Bounded Archiving using the Lebesgue Measure

Joshua D. Knowles[1]
Dept of Chemistry, UMIST
PO Box 88, Sackville St
Manchester M60 1QD, UK
jknowles@ulb.ac.uk

David W. Corne[2]
Dept of Computer Science
University of Exeter
Exeter EX4 4QF, UK
d.w.corne@exeter.ac.uk

Mark Fleischer[3]
Johns Hopkins University
Applied Physics Laboratory
Laurel, MD 20723-6099, US
mark.fleischer@jhuapl.edu

Abstract- Many modern multiobjective evolutionary algorithms (MOEAs) store the points discovered during optimization in an external *archive*, separate from the main population, as a source of innovation and/or for presentation at the end of a run. Maintaining a bound on the size of the archive may be desirable or necessary for several reasons, but choosing which points to discard and which to keep in the archive, as they are discovered, is not trivial. In this paper we briefly review the state-of-the-art in bounded archiving, and present a new method based on locally maximizing the hypervolume dominated by the archive. The new archiver is shown to outperform existing methods, on several problem instances, with respect to the quality of the archive obtained when judged using three distinct quality measures.

1 Introduction

Many modern MOEAs, and some other multiobjective optimizers, make use of an external population or *archive* to store the points discovered during search. Viewing the set of points discovered as a sequence, ordered by the time of discovery, it is the job of an *archiver* to take this *input sequence*, chunk by chunk, and update the archive. In many applications, storing only nondominated points from the input sequence is sufficient to 'represent' the outcome of the search. Fieldsend *et al.* [2] have recently advocated storing *all* of the nondominated points, arguing that this improves convergence of an MOEA, and they have provided new data structures to enable fast update of such an archive. However, in general, storing only a subset of the nondominated points, say bounded by N, is often necessary or desirable because [10]: true nondominated sets may be infinitely large; the computational and memory overhead of maintaining the archive are reduced; and, harmful genetic drift (caused by over-representation of some regions of the search space) may be diminished (where the archive is used by the MOEA as a source of new points). This problem of *bounded archiving*, however, turns out to be non-trivial from both a theoretical and practical viewpoint. The problem can best be framed by the following important question: *Assuming we want to bound archive size, so that during the search it* never *exceeds N, how can this be achieved to the greatest advantage?*

To tackle this question we first make the following simplifying restriction. We assume that the points arising from search arrive *one by one*, as a sequence. The archiver must take the 'oldest' remaining point from this input sequence, compare it with the current archive and decide how to update the archive. So, at most $N + 1$ points may be 'seen' by the archiver at once, and at most N can be stored. Any point that is no longer in the input sequence or the archive cannot be recovered/remembered or used in any way, in future iterations. In generational MOEAs (which generate $popsize$ individuals at once) our assumption that the points arrive one by one is not generally true (and better archiving performance may be possible when $N + popsize$ points can be 'seen' at once) but we leave analysis of this case until a later time.

Returning to the question above, notice that "the greatest advantage" refers only to the collective properties of the points stored in the archive. Various properties of this archive are desirable, including: the membership of all extremal points from the input sequence; as large a cardinality as possible $\leq N$; the close approximation of the whole input sequence; the membership of only Pareto optimal points (w.r.t. the whole input sequence); and, various types of convergence properties. Knowles and Corne have recently analysed these properties in greater detail in [5], where some theoretical limitations are also derived.

These various goals have led to a number of approaches to bounded archiving. For a more detailed introduction to this see [10] and [5]. Most archivers[4] work in the same way when the archive's cardinality is below the bound: they accept each new point that is not dominated by the archive, discarding all those in the archive (if any) that become dominated by the new introduction. Once the archive is full, however, different strategies can be followed, leading to a number of distinct archivers. Rudolph and Agapie's archiver [11] only accepts new points that dominate one or more archive members. This leads to a certain type of provable convergence but inhibits the attainment of an extensive nondominated front and the improvement, over time, of the distribution of points. Other strategies are less restrictive and are able to accept nondominated points and discard a (nondominated) member of the archive to make way for it. The choice of point to discard is often based on estimating the density of points in objective space, and removing one that is crowded by nearby points. Knowles and Corne's adaptive grid archiving (AGA) algorithm [6] works in this way, using a histogram technique to estimate local density, while the ever-popular SPEA [13] makes use of clustering to truncate its external population.

Laumanns *et al.* [8] (see also [9]) proposed a quite different and more theoretically well-founded approach based on ϵ-approximate sets. In this, a point is accepted into the archive if and only if there is, already in the archive, no point that closely approximates it. The level of approximation is chosen at the outset, using up to k ϵ-parameters that define

[1] Work done while at IRIDIA, Université Libre de Bruxelles, Belgium

[2] Work done while at School of Systems Engineering, University of Reading, UK

[3] Work done while at Institute for Systems Research, University of Maryland, College Park, MD

[4] Though not the ϵ-based approaches introduced in [8]

the distance thresholds in each of the k objectives. This approach has provable convergence and approximation properties and is computationally cheap to run. It also bounds archive size but only as a function of the ϵ-parameter(s), and the (possibly unknown) ranges of the Pareto front (PF). If one wants to archive N points, no more and little fewer, then selecting ϵ appropriately can be problematic. A strategy for setting and adapting ϵ on-the-fly, in order to remove the necessity of selecting, *a priori*, appropriate ϵ-values, was also proposed in [8]. However, in this approach, ϵ cannot be reduced over time (because otherwise convergence and approximation properties are lost) — a restriction that means that far fewer points than desired may be archived in many cases. The ϵ-approximation set approach was also extended to an ϵ-Pareto approach in [8], in which only Pareto optimal points of the input sequence are archived, but this suffers from similar drawbacks. Overall, the approaches of Laumanns *et al.* are excellent when a decision maker (DM) really desires a given level of approximation (caring not about the number of points), but seem not to work so well when N points are required [5].

In this paper we present a new archiver for bounded archives, similar to one previously sketched in [7]. It is based on computing the hypervolume (or Lebesgue measure) of the region dominated by a set of (nondominated) points, given some bounding point. This measure was proposed by Zitzler in [12] (where it was called the $\mathcal{S}$ metric) for assessing the outcome of search. It has become a popular measure and was recently placed on a secure footing [14] by the observation that it is currently the only unary quality measure that is complete with respect to weak out-performance [4], while also indicating with certainty that one set is not worse than another (compatible with the $\not\triangleright$ relation [14]).

The new archiver works by performing steepest-ascent hillclimbing in the space of hypervolumes of the regions dominated by the evolving archive. This approach has been made practically realizable by a recent innovation due to Fleischer [3]: an algorithm for calculating the Lebesgue measure in polynomial time for general input length and number of objectives. Fortunately, not only does Fleischer's algorithm compute the hypervolume of an entire approximation set, it is also trivially adapted to compute efficiently the *change* in hypervolume when a single point is added or removed from a nondominated set, making it ideal for use in archivers.

In the remainder of this paper, we first review Fleischer's algorithm and present the new archiver, LAHC, derived from this. A series of experiments follow. The first set, using sequences of unique, nondominated points, is designed to establish if LAHC operates *ideally*: actually maximizing the bounded archive's hypervolume. In the second set of experiments, LAHC is run on more general sequences (including repeated and dominated points) like those arising from search, and its performance is compared with three other archivers. Finally, we summarise our findings and discuss LAHC's computational overhead.

2 Review of Fleischer's Algorithm

Fleischer's algorithm [3] for computing the Lebesgue measure of the region dominated by a set of nondominated points is based on the following observations. From any set of nondominated points (in any number of dimensions), one can

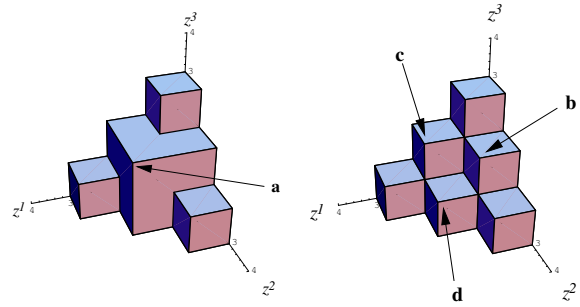

Figure 1: On the left, 4 points (with integer components) in a 3D space, dominating a region defined by a bounding point at (0,0,0). On the right, the point $\mathbf{a} = (2,2,2)$ has been removed, spawning points $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$. The region that has been 'lopped off' in this process is a rectangular polytope and its volume can be easily calculated. Repeatedly lopping off rectangular polytope regions is the basis of Fleischer's algorithm

easily identify and 'lop off' a dominated region that does not intersect with any other dominated region. Moreover, these dominated regions are rectangular polytopes so that their hypervolumes are trivial to compute. Therefore, by repeatedly lopping off such regions until none is left, and summing up over their hypervolumes, one can arrive at the hypervolume dominated by a set.

As an example, consider the four points:

$$(2,2,2) \quad (3,1,1) \quad (1,3,1) \quad (1,1,3),$$

which are illustrated in Figure 1, left. The four points represent objective vectors, in a maximization problem, dominating a region that is bounded from below by a lower bounding point, $\mathbf{lb} = (0,0,0)$. To compute the volume contribution of the point $\mathbf{a} = (2,2,2)$, one can remove it, replacing it with the three 'spawned' points shown in Figure 1, right. These are the three points:

$$\mathbf{b} = (1,2,2) \quad \mathbf{c} = (2,1,2) \quad \mathbf{d} = (2,2,1),$$

obtained by taking the point $\mathbf{a}$ and replacing each of its components $i = 1..k$ by a new value v_i. The value v_i is the nearest value lower (respectively higher in a minimization problem) than $\mathbf{a}$'s ith component, in a list of the ith components of all the points, including the lower bounding point. The volume 'lopped off' by this removal and replacement is given by the volume of the rectangular polytope:

$$(a_1 - b_1)(a_2 - c_2)(a_3 - d_3) = 1.0.$$

Now the same process is repeated for each of the spawned points. Taking point $\mathbf{b}$ first, we remove it and replace it with the spawned points:

$$\mathbf{e} = (0,2,2) \quad \mathbf{f} = (1,1,2) \quad \mathbf{g} = (1,2,1).$$

But each of these points is either dominated or it contains a component that is equal to the corresponding component of the bounding point, $\mathbf{lb}$. Thus these three points each contribute nothing to the hypervolume of $\mathbf{a}$ and can be ignored from further consideration.

Algorithm 1 $\Delta_{\mathcal{S}}(\mathbf{z}, Z, \mathbf{lb})$

```
Δ_S ← 0
len ← |Z|
push(z, Z)        //  |Z| is now len + 1
while |Z| > len do
   LopOffVol ← 1
   y ← pop(Z)
   for i ← 1..k do
      v_i ← GetBoundValue(i, y, Z)
      LopOffVol ← LopOffVol × (y_i − b_i)
      s ← SpawnVector(y, i, v_i)
      if Z ≯ s ∧ s_i ≠ lb_i then
         push(s, Z)
      end if
   end for
   Δ_S ← LopOffVol
end while
return Δ_S
```

// where $\mathrm{GetBoundValue}(i, \mathbf{y}, Z)$ returns the nearest value to the ith component of $\mathbf{y}$ from a list containing the ith components of the lower bounding point and of all the vectors in Z

// and $\mathrm{SpawnVector}(\mathbf{y}, i, v_i)$ returns the vector $\mathbf{y}$ with its ith component replaced by v_i.

From these three points we can calculate the contribution of $\mathbf{b}$ to $\mathbf{a}$'s contribution, as follows:

$$(b_1 - e_1)(b_2 - f_2)(b_3 - g_3) = 1.0.$$

The same process is repeated for $\mathbf{c}$ and $\mathbf{d}$, and it can be seen that the total contribution of $\mathbf{a}$ to the dominated region is 4.0, in this example.

Generalizing from the above example, Fleischer is able to give us an efficient and simple procedure for computing the hypervolume of any set of nondominated points; and the algorithm works equally well for maximization or minimization problems.

In Fleischer's algorithm, points are initially pushed onto a (LIFO) stack. When the first point is popped off the stack, it spawns k other points (where k is the number of objectives). Each of these (in no particular order) is then pushed onto the stack if and only if it is not dominated by any point in the stack, *and* it does not have a component equal to the bounding point's corresponding component. Otherwise it is discarded. The algorithm continues by again popping the next point off the stack. Upon each pop, the volume of the rectangular polytope that is lopped off is computed and added to a running total hypervolume.

Fleischer has shown that the total number of points in the stack can never be greater than $Size + k - 1$, where $Size$ is the stack's initial depth — since for any spawned point, all of its spawns except one are necessarily dominated by the remaining contents of the stack [3]. Therefore, despite the fact that spawned points can spawn points, and so on, the total number of points in the stack remains tightly bounded, leading to a worst-case complexity for the algorithm of $O(Size^3 k^2)$.

If Fleischer's algorithm is run until there are no points left in the stack, then the hypervolume of the entire set of points is computed. But for the purposes of archiving we would like to compute just the hypervolume *contribution* of a single point to the set. Fleischer's algorithm is ideally suited for this job too, as our earlier example illustrated. Given a set of nondominated points already in the stack, and a new nondominated point, whose hypervolume contribution we would like to know (were it added to the set), we need merely to push on the new point and run Fleischer's algorithm, terminating it when the depth of the stack is what it initially was (since this means the new point and all its spawns, spawns of spawns etc., have been removed), and all other points remain. This algorithm, $\Delta_{\mathcal{S}}(\mathbf{z}, Z, \mathbf{lb})$, is shown in Algorithm 1. It computes the contribution of a point $\mathbf{z}$ to a set Z, where Z contains only unique and mutually nondominated points, and the point $\mathbf{z}$ is also unique and nondominated in Z. For simplicity, we imagine Z is a stack already, and can accommodate the new point. We also assume a maximization problem and the point $\mathbf{lb}$ is the lower bounding point of the dominated region. (All points must have positive measure w.r.t. this bounding point). For more precise and extensive details of the procedures described above, the reader is referred to [3].

Algorithm 2 $\mathrm{LAHC}(S, N)$

```
i ← 0
A ← ∅
while i < |S| do
   if z_i ∉ A then
      if z_i ≻ A then
         A ← ND(A ∪ {z_i})
      else if z_i ∼ A then
         A ← A ∪ {z_i}
         if |A| > N then
            A ← A \ {argmin_{z∈A} [Δ_S(z, A)]}
         end if
      end if
   end if
   i++
end while
return A
```

// where $\mathcal{ND}(A)$ returns the nondominated points from A

// and $\mathbf{z}_i$ is the ith point from the sequence S.

3 Lebesgue Archiving Hillclimber LAHC

In the last section, we reviewed Fleischer's algorithm and showed that it can be used to efficiently calculate the hypervolume contribution of a single point to a set of points. This means that, given $N+1$ points in a set, we can compute the set of N points that maximizes the hypervolume. (To do this we must merely compute the contribution of each of the $N+1$ points to the hypervolume and retain all but the one contributing the least). A straightforward approach to this takes $O(N^3 k^2)$ time.

So, given a sequence of points S to be archived, and an archive A of capacity N, i.e. $|A| \leq N$, the principle behind the Lebesgue archiving hillclimber (LAHC) is to remove the point in A that contributes least to the hypervolume, whenever the archive is full and there is a new point available to be added. More precisely, the next point in the

sequence is archived in A (provided it is unique) if it: (i) dominates one or more points in A (in which case the dominated points in A are removed); or, (ii) it is nondominated in A and $|A| < N$; or, (iii) it is nondominated in A and $|A| = N$ and it is not the minimum contributing point to the hypervolume (in which case the minimum contributing point is removed). LAHC is defined in Algorithm 2.

4 Does LAHC Maximize Hypervolume?

How good is the LAHC algorithm presented above? *Ideally*, we might hope that LAHC, given any input sequence S, always obtains an archive of cardinality $\min(N, |\mathcal{ND}(S)|)$ that maximizes the hypervolume of the region dominated, over all such subsets of the sequence. However, on general input sequences (i.e. those including dominated points), we cannot expect this to happen since it has been proved in [5] that *no* archiver can maintain $\min(N, |\mathcal{ND}(S)|)$ points on general input sequences.

However, if we restrict our attention only to sequences of unique, mutually nondominating points then it *is* reasonable to wonder whether LAHC always obtains the set of points of cardinality N that maximizes the hypervolume over all such sets drawn from the input sequence. Supposing it did, one consequence (useful for testing the hypothesis) would be that if presented with any permutation of the same input sequence, this would result in the same hypervolume of the objective space being dominated.

To test the hypothesis that LAHC is ideal in this sense, we use sequences of points generated uniformly randomly (using polar co-ordinates) on the surface of a hypersphere of radius 1. Experiments are carried out on different sizes of sequence, number of objectives, and different archive sizes: archive size is set to 10%, 50%, and 90% of the input sequence length, for each of three different sequence sizes of two and three objectives. For each triple ($|S|$, k , N), we do 30 independent runs of LAHC, each time first permuting the input sequence randomly.

In Figures 2 and 3, boxplots indicating the hypervolume attained over these 30 runs are shown for each instance. From these it is clear that LAHC is *not* ideal in the sense defined above: there is some fluctuation in the hypervolume dominated by the archive when the input sequence is permuted. This result shows that there *are* local optima in the space of hypervolumes, when a 'single-swap neighbourhood' is used to move between point sets (archives).

Nonetheless, the fluctuation in LAHC's performance seems very small. To estimate more accurately how far from ideal the LAHC is, we also compared its performance with three other methods of selecting a subset of size N from the input sequence, described next.

Agglomerative clustering (AGG)

The first method is agglomerative clustering based on the average link — the method used in SPEA to reduce the size of its external population. In our experiments, clustering was applied to the entire input sequence, not iteratively. Thus, unlike LAHC, it was not constrained to maintain only N points at any time. Instead, it was given the opportunity of selecting the best N points, using all available information. The clustering method is deterministic, independent of sequence order, so only one run was carried out on each instance.

Random subsets (RAN)

The second method used to provide a lower bound on performance was to select randomly a subset of N points, R times, calculate the hypervolume of each subset, and return the best. The method is clearly stochastic, so 30 runs were carried out. $R = 1000$ was used in the experiments.

Simulated annealing (SA)

Finally, to provide an upper bound on what is possible with a point-by-point archiver (i.e. one that only maintains N points and can consider only one new point at a time, never revisiting previous points), we used simulated annealing to improve the set of points found by LAHC. First, LAHC is run on the input sequence. Then, starting from this archive, each iteration of the SA proceeds as follows. A point from the input sequence, and not in the archive, $\mathbf{z}_{new}$, and a point in the archive $\mathbf{z}_{in}$ are selected uniformly at random. The new point $\mathbf{z}_{new}$ is added to the archive and the contributions of both $\mathbf{z}_{new}$ and $\mathbf{z}_{in}$ are calculated. If $\Delta_{\mathcal{S}}(\mathbf{z}_{in}, A, \mathbf{lb}) \leq \Delta_{\mathcal{S}}(\mathbf{z}_{new}, A, \mathbf{lb})$ then $\mathbf{z}_{in}$ is discarded and the new point is accepted. On the other hand, when $\Delta_{\mathcal{S}}(\mathbf{z}_{in}, A, \mathbf{lb}) > \Delta_{\mathcal{S}}(\mathbf{z}_{new}, A, \mathbf{lb})$, the Metropolis criterion is used to determine the probability of accepting this 'move'. If the move is rejected, the new point is discarded from the archive.

A geometric cooling schedule was used with the temperature T being reduced every iteration. The SA was run for 50,000 iterations with the initial temperature calculated to give a probability of 0.5 to accept a move that decreases the hypervolume by 0.01%. At the final temperature, moves that decrease the hypervolume by 0.001% are accepted with probability 0.5. SA was run ten times, permuting the input sequence each time (so that the starting solution coming from the LAHC may be different).

Results of the comparison

Looking at all of the boxplots in Figures 2 and 3, it is clear that LAHC performs strongly. When the archive size N is large compared to the input sequence length (3rd column), it performs nearly as well as simulated annealing, in some cases *as* well. When the archive size is relatively smaller, there is some fluctuation in the hypervolume achieved but its performance is still much closer to the SA than to the much weaker clustering algorithm. As input sequence length increases (lower rows), for a given fixed percentage of points archived, there is no clear change in the relative performance of LAHC. Similarly, LAHC, performs equally well in the 2 and 3 objective cases.

Other results not shown

The results presented in the boxplots above have been supplemented by some limited testing on 4-objective instances of hypersphere sequences, and by testing on sequences of unique, nondominated points not on the surface of a hypersphere. These results are not shown here but in all cases the results are similar to those given above. These results can be obtained by email to the first author.

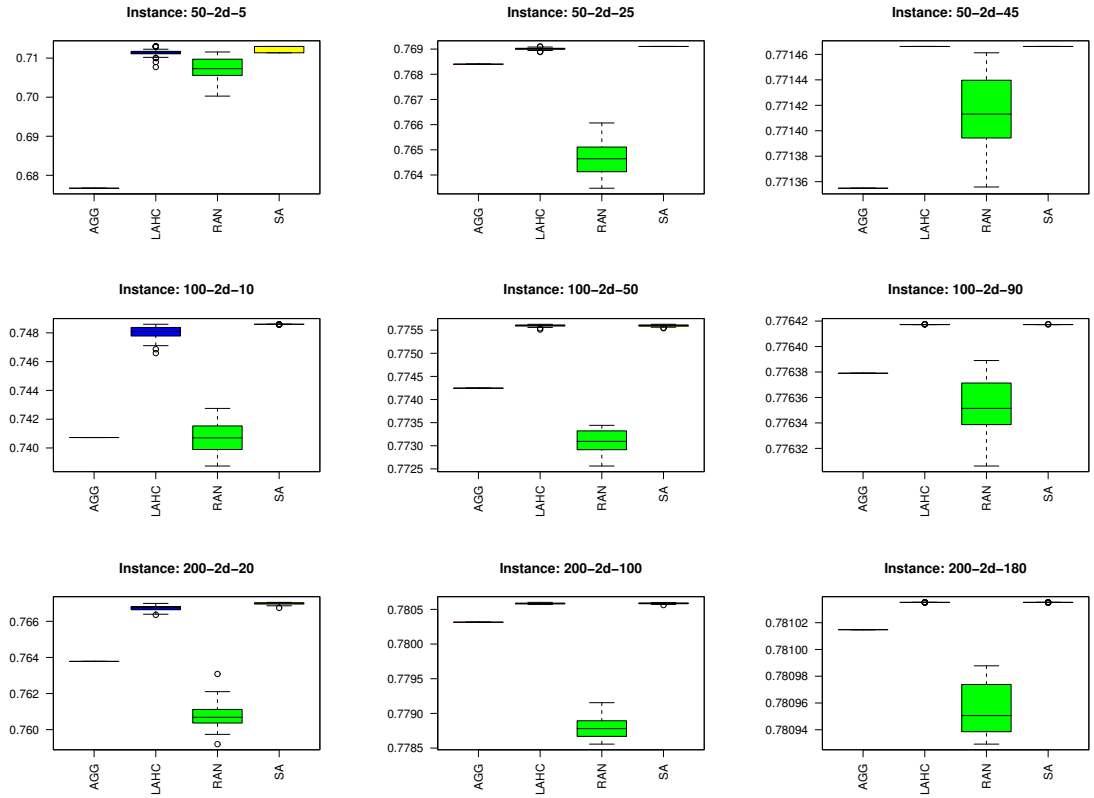

Figure 2: Boxplots showing the hypervolume of the dominated region obtained by each method on instances of $k = 2$ objectives with different input sequence lengths $|S|$ and archive capacities N. The names of instances are of the form $|S|$–kd–N. The central line in a boxplot is the median, the box height is the IQR and the whiskers extend to observed values up to 1.5 times the IQR. Outliers are shown as circles

5 Comparisons on General Point Sequences

In the last section we observed that on sequences of unique and mutually nondominated points, LAHC is able to maintain an archive of size N that is close to the ideal, in terms of maximizing the hypervolume of the dominated region. Sequences of points arising from an optimization process, such as an MOEA, however, generally contain repeated points and points that dominate each other. On these more general sequences, how does LAHC perform?

In this section, we consider a number of different sequences, each difficult for an archiver for different reasons, and compare the performance of LAHC with other state-of-the-art archivers. To compare the performance we use three different measures of approximation set quality.

In the following we describe each archiver and each performance measure used. Then each input sequence is described, with all results presented in Table 1. For all experiments the archive's capacity was set to $N = 20$. A bounding point is needed for both the ϵ-based archivers and for all three of the measures used. To choose this, we use our knowledge of the true PF to set the bounding point such that all points in the true PF have positive measure.

Archiver: Adaptive grid archiving algorithm (AGA)

AGA has been described in detail in [6]. It uses a histogram approach to density estimation — the objective space covering the points 'seen' by the archiver is divided up into grid regions of equal size. Points in crowded regions are susceptible to being removed from the archive to allow entry of new nondominated points. AGA is nondeterministic, so 10 runs were used for each sequence and all results reported are the mean measure over the ten runs.

Archiver: ϵ-Pareto set archiver

The archiver based on the ϵ-Pareto set is described in [8]. In order for the archiver to operate, one or more ϵ-parameters must be first given. Here we use the multiplicative version of ϵ-approximation and give one value only, which is used in each objective dimension. We run the archiver five times, trying to find the best value of ϵ such that the archive size never exceeds N but is as close as possible to N. Results are given for the best run only.

Archiver: Adaptive ϵ-approximate Pareto set archiver

The archiver based on the ϵ-approximate Pareto set is described in [8]. We use here the adaptive version that does not require us to set ϵ beforehand. Instead, ϵ is calculated 'on the fly' by using the current ranges of the points in the archive. This adaptation policy can never reduce ϵ, however, which leads to problems when the PF is small compared to the whole objective space. The algorithm is deterministic so only one run is necessary.

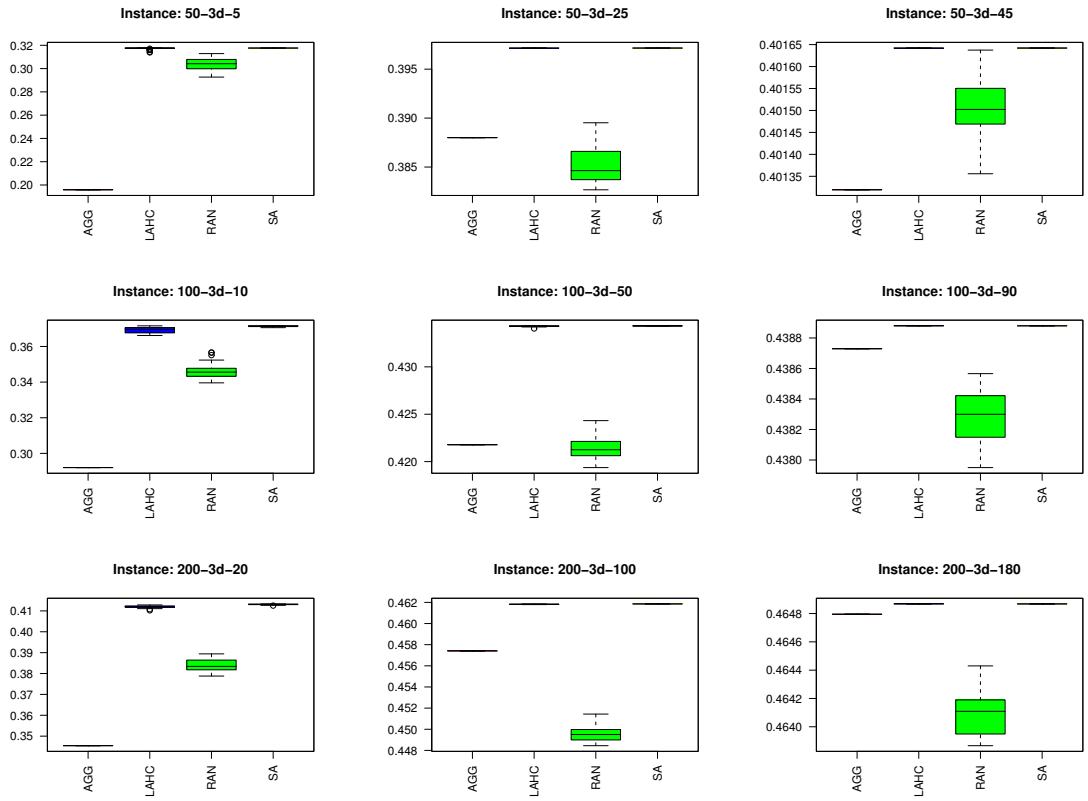

Figure 3: Boxplots showing the hypervolume of the dominated region obtained by each method on instances of $k = 3$ objectives with different input sequence lengths $|S|$ and archive capacities N. The names of instances are of the form $|S|$–kd–N

Measure: absolute Lebesgue measure ($\mathcal{S}(A)$)

This is commonly known as the $\mathcal{S}$ metric of Zitzler [12]. This measure is exactly what LAHC is trying to maximize.

Measure: binary ϵ-indicator ($\epsilon(A, B)$)

This measure (described in [14]) takes two approximation sets and compares them. The measure indicates the factor by which one set's points must be multiplied such that all of them 'cover' the points in the other set. Both $\epsilon(A, B)$ and $\epsilon(B, A)$ are considered. In the case where $\epsilon(A, B) > 1$ and $\epsilon(B, A) \leq 1$ then approximation set A is strictly better than B.

Measure: unary ϵ-indicator ($\epsilon_1(A)$)

This measure (described in [14]) is equivalent to $\epsilon(A, \mathrm{PF})$. That is, it measures the factor by which the points in an approximation set A must be multiplied in order for the whole PF to be covered.

Sequence I: small PF

The first sequence has a small PF compared with the extent of the whole objective space. This can cause problems particularly to the adaptive ϵ-approximate archiver. Consider the two-objective maximization problem described by:

$$\begin{aligned} f_1 &= (1-r) + r\sin(v\pi/2) \\ f_2 &= (1-r) + r\cos(v\pi/2), \text{ where} \\ r &= 0.01 + 0.99u^{1/10} \end{aligned}$$

and where u and v are two decision variables in the interval [0,1). Here, the radius, r, or the PF is very small compared to the biggest and smallest values in the objective space. We used 40 bits each to encode u and v and made a sequence of points by running the search algorithm PESA-II [1] on this function for $10,000$ evaluations. The sequence has 1824 Pareto optima of which 790 are unique.

Sequence II: three-objective problem with small PF

The second sequence is similar to the first but in three dimensions. This can cause problems to AGA because it can get into a cyclic behaviour where the same points are admitted and discarded from the archive repeatedly. Consider the 3-objective optimization problem:

$$\begin{aligned} f_1 &= (1-r) + r\cos(v\pi/2)\cos(w\pi/2) \\ f_2 &= (1-r) + r\cos(v\pi/2)\sin(w\pi/2) \\ f_3 &= (1-r) + r\sin(v\pi/2), \text{ where} \\ r &= 0.01 + 0.99u^{1/10} \end{aligned}$$

and where u, v and w are decision variables in the interval [0,1). We used 30 bits to encode each variable (u, v, and

	Measure	Archives produced by				
		$\mathcal{ND}$	LAHC	AGA	ϵ-Pareto	ϵ-approximate
Sequence I	$\lvert . \rvert$	1824	19	**20**	**20**	1
	$\mathcal{S}(.)$	0.000078426	**0.000076403**	0.000075571	0.000067285	N/A (see caption)
	$I_\epsilon(\text{LAHC}, .)$	–	–	1.030565394	**1.038158505**	N/A (see caption)
	$I_\epsilon(., \text{LAHC})$	–	–	1.055535693	**1.028020776**	N/A (see caption)
	$I_{\epsilon 1}()$	–	1.038171201	1.059230128	**1.037876042**	N/A (see caption)
Sequence II	$\lvert . \rvert$	65	19	**20**	19	5
	$\mathcal{S}(.)$	0.968080978	**0.964729636**	0.938582354	0.847524436	0.742310914
	$I_\epsilon(\text{LAHC}, .)$	–	**–**	1.011198133	1.039126209	*0.999732999*
	$I_\epsilon(., \text{LAHC})$	–	**–**	1.114107929	1.161638329	*1.270885307*
	$I_{\epsilon 1}()$	–	**1.083754288**	1.123195531	1.161638329	1.281251718
Sequence III	$\lvert . \rvert$	50	19	**20**	11	4
	$\mathcal{S}(.)$	23075.06598	**23071.23324**	23054.98198	23002.62088	21387.79461
	$I_\epsilon(\text{LAHC}, .)$	–	**–**	1.002219636	1.003942747	*1.0*
	$I_\epsilon(., \text{LAHC})$	–	**–**	1.011380203	1.008386320	*1.163402272*
	$I_{\epsilon 1}()$	–	**1.005833388**	1.011380203	1.009657601	1.168966854
Sequence IV	$\lvert . \rvert$	239	**20**	**20**	**20**	4
	$\mathcal{S}(.)$	0.237551138	**0.235358351**	0.228839624	0.230301490	0.190955751
	$I_\epsilon(\text{LAHC}, .)$	–	**–**	1.047078098	1.036835631	1.009708834
	$I_\epsilon(., \text{LAHC})$	–	**–**	1.303809817	1.085867009	1.437271827
	$I_{\epsilon 1}()$	–	**1.047078098**	1.303809817	1.085867009	1.437271827

Table 1: All results from the experiments with the four archivers over the four sequence sets. Bold face is used to indicate the best result in a row (excluding the nondominated filter $\mathcal{ND}$). For the binary epsilon indicator, where LAHC is compared with the other archivers, the upper row indicates the factor by which the points in LAHC's archive must be multiplied so as to dominate the column algorithm's archive, while the lower row indicates the converse. Thus, smaller numbers on the upper row indicate LAHC's archive is 'better', though it is not better in a strict sense unless the upper row entry is also ≤ 1. Bold face indicates the best archive in the weak sense, and italics are used to indicate a result where LAHC's archive is strictly better. Note that for Sequence 1, results for the ϵ-approx archiver are not available because the point it found had negative measure with respect to the lower bounding point used for computing these measures

w) and generated a sequence by running PESA-II on the problem for 1500 evaluations. This gave a sequence with 65 unique Pareto optima.

Sequence III: highly non-uniform spacing in the PF

Consider a sequence of nondominated points where some pairs are spaced much (orders of magnitude) closer together than others. This tests an archiver's ability to keep points well-separated, even when the points in the input sequence are not evenly spaced.

We generated a sequence of 500 points using the functions:

$$\begin{aligned} f_1^{(t)} &= 2^{2^{3x}} + \mathcal{U}^{(t)}(10), \text{ and} \\ f_2^{(t)} &= 257 - 2^{2^{3x}} + \mathcal{U}^{(t)}(10), \text{ where} \\ x^{(t)} &\text{ is uniformly at random} \in \{0.0, 0.1, 0.2, \ldots 1.0\}, \end{aligned}$$

and $\mathcal{U}(y)$ is a uniformly random variate in $[0, y)$.

Sequence IV: a discontinuous PF

In sequence IV, the points form a PF with three isolated regions. This sequence is particularly difficult for AGA since the regions are small compared to the whole extent of the PF, thus many points lie within each grid region used by AGA, hindering its ability to obtain a good distribution at the fine level.

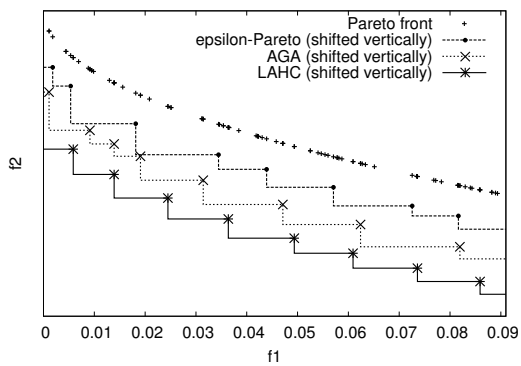

Figure 4: Points discovered by the three best algorithms on part of Sequence IV's PF. N.B: The points discovered by each algorithm have been shifted vertically to aid viewing; in reality they all lie on the PF. The more even spacing achieved by LAHC is clearly visible

The sequence is made up of 300 points generated using:

$$\begin{aligned} f_1^{(t)} &= v^{(t)} + \mathcal{U}^{(t)}(0.1) \\ f_2^{(t)} &= 1 - \sqrt{f_1^{(t)}}, \text{ where} \\ v^{(t)} &\text{ is selected uniformly at random from } \{0, 0.5, 0.9\} \end{aligned}$$

Visual results of the comparison are given in Figure 4.

6 Summary and Discussion

The Lebesgue measure of the region dominated by a nondominated set is a theoretically well-founded unary estimate of the ND set's overall quality. In this paper, a new archiver, LAHC, was presented, that uses the Lebesgue measure to perform steepest-ascent hillclimbing in the space of archive sets with a specific bound. Results when applying this technique to nondominated sequences of points showed that steepest-ascent hillclimbing, point by point, is very close to optimal in terms of arriving at the archive of size N that maximizes the Lebesgue measure over all possible such subsets of an input sequence. On more typical input sequences, such as those arising from an MOEA optimization, LAHC outperformed each of adaptive grid archiving (AGA) and two ϵ-based approaches of Laumanns *et al.* in terms of the quality of archive set obtained, given identical input sequences.

Performance in terms of archive quality is not the only factor in assessing practical archivers, however. Computational cost must also be considered. Fleischer's algorithm is an efficient method of computing changes to the Lebesgue measure when one point is added to or removed from a set. Nonetheless, it remains expensive to find, in a set, the point that is contributing the least to the size of the dominated region — and this has to be done at every step of LAHC (since the inclusion of just one new point can affect the contributions of all other points in the archive, except in the 2-d case). In fact, the computation time for the 2 and 3 objective problems considered in this paper are not high. However, when moving to 4 objectives we have noticed a significant increase in computation time, such as would make our current implementation impractical in all but the most computationally demanding of applications. The reason for this large step up in computation time needs further investigation. No doubt some of it is due to the fact that no attempt at efficiency has been made in our implementation (so that the actual worst-case complexity of our algorithm is worse than the theoretical worst-case), but a further reason is that as objective space dimension increases, more and more points are nondominated so that spawned points 'hang around' for much longer in the archive. This effect compounds over to the spawns of spawns and so on, so that computation time is much closer to the worst case than it is for the two and three objective sequences. It is unclear at present to what extent this problem can be alleviated by advanced bookkeeping and the use of efficient data structures for storing and querying nondominated sets but we believe that any improvement in the 4-objective case will carry over to still higher dimensions, since in the 4-d case the depth of the stack remains close to the worst case already. Our immediate future work will investigate this further.

Acknowledgments

JK gratefully acknowledges the support of a CEC Marie Curie Fellowship, contract no.: HPMF-CT-2000-00992 (to September 2003) and a BBSRC David Phillips Fellowship (from October 2003). Thanks also to Julia Handl for Figure 1 and for the implementation of average link clustering.

Bibliography

[1] David W. Corne, Nick R. Jerram, Joshua D. Knowles, and Martin J. Oates. PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 283–290. Morgan Kaufmann Publishers, 2001.

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

[3] Mark Fleischer. The measure of Pareto optima: Applications to multi-objective metaheuristics. In Carlos M. Fonseca *et al.*, editor, *Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2003*, LNCS No. 2632, pages 519–533. Springer, 2003.

[4] Michael Pilegaard Hansen and Andrzej Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Institute of Computing Science, Poznan University of Technology, Poznan, Poland, March 1998.

[5] Joshua Knowles and David Corne. Bounded Pareto archiving: Theory and practice. In X. Gandibleux, M. Sevaux, K. Sorensen, and V. T'kindt, editors, *Multiple Objective MetaHeuristics*, LNEMS. Springer, 2004. (To appear). Available on request from jknowles@ulb.ac.be.

[6] Joshua Knowles and David Corne. Properties of an Adaptive Archiving Algorithm for Storing Nondominated Vectors. *IEEE Transactions on Evolutionary Computation*, 7(2):100–116, April 2003.

[7] Joshua D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD thesis, University of Reading, UK, 2002.

[8] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. On the Convergence and Diversity-Preservation Properties of Multi-Objective Evolutionary Algorithms. Technical Report 108, TIK, ETH, Zurich, Switzerland, May 2001.

[9] Marco Laumanns, Lothar Thiele, Eckart Zitzler, and Kalyanmoy Deb. Archiving with Guaranteed Convergence and Diversity in Multi-Objective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 439–447. Morgan Kaufmann Publishers, 2002.

[10] Marco Laumanns, Eckart Zitzler, and Lothar Thiele. On the Effects of Archiving, Elitism, and Density Based Selection in Evolutionary Multi-objective Optimization. In *First International Conference on Evolutionary Multi-Criterion Optimization*,LNCS No. 1993, pages 181–196. Springer, 2001.

[11] Günter Rudolph. Evolutionary Search for Minimal Elements in Partially Ordered Finite Sets. In *Evolutionary Programming VII, Proceedings of the 7th Annual Conference on Evolutionary Programming*, pages 345–353. Springer, 1998.

[12] Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH, Zurich, Switzerland, November 1999.

[13] Eckart Zitzler and Lothar Thiele. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, TIK, ETH, Zurich, Switzerland, May 1998.

[14] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.