Bounded Pareto Archiving: Theory and Practice

Joshua Knowles[1] and David Corne[2]

[1] IRIDIA - CP 194/6, Université Libre de Bruxelles, 1050 Brussels, Belgium.
E-mail: jknowles@ulb.ac.be Web: http://iridia.ulb.ac.be/~jknowles/
[2] School of Systems Engineering, University of Reading, Reading RG6 6AY, UK.
E-mail: d.w.corne@exeter.ac.uk

Abstract. We consider algorithms for the sequential storage, or 'archiving', of solutions discovered during a Pareto optimization procedure. We focus particularly on the case when an a priori hard bound, N, is placed on the capacity of the archive of solutions. We show that for general sequences of points, no algorithm that respects the bound can maintain an ideal *minimum* number of points in the archive. One consequence of this, for example, is that a strictly bounded archive cannot be expected in general to contain the Pareto front of the points generated so far (where this set is smaller than the archive bound). Using the notion of an ideal ϵ-approximation set—the subset (of size $\leq N$) of a whole sequence of points which minimizes ϵ—we also show that no archiving algorithm can attain this ideal for general sequences. This means that in general no archiving algorithm can be expected to maintain an 'optimal' representation of the Pareto front when the size of that set is larger than the archive bound. Furthermore, if the ranges of the PF of the sequence are not known a priori, no algorithm that certifies (using its own internal epsilon parameter ϵ_{arc}) that it maintains an epsilon-approximate set of the sequence, can maintain ϵ_{arc} within any fixed multiple of the minimal (ideal) ϵ value. In a case study we go on to demonstrate several scenarios where ϵ-based archiving algorithms proposed by Laumanns *et al.*—which perform well when the archive's capacity is *not* a priori bounded—perform poorly when ϵ is adapted 'on the fly' in order to respect a capacity constraint. For each scenario we demonstrate that the performance of an adaptive grid archiving (AGA) algorithm (which does *not* assure a formally guaranteed approximation level) performs comparatively well, in practice.

1 Introduction

The goal of Pareto optimization is to obtain a complete set of Pareto optimal solutions to a problem, i.e., every solution whose image in objective space is globally nondominated. Unfortunately, this goal may be difficult to achieve in practice, even when a search algorithm effectively concentrates its samples on or near the global Pareto front, because the number of Pareto solutions may be prohibitively large. In this case, it is impractical, and often unnecessary, to store all of the nondominated solutions (or even their objective space images, called 'points' in this article) discovered during an optimization process. Thus, we have recently become accustomed to the notion of maintaining an 'archive' (a subset) of these discovered nondominated points. Indeed, it is now recognized,

by all of the research communities involved in multiobjective optimization, that maintaining archives of nondominated points is a very important issue.

We can identify two main explanations of this importance, as follows. First, and most obviously, the final contents of an archive represent (usually) the results returned by the optimization process. This leads to consideration of the distribution and cardinality of points within it, since we normally require this result to be a good approximation to the true Pareto front. Second, it is common (and highly effective) to employ the archive as a pool from which to guide the generation of new points. Some algorithms use points in the archive exclusively for this purpose, while others tend to rely on the archive to varying degrees according to parameter settings.

Both of these aspects, and also the computational complexity of maintaining the archive itself (checking for nondominance of newly generated points), lead to the requirement that the archive should strive to be of a bounded and relatively modest size. As far as the archive as a result is concerned, the issue is that an unlimited number of nondominated points may be obtained during the optimization process, and memory becomes an issue. Concerning the use of the archive during the optimization process, and also the processes involved in its maintenance, computational complexity becomes an issue, especially in near real-time applications. Since archives should therefore generally be bounded in size (by, say, N), it is therefore of great interest to examine the properties of algorithms which maintain bounded archives.

In this article we inquire about the properties of an imaginary *ideal* archiving algorithm and investigate the extent to which these properties can be, in principle, attained. Some of the most important properties relate to whether the archive converges in some well-defined sense. In our analysis we use the idea that some generating process (e.g a Pareto optimization procedure) is sending points, one by one, to the archiver to be stored. We call the sequence of these points an *input sequence*. Now, if we consider the archiving algorithm as generating a sequence of archives, one new archive as each new point from the input sequence is considered, one type of convergence is whether this sequence of archives converges to a Pareto subset of the input sequence. Another type of convergence occurs when the sequence of archives converges to an ϵ-approximate set [7] (explained below) of the input sequence. A third, slightly different type of convergence occurs if the sequence of archives simply stops changing eventually, provided that the input sequence is drawn from a finite set, and each point in it is encountered an unbounded number of times. Other desirable properties of archiving algorithms include the maintenance of a *minimum* number of points in the archive and the storage of all of the nondominated extrema of the input sequence.

In this article we also pay attention to the situation, commonly encountered, when prior to optimization we do not know the ranges of values that the nondominated points discovered during optimization will take (in at least one of the objectives). Although it *may* be possible to conduct preliminary experiments to establish bounds, this is not generally the case. Thus, in practice we may often be faced with a situation where we are completely ignorant of Pareto front

ranges and have only a single algorithm run from which to collect our solutions. For ease of statement, we call this situation a 'blind, one-shot' scenario.

Whether in a 'blind, one-shot' scenario or not, we would like our archiving algorithms to obtain an archive that approximates the sequence of points generated by the search. Seminal work on the subject of archiving by Laumanns *et al.* first reported in [7] (and later published in [8] and [9]) proposed the concept of ϵ-approximation as a means of characterizing *quantitatively* how closely one set (the archive) approximated another set (the sequence of points generated during the search). This notion of the quality of approximation is valuable because it gets away from the less well-defined and troublesome concepts of sets, such as being 'evenly-distributed' or 'well-spread'.

Two archiving algorithms making use of the ϵ-approximation concepts were also described in [7]: both have provable convergence properties, guarantee a certain quality of approximation, and at the same time ensure that the archive's size is bounded by some function of ϵ (a parameter set by the user) and the extent of the objective space. These algorithms are, in a sense, ideal when the user doesn't require an a priori hard limit N on the archive's capacity. But, when applying a search algorithm to an unknown problem, there can be no good guide to setting an appropriate ϵ value. With the 'wrong' choice too many or too few points may be archived, with a consequent effect on algorithm performance and/or user satisfaction. Seeing this 'weakness' in the proposed approaches, Laumanns *et al.* also described two methods for achieving size-bounded archives of the kind we focus on in this article. In one of these (effectively, for a 'blind, one-shot' scenario), ϵ is adapted 'on the fly' by measuring the range of values observed in the points encountered. Unfortunately, such an approach is over-conservative in many situations, leading to archives that are *too small*, as we later demonstrate. It is a main thesis of this article that other methods—those that do *not* assure a formally guaranteed ϵ-approximation level—may perform much better in practice, when a capacity bound must be respected, and especially under a 'blind, one-shot' scenario.

Such an alternative archiving approach, which seems better when archive capacity is limited (although being far from ideal), is adaptive grid archiving (AGA), as described in [6] and [5]. This algorithm does *not* have guaranteed convergence, in any of the three senses mentioned earlier, and can only be shown to 'settle down' when certain properties of the objective space are respected [5]. This may be considered a serious weakness because the quality of the archive achieved becomes a function of time, even when the whole of a problem space is sampled an unbounded number of times. Nonetheless, we believe that when archive capacity is limited, and especially in 'blind, one-shot' scenarios, it often gives more desirable results than the ϵ-based methods of Laumanns *et al.* To illustrate this, we provide a case study that makes use of several different sequences of points designed to be difficult for archiving algorithms. These sequences include a small Pareto front in a large objective space, a discontinuous Pareto front, and a highly non-uniformly spaced Pareto front. These sequences pose problems for each of the archiving algorithms tested here, but it is our finding that AGA copes much better than the ϵ-approximate methods in these situations.

The remainder of this article is organized as follows. Section 2 provides the essential mathematical framework needed to understand the archiving algorithms discussed in this article. Section 3 goes on to propose several properties of a theoretical, ideal archiving algorithm and goes on to show that some of these cannot be achieved in reality, when there is a capacity constraint on the archive. These theorems set the scene for Sections 4 and 5 where the two main archiving algorithms considered here, are respectively described and tested on a number of point sequences. Section 6 concludes with a discussion of our findings.

2 Preliminaries

In the following we assume, without loss of generality, that we are maximizing objective values. We further assume that all points are positive: $y, z \in Z, Z \subseteq \Re_+^k$. Throughout the paper we will assume that we do not care about keeping different solutions that map to the same point in objective space. Thus, our optimality definitions (below) and archiving algorithms are cast only in terms of points, and not solutions.

Definition 1 (Pareto dominance) *A point $y = (y_1, y_2, \ldots y_k)$ is said to Pareto dominate another point $z = (z_1, z_2, \ldots z_k)$ iff $\forall j \in 1..k, y_j \geq z_j \wedge \exists j \in 1..k, y_j > z_j$. This can be written as $y \succ z$.*

Definition 2 (Pareto optimal point) *A point $z \in Z$ is said to be Pareto optimal (in Z) iff $\nexists y \in Z, y \succ z$.*

Definition 3 (Pareto front) *The Pareto front, denoted Z^*, of a set Z is given by $\{z \in Z \mid \nexists y \in Z, y \succ z\}$.*

2.1 Input sequence and archive

We choose to describe the archiving process, and the properties of archives, in terms of an *input sequence* of points generated by some black-box process, following the approach used by Laumanns et al. [7].

Let $\langle f^{(1)}, f^{(2)}, f^{(3)}, \ldots, f^{(t)}, \ldots \rangle$ be a sequence of points where every point $f^{(t)} = (f_1^{(t)}, f_2^{(t)}, \ldots, f_k^{(t)}), \forall j \in 1..k, \forall t, f_j^{(t)} \in \Re_+$. As the input sequence is generally to be understood as being generated by some search algorithm that may revisit solutions and/or several solutions may map to the same objective function values, there is no requirement that the elements of this sequence are unique. Thus let $F^{(t)}$ denote the set of (unique) points in this sequence up to and including the tth point. Let $F^{*(t)}$ denote the Pareto optimal points from $F^{(t)}$. For convenience, we sometimes drop the t superscript and just refer to F, which is the set of points that appeared in a completed sequence.

Now define an archive $A^{(t)}$ to store some of the points in the input sequence up to and including time t: $A^{(t)} \subseteq F^{(t)}$. We also require that A is itself a nondominated set, i.e. that no element of A is dominated by another element of A. Using the notation of a Pareto front, we can then write that we require: $\forall t, A^{(t)} = A^{*(t)}$. We may also require a bound on the capacity of the archive,

i.e. that $\forall t, |A^{(t)}| \leq N$ for some constant N. As above, for convenience, we sometimes drop the t superscript and just refer to A. Then, for example, $|A| \leq N$ is shorthand for $\forall t, |A^{(t)}| \leq N$, and $A \subseteq F^*$ is shorthand for $\forall t, A^{(t)} \subseteq F^{*(t)}$. For convenience we also use the symbol A_N to stand for a general set of betweem 1 and N points: $A_N \in \{2^{\Re_+^k} \mid 1 \leq |A_N| \leq N\}$.

Now, we will need some mechanism for updating the archive, and this we call an *archiving algorithm*:

Definition 4 (Archiving algorithm) *Let a* deterministic *archiving algorithm be defined as a mapping:* $\mathcal{A}_{det} : \mathcal{A}_N \times \Re_+^k \mapsto \mathcal{A}_N$, *where* $\mathcal{A}_N$ *is the power set of* A_N. *Similarly, we define a* stochastic *archiving algorithm as a mapping :* $\mathcal{A}_{sto} : \mathcal{A}_N \times \mathcal{R}_N \times \Re_+^k \mapsto \mathcal{A}_N$, *where* $\mathcal{R}_N$ *is the power set of a set of up to* N *random variables.*

The definition above sets out an archiving algorithm as giving a definite output (an archive) for any well-defined input (an archive in addition to a new point, and perhaps some random variables). In addition, it is to be understood that an archiving algorithm as defined here is explicitly *not* allowed access to previous points from the input sequence, that are no longer in the archive itself. At time t, its update decisions must be based solely on the archive of the previous timestep and the tth point in the input sequence.

2.2 Approximation sets

In order for us to make some quantitative judgments about the quality of a final archive we will need some measure of its 'approximateness' to the Pareto front of a sequence. For this we can use the concepts of ϵ-dominance described in [7].

Definition 5 (ϵ-dominance [7]) *Let* $f, g \in \Re_+^k$. *Then* f *is said to* ϵ*-dominate* g *for some* $\epsilon > 0$, *denoted as* $f \succ_\epsilon g$, *iff for all* $i \in \{1, \ldots, k\}$

$$(1 + \epsilon).f_i \geq g_i.$$

Definition 6 (ϵ-approximate Pareto set [7]) *Let* $F \subseteq \Re_+^k$ *be a set of vectors and* $\epsilon > 0$. *Then a set* $F_\epsilon \subseteq F$ *is called an* ϵ*-approximate Pareto set if any vector* $g \in F$ *is* ϵ*-dominated by at least one vector* $f \in F_\epsilon$, *i.e.*

$$\forall g \in F : \exists f \in F_\epsilon \textit{such that } f \succ_\epsilon g.$$

The set of all ϵ*-approximate Pareto sets of* F *is denoted* $P_\epsilon(F)$.

The concept of an ϵ-approximate Pareto set formalizes what is meant by a well-distributed approximation of a Pareto front. Indeed, one can use the concept as a tool for measuring the quality of an approximation [12]: the minimum ϵ such that $A \in P_\epsilon(F)$, gives a measure of the quality of the approximation set stored, with lower values indicating a better approximation set.

Definition 7 (ϵ-Pareto set [7]) *Let* $F \subseteq \Re_+^k$ *be a set of vectors and* $\epsilon > 0$. *Then a set* $F_\epsilon^* \subseteq F$ *is called an* ϵ*-Pareto set if*

1. F_ϵ^* *is an* ϵ*-approximate Pareto set of* F, *i.e.* $F_\epsilon^* \in P_\epsilon(F)$, *and*
2. F_ϵ^* *contains Pareto optimal points of* F *only, i.e.* $F_\epsilon^* \subseteq F^*$.

The set of all ϵ*-Pareto sets of* F *is denoted* $P_\epsilon^*(F)$.

2.3 Normalization and translation

It is often desirable to normalize and translate the points in a sequence to keep them in some required range. A general system of normalization and translation is defined as follows.

For a maximization problem, a point f can be normalized and translated using:

$$\forall i \in 1..k,\ f_i^{norm} = c_i + \frac{(f_i - min_i)}{(max_i - min_i)} \tag{1}$$

where min_i and max_i denote the minimum and maximum values in the ith objective, respectively, and $c_i > 0$ is a constant used to translate the points away from the origin. For minimization problems the following equation can be used:

$$\forall i \in 1..k,\ f_i^{norm} = c_i + \frac{(max_i - f_i)}{(max_i - min_i)} \tag{2}$$

where f^{norm} is now to be maximized. For mixed minimization and maximization problems a combination of (1) and (2) can be used, so that all types of multiobjective optimization problem can be tackled within this framework. As we show later, it is desirable with the ϵ-base methods of Laumanns *et al.* to apply such normalization and translation to encourage an even distribution of points.

3 Theory: Desirable Behaviours and Limitations

Given the preliminaries of the last section, we are now in a position to consider the fundamental question of how an *ideal* archiving algorithm should behave. We choose to formulate this purely in terms of properties of the archive (though the computational complexity of the archiving algorithm may be a further important consideration).

We propose the following as a fairly complete 'wish list' of desirable properties of the archive produced by an archiving algorithm:

P1: $A = A^*$
P2: $|A| \leq N$
P3: $\exists t \forall u > 0, A^{(t+u)} = A^{(t)}$, i.e. the archive converges to a stable set in the limit of t
P4: $A \subseteq F^*$, i.e. the archive contains only Pareto optimal points from the sequence set, F
P5: all extremal Pareto optimal points from F are in A
P6: $|A| \simeq \min(N, |F^*|)$, i.e. that the archive is 'as close as possible' to N or the number of Pareto optimal points $|F^*|$
P7: for every point in F^* there is a point in A that is 'nearby'.

Let us review the list. Items **P1**, **P2**, **P4** and **P5** are straightforward and well-defined. Item **P3**, on the other hand, needs some further explanation. First, note that this property is different from the others in the list because it expresses a feature of the archive in the limit of t, and this further rests on the assumption

that F is finite (not unreasonable in any digital computer representation of a problem space). The other properties in the list only express some property of $A^{(t)}$, usually in terms of $F^{(t)}$. However, **P3** is really essential because it expresses the requirement that the archive should eventually converge; this is arguably desirable because otherwise stopping the generation process at the 'right' time may become critical! Notice that **P3** does not imply that after all points in the sequence have appeared *once* the archive should be converged, only that eventually it converges. This is not a trivial property: an archiving algorithm that always accepts a nondominated point – and removes a random point to make way for it – will not have this convergence property, for example. Although items **P1**–**P5** above are well-defined, clearly, **P6** and **P7** are not. The reason for leaving these last two in this 'loose' form is that we want to capture the ideas, respectively, of the final archive not becoming too small, and of being well-distributed, but without prejudicing how these properties *might* be formulated. As it is, the list covers, we think, everything that could conceivably be wanted of a size-limited archiving algorithm. Notice that the issue of the archive being well-distributed is covered by **P7**. We do not write that the points in the archive should be 'evenly spread' or some such because this is not, in general, true. Notice also that the list is not supposed to enumerate completely orthogonal properties. Some of the properties *are* coupled, but each is sufficiently independent to warrant its own place in the list.

We would now like to ask if any archiving algorithm exists that can guarantee to produce archives satisfying all of the properties. However, because of the loose definition of **P6** and **P7**, we are unable establish whether or not an *ideal* archiving algorithm is possible. So, to make progress, we need to sharpen up the definitions of these properties.

Let us start with **P6**. If we were to replace $\simeq$ by $=$, to tighten this desired property up then we would be confronted with:

Theorem 1. *No archiving algorithm, stochastic or deterministic, can maintain* $|A| = \min(N, |F^*|)$ *for general* N*, on general input sequences.*

(Informal) Proof of Theorem 1:
What we will show is that without access to the entire set F^*, it is impossible for an archiving algorithm to make decisions about what solutions to accept into, reject from, or maintain in, the archive A such that $|A|$ always equals either N or $|F^*|$; the latter when $|F^*|$ is less than N.

Consider F^*. What are the properties of its evolution over time? At any time step its cardinality can increase by one, stay the same, or decrease to a value between 1 and $|F^*| - 1$. So we have:

$$|F^{*(t+1)}| \in \{1, 2, ..., |F^{*(t)}| + 1\}. \tag{3}$$

Now consider a transition in F^* given by: $|F^{*(t)}| > N$ and $|F^{*(t+1)}| = M$, where $M \in 1..N-1$. At time t we must have, $A \neq F^*$ and $|A| = N$. And at time $t+1$ we need $|A| = |F^*(t+1)| = M < N$. Thus, exactly $N - M$ points must be ejected from A. But the number M is unknown to the archiving algorithm, in general, making it impossible to eject exactly

$N - M$ points. The value of M is unknown because M is a function of *all* the points in $F^{*(t)}$ (and of $f^{(t+1)}$, the new point), but at time t, $A \neq F^*$. Since the archiving algorithm as defined in Section 2.1 uses *only* A and the new point (plus some random variables in the case of a stochastic archiver) to determine A of the next time-step, and does *not* use any information about previous points, it cannot calculate M. Thus, no policy used by an archiving algorithm can follow these transitions, in general.

This is sufficient to prove the theorem.

□

We supplement the above proof with Figure 1. This shows the contents of A and of F^* at successive time-steps on a particular sequence of points. This example illustrates the fact that the archiving algorithm cannot predict the cardinality of F^* without direct knowledge of F^* itself, which it does not have.

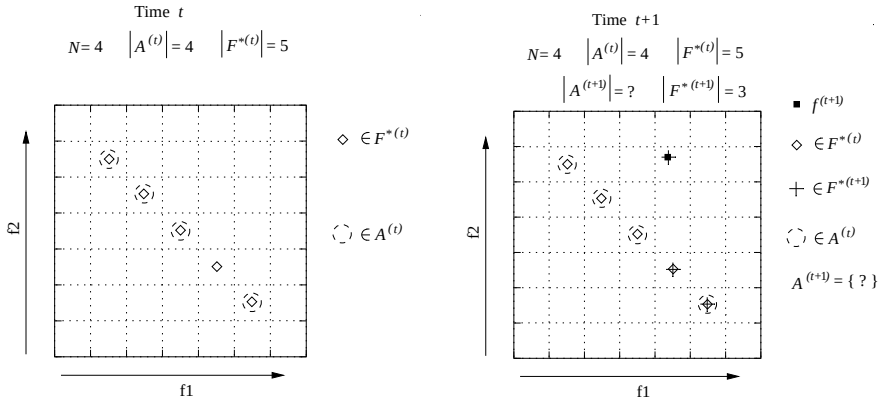


Fig. 1. The figure shows $|F^*|$ reducing from 5 to 3 in the transition from time t to time $t + 1$, due to the generation of $f^{(t+1)}$, which dominates three points in $F^{*(t)}$. However, the archiving algorithm cannot 'follow' this change in the cardinality of F^*, since $A^{(t)}$ only contains 4 points out of the 5 stored in $F^{*(t)}$. So, $|A^{(t+1)}|$ will not equal $|F^{*(t+1)}|$, *unless* the archiving algorithm 'guesses' correctly. Thus, *guaranteeing* $|A| = \min(N, |F^*|)$ is impossible

Thus we have shown that trying to keep the *minimum* size of the archive high is impossible in a strict sense. Notice that our proof does not rely on the archiving algorithm complying with *any* of the desirable properties **P1**–**P7**, except for **P2** impicitly. Thus it holds in a very general case: whenever the archiving algorithm cannot access *all* of the nondominated points from the sequence, i.e. the whole set F^*.

At present, it is not clear to us how to rigorously define a 'softer' requirement of keeping the archive's size 'as large as possible'. This does not mean that this

issue is not important. Clearly, if we want N solutions and obtain only $1 \ll N$ when there were $|F^*| \geq N$ available then we may feel (justifiably) aggrieved. However, for now, we must be content to know that no algorithm can *guarantee* keeping $\min(N, |F^*|)$ points. As we shall see in the next sections, the archiving algorithms proposed in [7] suffer from the problem of archiving far too few points in certain circumstances. What we have highlighted here is that *no* algorithm can escape from this problem entirely.

Moving now to **P7** in our 'wish list', we can make this property more explicit by using the notion of an ϵ-approximate set. We can say that ideally A should be an ϵ-approximate set of size up to N that minimizes ϵ:

$$A^{ideal}_{\epsilon\text{-approx}} = \text{argmin}_{A \in P_\epsilon}\{\epsilon \in \Re^k_+ \mid |A| \leq N\}$$

This seems reasonable but unfortunately we have:

Theorem 2. *No archiving algorithm, stochastic or deterministic, can maintain an 'ideal' archive $A^{ideal}_{\epsilon\text{-}approx}$, for general input sequences.*

Proof of Theorem 2:
For this proof it is sufficient to show that there exist two sets $F^{(t)}$ and $F^{(t+1)}$ with $F^{(t+1)} = F^{(t)} \cup \{f^{(t+1)}\}$ such that $\exists f \in F, f \neq f^{(t+1)}, f \in A^{ideal}_{\epsilon\text{-approx}}(F^{(t+1)})$ and $f \notin A^{ideal}_{\epsilon\text{-approx}}(F^{(t)})$. In words: a point in the sequence up to the point just before $f^{(t+1)}$ is generated, is needed in the ideal archive of $F^{(t+1)}$, but it does not appear in the ideal archive of $F^{(t)}$. This proves the theorem since, in this case, either the first or the second ideal archive *cannot* be produced by an archiving algorithm. (An archiving algorithm cannot make the transition between the two sets).

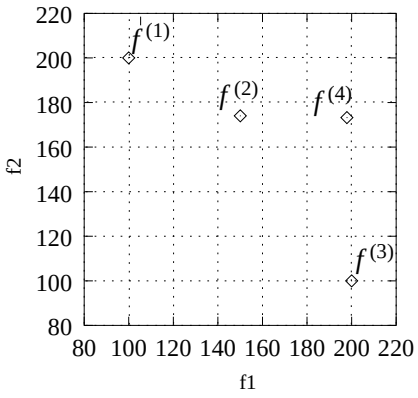
We now give an example to demonstrate this case. In Figure 2 (left) a sequence of four points is plotted in objective space. The points are $f^{(1)} = (100, 200), f^{(2)} = (150, 175), f^{(3)} = (200, 100)$ and $f^{(4)} = (199, 174)$. We now consider the situation where $N = 2$ and we wish to calculate the ideal archive at times $t = 3$ and $t = 4$. In the table (right), we calculate, for each possible pair of points in the archive, the minimal ϵ value that ensures all points in the sequence are ϵ-dominated by the two archived points, i.e.

$$\max_{\mathbf{y} \in Z} \min_{\mathbf{z} \in A} \max_{i \in 1..k} \frac{y_i}{z_i}.$$

The result shows that at time $t = 3$ the ideal archive is $\{f^{(2)}, f^{(3)}\}$ (with a corresponding ideal ϵ value of 0.14), and at time $t = 4$ the ideal archive is $\{f^{(1)}, f^{(4)}\}$ (with a corresponding ideal ϵ value of 0.0057), thus completing the proof.

□

The above shows that in general it is not possible to obtain the ideal minimum ϵ value. Moreover, we can make this result even stronger (i.e. more limiting) if we assume no knowledge of the extent of the Pareto front of the generated sequence of points, as follows.



$A^{(t=3)}$	lowest ϵ value
$\{f^{(1)}, f^{(2)}\}$	0.33
$\{f^{(1)}, f^{(3)}\}$	0.50
$\{f^{(2)}, f^{(3)}\}$	**0.14**

$A^{(t=4)}$	lowest ϵ value
$\{f^{(1)}, f^{(2)}\}$	0.33
$\{f^{(1)}, f^{(3)}\}$	0.99
$\{f^{(1)}, f^{(4)}\}$	**0.0057**
$\{f^{(2)}, f^{(3)}\}$	0.33
$\{f^{(2)}, f^{(4)}\}$	0.14
$\{f^{(3)}, f^{(4)}\}$	0.15

Fig. 2. A sequence of four points plotted in objective space and the corresponding minimum values of ϵ for archiving just two of the points such that all the points in the sequence up to time t are ϵ-dominated

Theorem 3. *No archiving algorithm, stochastic or deterministic, can maintain an ϵ-approximate set of cardinality $\leq N$ with an ϵ that is less than a constant κ of the ideal ϵ value, for general input sequences,* without additional knowledge of the extent of the PF of the input sequence.

In the following proof of Theorem 3 we construct a sequence of points in such a way that the Pareto front of the points in the sequence gets smaller (periodically) over time, without bound. Thus, any static value of ϵ, used internally by an archiving algorithm to certify that its archive is an ϵ-approximate set, will eventually be more than κ times larger than the ideal value of ϵ.

(Informal) Proof of Theorem 3:
We can easily construct an unbounded number of 'local' Pareto fronts of decreasing size in a box of side 1. For example, let the following sequence of points be generated: (1,0), (0,1), (0.2313,0.7687), ... and so on, until there are say, 1000 *random* points on the line going from (0,1) to (1,0). Now let the sequence continue with (0.6103, 0.8897),... and 1000 more random 'front-two' points, where each vector has components that sum to 1.5. The 'front-three' points can be the same with the sum of the components being 1.75. And so on ....

Notice that any line L in a box of side 1, composed of points (x, y) with $x + y = a, a \geq 1$ will have length $l = (2 - a).\sqrt{2}$, so as the value of a follows the sequence 1.0,1.5,1.75,1.875, ..., so the length l of the Pareto front tends to zero.

Now, assume that an ideal archiving algorithm exists with the required property. Then if we stop it at any time during this unbounded sequence it must return an ϵ-approximate set of cardinality less than or equal to N with ϵ less than κ times the ideal value.

To guarantee doing this the algorithm will need to keep its own internal value of ϵ, call it ϵ_{arc}, by which it can certify that the current archive is an ϵ-approximation of the sequence of points so far encountered. It needs ϵ_{arc} so that when a point (nondominated in the current archive) is rejected it can check that there is a point in the archive which ϵ-dominates it.

But now we can see that this internal ϵ_{arc} value must reduce over time as each new front in our sequence (defined above) is generated, otherwise, at some point, the value of ϵ_{arc} will become greater than κ times the ideal ϵ value. But if ϵ_{arc} is reduced, the algorithm must be able to guarantee that no point previously excluded using a prior, larger ϵ_{arc} value, should be in the archive, given the reduced ϵ_{arc} value—a contradiction since these points have been excluded and no information about them remains.

□

The above proof uses a sequence construction that is not likely to arise in practice, and indeed, it is of course limited by the accuracy with which numbers can be represented in a digital computer. Nonetheless, it illustrates a point: if we use an algorithm that guarantees an ϵ-approximation but with an initially *undefined* ϵ value (a value which is returned only at the end by the archiving algorithm), then difficulties may arise since there is no limit to how far from ideal this final ϵ value will be. Thus, even on more 'reasonable' optimization functions, the archive obtained may be arbitrarily too small, and be a very long way away from the true Pareto front (see next section).

On the other hand, it would seem that in practice if we use an algorithm that does not make the guarantee of ϵ-approximation, then we may be able to obtain a good approximation with close to the desired number of solutions, albeit without any formal guarantee of approximateness.

Notice that in the above we have excluded the possibility that we know in advance the extent of the PF of the sequence of generated points. If we do know these then we can calculate ϵ beforehand, to bound the archive size as required (see section 4.2, equations 4 and 5), and then simply discard any point that is not within the ranges of the final Pareto front. However, this approach will not *maintain* an ϵ-approximate set of the input sequence, but only obtain it after all points have been considered. For example, if the optimization algorithm is stopped early, the archiver may have stored *no* points at all.

This section has demonstrated the existence of some theoretical limitations on algorithms for maintaining bounded archives. In the next section, we describe several practical archiving algorithms, focusing on two in particular, and relate them back to our 'wish list' of desirable properties, and to the theoretical limitations we have described.

4 Practical Archiving Algorithms

In this section we will first briefly summarise the convergence properties of two practical archiving algorithms, previously analysed in [6] and [5]. Both of these

algorithms maintain bounded, nondominated archives but have some clear weaknesses, allowing us to neglect them from further empirical analysis. Following this summary, we present, in Sections 4.1–4.3, the adaptive grid archiving algorithm (AGA) and two ϵ-based archivers, all of which we go on to test empirically in Section 5.

In [6] and [5], we have analyzed the convergence properties of three algorithms for maintaining bounded archives. This analysis makes use of an assumption that a positive-definite generating function is used to generate points. This means that for all time-steps all points in the finite objective space have a non-zero probability of being generated. Using this assumption it is easy to show that some simple algorithms converge in the sense of item **P3** in the 'wish list' of the last section. For example, an algorithm we call here RA1 (because it is like one considered in Rudolph and Agapie [10]), which uses the following two rules, converges in this sense. Rule 1: When the archive is below its capacity all nondominated points are accepted, and any dominated points in the archive are removed. Rule 2: When the archive is at its capacity only dominating points are accepted, and any dominated points are removed. As far as our 'wish list' is concerned, this algorithm satisfies items **P1**, **P2**, **P3**, and **P4** but it does not attempt to address items **P5**, and **P7** at all. In other words, it does not have any mechanism *at all* for accepting a new, nondominated (but not dominating) point over one already in its archive (which it could discard) in order to obtain a better net approximation, in the sense of accepting points that would increase the objective space extent of the archive or improve its distribution. This algorithm will usually obtain a 'close to full' archive, i.e. item **P6** is partially satisfied.

A second algorithm analyzed in [6], $\mathcal{S}$ metric archiving, uses a similar strategy to that above, but it has an additional rule that, when the archive is full, it *will* accept a new nondominated point (and discard a current archive member) if this would lead to a net increase in the $\mathcal{S}$ metric value of the archive. The $\mathcal{S}$ metric [11] measures the quality of a nondominated set by measuring the hypervolume of the region dominated by the set. So, $\mathcal{S}$ metric archiving satisfies items **P1**, **P2** and **P3**, but it sacrifices **P4**. On the other hand, it attempts to address items **P5** and **P7**: if a new point is either extremal or very far away from any point in the archive, it will tend to be accepted by the algorithm, and a point which is 'not so important' will be discarded. As in RA1, item **P6** is also respected. In many ways, then, this archiving algorithm may be practically ideal but until recently its computational overhead was thought to be prohibitively high because of the computational cost of calculating the $\mathcal{S}$ metric. However, recent developments by Fleischer [2], in which a polynomial time algorithm is given for this calculation, suggest it may now be worth investigating further. However, since this development is very new we do not consider this archiving method further in this article.

4.1 Adaptive grid archiving, AGA

A stochastic archiving algorithm, described and analyzed in [5, 6], and one of the two we focus on in our experiments below, is adaptive grid archiving (AGA). AGA works on similar lines to the two above but is much more efficient than $\mathcal{S}$

metric archiving, while retaining much of the latter's ability to obtain a 'good' approximation.

As with the previous algorithms, when the archive is not full AGA accepts all nondominated points, while dominated members of the archive are discarded. When the archive *is* full a couple of computationally efficient stochastic rules are used to see if a new nondominated point should be accepted and a current member discarded. These rules are based on how crowded together are the different points in the archive. Basically, if a point is in a crowded region, estimated using a grid in the objective space (which adaptively changes in position and size to cover the archived points), then it is liable to be removed to make way for a new, less-crowded point. However, to ensure the archive maintains the PF extremes, points that are uniquely extremal on any objective are also protected from removal. A precise description of the entire algorithm can be found in [5], where it is also shown that the archive 'settles down' (in a clearly defined way) for some types of input sequence. However, a general convergence proof is not possible. In summary, we can say that this algorithm satisfies **P1** and **P2**, and does well on **P5**, **P6** and **P7**, while sacrificing on the strict convergence property **P3**. Furthermore, it can easily reject/discard Pareto optimal points and replace these with non-Pareto optimal ones, so it does not satisfy **P4**.

Despite the weaknesses, we believe this algorithm performs well in practice in our 'blind, one-shot' scenarios compared to other methods. In particular, we suggest that it generally performs better in these circumstances than those algorithms (described next) that have a convergence proof and guarantee an ϵ-approximation set.

4.2 Adaptive ϵ-approximate algorithm, LTDZ1

Laumanns *et al.* [7] proposed two deterministic archiving algorithms, one for obtaining a guaranteed ϵ-approximate Pareto set, the other for obtaining a guaranteed ϵ-Pareto set. In the basic versions, ϵ is a parameter set by the user prior to optimization. The algorithms then guarantee to archive a bounded number of points, at this given approximation level. But this bound is a function of the (possibly unknown) objective space ranges. Thus, in general, with ϵ set prior to optimization these algorithms will not keep $|A| < N$.

If a hard capacity bound *must* be satisfied, as we assume here, then one of two alternative strategies can be used. With prior knowledge of the ranges of values of the Pareto front in objective space, the algorithm is changed so that all points that lie outside the ranges of the PF are not archived, and a static ϵ value is calculated *a priori* using:

$$1 + \epsilon = R^{\frac{1}{N^{1/(k-1)}}} \tag{4}$$

where R is given by:

$$R = \max_{i \in 1..k} \left\{ \delta f_i \mid \delta f_i = \left(\frac{max_i}{min_i} \right)^{sign(max_i - min_i)} \right\} \tag{5}$$

where max_i and min_i are respectively the maximum and minimum values of the nondominated points from the sequence set F. (An alternative, approximate way to derive ϵ is given in [7]. We find Equation 3 to be more correct and effective in practice.)

The result will be an ϵ-approximate Pareto set of size less than N. However, as Equation 3 is maximally 'conservative' then the resulting set may have far fewer than the desired number of points. So, when using these algorithms, even *with* prior knowledge of objective ranges then items **P6** and also **P7** may not be well-satisfied. On the other hand, items **P1**, **P2**, **P3** and **P4** are respected. Item **P5** is not directly addressed by these algorithms.

In a 'blind, one-shot' scenario, a different strategy must be employed to keep the archive bounded by N. Laumanns *et al.* [7] proposed an adaptive version of each of their algorithms, where the algorithm's internal ϵ parameter, used for making archiving decisions, is adapted over time with respect to the range of values currently in the archive, by again making use of Equation 6. The description in [7] of this adaptive method is rather sketchy for the ϵ-Pareto set archiving algorithm and some important details are not given. Fortunately, for the adaptive ϵ-approximate Pareto set archiving algorithm, sufficient details *are* given to implement it, and since both of the adaptive algorithms work in essentially the same way we may infer from any problems we find with the one, that similar problems will arise in the other.

Algorithm 1 (shown overleaf), denoted LTDZ1 here, defines this adaptive algorithm, which guarantees to archive an ϵ-approximate Pareto set of any sequence of points, F. The algorithm works by calling a simple function for updating the archive set A with each successive vector f from the sequence. Lines 14–20 of Algorithm 1 comprise this update function. The vector f is accepted if it dominates any vector in A or if it is not ϵ-dominated by any vector in A, otherwise it is rejected. In case it is accepted, all points in A that are dominated by f are removed, to maintain A as a nondominated set. Note that the function works only if all points in the sequence are elements of $\Re_+^k$. To ensure this condition, normalization and translation of points can be applied, e.g. using Equations 1 and 2. In this case all comparisons of the vectors would use the normalized and translated points (not shown in Algorithm 1).

If max and min[1] are not known prior to optimization, then they must be updated during search. Lines 1 to 13 of Algorithm 1 describe the extra steps needed in the adaptive version of the algorithm. First, the new point f is checked against the current extrema (max and min vectors). The extrema are updated if f is beyond any of them. Using Equation 4 and the new extrema, ϵ is updated. If ϵ increases as a result of the update then all of the points in the archive, $A^{(t-1)}$, must be 'filtered'. This means they are considered one by one, oldest first. If a younger point is ever ϵ-dominated by an older one then it is removed from A. Initial extrema and an initial ϵ are calculated using the first two nondominated points in the sequence.

[1] These are sometimes known as the *ideal* and the *nadir* points. They are not, in general, points themselves in the sequence set F, but are the points defining the upper and lower boundaries, respectively, of the Pareto front

Algorithm 1 LTDZ1: Update function of the adaptive-ranges ϵ-approximate archiving algorithm

Input: $A^{(t)}, f, max, min, \epsilon, t$
$t \leftarrow t + 1$
$flag \leftarrow 0$
for $i \leftarrow 1$ **to** k **do**
  if $(f_i > max_i) \vee (f_i < min_i)$ **then**
    update_ranges(max_i, min_i, f_i)
    $flag \leftarrow 1$
  end if
end for
if $flag = 1$ **then**
  update_epsilon(f, max, min) `// see Equations 4, 5`
  filter$(A^{(t-1)}, \epsilon)$
end if
if $\exists f' \in A^{(t-1)}$ such that $f' \succ_\epsilon f$ **then**
  $A^{(t)} \leftarrow A^{(t-1)}$
else
  $D \leftarrow \{f' \in A^{(t-1)} \mid f \succ f'\}$
  $A^{(t)} \leftarrow A^{(t-1)} \cup \{f\} \setminus D$
end if
Output: $A^{(t)}, max, min, \epsilon, t$

4.3 Non-adaptive ϵ-Pareto archiving, LTDZ2

As we mentioned in the last section, if prior knowledge of Pareto front ranges are known in advance, it is possible, in both of the algorithms proposed and described in [7], to set ϵ to a constant value (e.g. using Equation (3)) and reject all points that lie outside of the known Pareto front region. We would like to test this strategy, together with the two adaptive methods, AGA and LTDZ1, if only to give some upper bound on how well the other two *could* perform. For this we opt to use the 'better' of the two algorithms from [7]—that which guarantees an ϵ-Pareto set. For convenience, we call this approach LTDZ2 here.

5 Case Studies

In this section we test empirically the archiving performance of AGA and of the ϵ-based archivers, described above. For the testing, we use several types of sequence that have been selected purposely for the difficulty they may cause to one or more of the archivers. For each sequence type we consider, we always generate *one* sequence first, (independently of any archiving method) and then use this same sequence as the input to each archiver. The sequences were generated sometimes just using a function, and other times actually running a search algorithm on a problem to be optimized. For the ϵ-based algorithms, LTDZ1 and LTDZ2, we need only make one archiving run on any given sequence, since both algorithms are deterministic. For AGA we need to make several runs because AGA contains

a stochastic procedure—when a point is discarded from the archive it is selected *at random* from a region that is one of the set of 'most-crowded' regions.

To present the results, we show plots of the archives, compared to the true Pareto front of the sequence (though not in the one 3-objective problem we tackle). For AGA we just select one run (the first) to show in these plots. In addition to these visual results, we use two performance metrics to provide quantitative results. The first metric is:

$$Absolute\_mean\_utility(A, U) = \frac{1}{|U|} \sum_{u \in U} u^*(A) \tag{6}$$

where u is a utility function, U is some set of utility functions, and $u^*(A) = \max_{f \in A}\{u(f)\}$. The second metric, defined in [3] compares two algorithms as follows:

$$R2(A, B, U) = \frac{1}{|U|} \sum_{u \in U} u^*(A) - u^*(B). \tag{7}$$

For both metrics we use the Tchebycheff utility function:

$$u(f) = \max_{i \in 1..k} \{\lambda_i.(max_i - f_i)/(max_i - min_i)\} \tag{8}$$

where $\sum_{i \in 1..k} \lambda_i = 1, \forall i \in 1..k, \lambda_i \geq 0$, and max and min are the ideal and nadir points, respectively, of the sequence. The set U is constructed by using 500 evenly distributed λ weight vectors.

To deal with the separate runs of AGA, we report on the mean values for AGA, giving in addition the standard deviation of the absolute mean utility.

A note on normalization and translation for LTDZ1

The distribution of points archived by the LTDZ1 algorithm is dependent on the normalization and translation of the points in the objective space. Equation 3 shows that the value of ϵ is dependent on R, the largest ratio of the maximum to the minimum objective value in any objective. Thus, the normalization and translation of points may affect ϵ and thus the distribution of archived points. If we normalize and translate points so that they lie in the semi-open interval (0,1] then this may lead to a non-uniform sampling of the Pareto front in some situations. To illustrate this effect we generated a sequence of 1000 points (f_1, f_2) with $f_2 = 1 - \sqrt{f_1}$, by starting at $f_1 = 0.001$ and incrementing it by 0.001 a further 999 times. We then applied LTDZ1 to this sequence with $N = 100$. The result is shown in Figure 3 (left). Only 14 points are actually archived and they are non-uniformly distributed despite the uniform distribution of the underlying sequence. The strong non-uniformity is caused by the fact that ϵ begins very small (because the ratio of max and min values is initially small), and grows ever larger. We can improve the behaviour of LTDZ1 greatly by just normalizing and translating all points so that they lie in the range (99,100]. Figure 3 (right) shows the archive of LTDZ1 in this case, on the same sequence. 69 points have been archived and they are more evenly spread. In all our following experiments, then, we translate all objective values to lie in (99,100], using (2) to avoid this effect occurring with LTDZ1.

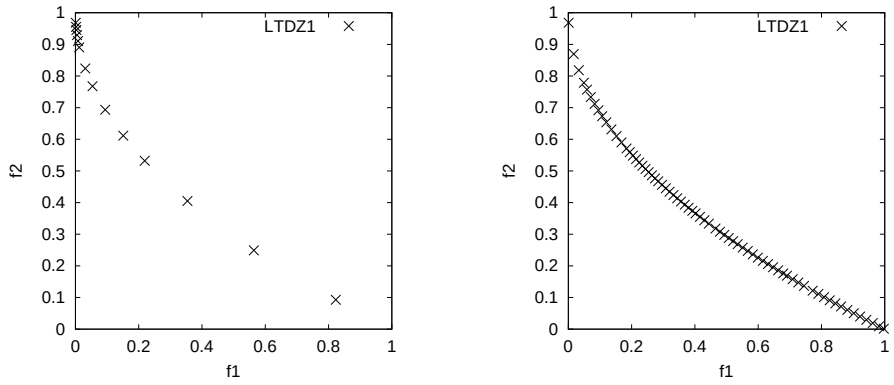


Fig. 3. Left: A non-uniform distribution of points generated by the LTDZ1 algorithm, caused by ϵ starting small and growing over time, as points were being generated from left to right. Right: A more uniform distribution of points, on the same sequence, obtained by translating all points far away from the origin so that ϵ is not so variable

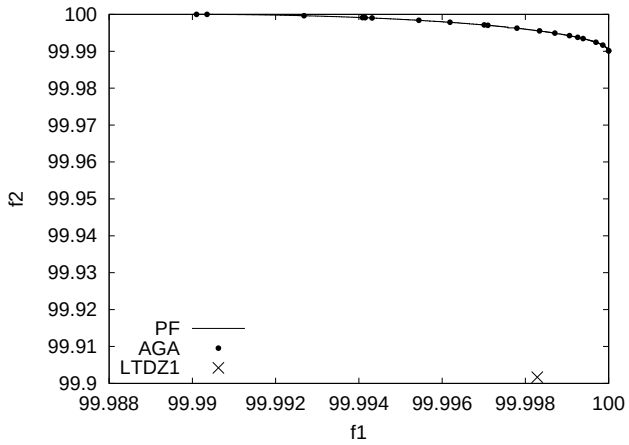


Fig. 4. The archived points when AGA and LTDZ1 are used on the same sequence of points generated by PESA-II on a problem with a small PF. LTDZ1 archives only one point because ϵ grows very large initially, and cannot be reduced subsequently

Sequence I: a small Pareto front

The LTDZ1 algorithm has the weakness that ϵ cannot be decreased over time. This means that if ϵ becomes large with respect to the size of the Pareto front then a very poor approximation results. Consider the two-objective maximiza-

tion problem described by:

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

and where u and v are two decision variables in the interval [0,1). Here, the radius, r, or the Pareto front 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. Figure 4 shows the set of points archived by LTDZ1, and by the first run of AGA when $N = 20$ is used. (The true Pareto front of the sequence is made up of 1824 points so it appears smooth in the plot). Clearly, the archive of AGA is significantly better. In total, 10 independent runs of AGA on this same sequence of points were collected. For these, quantitative results are given in Table 1.

On this problem, as on those that follow, we also ran the non-adaptive LTDZ2 algorithm. To do this, we found a fixed value of ϵ empirically by running the algorithm on the sequence several (here, five) times until we had found a value giving us as close as possible $|A| = N$. The results for this final run of LTDZ2 are also given in Table 1 for comparison.

Sequence II: a three-objective problem with small PF

The AGA algorithm can suffer from cyclic behaviour on problems having more than 2 objectives, and where there exist points that are extremal but not Pareto optimal [6]. For example, imagine a three-objective case where (assuming we wish to maximize on each objective), the Pareto front contains the points $\mathbf{x} = (4, 9, 7)$ and $\mathbf{y} = (4, 7, 9)$. Further, the point $\mathbf{z} = (3, 7, 8)$ is clearly not on the Pareto front. Now, assume that we have a full archive, and, of these three, only $\mathbf{x}$ is currently in the archive, and $\mathbf{x}$ is an extremal point with respect to the first objective. That is, no other point currently in the archive has a value of 4 or worse on this objective. Now, suppose point $\mathbf{z}$ is generated by the optimization process and therefore (since it is not dominated by anything in the archive) becomes a candidate for placing into the archive. As a consequence of the way AGA works, $\mathbf{z}$ will certainly enter the archive (if the archive is full, $\mathbf{z}$ will enter and some non-extremal point will be removed; since $\mathbf{z}$ extends the archive's range in at least one objective, AGA naturally requires that such a point enters the archive, in order to promote the idea of a good representation of the entire Pareto front. However, suppose that later on the point $\mathbf{y}$ is generated; it will certainly enter the archive because it dominates $\mathbf{z}$, and $\mathbf{z}$ will therefore be removed. This will then cause AGA to update the grid parameters in such a way as to reduce the represented range of objective 1, since the point in the archive worst on that objective is now $\mathbf{y}$. The crucial point is that it is possible for $\mathbf{y}$ to become lost from the archive at some future iteration (since AGA can lose Pareto optimal points). This would then allow $\mathbf{z}$ to enter again, creating a cycle. This cycle of expanding and shrinking grid boundaries may repeat. This fact undermines

Table 1. Results comparing the archives with the true Pareto front (PF) for the various sequences in the case study. The 'absolute' value is calculated using Equation 5. The comparative values are the R2 metric values (Equation 6). For AGA values are the mean of 10 runs. The standard deviation of the absolute value is also given, in brackets

algorithm	$\|A\|$	N	absolute	vs LTDZ1	vs LTDZ2	vs PF
	Sequence I: small PF					
AGA	20	20	0.7505 ($< 10^{-5}$)	0.3371	0.0001	0.0000
LTDZ1	1	20	0.4134	–	0.-0.3370	-0.3371
LTDZ2	20	20	0.7504	–	–	-0.0001
PF	1824	–	0.7505	–	–	–
	Sequence II: 3-objective small PF					
AGA	20	20	0.6246 ($< 10^{-5}$)	0.0369	0.0000	-0.0001 ( )
LTDZ1	5	20	0.5877	–	-0.0368	-0.0369
LTDZ2	19	20	0.6246	–	–	-0.0001
PF	78	–	0.6247	–	–	–
	Sequence III: highly non-uniform PF					
AGA	20	20	0.7505($< 10^{-5}$)	0.0184	0.0006	0.0000
LTDZ1	4	20	0.7321	–	-0.0178	-0.0184
LTDZ2	18	20	0.7499	–	–	-0.0006
PF	50	–	0.7505	–	–	–
	Sequence IV: discontinuous PF					
AGA	20	20	0.7503 ($< 10^{-5}$)	0.1055	0.0063	-0.0002
LTDZ1	5	20	0.6448	–	-0.0992	-0.1057
LTDZ2	19	20	0.7440	–	–	-0.0065
PF	78	–	0.7505	–	–	–

the convergence of AGA in the general (more than 2 objectives) case, since convergence of the archive rests on convergence of the grid boundaries (see [6] and [5] for more detailed analysis).

To see how much of a problem this behaviour is in practice, we used the following 3-objective optimization problem:

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

and where u, v and w are decision variables in the interval [0,1). Like, that of the previous sequence, the Pareto front is small compared to the objective space, so cyclic behaviour of AGA can certainly occur. We used 30 bits to encode each variable (u, v, and w) and generated a sequence by running PESA-II on the problem for 1500 evaluations. This gave a sequence with 78 Pareto optima. It is difficult to display the visual results on this problem so we just give the quantitative results (Table 1). Clearly AGA has little difficulty in approximating this set very accurately, even though non-Pareto optimal points *are* generated outside the ranges of the PF during archiving of the sequence. To show that this

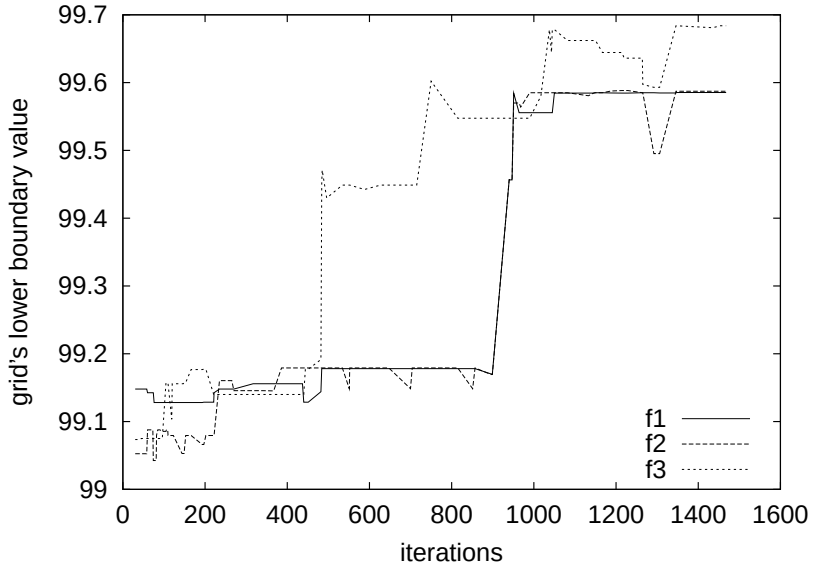


Fig. 5. The lower grid boundaries of AGA's grid in objective space, fluctuating over time on a 3-objective problem sequence

is happening we have plotted the lower boundary values of the adaptive grid against evaluations in Figure 5. As expected, there are fluctuations, although the values are tending towards the correct value of 99.99.

As in the previous sequence, LTDZ1 performs very poorly by comparison with AGA, partly because its ϵ value becomes too high early on and cannot be reduced later. As for LTDZ2, even with several runs to tune the fixed ϵ value, its performance is not perceivably better than that of AGA (sse Table 1).

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. Adaptive ϵ-dominance based approaches will update the value of R (Equation 3) based on the extremal values of the observed nondominated points, and by this ϵ will also be set. Using this method, all of the closely spaced points will be filtered out by the (relatively) large ϵ value. In contrast, adaptive grid methods do not take such drastic action to avoid over-filling the archive. Only one point is ever removed using the adaptive grid methods, except when a point f dominates more than one member of A. Thus, adaptive grid methods are better suited to these non-uniformly spaced Pareto fronts.

To illustrate this phenomenon, we randomly generated a sequence of 500 points using the functions:

$f_1^{(t)} = 2^{2^{3 \cdot x}} + \mathcal{U}^{(t)}(10)$, and
$f_2^{(t)} = 257 - 2^{2^{3 \cdot x}} + \mathcal{U}^{(t)}(10)$, where
$x^{(t)}$ is uniformly randomly chosen from $\{0.0, 0.1, 0.2, \ldots 1.0\}$,

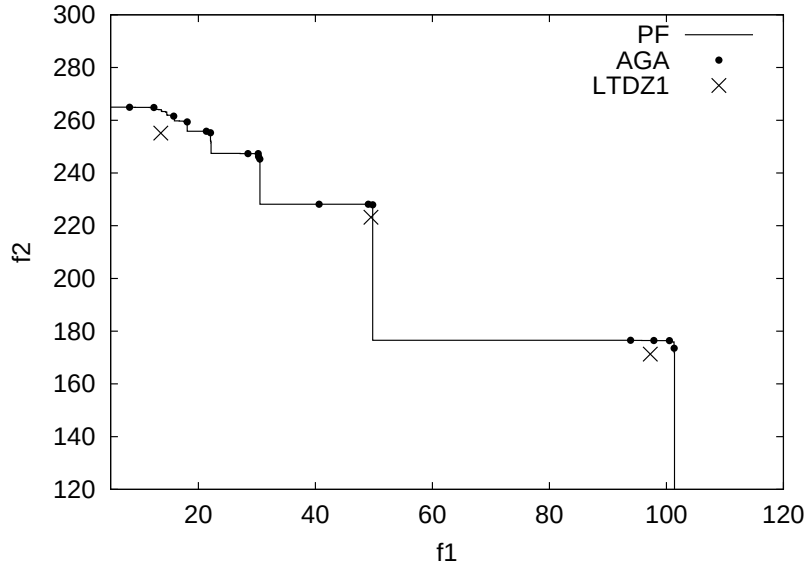


Fig. 6. A sequence of points forming an exponentially spaced PF. The archives achieved by AGA and LTDZ1 when $N = 20$ is used, compared to the true Pareto front (PF) which has 50 points. Notice that AGA keeps many more points at the top left of the PF so that it more accurately approximates the true PF. LTDZ1 is not able to do this

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

We then applied the archiving algorithms AGA and LTDZ1 to this sequence, and compared it to the Pareto front. The results are shown in Figure 6. As before, quantitative results are given for this sequence in Table 1. They indicate that AGA performs significantly better than both the ϵ-based archivers.

Sequence IV: a discontinuous Pareto front

If the Pareto front is discontinuous this may cause problems to either AGA or LTDZ1. If the nondominated points are concentrated only in small, isolated regions then AGA's grid may become ineffective at controlling the distribution of points. This is because *within* a grid region AGA has no control over distribution—so if all points lie in just, say, two or three isolated grid regions then we would expect the points to show a fairly random distribution at the fine level, rather than an even one. For LTDZ1, the problems are even more serious, however. As we have already seen, the setting of ϵ is conservative: it assumes that points may be distributed across the whole range of the objective space, with no gaps. Thus, when there are gaps in the PF, LTDZ1 may obtain much fewer than the desired number of points. This may severely affect the quality of the approximation set achieved. Figure 7 presents results on a sequence of points, forming a discontinuous Pareto front with three isolated regions. As predicted, AGA

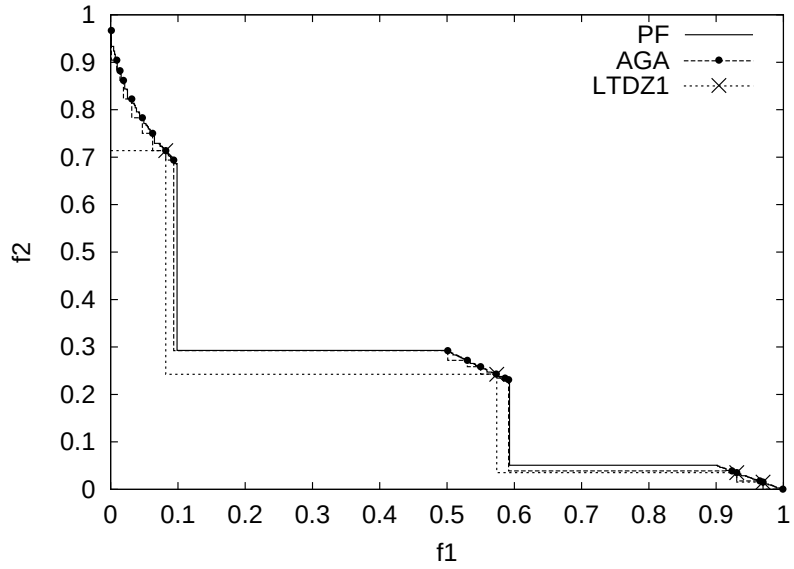


Fig. 7. A sequence of points forming a discontinuous PF made up 200 points. The archives achieved by AGA and LTDZ1 when $N = 20$ is used, compared to the true Pareto front (PF). Because LTDZ1 estimates ϵ based on the front being continuous, it over-estimates how big it should be, and consequently only 4 points are found. AGA correctly finds 20 points and distributes them relatively evenly between the isolated parts of the PF

obtains the maximum number of points allowed and samples all three isolated regions, but at the fine level its distribution of these points is somewhat uneven. Nonetheless, its performance is far better than LTDZ1's which only manages to find four points, and shows a much poorer approximation of the sequence. AGA also performs better than LTDZ2. Quantitative results are given in Table 1.

6 Concluding Discussion

Archiving algorithms are important components of search algorithms for Pareto optimization for two main reasons. First, they store the points found so that they can be presented at the end of a search run. Second, they can be used as an on-line memory, used to help generate new points. Although, ideally we would like to keep all Pareto optimal solutions encountered during a search, in practice setting a bound on the archive's capacity may be necessary or desirable. Both memory and computational overhead become important issues when the archive's capacity grows without bound. The computational overhead is a particularly important constraint when points in the archive are to be used as an on-line memory.

So, given that limiting the archive's size is a reasonable demand, what desirable properties of an archive *can* be achieved? We have shown that certain desirable properties are theoretically impossible in any archiving algorithm whatever: the number of points in the archive cannot, in general, be the minimum of N, the capacity bound, and $|F^*|$ the number of Pareto optimal solutions encountered; and it is not possible to guarantee an ϵ-approximate set with ϵ less than κ times the ideal minimum value. These results raise the question as to whether, in practice, it is better to use an archiving algorithm with 'guaranteed convergence and diversity' such as those proposed in [7], or algorithms which do not offer these guarantees but employ mechanisms that merely 'encourage' diversity and convergence. We have focused particularly on the case where we are not privileged with *a priori* knowledge of objective space or Pareto front ranges. Then, on several sequences possessing particular features, we compared the performance of LTDZ1 and AGA. Our findings suggest that, at least in these scenarios, AGA archives a significantly better approximation to a sequence than LTDZ1. The usual reason for this is that LTDZ1 is necessarily conservative in its adaptation of ϵ and so often ends up with far fewer points than desired, giving a poor approximation set.

We also compared the performance of AGA to the non-adaptive ϵ-Pareto set approach, called LTDZ2 here, when the latter was run multiple times to find an empirical setting of ϵ. Even against this approach, AGA performs remarkably well, with no statistical significance in its performance difference. If we had just used Equation 3 to set the ϵ value of LTDZ2, knowing the Pareto front ranges, the performance of AGA would have been significantly better in a comparison. Overall, this shows that *even with* prior knowledge of ranges, or even prior runs, it is difficult to set ϵ to give performance comparable to a single run of AGA.

Unfortunately, AGA is still not practically ideal. The fact that convergence is not generally guaranteed is a weakness. A potentially better approach, being considered now (see [4]), is the $\mathcal{S}$ metric archiving. This method has guaranteed convergence in the sense that cyclic behaviour is not possible: so eventually the archive stops changing given reasonable assumptions about the search space and generating function. Furthermore, it pursues an archive which maximizes the hypervolume of the dominated region—a measure which is one of the best unary measures of approximation set quality. Future work will investigate further the theoretical and practical performance of this promising method.

Acknowledgments

Joshua Knowles gratefully acknowledges the support of a European Commission 'Marie Curie' research fellowship, contract number HPMF-CT-2000-00992.

References

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, San Francisco, California, 2001. Morgan Kaufmann Publishers.

2. 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*, number 2632 in LNCS, pages 519–533. Springer, 2003.
3. Michael Pilegaard Hansen and Andrzej Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Technical University of Denmark, March 1998.
4. Joshua Knowles. Pareto archiving using the Lebesgue measure: Empirical observations. Technical report, IRIDIA, Université Libre de Bruxelles, Belgium, May 2003.
5. 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.
6. Joshua 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.
7. Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. On the Convergence and Diversity-Preservation Properties of Multi-Objective Evolutionary Algorithms. Technical Report 108, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.
8. Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282, Fall 2002.
9. Marco Laumanns, Lothar Thiele, Eckart Zitzler, and Kalyanmoy Deb. Archiving with Guaranteed Convergence and Diversity in Multi-Objective Optimization. In W. B. Langdon et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 439–447, San Francisco, California, July 2002. Morgan Kaufmann Publishers.
10. Günter Rudolph and Alexandru Agapie. Convergence Properties of Some Multi-Objective Evolutionary Algorithms. In *Proceedings of the 2000 Conference on Evolutionary Computation*, volume 2, pages 1010–1016, Piscataway, New Jersey, July 2000. IEEE Press.
11. Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications.* PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.
12. Eckart Zitzler, Marco Laumanns, Lothar Thiele, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Why Quality Assessment of Multiobjective Optimizers Is Difficult. In W. B. Langdon et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 666–673, San Francisco, California, July 2002. Morgan Kaufmann Publishers.