Using Unconstrained Elite Archives for Multi-Objective Optimisation.

Jonathan E. Fieldsend, *Student Member*, IEEE,
Richard M. Everson, *Member*, IEEE,
and Sameer Singh, *Member*, IEEE.

PANN Research Group, Department of Computer Science,
University of Exeter, Exeter EX4 4PT, UK

Abstract

Multi-Objective Evolutionary Algorithms (MOEAs) have been the subject of numerous studies over the past 20 years. Recent work has highlighted the use of an active archive of elite, non-dominated solutions to improve the optimisation speed of these algorithms. However preserving all elite individuals is costly in time (due to the linear comparison with all archived solutions needed before a new solution can be inserted into the archive). Maintaining an elite population of a fixed maximum size (by clustering or other means) alleviates this problem, but can cause retreating (or oscillitory) and shrinking estimated Pareto fronts - which can affect the efficiency of the search process. New data structures are introduced to facilitate the use of an unconstrained elite archive, without the need for a linear comparison to the elite set for every new individual inserted. The general applicability of these data structures is shown by their use in an Evolutionary Strategy based MOEA and a Genetic Algorithm based MOEA. It is demonstrated that MOEAs using the new data structures run significantly faster than standard, unconstrained archive MOEAs, and result in estimated Pareto fronts significantly ahead of MOEAs using a constrained archive.

It is also shown that the use of an unconstrained elite archive permits robust criteria for algorithm termination to be used, and that the use of the data structure can also be used to increase the speed of algorithms using ϵ-dominance methods.

Index Terms

Optimisation, Genetic Algorithms, Evolutionary Strategies, Multiple Objectives.

1 INTRODUCTION

Frequently a number of competing objectives have to be traded against one another whilst seeking a viable solution to a given problem, often without any *a priori* knowledge of exactly how the objectives interact with one another. For instance, in product design a firm may wish to maximise the performance of an appliance whilst also trying to minimise its production

cost. These two objectives cannot typically be met by a single solution, so, by adjusting the various design parameters, the firm may seek to discover what possible combinations of these two objectives are available, given a set of constraints (for instance legal requirements and size limits of the product).

In [1] for example, multi-objective optimisation is applied to four performance measures of a gas turbine, in [2] different loads in trusses are the competing objectives to be minimised and in [3] different properties of a pressurised water reactor load pattern are optimised.

The curve (for two objectives) or surface (more than two objectives) that describes the optimal trade-off possibilities between objectives is known as the Pareto front [4]. A feasible solution lying on the Pareto front cannot improve any objective without degrading at least one of the others, and, given the constraints of the model, no solutions exist beyond the true Pareto front. The goal, therefore, of multi-objective algorithms is to locate the Pareto front of these *non-dominated* solutions.

Multi-Objective Evolutionary Algorithms (MOEAs) represent a popular approach to solving these types of problem by using evolutionary search techniques. MOEAs have been in use for a considerable length of time now: Beale and Cook in 1978 used a random search technique in an attempt to simultaneously minimise a number of objectives in an aircraft simulator [5]. However, it is the work of Schaffer in 1985 [6], which recognised the need to return a set of solutions, that has been widely quoted as the first MOEA study [7, 8, 9]. The use of Evolutionary Algorithms (EAs) as the tool of choice is due to such problems being typically complex, with both a large number of parameters to be adjusted, and several objectives to be optimised. In addition, EAs, which maintain a population of solutions, are able to explore several parts of the Pareto front simultaneously.

Most recent investigations in the area [1, 7, 10, 11, 12] focus on a MOEA's ability to produce an accurate estimate of the Pareto front. Zitzler et al. [9] present a comparative study, on six test functions introduced by Deb [13], of a number of the most widely used MOEAs, including Fonseca and Fleming's multiobjective EA (FFGA) [1], the Niched Pareto Genetic Algorithm (NPGA) [11], Hajela and Lin's weighted-sum approach (HLGA) [2], the Vector Evaluated Genetic Algorithm (VEGA) [6] and the Nondominated Sorting Genetic Algorithm (NSGA) [12]. Their study suggests that their Genetic Algorithm (GA) based Strength Pareto Evolutionary Algorithm (SPEA) outperforms the other algorithms, with it consistently recording better results as measured by the $\mathcal{C}$ metric [9, 14, 15] on a number of the test functions. In an earlier paper [15] Zitzler and Thiele also demonstrated SPEA's superior performance in comparison to four other MOEAs on a 0/1 knapsack problem. Another MOEA which has demonstrated significant results is the Evolutionary Strategy (ES) based Pareto Archived Evolutionary Strategy (PAES) of Knowles and Corne [16, 17]. Both these MOEAs incorporate elitism and recent work by Laumanns et al. [18] provides a unified model for MOEAs with elitism (called UMMEA). The elitism within UMMEA is achieved by using an archive set of non-dominated solutions in addition to the usual GA population (or as a source of parents for ES perturbation). SPEA, PAES and the UMMEA all use an archive limited to a fixed maximum number of individuals, presumably to avoid the computational costs of maintaining a large archive.

It is shown in this paper that a consequence of restricting the number of solutions in the elite front can be shrinking [19] and oscillating/retreating estimated Pareto fronts [20, 21, 22]. These problems also occur in SPEA, PAES and other existing MOEAs which use a truncated elite archive. A remedy to this situation is simply to retain all the non-dominated solutions found (as an active input to the continuing search process), as, for example, used by

Parks and Miller [3]; however, this approach can be very time consuming (as any individual inserted into the elite archive must first be compared to every individual already present in the archive). It is important to note that in a large number of studies an elite offline store of solutions is maintained which is unbounded, even when truncation takes place in the active population, therefore the linear time update costs are still incurred (in addition to the time cost of truncation).

New data structures, called *dominated* and *non-dominated trees* are introduced in this study that permit faster searching of the elite archive, allowing even very large active elite sets to become feasible. It is shown that the use of the dominated and non-dominated trees in basic ES and GA MOEAs on a number of test sets leads to faster computation time in comparison to maintaining an unconstrained archive as a linear list. Also, the estimated Pareto front discovered is significantly better than that found with a truncated elite archive.

In addition, the use of an unconstrained elite archive is also shown to enable the introduction of a robust algorithm termination methodology, which has until now been largely absent from the MOEA literature.

The paper has the following structure: in Section 2 Pareto optimality is reviewed; in Section 3 the UMMEA is briefly described. In Section 4 a brief overview of the problems of elite set truncation is given. In Section 5 implementational considerations of the use of an unconstrained archive are discussed, the new data structures introduced and implementation described.

The results of a set of experiments quantifying the effects of using the data structures are reported in Section 6, together with the performance measures used. In Section 7 a set of robust stopping criteria are introduced, to replace the current methodology of *ad hoc* algorithm termination. The paper concludes with a discussion in Section 8.

2 PARETO OPTIMALITY

Most recent work on MOEAs hinges on the notions of non-dominance and Pareto optimality, which are now briefly reviewed.

The multi-objective optimisation problem seeks to simultaneously extremise D objectives:

$$y_i = f_i(\mathbf{x}), \qquad i = 1, \ldots, D \tag{1}$$

where each objective depends upon a vector $\mathbf{x}$ of P parameters or decision variables. The parameters may also be subject to the J constraints:

$$e_j(\mathbf{x}) \geq 0, \qquad j = 1, \ldots, J. \tag{2}$$

Without loss of generality it is assumed that the objectives are to be minimised, so that the multi-objective optimisation problem may be succinctly stated as:

$$\text{Minimise} \quad \mathbf{y} = f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_D(\mathbf{x})) \tag{3}$$

$$\text{subject to} \quad e(\mathbf{x}) = (e_1(\mathbf{x}), e_2(\mathbf{x}), \ldots, e_J(\mathbf{x})) \geq 0 \tag{4}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_P) \in X$ and $\mathbf{y} = (y_1, y_2, \ldots, y_D)$.

When faced with only a single objective an optimal solution is one which minimises the objective given the model constraints. However, when there is more than one objective to be minimised it is clear that solutions exist for which performance on one objective cannot be

improved without sacrificing performance on at least one other. Such solutions are said to be *Pareto optimal* [8] and the set of all Pareto optimal solutions is said to form the Pareto front.

The notion of *dominance* may be used to make Pareto optimality more precise. A decision vector $\mathbf{u}$ is said to *strictly dominate* another $\mathbf{v}$ (denoted $\mathbf{u} \prec \mathbf{v}$) iff

$$f_i(\mathbf{u}) \leq f_i(\mathbf{v}) \quad \forall i = 1, \ldots, D \quad \text{and} \quad f_i(\mathbf{u}) < f_i(\mathbf{v}) \quad \text{for some } i \tag{5}$$

Less stringently, $\mathbf{u}$ *weakly dominates* $\mathbf{v}$ (denoted $\mathbf{u} \preceq \mathbf{v}$) iff

$$f_i(\mathbf{u}) \leq f_i(\mathbf{v}) \quad \forall i = 1, \ldots, D \tag{6}$$

A set of M decision vectors $\{\mathbf{w}_i\}$ is said to be a *non-dominated set* (an estimate of the Pareto front) if no member of the set is dominated by any other member:

$$\mathbf{w}_i \not\prec \mathbf{w}_j \quad \forall i, j = 1, \ldots, M \tag{7}$$

3 THE UNIFIED MODEL

As evinced by a number of comparative studies [9, 14, 15] the SPEA provides an effective methodology for multi-objective optimisation problems, as does the PAES [16, 17]. Both algorithms can be seen as variants of the Unified Model for Multi-objective Evolutionary Algorithms, UMMEA, introduced by Laumanns et al. [18], as is outlined in Algorithm 1.

Algorithm 1 The sequential unified multi-objective evolutionary algorithm [18]. F_t denotes the elite archive, B_t the general (search) population and p_t^e the elitism intensity at generation t.

```
1:  t := 0
2:  (F_0, B_0, p_0^e) := initialise()
3:  while terminate(F_t, B_t, t) = false
4:      t := t + 1
5:      F'_t := update(F_{t-1}, B_{t-1})
6:      F_t := truncate(F'_t)
7:      p_t^e := adapt(F_t, B_{t-1}, p_{t-1}^e)
8:      B_t := vary(sample(evaluate(F_t, B_{t-1}, p_t^e)))
9:  end
```

The algorithm summarises the UMMEA framework in terms of stochastic operators. The general genetic population B_0 is initialised, together with the elite archive F_0 (generally to an empty set). At each generation the elite archive is augmented to form F'_t by incorporating those solutions in B_{t-1} which are not dominated by any members of $F_{t-1} \bigcup B_{t-1}$; in addition any elements of F_{t-1}which are dominated by members of B_{t-1} are deleted from F'_t. For reasons of computational efficiency the new archive F'_t is then truncated to create the archive F_t, usually to some fixed size. In the SPEA this truncation is achieved by clustering, and in PAES a density based method is used. Crossover/mutation/perturbation are abstractly represented by the $vary()$ operator. Individuals are selected ($sample()$) from F_t and B_{t-1} based on fitness ($evaluate()$) for crossover/mutation/perturbation; the 'elitism intensity', p_t^e, controlling the probability that an individual from the elite archive rather than the general population will be selected.

4 PROBLEMS WITH CONSTRAINED ELITE SETS

4.1 Shrinking Pareto fronts

For computational efficiency many MOEAs limit the size of their elite archive. In SPEA, for instance, this is achieved by clustering (in objective space) the individuals comprising F'_t, and replacing clusters by the individual closest to the cluster centroid.

However, an artifact of truncation is that the Pareto front (as represented by the truncated archive) can shrink, if individuals that define the boundaries of the estimated Pareto front[1] are removed (referred to here as the *extremal individuals).* If subsequent evolution fails to rediscover the extremal individuals repeated clustering will shrink the Pareto front and the final estimate Pareto set will lie across a narrow subset of the true frontier.

This effect is present even in an offline 'dormant' estimated Pareto front (that is, one that acts as a passive store for non-dominated individuals, which play no part in the search process), as search will not have been directed toward the extremal values. It is interesting to note that after the criticism [9, 10, 12] of Schaffer's VEGA [6] because of its bias toward extremal values, that its replacements should in turn be biased toward search in the centre of the front. (This is indeed supported by the results presented in Zitzler and Thiele [15] and Zitzler et al. [9], where VEGA outperforms NPGA, FFGA and HLGA).

The shrinking front effect can be detrimental in two ways. The main consequence is the narrow extent of the estimated front; secondly, extra search time is required in order to rediscover the extremes of the estimated Pareto front.

These problems are easily circumvented by removing the extremal individuals from the truncation process and passing them directly to F_t. Here this is referred to as the *pinning* of extremal individuals, and is also used in a recent extension to SPEA [19].

4.2 Persistent 'frontal' set

The elite archive is in essence a memory of where the algorithm has reached in previous generations in its estimation of the Pareto front and should contain the 'best' estimate of the Pareto front at any stage. The estimated front should 'advance' in the sense that no individual in F_t should be dominated by any member of an earlier set, $F_0, \ldots, F_{t-1}$. Informally, it is said that an individual $\mathbf{x}$ lies *behind* the front if a member of the elite set dominates $\mathbf{x}$. However, the requirement of many MOEAs that F_t be limited to $\widehat{M}$ members can produce 'retreating' or, more commonly, 'oscillating' estimates of the Pareto front. In these cases members of F_t may lie behind the earlier estimates of the Pareto set. The origin of this behaviour is now described.

An illustration of a retreating front is shown in Figure 1. Figure 1a illustrates an archive, F_t, with a maximum of $\widehat{M} = 6$ members. In Figure 1b a new non-dominated member (drawn as a filled circle) has entered the set from the current population. Since there are now 7 elements in F'_t, one must be removed by clustering; the pair of solutions nearest each other form a cluster of two (circled) and one of them (chosen at random) is deleted, resulting in truncated archive F_t shown in Figure 1c. If at a subsequent generation a new element enters the full archive (Figure 1d), the clustering process will truncate the archive to the set as shown in Figure 1e. This results in an archive F_t (Figure 1g) containing an element that lies

[1] Containing what are referred to as component minima and maxima in [18], in the case of the true Pareto set.

behind (is dominated by) elements of the original archive set (Figure 1a). Repetitions of this process can lead to the estimated front retreating or, more commonly, oscillating as the front advances in the MOEA search stage but retreats during truncation. This possibility was first noted by Hanne [20] in different situation. In Hanne's MOEA (an applied example of which is in [23]) a ES($\lambda + \mu$) scheme was used, with a child replacing the parent if the parent does not dominate the child. As such the population that Hanne used was not a Pareto archive as it was not a non-dominated set (eq. 7); indeed, the population constructed in Hanne may have only one Pareto solution at any generation. The oscillation highlighted by Hanne is generated as a result of the search process as opposed to the truncation of the elite set. Laumanns et al. [21], however, note its application to all elite archive truncation methods (but again without empirical examples) and is used as an impetus for their development of ϵ-dominance and ϵ-approximate Pareto sets.

A simple empirical example of oscillation can be shown using a ES(1+1) MOEA to optimise the ZDT1 function of [9]. The ZDT1 function was optimised using clustering to truncate the archive to 20 individuals; however, a dormant archive of unlimited size containing all non-dominated individuals was also maintained. Figure 2 shows the percentage of individuals in the truncated F_t which are dominated by members of the dormant unconstrained archive. As the number of generations progresses the number of dominated individuals in the clustered archive increases and then oscillates around the 40% mark: clearly the search process is being forced to rediscover portions of the front lost in the truncated archive but retained in the unconstrained archive. (By generation 100,000 the dormant archive consisted of 95 individuals.) Since the extremal individuals were pinned, the effect seen is oscillation, rather than shrinking or retreat.

This artifact has two main consequences. First, search time is wasted 'rediscovering' individuals and regions that have been eliminated by truncation. Secondly, convergence to the true Pareto front is impaired. Numerical simulations show that this oscillation is particularly serious when the estimated front lies close to the true front; the oscillation can prevent convergence to the true front leading to poor estimates and difficulties in assessing convergence.

In the light of the artifacts discussed here, it is recommended that a secondary population of *all* currently non-dominated individuals found during the evolutionary search is used actively within the search process. In terms of the UMMEA (Algorithm 1), this study advocates dropping the *truncate*() operation. This set of all currently non-dominated individuals is referred to as the *frontal set* F_t. This approach has previously been adopted by, for example, Murata and Ishibuchi [24] and Parks and Miller [3], and it should be noted that, while Laumanns et al [18] recommend the truncation of the external set, one of their numerical studies retains all non-dominated individuals. However, the issue of time cost of using an unconstrained elite archive has not been addressed, be it in the case of an active elite archive, or a dormant archive as recommended in [8].

5 USING AN ACTIVE FRONTAL SET

In most MOEAs using an active archive, the archive must be searched at two distinct junctures:

1. When representative individuals are selected for (e.g.) binary tournament selection (in the case of a GA) or for perturbation (in the case of an ES).

2. When individuals are compared with F_t to determine whether they dominate or are

dominated by members of F_t. A data structure to facilitate searching F_t which is faster than a linear list is the *dominated tree*.

Since the frontal set contains all the currently non-dominated decision vectors found, it may become very large as the search progresses. In order for an MOEA using one to be computationally viable, efficient ways of selecting elements of F_t for crossover, mutation and perturbation and of storing F_t to permit rapid query, insertion and deletion of its elements must be found. These problems and their solutions are now discussed.

5.1 Selection

In a MOEA implementing an unconstrained elite archive many more individuals than necessary may be available for selection in whatever evolutionary processes it uses. In this case the manner in which the *sample* () operator (Algorithm 1) selects individuals becomes important. Uniform random selection of individuals from F_t artificially concentrates the search on already densely populated regions of the front. It is therefore helpful to select a number of representative individuals. An approach could be to use the SPEA clustering method to find representatives, however, this proves to be too time consuming for large frontal populations. Here Partitioned Quasi-Random Selection (PQRS) is introduced as a selection routine to be used in conjunction with active unconstrained elite archives. Suppose that N elite individuals are required for selection.

In PQRS one of the D objective dimensions is partitioned into $(N-1)$ bins of equal width and an individual is selected at random from each of these bins. This ensures that individuals are selected uniformly across the extent of the front in the selected dimension. The objective dimension selected for partitioning rotates through the D dimensions with the generation, t. The individual on the extreme of the objective dimension is always selected ensuring the pinning discussed earlier.

An example of PQRS in a $D = 2$ objective problem is shown in Figure 3. Here $N = 5$ individuals are to be selected from a frontal population of $M = 24$ (where $M = |F_t|$ denotes the current size of the frontal set). As one extremal individual is selected immediately, four additional individuals must be selected from F_t. Having selected an objective coordinate, the frontal set is partitioned on that coordinate into $(N-1)$ equally spaced bins. In Figure 3 this can be seen to be four bins for the selected dimension, each spanning $1/4$ of the range of the front on that dimension. An individual in each bin is selected by generating a random number uniformly distributed across the range of the bin, and selecting the closest individual. If a bin is empty (for instance due to a discontinuity in the Pareto front), the closest individual from the entire front is used. In addition, no individual is selected twice (unless $M < N$).

Note that in SPEA clustering is used to reduce the archive before individuals are selected for binary tournament selection; in contrast, this method does not reduce the frontal set population: PQRS only selects individuals for breeding and does not remove them from F_t. This approach is similar to that used in PAES, however the grid knowledge does not need to be maintained and updated [17].

Rapid selection from the frontal set is enabled by maintaining D binary trees, one for each objective dimension. This means that each selection takes $\mathcal{O}(2\log_2 M)$ comparisons as opposed to $\mathcal{O}(M)$ for a linear search. Since the frontal set is constantly changing this can conveniently be implemented using self balancing trees (e.g. AVL or Red-Black trees [25]), or doubly linked lists.

If only one elite individual is desired (for instance a ES(1+1) is being used), then objective space is still separated into $(N-1)$ bins, with the particular bin (or extreme individual) selected uniformly, before being sampled. The larger the value of N, the less the selection process will be affected by dense areas of the archive, and the less the bias toward selecting the extremal individual.

5.2 Efficient storage of the frontal set

The second and greater constraint on using a large active frontal set is the number of comparisons that must be made with individuals in the frontal set at each generation. When the archive is small, for instance $M = 20$ (as in [9]), the time for a linear search of F_t is negligible. However, with no limits to the size of the frontal set in an MOEA, the linear search of 1000 individuals (for instance) before assigning an individual as non-dominated, may be simply too costly to make the method practical. Therefore intelligent storage is needed before the frontal set approach is viable.

5.3 DOMINATED TREES

To determine whether an individual, $\mathbf{y} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_D(\mathbf{x}))$, should become a member of the frontal set, F,[2] it must first be checked that $\mathbf{y}$ is not dominated by any element of F. At the same time any elements of F that are dominated by $\mathbf{y}$ should be deleted from F. When the frontal set is small a simple linear search is sufficiently cheap to perform these checks. However, as the size of the frontal set grows the cost of querying the frontal set becomes prohibitive. In this section two data-structures are described— *dominated trees* and *non-dominated trees* — for storing, rapidly querying, and updating the frontal set[3].

Here it is convenient to regard members of the frontal set and individual(s) from the search population as points $\mathbf{y}$ in D-dimensional space. Geometrically, finding individuals in F that dominate $\mathbf{y}$ amounts to finding frontal individuals that lie to the 'south-west' or 'left and below' $\mathbf{y}$. More formally, the set of dominating individuals is:

$$\{\mathbf{z} \in F \,:\, \mathbf{z}_i \leq \mathbf{y}_i \text{ for all } 1 \leq i \leq D \text{ and } \mathbf{z}_j < \mathbf{y}_j \text{ for at least one } 1 \leq j \leq D\} \tag{8}$$

It would be possible to use kd-trees [26, 25] or range trees [27, 25], but these are both suited to querying F for elements which lie in bounded (hyper-) rectangles. Priority trees, developed by McCreight [28], are suited to rectangular queries in which the rectangle is unbounded on a *single* side. Sun and Steuer [29] describe an alternative data structure adapted for answering queries about domination and non-domination; which has been extended recently by Mostaghim et al. [30].

The 'dominates' relation imposes a partial order on individuals. However, since the elements of F are mutually non-dominating, this relation cannot be used directly to construct, for example, a binary tree to enable fast searching.

Instead the dominated tree consists of a list of $L = \lceil M/D \rceil$ *composite points* ordered by the weakly-dominates relation, $\preceq$:

$$\mathcal{T} = \{\mathbf{c}_L \preceq \ldots \preceq \mathbf{c}_2 \preceq \mathbf{c}_1\} \tag{9}$$

[2]Since the genetic generation plays no role here, the subscript t has been dropped.

[3]An example implementation of dominated and non-dominated trees is available from `http://www.dcs.ex.ac.uk/people/reverson/dominated-trees`

Usually, the stronger condition, $\mathbf{c}_i \prec \mathbf{c}_j$ iff $i > j$, will hold. The coordinates of each composite point are defined by (up to) D elements of F, the *constituent points* of a composite point. An example of a dominated tree in two dimensions is shown in figure 4.

Denote by Y_i the constituent points of $\mathbf{c}_i$, namely the D-tuple defining the coordinates of $\mathbf{c}_i$; so that if

$$Y_i = \langle \mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(D)} \rangle \tag{10}$$

then the dth coordinate of the composite point is the dth coordinate of $\mathbf{y}^{(d)}$: $\mathbf{c}_i = \mathbf{y}_d^{(d)}$. Dominated trees are constructed to have the property that if $\mathbf{c}_i \prec \mathbf{y}$ then all the constituent points of $\mathbf{c}_i$ (at least) weakly dominate $\mathbf{y}$:

$$\text{If} \quad \mathbf{c}_i \preceq \mathbf{y} \quad \text{then} \quad \mathbf{y}^{(d)} \preceq \mathbf{y} \quad \forall \mathbf{y}^{(d)} \in Y_i \tag{11}$$

It follows from (11) that if $\mathbf{c}_i \prec \mathbf{c}_j$ then the constituent points of $\mathbf{c}_i$ also weakly dominate $\mathbf{c}_j$. Thus, for example, in Figure 4 the constituent points of $\mathbf{c}_4, \mathbf{c}_5$ and $\mathbf{c}_6$ dominate $\mathbf{c}_3$. Note, however, that they do not necessarily dominate the constituent points of $\mathbf{c}_3$, namely $\mathbf{y}_3$ and $\mathbf{y}_6$.

It should be emphasised that the points forming the tree in Figure 4 do not form a non-dominated set. Dominated and non-dominated trees do not require that the constituent elements to form a non-dominated set. The dominated tree is illustrated with a general set of two-dimensional points, because non-dominated sets of two-dimensional elements have the peculiar property that listing the points in order of increasing first coordinate (objective), y_1, is equivalent to listing them in order of decreasing second coordinate, y_2. This is easily seen by considering two mutually non-dominating points, say, $\mathbf{u} \in \mathbb{R}^2$ and $\mathbf{v} \in \mathbb{R}^2$: without loss of generality, suppose that $\mathbf{u}_1 < \mathbf{v}_1$. If $\mathbf{u}_2 \leq \mathbf{u}_2$ then $\mathbf{u} \prec \mathbf{v}$, contradicting the fact that they are mutually non-dominating, thus it may be concluded that $\mathbf{u}_2 \geq \mathbf{v}_2$. This ordering property is special to two dimensions, and can be misleading if one tries to generalise from sketches in two dimensions. The points and the tree illustrated in figure 4 are more akin to the general ($D > 2$) case. Dominated trees also have applications to general sets (that is, sets whose elements are not mutually non-dominating), such as answering queries about enclosing rectangles; see, for example, [28, 31].

5.3.1 Construction

Construction of a dominated tree from M points $F = \{\mathbf{y}_m\}_{m=1}^M$ is described in Algorithm 2 and proceeds as follows. The first composite point $\mathbf{c}_1$ is constructed by finding the individual $\mathbf{y}_m$ with maximum first coordinate; this value forms the first coordinate of the composite point:

$$\mathbf{c}_{1,1} = \max_{\mathbf{y}_m \in F} (\mathbf{y}_{m,1}) \tag{12}$$

Thus, for example, in figure 4, $\mathbf{c}_{1,1} = \mathbf{y}_{1,1}$. This individual $\mathbf{y}_m$ is now associated with $\mathbf{c}_1$ and removed from F. Likewise the second coordinate of $\mathbf{c}_1$ is given by the maximum second coordinate of the points remaining in F: $\mathbf{c}_{1,2} = \max_{\mathbf{y}_m \in F \setminus \mathcal{T}} (\mathbf{y}_{m,2})$; in figure 4 $\mathbf{c}_{1,2} = \mathbf{y}_{5,2}$. This procedure is repeated to construct $\mathbf{c}_2$ and subsequent composite points until all M elements of F are associated with the tree. In general the dth coordinate of the ith composite point is given by:

$$\mathbf{c}_{i,d} = \max_{\mathbf{y}_m \in F \setminus \mathcal{T}} (\mathbf{y}_{m,d}) \tag{13}$$

Algorithm 2 Construction of a dominated tree

Inputs:
$F = \{\mathbf{y}_m\}_{m=1}^{M}$ Vectors to form the dominated tree

$\mathcal{T} := \emptyset$
$L = \lceil M/D \rceil$
for $i := 1, \ldots, L$
  for $d := 1, \ldots, D$
    $\mathbf{y} := \max_{\mathbf{y}_m \in F}(\mathbf{y}_{m,d})$
    $\mathbf{c}_{i,d} := \mathbf{y}_d$ Keep pointers from $\mathbf{y}_m$ to and from $\mathbf{c}_i$
    if $|F| \neq 1$
      $F := F \setminus \mathbf{y}$ Delete $\mathbf{y}$ from F
    end
  end
  $\mathcal{T} := \{\mathbf{c}_i \preceq \mathbf{c}_{i-1} \preceq \ldots \preceq \mathbf{c}_1\}$ Append $\mathbf{c}_i$ to $\mathcal{T}$
end

Note that in construction of the final composite point (that is, the composite point that dominates all other composite points) the M elements of F may have been used before all the D coordinates of the final composite point $\mathbf{c}_L$ have been defined. As illustrated by $\mathbf{c}_7$ in figure 4, the last remaining point in F ($\mathbf{y}_{13}$ in figure 4) is reused to define the remaining coordinates.

If PQRS trees have been constructed, they can be used to efficiently find the successive constituent points of each composite point.

It is clear from the construction of $\mathcal{T}$ that it possesses properties (9) and (11). Since (except possibly for the dominating composite point) D elements of F are used in the construction of each composite point, the number of composite points in $\mathcal{T}$ is $L = \lceil M/D \rceil$.

5.3.2 Query

Given a test point $\mathbf{q}$, the properties of dominated trees can be used as follows to discover which points in F dominate $\mathbf{q}$. Although the dominated tree is efficiently implemented as a binary tree, the query procedure is most easily described in terms of an ordered list of L composite points. First, the list is searched to find the indices h and l of composite points $\mathbf{c}_h$ and $\mathbf{c}_l$ that dominate and are dominated by $\mathbf{q}$ respectively:

$$h = \begin{cases} L+1 & \text{if } \mathbf{q} \prec \mathbf{c}_L \\ \min\{i : \mathbf{c}_i \prec \mathbf{q}\} & \text{otherwise} \end{cases} \tag{14}$$

and

$$l = \begin{cases} 0 & \text{if } \mathbf{c}_1 \prec \mathbf{q} \\ \max\{i : \mathbf{q} \prec \mathbf{c}_i\} & \text{otherwise} \end{cases} \tag{15}$$

Also denote by $\mathbf{c}_H$ the 'least' composite point that strictly dominates $\mathbf{c}_h$:

$$H = \min\{i : \mathbf{c}_i \prec \mathbf{c}_h\} \tag{16}$$

For the query point illustrated in Figure 4, $l = 2$, $h = 5$ and $H = 6$. (Note that it is not necessarily true that $H = h + 1$.) Since $\mathbf{c}_h \prec \mathbf{q}$ it is clear from 11 that all the constituent

points of the composite points $\mathbf{c}_i$ that dominate $\mathbf{c}_h$, $H \leq i \leq L$, also dominate $\mathbf{q}$. (Note that the constituent points of $\mathbf{c}_h$, and indeed and $\mathbf{c}_i$ that only weakly dominate $\mathbf{c}_h$, need not dominate $\mathbf{q}$; in figure 4 $\mathbf{c}_5 \prec \mathbf{q}$, but $\mathbf{y}_9 \not\prec \mathbf{q}$.) Also, since $\mathbf{q} \prec \mathbf{c}_l$ and the constituent points of $\mathbf{c}_l$ have at least one coordinate equal to a coordinate of $\mathbf{c}_l$, it may be concluded that $\mathbf{q}$ is not dominated by any of the constituent points of $\mathbf{c}_1, \ldots, \mathbf{c}_l$. Each constituent point of $\mathbf{c}_i$ $l < i < H$ must be checked individually to determine whether it dominates $\mathbf{q}$; in figure 4 the points $\mathbf{y}_3, \mathbf{y}_6, \mathbf{y}_4, \mathbf{y}_8, \mathbf{y}_9, \mathbf{y}_{10}$ must be individually checked.

Note that when determining whether $\mathbf{q}$ is to be included in F, it can be immediately rejected if $h < L$ because it is certainly dominated by at least one of the constituents of $\mathbf{c}_L$.

Since the composite points are arranged as a sorted list, determination of l and h takes $\mathcal{O}(\log_2(M/D))$ domination comparisons each. Hence the total number of domination comparisons required to enumerate the elements in F that dominate a query point is $\mathcal{O}(2\log_2(M/D)+DK)$, where K is the number of points that have to be checked individually. Clearly, certain configurations of F and $\mathbf{q}$ can result in all elements of F being checked — in linear time. However, such arrangements are seldom encountered in practise. This rapid checking permits very large archive sets to be efficiently maintained.

If it is determined that $\mathbf{q}$ is to be included in F (because it is not dominated by any element of F), those elements of F which are dominated by $\mathbf{q}$ must be identified and deleted from F. Queries about which elements of F are dominated by $\mathbf{q}$ can be answered using the dominated tree, however, it may be inefficient. This is because although $\mathbf{q} \prec \mathbf{c}_i$ for $1 \leq i \leq l$, $\mathbf{q}$ need not dominate the constituent points of these $\mathbf{c}_i$ and the constituent points must therefore be checked individually. Thus in Figure 4 for example, $\mathbf{q} \prec \mathbf{c}_2 \prec \mathbf{c}_1$, but $\mathbf{y}_5$ and $\mathbf{y}_7$ are not dominated by $\mathbf{q}$. The *non-dominated tree* is a data structure which permits this sort of query to be answered efficiently.

Non-dominated trees are analogous to dominated trees. A non-dominated tree consists of an ordered list of composite points:

$$\mathcal{T} = \{\mathbf{c}_1 \preceq \mathbf{c}_2 \preceq \ldots \preceq \mathbf{c}_L\} \tag{17}$$

In addition non-dominated trees are constructed with the property that if $\mathbf{y} \prec \mathbf{c}_i$, then all the constituent points of $\mathbf{c}_i$ are (at least) weakly dominated by $\mathbf{y}$:

$$\text{If} \quad \mathbf{y} \prec \mathbf{c}_i \quad \text{then} \quad \mathbf{y} \preceq \mathbf{y}^{(d)} \quad \forall \mathbf{y}^{(d)} \in Y_i \tag{18}$$

An example of a non-dominated tree is shown in Figure 5. Construction and querying of non-dominated trees is analogous to dominated trees and they are not discussed further here.

5.3.3 Insertion and deletion.

Elements are continually added and deleted from the frontal set during the course of an optimisation. It is therefore important that the data structure used to support F can be modified dynamically. Online insertion of a new point $\mathbf{y}$ is straight-forwardly accomplished as outlined in the Algorithm 3.

If the new point dominates the $\mathbf{c}_L$, the composite point that dominates all others, then a new composite point, $\mathbf{c}'$, that is created with all its coordinates defined by $\mathbf{y}$ (steps 1-4 in Algorithm 3). It is clear that $\mathbf{c}' \preceq \mathbf{c}_L$, so $\mathbf{c}'$ may be appended to $\mathcal{T}$ as $\mathbf{c}_{L+1}$, preserving the ordering property (9).

Algorithm 3 Insertion into a dominated tree.

Inputs:

	$\mathcal{T} = \{\mathbf{c}_L \preceq \ldots \preceq \mathbf{c}_1\}$	Dominated tree of L composite points
	$\mathbf{y}$	Point to be inserted into $\mathcal{T}$
1:	if $\mathbf{y} \preceq \mathbf{c}_L$	
2:	for $k = 1, \ldots, D$	
3:	$\mathbf{c}'_k := \mathbf{y}_k$	
4:	end	
5:	$\mathcal{T} := \{\mathbf{c}' \preceq \mathbf{c}_L \preceq \ldots \preceq \mathbf{c}_1\}$	
6:	else	
7:	$j := \arg\min(\mathbf{c}_i \succ \mathbf{y})$	Binary search
8:	for $k = 1, \ldots, D$	
9:	$\mathbf{c}'_k := \max(\mathbf{y}_k, \mathbf{c}_{j+1,k})$	
10:	end	
12:	for $k = L, \ldots, j+1$	
13:	$\mathbf{c}_{k+1} := \mathbf{c}_k$	Relabel
14:	end	
15:	$\mathbf{c}_{j+1} := \mathbf{c}'$	$\mathcal{T} := \{\mathbf{c}_L \preceq \ldots \preceq \mathbf{c}_{j+1} \preceq \mathbf{c}' \preceq \mathbf{c}_j \preceq \ldots \preceq \mathbf{c}_1\}$
16:	end	

If $\mathbf{y}$ does not dominate $\mathbf{c}_L$ a bisection search is used to locate composite points, such that $\mathbf{c}_{j+1} \not\succ \mathbf{y} \preceq \mathbf{c}_j$. A new composite point $\mathbf{c}'$ is then constructed with coordinates determined by

$$\mathbf{c}'_k = \max(\mathbf{y}_k, \mathbf{c}_{j,k}) \qquad k = 1, \ldots, D \tag{19}$$

By construction, $\mathbf{c}_{j+1} \preceq \mathbf{c}' \preceq \mathbf{c}_j$, so $\mathbf{c}'$ may be inserted in $\mathcal{T}$ between $\mathbf{c}_{j+1}$ and $\mathbf{c}_j$ while preserving the ordering property (9). In Algorithm 3 this, and the necessary relabelling, is accomplished by lines 12-16. Most practical implementations of ordered lists (such as binary trees or linked lists) are not index based, so this relabelling is not necessary. The binary search of $\mathcal{T}$ takes $\mathcal{O}(\log_2(M/D))$ domination comparisons and the calculation of the coordinates of $\mathbf{c}'$ is equivalent to another domination comparison (D comparisons), so the cost of insertion is $\mathcal{O}(\log_2(M/D))$ domination comparisons.

Figure 6 shows the dominated tree resulting from the insertion of a new point $\mathbf{y}_{14}$ in the tree shown in Figure 4. Note that the tree resulting from insertion is less than optimal in the sense that a point (e.g., $\mathbf{y}_6$) may contribute to more than one composite point.

Deletion of a point $\mathbf{y}_m$ from the tree is slightly more complicated, because the composite point to which $\mathbf{y}_m$ contributed, say $\mathbf{c}_j$, must remain (at least) weakly dominated by $\mathbf{c}_{j+1}$ after the deletion. Assume that in the construction of $\mathcal{T}$ pointers are kept from each element $\mathbf{y}_m$ to the composite point $\mathbf{c}_j$ of which $\mathbf{y}_m$ is a constituent.

If $\mathbf{y}_m$ defines the kth coordinate of $\mathbf{c}_j$, then, as shown in Algorithm 3, upon deletion of $\mathbf{y}_m$, the deleted coordinate of $\mathbf{c}_j$ is replaced by the kth coordinate of $\mathbf{c}_{j+1}$. This assignment ensures that $\mathbf{c}_{j,k} \leq \mathbf{c}_{j+1,k}$, so that $\mathbf{c}_{j+1} \preceq \mathbf{c}_j$, as required by (9). Note also that the constituent points of $\mathbf{c}_j$ after the deletion are either the constituent points of $\mathbf{c}_j$ before the deletion or the constituent points of $\mathbf{c}_{j+1}$. Therefore, if before the deletion $\mathbf{c}_j \preceq \mathbf{y}$ (which implies that all the constituent points of $\mathbf{c}_j$ and $\mathbf{c}_{j+1}$ weakly dominate $\mathbf{y}$), then after the deletion the constituent points of $\mathbf{c}_j$ continue to weakly dominate $\mathbf{y}$, showing that property (11) is preserved.

Algorithm 4 Deletion from a dominated tree $\mathcal{T}$.

Inputs:

$\mathcal{T} = \{\mathbf{c}_L \preceq \ldots \preceq \mathbf{c}_1\}$	Dominated tree of L composite points
$\mathbf{y}_m$ with $\mathbf{c}_{j,k} = \mathbf{y}_{m,k}$	Point to be deleted from $\mathcal{T}$
	$\mathbf{y}_m$ defines the kth coordinate of $\mathbf{c}_j$

```
1:  if j < L
2:      c_{j,k} := c_{j+1,k}          Use coordinate from dominating composite point
3:  else
4:      for u ∈ Y_j \ y_m             Check the remaining constituents of c_j
5:          if u_k > c_{j,k}
6:              c_{j,k} := u_k
7:          end
8:      end
9:  end
```

Deletion of $\mathbf{y}_4$ from the tree shown in Figure 4 is illustrated in Figure 6. Note that if $\mathbf{y}_m$ contributes to more than one composite point, then each of the composite points to which it contributes must be dealt with in turn, beginning with the one which is dominated by all the others. On the other hand, if $\mathbf{y}_m$ is the sole constituent point of $\mathbf{c}_j$ (e.g., $\mathbf{c}_{13}$ in Figure 4) then $\mathbf{c}_j$ is deleted from $\mathcal{T}$ when $\mathbf{y}_m$ is deleted. If pointers are kept from each $\mathbf{y}_m$ to the composite points to which it contributes, then deletion is achieved in constant time.

Algorithm 5 Cleaning a dominated tree $\mathcal{T}$.

Inputs:

$\mathcal{T} = \{\mathbf{c}_L \preceq \ldots \preceq \mathbf{c}_1\}$	Dominated tree of L composite points

```
1:  j := 1
2:  while j < |T|
3:      for y ∈ Y_j
4:          L := |T|
5:          T := append(delete(y, T_{j+1,L}), T_{1,j})
6:      end
7:  end
```

5.3.4 Cleaning

Insertion and deletion operations lead to some points contributing to more than one composite point. The dominated trees therefore contain more composite nodes than the optimum $\lceil M/D \rceil$ and hence increased time is needed to search them. This can be alleviated by periodically 'cleaning' the tree in the following manner. Let $\mathcal{T}_{j,k}$ denote the sub-tree of $\mathcal{T}$ composed of the nodes $j, \ldots, k$:

$$\mathcal{T}_{j,k} = \{\mathbf{c}_k \preceq \ldots \preceq \mathbf{c}_j\} \tag{20}$$

and let the deletion operation (Algorithm 4) be denoted by $\mathit{delete}(\mathbf{y}, \mathcal{T})$. As described in Algorithm 5 cleaning is achieved by successively deleting constituent nodes from all composite

nodes except the least dominating node to which they contribute. This ensures that every point $\mathbf{y}_m$ contributes to exactly one composite point. A cleaned tree is shown in Figure 6. In practise it is not efficient to clean the tree following every insertion and deletion, but occasional cleaning may be triggered when the number of composite points exceeds a threshold, say $2M/D$. Alternatively, if PQRS trees have been constructed they may be used to efficiently be rebuild the dominated tree 'from scratch.'

6 EXPERIMENTS

In order to evaluate the efficiency (in terms of both time and quality of the estimated front) of the new data structures, results of a comparison with the GA and ES MOEAs using (a) full elite archives using PQRS and dominated trees (b) full elite archives with a standard linear search and (c) truncated archives with clustering are presented. New multi-objective test problems are also introduced that have the characteristic of large archive growth which this study wishes to investigate. However, first a brief critique of the current multi-objective test suite is given [9].

The ZDT test functions involve two objectives and have the following structure:

$$\begin{array}{lll} \text{Minimise} & T(\mathbf{x}) & = (f_1(x_1), f_2(\mathbf{x})), \\ \text{where} & f_2(\mathbf{x}) & = g(x_2, \ldots, x_P) \cdot h(f_1(x_1), g(x_2, \ldots, x_p)), \\ \text{and} & \mathbf{x} & = (x_1, \ldots, x_P). \end{array}$$

Although the six ZDT functions represent many different features and levels of difficulty, the first objective (f_1) is always a function solely of the first decision parameter (in fact, for the first four test functions, $f_1(x_1) = x_1$). This simple form of f_1 means that when genes describing x_1 are initialised as uniformly distributed random numbers, the initial estimate of the Pareto front extends over the full range of f_1. Consequently optimisation chiefly consists of minimising f_2, rather than the combined minimisation of both objectives.

This structure also appears to produce the artifact that the growth of the frontal set is quite limited, as shown in figures 11 and 12. As opposed to the experience of other applied studies that have experienced much larger archive growth (e.g. [3]). In addition, all six test functions are only two objective problems.

For the purposes of this study, where the focus is on maintaining large archives, the ZDT functions unfortunately are arguably of limited use. However, the first three are still used in the results section as they are known to the community at large.

In addition three new test functions are introduced, in which all objectives are dependent on all decision parameters. These functions do exhibit large archive growth. All three are combinations of the following five base functions, $B_i(\mathbf{x})$.

- Base functions

$$B_1(\mathbf{x}) = \sum_{i=1}^{m} \left| x_i - \frac{1}{3} \exp\left((i/m)^2 \right) \right|^{\frac{1}{2}} \tag{21}$$

$$B_2(\mathbf{x}) = \sum_{i=1}^{m} \left(x_i - \frac{1}{2} \left(\cos\left(10\pi \left(i/m \right) \right) + 1 \right) \right)^2 \tag{22}$$

$$B_3(\mathbf{x}) = \sum_{i=1}^{m} \left| x_i - \sin^2(i-1) \cos^2(i-1) \right|^{\frac{1}{2}} \tag{23}$$

$$B_4(\mathbf{x}) = \sum_{i=1}^{m} \left| x_i - \frac{1}{4}\left(\cos(i-1)\cos\left(2(i-1)\right)+2\right)\right|^{\frac{1}{2}} \tag{24}$$

$$B_5(\mathbf{x}) = \sum_{i=1}^{m} \left(x_i - \frac{1}{2}\left(\sin\left(1000\pi\,(i/m)\right)+1\right)\right)^2 \tag{25}$$

- Multi-objective functions

$$F_1(\mathbf{x}) = \left(B_1(\mathbf{x}), B_2(\mathbf{x})\right) \tag{26}$$

$$F_2(\mathbf{x}) = \left(B_2(\mathbf{x}), B_3(\mathbf{x}), B_4(\mathbf{x})\right) \tag{27}$$

$$F_3(\mathbf{x}) = \left(B_1(\mathbf{x}), B_3(\mathbf{x}), B_4(\mathbf{x}), B_5(\mathbf{x})\right) \tag{28}$$

In all cases m=30 and $x_i \in [0,1]$.

Parallel coordinated graphs shown in Figure 10 of these test functions, which illustrate the function bounds and the trade-off between objectives (with the lines crossing), on the true Pareto front.

6.1 Comparing Pareto fronts

Comparison of Pareto front estimates is difficult as there are several ways in which a front can be inferior or superior to another. Indeed it is unlikely that any one measure will be sufficient to encompass all desired information when evaluating the output of an MOEA. In this study two performance measures are used to encapsulate different properties of competing non-dominated sets.

First an alternative to the popular $\mathcal{C}$ metric [9, 14, 15] is discussed.

The $\mathcal{C}$ metric is defined as

$$\mathcal{C}(A,B) = \frac{|\{\mathbf{b} \in B : \exists\, \mathbf{a} \in A,\ \mathbf{a} \preceq \mathbf{b}\}|}{|B|},$$

where A and B are two sets of decision vectors and $A, B \subseteq X$.

The $\mathcal{C}$ metric measures the fraction of members of B which are (strictly or weakly) dominated by members of A. As such it measures the quality of A with respect to B. When $\mathcal{C}(A,B) = 1$ all the individuals in B are dominated by solutions in A; $\mathcal{C}(A,B) = 0$ represents the situation in which none of the individuals in B are dominated by any of those in A.

It should be noted that $\mathcal{C}(A,B)$ is not technically a metric, since $\mathcal{C}(A,A) \neq 0$ and $\mathcal{C}(A,B)$ is not symmetrical in its arguments and it doesn't satisfy the triangle inequality. Furthermore, $\mathcal{C}$ has the following undesirable property: suppose that W is a non-dominating set and $A \subseteq W$ and $B \subseteq W$, then $\mathcal{C}(A,B)$ can take on any value in $[0,1]$.

In this study the following modified version of the $\mathcal{C}$ measure is used:

$$\tilde{\mathcal{C}}(A,B) = \frac{|\{\mathbf{b} \in B : \exists\, \mathbf{a} \in A,\ \mathbf{a} \prec \mathbf{b}\}|}{|B|} \tag{29}$$

Now $\tilde{\mathcal{C}}(A,A) = 0$ and, in addition, it measures two mutually non-dominating sets as equivalent. That is, if $A \subseteq W$ and $B \subseteq W$ are each subsets of a non-dominating set W, then $\tilde{\mathcal{C}}(A,B) = 0$.

Nevertheless, $\tilde{\mathcal{C}}$ and $\mathcal{C}$ fail to account for the either the difference in the extent of the fronts being compared or the uniformity of the distribution of points along the front. For example,

Figure 8a illustrates two fronts with similar extent, but points describing A are uniformly distributed along the front, whereas those describing B are clustered in one region. However, $\mathcal{C}(A,B) = \mathcal{C}(B,A) = \tilde{\mathcal{C}}(A,B) = \tilde{\mathcal{C}}(B,A) = 4/12$, even though elements of A dominate elements of B along the majority of their extents. In Figure 8b although B has much greater extent, $\mathcal{C}(A,B) = \tilde{\mathcal{C}}(A,B) = 2/12$, whereas $\mathcal{C}(B,A) = \tilde{\mathcal{C}}(B,A) = 0/12$.

As some of the problems highlighted with the $\mathcal{C}$ and $\tilde{\mathcal{C}}$ measure show, there are a number of properties which are usually desired of estimated Pareto fronts. These include that the 'distance' of the estimated Pareto front to the true Pareto front should be minimised, and that the extent of the estimated front be maximised (a wide range of solutions in objective space be returned). A measure which is designed to include such information is now introduced.

The second measure used in this study is similar conceptually to the performance measure used in [18], and is a measure of the objective space volume that is dominated by one front but not the other. Loosely $\mathcal{V}(A,B)$ is the fraction of the volume of the minimum hypercube containing both fronts that is strictly dominated by members of A but is not dominated by members of B (and therefore lies on the range $[0,1]$). An illustration of this provided in Figure 9 where two continuous fronts A and B have differing extents and also dominate each other in different region of the objective space.

$\mathcal{V}(A,B)$ is defined in the following manner. For any set of D-dimensional vectors Y, let H_Y denote the smallest axis-parallel hypercube containing Y:

$$H_Y = \left\{ \mathbf{z} \in \mathbb{R}^D \ : \ a_i \leq z_i \leq b_i \text{ for some } \mathbf{a}, \mathbf{b} \in Y, \ i = 1, \ldots, D \right\} \tag{30}$$

Now denote by $h_Y(\mathbf{y}) : H_Y \longmapsto [0,1]^D$ the normalising scaling and translation that maps H_Y onto the unit hypercube. This transformation serves to remove the effects of scaling the objectives. Let

$$D_Y(A) = \left\{ \mathbf{z} \in [1,0]^D \ : \ \mathbf{z} \prec h_Y(\mathbf{a}) \ \text{ for some } \mathbf{a} \in A \right\} \tag{31}$$

be the set of points in the hypercube defined by Y which are dominated by the normalised A. Then $\mathcal{V}(A,B)$ is defined as

$$\mathcal{V}(A,B) = \lambda \left(D_{A \bigcup B}(A) \setminus D_{A \bigcup B}(B) \right) \tag{32}$$

where $\lambda(A)$ denotes the Lebesgue measure of A [32].

Despite this rather cumbersome description, $\mathcal{V}(A,B)$ and $\mathcal{V}(B,A)$ are easily calculated by Monte Carlo sampling of $H_{A \bigcup B}$ and counting the fraction of samples that are dominated exclusively by A or B. In this study 50000 samples were taken for Monte Carlo estimates.

The benefit of the volume measure $\mathcal{V}$ is that it will reward sets that are of greater extents when those extents are in front of the comparison set, but not when they are behind, it is not effected by the distribution of points across a front, and it also gives information regarding how far one set is (on average) in front of another.

Unfortunately the $\mathcal{V}$ measure, like the original $\mathcal{C}$ metric, has the property that, if W is a non-dominating set, and $A \subseteq W$ and $B \subseteq W$, both $\mathcal{V}(A,B)$ and $\mathcal{V}(B,A)$ may be positive.

6.2 Algorithm implementation

The implementation of both the ES and GA models use floating point representation of parameters in the individual chromosomes. In order to compare the linear search and the dominated tree method the two versions of each MOEA were each executed 50 times on each test problem,

with the cumulative time saved at each generation (the time spent in the methods dealing with comparison to the archive and selection from the archive). In order to compare the unconstrained and truncated approach, each was executed 50 times on each test problem, and the resultant non-dominated solutions saved at the end of each run. The runs were repeated twice; once for the same number of generations and once for the same empirical run time. In the case of the clustered algorithm no off-line store was maintained, as maintaining this store would take its time cost above the standard (unclustered) approach, which truncation is designed to alleviate. Each simulation was performed using the parameters shown below:

GA

Number of generations : 5000

Search population size : $|B_t| = 20$

Max. archive population size (when clustered) : $\widehat{M} = 50$

Crossover rate : 0.8

Mutation rate : 0.05

ES(1+1)

Number of generations : 100000

Max. archive population size (when clustered) : $\widehat{M} = 50$

Mutation rate : 0.2

Single point crossover was used and the mutator variable was drawn from a Gaussian distribution with 0.1 variance and zero mean. In each of the 50 different runs the MOEAs were initialised from identical decision vector populations of size 100, with the non-dominated individuals residing in these populations forming the initial elite archive. Initialisation of decision vectors was from $U(0,1)$. The random number generators were identically seeded for each of the 50 runs, so that the fronts found by the two unconstrained methods were identical. This means that time comparisons were made on strictly identical update/selection problems. Elitism intensity was 1.0 and the dominated and non-dominated trees were cleaned when they exceeded $1.2M/D$ composite points. In all algorithms selection of elite individuals from the archive used PQRS.

6.3 Timing results

When comparing the speed of the three algorithm types the sum of clock ticks that occurred in the relevant methods of each algorithm (those that dealt with the comparing of a new individual to the archive (and updating), and the selection of and individual from the archive as a parent) was maintained, with these value transformed into seconds[4].

As shown in Tables 3 and 1, the simple GA using the dominated trees were significantly faster than the standard linear approach GAs for all six test problems. The ES dominated tree

[4]The machine used in these experiments was a 1.4GHz AMD Athlon processor with 256Mbytes of DDRAM.

algorithms were significantly faster than the standard linear approach for the three new test problems, however there was no significant difference in speed on the three ZDT functions. The clustering algorithm was faster than both on the 3 and 4 objective problem using the simple ES, and faster than linear or dominated tree searches on all the test functions except F1 using the GA. This is a reflection of the size of the archive set – see figures 11 and12. Table 3 shows the generations for which one algorithm tested using the nonparametric Wilcoxon Signed Ranks Test [33] at the 5% level (2.5% in each tail)[5] is found to be significantly faster than another. As can be seen, both the unconstrained algorithms are initially faster than the clustered approach, however, as their archives grow their time cost per generation (the gradient of the cumulative time curves shown in Figure 13), surpasses that of the clustering approach.

It can also be seen that in a number of cases standard linear approach is initially faster than the dominated tree method. This is due to the time cost of maintaining the data structures outweighing the search cost reduction compared to the linear list, when the archive population is small. A number of additional experiments were therefore run where the time taken to reach different archive sizes was assessed. Again 50 separate runs of the two unconstrained approaches (for both the ES and GA MOEA) were made, with the time taken to reach archive sizes in 50 member increments recorded for each of the test functions. As can be seen in Table 5, for the ZDT functions it is generally the case that after the archive has exceeded 50-100 members the dominated tree method is significantly faster than the linear list. A higher threshold is experienced by the new functions introduced in this study, with the dominated tree not being significantly faster until the archive reaches 150-250 members. This is arguably because the test function surfaces are searched in all objective directions (unlike the ZDT functions that principally consist of 'pushing down' the second objective), and as such have a higher proportion of individuals discovered at the edges at each generation (meaning K is on average larger).

6.4 Performance results

Table 7 shows the $\tilde{\mathcal{C}}$ performance comparison of the fronts found by the unconstrained and truncated algorithms after 100000/5000 ES/GA generations. On all six test functions the unconstrained approach performed significantly better than the constrained approach using the GA over the 50 runs, both in terms of equal generation length and equal computation time. The ES unconstrained method performed significantly better than the clustered method using this measure on test functions F1, F2 and F3 on equal generation length, the clustered method being significantly better on ZDT2 and ZDT3. However, when the clustered algorithm was run for the same amount of computation time as the different dominated tree runs, the unconstrained method was significantly better than the constrained method for all the test functions except ZDT2 (where there was no observed significant difference between the two methods over the 50 runs).

These results are further supported by those using the $\mathcal{V}$ measure (Table 9). Fronts found by the unconstrained full archive are significantly ahead of the clustering truncated method using the GA, for both equal generations and equal computation time. Using the ES scheme

[5] t-tests cannot be used as the samples cannot reasonably be assumed to be drawn from Normal distributions, neither are the samples independent (each pair being correlated). The Wilcoxon Signed rank test does not assume normality, and its independence assumption is only in relation to the paired values (that each pair be independent in relation to all other pairs).

the fronts found by the dominated tree method are significantly better than those found by clustering for all test functions except ZDT2, where there was no significant difference between the two methods.

7 STOPPING CRITERIA

Robust stopping criteria are largely missing from the MOEA literature. Beale and Cook [5] include a fitness-based stopping criterion in their study, in which the algorithm is terminated if the fittest individual has remained unchanged for 1000 consecutive generations. As Coello [10] points out, MOEAs since Schaffer [6], which carry a set of non-dominated individuals, are usually terminated after a fixed number of generations, or the population is monitored at intervals and a decision made on a visual basis.

The use of an elite archive which can only 'advance' however allows a number of robust stopping methodologies to be introduced. Examples of these are:

- When no individual that dominates a member of F is discovered after a given number of consecutive generations.

This is similar to the approach taken by Beale and Cook [5], however in a MOEA there is a set of elite solutions instead of a single elite individual. It is to be emphasised that this sort of criterion can fail with an truncated archive which is prone to oscillation. Note that new individuals may be found that are non-dominated by the frontal set - and therefore are inserted into F. However, these individuals may only fill in the front (increasing its resolution), rather than pushing it forward.

- When there has been no change in the extremal values for a given number of consecutive generations.

The previous stopping criterion, when taken by itself, may lead to sub-optimal stopping in that, although the front may have ceased moving forwards, it may still be moving outwards toward the extremes. This second criterion takes this form of search into account and can usefully be combined with the previous criterion.

- When the average distance in objective space between neighbouring individuals in F reaches a specific threshold.

The practitioner may wish the front to be defined to a particular resolution, therefore they may not solely wish to stop the search process after the front has finished moving, but prefer to wait until this resolution is reached. With two objectives this can be achieved by calculating the maximum over F of the nearest neighbour distances[6]. With more than two objectives a similar termination criterion can be defined based on the maximum area of any triangle of a Delauney tessellation [25] of F.

In practise the third criterion alone may lead to stopping before a good estimate of the true Pareto front has been found (if the resolution is set too coarse), however in conjunction

[6]This can fail in the pathological case that the true front contains isolated points in objective space.

with the first two criteria a good estimate of the true Pareto front to any desired resolution can be achieved.

If a practitioner prefers a small number of evenly distributed individuals to be returned after algorithm termination, the final elite archive may be clustered using the method employed in standard SPEA (the computational cost is not too high as it only needs to be performed once).

The stopping criteria defined above are not readily applied to methods which do not use an unconstrained active elite archive because, even if a dormant offline store is used, these methods are susceptible to oscillating active estimates of the Pareto front, which may cause spurious early termination.

8 DISCUSSION

New data structures and methods – PQRS and dominated trees – have been introduced to facilitate faster performance when maintaining an unconstrained archive of non-dominated solutions. Unconstrained archives themselves are necessary to prevent oscillations and retreat of the frontal set, problems which beset MOEAs in which the elite archive is truncated. The problem of oscillating fronts being shown in this study for the first time to exist empirically.

The dominated trees reduce the time taken using an unconstrained archive, in both simple ES and GA MOEAs, although when the archive is relatively small (below approximately 50 members) the cost of maintaining these structures out-weighs the decreased search time. For the test sets used in this study even when running the (typically) quicker clustering algorithm for the same time as a simple MOEA with an unconstrained archive and dominated trees, the unconstrained algorithm still produces superior results. The precise impact of dominated trees on the overall run time of an algorithm will, of course, depend upon the complexity of test function itself and the complexity of the fitness assignment/selection procedure used.

The exact complexity of the search of a dominated tree is dependant on the number of individual composite points to be checked linearly K; as such the data structure described is more affected by individuals that lie on the extremes of the estimated Pareto front. This is shown empirically in Table 5, with a larger archive size needed for the test functions introduced in this study than the ZDT functions before the data structures are seen to be significantly faster than the linear search. It still may be the case that, at the extreme, maintenance of all solutions may be infeasible (simply in memory requirements), as such the data structures introduced here could be used in tandem with an unconstrained ϵ-approximate Pareto set.

In [21] the oscillating fronts problem is addressed by the development of the ϵ-dominance concept and ϵ-Pareto sets. In this approach the objective values of a solution are calculated, but also the ϵ-Pareto objective values. If the objectives are to be minimised these are

$$y_i^\epsilon = f_i(\mathbf{x}) \cdot (1 - \epsilon)\,, \qquad i = 1, \ldots, D \tag{33}$$

On initialisation the objective values of the seed solutions are calculated, and the non-dominated individuals stored in terms of their ϵ-objectives (an initial ϵ-Pareto set). On the generation of new individuals for insertion into the ϵ-Pareto set, comparison is based on the new solution's objective y and the stored solutions y^ϵ. Only if the new solution is not ϵ-dominated by any element of the set will it be inserted into the archive, its ϵ-objectives are then stored and any ϵ-dominated individuals removed. Laumanns et al. show that a result of this is a bound on the size of the ϵ-Pareto set. However this is at a cost; the final estimated

front may lie a factor of $(1+\epsilon)$ behind the true Pareto front in each dimension. In addition, although the overt oscillation of the Pareto set (in its ϵ-Pareto form) is removed, oscillation (and decreased search efficiency) will still occur; this is due to the discovery by the search population of a solution which dominates a ϵ-Pareto set solution (or is non-dominated), but does not ϵ-dominate (or is non-ϵ-dominated by the archive), and is therefore discarded. The empirical impact of this oscillation and that of restricting forward movements of the ϵ-Pareto set to steps of greater than $\epsilon \cdot f_i$ has not been reported. If the ϵ-dominance approach is to be adopted, a potentially time consuming search of the ϵ-Pareto set is still necessary when inserting a new solution, so the data structures introduced here are just as applicable.

Current areas of research of interest to the authors include the parallel implementation of MOEAs using dominated trees and PQRS, the use of structural information in local regions of objective space to improve search efficiency and the use of dominated trees to facilitate multi-objective particle swarm optimisation.

The data structures and methods introduced in this study are currently being applied to the evolution of neural network forecasting models [34], information retrieval and the optimisation of safety critical systems.

ACKNOWLEDGEMENTS

Jonathan Fieldsend gratefully acknowledges support from Invensys Climate Controls Europe and an Exeter University Studentship. The authors would like to thank Kevin Smith for useful discussions during the course of this work. They would also like to thank the anonymous referees for their many useful comments. This work uses methods from the the SPEA code provided online by E. Zitzler at the Swiss Federal Institute of Technology, as the base of the clustering algorithm (`http://www.tik.ee.ethz.ch/~zitzler/testdata.html#source`).

Table 1: Mean execution time (in seconds) of the three archive methods, standard deviation in parentheses. Results in bold signify significantly faster results under the Wilcoxon nonparametric signed ranks test (2 tailed, 2.5% in each tail) compared to the other two algorithms. Results in italics signify that the dominated tree algorithm is significantly faster than linear search.

		Linear	Dominated tree	Clustered
ES	ZDT1	0.61 (0.06)	0.60 (0.06)	1.04 (0.08)
	ZDT2	0.44 (0.06)	0.45 (0.05)	0.48 (0.07)
	ZDT3	0.59 (0.05)	0.60 (0.06)	1.00 (0.07)
	F1	7.29 (0.18)	***5.62*** (0.21)	48.39 (0.50)
	F2	134.00 (9.56)	*111.80* (7.63)	**77.26** (0.69)
	F3	191.82 (19.21)	*151.25* (14.71)	**78.79** (1.36)
GA	ZDT1	3.11 (0.20)	*1.79* (0.12)	**1.03** (0.08)
	ZDT2	2.44 (0.18)	*1.53* (0.11)	**0.54** (0.07)
	ZDT3	2.36 (0.12)	*1.27* (0.08)	**1.07** (0.08)
	F1	10.38 (0.82)	***7.87*** (0.86)	29.65 (0.65)
	F2	105.07 (12.80)	*81.86* (9.61)	**62.94** (0.96)
	F3	341.54 (38.25)	*245.61* (27.604)	**74.64** (1.961)

Table 3: Generations (multiples of 1000) for which algorithm A is significantly faster than algorithm B. $t(A, B)$, calculated using the Wilcoxon nonparametric signed ranks test at the 5% level with 2.5% in each tail. Where L denotes the standard unconstrained archive with linear search, D is the dominated tree archive and C is the clustered archive.

		$t(L,D)$	$t(L,C)$	$t(D,L)$	$t(D,C)$	$t(C,L)$	$t(C,D)$
ES	ZDT1	1-48	1-100	-	1-100	-	-
	ZDT2	1-64	1-100	-	1-100	-	-
	ZDT3	1-64	1-100	-	1-100	-	-
	F1	1-5	1-100	10-100	1-100	-	-
	F2	-	1-53	2-100	1-61	57-100	66-100
	F3	-	1-44	3-100	1-50	48-100	54-100
GA	ZDT1	1-16	-	22-100	-	12-100	1-100
	ZDT2	2, 4-10	-	22-100	-	4-100	2-100
	ZDT3	1-15	15-19	22-100	20-57	30-100	1-11, 71-100
	F1	1-3	1-100	8-100	1-100	-	-
	F2	-	1-56	2-100	1-69	62-100	76-100
	F3	-	1-26	3-100	1-31	29-100	35-100

Table 5: Table showing archive size beyond which the dominated tree is significantly faster than the linear search for the various test problems for the total (absolute) time cost of the method up to that archive size, the second column pair are for the incremental cost (the difference between the time taken in reaching one range and the next). Significance calculated using a Wilcoxon signed ranks test (2.5% in each tail).

	Trees Sig. Faster			
	Cumulative		Incremental	
	ES	GA	ES	GA
ZDT1	150	150	100	100
ZDT2	150	150	50	50
ZDT3	250	150	100	50
F1	450	300	250	150
F2	550	400	200	150
F3	400	350	200	150

Table 7: Comparison between end-of-run fronts from the unconstrained and clustered elite ES and GA archive models, using the $\tilde{\mathcal{C}}$ measure. $\tilde{\mathcal{C}}(U, C)$ is the mean proportion of the members of the clustered generated front dominated by members of the unconstrained generated front. Means are over 50 runs, with standard deviation in parentheses. The first two columns relate to the results after an equal number of generations. The third and fourth columns relate to the results after the clustered algorithms have run for an equal time as the dominated tree based unconstrained algorithm. Results in bold signify significantly better results under the Wilcoxon nonparametric signed ranks test (2 tailed, 2.5% in each tail).

		Equal generations		Equal time	
		$\tilde{\mathcal{C}}(U, C)$	$\tilde{\mathcal{C}}(C, U)$	$\tilde{\mathcal{C}}(U, C)$	$\tilde{\mathcal{C}}(C, U)$
ES	ZDT1	0.318 (0.142)	0.408 (0.172)	**0.636** (0.173)	0.163 (0.161)
	ZDT2	0.337 (0.156)	**0.465** (0.180)	0.383 (0.175)	0.413 (0.179)
	ZDT3	0.244 (0.114)	**0.382** (0.135)	**0.417** (0.152)	0.245 (0.152)
	F1	**0.899** (0.030)	0.005 (0.001)	**0.999** (0.003)	0.000 (0.001)
	F2	**0.894** (0.035)	0.001 (0.001)	**0.878** (0.039)	0.002 (0.001)
	F3	**0.816** (0.048)	0.003 (0.004)	**0.787** (0.055)	0.005 (0006)
GA	ZDT1	**0.981** (0.012)	0.000 (0.000)	**0.982** (0.011)	0.000 (0.000)
	ZDT2	**0.982** (0.010)	0.000 (0.000)	**0.981** (0.011)	0.000 (0.000)
	ZDT3	**0.984**(0.011)	0.000 (0.000)	**0.984** (0.011)	0.000 (0.000)
	F1	**0.995** (0.010)	0.001 (0.002)	**0.996** (0.010)	0.000 (0.001)
	F2	**0.947** (0.025)	0.000 (0.001)	**0.935** (0.030)	0.000 (0.001)
	F3	**0.962** (0.029)	0.000 (0.001)	**0.921** (0.032)	0.000 (0.001)

Table 9: Comparison between end-of-run fronts from the unconstrained and clustered elite ES and GA archive models, using the $\mathcal{V}$ measure. Where $\mathcal{V}(U,C)$ is the mean proportion of the volume of the minimum hypercube containing both estimated fronts, which is dominated by members of the unconstrained generated front but not by members of the constrained generated front. Means are over 50 runs, standard deviation in parentheses, value as a percentage. The first two columns relate to the results after an equal number of generations. The third and fourth columns relate to the results after the clustered algorithms have run for and equal amount of time as the data structure based unconstrained algorithm. Results highlighted in bold signify significantly better results under the Wilcoxon nonparametric signed ranks test (2 tailed, 2.5% in each tail).

		Equal generations		Equal time	
		$\mathcal{V}(U,C)$	$\mathcal{V}(C,U)$	$\mathcal{V}(U,C)$	$\mathcal{V}(C,U)$
ES	ZDT1	**0.760**% (0.360)	1.260% (4.810)	**1.650**% (0.530)	0.810% (4.550)
	ZDT2	0.802% (0.447)	1.021% (0.618)	0.957% (0.512)	0.836% (0.549)
	ZDT3	**0.581**% (0.320)	1.151% (5.530)	**0.912**% (0.466)	0.944% (5.374)
	F1	**9.287**% (0.708)	0.014% (0.031)	**4.601**% (0.288)	0.000% (0.000)
	F2	**9.466**% (0.669)	0.019% (0.011)	**5.614**% (0.413)	0.025% (0.018)
	F3	**11.215**% (0.748)	0.224% (0.115)	**8.517**% (0.765)	0.300% (0.161)
GA	ZDT1	**5.829**% (0.374)	0.000 (0.000)	**5.423**% (0.413)	0.000 (0.000)
	ZDT2	**8.882**% (0.762)	0.000 (0.000)	**7.617**% (0.666)	0.000 (0.000)
	ZDT3	**3.612**% (0.359)	0.000 (0.000)	**3.713**% (0.403)	0.000 (0.000)
	F1	**5.146**% (0.439)	0.000% (0.000)	**5.553**% (0.400)	0.000 (0.000)
	F2	**8.651**% (0.492)	0.004% (0.013)	**8.649**% (0.563)	0.006% (0.009)
	F3	**13.770**% (0.962)	0.001% (0.002)	**12.514**% (0.978)	0.046% (0.095)

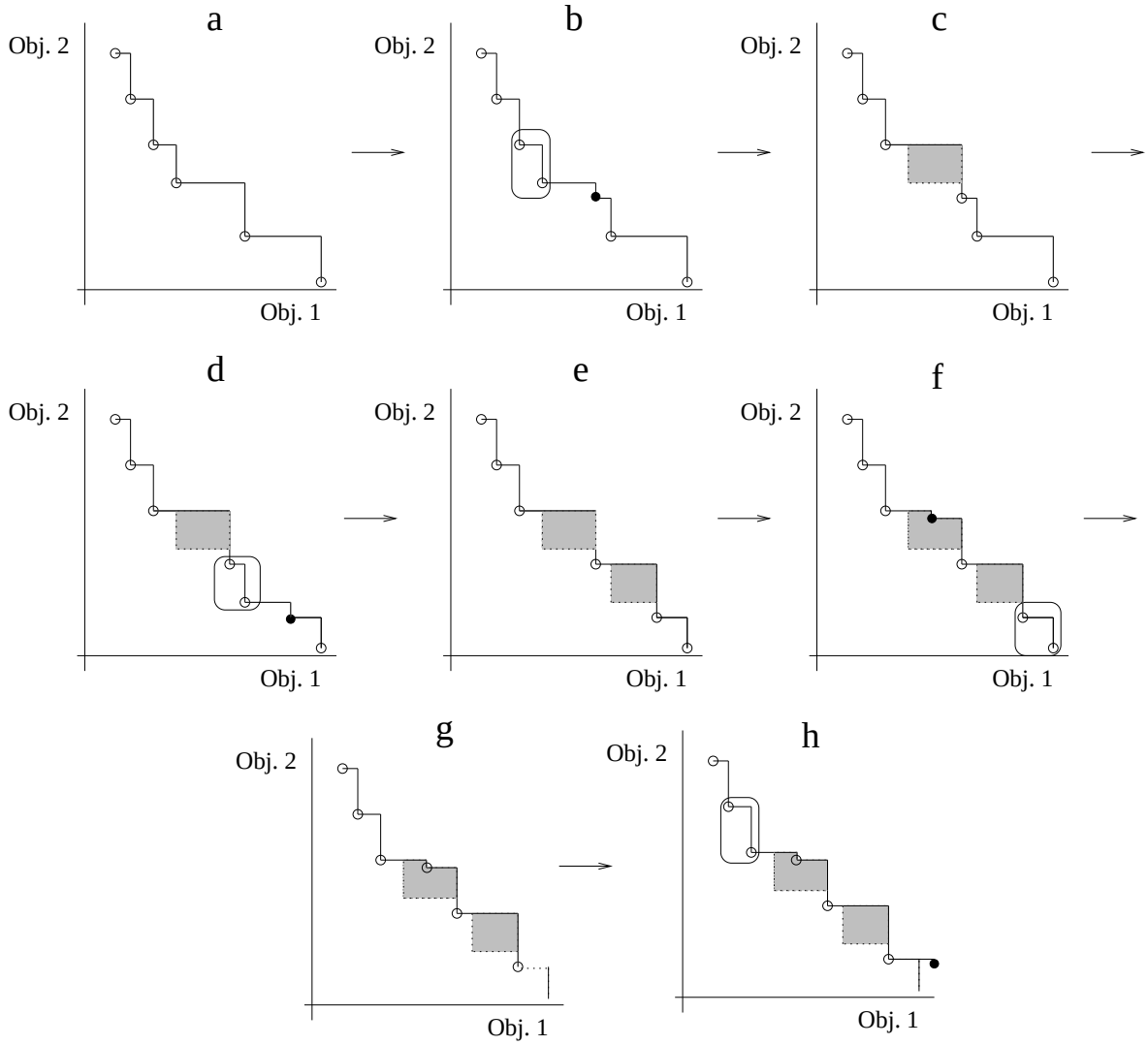


Figure 1: Example of a retreating estimated Pareto front, produced by truncation by clustering.

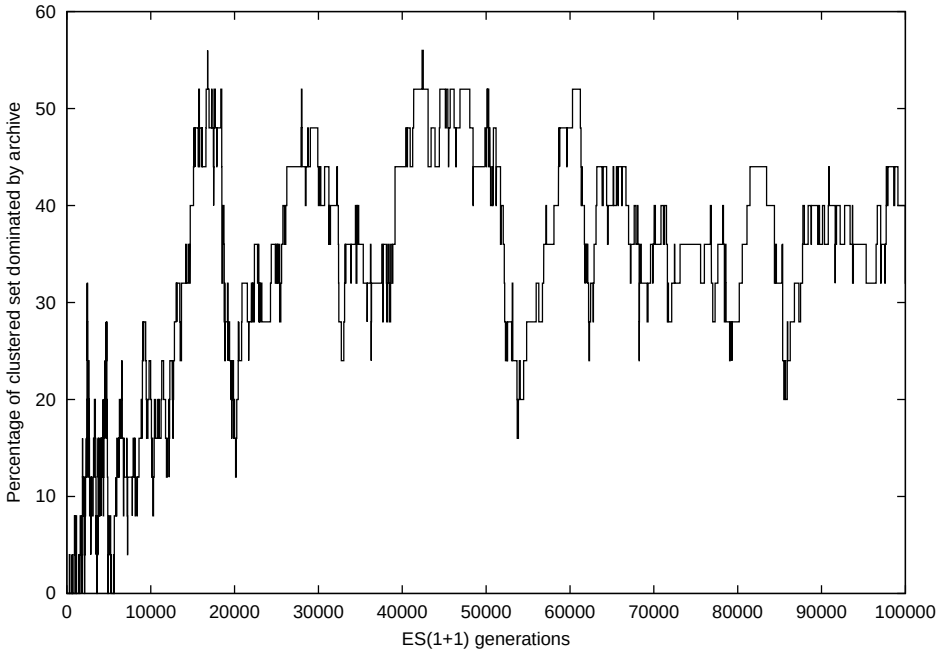


Figure 2: Percentage of individuals in active clustered archive dominated by members of the dormant unconstrained archive.

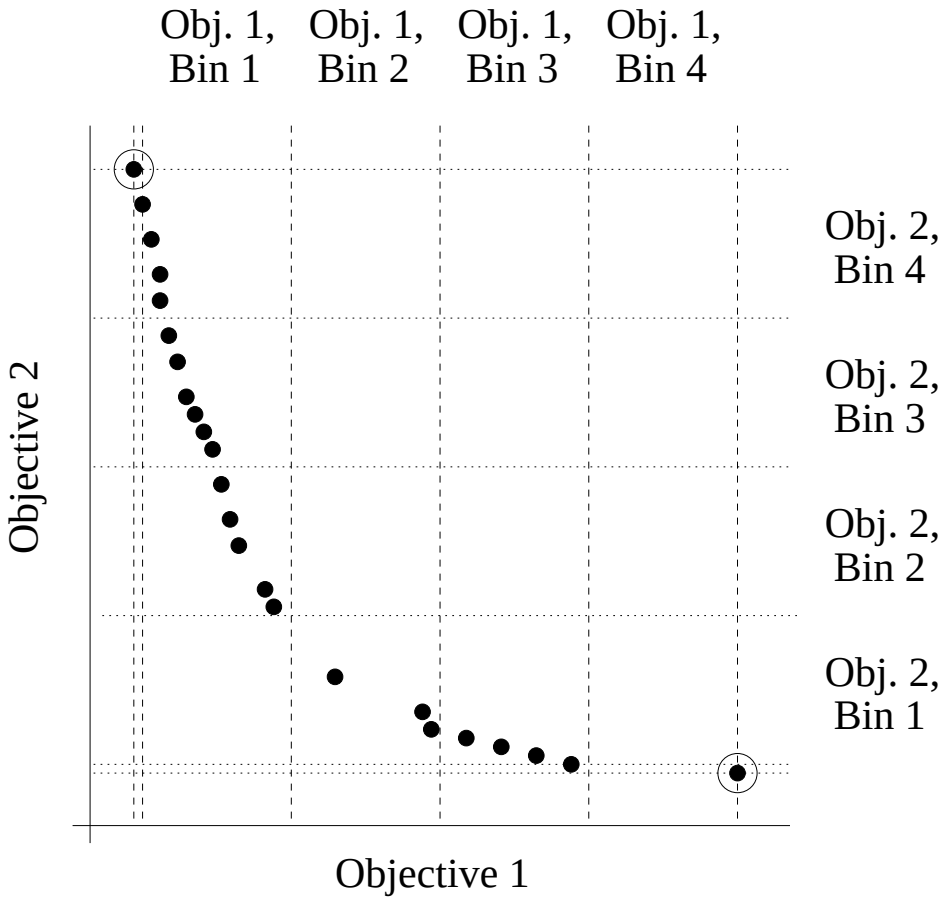
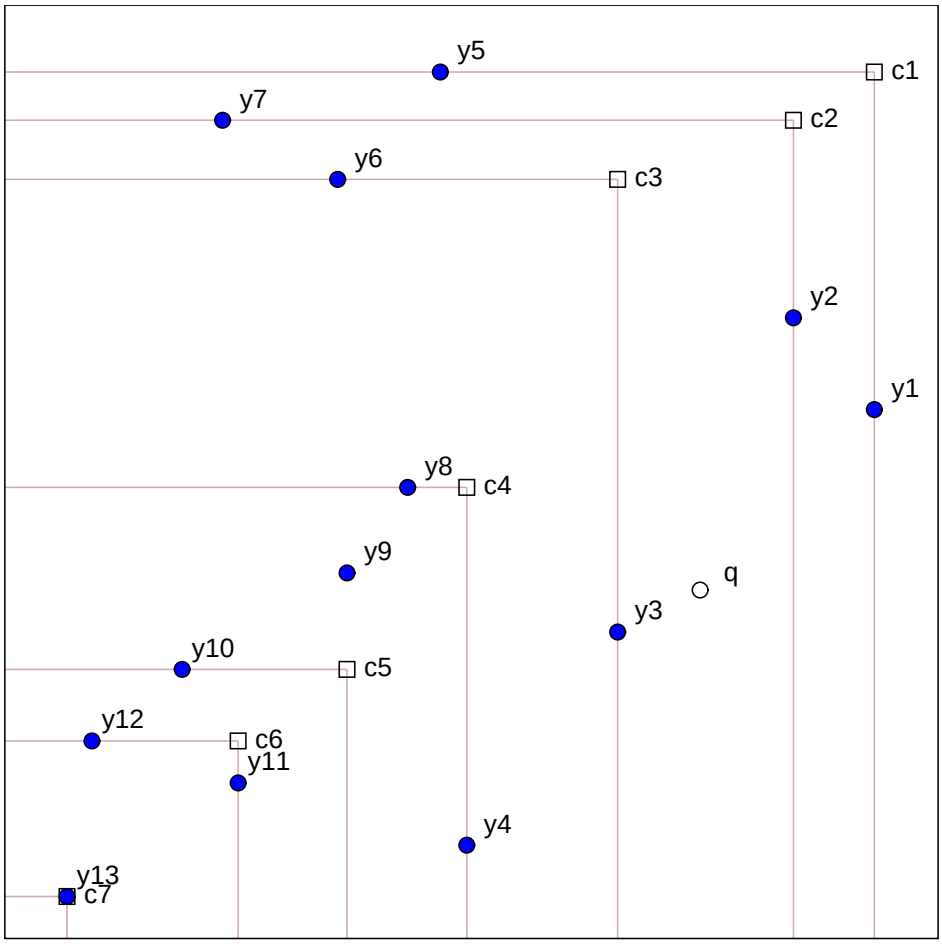


Figure 3: Two objective example of Partitioned Quasi-Random Selection, with the Objective 1 and 2 dimensions partitioned in illustration (during selection only one dimension is partitioned at each generation). $N = 5$ representative individuals are required, so selection, from $N-1 = 4$ bins, after pinning of relevant extremal value (circled).



Composite node	Constituent points
$\mathbf{c}_1$	$\langle \mathbf{y}_1, \mathbf{y}_5 \rangle$
$\mathbf{c}_2$	$\langle \mathbf{y}_2, \mathbf{y}_7 \rangle$
$\mathbf{c}_3$	$\langle \mathbf{y}_3, \mathbf{y}_6 \rangle$
$\mathbf{c}_4$	$\langle \mathbf{y}_4, \mathbf{y}_8 \rangle$
$\mathbf{c}_5$	$\langle \mathbf{y}_9, \mathbf{y}_{10} \rangle$
$\mathbf{c}_6$	$\langle \mathbf{y}_{11}, \mathbf{y}_{12} \rangle$
$\mathbf{c}_7$	$\langle \mathbf{y}_{13}, \mathbf{y}_{13} \rangle$

Figure 4: **Dominated tree.** Top: 13 points $\mathbf{y}_m$ in two dimensions and the composite points $\mathbf{c}_i$ (squares) forming a dominated tree. The open circle, $\mathbf{q}$ marks a query point. Bottom: Composite nodes listed as ordered by $\prec$.

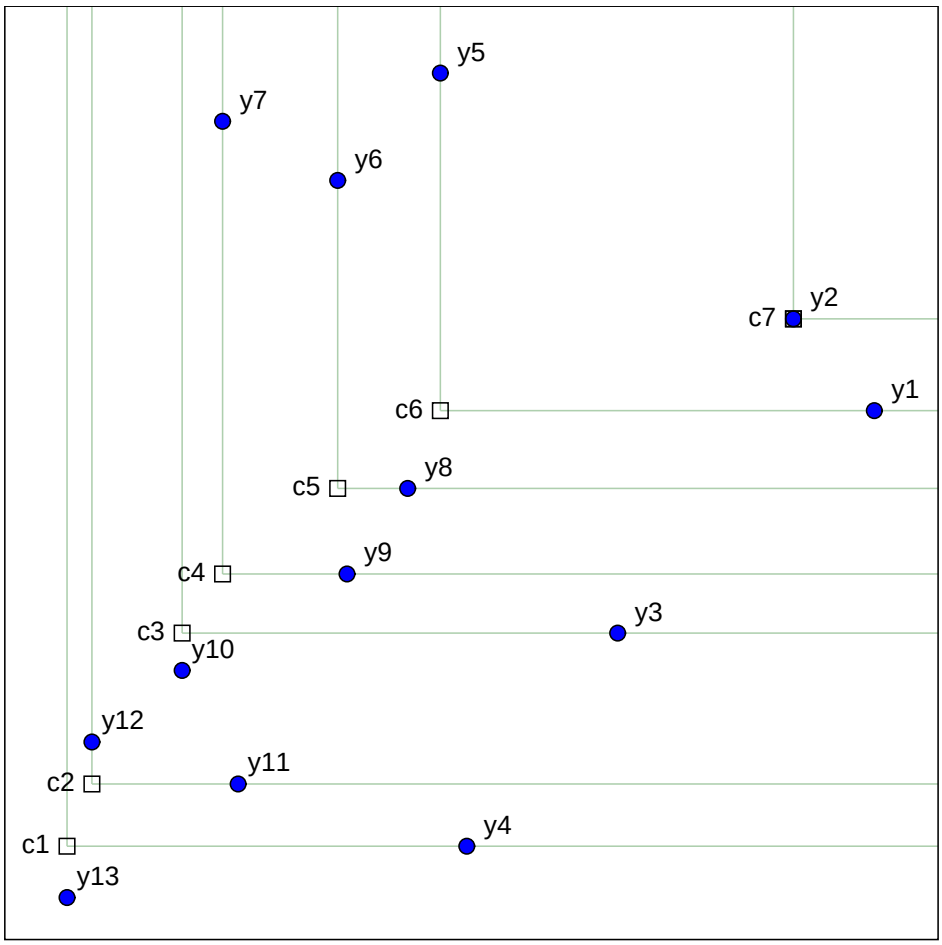


Figure 5: **Non-dominated tree.** The non-dominated tree representing the 13 points $\mathbf{y}_m$ illustrated in figure 4.

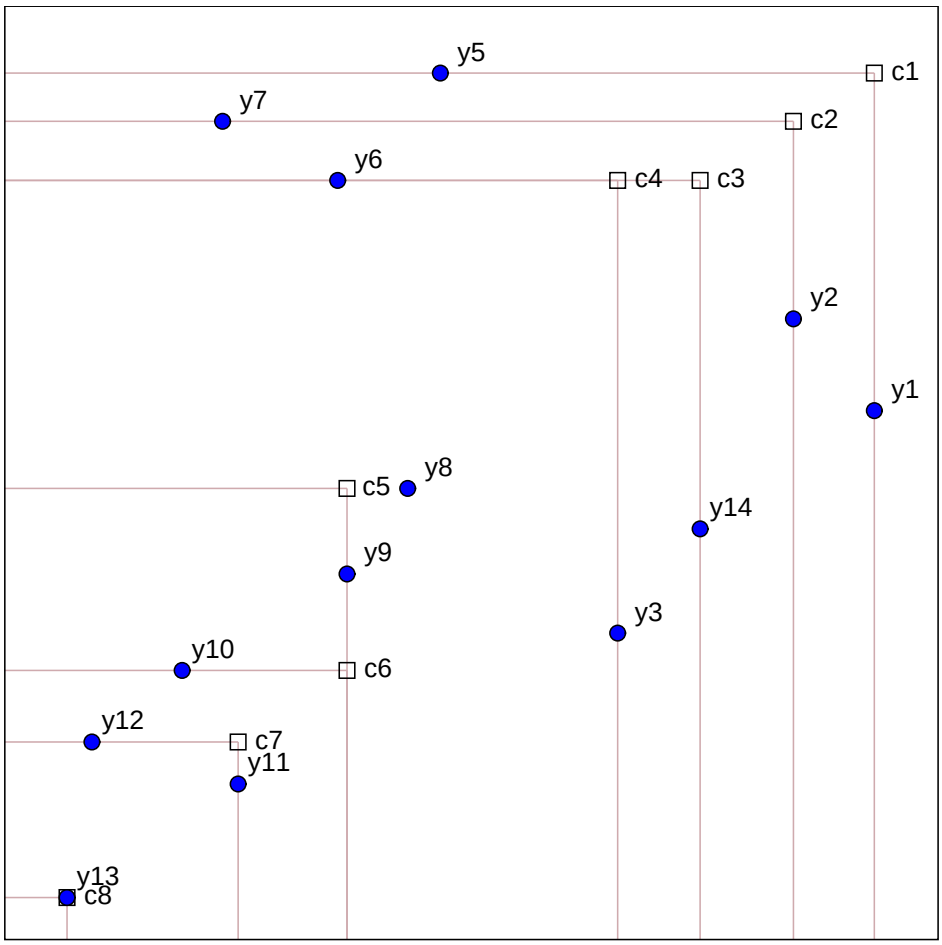


Figure 6: **Insertion and deletion into a dominated tree.** Dominated tree resulting from the tree shown in figure 4 after the insertion of $\mathbf{y}_{14}$ and deletion of $\mathbf{y}_4$.

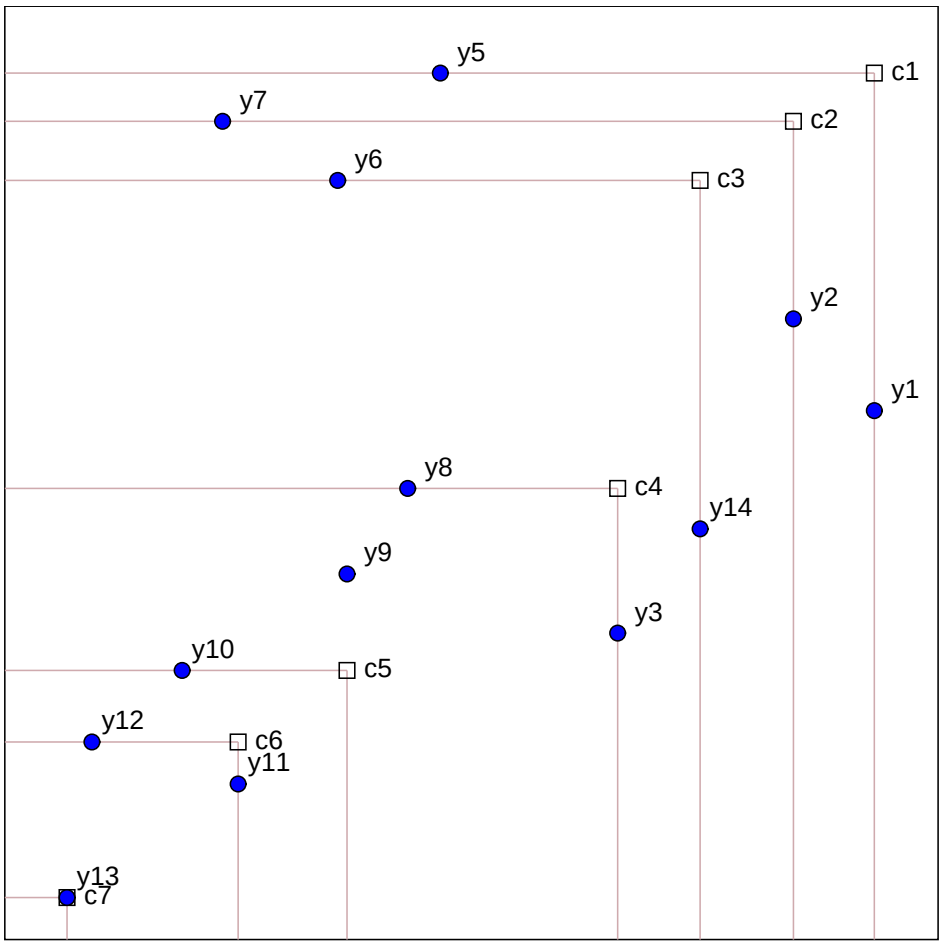


Figure 7: **Cleaning of a dominated tree.** Dominated tree resulting from 'cleaning' of the tree shown in figure 6.

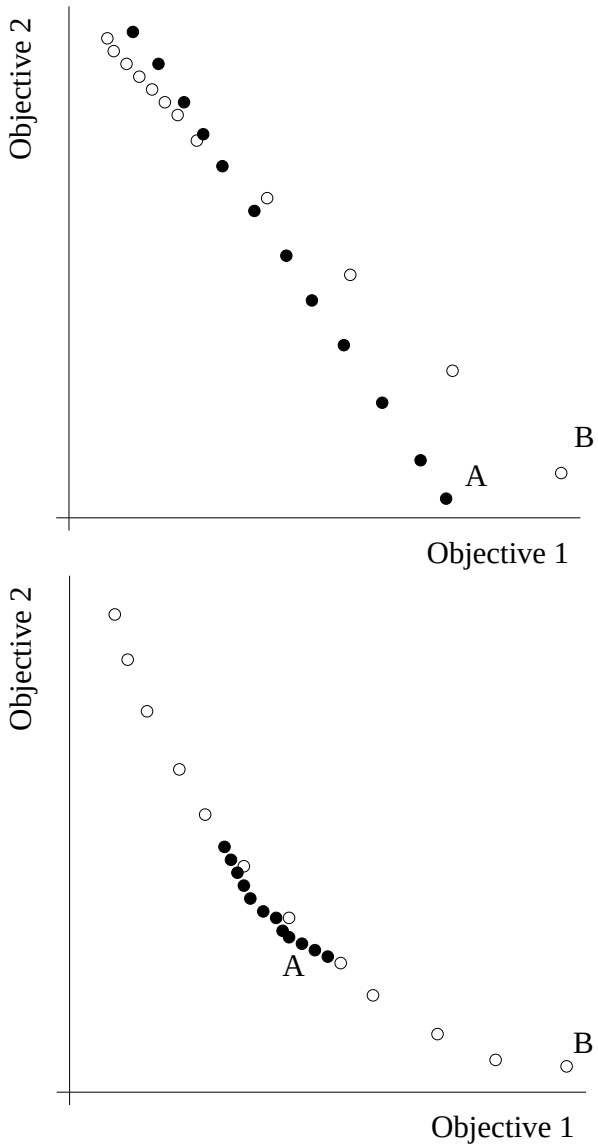


Figure 8: (a) Illustrates dense and evenly distributed estimated Pareto fronts. Front A dominates a larger extent of Front B, but both fronts dominate an equal number of each others' constituent members. (b) Illustrates estimated Pareto fronts of differing extents. Front B is of far greater extent than A, but will receive a lower $\mathcal{C}$ metric value.

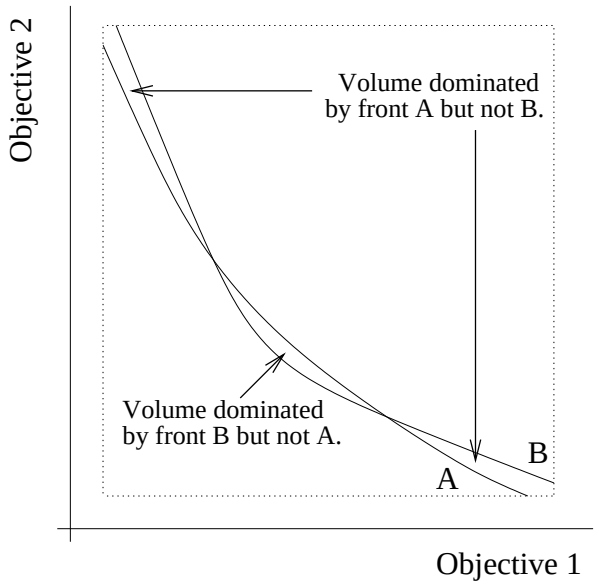


Figure 9: Two dimensional illustration of minimum surrounding hypercube volume dominated by two fronts (hypercube boundaries marked with dashed lines).

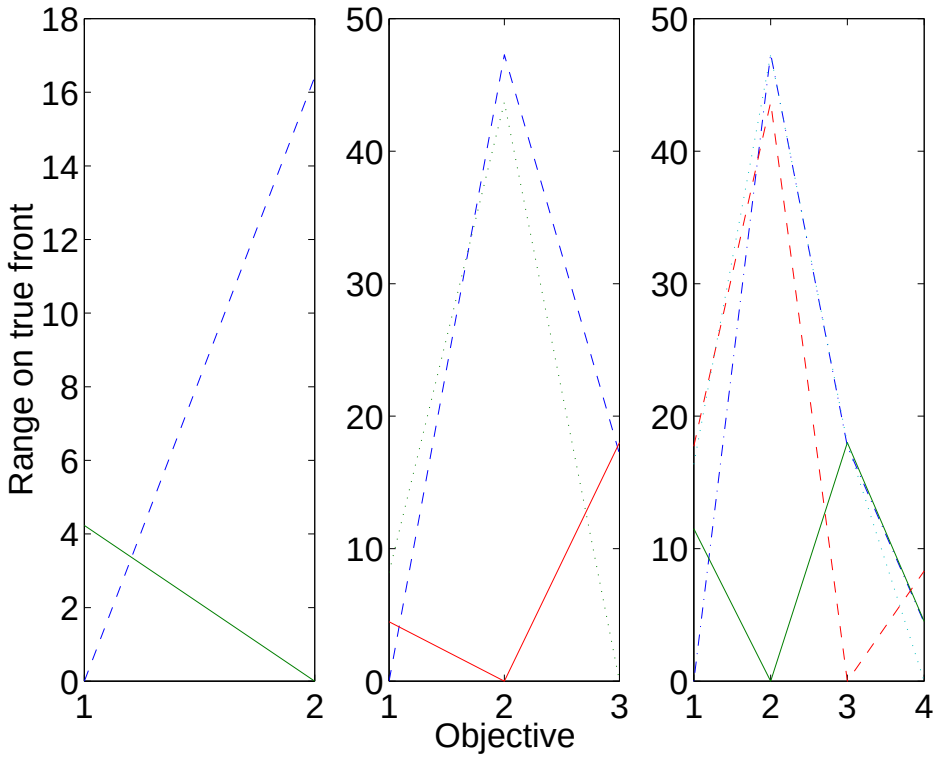


Figure 10: Parallel coordinated graphs of the objective bounds of the test functions (each line relating to an individual which minimises one of the objectives)

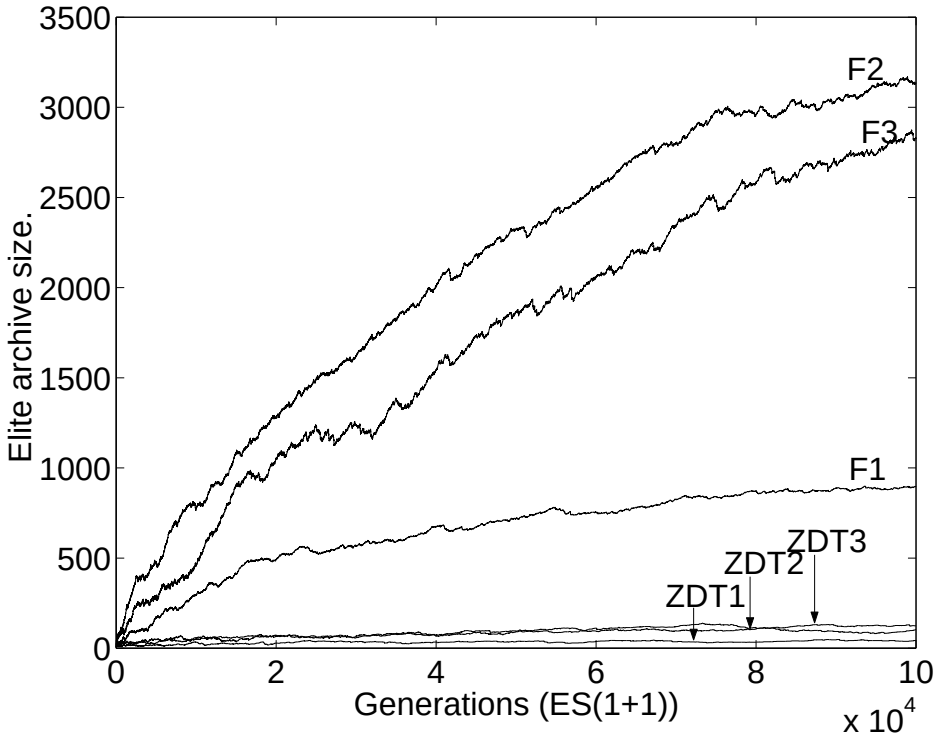


Figure 11: Growth of archive sizes using ZDT1, ZDT2 and ZDT3 [9] and the test functions F1, F2 and F3 introduced in this study, using a simple ES(1+1) based MOEA.

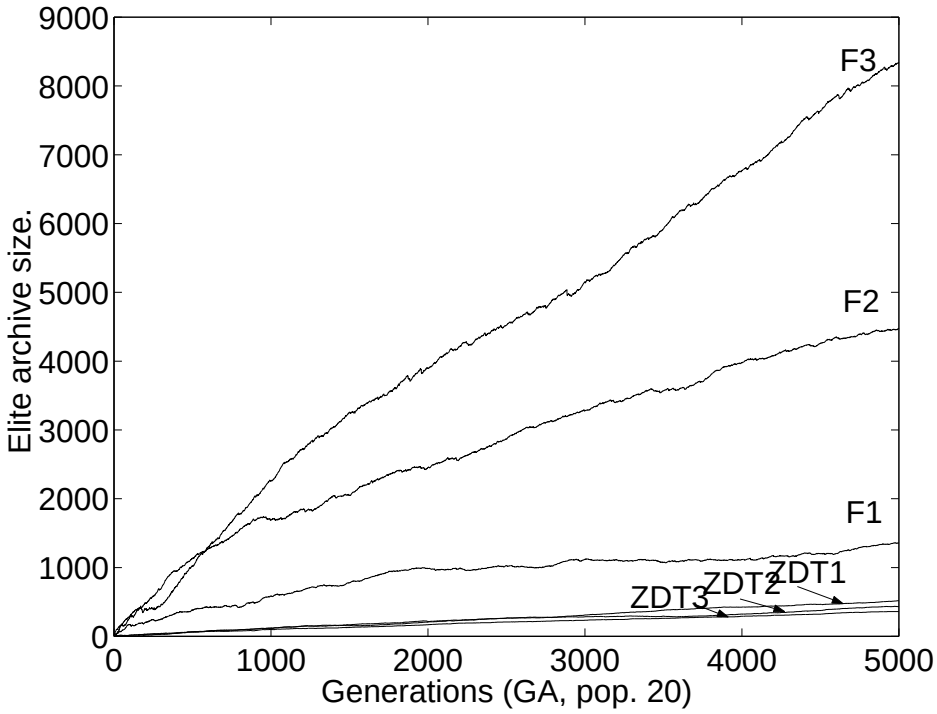


Figure 12: Growth of archive sizes using ZDT1, ZDT2 and ZDT3 [9] and the test functions F1, F2 and F3 introduced in this study, using a simple GA based MOEA.

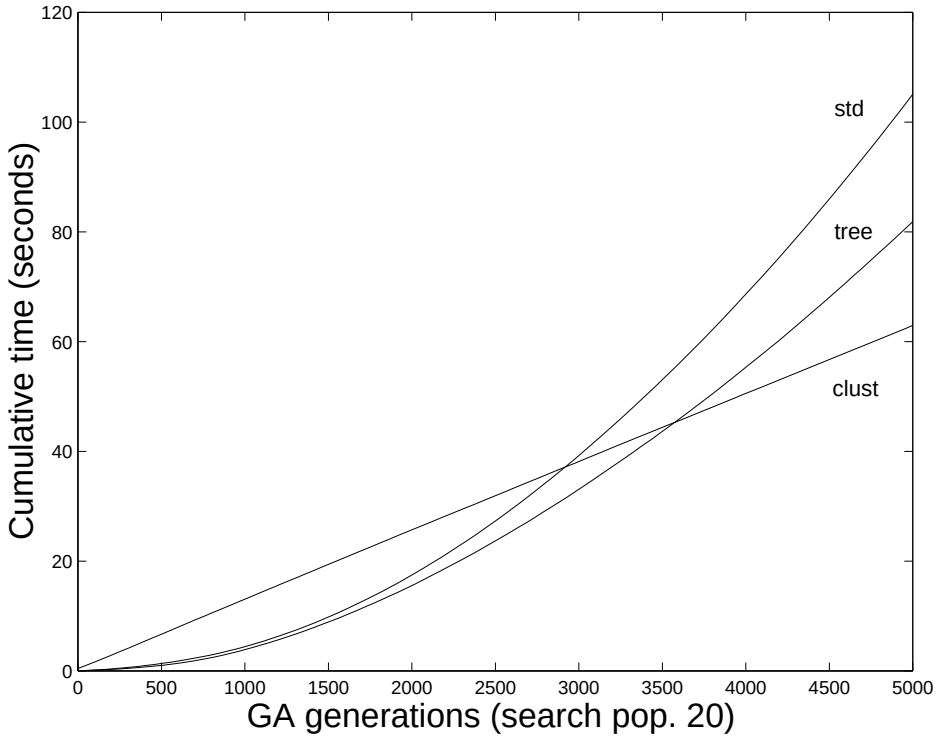


Figure 13: The average time time (in seconds) spent checking an individual against the elite archive and updating the archive in the three different archive methods (unconstrained standard, unconstrained with the new data structures and clustered), using the GA up to 5000 generations (with the three objective problem F_2).

References

[1] C.M. Fonseca and P.J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. Morgan Kaufmann.

[2] P. Hajela and C-Y. Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4:99–107, 1992.

[3] G. T. Parks and I. Miller. Selective Breeding in a Multiobjective Genetic Algorithm. *Lecture Notes in Computer Science*, 1498:250–259, 1998.

[4] V. Pareto. *Manuel D'Économie Politique*. Marcel Giard, Paris, 2nd edition, 1927.

[5] G.O. Beale and G. Cook. Optimal Digital Simulation of Aircraft via Random Search Techniques. *AIAA Journal of Guidance and Control*, 1(4):237–241, 1978.

[6] J.D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the First International Conference on Genetic Algorithms*, pages 99–100, 1985.

[7] C.M. Fonseca and P.J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[8] D. Van Veldhuizen and G. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 8(2):125–147, 2000.

[9] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

[10] C.A.C Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, 1999.

[11] J. Horn, N. Nafpliotis, and D.E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, 1994. IEEE Service Center.

[12] N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, 1995.

[13] K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[14] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology Zurich (ETH), 1999. Diss ETH No. 13398.

[15] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[16] J. Knowles and D. Corne. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *1999 Congress on Evolutionary Computation*, pages 98–105, Piscataway, NJ, 1999. IEEE Service Center.

[17] J. D. Knowles and D. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[18] M. Laumanns, E. Zitzler, and L. Thiele. A Unified Model for Multi-Objective Evolutionary Algorithms with Elitism. In *Proc. of the 2000 Congress on Evolutionary Computation*, pages 46–53, Piscataway, NJ, 2000. IEEE Service Center.

[19] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report TIK-Report 103, Swiss Federal Institute of Technology Zurich (ETH), May 2001.

[20] T. Hanne. On the convergence of multiobjective evolutionary algorithms. *European Journal of Operational Research*, (117):553–564, 1999.

[21] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. On the Convergence and Diversity-Preservation Properties of Multi-Objective Evolutionary Algorithms. Technical Report TIK-Report 108, Swiss Federal Institute of Technology Zurich (ETH), June 2001.

[22] R.M. Everson, J.E. Fieldsend, and S.Singh. Full Elite Sets for Multi-Objective Optimisation. In I.C. Parmee, editor, *Adaptive Computing in Design and Manufacture V*, pages 343–354. Springer, 2002.

[23] T. Hanne. Global Multiobjective Optimization Using Evolutionary Algorithms. *Journal of Heuristics*, 6:347–360, 2000.

[24] T. Murata and H. Ishibuchi. MOGA: Multi-objective genetic algorithms. In *Proceedings of the 1995 IEEE International Conference on Evolutionary Computation*, pages 289–294, Perth, November, 1995. IEEE Press.

[25] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational geometry : algorithms and applications*. Springer, Berlin, 1997.

[26] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, pages 509–517, 1975.

[27] J.L. Bentley and J.H. Friedman. Data structures for range searching. *Computing Surveys*, 111(4):398–409, 1979.

[28] E.M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14:257–276, 1985.

[29] M. Sun and R.E. Steuer. InterQuad: An ineteractive quad tree based procedure for solving the discrete alternative multiple criteria problem. *European Journal of Operational Research*, (89):462–476, 1996.

[30] S. Mostaghim, J. Teich, and A. Tyagi. Comparison of Data Structures for Storing Pareto-sets in MOEAs. In *Proceedings of the IEEE Conference on Evolutionary Computation, part of the IEEE World Congress on Computational Intelligence*, Hawaii, May 12-17, 2002. IEEE Press.

[31] E.M. McCreight. Efficient algorithms for enumerating intersecting intervals and rectangles. Technical Report CSL-80-9, Xerox Palo Alto Research Center, Palo Alto, CA, USA, 1980.

[32] F. Jones. *Lebesgue Integration on Euclidean Space.* Jones and Bartlett, Boston, 1993.

[33] F. Wilcoxon and R.A. Wilcox. *Some Rapid Approximate Statistical Procedures.* Lederle Labs, New York, 1964.

[34] J.E. Fieldsend and S. Singh. Pareto Multi-Objective Non-Linear Regression Modelling to Aid CAPM Analogous Forecasting. In *Proceedings of the IEEE International Conference on Computational Intelligence for Financial Engineering, part of the IEEE World Congress on Computational Intelligence*, Hawaii, May 12-17, 2002. IEEE Press.