

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 (IHSSO) 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 IHSSO, 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 Y , then X is taken to be a better set of solutions than Y .

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 $\Delta s = \text{Hypervolume}(f \cup \{p\}) - \text{Hypervolume}(f)$. 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 or 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 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.

# obj	# pts	Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
		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.

# obj	# pts	Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
		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.

# obj	# pts	Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
		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.

# obj	# pts	Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
		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.

# obj	# pts	Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
		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.

# obj	# pts	Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
		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.

# obj	# pts	Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
		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.

# obj	# pts	Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
		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.

# obj	# pts	Pt Reduction		Pt Addition		Local Search HV ratios		Random front selections
		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://sfbc.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