast, random and discontinuous fronts vary less between the removal of objectives and thus there is less to gain from exploiting these domination characteristics.

Exploiting the characteristics of fronts similar to the linear, spherical, and degenerate fronts is clearly desirable. However, it is important to distinguish between performance improvements that result from an algorithm exploiting properties inherent to the data, and those that result from techniques likely to benefit calculation performance in general, including difficult or unseen problem types. *Paper 6* and Table 1 indicate that discontinuous and random data may provide a better gauge of complexity of a hypervolume calculation algorithm, as the scope for exploiting these data-specific characteristics is smaller.

Benchmark test data for hypervolume based selection algorithms require a different set of considerations. For example, the discontinuity found in discontinuous data is largely irrelevant for hypervolume calculation. The same number and type of operations are needed regardless of the discontinuity. However, note that the discontinuous shape does influence the subset selection that maximises hypervolume. Therefore, good benchmark data for hypervolume calculation performance may not necessarily test important attributes of hypervolume based selection techniques, and vice versa. This is especially relevant to the random data sets which may include characteristics atypical for real world problems.

3.2 Objective Ordering Heuristics for HSO

Although the HSO algorithm [67] exhibits better performance than its predecessors (particularly LebMeasure), its performance quickly becomes infeasible as a result

of its exponential time complexity in the number of objectives. *Paper 1*, shows that the performance of HSO can be improved greatly by varying the order of the processed objectives (*Paper 6* includes experiments demonstrating performance variation of hypervolume algorithms enumerated over all ordering permutations). *Paper 1* improves objective ordering with two heuristics designed for HSO.

The first heuristic, Maximising Dominated Points (MDP), finds an objective ordering which maximises the number of points dominated after the removal of the first objective. This results in smaller lower dimensional slices. The second heuristic, Minimising Worst-case Work (MWW), aims to minimise the cost of calculating hypervolume by estimating the cost of calculating each of the top-most slices for an objective processing order. Using HSO's worst case complexity, an estimate of each top-most slice's cost is accumulated for a given ordering, and the ordering with the smallest total cost is chosen.

As shown in *Paper 1*, these heuristics substantially improve performance on benchmark data. On discontinuous DTLZ and random test data, a reduction in running time of around 25–50% is observed. While for linear and spherical DTLZ test data, an improvement of 90–98% is observed, due to the domination characteristics of these fronts. Indeed, Section 3.1 shows that a good objective ordering will result in up to 99% of the points in the spherical and linear fronts being dominated after the first objective is removed (See Table 1). The results shown in *Paper 1* imply that in most cases the heuristic is able to determine a valuable objective ordering that increases the number of dominated points. The resulting reduction in runtime leads to a modest to large improvement in the feasibility of HSO, particularly for data sets similar to spherical or linear.

3.3 Hypervolume Based Selection Techniques

The heuristics created in *Paper 1* were designed to be applied in HSO, a hypervolume algorithm intended for use as a performance assessment indicator. Although

hypervolume has been extensively utilised for this purpose, at the time of this research, little attention had been given to its use as a selection measure within the optimisation process. One prominent issue for when hypervolume is used for selection, or in an online archiving scheme, is the requirement to reduce the size of the solution set to a bounded size [39].

Finding the optimal subset of points is a major obstacle and requires many subset combinations to be evaluated, and as hypervolume calculations are already expensive, finding the optimal subset is usually infeasible for all but the smallest fronts in few objectives. By 2005, few solutions to this issue had been investigated. One method, found in SMS-EMOA [36, 54], uses a steady state approach in which a single solution is optimised in every generation; eliminating the need to reduce the front or archive by more than one solution and thus avoiding an increase in selection combinations.

Paper 2 investigates two non-steady state selection techniques for this purpose. The first is a front selection technique that uses a local search approach to evaluate possible front selections and find a good local optima selection. This scheme initially generates a random front subset and evaluates its hypervolume. The chosen front is then perturbed slightly and if the new subset has a better hypervolume it becomes the new choice. This process is continued until a time limit is reached.

The second approach consists of a simple greedy selection scheme which is often used by indicator-based algorithms including IBEA [73], SIBEA [71], and archive schemes [39], and is the simplest non-steady state basis for use of the hypervolume indicator within an MOEA. Note that hypervolume is only used as a binary indicator in IBEA (i.e. hypervolumes are calculated for a single point relative to another). The greedy front reduction method begins with a full front and then iteratively removes the point that contributes the least to the overall hypervolume of the front, until the front is the desired size.

Each of the above techniques was evaluated using a range of numbers of objectives, front sizes, and data types. Each of these algorithms was implemented using HSO

with the MWW heuristic from *Paper 1*.

As shown in *Paper 2*, the local search scheme finds front selections with as good or better hypervolumes than the greedy reduction scheme when 80% of the front is removed. It is able to do so in one second and when given the substantially longer time of the greedy scheme. When 50% of the front is removed, the local search tends to perform better than the greedy scheme, finding as good or better selections than the greedy reduction scheme, for spherical and discontinuous fronts in under a second. However, local search does not always find an equivalent selection for random fronts.

In contrast, the greedy reduction technique performs well in cases where a majority of the front is retained. In this situation, the local selection technique is not effective, as local search requires numerous evaluations to discover good front selections. However, as hypervolume calculations become increasingly expensive as the front size increases, local search tends to perform worse than the greedy reduction approach, as it is performing more evaluations on similar size fronts.

Conversely, the local selection technique works especially well when a small proportion of the front is retained. In this instance, hypervolume calculations for the local search technique are comparatively cheap as the front sizes are small. In comparison, the greedy technique requires many greedy solution removals in this case, and they are much more expensive than those made by the local search.

3.4 Finding the Least Contributing Point

After investigating the greedy front reduction approach in *Paper 2*, it is clear that finding the least contributing point is an expensive operation when calculated using a metric algorithm such as HSO [67]. Unfortunately, finding the least contributing point is an important problem for the greedy selection approach and related approaches [36, 39, 71]. Under the greedy selection approach used in *Paper 2*, the least contributing point is determined by calculating the hypervolume of the front

with each contributing point removed, and this is performed for each point. Using this method, the front with the largest hypervolume is equivalent to removing the point with the least individual contribution. However, the basic HSO operation is very expensive to perform as many slices are unnecessarily calculated.

In *Paper 3* introduces an algorithm, Incremental HSO (IHSO), that reduces the computation required to calculate the contribution exclusively dominated by a single point. IHSO, while similar to HSO in form, benefits from several customisations that improve its performance for this purpose. Firstly, it disregards slices which do not contain the contributing point (i.e. slices above it), as they do not contribute to the exclusive hypervolume of the point. Secondly, it disregards slices where the contributing point is dominated in the current number of objectives, as again, these slices do not contribute to the point's exclusive hypervolume. Figure 10 shows the operation of one step in IHSO, demonstrating the slicing of the hypervolume, the allocation of points to each slice, the elimination of newly-dominated points, and the disregarding of both a slice above p and a slice below p.

Additionally, in *Paper 3* introduces objective reordering heuristics similar to those in *Paper 1* which improve the performance of IHSO. Unlike MWW and MDP of *Paper 1*, these new heuristics examine the contributing point in relation to just the front to which it contributes. The *rank* heuristic orders objectives by order of the rank of the contributing point among the fronts. The *dominated* heuristic works by finding the point Q which dominates the contributing point P in the most objectives, and arranges the objectives in which P is better than Q first. The rank heuristic was found to perform best on the tested data.

Since IHSO is intended for use in selection schemes, it is inefficient to calculate the contribution of every point and then select the minimum. However, to find the least contributing point, the selection algorithm need only calculate enough of every point as is needed to determine that it has a greater contribution than the least contributing point. An intelligent Best First Search (BFS) technique using a priority queuing scheme was devised to improve the performance of IHSO when

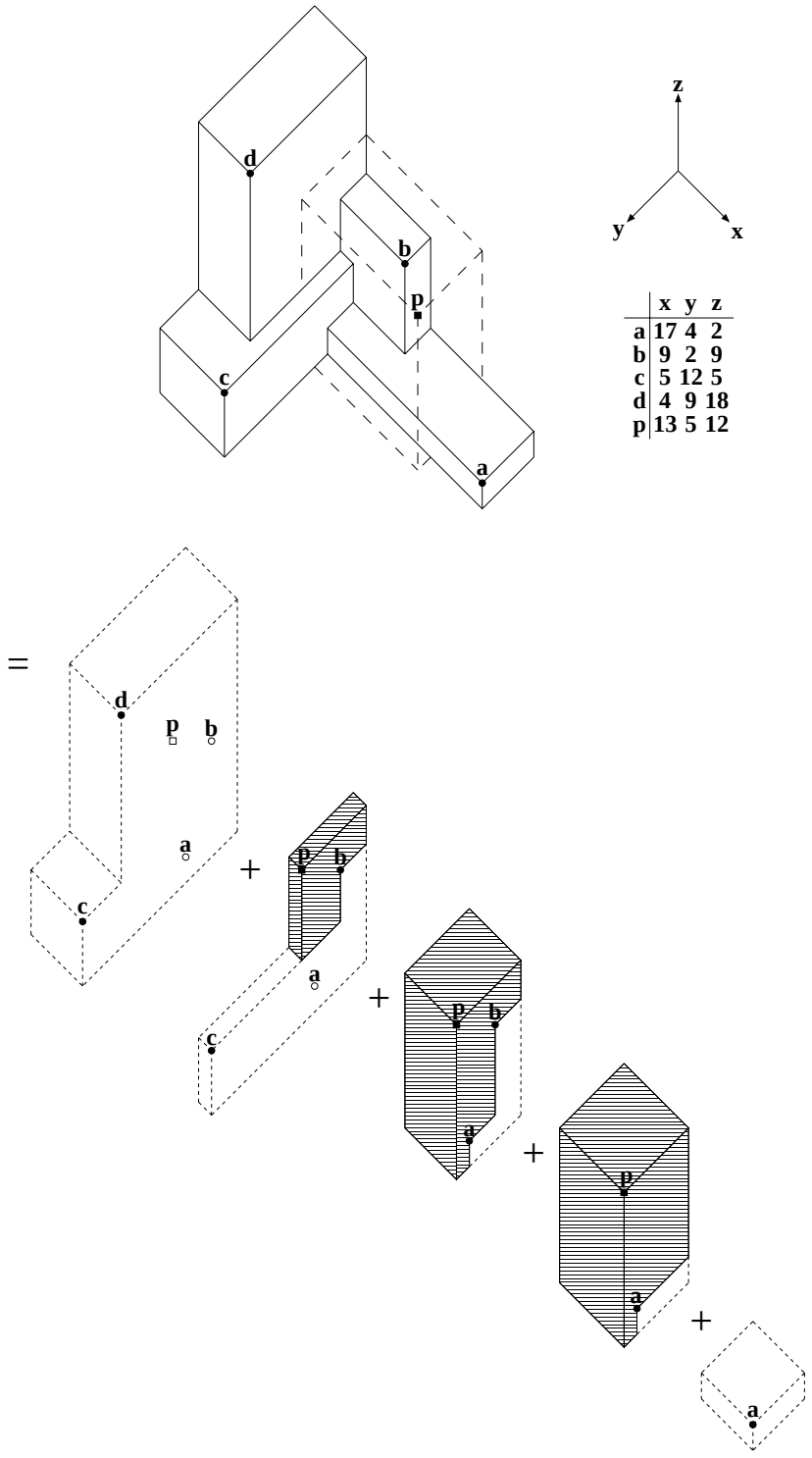


Figure 10: One step in IHSO for calculating the exclusive hypervolume of p relative to $\{a, b, c, d\}$, i.e. $ExcHyp(p, \{a, b, c, d\})$. Objective x is processed, leaving three non-empty slices (and two empty slices) in y and z. The volume to be calculated is indicated by the dashed box on the top diagram and by the shading on the slices on the bottom diagram. Filled marks indicate non-dominated points, and unfilled marks indicate points which are dominated in y and z by one of $\{a, b, c, d\}$. Figure and caption reproduced from *Paper 3*.

used for this purpose and is also described in *Paper 3*.

IHSO, when combined with the BFS technique, is able to find a front's least contributing point very quickly and much more efficiently than the naive approach using HSO and the MWW heuristic of *Paper 2*. For example, it takes 79 seconds to calculate the contribution of a point for the discontinuous data-set in 100 points in 7D using the naive approach, while in comparison, it takes 0.084 seconds using IHSO with BFS. The spherical data-set (4.68s for HSO with MWW versus 0.353s for IHSO with BFS) and randomly generated data (58.59s for HSO with MWW versus 0.006s for IHSO with BFS) perform similarly. These results point to a great improvement in the feasibility of hypervolume for online use.

n	random	discontinuous	spherical
5	955	750	700
6	950	280	240
7	940	170	92
8	880	105	47
9	830	75	32
10	490	58	28
11	220	44	24
12	70	36	20
13	42	28	16

Table 2: Front sizes in various numbers of objectives that IHSO with a BFS can process in one second. Table reproduced from *Paper 3*.

Table 2 shows the maximum front sizes that IHSO can process in under a second for each front-type. The IHSO algorithm, IHSO heuristics, and the BFS approach each significantly improve performance when removing points from fronts or archives. When combined, these techniques allow MOEAs to use hypervolume inline on much larger, more complex fronts, in more objectives.

3.5 Local and Greedy Selection Techniques Using IHSO

In *Paper 2*, two methods are introduced that can be used to select a subset of points from a front or archive during the optimisation process. With the introduction of the IHSO algorithm in *Paper 3*, it is important to reevaluate these front selection techniques after integrating IHSO, and this is covered in *Paper 4*.

The results presented in *Paper 2* show that the local search technique benefits from being able to use IHSO to update hypervolumes when points are removed or added. When a front is perturbed by adding or removing points, IHSO can calculate the resulting change and update the hypervolume without completely recalculating the entire front's hypervolume.

The greedy front reduction technique from *Paper 2* is hugely improved by the IHSO algorithm and BFS. The least contributing point can be found much faster using IHSO than using the prior naive approach, as IHSO is much faster than calculating full front hypervolumes. Additionally, the BFS scheme greatly improves the performance of this operation.

In addition, a new greedy scheme, using IHSO, which instead adds points into an initially empty set was examined. Rather than iteratively removing the least contributing point, as in the greedy reduction scheme, the greedy addition scheme starts with an empty front and iteratively adds the point that contributes the greatest hypervolume to the front. Unfortunately, unlike the greedy reduction scheme, the greedy addition scheme cannot benefit from the use of a BFS, as the maximum contribution can only be known when all contributions are calculated in full. In greedy addition, contributions are initially calculated quickly as they are calculated relative to a small front, but as the front becomes bigger, these contributions become more expensive to calculate, and therefore it is well suited for selecting small proportions of the original front.

As noted in *Paper 2*, the local search scheme performs very well compared to the greedy reduction approach when a small proportion of the front is retained. However, as shown in *Paper 4*, the introduction of IHSO and the BFS scheme led to the greedy schemes outperforming the local search scheme, with local search being outperformed by one of the greedy schemes in all the tested cases. Additionally, the algorithms and techniques introduced in *Paper 3* allow the greedy schemes using IHSO to easily outperform the greedy schemes from *Paper 2* implemented using HSO.

3.6 Updating Hypervolume Contributions

MOEAs using exclusive hypervolume contributions as a part of a selection scheme, as described in *Papers 2* and *4*, have a major inefficiency: the addition or removal of points to front subsets may lead to a change in the exclusive contribution of points. When these contributions are the basis for a selection measure, it is thus necessary to recalculate these contributions in order to maximise the accuracy of the selection measure.

As hypervolume calculations can be very expensive, recalculating previously calculated contributions should be minimised. *Paper 5* introduces a better technique to calculate the difference between an exclusive contribution before and after the addition of a point more quickly than recalculating the full contribution.

The Δ contribution method calculates the difference between a contributing point P to a front S, after the addition or subtraction of an independent point R from S. This is achieved by creating a new point PR that contains the least objective values from P and R in each objective. The contribution of PR is then calculated relative to $S - \{P, R\}$. PR's contribution is then added to P's contribution to reflect the removal of point R, or subtracted to reflect the addition of point R.

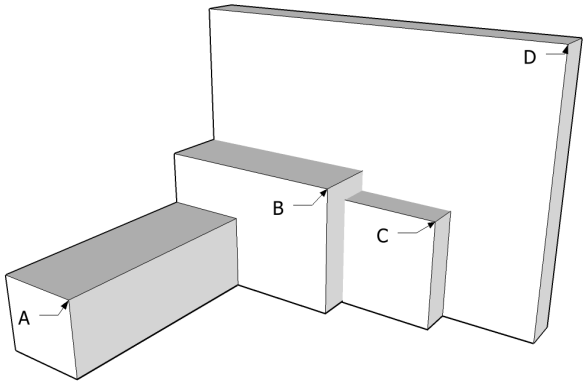
Figures 11(a)–11(c) show an example contribution calculated using this scheme. In this case, point C is discarded, increasing the exclusive contribution of point B. In

order to update B, B is combined with C to form a temporary point BC with the worst objective values from each. BC's exclusive contribution is then calculated and added to B's contribution, resulting in the correct exclusive hypervolume of B.

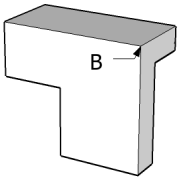
As shown in *Paper 5*, the use of this technique, when compared to full recalculation, yields a reduction in calculation time of 75%–99% over a range of data types, number of objectives, and front sizes. Results also show that the reduction in run-time increases as the number of objectives and front size increases. Since the recalculation cost is far greater than the cost of updating the contributions using this new technique, the savings increase as the front size increases. Further, as the number of objectives increases, the cost of calculating contributions increases exponentially. Effectively, the update scheme operates on much smaller slices than the full calculation approach because the update calculation uses a point worse than both P and R. As a result, its time performance grows much slower than full calculation as the number of objectives increases. This approach should thus improve the performance of many techniques which use hypervolume contributions as part of a selection process.

3.7 Hypervolume via Iterated Incremental Calculations

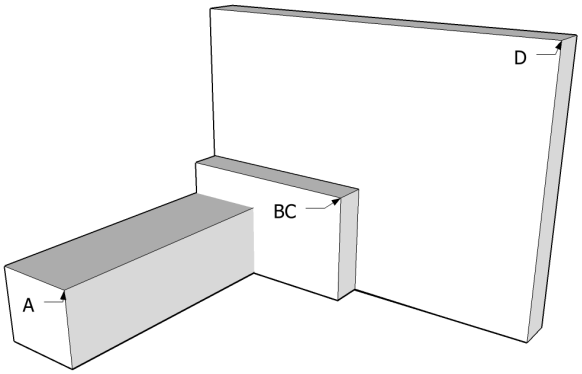
Section 2.4.2 describes the LebMeasure algorithm which calculates hypervolumes by adding a series of exclusive contribution calculations. *Paper 6* outlines an algorithm that uses the IHSO algorithm from *Paper 1* to build hypervolumes in a similar manner. Starting with an empty front, the Iterated IHSO algorithm (IIHSO) successively adds points to the front, accumulating an overall front hypervolume. Points are initially sorted in the first objective, chosen using the MWW heuristic of *Paper 1*. IHSO calculations use the rank heuristic used in *Paper 3*. Additionally, the IHSO algorithm was improved by the use of a linked list data-structure



(a) Example front containing points $A = (11, 2, 2)$, $B = (5, 4, 4)$, $C = (4, 6, 3)$, $D = (2, 8, 8)$.



(b) Exclusive contribution of B for the front shown in 11(a).



(c) Front used to calculate the contribution of BC $(4, 4, 3)$. The contributing volume of BC reflects the change in B's contribution resulting from the removal of point C.

Figure 11: Updating hypervolume contributions using the Δ technique. Figures reproduced from *Paper 5*.

introduced in the FPL algorithm [40] which is designed to minimise point sorting.

Paper 6 provides a useful comparison between the leading hypervolume calculation algorithms and IIHSO. This comparison includes a basic analysis of the potential for performance improvements using objective-reordering heuristics with existing algorithms, and a performance comparison between HOY, FPL, and IIHSO, which includes time performance and curve-fitting complexity coefficients.

The results of the objective-reordering analysis in *Paper 6*, Table 1, shows that HOY's performance varies little throughout different objective orderings, and thus an objective ordering heuristic is probably not likely to be productive for it, as the time difference between the worst case ordering and the best case is small. However, being a dimension sweeping algorithm, FPL benefits similarly to HSO and therefore *Paper 6* includes an adaptation of MWW, using FPL's worst-case time complexity, that improves its performance significantly.

HOY, despite its much better worst-case complexity, performs poorly on the tested experimental data. IIHSO performs well, generally outperforming FPL in higher numbers of objectives.

3.8 Metric Calculations Using WFG

The excellent performance of IIHSO in *Paper 6* on higher dimensional data demonstrates that calculating front hypervolumes using repeated contribution calculations is a valuable approach. A recently introduced simple bounding technique independently discovered by Bringmann and Friedrich [21] and Bradstreet et al. [18], allows the calculation of exclusive hypervolume contributions using metric hypervolume algorithms. This technique operates by limiting point values for a front to the region bounded by the contributing point, subtracting the hypervolume of the resulting front from the hypervolume dominated by the contributing point alone (see Figure 12). One benefit of this technique is metric hypervolume calculation algorithms can be used for exclusive hypervolume calculations without adaptation.

As demonstrated by IIHSO, metric volumes can be quickly calculated via iterated contribution calculations. Furthermore, contributions can be calculated using metric calculations using the bounding technique introduced above. By combining these techniques, the process of calculating both metric hypervolumes and exclusive contributions can be performed through the mutual application of each technique (i.e. calculate hypervolume by iteratively adding contributions calculated using the bounding technique above, which itself applies the iterative contribution method). *Paper 7* introduces a new algorithm, WFG, that calculates hypervolume using this approach.

Experiments show WFG performs substantially better than all existing exact calculation algorithms. Table 3 shows the size of fronts that WFG can calculate in ten seconds on average compared to the size of fronts that IIHSO can calculate in the same timeframe — a very significant improvement. As IIHSO generally outperforms existing algorithms, WFG is a large improvement over leading hypervolume calculation algorithms.

Table 3: Front sizes that WFG and IIHSO can process in ten seconds.

n	Random		Discontinuous		Spherical	
	IIHSO	WFG	IIHSO	WFG	IIHSO	WFG
3	$>$ 10,000	$>$ 10,000	$>$ 10,000	$>$ 10,000	$>$ 10,000	$>$ 10,000
4	$>$ 10,000	$>$ 10,000	$>$ 10,000	$>$ 10,000	9,600	$>$ 10,000
5	3,500	$>$ 10,000	4,600	9,600	5,500	7,800
6	900	5,800	900	5,600	1,700	4,400
7	510	2,300	410	2,000	500	2,100
8	190	900	145	600	240	1,300
9	115	450	80	300	135	1,000
10	70	235	50	150	85	700
11	50	185	40	105	65	575
12	42	130	33	105	51	475
13	34	100	26	70	34	450

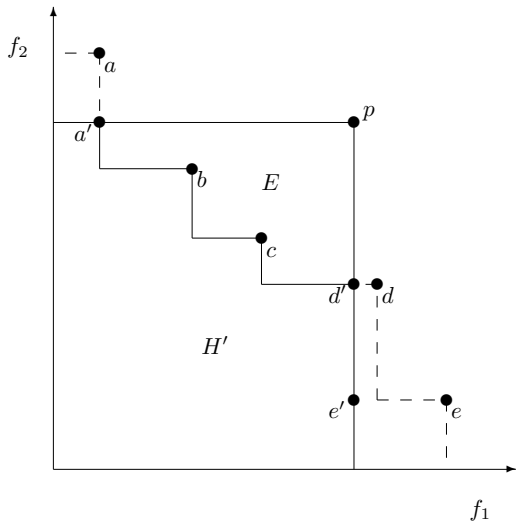


Figure 12: Maximising in both objectives, the exclusive hypervolume of p relative to a, b, c, d, e (i.e. E) = the inclusive hypervolume of p (i.e. the rectangle with p at the top-right corner) minus the hypervolume of a', b, c, d', e' (i.e. H'). Clearly e' is dominated by d' and can be discarded. Figure and caption reproduced from *Paper 7*.

Chapter 4

Included Papers

This submission is composed of seven papers co-authored by the PhD candidate. The included papers are the result of research ongoing from 2005. Five are conference papers accepted at the IEEE Congress on Evolutionary Computation (CEC) over various years. CEC is very competitive, with an acceptance rate of around 50%, and rated in the top tier of computer science conferences by the Computing Research and Education Association of Australasia (CORE). One paper was published in the IEEE Transactions on Evolutionary Computation (TEC), one of the top ranked journals in the Artificial Intelligence (AI) discipline. The final paper has been accepted for publication in TEC.

4.1 Summary of Papers

The seven papers included in this section discuss areas related to hypervolume and are presented in an order that preserves the logical progress of the research, not necessarily in order of publication.

Paper 1 Introduces heuristics that improve the typical performance of the HSO algorithm.

Paper 2 Discusses issues that arise when using hypervolume for selection and introduces a local search based selection scheme which is compared to a greedy selection scheme.

Paper 3 Introduces a new incremental hypervolume algorithm, IHSO, that calculates the exclusive contribution of a point to a set, a search technique to find the least contributing point with minimal computation, and heuristics for IHSO.

Paper 4 Integrates IHSO research from *Paper 3* into the selection schemes used in *Paper 2*. Further introduces an additional greedy selection scheme and reevaluates and compares these schemes.

Paper 5 Introduces a new technique to reduce unnecessary recalculation of exclusive point contributions resulting from changes in front composition.

Paper 6 Uses IHSO to create a new metric hypervolume algorithm using iterated contribution calculations.

Paper 7 Introduces a new hypervolume algorithm, WFG, for metric and contribution calculations. WFG uses a new method of contribution calculations and performs extremely well.

4.2 Errata

Paper 3: There was an error in the pseudo-code for IHSO that might impede its implementation. The corrected pseudo-code can be found in the published correction that follows the paper.

Paper 5: The original paper included incorrect results for Table 4. The version in this thesis has been corrected.

Paper 1 (Refereed)

L. While, L. Bradstreet, L. Barone, and P. Hingston. Heuristics for Optimising the Calculation of Hypervolume for Multi-Objective Optimization Problems. In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, pages 2225–2232, Edinburgh, Scotland, September 2005. IEEE Press

Heuristics for Optimising the Calculation of Hypervolume for Multi-objective Optimisation Problems

Lyndon While, Lucas Bradstreet, Luigi Barone
The University of Western Australia
Nedlands, Western Australia 6009
{lyndon, lucas, luigi}@csse.uwa.edu.au

Phil Hingston
Edith Cowan University
Mount Lawley, Western Australia 6050
p.hingston@ecu.edu.au

Abstract- The fastest known algorithm for calculating the hypervolume of a set of solutions to a multi-objective optimisation problem is the HSO algorithm (Hypervolume by Slicing Objectives). However, the performance of HSO for a given front varies a lot depending on the order in which it processes the objectives in that front. We present and evaluate two alternative heuristics that each attempt to identify a good order for processing the objectives of a given front. We show that both heuristics make a substantial difference to the performance of HSO for randomly-generated and benchmark data in 5–9 objectives, and that they both enable HSO to reliably avoid the worst-case performance for those fronts. The enhanced HSO will enable the use of hypervolume with larger populations in more objectives.

1 Introduction

Multi-objective optimisation problems abound, and many evolutionary algorithms have been proposed to derive good solutions for such problems, e.g. [1, 2, 3, 4, 5]. However, the question of what metrics to use in comparing the performance of these algorithms remains difficult[6, 7, 1]. One metric that has been favoured by many people is *hypervolume*[8], also known as the S-metric[9] or the Lebesgue measure[10]. The hypervolume of a set of solutions measures the size of the portion of objective space that is dominated by those solutions collectively. Generally, hypervolume is favoured because it captures in a single scalar both the closeness of the solutions to the optimal set and, to some extent, the spread of the solutions across objective space. Hypervolume also has nicer mathematical properties than many other metrics[11, 12]. Hypervolume has some non-ideal properties too: it requires the (sometimes arbitrary) definition of a reference point on which its calculations are based, and it is sensitive to the relative scaling of the objectives, and to the presence or absence of extremal points in a front.

While *et al.*[13] have shown that the fastest known algorithm for calculating hypervolume exactly is HSO (Hypervolume by Slicing Objectives)[14, 15]. HSO works by processing the objectives in a front, rather than the points. It divides the nD-hypervolume to be measured into separate $n-1$D-slices through one of the objectives, then it calculates the hypervolume of each slice and sums these values to derive the total. In the worst case HSO is exponential in the number of objectives, but it still easily outperforms all other known algorithms for calculating hypervolume exactly[13, 16].

However, the performance of HSO for a given front depends on the order in which it processes the objectives in that front. The number of points contributing hypervolume to each $n-1$D-slice depends on how many points are dominated within that slice: more dominated points implies a smaller set of points to process, which implies less work for that slice. The principal contribution of this paper is the presentation and evaluation of two alternative heuristics that each enhance HSO by trying to select a good order in which to process the objectives for a given front. We present performance data for basic and enhanced HSO showing that both heuristics make a substantial difference to the typical performance of the algorithm. The enhanced HSO will enable the use of hypervolume with larger populations in more objectives.

The rest of this paper is structured as follows. Section 2 defines the concepts and notation used in multi-objective optimisation and throughout this paper. Section 3 describes the operation of HSO, and Section 4 discusses its complexity and performance and shows how heuristics might help. Section 5 defines our two (alternative) heuristics, and Section 6 gives empirical data for a range of randomly-generated and benchmark fronts in 5–9 objectives showing how the heuristics improve the performance of HSO. Section 7 concludes the paper and outlines some future work.

2 Fundamentals

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.

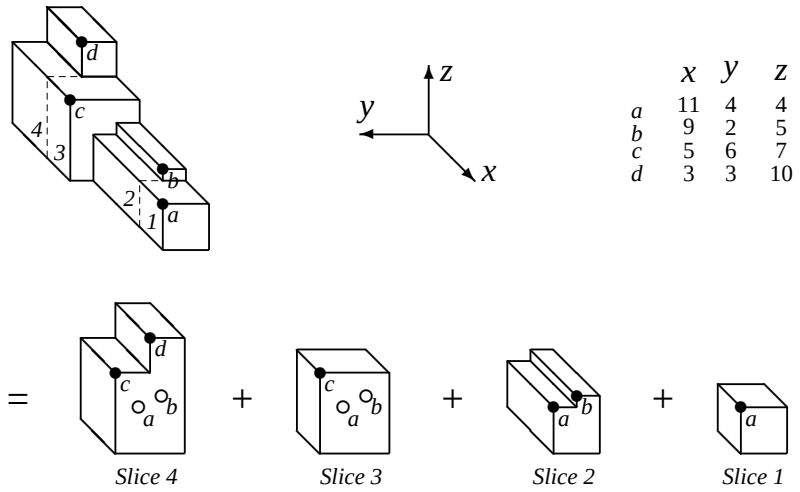


Figure 1: One step in HSO for the four three-objective points shown. Objective *x* is processed, leaving four two-objective shapes in *y* and *z*. Points are marked by circles and labelled with letters: unfilled circles represent points that are dominated in *y* and *z*. Slices are labelled with numbers, and are separated on the main picture by dashed lines.

Given a set X of solutions returned by an algorithm, the question arises how good the set X is, i.e. how well it approximates the Pareto optimal set. One metric used for comparing sets of solutions is to measure the *hypervolume* of each set. The hypervolume of X is the total size of the space that is dominated by the solutions in X. 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 fronts being compared.) If a set X has a greater hypervolume than a set X', then X is taken to be a better set of solutions than X'.

Precise definitions of these terms can be found in [17].

3 The HSO Algorithm

Given m mutually non-dominating points in n objectives, the HSO algorithm is based on the idea of processing the set of points *one objective at a time*.

Initially, the points are sorted by their values in the first objective to be processed. These values are then used to cut cross-sectional "slices" through the hypervolume: each slice will itself be an $n-1$-objective hypervolume in the remaining objectives. The $n-1$-objective hypervolume in each slice is calculated and each slice is multiplied by its depth in the first objective, then these n-objective values are summed to obtain the total hypervolume.

Each slice through the hypervolume will contain a different subset of the original points. Because the points are sorted, they can be allocated to the slices easily. The top slice can contain only the point with the best value in the first objective; the second slice can contain only the points with the two best values; the third slice can contain only the points with the three best values; and so on, until the bottom slice, which can contain all of the points. However, not all points "contained" by a slice will contribute volume to that slice: some points may be dominated in whatever objectives remain and will contribute nothing. After each step (i.e. after each slicing action), the number of objectives is reduced by one, the points are re-sorted in the next objective, and newly-dominated points within each slice are discarded.

Figure 1 shows the operation of one step in HSO, including the slicing of the hypervolume, the allocation of points to each slice, and the elimination of newly-dominated points.

The most natural base case for HSO is when the points are reduced to one objective, when there can be only one non-dominated point left in each slice. The value of this point is then the one-objective hypervolume of its slice. However, in practice, for efficiency reasons, HSO terminates when the points are reduced to two objectives, which is an easy and fast special case.

Figure 2 gives pseudo-code for HSO. Note that, for exposition purposes, the function `hso` builds explicitly a set containing the slices to be processed after each iteration. We can improve the performance of the algorithm by processing these slices on-the-fly, as they are generated.

4 The Complexity and Performance of HSO

While *et al.*[13] give a recurrence relation that captures the worst-case complexity of HSO:

$$f(m,1) = 1 \quad (1)$$

$$f(m,n) = \sum_{k=1}^{m} f(k,n-1) \quad (2)$$

```
hso (ps):
  pl = sort ps worsening in Objective 1
  s = {(1, pl)}
  for k = 1 to n-1
    s' = {}
    for each (x, ql) in s
      for each (x', ql') in slice (ql, k)
        add (x * x', ql') into s'
    s = s'
  vol = 0
  for each (x, ql) in s
    vol = vol + x * |head (ql)[n] - refPoint[n]|
  return vol

slice (pl, k):
  p = head (pl)
  pl = tail (pl)
  ql = []
  s = {}
  while pl /= []
    ql = insert (p, k+1, ql)
    p' = head (pl)
    add (|p[k] - p'[k]|, ql) into s
    p = p'
    pl = tail (pl)
  ql = insert (p, k+1, ql)
  add (|p[k] - refPoint[k]|, ql) into s
  return s

insert (p, k, pl):
  ql = []
  while pl /= [] && head (pl)[k] beats p[k]
    append head (pl) to ql
    pl = tail (pl)
  append p to ql
  while pl /= []
    if not (dominates (p, head (pl), k))
      append head (pl) to ql
    pl = tail (pl)
  return ql

dominates (p, q, k):
  d = True
  while d && k <= n
    d = not (q[k] beats p[k])
    k = k + 1
  return d
```

Figure 2: Pseudo-code for HSO.

The summation in (2) represents the fact that each slicing action generates m slices that are processed independently to derive the hypervolume of the front.

Furthermore, While *et al.*[13] solve this recurrence relation to give the following identity:

$$f(m,n) = \left(\begin{array}{c} m+n-2 \\ n-1 \end{array} \right) \tag{3}$$

Thus HSO is exponential in the number of objectives n, in the worst case (we assume that $m > n$).

The "worst case" in this context means we assume that no (partial) point is ever dominated during the execution of HSO, thus maximising the number of points in each slice that is processed. However, this is unlikely to be true for real-world fronts. The amount of time required to process a given front depends crucially on how many points are dominated at each stage, and, in addition, on how early in the process points dominate other points.

From this fact, we can infer that the time to process a given front varies with the order in which the objectives are processed. A simple example illustrates how. Consider the set of points in Figure 3, in a maximisation problem.

5	$\cdots$	5	1
4	$\cdots$	4	2
3	$\cdots$	3	3
2	$\cdots$	2	4
1	$\cdots$	1	5

Figure 3: A pathological example for HSO. This pattern describes sets of five points in n objectives, $n \geq 3$. All columns except the last are identical. The pattern can be generalised for other numbers of points.

If we process the first objective (or in fact any objective except the last)**:** no point dominates any other point in the list in the remaining $n-1$ objectives. Thus we do indeed have the worst case for HSO, generating m slices containing respectively $1, 2, \ldots, m$ points.

If we process the last objective: each point dominates all subsequent points in the list in the remaining $n-1$ objectives. Then we generate m slices *each containing only one point*. Specifically, the top slice (corresponding to the highest value in the last objective) contains only the point $1 \cdots 1$, the second slice contains only the point $2 \cdots 2$, all the way down to the bottom slice, which contains only the point $m \cdots m$. This is of course the best case for HSO, and the hypervolume is calculated much more quickly.

Note that, in general, there is a continuum of performance improvement available: e.g. for the points in Figure 3, the earlier the last objective is processed, the faster the hypervolume will be calculated.

Thus it seems that enhancing HSO with a mechanism to help the algorithm to identify a good order in which to process the objectives in a given front could make a substantial difference to the real performance of the algorithm.

5 Heuristics

We present and evaluate two alternative heuristics that attempt to derive a good order for HSO to process the objectives in a given front.

5.1 Maximising the number of dominated points

A good order for the objectives is one in which many partial points are dominated by other points early in the process. One obvious tactic then is to calculate for each objective how many points will be dominated immediately if that objective is processed, and to process first the objective that will generate the most dominated points. We call this heuristic MDP: Maximising the number of Dominated Points.

We can apply this idea in two ways.

- We can simply calculate the heuristic once, then sort the objectives in decreasing order of numbers of dominated points.
- Alternatively, we can calculate the heuristic once, eliminate the best objective, then re-calculate the heuristic to identify the next objective, and so on, until all the objectives have been ordered.

Our experience shows that applying the heuristic iteratively works better, especially for large numbers of objectives, but that diminishing returns apply to some extent. We therefore iterate until four objectives remain, at which point we order those four objectives according to the last calculation.

The complexity of MDP is easy to calculate: at each iteration, for each objective, we (nominally) compare each point with every other point for domination. Thus for m points in n objectives, each iteration of MDP has complexity $O(m^2n^2)$, and applying MDP iteratively has complexity $O(m^2n^3)$, While this may sound expensive, remember that HSO is exponential in n in the worst case, so a good polynomial-time heuristic is likely to pay large dividends.

5.2 Minimising the amount of worst-case work

For each objective, MDP effectively counts the number of points that will contribute to the bottom $n-1$D-slice of the hypervolume. However, in some cases, this number might be misleading: it is theoretically possible to generate m slices where the first $m-1$ slices contain respectively $1, 2, \ldots, m-1$ points, but the bottom slice contains only 1 point. Example data that exhibits this behaviour is given in Figure 4, for a maximisation problem.

5	1	4
4	2	3
3	3	2
2	4	1
1	5	5

Figure 4: A pathological example for MDP. MDP will choose to process the first objective, but processing the second objective would be faster.

We can avoid this possibility with a slightly more involved heuristic that calculates explicitly for each objective the number of non-dominated partial points in each slice, estimates the amount of work required to process each slice, and sums these values to estimate the amount of work required if HSO processes that objective first. This heuristic effectively models the recurrence relation in (2), by summing the work required to process each slice individually. For each slice, we use the worst-case complexity of HSO given in (3) to estimate the work required to process that slice. Thus we call this heuristic MWW: Minimising the Worst-case Work.

Again, we can apply this idea once only, or iteratively, and again, our experience shows that iteration works better. As with MDP, we apply MWW iteratively until four objectives remain, at which point we order those four objectives according to the last calculation.

The complexity of MWW is similar to that of MDP. For each objective, we sort the points in that objective, then we build incrementally the sets of points in each slice, much as in the functions `slice` and `insert` in Figure 2. This leads to the worst-case complexity for each iteration being $O(n(m \log m + m^2 n))$, which again simplifies to $O(m^2n^2)$. The need to maintain an explicit set of non-dominated points during the calculation of MWW may make it more expensive than MDP in some cases, although any difference is likely to be small.

6 Empirical Performance Data

We evaluated the performance of the two heuristics vs. basic HSO on two different types of data: randomly-generated fronts, and samples taken from the four distinct Pareto optimal fronts of the problems in the well-known DTLZ test suite[18].

We evaluated the heuristics (mostly) on data in 5–9 objectives, so to estimate the best-, average-, and worst-case timings for each front using basic HSO, we used the following procedure.

For $n \leq 5$: we evaluated all $n!$ permutations of the objectives.

For $n > 5$: we sampled the $n!$ permutations in two ways, and we combined all of the results in the calculations.

- We evaluated all $n(n-1)$ permutations of the first two objectives (with the remaining objectives randomised).
- Additionally, we evaluated 120 randomly-chosen permutations.

All timings were performed on a dedicated 2.8Ghz Pentium IV machine with 512Mb of RAM, running Red Hat Enterprise Linux 3.0. All algorithms were implemented in C and compiled with *gcc -O3*. All times include the costs of calculating the heuristics, where appropriate. The data used in the experiments are available at

`http://wfg.csse.uwa.edu.au/Hypervolume`

6.1 Benchmark data

We evaluated the heuristics on the four distinct fronts from the DTLZ test suite: the spherical front, the linear front, the discontinuous front, and the degenerate front. For each front, we generated mathematically a representative set of 10,000 points from the (known) Pareto optimal set: then to generate a front of size m, we sampled this set randomly. Each hypervolume was calculated as a minimisation problem in every objective, relative to the point $1 \cdots 1$.

Tables 1(a)–1(c) and Figures 5(a)–5(d) and 6 show the resulting comparisons. Each row of each table is based on runs with ten different fronts, and it gives the following data.

- For HSO:

  $wrst$ is the longest time for any run on any front.

  $awst$ is the average of the longest time for each front.

		basic HSO					HSO+MDP			HSO+MWW		
n	m	$wrst$	$awst$	$\boldsymbol{avrg}$	$abst$	$best$	$wrst$	$\boldsymbol{avrg}$	$best$	$wrst$	$\boldsymbol{avrg}$	$best$
5	800	71.45	61.39	**11.91**	1.18	1.02	1.64	**1.30**	1.13	1.67	**1.31**	1.13
6	230	84.68	68.79	**10.50**	0.43	0.33	1.33	**0.74**	0.34	0.67	**0.46**	0.34
7	110	81.45	73.18	**10.50**	0.25	0.16	4.44	**1.15**	0.17	0.50	**0.25**	0.13
8	65	74.07	65.43	**10.29**	0.21	0.15	4.21	**1.00**	0.14	0.25	**0.17**	0.11
9	45	66.68	56.61	**10.25**	0.19	0.06	0.80	**0.30**	0.07	0.25	**0.13**	0.05

(a) The spherical DTLZ front.

		basic HSO					HSO+MDP			HSO+MWW		
n	m	$wrst$	$awst$	$\boldsymbol{avrg}$	$abst$	$best$	$wrst$	$\boldsymbol{avrg}$	$best$	$wrst$	$\boldsymbol{avrg}$	$best$
5	800	65.16	52.31	**11.40**	1.06	0.91	1.37	**1.18**	1.03	1.37	**1.21**	1.04
6	220	65.74	55.58	**9.93**	0.41	0.26	1.67	**0.77**	0.40	0.80	**0.46**	0.28
7	110	83.76	73.60	**10.95**	0.23	0.15	2.19	**0.57**	0.16	0.35	**0.25**	0.15
8	65	74.81	69.93	**10.96**	0.25	0.09	1.32	**0.52**	0.11	0.50	**0.20**	0.08
9	45	66.59	59.00	**10.14**	0.22	0.10	1.26	**0.48**	0.08	0.39	**0.19**	0.06

(b) The linear DTLZ front.

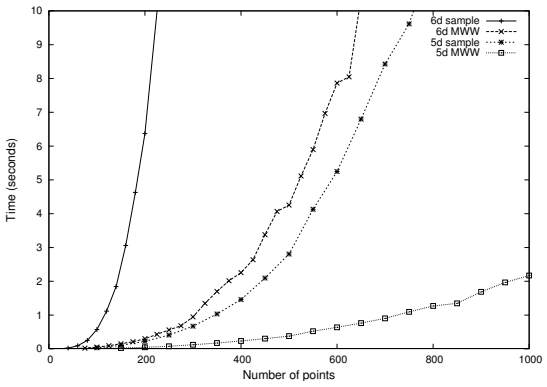
		basic HSO					HSO+MDP			HSO+MWW		
n	m	$wrst$	$awst$	$\boldsymbol{avrg}$	$abst$	$best$	$wrst$	$\boldsymbol{avrg}$	$best$	$wrst$	$\boldsymbol{avrg}$	$best$
5	1000	20.68	16.44	**9.90**	3.40	3.14	4.17	**3.78**	3.52	3.96	**3.66**	3.37
6	250	28.64	20.75	**10.71**	3.38	2.79	4.49	**3.88**	3.43	5.49	**4.12**	3.42
7	110	29.36	25.57	**13.00**	4.28	3.10	7.61	**5.77**	3.35	7.54	**5.16**	3.82
8	60	24.20	20.11	**10.29**	3.54	2.36	6.51	**4.67**	3.01	7.17	**4.40**	3.02
9	40	24.69	19.38	**9.84**	3.82	2.49	6.80	**5.00**	2.37	5.93	**4.38**	2.24

(c) The discontinuous DTLZ front.

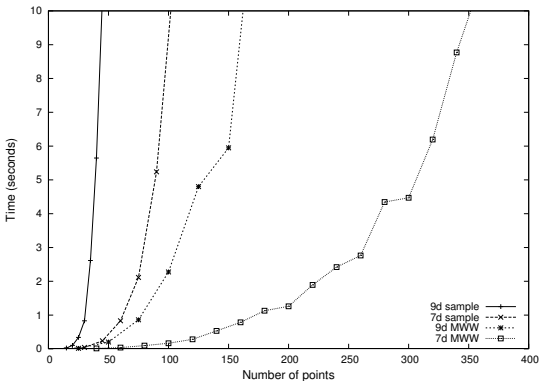
		basic HSO					HSO+MDP			HSO+MWW		
n	m	$wrst$	$awst$	$\boldsymbol{avrg}$	$abst$	$best$	$wrst$	$\boldsymbol{avrg}$	$best$	$wrst$	$\boldsymbol{avrg}$	$best$
5	1200	23.20	17.46	**11.39**	7.97	7.02	16.37	**9.87**	7.66	10.38	**8.32**	7.15
6	300	22.97	18.92	**11.16**	6.24	3.73	12.25	**9.11**	3.65	12.55	**7.03**	3.71
7	130	30.89	24.96	**13.74**	7.50	6.18	15.77	**10.47**	6.39	13.82	**8.17**	6.17
8	70	37.44	26.99	**12.39**	4.47	2.63	6.65	**5.31**	3.80	8.07	**5.26**	2.58
9	45	32.63	22.31	**9.73**	3.59	1.39	9.39	**4.76**	1.42	8.16	**4.12**	1.62

(d) Randomly-generated fronts.

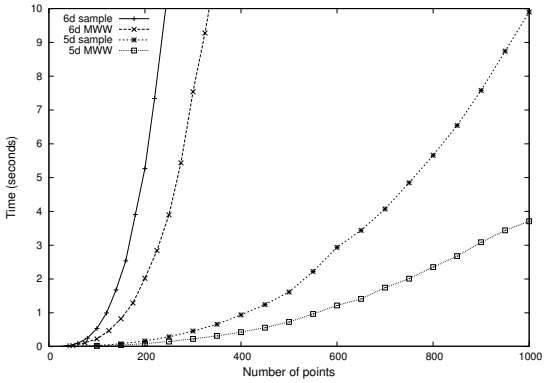
Table 1: Comparison of the performance of HSO, HSO+MDP, and HSO+MWW on various fronts. Each datum is based on ten different data sets: the figures for basic HSO are calculated using the sampling procedure described in Section 6. For each value of n, m is chosen so that the HSO $avrg \approx 10$s.



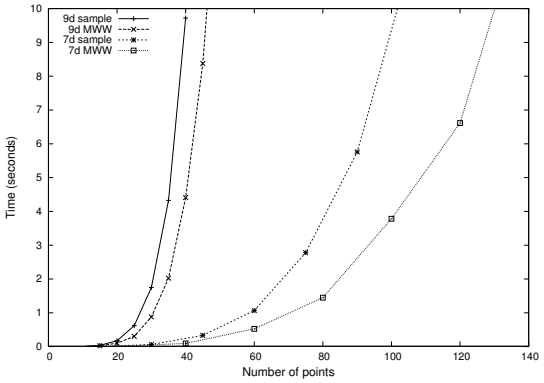
(a) The spherical DTLZ front in 5 and 6 objectives.



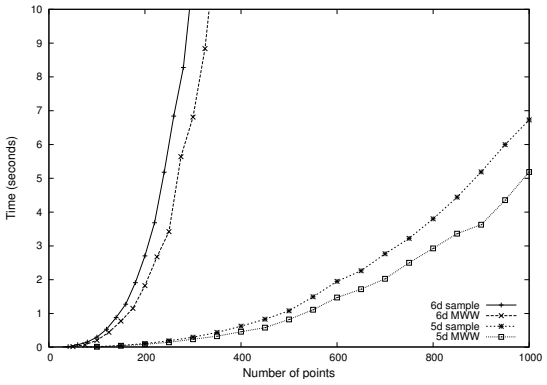
(b) The spherical DTLZ front in 7 and 9 objectives.



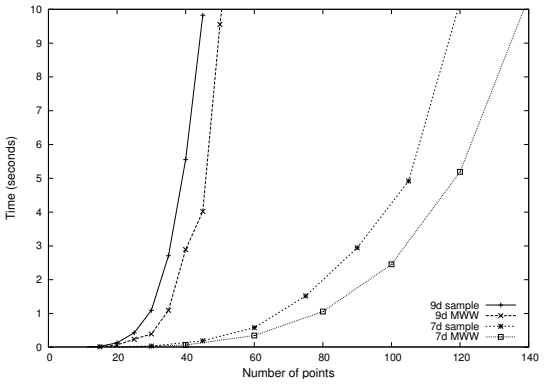
(c) The discontinuous DTLZ front in 5 and 6 objectives.



(d) The discontinuous DTLZ front in 7 and 9 objectives.



(e) Randomly-generated fronts in 5 and 6 objectives.



(f) Randomly-generated fronts in 7 and 9 objectives.

Figure 5: Comparison of the performance of HSO and HSO+MWW on various fronts. Each datum is based on ten different data sets: the figures for basic HSO are calculated using the sampling procedure described in Section 6. The plot for the linear DTLZ front is similar to that for the spherical front and is excluded for space reasons.

$avrg$ is the average time for all of the runs.

$abst$ is the average of the shortest time for each front.

$best$ is the shortest time for any run on any front.

- For each heuristic:

  $wrst$ is the longest time for any run on any front.

  $avrg$ is the average time for all of the runs.

  $best$ is the shortest time for any run on any front.

As observed previously by While *et al.*[13], the best-case objective order for the degenerate front gives performance that is polynomial in the number of objectives, so for that front, we plot only the performance of HSO+MWW. Each other plot compares the performance of HSO+MWW with the average performance of HSO over the sample of permutations of the objectives.

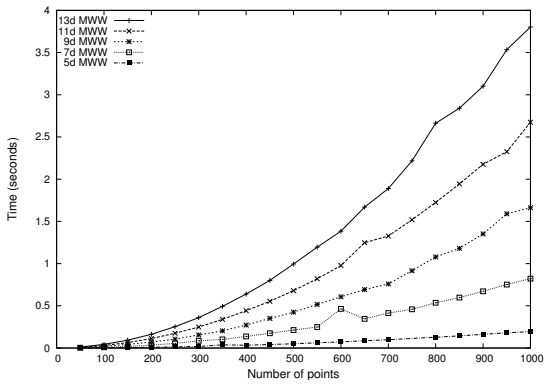


Figure 6: The performance of HSO+MWW on the degenerate front. Each datum is based on ten different data sets.

6.2 Randomly-generated data

We generated sets of m mutually non-dominating points in n objectives simply 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 kept adding points until $|S| = m$. Each hypervolume was calculated as a maximisation problem in every objective, relative to the origin.

Table 1(d) and Figures 5(e)–5(f) show the resulting comparison.

6.3 Discussion

For each heuristic in each row of each table, we make the following comparisons.

- We compare $avrg$ for the heuristic with the range $awst \ldots avrg \ldots abst$ for basic HSO, to determine how much improvement the heuristic delivers in typical cases.
- We compare $wrst$ for the heuristic with $wrst$ and $awst$ for basic HSO, to determine how well the heuristic avoids the worst-case ordering.
- We compare $best$ for the heuristic with $abst$ and $best$ for basic HSO, to determine how close the heuristic gets to the best-case ordering. (Note that the best cases for the heuristics sometimes beat the best case for basic HSO: this is due to the incomplete nature of the sampling used for the basic HSO figures.)

We make the following observations.

- For all of the DTLZ fronts, both heuristics deliver major performance gains, and MWW in particular delivers performance that is not far from optimal. The performance gains for the spherical and linear fronts in particular are spectacular: speed-up factors of 10–80 in the average cases. The performance gain for the discontinuous front is somewhat less (speed-up factors of 2–3): no doubt this is due to some property of the front itself.
- Random fronts may be the worst-case form of data for the heuristics, but both heuristics still always outperform basic HSO in the average case, with speed-up factors up to 2.5.
- Both heuristics avoid the worst-case objective ordering in all cases: in fact, the worst-case for the heuristics is nearly always better than the average case for basic HSO, usually by a substantial amount.
- The performance gain increases both with increasing number of objectives, and with increasing number of points.
- MWW generally out-performs MDP.
- The graphs however highlight the fact that exponential performance makes life tough: although the heuristics deliver useful speed-ups for processing fronts of a given size, they do not always greatly improve the sizes of fronts that can be processed in a given time.

The question arises what size of fronts the enhanced algorithm can process in various times. Table 2 shows this data for HSO+MWW on the spherical front. We chose ten seconds as indicative of the performance required to use hypervolume in off-line metric calculations after the EA is complete, and one second as indicative of the performance required to use hypervolume in an on-line diversity or archiving mechanism during the execution of the EA.

n	10 seconds	1 second
5	1,900	700
6	650	320
7	350	170
8	240	110
9	160	80
10	110	60
11	80	50
12	70	40
13	50	30

Table 2: Sizes of fronts in various numbers of objectives that HSO+MWW can process in the times indicated, for spherical DTLZ data.

We also performed some minor experimentation to estimate the cost of calculating the heuristics themselves. Our experiments indicate that these calculations usually take less than 1% of the run-time of the enhanced algorithm, and that they never exceed about 6% of the run-time, even with populations up to 2,000. This is of course to be expected, because of the exponential complexity of HSO itself.

7 Conclusions and Future Work

We have described two alternative heuristics that each improve the performance of the HSO algorithm for calculating hypervolume, itself the fastest algorithm described to date. Each heuristic works by re-ordering the objectives in a front to reduce the sizes of the sets of points that have to be processed during the execution of the algorithm. Both heuristics deliver significant improvement to the performance of HSO, with reductions in the run-time of the algorithm of 25–98%. The enhanced HSO will enable the use of hypervolume with larger populations in more objectives.

We intend to speed-up the calculation of our heuristics, e.g. by minimising the cost of dominance-checking, although we do not expect this to deliver serious further improvements. We also intend to pursue other avenues for making HSO faster, such as reducing the amount of repeated work that results from processing slices independently.

We also intend to design an incremental version of HSO, for use as a diversity or archiving mechanism in an evolutionary algorithm.

Acknowledgments

We thank Simon Huband for discussions on hypervolume and HSO, and for providing the raw DTLZ data.

This work was supported partly by The University of Western Australia Research Grants Scheme, and also partly by an ARC Linkage grant.

Bibliography

[1] S. Huband, P. Hingston, L. While, and L. Barone, "An evolution strategy with probabilistic mutation for multi-objective optimization," in *CEC 2003*, H. Abbass and B. Verma, Eds., vol. 4. IEEE, 2003, pp. 2284–2291.

[2] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *EUROGEN 2001*, K. C. Giannakoglou *et al.*, Ed., 2001, pp. 95–100.

[3] R. C. Purshouse and P. J. Fleming, "The MultiObjective Genetic Algorithm applied to benchmark problems — an analysis," The University of Sheffield, UK, Research Report 796, 2001.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[5] J. Knowles and D. Corne, "M-PAES: A memetic algorithm for multiobjective optimization," in *CEC 2000*, vol. 1. IEEE, 2000, pp. 325–332.

[6] T. Okabe, Y. Jin, and B. Sendhoff, "A critical survey of performance indices for multi-objective optimisation," in *CEC 2003*, H. Abbass and B. Verma, Eds., vol. 2. IEEE, 2003, pp. 878–885.

[7] 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.

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

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

[10] M. Laumanns, E. Zitzler, and L. Thiele, "A unified model for multi-objective evolutionary algorithms with elitism," in *CEC 2000*, vol. 1. IEEE, 2000, pp. 46–53.

[11] 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.

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

[13] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Transactions on Evolutionary Computation*, 2005.

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

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

[16] L. While, "A new analysis of the Lebmeasure algorithm for calculating hypervolume," in *EMO 2005*, ser. LNCS, C. Coello Coello *et al.*, Ed., vol. 3410. Springer-Verlag, 2005, pp. 326–340.

[17] T. Bäck, D. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. Iop Institute of Physics, 1997.

[18] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *CEC 2002*, D. B. Fogel *et al.*, Ed., vol. 1. IEEE, 2002, pp. 825–830.

Paper 2 (Refereed)

L. Bradstreet, L. Barone, and L. While. Maximising Hypervolume for Selection in Multi-objective Evolutionary Algorithms. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 6208–6215, Vancouver, BC, Canada, July 2006. IEEE Press

Maximising Hypervolume for Selection in Multi-objective Evolutionary Algorithms

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

***Abstract*— When hypervolume is used as part of the selection or archiving process in a multi-objective evolutionary algorithm, the basic requirement is to choose a subset of the solutions in a non-dominated front such that the hypervolume of the subset is maximised. We describe and evaluate two algorithms to approximate this process: a greedy algorithm that assesses and eliminates solutions individually, and a local search algorithm that assesses entire subsets. We present empirical data which suggests that a hybrid approach is needed to get the best trade-off between good results and computational cost.**

I. Introduction

Multi-objective problems are common in the optimisation field and much of the current research into evolutionary algorithms revolves around the theory and practice of multi-objective optimisation. Many evolutionary algorithms have been created in order to solve these difficult problems, e.g. SPEA[1], NSGA-II[2]. However, despite recent work, there remains the question of how to compare the performance of different multi-objective optimisation (MOO) algorithms. The result of a multi-objective algorithm is a set of solutions, or non-dominated front, each representing a trade-off between objectives. A front generated by one MOO algorithm is not easily comparable with a front generated by another algorithm. A popular metric used to compare these fronts is the hypervolume measure (otherwise known as S-metric[3] or the Lebesgue measure[4]). Hypervolume is the n-dimensional space that is "contained" by the points (solutions) in a front. A front with a larger hypervolume is likely to present a better set of trade-offs to a user than a front with a smaller hypervolume.

While most research into hypervolume has revolved around its use as a metric, recent research has seen it applied during the operation of Multi-objective Evolutionary Algorithms (MOEAs). An example of a technique used in this way is Deb's[2] NSGA-II crowdedness comparison operator, used to select solutions within a front to minimise solutions crowding in each objective. Knowles et al. [5] introduce the use of hypervolume during optimisation and describe a bounded archiving algorithm that maintains an archive to retain the 'best' solutions, or points, found throughout optimisation. As this archive is potentially of unbounded size, they apply hypervolume to determine the solutions that maximise the coverage of the solution set. During optimisation, when a new solution is added to this bounded archive, the solution that contributes the least hypervolume is removed in order to make room.

Emmerich et al. [6] extend this idea to selection in an MOEA in their optimiser SMS-EMOA. Rather than applying bounded archiving, during optimisation they apply hypervolume to remove the solution that contributes the least hypervolume from the worst ranked front. By doing so, they reduce the size of the population in order to provide room for the next generation.

While this idea has merit, and has achieved excellent results for tested experiments, it has some limitations. One such limitation is the use of a steady state MOEA. Emmerich et al. use a steady state MOEA because finding the optimal composition that maximises the hypervolume of a reduced front is a difficult problem and the effectiveness of heuristic approaches is unknown. However, it is possible in some cases, a steady state MOEA may not achieve as high quality results as other MOEAs.

The principal contribution of this paper is the presentation and evaluation of two algorithms to choose a subset of the solutions in a non-dominated front such that the hypervolume of the subset is maximised. To do this exactly is prohibitively expensive, so the two algorithms use different methods to approximate the process. The first is a greedy algorithm that eliminates solutions one-at-a-time based on their perceived exclusive contribution to the hypervolume of the front, and the second is a local search algorithm that assesses entire candidate subsets and tries to improve the current candidate by perturbing it. Empirical data suggests that each algorithm out-performs the other in the right circumstances, so a hybrid approach is suggested to get the best trade-off between good results and computational cost.

The rest of this paper is structured as follows. Section II defines the concepts and notation used in multi-objective optimisation. Section III defines the front reduction techniques that we have created and why we may use them. Section IV gives empirical data, for a variety of front types, demonstrating situations where each front reduction technique is valuable and why. Section V concludes the paper and outlines future work.

II. Fundamentals

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 $\bar{x}$ and $\bar{y}$, $\bar{x}$ *dominates* $\bar{y}$ iff $\bar{x}$ is at least as good as $\bar{y}$ in all objectives, and better in at least

Lucas Bradstreet, Luigi Barone, and Lyndon While are with the School of Computer Science & Software Engineering, The University of Western Australia, Crawley 6009, Australia (phone: +61864881944; fax: +61864881089; email: {lucas, luigi, lyndon}@csse.uwa.edu.au).

one. A vector $\bar{x}$ is *non-dominated* with respect to a set of solutions X iff there is no vector in X that dominates $\bar{x}$. 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 $\bar{x}$ is *Pareto optimal* iff $\bar{x}$ 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.

Given a set X of solutions returned by an algorithm, the question arises how good the set X is, i.e. how well it approximates the Pareto optimal set. One metric used for comparing sets of solutions is to measure the *hypervolume* of each set. The hypervolume of X is the total size of the space that is dominated by the solutions in X. 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 fronts being compared.) If a set X has a greater hypervolume than a set X, then X is taken to be a better set of solutions than X.

Knowles[7] applies hypervolume in order to recast multi-objective problems as single objective problems with the single goal of maximising the hypervolume of a set of solutions, bounded in size. As this set is of finite size, when adding a solution to this set another must often be removed to make room. This is achieved by removing the solution contributing the minimal 'exclusive hypervolume' to the front. The exclusive hypervolume contribution, Δs, of a solution, p, to a front, f, can be defined as $\Delta s = Hypervolume(f \cup \{p\}) - Hypervolume(f)$. Following this definition, one realises that over the course of operation the hypervolume of the set of archived solutions is maximised.

Emmerich et al.[6] apply this technique to selection in a MOEA. Rather than maintain an archive of solutions, as in Knowles[7], hypervolume is used to determine which solutions should be allowed to reproduce and while should be thrown away. When a front is too large to be included in the population during selection, hypervolume is used to reduce its size. The solution that contributes the smallest hypervolume to the worst ranked front is removed from the front with the aim of maximising the hypervolume of the population during the lifetime of the MOEA.

III. Point Removal Techniques

When using hypervolume as a selection metric, we wish to reduce the size of non-dominated fronts. Ideally, we wish to find a front composition that maximises the hypervolume of the reduced front. However, in order to guarantee an optimally composed front, it is necessary to calculate all $\binom{n}{m}$ subsets of the front, where n is the size of the front and m is the size of the reduced front. Calculating all possible subset combinations can be extremely expensive if even a small number of solutions are to be removed. In order to work around this problem, Emmerich et al. use a steady state MOEA, where only one point is removed at a time. However, use of steady state MOEAs may not always be desirable. As a result, alternative approaches to using hypervolume for selection should be researched.

We present two alternative techniques: a search algorithm and a greedy algorithm that removes one solution at a time until a front of the desired size is created. We believe that each of these algorithms is useful in different scenarios, and we thus aim to identify these cases so that an informed decision may be made when designing an MOEA using hypervolume for selection.

A. Greedy Front Reduction Algorithm

Given a non-dominated front, S, of size m in n objectives, removing p solutions.

```
i = 0
while time limit not exceeded and i < p
  calculate contribution of each pt in S
  remove smallest contributing pt from S
  increment i
if i < p
  remove p-i solutions from S using
  latest contributions calculated
```

This scheme, which we have generalised from Emmerich et al.[6]'s MOEA to reduce fronts by more than one solution, may be computationally expensive in cases in which a large number solutions are removed. In order to remove a single solution from a front containing m solutions, m hypervolume calculations are required. This process must be repeated for each solution to be removed. Thus, in order to remove p solutions from the front, $\sum_{i=1}^{p} m - 1 + 1$ hypervolume evaluations are required. Furthermore, for large fronts the cost of each hypervolume evaluation may be expensive due to the exponential complexity of current hypervolume algorithms, as proved by While [8]. The expensive nature of this algorithm required us to implement a shortcut, that allows the algorithm to stop at each iteration if a time constraint has been reached. In this case, the algorithm sorts the solutions by the exclusive hypervolume calculations from the previous iteration and composes the front using the best ranked solutions. This shortcut was made in order to present a viable alternative to the local search method.

To implement this algorithm, we apply the Hypervolume By Slicing Objectives (HSO) algorithm using the Minimising Worst-case Work (MWW) heuristic described by While et al[9]. The MWW heuristic reduces the run-time of HSO by 25-98% on the DTLZ[10] test suite fronts and randomly generated non-dominated fronts.

B. Front Reduction by Local Search

In contrast to the greedy front reduction method we propose a local search algorithm. A local search should

improve performance of point removal methods in situations where many solutions are eliminated from a front. Rather than perform $\sum_{i=1}^{p} m - 1 + 1$ expensive hypervolume evaluations, the local search technique performs a larger number of computationally cheaper hypervolume evaluations. Even though the local search requires more individual hypervolume evaluations, this method can be faster than the front reduction method used by Emmerich et al.

1) Benefits of Search for Front Composition: When reducing a front's size we wish to find the reduced front that maximises the hypervolume of the front. However, the use of a greedy front reduction algorithm does not guarantee the optimally composed reduced front will be found. Furthermore, calculating all possible reduced fronts is generally infeasible. Thus other optimisation algorithms, such as a local search or evolutionary algorithm, should theoretically achieve a better hypervolume on some fronts.

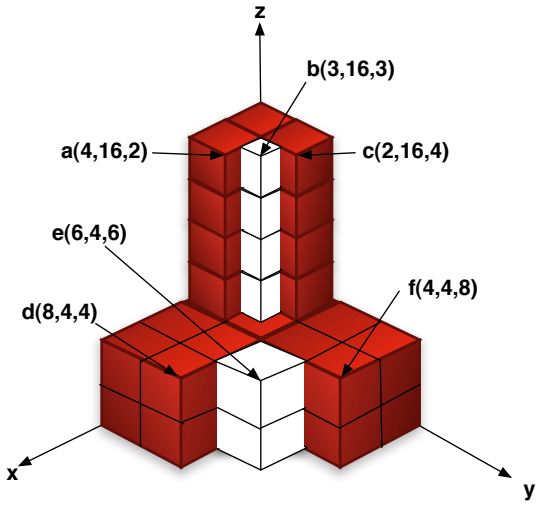


Fig. 1. Demonstrates a situation where the greedy point removal scheme described will fail to achieve the maximum hypervolume obtainable for the subset of points. If the selection scheme aims to remove two thirds of the points, then it will keep points A and F, obtaining a hypervolume of 224 rather than keeping points B and E obtaining a hypervolume of 252.

It is easy to generate an example where the greedy front reduction scheme will not find the reduced front with the optimal hypervolume. Figure 1 depicts an example where a local search technique could discover a substantially better front composition than the greedy scheme.

2) Ability to bound time taken by algorithm: We propose the use of a local search algorithm to maximise the hypervolume obtained by a reduced front. One advantage of this technique is that it is possible to stop the search after a given time limit. In contrast, it is not possible to reduce the time taken by the greedy front reduction method to less than the time taken for one iteration. Using a local search, a MOEA is able to optimise the composition of a front in a time-frame that is satisfactory to a user. Additionally, if the user is willing to allow further computation, the local search may achieve a better front composition than other techniques within an equivalent computation time.

3) HV Local Search: Our hypervolume front reduction local search algorithm operates as follows:

1) Generate initial front composition.
2) Perturb the front (resulting in a modified front of the same size).
3) Accept the new front composition as the current front if it has a better hypervolume.
4) Repeat steps 2-4 until run-time constraint is exceeded.

We tested other search algorithms such as Simulated Annealing and Evolutionary Algorithms and found that they did not achieve major improvements compared to a local search in the tested time frames.

IV. Experiments

We evaluated the front reduction techniques on two distinct fronts from the DTLZ[10] test suite: the spherical front and the discontinuous front. For each front, we generated mathematically a representative set of 10,000 points from the (known) Pareto optimal set: then to generate a front of size m, we sampled this set randomly. The linear front from DTLZ gives similar results to the spherical front, and the degenerate front gives anomalous results as its hypervolumes can be calculated in polynomial time[9].

We also tested these techniques on randomly generated fronts. For these fronts we generated sets of m mutually non-dominating points in n objectives simply 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 x to S only if $\bar{x} \cup S$ would be mutually non-dominating. We kept adding points until $|S| = m$.

We use the method used by Emmerich et al. where the reference point used is calculated using the smallest individual value in the front in each dimension. Discontinuous and spherical fronts were cast as a minimisation problem, as in DTLZ, while random fronts were cast as a maximisation problem.

Firstly the greedy front reduction algorithm was run on a diverse range of front types (varying objective functions, numbers of objectives, numbers of points) to determine an acceptable front composition containing half the individuals in a front. For each of these front types, we ran the greedy algorithm and local search on five different fronts. The hypervolume selection local search was allowed to run for an duration equivalent to the greedy front reduction method. The local search was run five times on each front and these results averaged.

All timings were performed on a dedicated 2.8GHz Pentium IV machine with 512MB of RAM, running Red Hat Enterprise Linux 3.0. All algorithms were implemented in C and compiled with gcc -O3.

TABLE I
SPHERICAL, 80% OF POINTS REMOVED

		One iteration, greedy	Local Search results			
# obj	# pts	Time greedy (s)	Time to match greedy (s)	HV ratio in equal time	HV ratio in 10s	HV ratio in 1s
5d	200	7.1939	2.0263 (5 fronts)	1.0014	1.0014	0.9989
6d	150	47.1758	2.6492 (5 fronts)	1.0089	1.0085	1.0009
7d	150	118.4760	5.2322 (5 fronts)	1.0060	1.0049	0.9896
8d	100	59.2300	3.0067 (5 fronts)	1.0033	1.0029	0.9949
9d	90	87.4727	3.2367 (5 fronts)	1.0169	1.0165	1.0049

TABLE II
DISCONTINUOUS, 80% OF POINTS REMOVED

		One iteration, greedy	Local Search results			
# obj	# pts	Time greedy (s)	Time to match greedy (s)	HV ratio in equal time	HV ratio in 10s	HV ratio in 1s
5d	300	77.0063	6.6908 (5 fronts)	1.0074	1.0028	0.9629
6d	150	130.7311	3.1080 (5 fronts)	1.0196	1.0148	0.9685
7d	80	116.0004	2.4076 (5 fronts)	1.0125	1.0120	0.9881
8d	45	95.7584	0.2384 (5 fronts)	1.0261	1.0261	1.0247
9d	40	140.9556	0.3205 (5 fronts)	1.0119	1.0118	1.0102

TABLE III
RANDOM, 80% OF POINTS REMOVED

		One iteration, greedy	Local Search results			
# obj	# pts	Time greedy (s)	Time to match greedy (s)	HV ratio in equal time	HV ratio in 10s	HV ratio in 1s
5d	200	22.4086	10.3430 (4 fronts)	1.0024	1.0006	0.9567
6d	100	16.3085	8.5742 (5 fronts)	1.0011	1.0002	0.9815
7d	80	83.9302	9.0946 (5 fronts)	1.0005	0.9999	0.9690
8d	45	27.9507	0.5876 (5 fronts)	1.0000	1.0000	0.9992
9d	40	125.4489	0.8926 (5 fronts)	1.0000	1.0000	0.9973

TABLE IV
SPHERICAL, 50% OF POINTS REMOVED

		One iteration, greedy	Local Search results			
# obj	# pts	Time greedy (s)	Time to match greedy (s)	HV ratio in equal time	HV ratio in 10s	HV ratio in 1s
5d	200	7.3649	26.0808 (4 fronts)	1.0000	0.9998	0.9923
6d	150	46.8909	89.7693 (2 fronts)	1.0000	0.9986	0.9890
7d	150	118.8479	216.6832 (2 fronts)	1.0000	0.9944	0.9803
8d	100	59.6259	57.8394 (4 fronts)	1.0001	0.9964	0.9790
9d	90	87.9446	84.8738 (3 fronts)	1.0000	0.9963	0.9780

TABLE V
DISCONTINUOUS, 50% OF POINTS REMOVED

		One iteration, greedy	Local Search results			
# obj	# pts	Time greedy (s)	Time to match greedy (s)	HV ratio in equal time	HV ratio in 10s	HV ratio in 1s
5d	300	77.0143	211.2232 (4 fronts)	1.0000	0.9880	0.9615
6d	150	131.1641	272.1418 (5 fronts)	1.0001	0.9717	0.9373
7d	80	120.8086	135.4446 (5 fronts)	1.0005	0.9773	0.9170
8d	45	96.0224	40.6998 (5 fronts)	1.0022	0.9885	0.9257
9d	40	139.9397	71.3201 (5 fronts)	1.0018	0.9822	0.8972

TABLE VI
RANDOM, 50% OF POINTS REMOVED

		One iteration, greedy	Local Search results			
# obj	# pts	Time greedy (s)	Time to match greedy (s)	HV ratio in equal time	HV ratio in 10s	HV ratio in 1s
5d	200	22.4266	N/A (0 fronts)	0.9994	0.9888	0.9158
6d	100	16.3825	N/A (0 fronts)	0.9997	0.9916	0.9058
7d	70	49.6235	N/A (0 fronts)	0.9998	0.9847	0.8672
8d	50	70.4793	99.3908 (3 fronts)	1.0000	0.9838	0.8485
9d	40	125.2400	113.9819 (5 fronts)	1.0000	0.9782	0.8392

TABLE VII
SPHERICAL, 20% OF POINTS REMOVED

		One iteration, greedy	Local Search results			
# obj	# pts	Time greedy (s)	Time to match greedy (s)	HV ratio in equal time	HV ratio in 10s	HV ratio in 1s
5d	200	7.2719	N/A (0 fronts)	1.0000	0.9999	0.9975
6d	200	45.0991	N/A (0 fronts)	1.0000	0.9982	0.9954
7d	150	120.7756	N/A (0 fronts)	1.0000	0.9981	0.9938
8d	100	59.4160	N/A (0 fronts)	1.0000	0.9981	0.9934
9d	90	89.0555	N/A (0 fronts)	1.0000	0.9969	0.9921

TABLE VIII
DISCONTINUOUS, 20% OF POINTS REMOVED

		One iteration, greedy	Local Search results			
# obj	# pts	Time greedy (s)	Time to match greedy (s)	HV ratio in equal time	HV ratio in 10s	HV ratio in 1s
5d	300	77.1823	N/A (0 fronts)	0.9999	0.9945	0.9884
6d	150	131.4490	N/A (0 fronts)	0.9998	0.9856	0.9779
7d	80	117.8601	N/A (0 fronts)	0.9998	0.9810	0.9672
8d	45	97.8401	N/A (0 fronts)	0.9995	0.9773	0.9534
9d	40	143.4202	451.7484 (3 fronts)	0.9997	0.9653	0.9513

TABLE IX
RANDOM, 20% OF POINTS REMOVED

		One iteration, greedy	Local Search results			
# obj	# pts	Time greedy (s)	Time to match greedy (s)	HV ratio in equal time	HV ratio in 10s	HV ratio in 1s
5d	200	22.5636	N/A (0 fronts)	0.9999	0.9971	0.9714
6d	100	16.4245	N/A (0 fronts)	0.9999	0.9959	0.9663
7d	70	50.4603	N/A (0 fronts)	0.9999	0.9853	0.9285
8d	45	28.5857	N/A (0 fronts)	0.9999	0.9941	0.9379
9d	40	128.5325	N/A (0 fronts)	0.9997	0.9608	0.9268

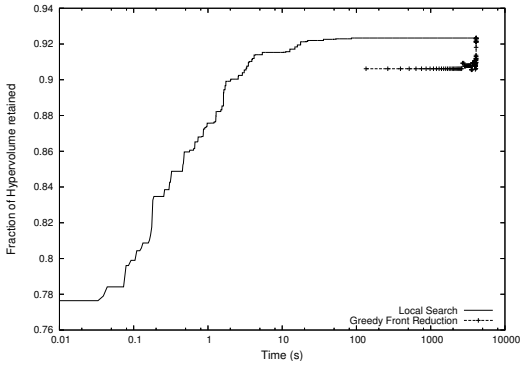


Fig. 2. Discontinuous 6d, 150pts, 80% removed.

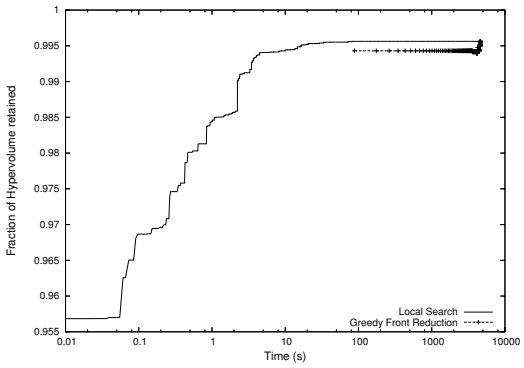


Fig. 3. Spherical 6d, 200pts, 80% removed.

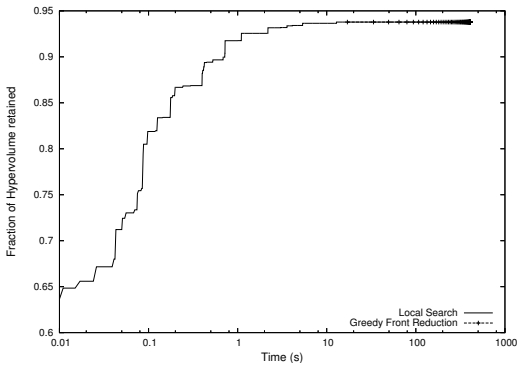


Fig. 4. Random 6d, 100pts, 80% removed.

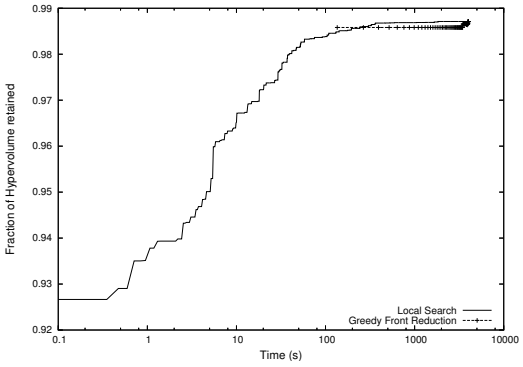


Fig. 5. Discontinuous 6d, 150pts, 50% removed.

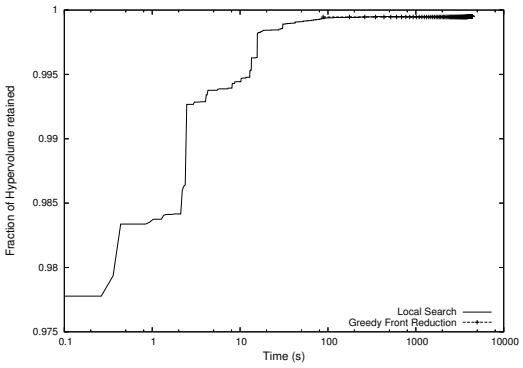


Fig. 6. Spherical 6d, 200pts, 50% removed.

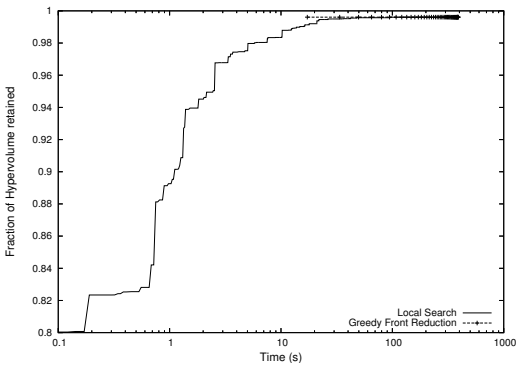


Fig. 7. Random 6d, 100pts, 50% removed.

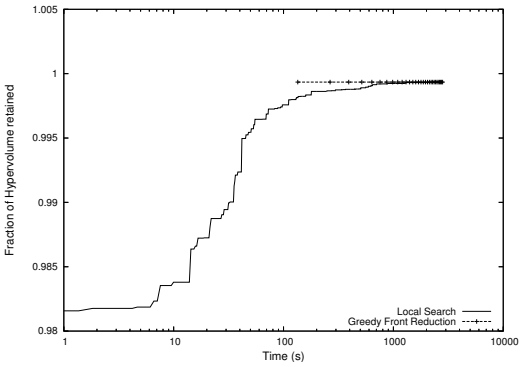


Fig. 8. Discontinuous 6d, 150pts, 20% removed.

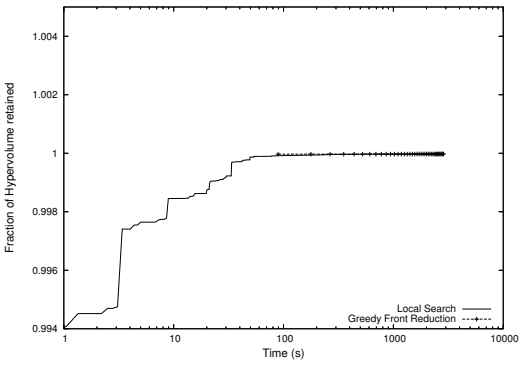


Fig. 9. Spherical 6d, 200pts, 20% removed.

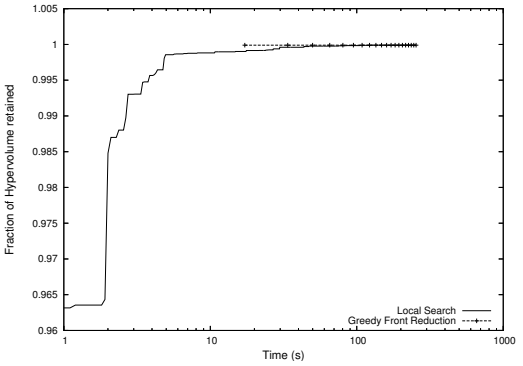


Fig. 10. Random 6d, 100pts, 20% removed.

A. Experiments reducing front by 80%

Tables I, II and III demonstrate that the local search has a significant advantage when a large proportion of the front is removed. In this case the local search quickly converges on an acceptable selection of points, and the advantage over the greedy selection algorithm is evident. In every case, the local search is able to dominate the greedy approach in both the time taken to find a solution as well as the solution's hypervolume. We believe this advantage is due to the relatively inexpensive cost of hypervolume evaluations compared to those made by the greedy approach. This allows the local search to perform more generations to find a better solution.

In all cases, the local search is able to achieve better hypervolumes than the greedy method in equivalent time, and is able to achieve equivalent results anywhere from 0.2% to 52% of the time taken by the greedy method.

Figures 2, 3 and 4 show that local search is able to dominate the greedy method in all cases. In these cases, when using only one iteration of the greedy method the solutions provided by the algorithm may be sub-par as evidenced by the improvement in hypervolume in later iterations.

We believe that these cases demonstrate a flaw in the greedy approach which finds it difficult to maximise the coverage of the front given that many points are removed.

B. Experiments reducing front by 50%

Tables IV, V and VI demonstrate that the local search achieves similar results to the greedy front reduction technique when only half of the front is removed. In less than half of the cases the local search is able to find a better front compositions than the greedy approach. To find equivalent results, the local search takes from 42% to 353% of the time taken by the greedy method. In most cases the local search takes longer to find equivalent solutions than the greedy approach. Despite this, use of a local search algorithm may still be preferable because it can stop at any point while optimising front composition and will still yield a better front composition than a random selected reduced front. Figures 5, 6 and 7 demonstrate this concept and show that while the local search does take a long time to find equivalent answers to the greedy approach, it is able to find solutions that may be considered worthwhile in less time while coming close to the greedy approach in equivalent time. It is also evident that the local search takes a long time to converge in these cases.

C. Experiments reducing front by 20%

Tables VII, VIII and IX demonstrate that the local search is relatively ineffective when only a small proportion of the points are to be removed. In these cases, the local search's hypervolume evaluations are expensive, due to the large fronts, and the local search is not able to perform many generations in which to optimise the composure of the front. In this case the greedy algorithm's performance is probably acceptable.

D. Experimental Discussion

Keep in mind that small changes in hypervolume may relate to substantially improved coverage of the front. For example, while a reduced front with only one solution may have a hypervolume that is a substantial fraction of the original front, this is not necessarily a desirable result and we still wish for the best coverage of the Pareto front possible.

Another consideration is that diminishing hypervolume improvements are realised as the proportion of the front increases. Every additional solution will obtain a smaller exclusive hypervolume due to the improved coverage of the front. As a result, a front retaining a large proportion of its solutions, even if these solutions are randomly selected, may have a hypervolume that appears very close to the entire front. Despite this, a front with better coverage of the Pareto front is still desirable and the relative hypervolumes does not necessarily reflect that an improvement in hypervolume of the reduced front is not beneficial. However, as a result of this observation, we believe that the use of hypervolume as a selection metric is most important when only retaining a small proportion of a front. In this case it is more difficult to obtain a good coverage of the Pareto front and thus it is very important that an algorithm is able to find a reduced front that best covers the Pareto front.

V. Conclusions and Future Work

We believe the results of the local search algorithm present a strong case for its use in MOEAs using hypervolume as a selection mechanism for problems in large numbers of objectives. The local search results in a large reduction in computation time over a greedy technique applying HSO to remove a single point at a time. Furthermore, in cases where a large proportion of the front is removed, the local search leads to a front composition with a better hypervolume than the greedy technique.

Overall, the point reduction algorithm and local search achieve similar hypervolumes over the test fronts. A more sophisticated hypervolume algorithm that calculates only the exclusive hypervolume contribution of a point, given the remaining front, may achieve better time performance for many of the tested fronts. Even so, we do not expect such an algorithm to surpass the local search in cases where a large number of points are to be removed from a front.

References

[1] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," in *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, Eds., Athens, Greece, 2002, pp. 95–100.

[2] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," Indian Institute of Technology, Kanpur, India, KanGAL report 200001, 2000.

[3] E. Zitzler, "Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications," Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[4] M. Laumanns, E. Zitzler, and L. Thiele, "A Unified Model for Multi-Objective Evolutionary Algorithms with Elitism," in *2000 Congress on Evolutionary Computation*, vol. 1. Piscataway, New Jersey: IEEE Service Center, July 2000, pp. 46–53.

[5] J. D. Knowles, D. W. Corne, and M. Fleischer, "Bounded archiving using the Lebesgue measure," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE Press, 2003, pp. 2490–2497.

[6] M. Emmerich, N. Beume, and B. Naujoks, "An EMO algorithm using the hypervolume measure as selection criterion," in *Proc. Evolutionary Multi-Criterion Optimization: Third Int'l Conference (EMO 2005)*, ser. Lecture Notes in Computer Science, C. A. C. Coello, A. H. Aguirre, and E. Zitzler, Eds., vol. 3410. Berlin: Springer, 2005, pp. 62–76.

[7] J. D. Knowles, D. W. Corne, and M. Fleischer, "Bounded Archiving using the Lebesgue Measure," in *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, vol. 4. Canberra, Australia: IEEE Press, December 2003, pp. 2490–2497.

[8] R. L. While, "A new analysis of the lebmeasure algorithm for calculating hypervolume." in *EMO*, ser. Lecture Notes in Computer Science, C. A. C. Coello, A. H. Aguirre, and E. Zitzler, Eds., vol. 3410. Springer, 2005, pp. 326–340.

[9] L. While, L. Bradstreet, L. Barone, and P. Hingston, "Heuristics for optimising the calculation of hypervolume for multi-objective optimisation problems," in *Proceedings of 2005 Congress on Evolutionary Computation (CEC'2005)*, 2005.

[10] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable Multi-Objective Optimization Test Problems," in *Congress on Evolutionary Computation (CEC'2002)*, vol. 1. Piscataway, New Jersey: IEEE Service Center, May 2002, pp. 825–830.

Paper 3 (Refereed)

L. Bradstreet, L. While, and L. Barone. A Fast Incremental Hypervolume Algorithm. *IEEE Transactions on Evolutionary Computation*, 12(6):714–723, December 2008

A Fast Incremental Hypervolume Algorithm

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

***Abstract*—When hypervolume is used as part of the selection or archiving process in a multiobjective evolutionary algorithm, it is necessary to determine which solutions contribute the least hypervolume to a front. Little focus has been placed on algorithms that quickly determine these solutions and there are no fast algorithms designed specifically for this purpose. We describe an algorithm, IHSO, that quickly determines a solution's contribution. Furthermore, we describe and analyse heuristics that reorder objectives to minimize the work required for IHSO to calculate a solution's contribution. Lastly, we describe and analyze search techniques that reduce the amount of work required for solutions other than the least contributing one. Combined, these techniques allow multiobjective evolutionary algorithms to calculate hypervolume inline in increasingly complex and large fronts in many objectives.**

***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], [4], has recently been finding favor as a metric for comparing the performance of multiobjective evolutionary algorithms (MOEAs). The hypervolume of a set of solutions measures the size of the portion of objective space that is dominated by those solutions collectively. Generally, hypervolume is favored because it captures in a single scalar both the closeness of the solutions to the optimal set and, to some extent, the spread of the solutions across objective space. Hypervolume also has nicer mathematical properties than many other metrics: Zitzler *et al.* [5] state that hypervolume is the only unary metric of which they are aware that is capable of detecting that a set of solutions X is not worse than another set X', and Fleischer [6] has proved that hypervolume is maximized if and only if the set of solutions contains only Pareto optima. Hypervolume has some nonideal properties too: it is sensitive to the relative scaling of the objectives, and to the presence or absence of extremal points in a front.

A fast algorithm for calculating hypervolume exactly is the *hypervolume by slicing objectives* algorithm (HSO) [7]–[9]. HSO works by processing the objectives in a front, rather than the points. It divides the n D-hypervolume to be measured into separate $n - 1$ D-slices through one of the objectives, then it calculates the hypervolume of each slice and sums these values to derive the total. In the worst case, HSO is exponential in the number of objectives, but until recently, it had better complexity than other algorithms. In addition, While *et al.* [10] have described good heuristics that optimize the order in which the objectives should be processed for a given front by estimating the "worst-case work" required to process the slices remaining after eliminating each objective. These heuristics reduce the running time of HSO for representative data by 25%–98%.

Algorithms from the computational geometry field have recently been applied to hypervolume calculation by Beume and Rudolph and separately by Fonseca *et al.* Beume and Rudolph [11] adapt the Overmars and Yap [12] algorithm for solving the Klee's measure problem to instead calculate the hypervolume of a front. Similarly, Fonseca *et al.* [13] apply the Overmars and Yap algorithm for the 3-D base case in order to provide a performance boost to HSO. Beume and Rudolph's adaptation boasts an impressive improvement in worst-case complexity, from $O(n^{d-1})$ to $O(n \log n + n^{d/2})$, however, as of yet there are no performance comparisons between their algorithm and HSO with heuristics.

Hypervolume is also used inline in some evolutionary algorithms, as part of a diversity mechanism [14], as part of an archiving mechanism [15], or recently as part of the selection mechanism [16], [17]. The requirement in such cases is to compare the *exclusive hypervolume* contributed by different points, i.e., the amount by which each point increases the hypervolume of the set. Clearly, if hypervolume calculations are incorporated into the execution of an algorithm (as opposed to hypervolume used as a metric after execution is completed), there is a much stronger requirement for those calculations to be efficient. The ideal for such uses is an incremental algorithm that minimizes the expense of repeated invocations.

The principal contributions of this paper are a version of HSO which is customized for inline incremental hypervolume calculations, and queueing techniques and heuristics that improve performance for hypervolume algorithms used within a MOEA.

The customized algorithm has two parts.

- The algorithm *incremental HSO* (IHSO) calculates the exclusive hypervolume of a point p relative to a set of points S. The principal optimizations in IHSO are minimizing the number of slices that have to be processed, and ordering the objectives intelligently.
- The algorithm IHSO* performs point selection for diversity, archiving, or fitness. IHSO* works by repeated application of IHSO to calculate the exclusive hypervolume for each point in a set. The principal optimizations in IHSO* are ordering the points intelligently, and calculating as little hypervolume as possible for each point.

IHSO* will provide a substantial performance improvement for evolutionary algorithms that perform inline incremental hypervolume calculations. We note that although Beume and Rudolph's recent work [11] does improve the complexity of hypervolume algorithms for metric calculations, customized algorithms are not yet available for incremental hypervolume calculations.

Manuscript received December 21, 2006; revised July 30, 2007. First published March 28, 2008; current version published December 12, 2008.

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

Digital Object Identifier 10.1109/TEVC.2008.919001

The rest of this paper is structured as follows. Section II defines the concepts and notation used in multiobjective optimization and throughout this paper. Section III describes HSO and the heuristics used to optimize its performance. Section IV describes how HSO can be customized into IHSO and IHSO* to calculate exclusive hypervolume efficiently. Section V reports on some experiments to determine the fastest incremental algorithm, and to explore some important issues for users of the algorithm. Section VI concludes this paper and discusses some possibilities for future work.

II. Definitions

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 nondomination 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 *nondominated* with respect to a set of solutions X iff there is no vector in X that dominates $\overline{x}$. X is a *nondominated set* iff all vectors in X are mutually nondominating. Such a set of objective vectors is sometimes called a *nondominated front*.

A vector $\overline{x}$ is *Pareto optimal* iff $\overline{x}$ is nondominated 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.

Given a set X of solutions returned by an algorithm, the question arises how good the set X is, i.e., how well it approximates the Pareto optimal set. One metric used for comparing sets of solutions is to measure the *hypervolume* of each set. The hypervolume of X is the total size of the space that is dominated by the solutions in X. 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 fronts being compared.) If a set X has a greater hypervolume than a set X', then X is taken to be a better set of solutions than X'.

Precise definitions of these terms can be found in [18].

III. Hypervolume By Slicing Objectives

Given a set of mutually nondominating points in n objectives, HSO is based on the idea of processing the points *one objective at a time*.

Initially, the points are sorted by their values in the first objective to be processed. These values are then used to cut cross-sectional "slices" through the hypervolume: each slice will itself be an $n-1$-objective hypervolume in the remaining objectives. The $n-1$-objective hypervolume in each slice is calculated and each slice is multiplied by its depth in the first objective, then these n-objective values are summed to obtain the total hypervolume. Each slice through the hypervolume will contain a different subset of the original points. The kth slice from the top can contain only the points with the best k values in the first objective. However, not all points "contained" by a slice will contribute volume to that slice: some points may be dominated in the remaining objectives and will contribute nothing. After each step, the number of objectives is reduced by one, the points are resorted in the next objective, and newly dominated points within each slice are discarded.

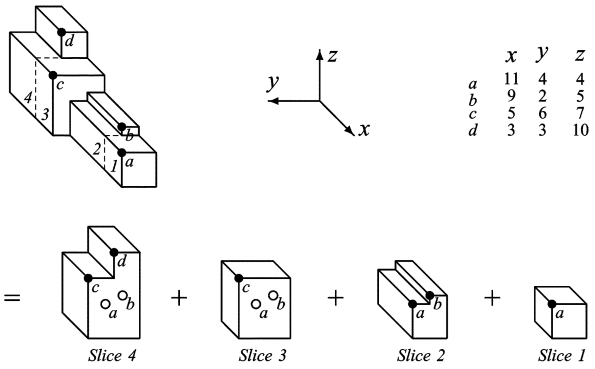


Fig. 1. One step in HSO for the four three-objective points shown. Objective x is processed, leaving four two-objective shapes in y and z. Points are marked by circles and labeled with letters: unfilled circles represent points that are dominated in y and z. Slices are labeled with numbers, and are separated on the main picture by dashed lines. (Figure reproduced from [9].)

Fig. 1 shows the operation of one step in HSO, including the slicing of the hypervolume, the allocation of points to each slice, and the elimination of newly dominated points.

The natural base case for HSO is when only one objective remains, when there can be only one nondominated point left in each slice. The value of this point is then the one-objective hypervolume of its slice. However, in practice, for efficiency reasons, HSO terminates when two objectives remain, which is an easy and fast special case.

Fig. 2 gives pseudocode for HSO.

A. The Complexity and Performance of HSO

The following recurrence relation captures the worst-case complexity of HSO [9]:

$$f(m,1) = 1 \tag{1}$$

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

The summation in (2) represents the fact that each slicing action generates m slices that are processed independently to derive the hypervolume of the front.

Solving this recurrence relation gives the following [9]:

$$f(m,n) = \binom{m+n-2}{n-1}. \tag{3}$$

Thus, HSO is exponential in the number of objectives n, in the worst case (we assume that $m > n$).

The "worst case" in this context means we assume that no (partial) point is ever dominated during the execution of HSO, thus maximizing the number of points in each slice that is processed. However, this is unlikely to be true for real-world fronts. The amount of time required to process a given front depends

```
hso (ps):
  pl = sort ps worsening in Objective 1
  s = {(1, pl)}
  for k = 1 to n-1
    s' = {}
    for each (x, ql) in s
      for each (x', ql') in slice (ql, k)
        add (x * x', ql') into s'
    s = s'
  vol = 0
  for each (x, ql) in s
    vol = vol + x * |head (ql)[n] - refPoint[n]|
  return vol

slice (pl, k):
  p = head (pl)
  pl = tail (pl)
  ql = []
  s = {}
  while pl /= []
    ql = insert (p, k+1, ql)
    p' = head (pl)
    add (|p[k] - p'[k]|, ql) into s
    p = p'
    pl = tail (pl)
  ql = insert (p, k+1, ql)
  add (|p[k] - refPoint[k]|, ql) into s
  return s

insert (p, k, pl):
  ql = []
  while pl /= [] && head (pl)[k] beats p[k]
    append head (pl) to ql
    pl = tail (pl)
  append p to ql
  while pl /= []
    if not (dominates (p, head (pl), k))
      append head (pl) to ql
    pl = tail (pl)
  return ql

dominates (p, q, k):
  d = True
  while d && k <= n
    d = not (q[k] beats p[k])
    k = k + 1
  return d
```

Fig. 2. Pseudocode for HSO. (Code reproduced from [10].)

5	⋯	5	1
4	⋯	4	2
3	⋯	3	3
2	⋯	2	4
1	⋯	1	5

Fig. 3. A pathological example for HSO. This pattern describes sets of five points in n objectives, $n \geq 3$. All columns except the last are identical. The pattern can be generalized for other numbers of points. (Example reproduced from [10].)

crucially on how many points are dominated at each stage and, in addition, on how early in the process points dominate other points.

From this fact, we can infer that the time to process a given front varies with the order in which the objectives are processed. A simple example illustrates how. Consider the set of points in Fig. 3, in a maximization problem.

If we process the first objective (or, in fact, any objective except the last), no point dominates any other point in the list in the remaining $n - 1$ objectives. Thus, we do indeed have the worst case for HSO, generating m slices containing, respectively, $1, 2, \ldots, m$ points.

If we process the last objective, each point dominates all subsequent points in the list in the remaining $n-1$ objectives. Then, we generate m slices *each containing only one point*. Specifically, the top slice (corresponding to the highest value in the last objective) contains only the point $1 \cdots 1$, the second slice contains only the point $2 \cdots 2$, all the way down to the bottom slice, which contains only the point $m \cdots m$. This is of course the best case for HSO, and the hypervolume is calculated much more quickly.

Note that, in general, there is a continuum of performance improvement available: for example, for the points in Fig. 3, the earlier the last objective is processed, the faster the hypervolume will be calculated. Thus, enhancing HSO with a mechanism to identify a good order in which to process the objectives in a given front can make a substantial difference to its performance.

B. Optimizing the Performance of HSO

While *et al.* describe and evaluate two heuristics for choosing the order in which the objectives should be processed for a given front [10]. They characterize the better heuristic as "minimizing the amount of worst-case work" (MWW). For each objective, MWW:

- calculates the number of nondominated partial points that will be in each slice;
- estimates the worst-case amount of work required to process each slice, using (3);
- and sums these values to estimate the amount of work required if HSO processes this objective first.

Then, HSO processes the objective that represents the least work. MWW is applied at each iteration of HSO until only four objectives remain.

An empirical comparison of HSO versus HSO+MWW on randomly generated fronts and on fronts from the well-known DTLZ test suite [19] shows that MWW can reduce the time to process fronts in 5–9 objectives by 25%–98%.

IV. Running HSO Incrementally

Hypervolume is used inline in an evolutionary algorithm in three ways:

- as part of a diversity mechanism;
- as part of an archiving mechanism;
- as part of the selection mechanism.

In all three contexts, the requirement is to calculate the *exclusive hypervolume* contributed by a point p relative to a set of points S, i.e., how much additional hypervolume we get by adding p to S. This can be defined as

$$\mathrm{ExcHyp}(p, S) = \mathrm{Hyp}(S \cup \{p\}) - \mathrm{Hyp}(S). \qquad (4)$$

For example, the exclusive hypervolume contributed by Point **b** in Fig. 1 is the cuboid bounded by **b** and by the point (5, 0, 4), i.e., the long thin cuboid on which **b** sits. Note that exclusive hypervolumes usually have a much more complicated shape than this: consider as an example Point **c** in Fig. 1.

A typical requirement when hypervolume is used in this way is to calculate the exclusive hypervolume contributed by each of a set of points S, then to discard the point in S that contributes

```
ihso (z, ps):
  pl = sort ps worsening in Objective 1
  s = {(1, pl)}
  for k = 1 to n-1
    s' = {}
    for each (x, ql) in s
      for each (x', ql') in slice (z, ql, k)
        add (x * x', ql') into s'
    s = s'
  vol = 0
  for each (x, ql) in s
    vol = vol + x * |head (ql)[n] - refPoint[n]|
  return vol

slice (z, pl, k):
  ql = []
  s = {}
  v = z[k]
  dominated = false
  while pl /= [] and not dominated
    p = head (pl)
    pl = tail (pl)
    if v beats p[k]
      then add (|v - p[k]|, ql) into s
           v = p[k]
    ql = insert (p, k+1, ql)
    dominated = dominates (p, z, k+1)
  if not dominated
    then add (|v - refPoint[k]|, ql) into s
  return s
```

Fig. 4. Pseudocode for IHSO. Other functions are defined in Fig. 2. The reordering of the objectives is not shown.

the least exclusive hypervolume. Thus, we return the subset of S of cardinality $|S|-1$ that has the largest hypervolume. This idea can be extended to situations where we need to discard multiple points from S[20], [21], but we do not deal with this issue here.

Obviously, we can calculate the exclusive hypervolume contributed by each point in S by $|S|+1$ applications of HSO: one to S itself, and one to each subset of size $|S|-1$. However, we can do far better than this performance-wise by customizing HSO to calculate exclusive hypervolumes directly. We define a new algorithm IHSO that takes a point p and a set of mutually nondominating points S and returns $\mathrm{ExcHyp}(p, S)$. We customize HSO in three ways to derive IHSO.

1) Disregarding "higher" slices: p will not contribute to any slice above itself in the current objective, therefore, the hypervolumes of these slices need not be calculated. For example, in Fig. 1, Point **b** contributes nothing to Slice 1.
2) Disregarding some "lower" slices: if p is dominated by a point q in S in the objectives after the current one, then p will not contribute to any slice containing q (or any point that dominates q), and the hypervolumes of these slices need not be calculated. For example, in Fig. 1, Point **b** contributes nothing to Slices 3 or 4, because it is dominated by Point **c** in y and z.
3) Processing the objectives in the right order: as with HSO, we can optimize the performance of IHSO by selecting a good order in which to process the objectives.

Fig. 4 gives pseudocode for IHSO. The code assumes that none of the points in `ps` dominates `z`, although the converse is not true: `z` may dominate one or more points in `ps`. The principal differences from HSO in Fig. 2 are in the function `slice`.

- A slice is added to `s` only if it is below `z` in the current objective, i.e., below `z[k]`.

```
Order the points by some metric
Evaluate the first point
Save this point as the current smallest
for each subsequent point
  Evaluate point while worse than the smallest
  Save point if it is the smallest
return the smallest point
```

Fig. 5. Outline of the point-ordering scheme in IHSO*.

- If at any time `z` is dominated in the remaining objectives, no more slices are added to `s`.

Given IHSO, we can define an algorithm IHSO* to identify the point in a set S that contributes the least exclusive hypervolume to S. We use $|S|$ applications of IHSO to calculate $\mathrm{ExcHyp}(p, S - \{p\})$ for each point p in S, then simply return the point with the smallest value. Within IHSO*, it is useful to order the calculations so that small points are likely processed first. This enables early termination for subsequent points: if the exclusive hypervolume for p is known, then as soon as the exclusive hypervolume for q is known to be bigger, we can eliminate q from consideration as the smallest contributor. Note also that the order in which the objectives are processed can be different for different points in S.

Thus, there are two questions to be answered in order to derive an efficient implementation of IHSO and IHSO*.

A. How Do We Order the Objectives When Calculating $ExcHyp(p, S)$ in IHSO?

We have tried several heuristics that can be used to order the objectives for a point p.

1) *Rank*: process first the objective in which p is best, so that it is more likely to be dominated early.
2) *Reverse rank*: process first the objective in which p is worst, so that there are fewer slices to calculate.
3) *Dominated*: find the point q that beats p in the most objectives, and first process the slices in which p beats q. This method partitions the objectives between those where q beats p, and those where p beats q. Within these partitions, order objectives by the rank heuristic.
4) *MWW*: as defined in Section III-B.

While these heuristics can be used to reorder objectives for all calculations [10], we find experimentally that it is more effective to reorder the objectives for each individual slice recursively calculated by IHSO. Although this comes at additional cost, savings are made on slices that are expensive to calculate, for example, a slice with a difficult shape, or one with many points or objectives. One example of where savings are made using this approach is for the dominated heuristic, where the point q used to reorder the objectives may not even exist in a given slice.

B. How Do We Order the Points When Calculating Their Exclusive Hypervolumes in IHSO*?

We have devised two schemes to improve the performance of IHSO* when used to find the worst contributing point. The first scheme reorders the points with the aim of calculating the worst point early. Unnecessary calculations are then saved on subsequent points. This scheme is outlined in Fig. 5.

```
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 queueing scheme in IHSO*.

We have defined two measures that can be used to order the points. Each point p can be assessed by the following.

1) *Rank*: the sum of the number of points that beat p in each objective. Points are sorted in descending order.
2) *Volume*: the "inclusive hypervolume" of p: the product of its objectives. Points are sorted in ascending order.

The point with the least hypervolume contribution to the set is more likely to have a large rank value and to have a small inclusive hypervolume.

We have also defined an alternative "best-first" queueing (BFQ) scheme that processes the points "concurrently," to avoid the question of ordering. This scheme is outlined in Fig. 6. The principal parameter in the queueing scheme is the definition of "a bit," i.e., the granularity of the concurrency. If the granularity is too coarse, the algorithm will do more calculation than necessary: if it is too fine, the overhead of managing the queue will become significant. At present, we use a simple granularity scheme based on specifying the dimensionality of a hypervolume to be calculated in each iteration of the loop. A granularity of ρ means that one slice in ρ objectives is calculated in each iteration. This dimensionality is set according to (5)

$$\rho = \max(\min(n, 4), n - 2). \tag{5}$$

The application of max in (5) means that for low dimensionalities, we abandon the queueing scheme and just calculate the complete exclusive hypervolume of each point. This system works well for the limited range of dimensionalities studied so far, but it is likely to need updating in the future.

We have implemented the BFQ scheme using a heap based priority queue. Note that this approach requires the IHSO* algorithm to be modified to update the overall hypervolume contribution of points whenever a slice is calculated in ρ dimensions.

Section V describes an empirical comparison of the performance of these methods.

V. Experiments and Evaluation

We performed a series of experiments to explore issues with IHSO and IHSO*, and to determine the combination of heuristics that offers the best performance. We used two types of data in the experiments.

- We used 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 nondomination, we initialized $S = \phi$ and added each point $\overline{x}$ to S only if $\overline{x} \cup S$ would be mutually nondominating.
- We used the discontinuous and spherical fronts from the DTLZ test suite [19]. 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, and we omit the degenerate front because it can be processed in polynomial time [9], [10], and it is somewhat unrealistic anyway.

The DTLZ fronts may not realistically represent real-world data, and therefore we believe that random fronts provide a better performance baseline for most problems. As it is hard to give performance comparisons for all front shapes and types, random data may provide a better approximation of IHSO*'s performance on these fronts than specific DTLZ fronts.

The data used in the experiments are available [22]. Source code for our optimized algorithm is available from the same site. All timings were performed on a dedicated 2.8 GHz Pentium IV machine with 512 Mb of RAM, running Red Hat Enterprise Linux 3.0. All algorithms were implemented in C and compiled with *gcc -O3*. All times include the costs of calculating the heuristics, where appropriate.

A. Does Point-Ordering Matter?

We performed a series of experiments to establish whether the time needed to identify the least-contributing point in a front depends on the order in which the points in the front are processed. Each graph in Fig. 7 shows five lines, each of which corresponds to one front with 30 points in nine objectives. Each line plots a continuous cumulative histogram of the distribution of times needed to determine the least-contributing point for 50 000 randomly generated point-orderings of that front. No objective-reordering is applied.

Fig. 7 shows clearly that evaluating points in the right order can make a huge difference to the performance of the algorithm. The raw data show that typically the best order is processed 300–6000 times faster than the worst order, and 60–200 times faster than the median order.

Thus, point-ordering will play an important role in optimizing the performance of IHSO*.

B. What Is the Best Algorithm?

We performed a series of experiments to identify a good combination of heuristics to use in IHSO and IHSO*. Each graph in Figs. 8–10 shows the performance for varying front-sizes of the six combinations of the following heuristics from Section IV.

- *Point-ordering:* Rank, volume, and the best-first queueing scheme.
- *Objective-ordering:* Rank and dominated.

Other heuristics discussed in Section IV performed consistently worse than those illustrated in the figures. The graphs plot fronts up to 1000 points that can be processed in 1.5 s: this should be a reasonable amount of time for an incremental hypervolume calculation for most applications. Figs. 8–10 show that using the BFQ approach gives the best results other than in five objectives. Although BFQ loses slightly in five objectives, this is probably as a result of the overhead caused by the priority queue. If point reordering algorithms make perfect decisions, they will always outperform the BFQ approach. However, the savings made for more complex fronts outweigh the cost of maintaining the queue as can be observed for all front types in 8 and 11 objectives. Additionally, using a point-ordering algorithm rather than BFQ introduces a greater uncertainty in the

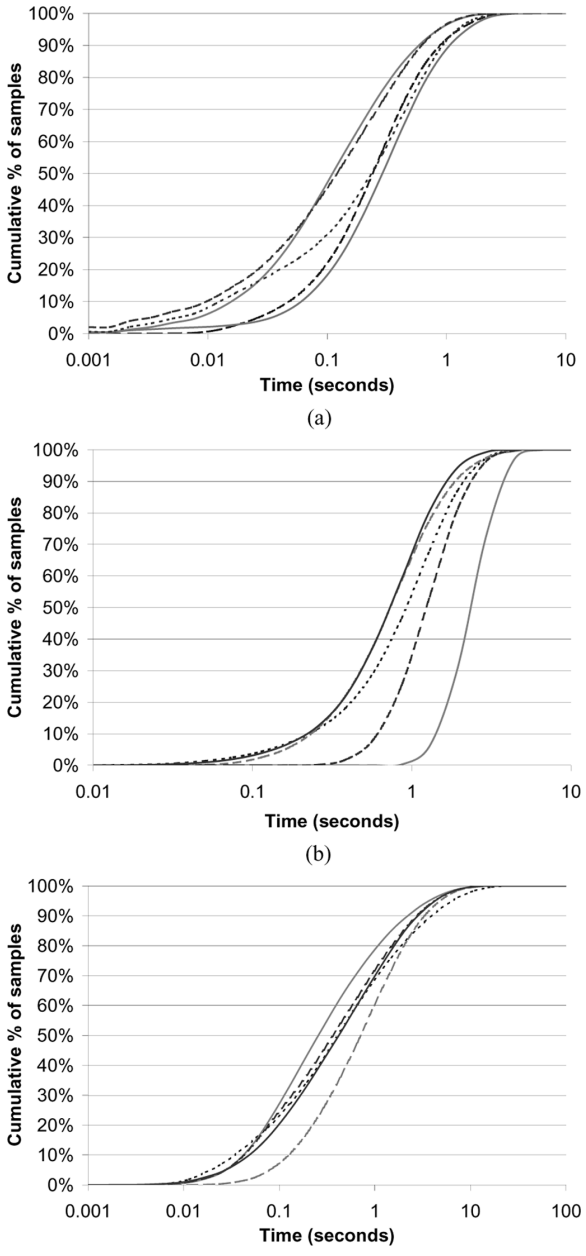


Fig. 7. Variation in processing time for IHSO* for different point-orderings. The five lines on each graph each plot a continuous cumulative histogram of the (log-scale) processing times for 50 000 distinct orderings for one front. (a) Random fronts: 30 points in nine objectives. (b) Discontinuous fronts: 30 points in nine objectives. (c) Spherical fronts: 30 points in nine objectives.

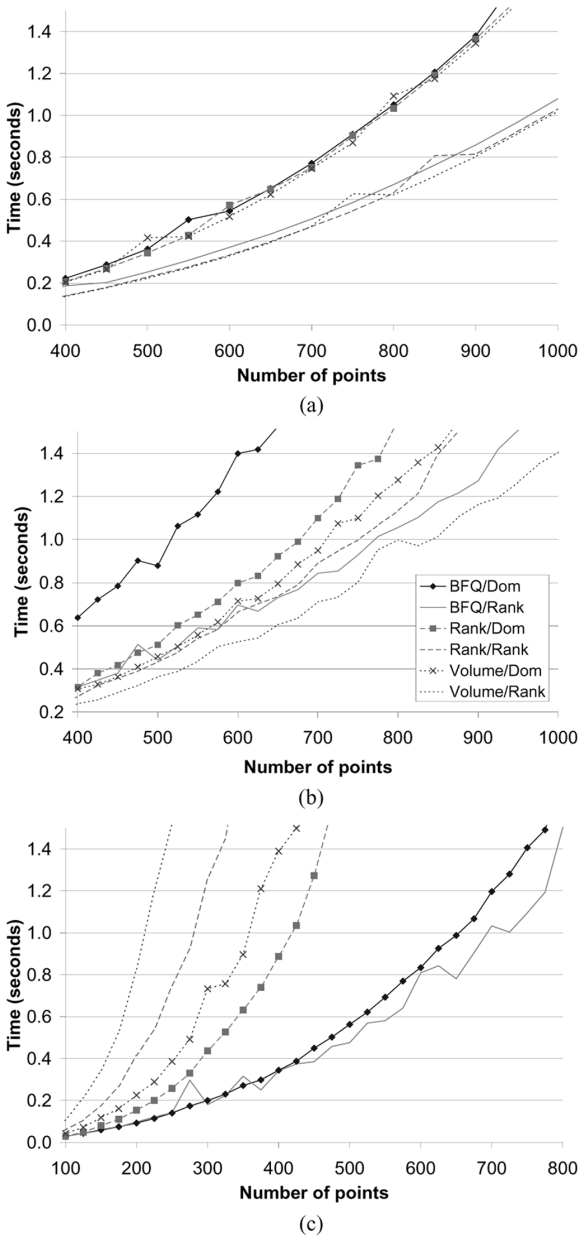


Fig. 8. Comparison of the performance of IHSO* with various heuristic combinations. Each line plots the average processing time for 200 distinct fronts in five objectives. The legend on the middle graph applies for all three. (a) Random fronts in five objectives. (b) Discontinuous fronts in five objectives. (c) Spherical fronts in five objectives.

results. A bad point-ordering decision can, in the worst case, require the calculation of every point's entire exclusive hypervolume. This effect is evident when comparing the BFQ/rank algorithm to the rank/rank algorithm for random fronts, shown in Fig. 9(a). While the results are reasonably close for most of the data points, large fluctuations are observed.

For all discontinuous and random data, BFQ/rank compares favorably to or beats all other objective heuristic and point-ordering techniques. This dominance increases with the number of objectives.

For spherical data, the dominated heuristic performs extremely well. However, we believe spherical data is being especially exploited by this heuristic. Examination of the data

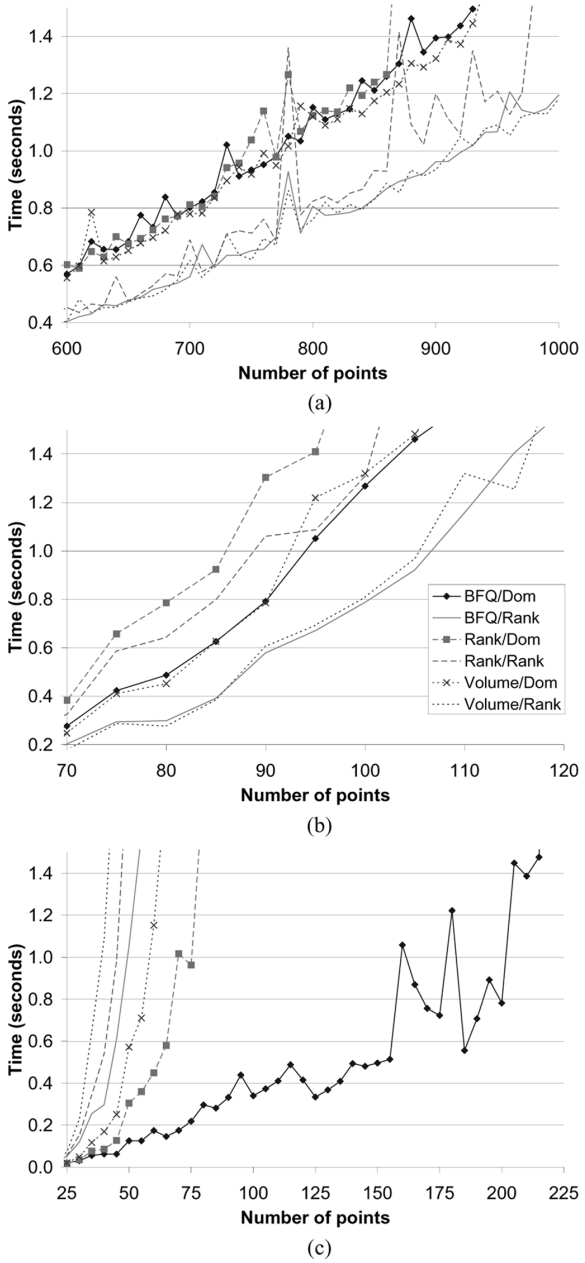


Fig. 9. Comparison of the performance of IHSO* with various heuristic combinations. Each line plots the average processing time for 200 distinct fronts in eight objectives. The legend on the middle graph applies for all three. (a) Random fronts in eight objectives. (b) Discontinuous fronts in eight objectives. (c) Spherical fronts in eight objectives.

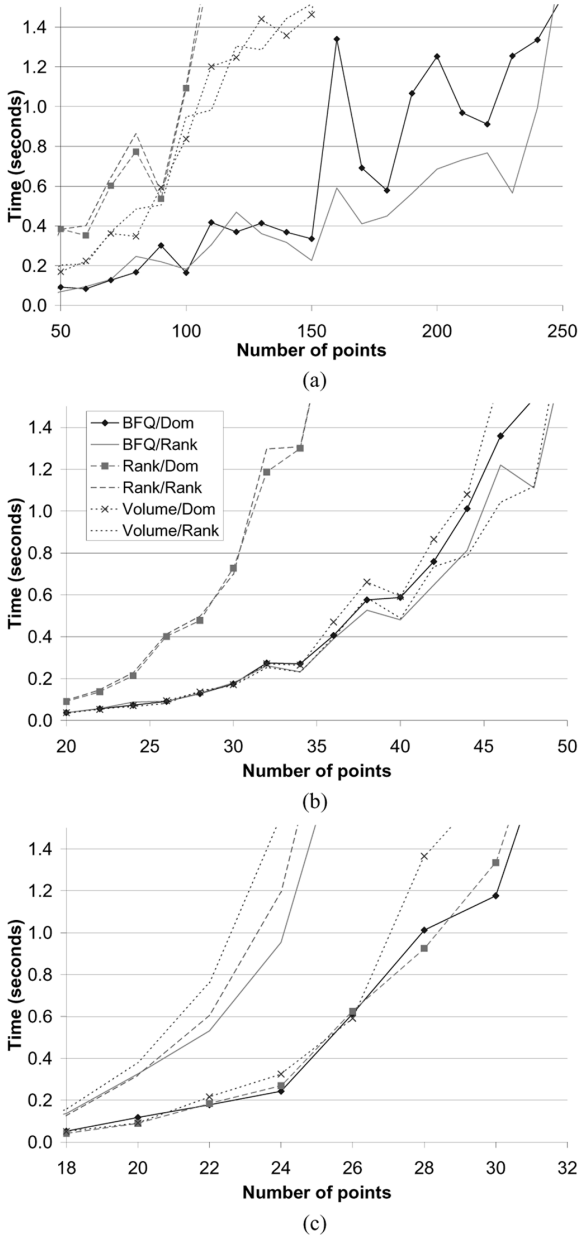


Fig. 10. Comparison of the performance of IHSO* with various heuristic combinations. Each line plots the average processing time for 200 distinct fronts in 11 objectives. The legend on the middle graph applies for all three. (a) Random fronts in 11 objectives. (b) Discontinuous fronts in 11 objectives. (c) Spherical fronts in 11 objectives.

reveals that a large proportion of the points are dominated early if two particular objectives are processed first. As such, we believe that spherical data does not very well represent real-world data. However, the dominated heuristic will, due to its nature, provide better results for exploitable real-world fronts, where many of the points contribute in only a small proportion of objectives. Additionally, the spherical data does point out a reason why the BFQ approach is superior to point-ordering heuristics. In all cases, BFQ commands a massive lead over the point heuristics which demonstrates the sensitivity of our point-ordering heuristics to front shapes. We take this as further evidence that the BFQ technique is more robust than point-ordering techniques.

TABLE I
TYPICAL SIZES OF FRONTS IN VARIOUS NUMBERS OF OBJECTIVES THAT IHSO* CAN PROCESS IN 1 S

n	Random	Discontinuous	Spherical
5	955	750	700
6	950	280	240
7	940	170	92
8	880	105	47
9	830	75	32
10	490	58	28
11	220	44	24
12	70	36	20
13	42	28	16

Table I shows what size of front optimized IHSO* (rank heuristic and BFQ) can process in 1 s, on average, for each front-type. Average processing time is the most important consideration, as IHSO will be called many times in a typical MOEA run.

C. How Does Performance Vary With the Data?

Figs. 8–10 only plot the average performance of the various algorithms as front size increases. We also performed a series of experiments to investigate how the performance of optimized IHSO* varies for a given front with the nature of a front.

Each graph in Fig. 11 plots a histogram showing the distribution of times needed to process 50 000 different fronts of the relevant type, and also the cumulative proportions of those fronts that are processed within a given time.

While the great majority of fronts are processed very quickly, there are cases where finding the least contributing point takes a disproportionate amount of time. Such outliers could be due to several factors. As the fronts become large, in some cases there are many points that contribute similar hypervolumes and their contribution may be difficult to calculate. For example, in the case where every point contributes the same hypervolume, neither the point-ordering nor BFQ techniques help performance.

Thus, while the average performance of IHSO* is extremely good, this performance cannot be guaranteed for complex fronts.

D. Does the Choice of Reference Point Affect Performance? and Does Scaling Objective Values Affect Performance?

Choice of reference point can significantly affect performance for the BFQ scheme. Fig. 12 illustrates these effects.

Each graph in Fig. 12 shows the performance of optimized IHSO* at m points in 9-D for the relevant front type, with two lines: the first plots time to determine the smallest point versus reference point offset, averaged over 200 fronts, and the second plots the same with all objectives scaled to $[0, 1]$. Keep in mind that reference point offsets are relatively "larger" for the scaled fronts, for example, compare point $(10, 10)$ with reference point of $(11, 11)$ to a point $(1, 1)$ with reference point $(2, 2)$. Therefore, scaled and unscaled fronts are not necessarily comparable for a given offset value.

The graphs show the observed results are due to several effects that result from a change in reference point. First, the choice of reference point influences which point has the smallest contribution. Second, regardless of whether a change in reference point changes which point is the smallest, a change in a point's contribution may also require further calculation of other points to prove that it is the smallest. Similar effects are caused by scaling objectives.

Although the reference point should not be chosen for performance criteria and rather so that the "best" points are retained,

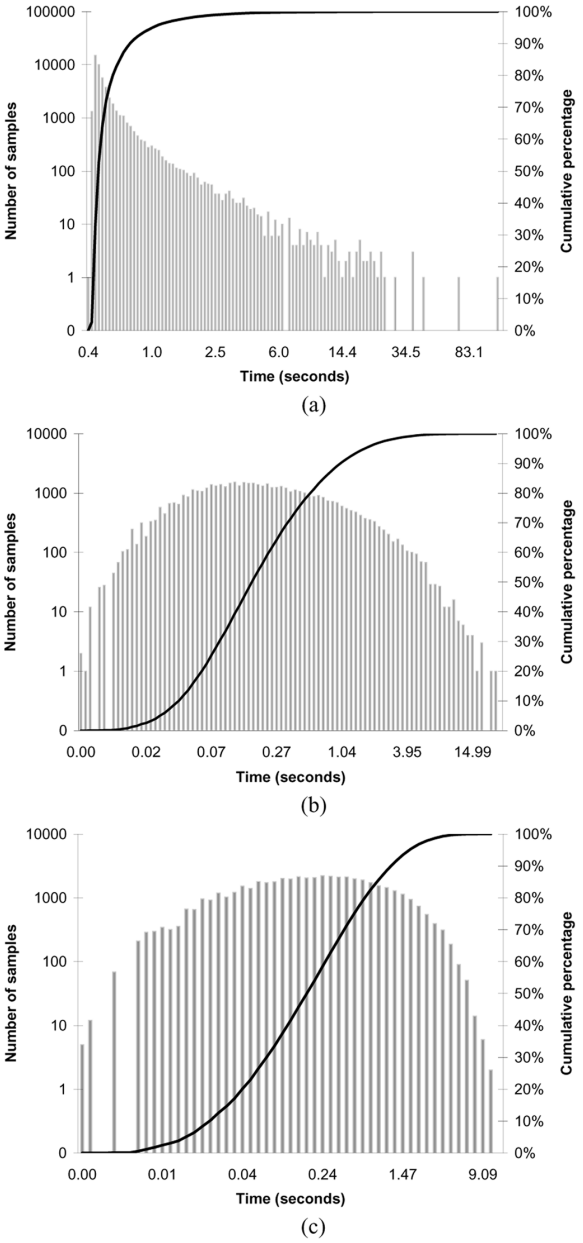


Fig. 11. Variation in processing time for optimized IHSO* for different fronts. Each graph plots a histogram of the (log-scale) processing times for 50 000 distinct fronts, and also the proportion of the fronts that were processed within a given time. (a) Random fronts: 650 points in nine objectives. (b) Discontinuous fronts: 65 points in nine objectives. (c) Spherical fronts: 30 points in nine objectives.

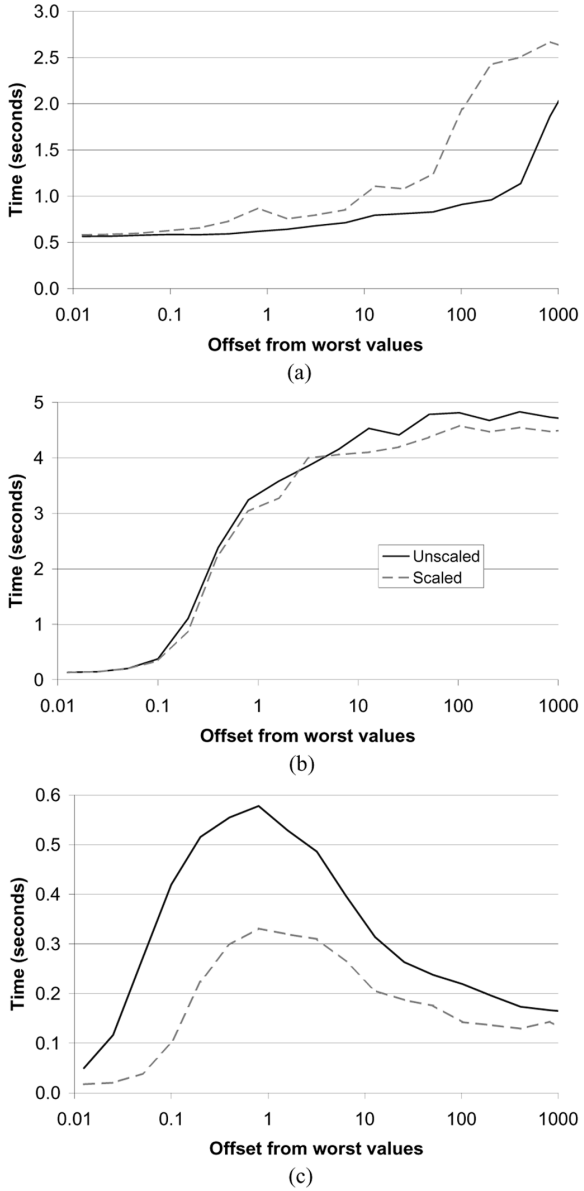


Fig. 12. Variation in processing time for optimized IHSO* for different reference points. Each graph plots the average (log-scale) processing time for 200 distinct fronts against the offset of the reference point from the worst value in each objective. The graphs show IHSO* applied to raw data, and to data scaled to [1] in each objective. The legend on the middle graph applies for all three. (a) Random fronts: 650 points in nine objectives. (b) Discontinuous fronts: 65 points in nine objectives. (c) Spherical fronts: 30 points in nine objectives.

the resulting effect on performance should be kept in mind when evaluating IHSO* on difficult fronts.

VI. Conclusion and Future Work

We have described a new algorithm for the calculation of incremental hypervolume when used within evolutionary algorithms, and techniques to apply this algorithm that minimize its cost. By applying heuristics to reorder objectives, we are able to increase the size of the fronts we are able to process. Additionally, by applying a best-first queueing approach we are able to calculate only as much of a point's hypervolume as is necessary to prove that it is not the smallest. We have demonstrated that, in general, this approach is superior to processing points using point reordering heuristics.

Through the combination of these techniques, we have described a method to effectively deal with very large numbers of points in many objectives within an EA. In doing so, the use of hypervolume should be computationally practical in tackling most complex real-world multiobjective problems. We recommend the BFQ strategy and the rank objective heuristic as a combination that performs well on a range of problems. However, better objective heuristics may exist for some particular front types.

Given the introduction of recent work on hypervolume for metric calculations, future work will look at adapting solutions to the Klee's measure problem to incremental hypervolume calculations. This would involve an adaptation of ideas from the Overmars and Yap algorithm to quickly perform incremental hypervolume calculations, and the application of our BFQ strategy for worst-point search. This new algorithm may also benefit from objective reordering heuristics similar to those described. Ideally, the combination of these works would allow the use of hypervolume within EA optimization to be not only practical but relatively inexpensive for all but the most difficult problems.

Acknowledgment

The authors would like to thank S. Huband for providing the raw DTLZ data, and K. Murray for advice on statistical issues.

References

[1] R. Purshouse, "On the evolutionary optimization of many objectives," Ph.D. dissertation, Univ. of Sheffield, Sheffield, U.K., 2003.
[2] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications," Ph.D. dissertation, Swiss Federal Inst. 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.*, R. Eberhart, Ed., 2000, pp. 46–53.
[4] M. Fleischer and L. E. Thiele, "The measure of Pareto optima: Applications to multi-objective metaheuristics," in *Proc. Evol. Multi-objective Opt.*, C. M. Fonseca, P. J. Fleming, E. Zitzler, and K. Deb, Eds., 2003, vol. 2632, pp. 519–533.
[5] 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, pp. 117–132, Apr. 2003.
[6] M. Fleischer, The measure of Pareto optima: Applications to multi-objective metaheuristics Inst. Syst. Res., Univ. Maryland, College Park, MD, Tech. Rep. ISR TR 2002–32, 2002.
[7] E. Zitzler, "Hypervolume metric calculation." 2001 [Online]. Available: ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/hypervol.c
[8] J. Knowles, "Local-search and hybrid evolutionary algorithms for Pareto optimization," Ph.D., Univ. Reading, Reading, U.K., 2002.
[9] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Trans. Evol. Comput.*, vol. 10, pp. 29–38, Feb. 2006.
[10] L. While, L. Bradstreet, L. Barone, and P. Hingston, "Heuristics for optimizing the calculation of hypervolume for multi-objective optimization problems," in *Proc. Congr. Evol. Comput.*, B. McKay, Ed., 2005, pp. 2225–2232.

[11] N. Beume and G. Rudolph, "Faster s-metric calculation by considering dominated hypervolume as Klee's measure problem," Univ. Dortmund, Dortmund, Germany, Tech. Rep. CI 216/06, 2006.
[12] M. H. Overmars and C.-K. Yap, "New upper bounds in Klee's measure problem," *SIAM J. Computing*, vol. 20, no. 6, pp. 1034–1045, Dec. 1991.
[13] C. M. Fonseca, L. Paquete, and M. López-Ibáñez, "An improved dimension-sweep algorithm for the hypervolume indicator," in *Proc. Congr. Evol. Comput.*, C. L. P. Chen, Ed., 2006, pp. 3973–3979.
[14] S. Huband, P. Hingston, L. While, and L. Barone, "An evolution strategy with probabilistic mutation for multi-objective optimization," in *Proc. Congr. Evol. Comput.*, H. Abbass and B. Verma, Eds., 2003, pp. 2284–2291.
[15] J. Knowles, D. Corne, and M. Fleischer, "Bounded archiving using the Lebesgue measure," in *Proc. Congr. Evol. Comput.*, H. Abbass and B. Verma, Eds., 2003, pp. 2490–2497.
[16] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Proc. Parallel Problem Solving from Nature VIII*, X. Yao, E. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervos, J. A. Bullinaria, J. Rowe, P. Tino, A. Kaban, and H. Schwefel, Eds., 2004, vol. 3242, pp. 832–842.
[17] M. Emmerich, N. Beume, and B. Noujoks, "An EMO algorithm using the hypervolume measure as selection criterion," in *Proc. Evol. Multi-Objective Opt.*, M. Emmerich, N. Beume, B. Naujoks, C. A. C. Coello, A. H. Aguirre, and E. Zitzler, Eds., 2005, vol. 3410, pp. 62–76.
[18] , T. Bäck, Ed., *Handbook of Evolutionary Computation*. Oxford, U.K.: Oxford Univ. Press, 1997.
[19] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proc. Congr. Evol. Comput.*, R. Eberhart, Ed., 2002, pp. 825–830.
[20] L. Bradstreet, L. Barone, and L. While, "Maximizing hypervolume for selection in multi-objective evolutionary algorithms," in *Proc. Congr. Evol. Comput.*, C. L. P. Chen, Ed., 2006, pp. 1744–1751.
[21] L. Bradstreet, L. Barone, and L. While, "Incrementally maximizing hypervolume for selection in multi-objective evolutionary algorithms," in *Proc. Congr. Evol. Comput.*, A. Tay, Ed., 2007, pp. 3203–3210.
[22] Hypervolume Test Data "Walking fish group", 2006. [Online]. Available: http://wfg.csse.uwa.edu.au/Hypervolume

Lucas Bradstreet (S'06) received a BCM degree in computer and mathematical sciences from the University of Western Australia, Nedlands, in 2004. He is currently working towards the Ph.D. degree at the School of Computer Science and Software Engineering, University of Western Australia.

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

Lyndon While (M'01–SM'04) received the B.Sc.(Eng) and Ph.D. degrees from the Imperial College of Science and Technology, London, U.K., in 1985 and 1988, respectively.

He is currently a Senior Lecturer in the School of Computer Science and Software Engineering, University of Western Australia. His research interests include evolutionary algorithms, multiobjective optimization, and the semantics and implementation of functional programming languages.

Luigi Barone (M'01) received the B.Sc. and Ph.D. degrees from the University of Western Australia, Nedlands, in 1994 and 2004, respectively.

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

Paper 3 Errata (Refereed)

L. Bradstreet, L. While, and L. Barone. Incremental Hypervolume by Slicing Objectives: a Correction to the Pseudo-code. *IEEE Transactions on Evolutionary Computation*, 13(5):1193–1193, November 2009

Incremental Hypervolume by Slicing Objectives: a Correction to the Pseudo-code

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

***Abstract*— This letter describes a correction to the pseudo-code of the IHSO algorithm published in *A Fast Incremental Hypervolume Algorithm* in IEEE TEC in December 2008.**

***Index Terms*— Multi-objective optimisation, evolutionary computation, diversity, performance metrics, hypervolume.**

BRADSTREET *et al*. [1] describes the IHSO (Incremental HSO) algorithm for calculating and comparing the exclusive hypervolumes [2]–[4] of a set of nD points. IHSO is a variant of the HSO (Hypervolume by Slicing Objectives) algorithm [5]–[7], fine-tuned for minimising the calculations required for incremental use and for finding the point in a set that contributes the least exclusive hypervolume. IHSO is the fastest exact incremental hypervolume algorithm known to date.

However, the pseudo-code for IHSO given in Fig. 4 of [1] contains an error that might prevent its easy implementation. A corrected version of the pseudo-code is given in Fig. 1 opposite. The corrected part is the body of the final loop of the first procedure `ihso`: the three relevant lines are marked with a `*`. For completeness, Fig. 1 also contains some procedures that were given in Fig. 2 of [1].

We apologise for this error and for any difficulties that it may have caused. We are grateful to Louis-Claude Canon of Nancy University and Kiyoharu Tagawa of Kinki University for bringing the error to our attention, and to the latter for suggesting a correction.

REFERENCES

[1] L. Bradstreet, L. While, and L. Barone, "A fast incremental hypervolume algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 714–723, December 2008.

[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] 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.

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

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

[7] 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.

The authors are with The University of Western Australia.

```
  ihso (z, ps):
    pl = sort ps worsening in Objective 1
    s = {(1, pl)}
    for k = 1 to n-1
      s' = {}
      for each (x, ql) in s
        for each (x', ql') in slice (z, ql, k)
          add (x * x', ql') into s'
      s = s'
    vol = 0
    for each (x, ql) in s
*     if ql = []
*       then vol = vol + x * |z(n) - refPoint[n]|
*       else vol = vol + x * |z(n) - head(ql)[n]|
    return vol

  slice (z, pl, k):
    ql = []
    s = {}
    v = z[k]
    dominated = false
    while pl /= [] and not dominated
      p = head (pl)
      pl = tail (pl)
      if v beats p[k]
        then add (|v - p[k]|, ql) into s
             v = p[k]
      ql = insert (p, k+1, ql)
      dominated = dominates (p, z, k+1)
    if not dominated
      then add (|v - refPoint[k]|, ql) into s
    return s

  insert (p, k, pl):
    ql = []
    while pl /= [] && head (pl)[k] beats p[k]
      append head (pl) to ql
      pl = tail (pl)
    append p to ql
    while pl /= []
      if not (dominates (p, head (pl), k))
        then append head (pl) to ql
      pl = tail (pl)
    return ql

  dominates (p, q, k):
    d = True
    while d && k <= n
      d = not (q[k] beats p[k])
      k = k + 1
    return d
```

Fig. 1. Pseudo-code for IHSO.

Paper 4 (Refereed)

L. Bradstreet, L. While, and L. Barone. Incrementally Maximising Hypervolume for Selection in Multi-Objective Evolutionary Algorithms. In *2007 IEEE Congress on Evolutionary Computation (CEC'2007)*, pages 3203–3210, Singapore, September 2007. IEEE Press

Incrementally Maximising Hypervolume for Selection in Multi-objective Evolutionary Algorithms

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

***Abstract*— Several multi-objective evolutionary algorithms compare the hypervolumes of different sets of points during their operation, usually for selection or archiving purposes. The basic requirement is to choose a subset of a front such that the hypervolume of that subset is maximised. We describe and evaluate three new algorithms based on incremental calculations of hypervolume using the new Incremental Hypervolume by Slicing Objectives (IHSO) algorithm: two greedy algorithms that respectively add or remove one point at a time from a front, and a local search that assesses entire subsets. Empirical evidence shows that using IHSO, the greedy algorithms are generally able to out-perform the local search and perform substantially better than previously published algorithms.**

I. Introduction

Multi-objective problems are common in the optimisation field and much of the current research into evolutionary algorithms revolves around the theory and practice of multi-objective optimisation. Many evolutionary algorithms have been created in order to solve these difficult problems, e.g. SPEA [1], NSGA-II [2]. However, despite recent work, there remains the question of how to compare the performance of different multi-objective optimisation (MOO) algorithms. The result of a multi-objective algorithm is a set of solutions, a non-dominated front, representing a trade-off between objectives. A popular metric used to compare these fronts is the hypervolume measure (otherwise known as the S-metric [3] or the Lebesgue measure [4]). Hypervolume is the n-dimensional space that is "contained" by the points (solutions) in a front. A front with a larger hypervolume is likely to present a better set of trade-offs to a user than one with a smaller hypervolume.

While most research into hypervolume has revolved around its use as a metric, recent research has seen it applied during the operation of Multi-objective Evolutionary Algorithms (MOEAs). An example of a technique used in this way is Deb's [2] NSGA-II crowdedness comparison operator, used to select solutions within a front to minimise solutions crowding in each objective. Knowles *et al.* [5] introduced the use of hypervolume during optimisation. They describe a bounded archiving algorithm that retains the 'best' solutions found throughout optimisation. As this archive is potentially of unbounded size, they apply hypervolume to determine the solutions that maximise the coverage of the solution set. During optimisation, when a new solution is added to this bounded archive, the solution that contributes the least hypervolume is removed.

Emmerich *et al.* [6] extend this idea to selection in a 2 dimensional MOEA in their optimiser SMS-EMOA. Rather than applying bounded archiving, in selection they apply hypervolume to remove the solution that contributes the least hypervolume from the worst ranked front. By doing so, they reduce the size of the population in order to provide room for the next generation. This concept has been extended by Naujoks *et al.* [7] for 3 objective MOEAs.

While this idea has merit and has achieved excellent experimental results, it has some limitations. One such limitation is the use of a steady state MOEA. Emmerich *et al.* use a steady state MOEA because finding the optimal composition that maximises the hypervolume of a reduced front is a difficult problem and the effectiveness of heuristic approaches is unknown. However, in some cases a steady state MOEA may not achieve as high quality results as other MOEAs. Furthermore, these MOEAs do not apply hypervolume selection for more than 3 objectives.

The main contribution of this paper is a comparison between front selection algorithms based on hypervolume, using recent incremental hypervolume algorithms designed for this task. A previous paper by Bradstreet *et al.* [8] compared two front reduction algorithms: a greedy approach and a local search algorithm that each attempt to maximise the hypervolume of reduced fronts. New incremental hypervolume techniques that calculate a point's exclusive hypervolume, due to Bradstreet *et al.* [9], are easily incorporated into these algorithms and warrants a new comparison between front selection algorithms.

The use of an incremental hypervolume approach, in lieu of the metric method used in [8], reduces the cost of point removal using the greedy method. We introduce two greedy front selection techniques using an incremental approach. These approaches iteratively reduce or increase the size of the selected front until it reaches the desired size. Use of an incremental hypervolume algorithm also benefits the local search front selection technique. It does so by reducing the cost of evaluating new front selections that result from changing relatively few points. The presence of these new techniques merits revisiting the comparison between these approaches in order to determine which technique should be recommended for general use. We find that new greedy selection techniques provide effective for front selection

The authors are with the School of Computer Science & Software Engineering, The University of Western Australia, Crawley 6009, Australia (phone: +61864881944; fax: +61864881089; email: {lucas, lyndon, luigi}@csse.uwa.edu.au).

when retaining a various proportions of a front.

The rest of this paper is structured as follows. Section II defines the concepts and notation used in multi-objective optimisation. Section III defines the front reduction techniques that we have created and why we may use them. Section IV gives empirical data, for a variety of front types, demonstrating situations where each front reduction technique is valuable and why. Section V concludes the paper and outlines future work.

II. Fundamentals

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 $\bar{x}$ and $\bar{y}$, $\bar{x}$ *dominates* $\bar{y}$ iff $\bar{x}$ is at least as good as $\bar{y}$ in all objectives, and better in at least one. A vector $\bar{x}$ is *non-dominated* with respect to a set of solutions X iff there is no vector in X that dominates $\bar{x}$. 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 $\bar{x}$ is *Pareto optimal* iff $\bar{x}$ 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.

Given a set X of solutions returned by an algorithm, the question arises how good the set X is, i.e. how well it approximates the Pareto optimal set. One metric used for comparing sets of solutions is to measure the *hypervolume* of each set. The hypervolume of X is the size of the space that is dominated by the solutions in X. 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 fronts being compared.) If a set X has a greater hypervolume than a set X, then X is taken to be a better set of solutions than X.

Knowles [5] applies hypervolume in order to recast multi-objective problems as single objective problems with the single goal of maximising the hypervolume of a set of solutions, bounded in size. As this set is of finite size, when adding a solution to this set another must often be removed to make room. This is achieved by removing the solution contributing the minimal 'exclusive hypervolume' to the front. The exclusive hypervolume contribution, Δs, of a solution, p, to a front, f, can be defined as $\Delta s = Hypervolume(f \cup \{p\}) - Hypervolume(f)$. Following this definition, one realises that over the course of operation the hypervolume of the set of archived solutions is maximised.

Emmerich *et al.* [6] apply this technique to selection in a MOEA. Rather than maintaining an archive of solutions, hypervolume is used to determine which solutions should be allowed to reproduce and which should be thrown away. When a front is too large to be included in the population during selection, hypervolume is used to reduce its size. The solution that contributes the smallest hypervolume to the worst ranked front is removed from the front with the aim of maximising the hypervolume of the population during the lifetime of the MOEA.

III. Point Removal Techniques

When using hypervolume as a selection metric, we wish to reduce the size of non-dominated fronts. Ideally we wish to find a front composition that maximises the hypervolume of the reduced front. However, in order to guarantee an optimally composed front, it is necessary to calculate all $\binom{n}{m}$ subsets of the front, where n is the size of the front and m is the size of the reduced front. Calculating all possible subset combinations is extremely expensive if even a small number of solutions are to be removed. In order to work around this problem, Emmerich *et al.* use a steady state MOEA, where only one point is removed at a time. However, use of steady state MOEAs may not always be desirable. Therefore alternative approaches to using hypervolume for selection should be researched.

We present three new techniques: a search algorithm, and two greedy algorithms that either add or remove a single solution until a front of the desired size is created. These algorithms are each useful in different front selection scenarios, and we thus aim to identify these so the algorithms can be incorporated effectively into an MOEA that uses hypervolume for selection.

These algorithms have been adapted from [8] to use an incremental hypervolume algorithm, IHSO, that provides performance benefits for front selection. The Incremental Hypervolume by Slicing Objectives (IHSO) algorithm, due to Bradstreet *et al.* [9] is able to quickly calculate the hypervolume exclusively contributed by a point, p, relative to a set of points.

A. Greedy Front Reduction Algorithms

Greedy front reduction schemes remove or add a single point at a time without regard for whether that choice will lead to a worse overall front selection. Bradstreet *et al.* [8] previously used the HSO hypervolume algorithm for metric calculations to reduce the number of points. This was done by calculating the hypervolume of the entire front missing a single point. The point chosen is the one that maximises the hypervolume of the front when it is removed. Unfortunately, this method is very slow as HSO is not designed for this kind of use and will make many unnecessary calculations. We present two alternative greedy techniques that quickly select a subset of the points using IHSO.

```
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. 1. Outline of the best-first queuing scheme in IHSO*.

1) Greedy Front Reduction Algorithm using IHSO: In order to remove the worst point from a front we use the following scheme described in Bradstreet *et al.* [9].

This algorithm in Fig. 1 is applied iteratively until the front is pruned to the desired size. The algorithm is considered greedy as the removal of a single point in each iteration may not result in an optimal front selection when additional points are removed.

After each iteration, in which a single point is eliminated from the front, the hypervolume contributions calculated in the last iteration will be discarded. The removal of the point means that contributions calculated thus far may be incorrect due to the removed point sharing contributions exclusively with other points.

However, it is possible to significantly optimise performance in cases where one ore more point contributions are not affected. These contributions can be retained for the next iteration. We use a simple scheme that improves run-time considerably. This scheme examines the slices calculated by IHSO thus far. If the removed point has not been used in any slice yet it is removed from all slices. In this case, the point has not influenced the contribution of the point and the hypervolume is retained. If the removed point has already been examined (i.e. may have had an effect on the contributed hypervolume so far) the point's contributed hypervolume is set to zero and hypervolume contribution calculation is restarted.

2) Greedy Front Addition Algorithm using IHSO: An alternative approach builds up the the reduced size front from an empty front.

Given a non-dominated front, S, with n solutions, retaining m solutions.

```
SS = S
RS = {}
Repeat until RS contains m solutions
    Find the solution s in SS that contributed
      the maximum hypervolume to RS
    Move s from SS to RS
return RS
```

This approach should have performance advantages when only a small proportion of the front is retained. Under this algorithm, the hypervolume contributions of each point will need to be calculated entirely, however each evaluation should be computationally cheap when the front is small. When a small proportion of the front is removed this algorithm will be very slow as it does not benefit from best first search and will require more iterations than the reduction method.

3) Greedy Algorithm Discussion: Unfortunately, this scheme may be computationally expensive in cases in which a large number solutions are removed. In order to remove a single solution from a front containing m solutions, m hypervolume contribution calculations are required.

Thus, in order to remove m solutions from the front using algorithm 2, up to $\sum_{i=1}^{p} m - 1 + 1$ IHSO evaluations are required. Furthermore, for large fronts the cost of each hypervolume evaluation may be expensive due to the exponential complexity of current hypervolume algorithms, as proved by While *et al.* [10], [11]. As a result of this computational complexity, a local search may be more desirable for large fronts or many objectives.

To implement the greedy algorithms above, we apply the Incremental Hypervolume By Slicing Objectives (IHSO) algorithm using the rank heuristic and best first search described by Bradstreet *et al.* [9]. Use of this algorithm and search strategy allows us to determine the worst point from large fronts very quickly. While IHSO still has exponential complexity, these techniques improve run-time substantially compared to the naive greedy scheme that uses HSO [8].

B. Front Reduction by Local Search

In contrast to the greedy front reduction method, Bradstreet *et al.* propose a local search algorithm [8]. Local search achieves good performance in cases where a large proportion of solutions are eliminated from a front. Rather than perform numerous expensive hypervolume evaluations, using the entire front in early calculations, the local search technique performs a larger number of computationally cheaper hypervolume evaluations on reduced size fronts. As a result, even though the local search requires more individual hypervolume evaluations, this method can be faster than the front reduction method if a large number of points is removed (see Bradstreet et al. [8]).

Additionally, the local search scheme has the following benefits:

- One can bound the time taken by the algorithm and still achieve useful results.
- Use may result in higher quality fronts than the greedy approach. This is due to cases where the greedy algorithm removes solutions that may be desirable in the reduced front but that contribute little in the full front. This effect is due to regions covered by other points that do not exist in the optimal reduced front.

1) HV Local Search: Bradstreet *et al.*'s [8] hypervolume front reduction local search algorithm operates as follows:

1) Generate initial front composition.
2) Perturb the front (resulting in a modified front of the same size).
3) Accept the new front composition as the current front if it has a better hypervolume.
4) Repeat steps 2-4 until run-time constraint is exceeded.

We compared other search methodologies, such as Simulated Annealing and Evolutionary Algorithms, and found

that they did not achieve major improvements compared to a local search in the tested time frames.

The local search algorithm adapted from [8] uses IHSO to evaluate the hypervolume of new front compositions that differ from the previous front by a small number of points. Using IHSO to remove and add a small proportion of points will be faster than completely recalculating the entire hypervolume of the front using HSO.

IV. Experiments

We evaluated the front reduction techniques on two distinct fronts from the DTLZ [12] test suite: the spherical front and the discontinuous front. For each front, we mathematically generated a representative set of 10,000 points from the (known) Pareto optimal set: then to generate a front of size m, we sampled this set randomly. The linear front from DTLZ gives similar results to the spherical front, and the degenerate front gives anomalous results as its hypervolumes can be calculated in polynomial time [13].

We also tested the techniques on randomly generated fronts. For these fronts we generated sets of m mutually non-dominating points in n objectives simply 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 x to S only if $\bar{x} \cup S$ would be mutually non-dominating. We kept adding points until $|S| = m$.

Reference points are determined by the method used by Naujoks *et al.* [7], where the reference point takes the worst value in each dimension in the front shifted by 1.0 in each dimension. Discontinuous and spherical fronts were cast as minimisation problems, while random fronts were cast as a maximisation problem.

Firstly the greedy front reduction algorithm was run on a diverse range of front types (varying objective functions, numbers of objectives, numbers of points) to determine an acceptable front composition containing half the individuals in a front. For each of these front types, we ran the greedy algorithm and local search on five different fronts. The local search was allowed to run for twice as long as the greedy front reduction method. We gave the local search additional run-time in order to determine whether it can achieve better selections when computation time is not a high priority. The local search was run five times on each front and these results averaged.

Results relating to hypervolumes are shown as a ratio of the hypervolume of the selected front and the hypervolume of the full front. This ratio is intended to give an assessment of how closely the reduced front covers to space covered by the full front.

All timings were performed on a dedicated Apple iMac with Intel 2.16GHz Core 2 Duo processor and 3GB of RAM, running Mac OS X 10.4.8. All algorithms were implemented in C and compiled using gcc 4.0.1 using the -O3 compiler flag.

A. Experiments retaining 80% of the front

Tables I, II and III demonstrate that the performance of the greedy reduction algorithm using IHSO is superior to local search when 80% of the front is retained. For all front types the local search is unable to achieve front selections with hypervolumes as good as the greedy reduction approach when given comparable time.

The greedy addition method results in front selections that are equivalent to the reduction approach. However, it takes much longer to do so in a majority of cases (28 times longer in the case of random 5d). This partially results from the fact that greedy addition does not benefit the use of best first search and must calculate every contribution in its entirety. Furthermore, it requires more iterations than the reduction approach which only has to remove a small proportion of the front when most of the front is retained. Consequently, we do not recommend the addition method in cases where a large proportion of the front is retained.

B. Experiments retaining 50% of the front

Tables IV, V and VI demonstrate that the performance of the greedy algorithms using IHSO is superior to local search when half of the front is removed. Unlike the previous results in Bradstreet *et al.* [8], local search is unable to provide a competitive solution when given twice as much computation time. While the local search gains performance improvements as a result of the incorporation of IHSO, the greedy algorithms used in this paper are many times faster than the naive greedy approach.

In most cases both greedy algorithms find front selections that have equivalent hypervolumes. While they differ for several fronts, each approach does have situations in which they obtain better selections. As far as run-time is concerned, the reduction approach has superior performance for random fronts. However, the addition approach has superior performance for spherical fronts. For discontinuous data, Each is competitive for different front sizes and numbers of objectives. We believe that the performance of each approach is very dependent on the type of data. This effect is possibly increased by the overall effect of heuristics and the best-first search used by the reduction approach. For example, as the heuristics do not perform well on spherical fronts, the fact that the addition approach performs hypervolume evaluations on small fronts is a great advantage. In contrast, for random data, where best-first search and heuristics improve IHSO's performance greatly (see Bradstreet et al. [9]), the reduction approach performs very well.

C. Experiments retaining 20% of the front

Bradstreet et al. [8] previously showed that local search is very competitive with other techniques when a majority of the front is removed. However, that paper only compares the naive greedy reduction approach to local search. The new greedy reduction approach, using IHSO, compares more favourably to the local search, and finds better front selections in the majority of cases (see Tables VII-IX).

TABLE I

SPHERICAL DATA: TIMINGS AND HYPERVOLUME COMPARISON FOR GREEDY ALGORITHMS AND LOCAL SEARCH, RETAINING 80% OF THE FRONT.

		Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
# obj	# pts	Time (s)	HV ratio	Time (s)	HV ratio	Equivalent time	Twice time	HV ratio
5	200	6.93	0.999983	52.54	0.999983	0.999520	0.999785	0.991415
6	120	8.00	0.999963	56.95	0.999962	0.997878	0.998907	0.990084
7	80	28.78	0.999869	48.08	0.999869	0.998483	0.999405	0.988738
8	60	32.24	0.999867	62.90	0.999867	0.997379	0.998599	0.984024
9	40	13.73	0.999748	10.92	0.999748	0.995567	0.997278	0.979463

TABLE II

DISCONTINUOUS DATA: TIMINGS AND HYPERVOLUME COMPARISON FOR GREEDY ALGORITHMS AND LOCAL SEARCH, RETAINING 80% OF THE FRONT.

		Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
# obj	# pts	Time (s)	HV ratio	Time (s)	HV ratio	Equivalent time	Twice time	HV ratio
5	300	5.54	0.999866	97.43	0.999866	0.997916	0.998854	0.984556
6	150	4.58	0.999175	90.01	0.999175	0.989416	0.993200	0.972714
7	80	2.03	0.997814	30.05	0.997814	0.936433	0.982418	0.956477
8	45	1.01	0.994320	6.17	0.994320	0.958196	0.972569	0.937695
9	40	3.12	0.989953	12.38	0.989953	0.882196	0.969089	0.928322

TABLE III

RANDOM DATA: TIMINGS AND HYPERVOLUME COMPARISON FOR GREEDY ALGORITHMS AND LOCAL SEARCH, RETAINING 80% OF THE FRONT.

		Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
# obj	# pts	Time (s)	HV ratio	Time (s)	HV ratio	Equivalent time	Twice time	HV ratio
5	200	0.17	0.999901	5.05	0.999901	0.976755	0.988297	0.950789
6	100	0.16	0.999781	1.96	0.999781	0.946305	0.973519	0.923245
7	70	0.37	0.999795	2.03	0.999795	0.906626	0.968238	0.895972
8	50	0.76	0.999649	2.17	0.999649	0.830657	0.928200	0.868234
9	40	1.09	0.999644	2.62	0.999644	0.689511	0.916466	0.861498

TABLE IV

SPHERICAL DATA: TIMINGS AND HYPERVOLUME COMPARISON FOR GREEDY ALGORITHMS AND LOCAL SEARCH, RETAINING 50% OF THE FRONT.

		Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
# obj	# pts	Time (s)	HV ratio	Time (s)	HV ratio	Equivalent time	Twice time	HV ratio
5	200	19.89	0.999664	17.37	0.999664	0.999474	0.999574	0.973390
6	120	20.53	0.999307	15.04	0.999302	0.998291	0.998843	0.966192
7	80	60.48	0.998518	7.11	0.998515	0.997897	0.998231	0.960005
8	60	70.17	0.997905	5.81	0.997897	0.996700	0.997493	0.947553
9	40	23.69	0.996841	1.06	0.996771	0.995490	0.996382	0.932470

TABLE V

DISCONTINUOUS DATA: TIMINGS AND HYPERVOLUME COMPARISON FOR GREEDY ALGORITHMS AND LOCAL SEARCH, RETAINING 50% OF THE FRONT.

		Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
# obj	# pts	Time (s)	HV ratio	Time (s)	HV ratio	Equivalent time	Twice time	HV ratio
5	300	14.15	0.997429	32.49	0.997421	0.994376	0.996100	0.947237
6	150	9.59	0.988378	19.07	0.988371	0.972854	0.980517	0.909812
7	80	3.74	0.975898	4.83	0.975903	0.939631	0.956333	0.863012
8	45	1.41	0.944419	0.69	0.944419	0.907395	0.925562	0.800250
9	40	4.09	0.937734	1.02	0.937800	0.906253	0.924329	0.771831

TABLE VI

RANDOM DATA: TIMINGS AND HYPERVOLUME COMPARISON FOR GREEDY ALGORITHMS AND LOCAL SEARCH, RETAINING 50% OF THE FRONT.

		Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
# obj	# pts	Time (s)	HV ratio	Time (s)	HV ratio	Equivalent time	Twice time	HV ratio
5	200	0.57	0.995742	2.62	0.995742	0.941199	0.971844	0.845316
6	100	0.28	0.993940	0.87	0.993940	0.873711	0.930382	0.767120
7	70	0.26	0.993271	0.67	0.993271	0.839151	0.893817	0.720372
8	50	0.21	0.990495	0.48	0.990495	0.720431	0.862161	0.653691
9	40	0.28	0.983387	0.44	0.983387	0.744885	0.860619	0.621136

TABLE VII

SPHERICAL DATA: TIMINGS AND HYPERVOLUME COMPARISON FOR GREEDY ALGORITHMS AND LOCAL SEARCH, RETAINING 20% OF THE FRONT.

		Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
# obj	# pts	Time (s)	HV ratio	Time (s)	HV ratio	Equivalent time	Twice time	HV ratio
5	200	21.36	0.996655	0.58	0.996654	0.996581	0.996635	0.920630
6	120	23.18	0.991394	0.26	0.991358	0.991308	0.991381	0.899922
7	80	61.78	0.985356	0.08	0.985331	0.985415	0.985441	0.883446
8	60	72.00	0.979722	0.05	0.979637	0.979780	0.979780	0.857163
9	40	23.51	0.968414	0.01	0.968315	0.968565	0.968565	0.816699

TABLE VIII

DISCONTINUOUS DATA: TIMINGS AND HYPERVOLUME COMPARISON FOR GREEDY ALGORITHMS AND LOCAL SEARCH, RETAINING 20% OF THE FRONT.

		Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
# obj	# pts	Time (s)	HV ratio	Time (s)	HV ratio	Equivalent time	Twice time	HV ratio
5	300	16.98	0.976898	1.95	0.976738	0.975666	0.976295	0.859237
6	150	11.01	0.925353	0.59	0.924692	0.921766	0.923866	0.768118
7	80	4.11	0.876777	0.09	0.876301	0.875112	0.876464	0.665712
8	45	1.54	0.770361	0.01	0.769618	0.769024	0.769943	0.556917
9	40	4.27	0.756236	0.01	0.756325	0.756673	0.756773	0.501841

TABLE IX

RANDOM DATA: TIMINGS AND HYPERVOLUME COMPARISON FOR GREEDY ALGORITHMS AND LOCAL SEARCH, RETAINING 20% OF THE FRONT.

		Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
# obj	# pts	Time (s)	HV ratio	Time (s)	HV ratio	Equivalent time	Twice time	HV ratio
5	200	0.94	0.946104	0.32	0.946104	0.933146	0.940626	0.636039
6	100	0.40	0.937245	0.06	0.937219	0.914438	0.927559	0.502814
7	70	0.31	0.914051	0.03	0.914081	0.886285	0.898815	0.405025
8	50	0.21	0.895494	0.01	0.895494	0.866009	0.886169	0.342082
9	40	0.26	0.859679	0.01	0.859679	0.848643	0.854726	0.295887

Furthermore, the new greedy addition approach is particularly suited for front selection retaining small proportions of the front. In these cases, only a small number of iterations of the addition algorithm are required – many less than the reduction approach. Additionally, IHSO evaluations will be performed on much smaller fronts, which is a boon given IHSO's exponential complexity. Experiments show that greedy addition reduces run-time, compared to reduction, by 63.6% (Random 5d 200pts) to 99.96% (Spherical 9d 40pts). Each approach finds fronts with similar hypervolumes, but for some cases the reduction approach does result in slightly better front selections than addition. Addition achieves fronts with better hypervolumes in a minority of cases.

That the reduction approach finds better front selections may be due to the fact that during the first few iterations of the addition algorithm, IHSO has little context when adding points to the front. For example, the selection of the first point added will be due to a hypervolume contribution that is the outright hypervolume of that point.

In contrast, cases where large numbers of points are removed can also result in the greedy reduction approach performing badly. One possible cause is due to the result of removing points during the early iterations of the algorithm. As these point contributions are based on the current selected

front they are based on fronts containing points that are removed later. As only 20% of the front is retained the current state of the selected front is very dissimilar to the final front selection.

Ignoring these flaws in the greedy algorithms, the greedy addition approach is able to generate good selections much quicker than the reduction approach. As the addition approach only has to add 20% of the points, and all of the IHSO calculations operate on small fronts it is able to quickly select a front. In the case of spherical data in 9d, the reduction approach takes 2500 times longer than the addition approach and the produced fronts have the same hypervolume. Furthermore, the results achieved when the local search is given equivalent computation time to reduction approach are always worse than either greedy algorithm.

D. Experimental Discussion

The results achieved by the new greedy approach are generally favourable in comparison to the local search in terms of the run-time and hypervolumes of selected fronts. In the case where 80% and 50% of the front is retained, the new greedy reduction approach gives superior results to the local search. Similarly, when 20% of the front is retained the greedy addition approach is recommended.

Bear in mind that small changes in hypervolume may relate to substantially improved coverage of the front. For example, while a reduced front with only one solution may have a hypervolume that is very close to that of the original front, this is not a desirable result and we still wish for the best possible coverage of the Pareto front. Given a particular choice of reference point, front selections may result in hypervolumes that are similar to many decimal places, however differ greatly in terms of quality. As such, whether one algorithm outperforms another algorithm in terms of hypervolume is perhaps more important than the magnitude of difference.

We have included figures in Tables I-IX that show the hypervolume averages obtained for 100 random front selections. These are intended to be used as a gauge of the importance of any difference in hypervolumes obtained by the different techniques. For example, if a random selection obtains a hypervolume ratio close to 1 then a small difference between the hypervolumes obtained by different techniques may be very important. For example, random front selections for Spherical 5d 200pts in Table I average a ratio of 0.991415, and therefore the 0.00046 difference between the greedy approaches and local search is probably significant.

V. Conclusions and Future Work

The overall conclusion given in the comparison between local search and greedy selection in Bradstreet *et al.* [8] was that local search is competitive in most cases except for when a majority of a front is retained. Results for 50% and 80% of front removal showed that local search is a candidate for use in a hypervolume selection based MOEA.

However, the use of IHSO and other new techniques in this paper leads to different conclusions. Local search is now competitive with the greedy techniques only when a majority of the front is removed. In all other cases the new greedy reduction algorithm finds good front selections quicker and more reliably. Furthermore, when comparing the greedy addition method to the local search algorithm, the local search is unable to give comparable front selections in equivalent computation time for the 20% retained fronts. As the addition method gives similar front selections to the reduction method in a greatly reduced time, we believe that a combination of the reduction method and addition method provide an excellent overall solution for reducing a front to any size within an MOEA.

The introduction of recent work by Beume and Rudolph [14] that uses the Overmars and Yap algorithm [15] for the Klee's Measure Problem to quickly perform hypervolume metric calculations means that this work will require revision in the future. Future work will look at adapting this algorithm to calculate exclusive hypervolume contributions. Such an algorithm may provide improvements to both the greedy approaches and local search.

References

[1] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," in *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, Eds., Athens, Greece, 2002, pp. 95–100.

[2] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," Indian Institute of Technology, Kanpur, India, KanGAL report 200001, 2000.

[3] E. Zitzler, "Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications," Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[4] M. Laumanns, E. Zitzler, and L. Thiele, "A Unified Model for Multi-Objective Evolutionary Algorithms with Elitism," in *2000 Congress on Evolutionary Computation*, vol. 1. Piscataway, New Jersey: IEEE Service Center, July 2000, pp. 46–53.

[5] J. D. Knowles, D. W. Corne, and M. Fleischer, "Bounded Archiving using the Lebesgue Measure," in *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, vol. 4. Canberra, Australia: IEEE Press, December 2003, pp. 2490–2497.

[6] M. Emmerich, N. Beume, and B. Naujoks, "An EMO algorithm using the hypervolume measure as selection criterion," in *Proc. Evolutionary Multi-Criterion Optimization: Third Int'l Conference (EMO 2005)*, ser. Lecture Notes in Computer Science, C. A. C. Coello, A. H. Aguirre, and E. Zitzler, Eds., vol. 3410. Berlin: Springer, 2005, pp. 62–76.

[7] B. Naujoks, N. Beume, and M. Emmerich, "Multi-objective optimisation using S-metric selection: Application to three-dimensional solution spaces," in *Proc. 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, B. McKay *et al.*, Eds., vol. 2. Piscataway NJ: IEEE Press, 2005, pp. 1282–1289.

[8] L. Bradstreet, L. Barone, and L. While, "Maximising hypervolume for selection in multi-objective evolutionary algorithms," in *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B.-T. Zhang, C. A. C. Coello, and T. P. Runarsson, Eds. Vancouver, BC, Canada: IEEE Press, 16-21 July 2006, pp. 1744–1751. [Online]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=11108

[9] L. Bradstreet, L. While, and L. Barone, "A fast incremental hypervolume algorithm," The University of Western Australia, Technical Report UWA-CSSE-07-001, 2007. [Online]. Available: http://web.csse.uwa.edu.au/__data/page/58425/UWA-CSSE-07-001.pdf

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

[11] R. L. While, "A new analysis of the LebMeasure algorithm for calculating hypervolume." in *EMO*, ser. Lecture Notes in Computer Science, C. A. C. Coello, A. H. Aguirre, and E. Zitzler, Eds., vol. 3410. Springer, 2005, pp. 326–340.

[12] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable Multi-Objective Optimization Test Problems," in *Congress on Evolutionary Computation (CEC'2002)*, vol. 1. Piscataway, New Jersey: IEEE Service Center, May 2002, pp. 825–830.

[13] L. While, L. Bradstreet, L. Barone, and P. Hingston, "Heuristics for optimising the calculation of hypervolume for multi-objective optimisation problems," in *Proceedings of 2005 Congress on Evolutionary Computation (CEC'2005)*, 2005.

[14] N. Beume and G. Rudolph, "Faster s-metric calculation by considering dominated hypervolume as klee's measure problem," University of Dortmund, Technical Report CI 216/06, 2006. [Online]. Available: http://sfbci.uni-dortmund.de/Publications/Reference/Downloads/21606.pdf

[15] M. H. Overmars and C.-K. Yap, "New upper bounds in klee's measure problem (extended abstract)," in *IEEE Symposium on Foundations of Computer Science*, 1988, pp. 550–556. [Online]. Available: citeseer.ist.psu.edu/article/overmars88new.html

Paper 5 (Refereed)

L. Bradstreet, L. Barone, and L. While. Updating Exclusive Hypervolume Contributions Cheaply. In *2009 IEEE Congress on Evolutionary Computation (CEC'2009)*, pages 538–544, Thronheim, Norway, May 2009. IEEE Press

Updating Exclusive Hypervolume Contributions Cheaply

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

***Abstract*— Several multi-objective evolutionary algorithms compare the hypervolumes of different sets of points during their operation, usually for selection or archiving purposes. The basic requirement is to choose a subset of a front such that the hypervolume of that subset is maximised. We describe a technique that improves the performance of hypervolume contribution based front selection schemes. This technique improves performance by allowing the update of hypervolume contributions after the addition or removal of a point, where these contributions would previously require full recalculation. Empirical evidence shows that this technique reduces runtime by up 72-99% when compared to the cost of full contribution recalculation on DTLZ and random fronts.**

I. Introduction

Multi-objective problems are common in the optimisation field and much of the current research into evolutionary algorithms revolves around the theory and practice of multi-objective optimisation. Many evolutionary algorithms have been created in order to solve these difficult problems, e.g. SPEA [1], NSGA-II [2]. However, despite recent work, there remains the question of how to compare the performance of different multi-objective optimisation (MOO) algorithms. The result of a multi-objective algorithm is a set of solutions, a non-dominated front, representing a trade-off between objectives. A popular metric used to compare these fronts is the hypervolume measure (otherwise known as the S-metric [3][4] or the Lebesgue measure [5]). Hypervolume is the n-dimensional space that is "contained" by the points (solutions) in a front. A front with a larger hypervolume is likely to present a better set of trade-offs to a user than one with a smaller hypervolume.

While most research into hypervolume has revolved around its use as a metric, recent research has seen it applied during the operation of Multi-objective Evolutionary Algorithms (MOEAs). An example of a technique used in this way is Deb's [2] NSGA-II crowdedness comparison operator, used to select solutions within a front to minimise solutions crowding in each objective. Knowles *et al.* [6] introduced the use of hypervolume during optimisation. They describe a bounded archiving algorithm that retains the 'best' solutions found throughout optimisation. As this archive is potentially of unbounded size, they apply hypervolume to determine the solutions that maximise the coverage of the solution set. During optimisation, when a new solution is added to this bounded archive, the solution that contributes the least hypervolume is removed.

Emmerich *et al.* [7] extend this idea to selection in a MOEA in their optimiser SMS-EMOA, and demonstrate an example 2-D optimiser. Rather than applying bounded archiving, in selection they apply hypervolume to remove the solution that contributes the least hypervolume from the worst ranked front. By doing so, they reduce the size of the population in order to provide room for the next generation. This concept has been extended by Naujoks *et al.* [8] for 3 objective MOEAs.

Bradstreet *et al.* [9][10] have looked into front reduction techniques using both greedy front addition and reduction techniques as well as a local search techniques. The greedy techniques have a lot of merit, however still suffer from IHSO's exponential complexity in the number of objectives. Thus, techniques to improve their performance are hugely beneficial. One area in which their performance could be improved lies in reusing previously calculated results, which are even though they may only differ slightly. As these contributions become outdated due to changes in the front, a technique to quickly update their hypervolume to reflect the addition or removal of a point would be hugely beneficial to these selection algorithms.

The principal contribution of this paper is a a technique to update these contributions and then compare the performance of this technique to a method that requires the recalculation of these hypervolume contributions. This technique utilises current incremental hypervolume algorithms to calculate the contribution of a generated point that represents the change in contribution. We find that performance of this technique greatly improves performance of front reduction algorithms using incremental hypervolume when compared to algorithms that recalculate contributions in full.

The rest of this paper is structured as follows. Section II defines the concepts and notation used in multi-objective optimisation. Section III introduces existing front reduction techniques that benefit from the work introduced in this paper. Section IV introduces a simple method to find the change in point contributions resulting from the removal or addition of a point. Section V gives empirical data, for a variety of front types, demonstrating performance improvements resulting from the contribution update technique described within this paper.

II. Fundamentals

In a multi-objective optimisation problem, we aim to find the set of optimal trade-off solutions known as the Pareto

The authors are with the School of Computer Science & Software Engineering, The University of Western Australia, Crawley 6009, Australia (phone: +61864881944; fax: +61864881089; email: {lucas, lyndon, luigi}@csse.uwa.edu.au).

optimal set. Pareto optimality is defined with respect to the concept of non-domination between points in objective space. Given two objective vectors $\bar{x}$ and $\bar{y}$, $\bar{x}$ *dominates* $\bar{y}$ iff $\bar{x}$ is at least as good as $\bar{y}$ in all objectives, and better in at least one. A vector $\bar{x}$ is *non-dominated* with respect to a set of solutions X iff there is no vector in X that dominates $\bar{x}$. 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 approximation front or set*.

A vector $\bar{x}$ is *Pareto optimal* iff $\bar{x}$ 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.

Given a set X of solutions returned by an algorithm, the question arises how good the set X is, i.e. how well it approximates the Pareto optimal set. One metric used for comparing sets of solutions is to measure the *hypervolume* of each set. The hypervolume of X is the size of the space that is dominated by the solutions in X. The hypervolume of a set is measured relative to a reference point, usually the anti-optimal point or "bounding" 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 fronts being compared.) If a set X has a greater hypervolume than a set X, then X is taken to be a better set of solutions than X.

Knowles [6] applies hypervolume in order to recast multi-objective problems as single objective problems with the single goal of maximising the hypervolume of a set of solutions, bounded in size. As this set is of finite size, when adding a solution to this set another must often be removed to make room. This is achieved by removing the solution contributing the minimal 'exclusive hypervolume' to the front. The exclusive hypervolume contribution, s, of a solution, p, to a front, f, can be defined as $s = Hypervolume(f \cup \{p\}) - Hypervolume(f)$. Following this definition, one realises that over the course of operation the hypervolume of the set of archived solutions is maximised.

Emmerich *et al.* [7] apply this technique to selection in a MOEA. Rather than maintaining an archive of solutions, hypervolume is used to determine which solutions should be discarded. When a front is too large to be included in the population during selection, hypervolume is used to reduce its size. The solution that contributes the smallest hypervolume to the worst ranked front is removed from the front with the aim of maximising the hypervolume of the population during the lifetime of the MOEA.

III. Point Removal Techniques

When using hypervolume as a selection metric, we wish to reduce the size of non-dominated fronts. Ideally we wish to find a front composition that maximises the hypervolume of the reduced front. However, in order to guarantee an optimally composed front, it is necessary to calculate all $\binom{n}{m}$ subsets of the front, where n is the size of the front and m is the size of the reduced front. Calculating all possible subset combinations is extremely expensive if even a small number of solutions are to be removed.

We have previously experimented with various methods for composing the front that work around this problem [9][10]. To help motivate how performance can be improved, we will first introduce greedy front composition algorithms that either add or remove a single point at a time until the front is the desired size, using exclusive hypervolume contributions as a selection method.

These selection algorithms were researched by Bradstreet *et al.* [9][10] and use an incremental hypervolume algorithm, IHSO, that high performance hypervolume contribution calculations. The Incremental Hypervolume by Slicing Objectives (IHSO) algorithm, due to Bradstreet *et al.* [11] is able to quickly calculate the hypervolume exclusively contributed by a point, p, relative to a set of points. All of these algorithms are based around greedy front selection methods.

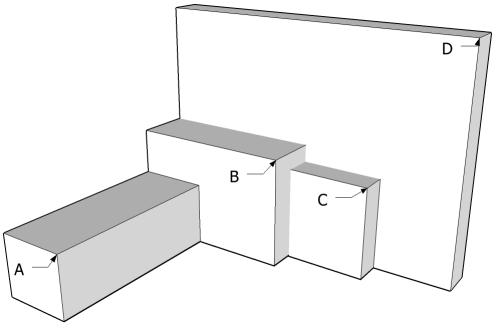


Fig. 1. Example front containing points $A = (11, 2, 2)$, $B = (5, 4, 4)$, $C = (4, 6, 3)$, $D = (2, 8, 8)$.

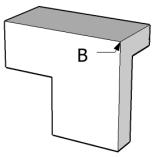


Fig. 2. Exclusive contribution of B to example front shown in Figure 1.

The greedy front selection techniques use IHSO to calculate the exclusive hypervolume contributions of every point contained in a front. An example of a front is shown in Figure 1 and contains the points $A = (11, 2, 2)$, $B = (5, 4, 4)$, $C = (4, 6, 3)$, $D = (2, 8, 8)$. In this example, the exclusive contribution for each point is calculated so that the least contributing point can be removed to reduce the size of the front. An example shape, representing the exclusive contribution of B, is shown in Figure 2. The exclusive contributions of points contained in this front relative to reference point $(0, 0, 0)$, under a maximisation problem, are $cA = (6*2*2) = 24, cB = (4*4-2*2)*1+(4*4-3*4)*2 = 20, cC = (2*3)*2 = 12, cD = (8*8-4*4-2*3)*2 = 84$.

Greedy front reduction schemes remove or add a single point at a time without regard for whether that choice will lead to a worse overall front selection. These schemes have been evaluated by Bradstreet *et al.* [9][10]. These algorithms find the point that contributes the most or least to the hypervolume of the front and add or remove that point. This process is repeated until the front is the desired size.

In the example in Figure 1 and using the greedy point reduction method, the point C would be removed from the front as it contributes the least hypervolume. The contributions of A, B and D would then require recalculation.

A. Greedy Front Reduction Algorithm using full IHSO calculations

We will first introduce the simplest front composition algorithm. In every iteration exclusive contributions are calculated in full and the point with the smallest hypervolume is removed. The algorithm is considered greedy as the removal of a single point in each iteration may not result in an optimal front selection when additional points are removed. This algorithm is shown in Figure 3.

```
Repeat until front contains m solutions
  Calculate the entire contribution of each point
  Delete the point with the smallest from the front
```

Fig. 3. Outline of the greedy reduction algorithm using full IHSO calculations.

After each iteration, resulting in the elimination of a point, the hypervolume contributions calculated in the last iteration will be discarded. The removal of the point means that partially calculated contributions may be incorrect due to the removed point sharing contributions exclusively with other points.

B. Greedy Front Reduction Algorithm using IHSO and priority queue

The front reduction algorithm using IHSO and a priority queue as described in Bradstreet *et al.* [11] is an improvement on the previous scheme. It allows only as much of each point's contribution to be calculated as is necessary to prove that it does not have the smallest contribution to the front. This algorithm is shown in Figure 4.

```
Repeat until front contains m solutions
 Partially evaluate each point
 Identify the smallest point
 while the smallest point is not completed
   Evaluate the smallest point a bit more
   Identify the new smallest point
 remove the smallest point
```

Fig. 4. Outline of the reduction algorithm using the best-first queuing scheme in IHSO.

A priority queue is used to quickly identify the smallest point. Another slice is then calculated at a particular depth and the priority queue is updated and then returns the current smallest partially calculated hypervolume. As with the previous scheme, the contributions from slices calculated in the previous iteration must be discarded before beginning the process to remove the next point. The number of partial calculations made in each iteration depend on the chosen granularity. In the case of [11], a single slice at a the chosen depth (granularity) would be fully calculated.

C. Greedy Front Addition Algorithm using IHSO

An alternative approach builds up the the reduced size front from an empty front.

Given a non-dominated front, S, with n solutions, retaining m solutions.

```
SS = S
RS = {}
Repeat until RS contains m solutions
    Find the solution s in SS that contributes
      the maximum hypervolume to RS
    Move s from SS to RS
return RS
```

Fig. 5. Outline of the greedy reduction algorithm using full IHSO calculations.

This algorithm, outlined in Figure 5, has performance advantages when only a small proportion of the front is retained. Under this algorithm, the hypervolume contributions of each point are calculated entirely, however each evaluation should be computationally cheap when the front is small. When a small proportion of the front is removed this algorithm will be very slow as it does not benefit from best first search and will require more iterations than the reduction method. As with the other algorithms, contributions will be recalculated in full for each iteration.

D. Greedy Algorithm Discussion

While these techniques are very effective, it becomes evident that in most cases previously calculated hypervolume contributions are wasted. They are discarded to ensure accuracy of the contributions with the aim of best maximising the hypervolume of the overall front. The priority queue algorithm suffers from this the least, as the entire contribution is not calculated in each iteration. However, this waste should be proportion to the cost to calculate the volumes initially. This waste occurs despite the fact that contributions may not change much from iteration to iteration.

The algorithms tested in [9][10] used a simple scheme to improve this case. When the contribution of a point has not been affected by a point removal the contributions are retained for the next iteration. This scheme works by examining the slices calculated by IHSO thus far. If the removed point has not been used in any slice yet, it is removed from all slices. In this case, the point has not influenced the point's removal and the previously calculated contribution is retained. If the removed point has already been examined (i.e. may have had an effect on the contributed hypervolume so far) the point's contributed hypervolume is set to zero and hypervolume contribution calculation is restarted. However, we shall show that it is possible to greatly improve on this scheme.

IV. Δ Hypervolume Contribution Calculation

It is clear that a fast technique that updates the previously calculated contributions would be beneficial. Such a technique would reflect any change in a point's contribution, due to an addition or removal of a point. If the technique were quicker than recalculation, the performance of the point selection techniques could be improved. In the case of the priority queue, only updates to IHSO slices already calculated would be required. Similarly, the greedy front addition technique would benefit from a similar approach. Instead the addition of a point would bring about a decrease in the contribution of the list of points yet to be added to the front.

We have formulated a simple method for these update calculations that utilises the existing hypervolume contribution calculation functions (IHSO). As we know, the contribution of a point, P, is the space exclusively dominated by that point and no other point. Therefore, when point R is removed any increase in the contribution to P will be about due to the space exclusively dominated by either P and R, relative to the front containing neither P or R. Therefore, this difference in the contribution of the point P after the removal of R can be viewed as the contribution of the point resulting from the intersection of the space dominated by both P and R. This point can be generated by taking the worst value from either P and R in each objective. Where S is the front and P the contributing point, and PR is the point generated using P and the removed point R, this can be generalised to the form:

$$c(P, S) = c(P, S \cup \{R\}) + c(PR, S)$$

Under this definition, the calculation of a contribution of a point is still required. However, this contributing point should be quicker to calculate, as it is generated from a combination of the previous worst point (which is usually less complex to calculate as it covers a smaller space), and the point being updated. PR should cover a less complex space than P, as it only contains regions covered by P, but not all.

Using the example in Figure 1, we wish to update the contributions of A, B and D after the removal of point C. For example, in order to update the contribution of C, we generate a new point BC which contains the least values of B and C in each objective. Under the above form, the updated volume of B is:

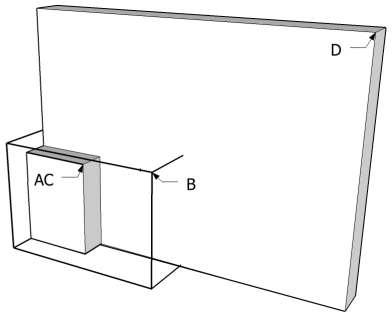


Fig. 6. Front used to calculate the contribution of AC, $(4, 2, 2)$. Contributing volume of AC reflects the change in A's contribution resulting from the removal of point C. As AC is dominated by point B, AC's contribution is 0.

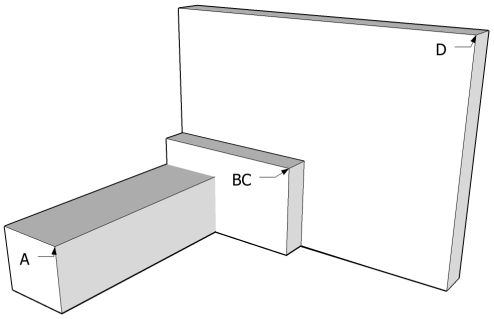


Fig. 7. Front used to calculate the contribution of BC, $(4, 4, 3)$. Contributing volume of BC reflects the change in B's contribution resulting from the removal of point C.

$$c(B, \{A, D\}) = c(B, \{A, C, D\}) + c(BC, \{A, D\})$$

The front used to determine the contribution of BC is shown in Figure 7. The contribution of the point BC is $c(BC, \{A, D\}) = (4 * 3 - 2 * 2) * 2 = 16$. Using the contribution, $c(B, \{A, C, D\}) = 20$ calculated earlier, the updated contribution of B is therefore $c(B, \{A, D\}) = 20 + 16 = 36$ The contributions of A and D would then be updated in a similar manner, using the fronts shown in Figure 6 and 8. Note that when updating A, AC has no contribution as it is dominated by B, so the volume of A remains the same.

The technique can also be used within the priority queue approach, however care must be taken so that contribution updates can be correctly performed on partially calculated slices at each depth of suspended IHSO calculations.

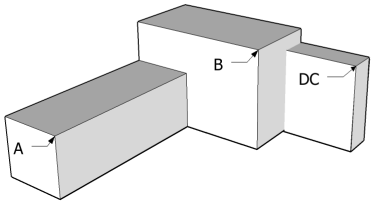


Fig. 8. Front used to calculate the contribution of DC, $(2, 6, 3)$. Contributing volume of DC reflects the change in D's contribution resulting from the removal of point C.

V. Experiments

A. Performance Comparison of Recalculation vs Δ Update

In order to compare the performance of the update technique, we decided to evaluate the cost of full contribution recalculation, compared to the cost of merely updating contributions. We have done so by comparing the greedy reduction algorithm, described in Section III-B, to a modified version that updates contributions. In both algorithms the contributions of every point is first calculated in its entirety and a point is removed. In the full calculation version, each remaining point's exclusive contribution is recalculated for each iteration. In update mode, these contributions are updated instead. For these experiments, we chose to reduce the front size by half. The cost of calculating the first set of contributions is not included in the comparison as these experiments are intended to compare the cost of an update to full recalculation.

B. Experimental setup

We evaluated the front reduction techniques on several distinct fronts from the DTLZ [12] test suite: spherical, discontinuous, linear and degenerate fronts. For each front, we mathematically generated a representative set of 10,000 points from the (known) Pareto optimal set: then to generate a front of size m, we sampled this set randomly.

We also tested the techniques on randomly generated fronts. For these fronts we generated sets of m mutually non-dominating points in n objectives simply 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 x to S only if $\bar{x} \cup S$ would be mutually non-dominating. We kept adding points until $|S| = m$.

Reference points are determined by the method used by Naujoks *et al.* [8], where the reference point takes the worst value in each dimension in the front shifted by 1.0 in each dimension. All fronts were cast as as minimisation problems.

All timings were performed on an Intel 2.4GHz Core 2 Duo processor with 2GB of RAM, running Ubuntu Linux 7.04. All algorithms were implemented in C and compiled with *gcc -O3 -funroll-all-loops* (version 4.1.2). Calculations operated only on a single core as we have not written the algorithm in a parallel manner although improvements in this area would be relatively simple to achieve as each point requires an independent contribution calculation. The code used for these experiments is available at `http://www.wfg.csse.uwa.edu.au`.

The front reduction algorithm was run on a diverse range of front types (varying objective functions, numbers of objectives, numbers of points). Front sizes were increased until the full contribution calculation algorithm took a total of 20 seconds to remove half of the points from a given front. Timings were averaged over 20 runs.

C. Experimental Results & Discussion

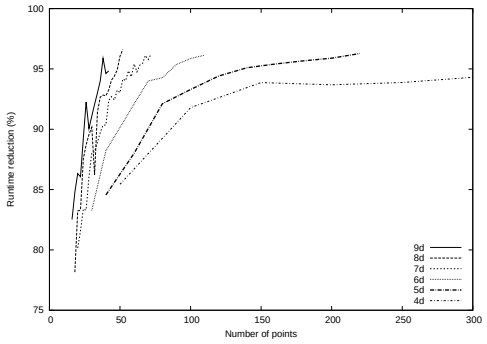


Fig. 9. Spherical data: Runtime reduction achieved by update technique when removing 50% of the front.

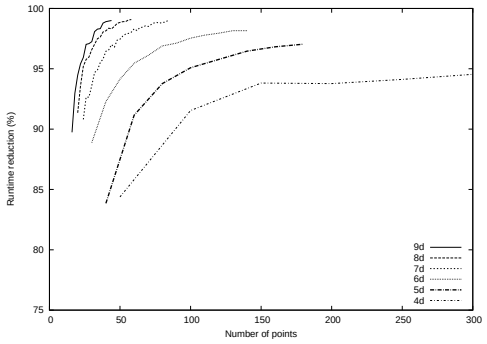


Fig. 10. Random data: Runtime reduction achieved by update technique when removing 50% of the front.

Figures 9-13 show the average computation saved using the update approach for the different front front types. Observe that computation savings improve as the complexity of the hypervolume calculations increase i.e. increasing either the size of front or number of objectives. Savings from

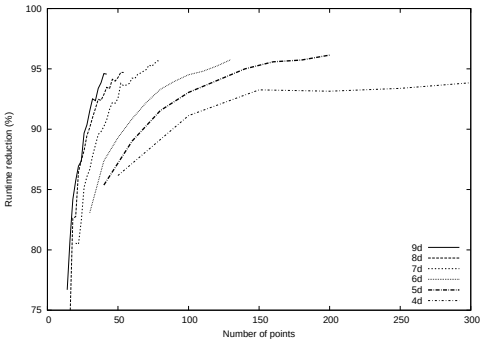


Fig. 11. Discontinuous data: Runtime reduction achieved by update technique when removing 50% of the front.

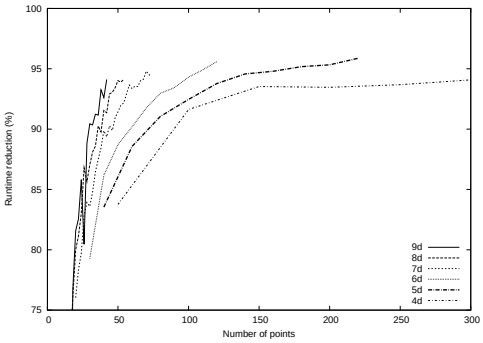


Fig. 12. Linear data: Runtime reduction achieved by update technique when removing 50% of the front.

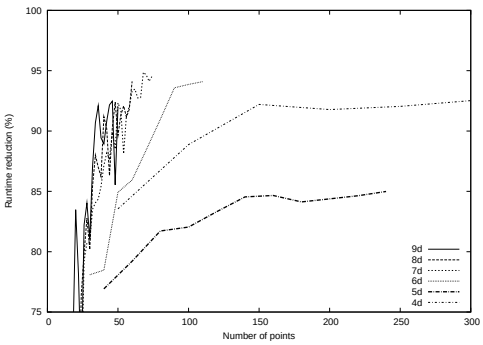


Fig. 13. Degenerate data: Runtime reduction achieved by update technique when removing 50% of the front.

72% to 99% can be observed. Results show that random data appears to benefit the most from the delta approach, saving up to 99% of the run time compared to recalculation, compared to around 95% for the other fronts.

This could be due to the least contributing point being less complex to compute than the average point. This hypothesis seems reflected by the results found in Bradstreet *et al.* [11] which showed that random data requires the least computation to find the least contributing point under the best first search based priority queue approach. This would likely result from the contribution of the smallest point requiring little computation and only minimal calculation of other contributions being necessary to determine the least contributing point. For the update technique, if the point with the least contribution does not exclusively cover a large (and therefore usually complex) space, the combination of that point and each of the remaining points (e.g. Figure 6) is likely to cover an even smaller region of space that is calculated even quicker than the contribution of the removed point which contributed least in the last iteration.

TABLE I
SPHERICAL DATA: TIME REQUIRED TO REMOVE 50% OF THE FRONT USING UPDATES OR RECALCULATION.

# obj	# pts	Delta time (s)	Full time (s)
4	300	1.68	29.64
5	220	1.10	29.74
6	110	0.82	21.25
7	72	0.72	19.17
8	52	0.63	18.57
9	42	1.03	20.07

TABLE II
RANDOM DATA: TIME REQUIRED TO REMOVE 50% OF THE FRONT USING UPDATES OR RECALCULATION.

# obj	# pts	Delta time (s)	Full time (s)
4	300	1.83	33.63
5	180	0.63	21.50
6	140	0.43	23.81
7	84	0.195	19.53
8	58	0.171	18.63
9	44	0.220	21.98

TABLE III
DISCONTINUOUS DATA: TIME REQUIRED TO REMOVE 50% OF THE FRONT USING UPDATES OR RECALCULATION.

# obj	# pts	Delta time (s)	Full time (s)
4	300	1.88	30.59
5	200	1.13	29.32
6	130	1.04	24.66
7	80	0.94	21.70
8	54	1.00	19.02
9	42	1.14	21.14

TABLE IV
LINEAR DATA: TIME REQUIRED TO REMOVE 50% OF THE FRONT USING UPDATES OR RECALCULATION.

# obj	# pts	Delta time (s)	Full time (s)
4	300	1.76	29.67
5	220	1.23	29.84
6	120	1.19	27.10
7	74	1.09	19.34
8	54	1.25	21.14
9	42	1.20	20.31

TABLE V
DEGENERATE DATA: TIME REQUIRED TO REMOVE 50% OF THE FRONT USING UPDATES OR RECALCULATION. TIMES DO NOT INCLUDE INITIAL CONTRIBUTION CALCULATION REQUIRED TO REMOVE THE FIRST POINT.

# obj	# pts	Delta time (s)	Full time (s)
4	300	1.60	21.43
5	240	3.08	20.52
6	110	1.35	22.95
7	74	1.05	19.26
8	60	1.37	23.38
9	50	2.17	27.18

Differences in the maximum time required for recalculation compared to update can be assessed in Tables I-V. Typical time savings range from 22 seconds compared to 0.22 seconds to 20.5 seconds compared to 3.08 seconds. These timings do not include the initial full contribution calculation required to remove the first point.

These experiments were intended to determine an approximate cost of full calculation to the cost of contribution update, and to show that updating hypervolume contributions to reflect changes in a front can be done much quicker than the full recalculation of those contributions. While the update technique will work correctly for front reduction using IHSO and the priority queue approach or the front addition technique, we have only evaluated recalculation of full contributions to updates. We believe that improvements achievable with this technique should be similar using the other techniques. In the priority queue case, less contributions are required for each point removal, and therefore the calculations necessary to update these contributions should also be minimised, as only the already calculated slices will require updating. Normally, these slices would require recalculation, and therefore the number of updated slices should be in proportion to the number of slices that would have required recalculation. Large savings are also expected when the technique is used on the front addition algorithm.

VI. CONCLUSIONS AND FUTURE WORK

We have introduced a simple technique to update hypervolume contributions quickly to reflect the addition or removal of points to a front. This technique has been shown to improve the performance of front selection methods that calculate contributions entirely in order to remove a point. Typical improvements range from 75% runtime reduction to 99% under the range of data types, front sizes and number of objectives tested.

In the future, we will implement this technique for use in the front addition algorithm and under the front reduction technique using the best first search priority queue. We expect the performance of each of these techniques to be improved greatly as we hope contribution recalculation in each of these methods has a high cost as with the greedy full contribution method that we have evaluated.

REFERENCES

[1] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," in *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, Eds., Athens, Greece, 2002, pp. 95–100.

[2] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," Indian Institute of Technology, Kanpur, India, KanGAL report 200001, 2000.

[3] E. Zitzler, "Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications," Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[4] N. Beume, C. M. Fonseca, M. L.-I. nez, L. Paquete, and J. Vahrenhold, "On the complexity of computing the hypervolume indicator," University of Dortmund, Technical Report CI 235/07, 2007. [Online]. Available: http://sfbci.uni-dortmund.de/Publications/Reference/Downloads/23507.pdf

[5] M. Laumanns, E. Zitzler, and L. Thiele, "A Unified Model for Multi-Objective Evolutionary Algorithms with Elitism," in *2000 Congress on Evolutionary Computation*, vol. 1. Piscataway, New Jersey: IEEE Service Center, July 2000, pp. 46–53.

[6] J. D. Knowles, D. W. Corne, and M. Fleischer, "Bounded Archiving using the Lebesgue Measure," in *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, vol. 4. Canberra, Australia: IEEE Press, December 2003, pp. 2490–2497.

[7] M. Emmerich, N. Beume, and B. Naujoks, "An EMO algorithm using the hypervolume measure as selection criterion," in *Proc. Evolutionary Multi-Criterion Optimization: Third Int'l Conference (EMO 2005)*, ser. Lecture Notes in Computer Science, C. A. C. Coello, A. H. Aguirre, and E. Zitzler, Eds., vol. 3410. Berlin: Springer, 2005, pp. 62–76.

[8] B. Naujoks, N. Beume, and M. Emmerich, "Multi-objective optimisation using S-metric selection: Application to three-dimensional solution spaces," in *Proc. 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, B. McKay *et al.*, Eds., vol. 2. Piscataway NJ: IEEE Press, 2005, pp. 1282–1289.

[9] L. Bradstreet, L. Barone, and L. While, "Maximising hypervolume for selection in multi-objective evolutionary algorithms," in *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B.-T. Zhang, C. A. C. Coello, and T. P. Runarsson, Eds. Vancouver, BC, Canada: IEEE Press, 16-21 July 2006, pp. 1744–1751. [Online]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=11108

[10] L. Bradstreet, L. While, and L. Barone, "Incrementally maximising hypervolume for selection in multi-objective evolutionary algorithms," *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 3203–3210, Sept. 2007.

[11] L. Bradstreet, L. While, and L. Barone, "A fast incremental hypervolume algorithm," *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 6, pp. 714–723, Dec. 2008.

[12] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable Multi-Objective Optimization Test Problems," in *Congress on Evolutionary Computation (CEC'2002)*, vol. 1. Piscataway, New Jersey: IEEE Service Center, May 2002, pp. 825–830.

Paper 6 (Refereed)

L. Bradstreet, L. While, and L. Barone. A Fast Many-objective Hypervolume Algorithm using Iterated Incremental Calculations. In *2010 IEEE Congress on Evolutionary Computation (CEC'2010)*, pages 179–186, Barcelona, Spain, July 2010. IEEE Press

A Fast Many-objective Hypervolume Algorithm using Iterated Incremental Calculations

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

***Abstract*— Three fast algorithms have been proposed for calculating hypervolume exactly: the Hypervolume by Slicing Objectives algorithm (HSO) optimised with heuristics designed to improve the average case; an adaptation of the Overmars and Yap algorithm for solving the Klee's measure problem; and a recent algorithm by Fonseca *et al.* We propose a fourth algorithm IIHSO based largely on the Incremental HSO algorithm, a version of HSO adapted to calculate the exclusive hypervolume contribution of a point to a front. We give a comprehensive analysis and performance comparison of these algorithms, and conclude that IIHSO outperforms the others on most important and representative data in higher numbers of objectives.**

I. Introduction

THE hypervolume indicator [1] (also known as the S-metric [2] or the Lebesgue measure [3], [4]) is a popular metric for comparing the performance of multi-objective evolutionary algorithms (MOEAs). 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 the spread of the solutions across objective space. Hypervolume also has nicer mathematical properties than other metrics: it was the first unary metric that detects when a set of solutions X is not worse than another set X' [5], and it is maximised if and only if the set of solutions contains all Pareto optimal points [6]. Wagner et al. [7] have shown that hypervolume based selection is valuable for many-objective selection based optimisation. However, hypervolume is sensitive to the relative scaling of the objectives, and to the presence or absence of extremal points in a front. While fast approximation algorithms exist, such as Everson et al. [8] and Bader et al. [9], this paper will focus on exact hypervolume calculation algorithms.

Three fast algorithms have been proposed for calculating hypervolume exactly. The Hypervolume by Slicing Objectives algorithm (HSO) [10]–[12] processes the objectives in a front, rather than the points. HSO divides the nD-hypervolume to be measured into separate $(n-1)$D-slices through the values in one of the objectives, then it calculates the hypervolume of each slice and sums these values to derive the total. HSO's worst-case complexity is $O(m^{n-1})$ [12] (m being the number of points in the front), but While *et al.* have described good heuristics for re-ordering objectives [13] that typically deliver better performance.

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

In addition, algorithms from the computational geometry field have recently been applied to hypervolume calculation. Beume and Rudolph adapt the Overmars and Yap algorithm [14] for solving the Klee's measure problem to instead calculate the hypervolume of a front [15]. We refer to their adapted algorithm as HOY. HOY recursively splits the space into smaller and smaller regions, until every region is either completely covered by the hypervolume, or it is covered in all objectives bar one, in which case the size of the hypervolume in the region can be calculated quickly by a simple inclusion-exclusion algorithm. HOY's region splitting procedure gives it a worst-case complexity of $O(m \log m + m^{n/2})$, far better than HSO.

Paquete et al. [16] use a geometry-inspired algorithm to calculate the maxima of a point set in 3D which has shown to be optimal by Beume et al. [17]. Fonseca et al. [18] combine this 3D algorithm, a specialised data structure and other improvements to provide a performance boost to a HSO-like dimension sweep to calculate hypervolume in n-dimensions. We refer to Fonseca et al.'s algorithm as FPL. Fonseca *et al.* state that the worst-case complexity of FPL is $O(m^{n-2} \log m)$ [18].

Another recent development is the Incremental HSO algorithm [19] (IHSO). This is an adaptation of HSO to calculate the exclusive hypervolume contribution of a point to a front. IHSO is especially useful where hypervolume is used in-line within a MOEA, either for diversity calculations [20], or for archiving purposes [21], or in selection [22], [23]. However, as we will demonstrate, IHSO can applied iteratively to create a new method for hypervolume metric calculations.

This paper makes four principal contributions.

- We describe a new algorithm IIHSO (*Iterated IHSO*) for calculating hypervolume exactly. IIHSO applies IHSO iteratively, starting with an empty set and adding one point at a time until the entire front has been processed. The idea of calculating hypervolume as a summation of exclusive hypervolumes was introduced by LebMeasure [4].
  IIHSO also incorporates ideas from FPL, principally its linked data structure for optimising dominance calculations.
- We describe heuristics designed to optimise the typical performance of IIHSO, mainly for choosing a good

order for adding the points to the set and a good order for processing the objectives.
- We show that while HOY has by far the best worst-case complexity of these algorithms, its performance varies little with different fronts or front types, or with different objective order, and thus it may be difficult to improve HOY's performance significantly via use of heuristics or manipulation of the data.
- We give a comprehensive performance comparison of IIHSO, HOY, and FPL with heuristics on a range of front types and sizes, and we show that IIHSO outperforms the other algorithms on a number of multi-objective test problems.

The rest of this paper is structured as follows. Section II defines the concepts and notation used in multi-objective optimisation and throughout this paper. Section III describes our new algorithm IIHSO and associated optimisations. Section IV describes our experiments and analyses the results, and Section V concludes the paper.

II. Definitions

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 vectors 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 usually suffices.

Given a set X of solutions returned by an algorithm, the question arises how good the set X is, i.e. how well it approximates the Pareto optimal set. One metric used for comparing sets of solutions is to measure the *hypervolume* of each set. The hypervolume of X is the total size of the space that is dominated by the solutions in X. 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 fronts being compared.) If a set X has a greater hypervolume than a set X', then X is taken to be a better set of solutions than X'.

Precise definitions of these terms can be found in [24].

A. Incremental HSO

IHSO is an adaptation of HSO created to calculate the exclusive hypervolume contribution of a point to a front [19]. That is, given a point p and a front S, IHSO calculates $ExcHyp(p, S)$, defined as

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

This is important when hypervolume is used in-line within a MOEA, either for diversity calculations, or for archiving purposes, or in selection. Our new algorithm IIHSO (described in Section III) uses IHSO in metric calculations too.

The operation of IHSO is substantially similar to HSO, differing in two important ways:

- It disregards all slices "higher" than p: p will not contribute to any slice above itself in the current objective, therefore the hypervolumes of these slices need not be calculated.
- It disregards some slices "lower" than p: if p is dominated by a point q in S in the objectives after the current one, then p will not contribute to any slice containing q (or any point that dominates q), and the hypervolumes of these slices need not be calculated.

As with HSO, the typical performance of IHSO can be improved substantially by selecting a good order in which to process the objectives. Bradstreet *et al.* [19] evaluate a range of objective-reordering heuristics for IHSO: they conclude that the best heuristic is one which processes first the objective in which a point is best, so that the point is likely to be dominated early in the process and the number of iterations is minimised.

III. The IIHSO Algorithm

The IIHSO (*Iterated IHSO*) algorithm uses IHSO to calculate the hypervolume of a front by breaking it up into a series of exclusive hypervolumes, one for each point in the front. Specifically, it calculates the exclusive hypervolume of each point p relative to the points which have better values than p in the first objective. Thus, we calculate the exclusive hypervolume of the best point relative to the empty set, then the exclusive hypervolume of the second point relative to the first, then the exclusive hypervolume of the third point relative to the first two, etc. The sum of these is the collective hypervolume of the whole front.

Fig. 1 shows the operation of IIHSO. The hypervolume is calculated as the sum of the exclusive hypervolumes of each point p relative to the set of points with better values than p in the first objective. Fig. 2 gives pseudo-code for IIHSO.

Fig. 1 shows the significance of sorting the points in the first objective: IIHSO calculates only the bottom slice in each $n - 1$-objective hypervolume. However, as usual with HSO-based algorithms, we can employ heuristics to select the first objective to be processed in order to improve the typical performance of IIHSO. Additionally, within each application of IHSO, we can employ heuristics to re-order the objectives for that calculation. Note that this second heuristic can exploit the properties of the point whose exclusive

hypervolume is being calculated, so the objectives can be re-ordered differently for each application of IHSO.

We have found through experiments that the following combination of heuristics works well for IIHSO:

- we use the MWW heuristic of HSO [13] to select the first objective;
- we use the Rank heuristic of IHSO [19] to process first the objectives in which a point is best, so that is more likely to be dominated early. Rank is applied only if five or more objectives remain: in our experience, with fewer objectives, the performance improvement does not recoup the cost of the heuristic calculation.

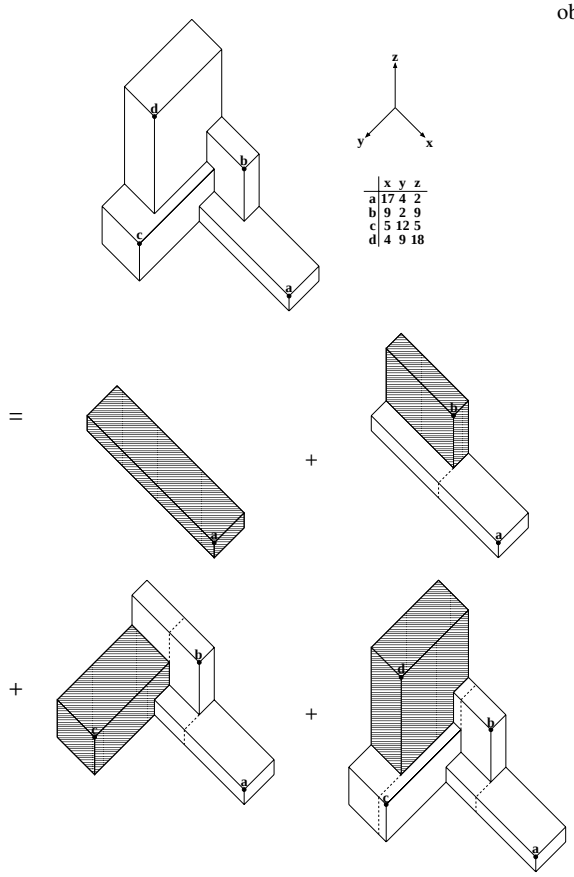


Fig. 1. Using IIHSO to calculate the hypervolume of $\{a, b, c, d\}$, equal to $ExcHyp(a, \{\}) + ExcHyp(b, \{a\}) + ExcHyp(c, \{a, b\}) + ExcHyp(d, \{a, b, c\})$. Marks indicate points. Note that in each two-objective volume, we need to calculate only the bottom slice.

IIHSO's other principal improvement from HSO is a change to the way that IHSO itself operates. We denote as IHSO† our modification to IHSO that uses a new linked data structure based on the data structure used in FPL [18]. The structure contains a linked list for each objective, with the points in each linked list sorted by their values in that objective. IHSO† operates differently to IHSO in four ways:

```
IIHSO(ps):
  Order the objectives by the MWW heuristic
  pl = sort ps by worsening value of Objective 1
  vol = 0
  ql = []
  for i = 1 to m
    vol = vol + ihso (pl[i], ql)
    ql  = insert (pl[i], ql)
  return vol

insert (p, pl) adds p into pl, deleting any points
from pl which are dominated by p in Objectives 2..n
```

Fig. 2. Pseudo-code for IIHSO.

- When a point is deleted from a front, it is deleted from the linked lists for unprocessed objectives, but it is retained within the data structure so that it can be reinserted quickly later.
- When a previously-deleted point is reinserted into the linked lists, existing points are not checked to see whether they are dominated by the new point. Instead, whenever IHSO† calculates a slice which has a hypervolume equal to the previous slice, the newly added point is flagged as dominated at the current depth so that it can be skipped in lower objectives. These points are reset (marked as non-dominated) at the conclusion of processing the slice.
- Unlike in HSO or FPL, where the addition of a non-dominated point always increases the hypervolume of a front, the addition of a point in IHSO does not necessarily change the exclusive hypervolume of the contributing point. The exclusive hypervolume of the new point may be disjoint from that of the contributing point. Therefore, the domination-flag check discussed in the previous bullet point also improves performance by skipping points that do not influence the contributing point's hypervolume, even when they are non-dominated within the front.
- IHSO† can re-order the objectives upon the addition of any point to a slice. Previously the front would already be sorted in the next objective, and additional points would be inserted into the correct position in the front. As fronts are now ordered in all objectives, there is no additional cost to switching the next objective to be processed on-the-fly.

IV. Experimental Comparison

In this section we compare the performance of the three hypervolume algorithms: IIHSO, HOY and FPL.

We used the HOY algorithm implementation from [25] and the FPL implementation from [26]. Source code for IIHSO is available from [27]. All timings were performed on an Intel 2.4GHz Core 2 Duo processor with 2GB of RAM, running Ubuntu Linux 7.04. All algorithms were compiled with *gcc/g++ -O3 -march=nocona -funroll-all-loops* (version 4.1.2).

For these experiments, a value of 100 was used for each reference point objective. Note that unlike searching for the least contributing point (e.g. as in [19]) where only the minimum necessary work is completed to determine which point contributes the least, the choice of reference point for calculating entire hypervolumes (all slices) does not affect time peformance as the algorithm will perform the same operations regardless.

A. Test data

We compare the performance of the three algorithms on two different types of data.

- We used 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 used the spherical (DTLZ2), degenerate (DTLZ5) and discontinuous (DTLZ7) from the DTLZ test suite [28]. Properties of these test problems can be found in [28], [29]. 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. The degenerate front can be processed in in polynomial time [13] and we have omitted performance results in Figs. 3- 8 for these fronts for brevity. Experiments also show that the linear (DTLZ1) data behaves almost identically to the spherical data set and is thus omitted completely.

The test data used in the experiments are also available online [30].

B. Variation in HOY's performance

In order to compare the three algorithms, we first need to consider whether HOY's performance can be improved using similar methods to those we use with FPL and IIHSO. We performed a series of experiments to measure how the processing time of HOY varies with different fronts of a given type and size, and with different objective orders used to process a given front. We examined all 720 unique objective orderings for a unique 6D front containing 640 points. Table I shows these results.

We can see from the last column of Table I, that the ratio between the "average" objective ordering and the optimal ordering is significantly larger for HOY than for FPL and IIHSO. In particular, for the degenerate and spherical data types where the dimension-sweep algorithms benefit enormously from objective reordering, an optimal objective reordering for HOY offers little benefit. This limits the benefit that can be realised by optimising HOY using an objective reordering approach. Seemingly, the region splitting procedure that guarantees HOY its worst-case complexity may also limit the variation in processing time that could be exploited to improve its typical performance.

Therefore objective reordering heuristics, even one that guarantees an optimal ordering, could only provide a minor performance improvement for HOY. In the subsequent performance comparison, we thus use HOY without the use of heuristics.

C. Performance comparison

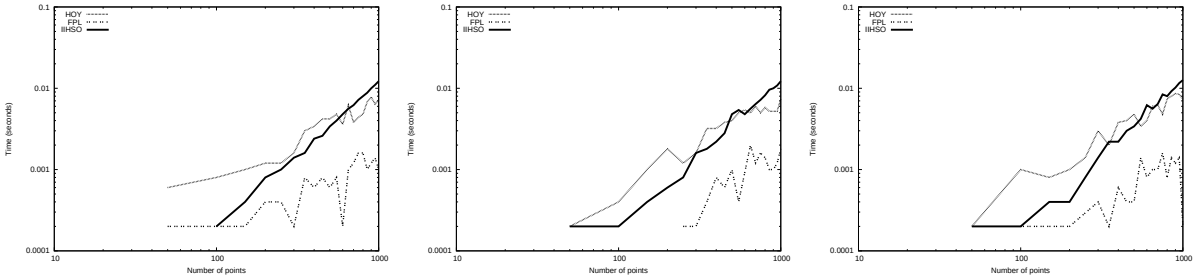
We optimised the FPL algorithm by re-ordering objectives in the data using the MWW heuristic from [13] prior to applying the baseline algorithm. We found that MWW did not in general improve the performance of FPL in less than 5 objectives, therefore we only applied the heuristic when calculating 5 or more objectives. Similar behaviour was noted for IIHSO using the MWW or Rank heuristics (the Rank heuristic is not applicable for FPL since it is only usable with contribution algorithms), and hence these heuristics were also only applied when calculating 5 or more objectives. As detailed above, no heuristics are applied for the HOY algorithm. Results below are the average of 20 independent runs on unique fronts. All timings include the cost of calculating heuristics.

Figures 3-8 plot performance timings for the three different algorithms on varying numbers of points and objectives using a log-log scale. Note that for a small number of points in few objectives, the resolution in timings can lead to erratic performance measurements.

We observe the following:

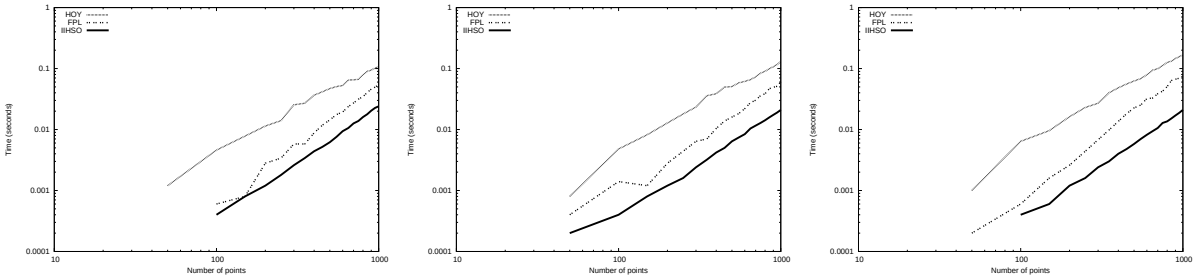
- In 4-11D, both IIHSO and FPL always outperform HOY, usually by a substantial amount. As we discussed in Section IV-B, the difference is unlikely to be covered by optimising HOY.
- For 3 objectives, the performance of IIHSO and FPL is always less than 0.01 seconds in up to 1000 points and the performance of all three algorithms is more than fast enough such that that any difference is insignificant.
- On random and discontinuous fronts, IIHSO outperforms FPL in five or more objectives, and for random data usually by a substantial amount (recall that the plots use a log-log scale).
- In contrast, FPL outperforms IIHSO on spherical and linear fronts in more than seven objectives, and the difference appears to increase as the number of objectives increases. Experiments confirm that this is largely due to the use of MWW to order the objectives for FPL. We believe that:
  - MWW works especially well for spherical and linear data [13];
  - Moreover, we believe that truly spherical data will be rare in real world problems with a large number of objectives.
- HOY, despite the fact that it has the best worst case complexity, performs significantly worse than the other algorithms in four or more objectives for all data types.

Table II shows the sizes of fronts that IIHSO using MWW and rank can process in ten seconds, for each front-type. The table shows that IIHSO can process substantial fronts in all types up to 8–9 objectives.



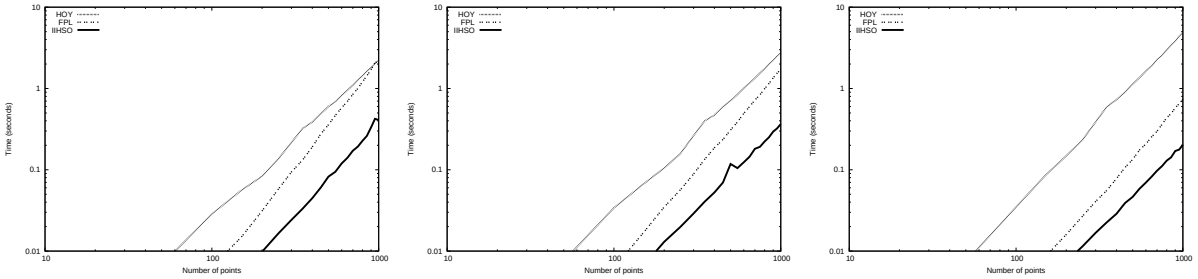
(a) Random fronts in 3 objectives. (b) Discontinuous fronts in 3 objectives. (c) Spherical fronts in 3 objectives.

Fig. 3. Comparison of the performance of IIHSO, HOY, and optimised FPL on 3-objective data. Each line plots the mean processing time for twenty distinct fronts.



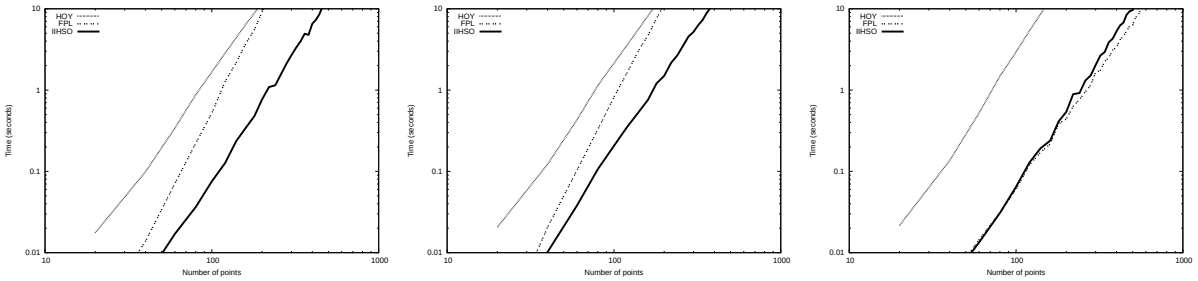
(a) Random fronts in 4 objectives. (b) Discontinuous fronts in 4 objectives. (c) Spherical fronts in 4 objectives.

Fig. 4. Comparison of the performance of IIHSO, HOY, and optimised FPL on 4-objective data. Each line plots the mean processing time for twenty distinct fronts.



(a) Random fronts in 5 objectives. (b) Discontinuous fronts in 5 objectives. (c) Spherical fronts in 5 objectives.

Fig. 5. Comparison of the performance of IIHSO, HOY, and optimised FPL on 5-objective data. Each line plots the mean processing time for twenty distinct fronts.



(a) Random fronts in 7 objectives. (b) Discontinuous fronts in 7 objectives. (c) Spherical fronts in 7 objectives.

Fig. 6. Comparison of the performance of IIHSO, HOY, and optimised FPL on 7-objective data. Each line plots the mean processing time for twenty distinct fronts.

TABLE I

TIME PERFORMANCE VARIATION FOR FPL, HOY AND IIHSO ON DIFFERENT DATA TYPES. FRONTS CONTAIN 640 POINTS IN 6D: ALL 720 UNIQUE OBJECTIVE ORDERINGS ARE EVALUATED.

Data type	Algorithm	mean (s)	std dev (s)	std dev/mean (s)	min time/mean (s)
random	FPL	25.957228	4.9649713	0.1912751	0.6768480
	HOY	25.584465	1.7647406	0.0689770	0.8490057
	IIHSO	4.4998702	0.7968367	0.1770799	0.5884983
discontinuous	FPL	39.576657	14.488473	0.3660863	0.3207141
	HOY	28.267166	2.3920060	0.0846214	0.8732939
	IIHSO	8.8341855	3.8786478	0.4390498	0.3418752
spherical	FPL	44.5003587	52.2093637	1.1732347	0.0316421
	HOY	35.0217941	10.1819085	0.2907306	0.5068018
	IIHSO	9.05977176	10.9557998	1.2092799	0.0737372
degenerate	FPL	0.80086665	1.21764017	1.5204031	0.0199796
	HOY	157.441001	31.2628222	0.1985685	0.6721160
	IIHSO	0.03517998	0.01340135	0.3809368	0.2274021

TABLE II

SIZES OF FRONTS THAT IIHSO CAN PROCESS IN TEN SECONDS.

n	Random	Discontinuous	Spherical
3	$>$ 10,000	$>$ 10,000	$>$ 10,000
4	$>$ 10,000	$>$ 10,000	9,600
5	3,500	4,600	5,500
6	900	900	1,700
7	510	410	500
8	190	145	240
9	115	80	135
10	70	50	85
11	50	40	65
12	42	33	51
13	34	26	34

D. Complexity analysis

In order to better compare performance, we have calculated the regression coefficients for the three hypervolume algorithms. A linear least squares fit was applied to the mean run time for each front size. Regression was performed on models using the stated complexities: HOY $O(n^{d/2})$ [15], FPL $O(n^{d-2}\log n)$ [18] and IIHSO $O(n^{d-1})$. Following Fonseca et al. [18] we use the regression model $\log_{10}(t/\log_2 n) = \alpha \log_{10} n + \log_{10} c$ for FPL. For IIHSO and HOY we use the regression model $\log_{10} t = \alpha \log_{10} n + \log_{10} c$. Base 10 was used for the log factors to be consistent with Fonseca et al. In order to provide a better fit, only data points with a mean greater than 0.1s were used for best fit regression. Table III shows the results of regressional analysis. It should be noted that the complexity coefficients for IIHSO and FPL cannot be directly compared due to the additional $\log_2$ factor in the theoretical time complexity for FPL.

For this reason, we have calculated the point at which each algorithm will be perform equally using the complexity coefficients and shown the range of front sizes where an algorithm outperforms the others. For all data types tested, IIHSO algorithm outperforms HOY for all (at this time) feasible front sizes. For example, for 9D discontinous data HOY outperforms IIHSO only on front sizes of 9145 points or more. A front of this size and dimensionality would take an astronomical time to compute. IIHSO fares very well in most cases. The key exceptions are for 3D data, where FPL's clear complexity advantage due to its superior 3D basecase becomes evident in the complexity exponents. Additionally, FPL outperforms IIHSO on spherical data with high dimensionality.

V. CONCLUSIONS

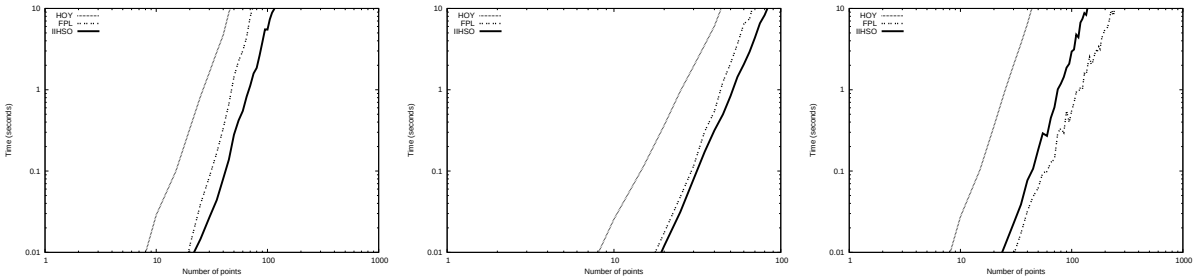
We have described a new algorithm IIHSO for computing exactly the hypervolume of a set of solutions returned by a multi-objective evolutionary algorithm. IIHSO works by repeated application of the incremental algorithm IHSO: starting from the empty set, it adds the solutions to the set one at a time, adding to an accumulator the extra volume contributed by each solution relative to those added previously. IIHSO also incorporates some ideas from the FPL algorithm, principally its linked data structure for optimising dominance calculations.

We have shown empirically that although the HOY algorithm for calculating hypervolume has by far the best worst-case complexity, IIHSO always outperforms HOY and usually outperforms FPL on the tested data in higher numbers of objectives. We have also shown empirically that it may be difficult to improve the typical performance of HOY. As demonstrated in Section IV-B, limited performance improvements can be made by objective reordering and it remains to be seen whether HOY can be improved further by achievements made possible by removing complexity in the Overmars and Yap algorithm that is not required when the Klee's measure problem is reduced to the special case of the hypervolume problem.

Our future work in this area is likely to focus on improving the performance of our incremental algorithm IHSO, for in-line use in a MOEA. In particular we plan to investigate the elimination of repeated work between slices at different stages of the algorithm. Any improvements in IHSO should improve the performance of IIHSO too.

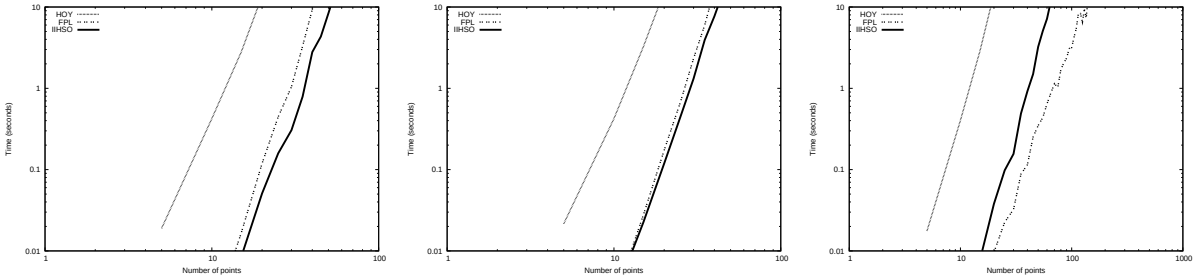
ACKNOWLEDGMENTS

We thank Simon Huband for generating the raw DTLZ data.



(a) Random fronts in 9 objectives. (b) Discontinuous fronts in 9 objectives. (c) Spherical fronts in 9 objectives.

Fig. 7. Comparison of the performance of IIHSO, HOY, and optimised FPL on 9-objective data. Each line plots the mean processing time for twenty distinct fronts.



(a) Random fronts in 11 objectives. (b) Discontinuous fronts in 11 objectives. (c) Spherical fronts in 11 objectives.

Fig. 8. Comparison of the performance of IIHSO, HOY, and optimised FPL on 11-objective data. Each line plots the mean processing time for twenty distinct fronts.

TABLE III

CURVE FITTING COEFFICIENTS AND INTERCEPTS FOR VARIOUS DATA TYPES.

Data type	d	IIHSO			FPL			HOY		
		α	$\log_{10} c$	Best (pts)	α	$\log_{10} c$	Best (pts)	α	$\log_{10} c$	Best (pts)
random	3	1.770704	-7.240801	1-106	0.558358	-5.611359	107+	0.935127	-4.967349	–
	4	1.810152	-7.060932	29-80263	1.849852	-7.801893	1-28	1.445007	-5.270071	80263+
	5	2.227268	-7.104002	9-43210	2.340898	-7.710544	1-8	1.935844	-5.461041	434211+
	7	3.089519	-7.247319	10-2.09829e8	3.609284	-8.267449	1-9	2.887177	-5.563456	2.09829e8+
	9	4.175577	-7.635535	10-2.40306e13	4.690582	-8.629039	1-9	4.033135	-5.729552	2.40306e13+
	11	5.463075	-8.408977	1-367	4.827081	-7.708278	368-2.85651e9	4.702927	-5.037208	2.85651e9+
discontinuous	3	1.603184	-6.764715	–	1.262008	-7.542020	1-1.92965e6	1.161358	-5.589658	1.92965e6+
	4	1.651665	-6.660594	5-7.00165e11	1.558649	-6.943409	1-4	1.555121	-5.517011	7.00165e11+
	5	2.068377	-6.644558	16-7.7891e10	2.201865	-7.391492	1-15	1.957164	-5.433283	3971+
	7	3.067595	-6.868993	26-1.68843e9	3.856289	-8.657011	1-25	2.922631	-5.531340	1.68843e9+
	9	4.650604	-7.950299	18-9144	5.168483	-9.204628	1-17	4.095770	-5.752515	9145+
	11	5.841080	-8.483601	11-1231	6.003370	-9.187951	1-10	4.716577	-5.008529	1232+
spherical	3	1.636459	-6.860508	1-48	0.665682	-5.973973	49+	1.184615	-5.607789	–
	4	1.777519	-7.025577	16+	1.880756	-7.729189	1-15	1.585746	-5.493311	–
	5	1.988159	-6.679650	46+	2.222188	-7.806057	1-45	2.152739	-5.764965	–
	7	3.046349	-7.232591	1-90	2.611195	-7.194956	89+	3.143627	-5.823398	–
	9	3.635945	-6.864695	–	2.833817	-6.703751	1+	4.171840	-5.856814	–
	11	4.453226	-7.154271	–	3.395058	-7.051992	1+	4.805196	-5.147425	–
degenerate	3	1.755361	-7.243959	1-99	0.414296	-5.388624	100+	1.352020	-5.994287	–
	4	1.584132	-6.725340	5+	1.680270	-7.152688	1-4	1.925496	-6.191779	–
	5	1.794844	-6.847725	1-6, 1.63438e13+	1.697024	-7.197598	7-1.63438e13	2.408819	-6.092468	–
	7	1.910468	-6.697617	1-6, 5.96077e6+	1.757720	-7.015022	7-5.96077e6	3.660178	-6.490656	–
	9	1.884317	-6.362168	4379+	1.551482	-6.232864	1-4378	4.545753	-6.178759	–
	11	1.788522	-5.912601	1-30, 1.75216e15+	1.715111	-6.498032	31-1.75216e15	5.828428	-6.297117	–

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] 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.

[5] 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.

[6] 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.

[7] T. Wagner, N. Beume, and B. Naujoks, "Pareto-, aggregation-, and indicator-based methods in many-objective optimization," *Lecture Notes in Computer Science*, vol. 4403, p. 742, 2007.

[8] 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.

[9] J. Bader, K. Deb, and E. Zitzler, "Faster Hypervolume-based Search using Monte Carlo Sampling," in *Conference on Multiple Criteria Decision Making (MCDM 2008)*. Springer, 2008, pp. 313–326.

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

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

[12] 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.

[13] 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.

[14] 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.

[15] N. Beume and G. Rudolph, "Faster S-metric calculation by considering dominated hypervolume as klee's measure problem," University of Dortmund, Technical Report CI 216/06, 2006.

[16] L. Paquete, C. M. Fonseca, and M. López-Ibáñez, "An optimal algorithm for a special case of Klee's measure problem in three dimensions," CSI, Universidade do Algarve, Tech. Rep. CSI-RT-I-01/2006, 2006.

[17] N. Beume, C. Fonseca, M. López-Ibáñez, L. Paquete, and J. Vahrenhold, "On the complexity of computing the hypervolume indicator," 2007.

[18] 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.

[19] L. Bradstreet, L. While, and L. Barone, "A fast incremental hypervolume algorithm," *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 6, pp. 714–723, Dec. 2008.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] N. Beume, "HOY source code," 2006. [Online]. Available: http://ls11-www.cs.uni-dortmund.de/people/beume/publications/hoy.cpp

[26] C. M. Fonseca, M. López-Ibáñez, and L. Paquete, "Computation of the hypervolume indicator," 2007. [Online]. Available: *http://sbe.napier.ac.uk/ manuel/hypervolume*

[27] Walking Fish Group, "IIHSO code," 2009. [Online]. Available: *wfg.csse.uwa.edu.au/Hypervolume*

[28] 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.

[29] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multi-objective test problems and a scalable test problem toolkit," *Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, 2006.

[30] Walking Fish Group, "Hypervolume test data," 2009. [Online]. Available: *wfg.csse.uwa.edu.au/Hypervolume*

Paper 7 (Refereed)

L. While, L. Bradstreet, and L. Barone. A Fast Way of Calculating Exact Hypervolumes, 2010. To appear in *IEEE Transactions on Evolutionary Computation*

A Fast Way of Calculating Exact Hypervolumes

Lyndon While, *Senior Member, IEEE*, Lucas Bradstreet, *Student Member, IEEE*, 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 than all previously-described algorithms that calculate hypervolume exactly.**

***Index Terms*— Multi-objective optimisation, evolutionary computation, diversity, performance metrics, hypervolume.**

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 multi-objective optimisers. 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 multi-objective evolutionary algorithms, either to promote diversity [6], or as part of an archiving mechanism [7], or as part of the selection process [8], [9]. 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

- the HOY algorithm [10], 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);
- a provably optimal $O(m \log m)$ algorithm [11] for the 3D case;
- the IIHSO algorithm [12], 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 [13], [14] 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 optimisations, this bounding technique gives a new and very simple algorithm WFG that is far faster than any previously-published algorithm for calculating hypervolumes exactly.

The authors are with The University of Western Australia.

The remainder of the paper is structured as follows. Section II describes the necessary background material in multi-objective optimisation 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 optimised, an experimental comparison of WFG with other recent hypervolume algorithms, and an analysis of WFG's complexity. Section V concludes the paper and suggests some future work, including a discussion of using the bounding technique in-line in evolutionary algorithms.

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]–[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 a 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], or as part of the selection process [8], [9]: 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.

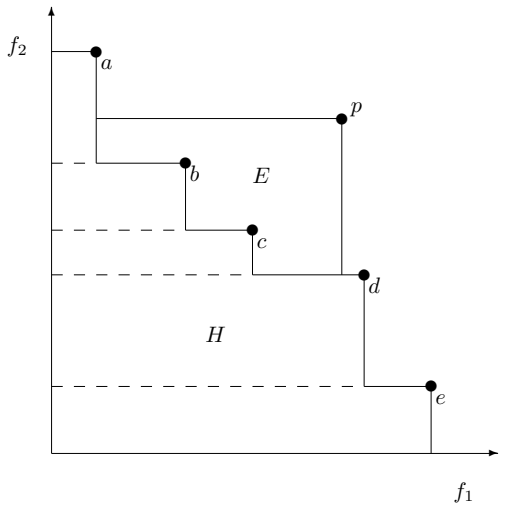


Fig. 1. Maximising in both objectives relative to the origin, the hypervolume of $\{a, b, c, d, e\}$ is the solid-bordered shape labeled H, the exclusive hypervolume of p relative to $\{a, b, c, d, e\}$ is the shape labeled E, and the inclusive hypervolume of any point is the rectangle bounded by that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

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 $O(n2^m)$ as it examines every subset of the set of points, 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 (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 [18] and even when optimised it is very slow [19].

HSO (Hypervolume by Slicing Objectives) [19]–[21] 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 $O(m^{n-1})$ [19] , 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] optimises HSO further, mainly through the use of an advanced data structure to optimise repeated domination checks and the recalculation of partial hypervolumes. It also incorporates a recent optimal algorithm for the 3D case [11], that works by maintaining a sorted 2D front in a balanced tree structure as it descends in the third objective. The key to the 3D algorithm is that this front can be updated in logarithmic time.

IHSO (Incremental HSO) [24] is a version of HSO customised 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.

IIHSO (Iterated IHSO) [12] combines the efficient incremental calculations of IHSO with the point-wise processing of LebMeasure, again with heuristics to optimise the performance for a given data set. The resulting algorithm is the fastest yet known on much typical benchmark data in many objectives.

HOY (Hypervolume by Overmars and Yap) [10] adapts an algorithm for solving Klee's measure problem [25] 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 [12], [26] 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 [26]–[28]). Using such algorithms introduces a trade-off between precision and performance, and much improvement has been reported recently on understanding this trade-off [26]. However, we do not compare with approximation algorithms here.

D. Calculating Hypervolume Point-wise

LebMeasure [17] introduced the idea of calculating hypervolume as a summation of exclusive hypervolumes, i.e.

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

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, i.e.

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

This technique is used in IIHSO [12], and we use it again in our new algorithm WFG, described in Section IV.

III. A New Method for Calculating Exact Exclusive Hypervolumes

(1) provides a clear definition for exclusive hypervolume, but it doesn't provide an efficient mechanism for its calculation: it requires two separate hypervolume calculations. IHSO [24] gets around this by calculating the differences between the two hypervolumes directly, using two main tricks.

- 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.
- 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 [13] and Bradstreet *et al.* [14] describe 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 (5)–(7).

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

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 optimisations described in Section IV-B increase these percentages significantly.

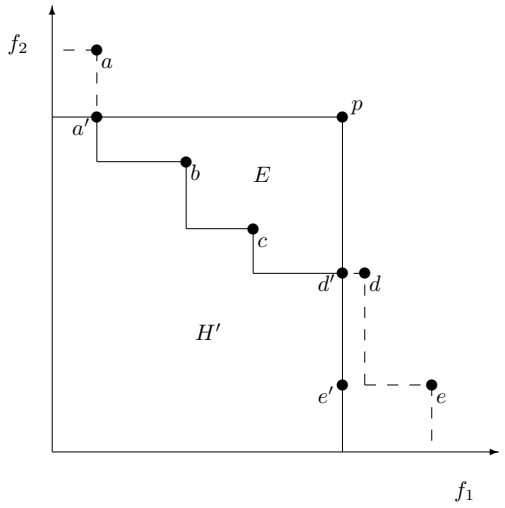


Fig. 2. Maximising in both objectives, the exclusive hypervolume of p relative to $\{a, b, c, d, e\}$ (i.e. E) = the inclusive hypervolume of p (i.e. the rectangle with p at the top-right corner) minus the hypervolume of $\{a', b, c, d', e'\}$ (i.e. H'). Clearly e' is dominated by d' and can be discarded.

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.

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. A pathological example for the bounding technique. This pattern describes sets of m points in five objectives, all being maximised and all (except the first) increasing monotonically after the first four points. The pattern can be generalised for other numbers of objectives.

Bringmann and Friedrich [13] make 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 optimising the performance of WFG in Section IV-B.

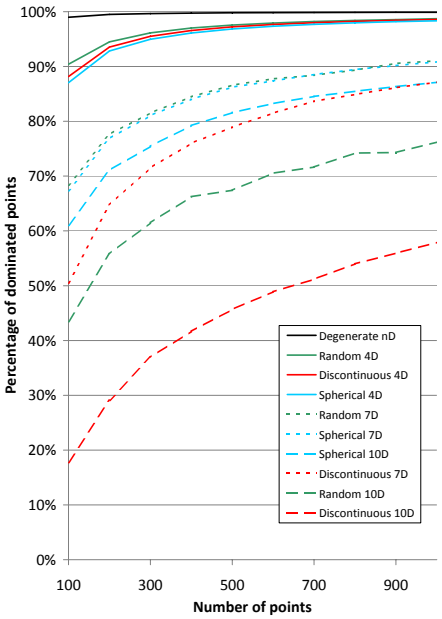


Fig. 4. Percentage of dominated points in a front after it has been limited by one of its points. Each datum is the average of twenty distinct fronts, where for each point in each front we calculate the number of dominated points after the front is limited by that point, then we average those figures.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

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, 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. Optimisations

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

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.

```
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.

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 optimisation 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 optimisation.

2) Slicing the points: Once the points are sorted in an objective, we can optimise 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 3D algorithm of Beume *et al.* [11]. This algorithm has complexity $O(m \log m)$, but of course it saves one level of recursion in the main algorithm.

Other optimisations 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 optimisations, and to compare its performance with other recent hypervolume algorithms. We used three 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,

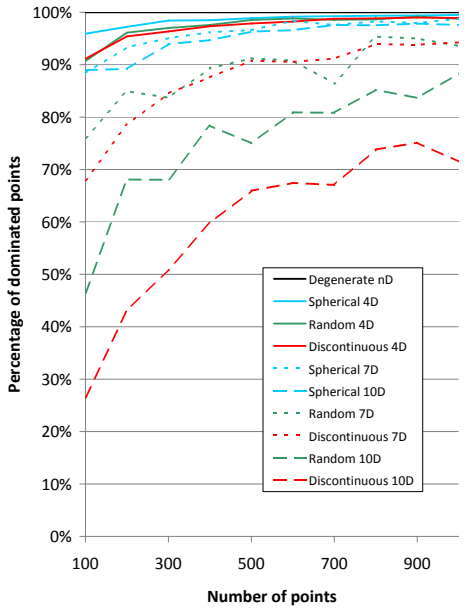


Fig. 6. Percentage of dominated points in a front after it has been limited by the worst point in the last objective. Each datum is the average of twenty distinct fronts. The degenerate line is 100% everywhere. The data is described in Section IV-C.

we initialised $S = \phi$ and added each point $\overline{x}$ to S only if $\{\overline{x}\} \cup S$ would be mutually-non-dominating.

- The discontinuous, spherical, and degenerate fronts from the DTLZ test suite [29]. 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.
- The "worst-case data" for the bounding technique, from Fig. 3.

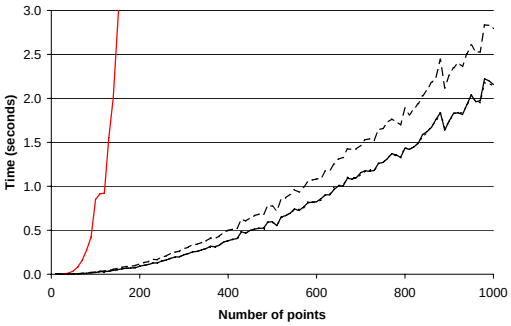
The data used in the experiments is available from `wfg.csse.uwa.edu.au/hypervolume/hv8`, and the code is available as follows.

- HOY: `ls11-www.cs.uni-dortmund.de/people/beume/publications/hoy.cpp`.
- FPL: `iridia.ulb.ac.be/~manuel/hypervolume` (version 1.2).
- IIHSO and WFG: `wfg.csse.uwa.edu.au/hypervolume/hv8`.

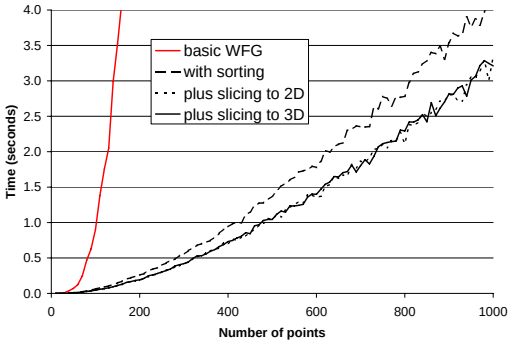
All timings were performed on an Intel 2.4GHz Core 2 Duo processor with 2GB of RAM, running Ubuntu Linux 8.04.3. All algorithms were compiled with `gcc/g++ -O3 -funroll-loops -march=nocona` (version 4.2.4).

Figs. 7–12 are all available in log-log format at `wfg.csse.uwa.edu.au/hypervolume/hv8`.

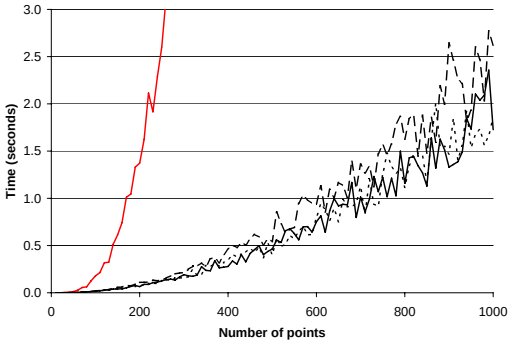
1) Comparing the optimisations: Fig. 7 reports experiments on fronts of various sizes and types comparing the performance of WFG using the optimisations discussed in



(a) Random fronts in 7 objectives.



(b) Discontinuous fronts in 7 objectives.



(c) Spherical fronts in 7 objectives.

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

Section IV-B. The graphs show clearly that simply sorting the points so that they are improving in one objective provides a substantial speed-up for WFG: slicing the points to either a 2D or 3D base case provides an additional speed-up 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 optimised WFG with FPL [23], HOY [10], and IIHSO [12]. FPL was optimised by using the MWW heuristic [22] (adapted for FPL's complexity model) to re-order the objectives, if there are more than four. HOY was run as is: no-one has yet published any useful optimisations for this algorithm. 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 4D data in Fig. 8, but the actual difference in the timings is very small.

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 by these results. However, degenerate appears to be the worst-case data for HOY: it can process only around 10–15 points in 13D in a reasonable time frame. Although WFG is the fastest of the other algorithms, the actual differences are minor.

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 eleven objectives, and probably a lot more objectives in spherical.

TABLE I
SIZES OF FRONTS THAT WFG CAN PROCESS IN TEN SECONDS. n IS THE NUMBER OF OBJECTIVES.

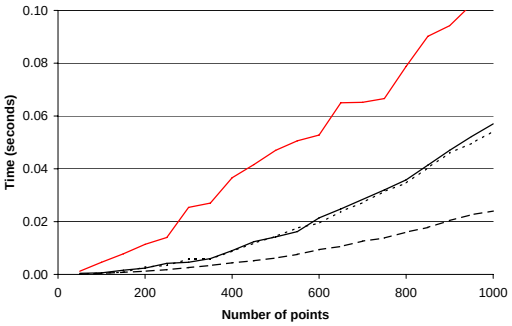
n	Random	Discontinuous	Spherical
3	$>$ 10,000	$>$ 10,000	$>$ 10,000
4	$>$ 10,000	$>$ 10,000	$>$ 10,000
5	$>$ 10,000	9,600	7,800
6	5,800	5,600	4,400
7	2,300	2,000	2,100
8	900	600	1,300
9	450	300	1,000
10	235	150	700
11	185	105	575
12	130	85	475
13	100	70	450

D. Complexity

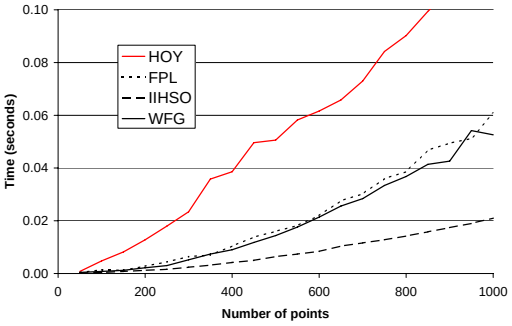
The complexity of basic WFG for m points in n objectives can be modelled by the following recurrence relations.

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

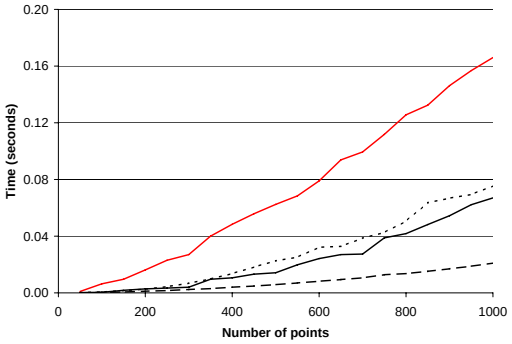
$$g(m,n) = 1 + f(m',n) \quad (9)$$



(a) Random fronts in 4 objectives.

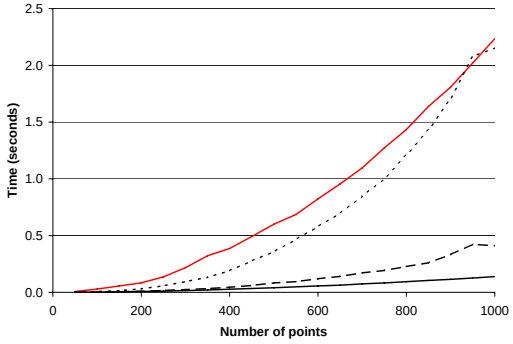


(b) Discontinuous fronts in 4 objectives.

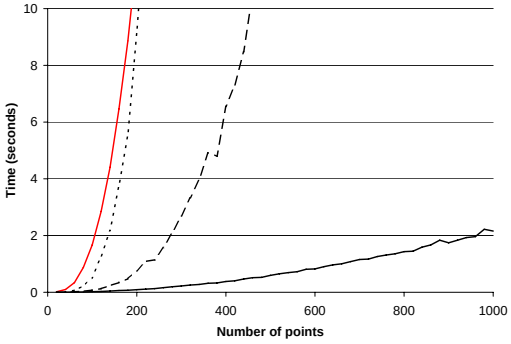


(c) Spherical fronts in 4 objectives.

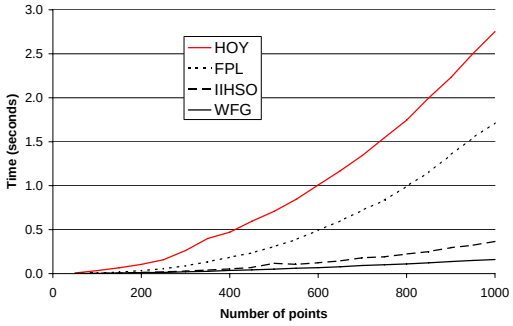
Fig. 8. Comparison of the performance of WFG with FPL, HOY, and IIHSO in 4 objectives. Each datum is the average processing time for twenty distinct fronts. The legend on the middle graph applies for all three.



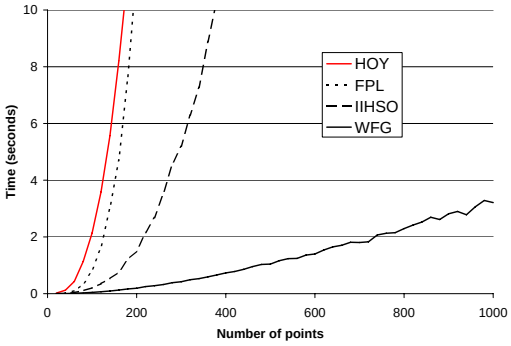
(a) Random fronts in 5 objectives.



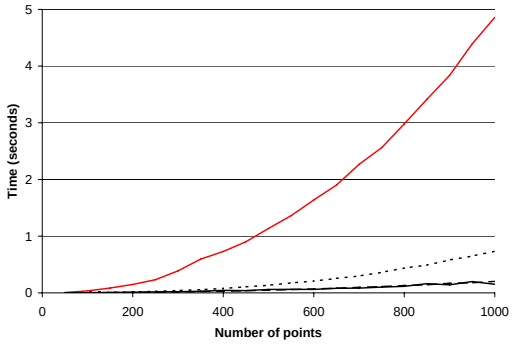
(a) Random fronts in 7 objectives.



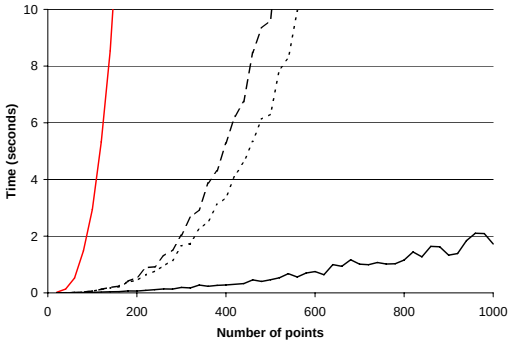
(b) Discontinuous fronts in 5 objectives.



(b) Discontinuous fronts in 7 objectives.



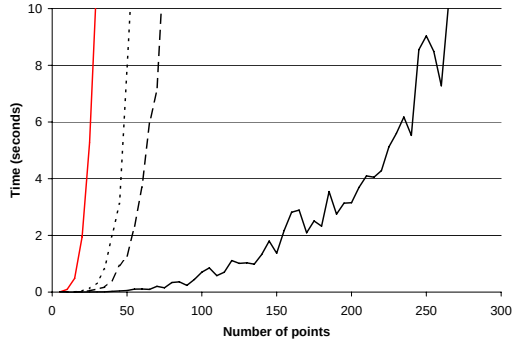
(c) Spherical fronts in 5 objectives.



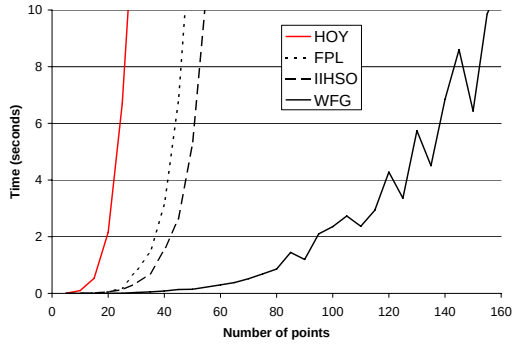
(c) Spherical fronts in 7 objectives.

Fig. 9. Comparison of the performance of WFG with FPL, HOY, and IIHSO in 5 objectives. Each datum is the average processing time for twenty distinct fronts. The legend on the middle graph applies for all three.

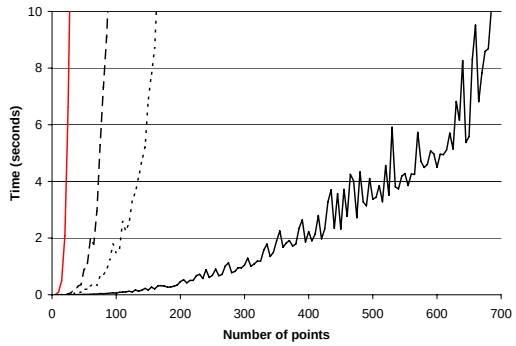
Fig. 10. Comparison of the performance of WFG with FPL, HOY, and IIHSO in 7 objectives. Each datum is the average processing time for twenty distinct fronts. The legend on the middle graph applies for all three.



(a) Random fronts in 10 objectives.

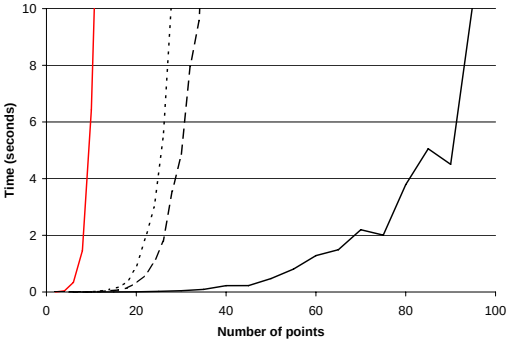


(b) Discontinuous fronts in 10 objectives.

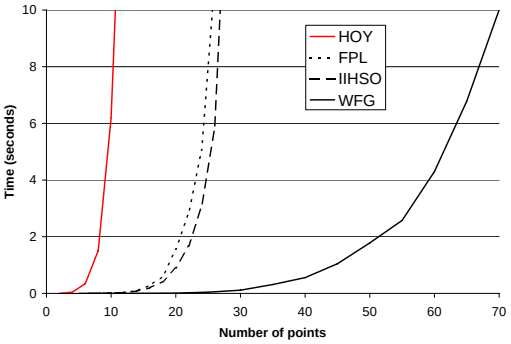


(c) Spherical fronts in 10 objectives.

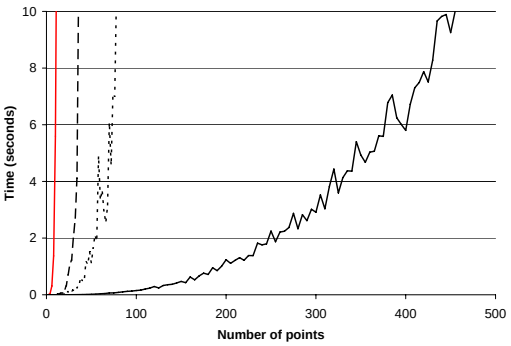
Fig. 11. Comparison of the performance of WFG with FPL, HOY, and IIHSO in 10 objectives. Each datum is the average processing time for twenty distinct fronts. The legend on the middle graph applies for all three.



(a) Random fronts in 13 objectives.



(b) Discontinuous fronts in 13 objectives.



(c) Spherical fronts in 13 objectives.

Fig. 12. Comparison of the performance of WFG with FPL, HOY, and IIHSO in 13 objectives. Each datum is the average processing time for twenty distinct fronts. The legend on the middle graph applies for all three.

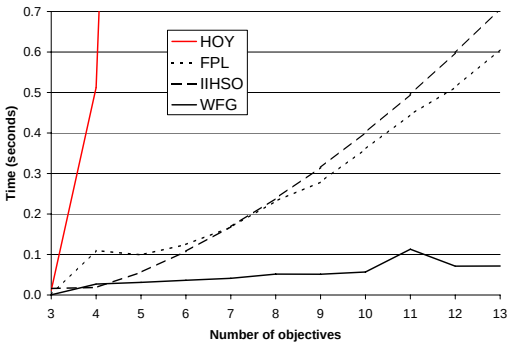


Fig. 13. Comparison of the performance of WFG with FPL and IIHSO on the degenerate front-type. Each datum is the average processing time for twenty distinct fronts, each with 1,000 points.

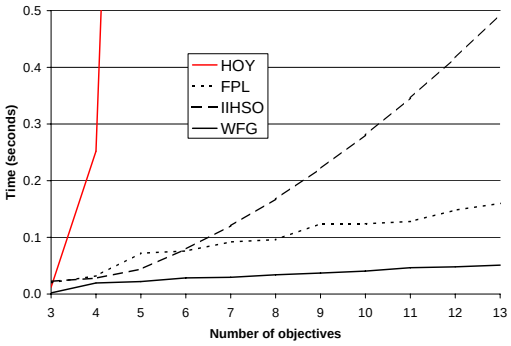


Fig. 14. Comparison of the performance of WFG with FPL, HOY, and IIHSO on the data from Fig. 3. Each front has 1,000 points.

(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

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

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 $m = j$, i.e.

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

Then

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

■

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 realised for typical real-world data.

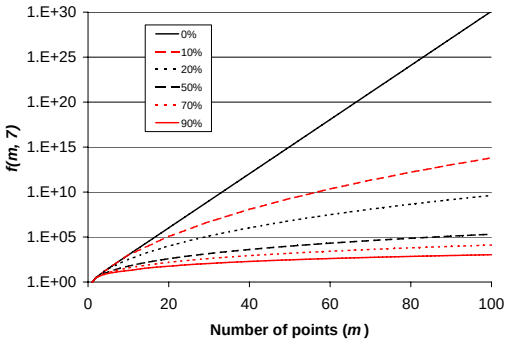


Fig. 15. Plot of the complexity of WFG in 7 objectives, for various consistent percentages of dominated points. 0% corresponds to the theoretical worst case for the algorithm.

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. Figs. 7–12 are all available in log-log format at `wfg.csse.uwa.edu.au/hypervolume/hv8`.

V. Conclusions

We have described a new algorithm WFG that can calculate exact hypervolumes 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 leads to very fast calculations. WFG applies this technique iteratively over a set to calculate its hypervolume. We also describe some minor but effective optimisations for WFG.

Future work on WFG will include investigating further optimisation by

- using heuristics to select a good objective for sorting and slicing;
- using different sorting rules to generate smaller underlying sets;
- re-organising 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.

Acknowledgements

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, 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.

[8] 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.

[9] 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.

[10] N. Beume and G. Rudolph, "Faster S-metric calculation by considering dominated hypervolume as Klee's measure problem," University of Dortmund, Technical Report CI 216/06, 2006.

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

[12] L. Bradstreet, L. While, and L. Barone, "A faster many-objective hypervolume algorithm using iterated incremental calculations," in *Congress on Evolutionary Computation*. IEEE, 2010.

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

[14] 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.

[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, "A new analysis of the LebMeasure algorithm for calculating hypervolume," in *EMO 2005*, ser. LNCS, C. Coello Coello *et al.*, Ed., vol. 3410. Springer-Verlag, 2005, pp. 326–340.

[19] 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.

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

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

[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] L. Bradstreet, L. While, and L. Barone, "A fast incremental hypervolume algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 714–723, December 2008.

[25] 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.

[26] K. Bringmann and T. Friedrich, "Approximating the least hypervolume contributor: NP-hard in general, but fast in practice," in *EMO 2009*, ser. LNCS, vol. 5467. Springer-Verlag, 2009, pp. 6–20.

[27] 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.

[28] J. Bader, K. Deb, and E. Zitzler, "Faster hypervolume-based search using Monte Carlo sampling," in *Proceedings of MCDM-08*. Springer, 2008.

[29] 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.

Chapter 5

Conclusions

Assessment of multi-objective optimisers is a difficult problem and many metrics have been proposed for this purpose. Hypervolume, as one of the most promising, has lead to a large volume of research relating to both its calculation and its use.

This thesis examined the computation and application of hypervolume in multi-objective optimisation and introduced several new calculation algorithms and techniques which improve hypervolume's use.

Research was focused on three distinct areas:

- metric hypervolume calculations for performance assessment,
- exclusive hypervolume contribution calculations, and
- hypervolume based selection techniques.

In the first area, this thesis focused on hypervolume metric algorithms and heuristics with a view to improving hypervolume calculation performance. *Paper 1* introduced the use of heuristics in the HSO dimension-sweep hypervolume algorithm, improving the typical run-time performance of the HSO algorithm substantially. A better hypervolume calculation algorithm, IIHSO, was introduced in *Paper 6* that utilises the IHSO hypervolume contribution algorithm. Further improving on this, *Paper 7* introduced the WFG algorithm, an algorithm which combines the iterative

contribution based approach used by LebMeasure and IIHSO with an inclusion-exclusion approach.

In the second focus, *Paper 3* introduced IHSO, an HSO adaptation that directly calculates exclusive hypervolume contributions. IHSO eliminates operations from HSO which are unnecessary for exclusive hypervolume contribution calculations. *Paper 3* also introduced performance improving heuristics designed for IHSO. As IHSO can be used within steady-state MOEAs, or MOEAs using greedy front selection, *Paper 3* also introduced a BFS technique to minimise the calculations necessary to find the least contributing point.

Exclusive hypervolume contributions become inaccurate when a front changes. *Paper 5* introduced a technique to quickly update the difference in contribution that reflects the addition or removal of a solution. This technique can greatly reduce run-time in MOEAs that use hypervolume contributions in selection, such as SIBEA [30, 71] or SMS-EMOA [36, 54].

In the third area of contribution, this thesis investigated and benchmarked some simple selection techniques that can be used within multi-objective optimisers, focusing on techniques for maximising the hypervolume of front selections. Techniques for using hypervolume for this purpose are rapidly improving and are only now becoming viable for real world use [6, 9, 21, 22, 71].

Paper 2 introduced two basic selection techniques for use within a non-steady state MOEA. A greedy front reduction technique and a local search algorithm were compared and each were shown to be useful depending on the proportion of the front selected. The greedy reduction selection technique in *Paper 2* calculated the exclusive contribution of solutions using a naive application of HSO. *Paper 4* reevaluated these selection techniques after incorporating the IHSO algorithm and the BFS scheme, offering performance improvements to both the greedy and local search selection schemes. As these techniques allow the calculation of the least contributing point with a much smaller overhead, they improve the performance of MOOs using greedy selection techniques. *Paper 4* also introduced a greedy addition scheme,

useful when small proportions of a front are selected.

5.1 Summary of Main Achievements

The main achievements of this work are:

1. The introduction of two fast metric algorithms, IIHSO and WFG, that can be used to calculate higher objective problems considered infeasible for many existing hypervolume algorithms.

   IIHSO is based on iterative applications of IHSO and builds front hypervolumes using exclusive hypervolume contributions. Given similar front sizes and equivalent time, IIHSO can calculate hypervolumes of fronts in roughly two more objectives than HSO.

   WFG combines iterative calculation of exclusive hypervolume contributions with a technique to calculate contributions using a metric hypervolume algorithm. WFG outperforms all existing hypervolume algorithms, including IIHSO, on tested front types.

2. The introduction and evaluation of objective reordering heuristics. These heuristics improve the typical performance of many hypervolume algorithms based on a dimension-sweep approach, such as HSO, IHSO, and FPL.

3. The introduction of a fast incremental exclusive contribution calculation algorithm, IHSO, and a search technique to minimise the computation required to find the least contributing solution.

   Exclusive hypervolume contributions can be used as part of a multi-objective selection scheme and represent the change in the hypervolume of a set that results when a solution is removed. The removal of a solution that contributes the least to the hypervolume of a set maximises the hypervolume of the set with one fewer solution. For this purpose, this thesis also introduced a search

scheme to minimise the work necessary to determine the least contributing point. Additional objective ordering heuristics were also created to improve the typical performance of IHSO.

The operation to find the least contributing point may be used to repeatedly reduce the size of a set. Contributions calculated in one iteration may not be correct in the next, but completely recalculating contributions is an expensive proposition. This thesis also introduced a technique to greatly reduce the computation required to update previously calculated contributions.

4. The introduction and comparison of several hypervolume based solution selection techniques that can be used in multi-objective optimisation. These are intended to be used in MOEAs as part of a hypervolume-based selection measure and aim to maximise the hypervolume of reduced size front subsets. They do so without resorting to the calculation of the hypervolume of every selection combination, an operation which is usually infeasible. A search based approach and two greedy methods for selection are introduced and compared. While these are relatively simple methods they were some of the first to allow hypervolume to be used to select more than one solution per generation. They provide a performance and quality baseline for comparison with newer hypervolume selection techniques.

5.2 Future Directions

Recent research into algorithms with the best worst case time complexity has concentrated on application of the Overmars and Yap algorithm for the Klee's Measure Problem, an old problem in computational geometry. The HOY algorithm [11] is one such adaptation and improves slightly upon the complexity of the Overmars and Yap algorithm by taking advantage of the fact that hypervolume calculations are a restricted case of the Klee's Measure Problem. It is possible that further similar improvements could also be made. Additionally, heuristics which improve

the performance of the HOY algorithm may be possible, however results in *Paper 6* (see Table 1) and in Palm [57] show that they may be non-trivial in practice. As also described in *Paper 6*, algorithms with the best worst case complexity may not perform well in practice. Therefore, algorithms with seemingly unfavourable worst case complexities which perform well on real world problems should not necessarily be discarded in favour of those with a lower upper bound which may in fact perform relatively poorly on common problems.

Papers 2 and *4* described selection methods, such as the local search approach and greedy reduction and addition approaches, intended to be used in the selection operators of MOEAs. The update method discussed in *Paper 5* further improves the performance of these methods by quickly updating contributions as front composition changes. However, the full effectiveness of these selection methods when used within actual MOEAs is unknown. Future research should evaluate these techniques within MOEAs on test problems and real world problems.

A recent approach by Bringmann and Friedrich [22] is able to calculate the optimal front selection without calculating all subset combinations individually, and without a multiplicative combinatorial explosion in run-time. For problems in few objectives, this scheme is likely to find favour in the future. For problems with many-objectives, where HOY (the algorithm from which it was derived) has been shown to perform poorly, this algorithm is likely infeasible even ignoring the additional m^{λ} term. Furthermore, problems with additional objectives generally lead to larger front sizes and fewer unique ranks [38, 61], increasing the size of m. Therefore this, algorithm could be extremely infeasible for many-objective problems most of the time. Further research is required to improve the feasibility of this approach. A combination of greedy and optimal approaches is one suggestion, albeit at the cost of an optimal selection.

Hypervolume based selection can also benefit from further research into objective reduction techniques. Objective reduction reduces the cost of calculating hypervolume greatly by removing objectives which do not affect selection decisions greatly.

The removal of a single objective can hugely improve the cost of hypervolume calculation. For example, calculating a random 5D front with 1000 points using IIHSO takes roughly 17 times longer than a random 4D front of identical size.

While the WFG algorithm, described in *Paper 7*, performs extremely well on tested experimental data, it could benefit from additional work to improve its performance. Possible optimisations for WFG include:

- heuristics to select a good objective for sorting and slicing,
- different sorting rules to generate smaller underlying sets, and
- reorganising the domination calculations in WFG.

Additionally, WFG's performance for exclusive hypervolume contribution calculations should be tested. When used to find the least contributing point, it may be possible to apply the BFS scheme used with IHSO in *Paper 3*. Judging by WFG's excellent performance as a metric, it is possible that WFG, when used with a BFS scheme, will outperform IHSO with BFS when used for this purpose.

Recent approximation techniques, such as those used in HypE by Bader et al. [5, 6] and Bringmann and Friedrich [21, 23], are another promising area of study. While recent research into exact hypervolume algorithms has led to greatly improved performance, it is likely that sampling techniques will remain necessary for many-objective problems as hypervolume calculation is NP-hard. It may be possible to adapt some of the calculation algorithms and selection techniques discussed in this thesis to incorporate sampling techniques, and this is an area worthy of further investigation.

Bibliography

[1] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Articulating User Preferences in Many-objective Problems by Sampling the Weighted Hypervolume. In *2009 Genetic and Evolutionary Computation Conference (GECCO'2009)*, pages 555–562, Montreal, Canada, July 8–12 2009. ACM Press. ISBN 978-1-60558-325-9.

[2] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Investigating and Exploiting the Bias of the Weighted Hypervolume to Articulate User Preferences. In *2009 Genetic and Evolutionary Computation Conference (GECCO'2009)*, pages 563–570, Montreal, Canada, July 8–12 2009. ACM Press. ISBN 978-1-60558-325-9.

[3] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Theory of the Hypervolume Indicator: Optimal $\{\mu\}$-Distributions and the Choice Of The Reference Point. In *FOGA '09: Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms*, pages 87–102, Orlando, Florida, USA, January 2009. ACM.

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

[5] J. Bader, K. Deb, and E. Zitzler. Faster Hypervolume-based Search Using Monte Carlo Sampling. In *Conference on Multiple Criteria Decision Making*

(MCDM 2008), pages 313–326. Springer, 2008.

[6] J. Bader and E. Zitzler. HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. *Evolutionary Computation*, 2009.

[7] L. Barone, L. While, and P. Hingston. Designing Crushers with a Multi-Objective Evolutionary Algorithm. In W. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. Schultz, J. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 995–1002, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[8] N. Beume, C. Fonseca, M. Lopez-Ibanez, L. Paquete, and J. Vahrenhold. On the Complexity of Computing the Hypervolume Indicator. *Evolutionary Computation, IEEE Transactions on*, 13(5):1075–1082, Oct. 2009.

[9] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 16 September 2007.

[10] N. Beume, B. Naujoks, M. Preuss, G. Rudolph, and T. Wagner. Effects of 1-Greedy S-Metric-Selection on Innumerably Large Pareto Fronts. In M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, editors, *Evolutionary Multi-Criterion Optimization. 5th International Conference, EMO 2009*, pages 21–35. Springer. Lecture Notes in Computer Science Vol. 5467, Nantes, France, April 2009.

[11] N. Beume and G. Rudolph. Faster S-Metric Calculation by Considering Dominated Hypervolume as Klee's Measure Problem. In B. Kovalerchuk, editor, *Computational Intelligence*, pages 233–238. IASTED/ACTA Press, 2006.

[12] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA—A Platform and Programming Language Independent Interface for Search Algorithms. In C. M.

Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 494–508, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[13] L. Bradstreet, L. Barone, and L. While. Maximising Hypervolume for Selection in Multi-objective Evolutionary Algorithms. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 6208–6215, Vancouver, BC, Canada, July 2006. IEEE Press.

[14] L. Bradstreet, L. Barone, and L. While. Updating Exclusive Hypervolume Contributions Cheaply. In *2009 IEEE Congress on Evolutionary Computation (CEC'2009)*, pages 538–544, Thronheim, Norway, May 2009. IEEE Press.

[15] L. Bradstreet, L. Barone, L. While, S. Huband, and P. Hingston. Use of the WFG Toolkit and PISA for Comparison of MOEAs. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (MCDM'2007)*, pages 382–389, Honolulu, Hawaii, USA, April 2007. IEEE Press.

[16] L. Bradstreet, L. While, and L. Barone. Incrementally Maximising Hypervolume for Selection in Multi-Objective Evolutionary Algorithms. In *2007 IEEE Congress on Evolutionary Computation (CEC'2007)*, pages 3203–3210, Singapore, September 2007. IEEE Press.

[17] L. Bradstreet, L. While, and L. Barone. A Fast Incremental Hypervolume Algorithm. *IEEE Transactions on Evolutionary Computation*, 12(6):714–723, December 2008.

[18] L. Bradstreet, L. While, and L. Barone. A New Way Of Calculating Exact Exclusive Hypervolumes. Technical Report, The University of Western Australia, 2009.

[19] L. Bradstreet, L. While, and L. Barone. Incremental Hypervolume by Slicing Objectives: a Correction to the Pseudo-code. *IEEE Transactions on Evolutionary Computation*, 13(5):1193–1193, November 2009.

[20] L. Bradstreet, L. While, and L. Barone. A Fast Many-objective Hypervolume Algorithm using Iterated Incremental Calculations. In *2010 IEEE Congress on Evolutionary Computation (CEC'2010)*, pages 179–186, Barcelona, Spain, July 2010. IEEE Press.

[21] K. Bringmann and T. Friedrich. Approximating the Least Hypervolume Contributor: NP-Hard in General, But Fast in Practice. In M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, editors, *Evolutionary Multi-Criterion Optimization. 5th International Conference, EMO 2009*, pages 6–20. Springer. Lecture Notes in Computer Science Vol. 5467, Nantes, France, April 2009.

[22] K. Bringmann and T. Friedrich. Don't Be Greedy When Calculating Hypervolume Contributions. In *FOGA '09: Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms*, pages 103–112, Orlando, Florida, USA, January 2009. ACM.

[23] K. Bringmann and T. Friedrich. Approximating the Volume of Unions and Intersections of High-dimensional Geometric Objects. *Algorithms and Computation*, pages 436–447, 2010.

[24] D. Brockhoff, T. Friedrich, N. Hebbinghaus, C. Klein, F. Neumann, and E. Zitzler. Do Additional Objectives Make a Problem Harder? In D. Thierens, editor, *2007 Genetic and Evolutionary Computation Conference (GECCO'2007)*, volume 1, pages 765–772, London, UK, July 2007. ACM Press.

[25] D. Brockhoff, T. Friedrich, and F. Neumann. Analyzing Hypervolume Indicator Based Algorithms. In G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, editors, *Parallel Problem Solving from Nature–PPSN X*, pages

651–660. Springer. Lecture Notes in Computer Science Vol. 5199, Dortmund, Germany, September 2008.

[26] D. Brockhoff and E. Zitzler. Are All Objectives Necessary? On Dimensionality Reduction in Evolutionary Multiobjective Optimization. In T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, and X. Yao, editors, *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference*, pages 533–542. Springer. Lecture Notes in Computer Science Vol. 4193, Reykjavik, Iceland, September 2006.

[27] D. Brockhoff and E. Zitzler. Dimensionality Reduction in Multiobjective Optimization with (Partial) Dominance Structure Preservation: Generalized Minimum Objective Subset Problems. TIK Report 247, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich, April 2006.

[28] D. Brockhoff and E. Zitzler. On Objective Conflicts and Objective Reduction in Multiple Criteria Optimization. TIK Report 243, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich, February 2006.

[29] D. Brockhoff and E. Zitzler. Dimensionality Reduction in Multiobjective Optimization: The Minimum Objective Subset Problem. In K. H. Waldmann and U. M. Stocker, editors, *Operations Research Proceedings 2006*, pages 423–429, Saarbücken, Germany, 2007. Springer.

[30] D. Brockhoff and E. Zitzler. Improving Hypervolume-based Multiobjective Evolutionary Algorithms by Using Objective Reduction Methods. In *2007 IEEE Congress on Evolutionary Computation (CEC'2007)*, pages 2086–2093, Singapore, September 2007. IEEE Press.

[31] D. Brockhoff and E. Zitzler. Offline and Online Objective Reduction in Evolutionary Multiobjective Optimization Based on Objective Conflicts. TIK Report 269, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich, April 2007.

[32] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.

[33] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.

[34] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.

[35] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable Multi-Objective Optimization Test Problems. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 825–830, Piscataway, New Jersey, May 2002. IEEE Service Center.

[36] M. Emmerich, N. Beume, and B. Naujoks. An EMO Algorithm Using the Hypervolume Measure as Selection Criterion. In C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 62–76, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.

[37] R. M. Everson, J. E. Fieldsend, and S. Singh. Full Elite Sets for Multi-Objective Optimisation. In I. Parmee, editor, *Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture (ACDM 2002)*, volume 5, pages 343–354, University of Exeter, Devon, UK, April 2002. Springer-Verlag.

[38] M. Farina and P. Amato. On the Optimal Solution Definition for Many-criteria Optimization Problems. In *Proceedings of the NAFIPS-FLINT International*

Conference'2002, pages 233–238, Piscataway, New Jersey, June 2002. IEEE Service Center.

[39] M. Fleischer. The Measure of Pareto Optima. Applications to Multi-objective Metaheuristics. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 519–533, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[40] C. M. Fonseca, L. Paquete, and M. López-Ibáñez. An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 3973–3979, Vancouver, BC, Canada, July 2006. IEEE Press.

[41] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[42] V. Grunert da Fonseca, C. M. Fonseca, and A. O. Hall. Inferential Performance Assessment of Stochastic Optimisers and the Attainment Function. In E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 213–225. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[43] S. Huband, L. Barone, L. While, and P. Hingston. A Scalable Multi-objective Test Problem Toolkit. In C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 280–295, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.

[44] S. Huband, P. Hingston, L. Barone, and L. While. A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, October 2006.

[45] S. Huband, P. Hingston, L. While, and L. Barone. An Evolution Strategy with Probabilistic Mutation for Multi-Objective Optimisation. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 2284–2291, Canberra, Australia, December 2003. IEEE Press.

[46] E. J. Hughes. Evolutionary Many-Objective Optimisation: Many Once or One Many? In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, volume 1, pages 222–227, Edinburgh, Scotland, September 2005. IEEE Service Center.

[47] H. Ishibuchi, N. Tsukamoto, Y. Sakane, and Y. Nojima. Hypervolume Approximation Using Achievement Scalarizing Functions for Evolutionary Many-Objective Optimization. In *2009 IEEE Congress on Evolutionary Computation (CEC'2009)*, pages 530–537, Thronheim, Norway, May 2009. IEEE Press.

[48] V. Klee. Can the Measure of $\cup [a_i, b_i]$ be Computed in Less Than $O(n \log n)$ Steps? *American Mathematical Monthly*, 84(4):284–285, 1977.

[49] J. Knowles and D. Corne. M-PAES: A Memetic Algorithm for Multiobjective Optimization. In *2000 Congress on Evolutionary Computation*, volume 1, pages 325–332, Piscataway, New Jersey, July 2000. IEEE Service Center.

[50] J. D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD thesis, The University of Reading, Department of Computer Science, Reading, UK, January 2002.

[51] H. T. Kung, F. Luccio, and F. P. Preparata. On Finding the Maxima of a Set of Vectors. *J. ACM*, 22(4):469–476, 1975.

[52] M. Laumanns, E. Zitzler, and L. Thiele. A Unified Model for Multi-Objective Evolutionary Algorithms with Elitism. In *2000 Congress on Evolutionary Computation*, volume 1, pages 46–53, Piscataway, New Jersey, July 2000. IEEE Service Center.

[53] B. Naujoks, N. Beume, and M. Emmerich. Metamodel-assisted SMS-EMOA Applied to Airfoil Optimization Tasks. In *Proceedings EUROGEN*, volume 5, 2005.

[54] B. Naujoks, N. Beume, and M. Emmerich. Multi-objective Optimization Using S-metric Selection: Application to Three-dimensional Solution Spaces. In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, volume 2, pages 1282–1289, Edinburgh, Scotland, September 2005. IEEE Press.

[55] T. Okabe, Y. Jin, and B. Sendhoff. A Critical Survey of Performance Indices for Multi-Objective Optimization. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 2, pages 878–885, Canberra, Australia, December 2003. IEEE Press.

[56] M. H. Overmars and C.-K. Yap. New upper bounds in Klee's measure problem. *SIAM Journal on Computing*, 20(6):1034–1045, December 1991.

[57] J. Palm. A Look at the Characteristics of the HOY Algorithm and a Generic Approach to Incremental Hypervolume Calculation. Honours dissertation, Department of Computer Science, The University of Western Australia, 2009.

[58] L. Paquete, C. M. Fonseca, and M. López-Ibáñez. An Optimal Algorithm for a Special Case of Klee's Measure Problem in Three Dimensions. Technical Report CSI-RT-I-01/2006, CSI, Universidade do Algarve, 2006.

[59] R. Purshouse and P. Fleming. The Multi-Objective Genetic Algorithm Applied to Benchmark Problems—An Analysis. Technical Report 796, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK, August 2001.

[60] R. C. Purshouse. *On the Evolutionary Optimisation of Many Objectives*. PhD thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK, September 2003.

[61] T. Wagner, N. Beume, and B. Naujoks. Pareto-, Aggregation-, and Indicator-Based Methods in Many-Objective Optimization. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007*, pages 742–756, Matshushima, Japan, March 2007. Springer. Lecture Notes in Computer Science Vol. 4403.

[62] Walking Fish Group. Hypervolume Test Data. `http://wfg.csse.uwa.edu.au/Hypervolume`.

[63] Walking Fish Group. IIHSO Source Code. `http://wfg.csse.uwa.edu.au/Hypervolume`.

[64] L. While. A New Analysis of the LebMeasure Algorithm for Calculating Hypervolume. In C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 326–340, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.

[65] L. While, L. Bradstreet, and L. Barone. A Fast Way of Calculating Exact Hypervolumes, 2010. To appear in *IEEE Transactions on Evolutionary Computation*.

[66] L. While, L. Bradstreet, L. Barone, and P. Hingston. Heuristics for Optimising the Calculation of Hypervolume for Multi-Objective Optimization Problems. In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, pages 2225–2232, Edinburgh, Scotland, September 2005. IEEE Press.

[67] L. While, P. Hingston, L. Barone, and S. Huband. A Faster Algorithm for Calculating Hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, February 2006.

[68] J. Wu and S. Azarm. Metrics for Quality Assessment of a Multiobjective Design Optimization Solution Set. *Transactions of the ASME, Journal of Mechanical Design*, 123:18–25, 2001.

[69] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[70] E. Zitzler. Hypervolume Metric Calculation. `ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/hypervol.c`, 2001.

[71] E. Zitzler, D. Brockhoff, and L. Thiele. The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicator Via Weighted Integration. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007*, pages 862–876, Matshushima, Japan, March 2007. Springer. Lecture Notes in Computer Science Vol. 4403.

[72] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

[73] E. Zitzler and S. Künzli. Indicator-based Selection in Multiobjective Search. In X. Y. et al., editor, *Parallel Problem Solving from Nature - PPSN VIII*, pages 832–842, Birmingham, UK, September 2004. Springer-Verlag. Lecture Notes in Computer Science Vol. 3242.

[74] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2002.

[75] E. Zitzler and L. Thiele. Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In A. E. Eiben, editor, *Parallel Problem Solving from Nature V*, pages 292–301, Amsterdam, September 1998. Springer-Verlag.

[76] E. Zitzler, L. Thiele, and J. Bader. On Set-Based Multiobjective Optimization. Technical Report 300, Computer Engineering and Networks Laboratory, ETH Zurich, February 2008.

[77] 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*, 7(2):117–132, April 2003.