Multi-Objective Binary Search Optimisation

Evan J. Hughes

Department of Aerospace, Power, and Sensors,
Cranfield University, Royal Military College of Science,
Shrivenham, Swindon, SN6 8LA, UK,
ejhughes@iee.org,
http://www.rmcs.cranfield.ac.uk/radar

Abstract. In complex engineering problems, often the objective functions can be very slow to evaluate. This paper introduces a new algorithm that aims to provide controllable exploration and exploitation of the decision space with a very limited number of function evaluations. The paper compares the performance of the algorithm to a typical evolutionary approach.

1 Introduction

Multi-Objective Evolutionary Algorithms [1] are becoming a well established technique for solving hard engineering problems. Like all optimisation algorithms they were first developed when processing resources and memory were scarce. As the use of optimisation algorithms migrates deeper into industry, and with more processing power available, the scale and characteristics of the problems being solved are changing. The objective functions are becoming more complex and consequently can take a long time to evaluate.

Problems such as aerodynamic optimisation and electromagnetic simulation often rely on finite element methods in order to simulate the systems of interest. These simulations can take from seconds to hours to run. The better the resolution and fidelity required, the longer the simulation time.

Many of the problems are highly multi-modal and so gradient based optimisers do not perform well. Unfortunately, with long simulation times, either small population sizes must be used within an evolutionary algorithm, or the algorithm must be run for a reduced number of generations in order to keep the total processing time within reasonable bounds. In both gradient based searches and evolutionary algorithms, every iteration or generation, the results of previous objective calculations are discarded to reduce processing and storage costs.

For example, a simple aerodynamic simulation takes 10 variables, produces two objectives and one constraint, and requires 1 hour per evaluation. An evolutionary algorithm with a population of 20 for 100 generations would take over 83 days to complete. With some multi-objective algorithms (e.g. NSGA and MOGA) only 20 points on the Pareto surface could be generated per generation. Algorithms such as SPEA with an external store for the Pareto set could retain more points. A Pareto surface would need to be generated from every point that was evaluated in the run of the evolutionary algorithm to make sure that no points had been lost. To confine the problem to a more

realistic time scale, only about 500 points could be generated, taking nearly 21 days to complete. Generating only 500 evaluations is equivalent to a population of 20 for 25 generations only, not much for many evolutionary multi-objective optimisation algorithms.

This paper proposes a new algorithm that is designed specifically to provide controllable exploration and exploitation, but with few objective calculations. The new algorithm uses many of the techniques developed for multi-objective evolutionary algorithms to guide the search process. The algorithm is not generational however and utilises all the objective calculations made when deciding where to place the next point in the hypercube that defines the search space.

This paper describes two approaches to implementing the idealised algorithm. The first uses Voronoi decomposition to locate the exact centre of the largest empty hypersphere in each case, but is computationally expensive for even moderate numbers of variables. The second algorithm is the prime focus of the paper and uses binary space subdivision to approximate the unexplored regions, producing a faster and more scalable algorithm.

Section 2 describes the ideal search algorithm, section 3 discusses multi-objective optimisation, section 4 details one approach to implementing the ideal algorithm using Voronoi diagrams and section 5 details an alternative implementation using a binary search. Section 6 describes the two multi-objective test functions used, section 7 presents results of the optimisation trials and a comparison with a typical evolutionary approach, and section 8 concludes.

2 The Ideal Algorithm

2.1 Introduction

The idealised algorithm is:

1. **Exploration:** Next point is the centre of the largest empty convex region.
2. **Exploitation:** Next point is the centre of the largest empty convex region that has a selected *good* point at one edge.

The aim of the idealised algorithm is to reduce the size of unexplored regions, resulting in uniform search coverage, while still being able to focus on the areas forming the Pareto surface. With only a limited number of function evaluations available, every evaluation must count.

2.2 Exploration

The exploration search step of the algorithm identifies the most unexplored region of the search hypercube, and places the next point at the centre of the region. The region could be described in a number of ways, the ideal being to find the largest convex region that will reside between the existing evaluation points. Sections 4 & 5 describe two alternative methods for approximating the most unexplored region.

2.3 Exploitation

The exploitation step involves first identifying a good point. In both algorithm implementations presented in this paper, tournament selection is used to identify a *good* point for a localised search to begin from. Once a point has been selected, the largest unexplored volume that contains the point at its edge is identified, and a new evaluation generated for the point corresponding to the centre of the volume.

2.4 Exploration versus Exploitation

The two phases of the algorithm, exploration and exploitation, must be controlled in order to provide effective coverage of the decision space. The algorithm must begin with an exploration phase to allow interesting regions to be identified, then the exploitation phase can be applied to refine the regions. As noted in most evolutionary algorithms, it is wise to always have a low level of exploration, even in the exploitation phase.

In evolutionary algorithms, the initial population provides pure exploration. The selective pressure and crossover in subsequent generations provide exploitation, with a low level mutation providing exploration of the decision space throughout the remaining optimisation process.

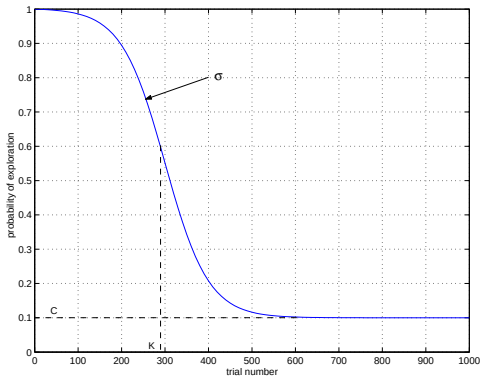

Fig. 1. Probability of performing exploration for C=0.1, K=0.3, $\sigma = 0.1$ and 1000 trials.

In both the algorithms presented, at each iteration the decision of whether to perform exploration or exploitation is made based on a probability distribution, $P(n)$, that varies with the number of function evaluations, n. The distribution is detailed in (1) and illustrated graphically in Fig.1. In (1), C is the minimum probability of performing an exploration step, σ is the rate at which the probability of exploration decays (smaller σ gives faster decay), K is the mid point of the decay (midpoint of the range [C,1]) and n_x is the maximum number of trials that are to be performed. Note, in Fig.1, C is not zero and so the point marked for K does not occur at 300 as would be the case if $C = 0$.

$$P(n) = (C-1)\frac{\tanh\left(\frac{n/n_x - K}{\sigma}\right) - \tanh\left(\frac{-k}{\sigma}\right)}{\tanh\left(\frac{1-k}{\sigma}\right) - \tanh\left(\frac{-k}{\sigma}\right)} + 1 \qquad (1)$$

3 Multi-Objective Optimisation

Much research has been performed on discriminating between members of a Pareto set to allow the entire Pareto set to be approximated in a single run of an evolutionary algorithm. In an evolutionary algorithm, the Pareto set may either be maintained within the base population (NSGA, MOGA, etc.) or externally (SPEA, PAES, etc.)[1]. For the methods that use the population to store the set, a large population size is required to provide sufficient sampling of the Pareto surface.

A Pareto ranking method is required to allow 'good' solutions to be chosen. In the algorithms described in this paper, with the small number of evaluations used, **all** the points generated in the run of the optimiser will be used to create the final Pareto surface. Methods that do not maintain an external store of the Pareto set will be used in the examples as at each iteration of the algorithm, all the points that satisfy the constraints created so far will be accounted for. If larger numbers of evaluations are to be performed, only the solutions in the current tournament set need to be ranked. This approach will lead to reduced performance, but the processing overhead of the algorithms will scale better with increasing function evaluations.

4 Voronoi Optimisation Algorithm

4.1 Largest Empty Convex Region

The idealised algorithm in section 2 relies on being able to identify the largest empty convex region either in the entire search space, or with a chosen point at its edge. Figure 2 shows a 2D Euclidean plane representation of the largest empty convex region between a set of points in the decision space. The region may be approximated by finding the largest empty hypersphere that can be placed between the existing points. The new point would then be generated at the centre of the hypersphere. Finding the centre of the largest empty hypersphere is still not a trivial problem to solve.

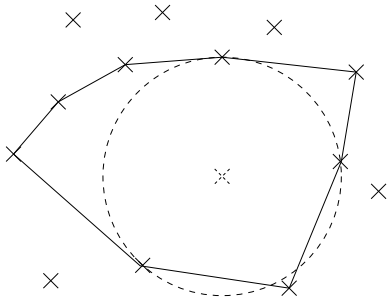

Fig. 2. Largest empty hypersphere

4.2 Voronoi Diagrams

The *Voronoi diagram* [2, 3] can be used to identify the centre of the largest empty hypersphere. A typical Voronoi Diagram is shown in Fig. 3 with the largest empty circle indicated. The centre of the largest empty circle will always coincide with either a Voronoi vertex, or a vertex generated by the intersection of the Voronoi diagram with the convex hull of the set of points.

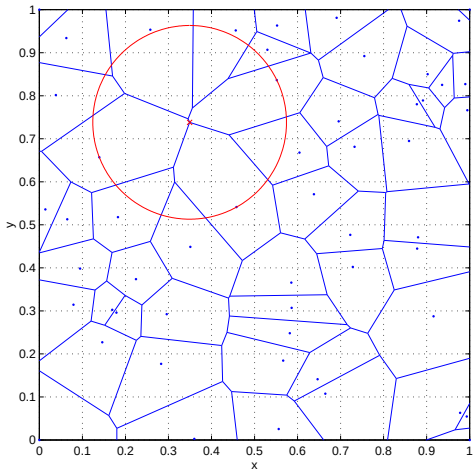

Fig. 3. Example Voronoi diagram showing how centre of largest empty hypersphere lies at a Voronoi vertex

The Voronoi diagram divides a hyperspace containing points into regions, each region surrounding a single point. The space is divided so each point is associated with the region of space closest to it. If $P = p_1, p_2, \ldots, p_n$ is a set of points (or *sites*) in the 2D Euclidean plane, the plane is partitioned by assigning every point in the plane to its nearest site. All those points assigned to p_i form the Voronoi region $V(p_i)$.

$$V(p_i) = \{x : |p_i - x| \leq |p_j - x|, \forall j \neq i\} \tag{2}$$

The Voronoi diagram is formed as the boundaries of the set of Voronoi regions. The Voronoi edges are points on the plane that lie on the boundaries of the Voronoi regions and will be by definition equidistant from two sites. A Voronoi vertex is formed at the junction of multiple Voronoi edges. The generation of Voronoi diagrams is computationally expensive and so direct use is only really possible for problems with low-dimensionality. Indirect calculation of the Voronoi diagram is still slow but can lead to useful optimisation systems [4].

In all the optimisation algorithms used in this paper, the full ranges of the parameter optimisation in each dimension are mapped into a hypercube with each side having the limits of [0,1]. This mapping scales each of the parameters with respect to their minimum and maximum values, allowing unbiased Euclidean distances to be calculated.

To simplify the processing for finding the largest empty hypersphere, a point is placed at each corner of the hypercube in the decision space, simplifying the calculation of the intersection of the Voronoi diagram with the convex hull of the points. The next point is then placed uniformly at random within the hypercube, allowing the Voronoi diagram to be generated and the optimisation process to begin. With a 10 dimensional problem, the hypercube has 1024 corners, therefore 1025 points would be required in the initial sampling of the decision space. In practice, for many engineering problems that are to be optimised on a single processor, the direct Voronoi approach is limited to problems with less than 10 dimensions due to a rapid expansion of computational complexity with increasing dimensionality. With multiple processors, the objective calculations and calculation of the Voronoi diagram can be 'farmed' out to the next free processor, or a slightly sub-optimal approach can be used of generating a small set of points, one for each processor, from each Voronoi diagram.

5 Binary Search Algorithm

Although the Voronoi approach gives a very neat and near optimal algorithm, the computational complexity for high dimensionality problems is immense. An alternative strategy has been developed that uses a binary search tree [5] to divide the search space into empty regions, allowing the largest empty region to be approximated. The search tree is constructed as shown in Fig. 4 by generating a point within the chosen hypercube, then dividing the hypercube along the dimension that yields the most 'cube-like' subspaces. The definition of 'cube-like' is the split that minimises (3), where $d_{\max}$ is the maximum side length of the sides of the two subspaces, and correspondingly $d_{\min}$ is the overall shortest side length.

$$\mathcal{C} = \frac{d_{\max}}{d_{\min}} \tag{3}$$

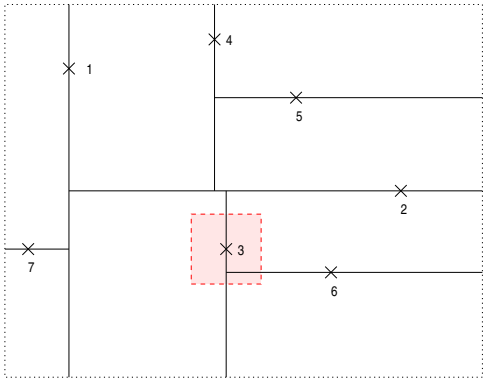

Fig. 4. Binary search process

The ideal algorithm in section 2 has been modified for the binary search tree thus:

1. **Exploration:** Next point is generated within the largest empty region,
2. **Exploitation:** Next point is generated within the largest empty region that is within a small distance of a selected *good* point.

When the region with the largest area is identified for exploitation, a point is generated at random that lies within the bounds of the region. For this paper, the point is generated with a normal distribution about the mean of the region axes in each dimension. The normal distribution is scaled so that each dimension of the region is ± 4 standard deviations wide.

The identification of a local region for exploitation is illustrated in Fig. 4. A small offset distance d_p is used to generate a hypercube of interest about the chosen point (chosen with tournament selection). The small hypercube is placed around the point of interest simply to provide an efficient means of identifying neighbouring regions. A new point is then generated at random using a normal distribution in the largest region that intersects the hypercube. For a region A to intersect the hypercube about the point P, (4) &(5) must not be satisfied for any dimension i, where A_{i_L} and A_{i_H} are the lower and upper extents of region A in dimension i.

$$A_{i_L} > P_i + d_p \tag{4}$$

$$A_{i_H} < P_i - d_p \tag{5}$$

As the tree is traversed, the subspaces of the higher order nodes (say upper subspace of node 2, marked with a cross in Fig. 4 for example) are also tested. If the subspace fails, all the child nodes of the subspace can be ignored.

At each iteration, the tree search to find the largest empty region is at worst $O(mn)$, where n is the number of evaluation points so far and m is the number of dimensions. The tree pruning can lead to $O(mlog(n))$ performance for exploitation, and at worst $O(mn)$. Thus the computational explosion associated with the Voronoi approach is avoided. Overall, the computational complexity for the binary search is at worst $O(mn{x}^4)$ if a Pareto ranking method of $O(n^2)$ is used to rank all the points. If only the members of the tournament are ranked, the computational complexity will be between $O(mlog(n)^2)$ and $O(mn^2)$, depending on how much exploitation is performed.

6 Test Functions

For the trials in this paper, two multi-objective test functions have been used. The test functions both have concave Pareto sets, with one also being discontinuous. Both test functions are detailed in [6] and are given in (6) and (7). It should be noted that for (7), the decision space and objective space are co-located, with the Pareto set being defined by the intersection of two constraint boundaries. Thus points can appear in the objective space on the lower side of the Pareto set, but these points violate the constraints and as such are not acceptable.

$$\mathcal{O}_1 = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right)$$

$$
\begin{aligned}
\mathcal{O}_2 &= 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right) \\
-2 &\leq x_i \leq 2
\end{aligned} \tag{6}
$$

$$
\begin{aligned}
\mathcal{O}_1 &= x \\
\mathcal{O}_2 &= y \\
0 &\geq -(x)^2 - (y)^2 + 1 + 0.1\cos\left(16\arctan\left(\frac{x}{y}\right)\right) \\
0.5 &\geq (x - 0.5)^2 + (y - 0.5)^2 \\
0 &\leq x, y \leq 1
\end{aligned} \tag{7}
$$

7 Experimental Results

Both of the optimisation algorithms described in the paper have been trialed on the two test functions. For comparison, a typical multi-objective evolutionary strategy has been used, and constrained to generate the same number of sample points. In all three algorithms, NSGA [1] has been used at each iteration (and correspondingly generation for the evolutionary strategy) to perform the Pareto ranking of the objective values generated so far that meet all the constraints. Only the Pareto ranking elements of NSGA are required for the Voronoi and binary algorithms. The same $\sigma_{\text{share}} = 0.05$ has been used for consistency. The ranking algorithm of NSGA is well suited to the Voronoi and binary search techniques as the Pareto set is held within the 'population' being ranked. Alternative methods based on SPEA, PAES [1] are less well suited to this application as they rely on holding the Pareto set externally. However, although SPEA and PAES have been demonstrated to out perform NSGA on many optimisation problems when used in an evolutionary algorithm, they still search with objective calculation distributions very similar to NSGA. Thus in this trial, only the point generation process is different for each of the algorithms, demonstrating how the point distribution is controlled in the new algorithms to give a much wider search with fewer unexplored regions.

The evolutionary strategy had a population of 20 and ran for 24 generations, giving a total of 500 evaluations (including the initial population). A crossover rate of 0.7 was used with real-valued intermediate crossover. The initial standard deviation of the mutations was set to one eighth of the range of each parameter, and then allowed to adapt during the run.

For the Voronoi and binary search optimisation, a total of 500 points were generated, with the exploration and exploitation parameters set to $K = 0.04$, $C = 0.02$ & $\sigma = 0.1$ and a tournament size of 10. For the binary search optimisation, the local exploitation hypercube was set to be ± 0.02 in each normalised dimension.

Figures 5 & 6 show the objective and decision space for (6), the first test function and Voronoi optimisation. In Fig. 6 the regular pattern outside of the region of the optima is very clear, demonstrating well controlled and uniform exploration. The tight clustering around the centre of the plot corresponds to the Pareto set and the exploitation phase of the algorithm.

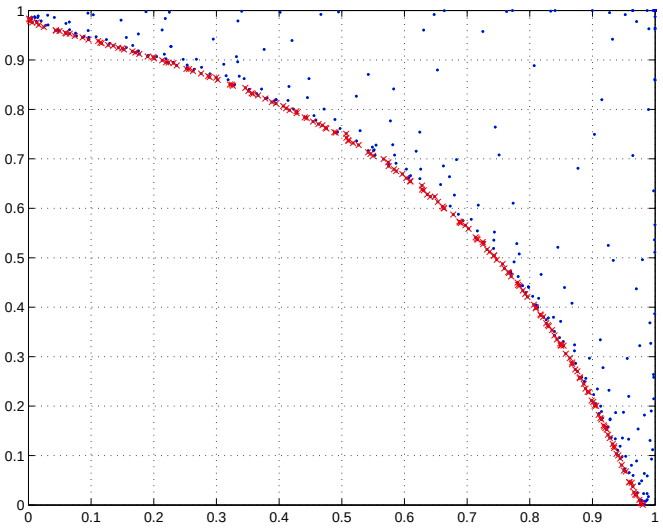

Fig. 5. Objective space for equation 6 and Voronoi Optimisation

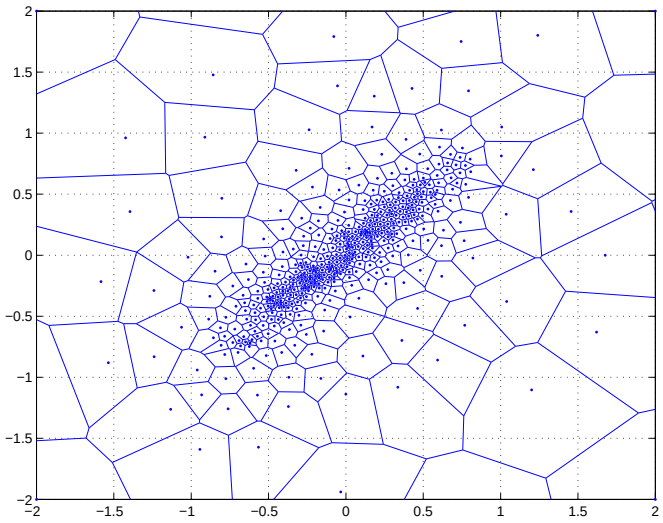

Fig. 6. Decision space for equation 6 and Voronoi Optimisation

Figures 7 & 8 show the objective and decision space for (7), the second test function and for Voronoi optimisation. Again, the regular pattern, created during the exploration phases, outside of the region of the optima is very clear in Fig. 8. The tight clustering in the lower left corner of the plot corresponds to the Pareto set and the exploitation phase of the algorithm. The non-convex and discontinuous Pareto set is clear in the plot. The

light coloured points also indicate the region of the plot which is constrained. It is interesting to note that the boundary between the constrained and feasible regions that form the Pareto set has been explored in detail, providing an accurate approximation to the Pareto set.

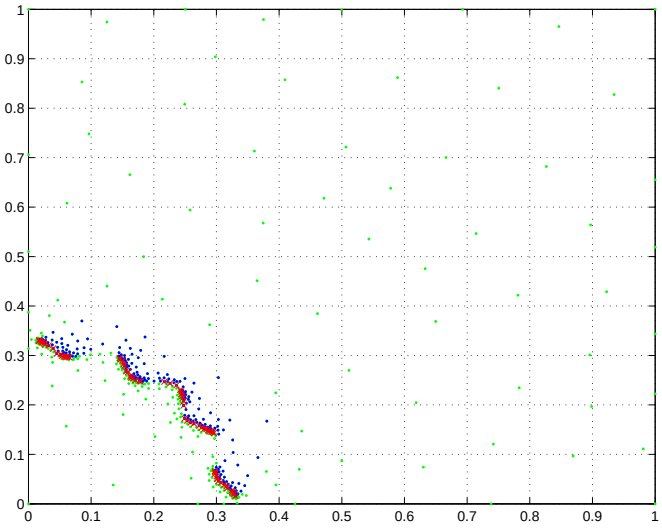

Fig. 7. Decision and Objective space for equation 7 and Voronoi Optimisation

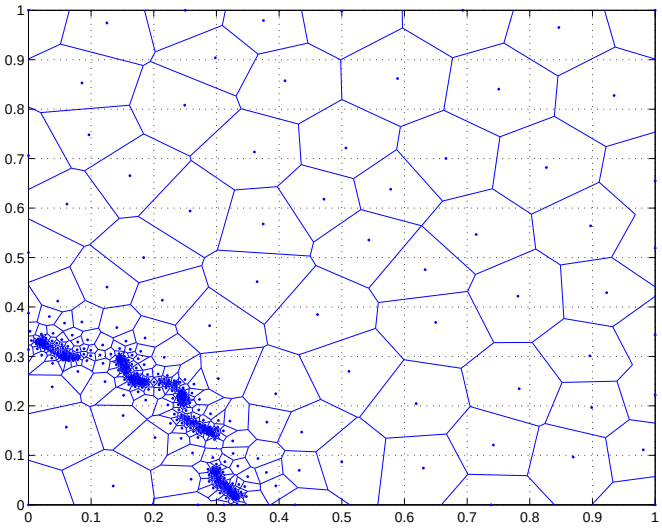

Fig. 8. Decision and Objective space for equation 7 and Voronoi Optimisation

Figures 9 & 10 show the objective and decision space for (6) and binary search optimisation. The regular pattern outside of the region of the optima is not quite as clear as with the Voronoi approach, but is still well defined and demonstrates a controlled and uniform exploration. Again we see a tight clustering around the centre of the plot which corresponds to the Pareto set and the exploitation phase of the algorithm.

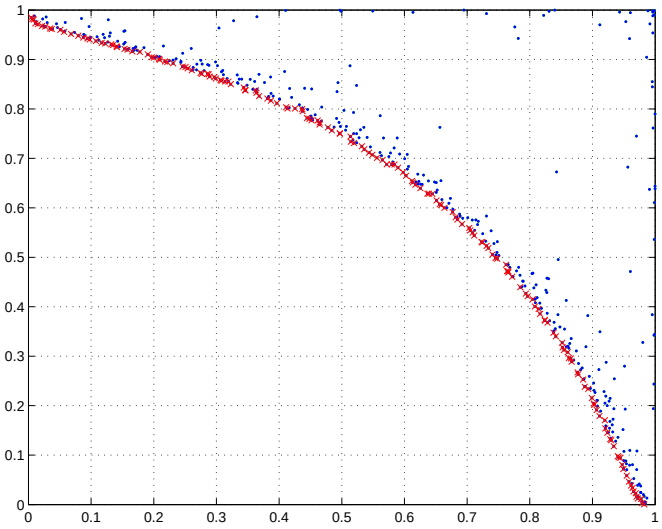

Fig. 9. Objective space for equation 6 and Binary Optimisation

Figures 11 & 12 show the objective and decision space for (7) and binary search optimisation. Again, the exploration pattern outside of the region of the optima is clear with most of the unexplored regions being of a similar size. The tight clustering in the lower left corner of the plot again corresponds to the Pareto set and the exploitation phase of the algorithm. The non-convex and discontinuous Pareto set is clear in the plot. The light coloured points indicating the region of the plot which is constrained also show that the regions that form the Pareto set have been explored in detail.

Figures 13 & 14 show the objective and decision space for (6) and optimisation using a multi-objective evolutionary strategy. There is no regular pattern outside of the region of the optima and there are some large unexplored areas near the top of the plot. This demonstrates that the exploration is not as controlled or as uniform with the low number of sample points used in the experiment. Again we see a clustering around the centre of the plot which corresponds to the Pareto set and the exploitation phase of the algorithm, but here the cluster is very loose and ill-defined.

Figure 15 shows the objective / decision space for (7) optimised with the evolutionary strategy. The exploration pattern away from the Pareto set is clear, but has large unexplored regions. The Pareto set is patchy and the constrained region in the lower left corner has been searched more heavily than the other constrained regions, but not in a controlled way near the Pareto set. This bias in part due to the performance of NSGA.

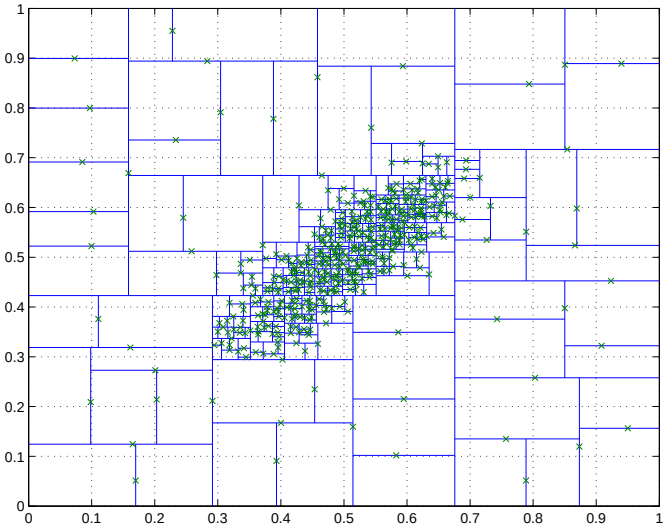

Fig. 10. Decision space for equation 6 and Binary Optimisation

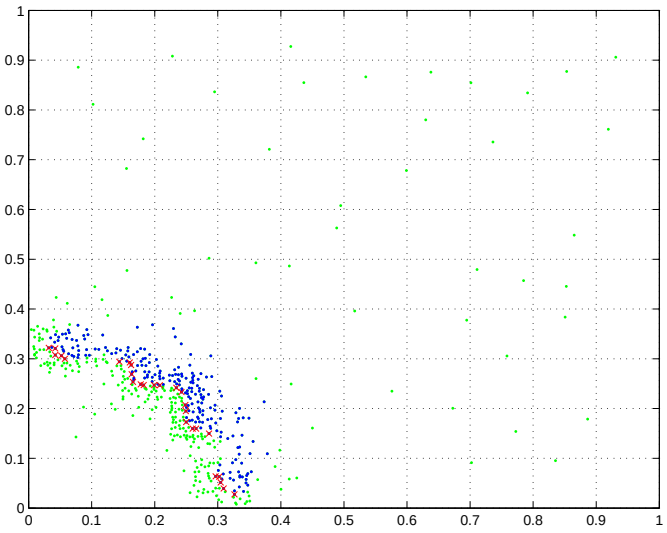

Fig. 11. Decision and Objective space for equation 7 and Binary Optimisation

The location of the Pareto set is not as accurate and well defined as with either the Voronoi or binary search optimisation algorithms.

To quantify the ability of the algorithms to identify points on the Pareto set, 100 trials of each algorithm for each objective were performed and the number of distinct points on the final non-dominated set recorded. Table 1 shows that the Voronoi op-

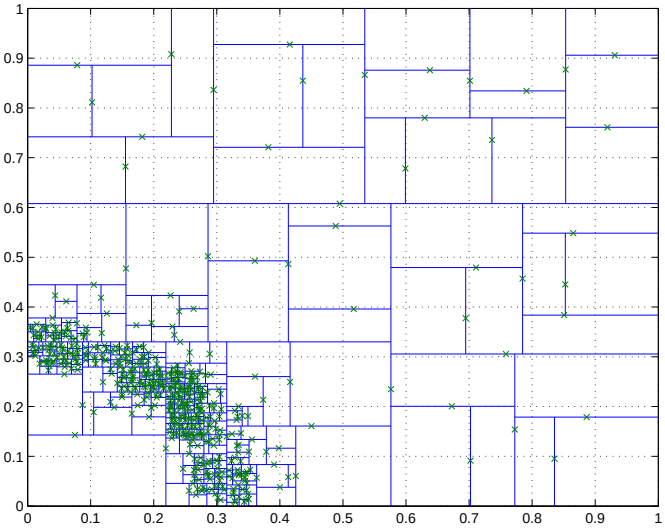

Fig. 12. Decision and Objective space for equation 7 and Binary Optimisation

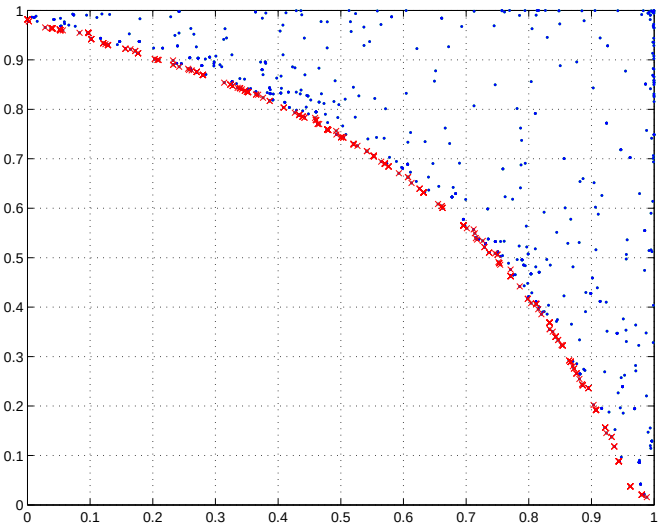

Fig. 13. Objective space for equation 6 and Evolutionary Strategy

timisation algorithm is by far the most effective at identifying non-dominated points, with the binary search algorithm a close second. Experiments have also shown that the algorithms are well suited to problems where up to $10,000$ points or more are to be generated.

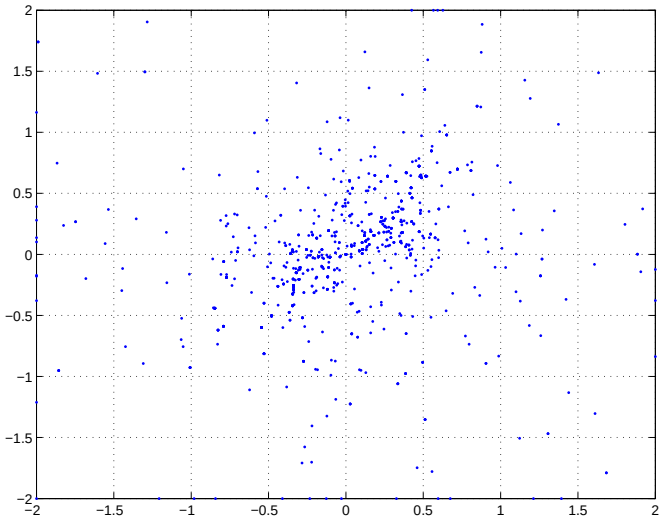

Fig. 14. Decision space for equation 6 and Evolutionary Strategy

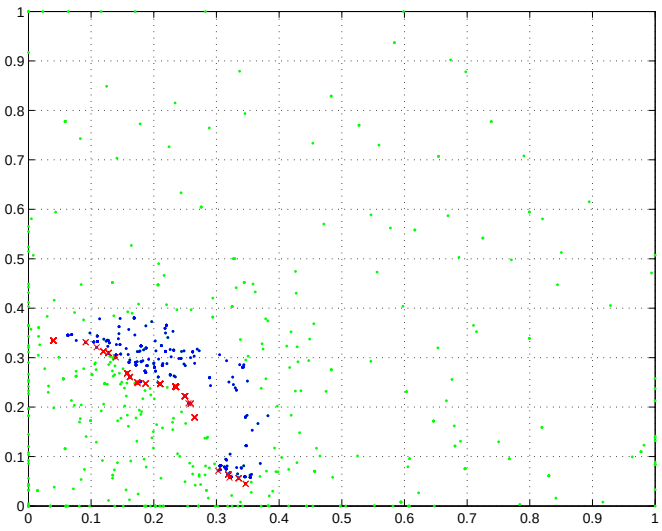

Fig. 15. Decision and Objective space for equation 7 and Evolutionary Strategy

8 Conclusions

This paper has introduced the concept of a new idealised search algorithm and two methods for approximating it, for problems where very few objective calculations can be performed. The results have shown that both the Voronoi and binary search ap-

Table 1. Number of distinct points on Pareto set over 100 trials.

Algorithm	Function	Minimum	Mean	Maximum	σ
Voronoi	(6)	170	194.9	221	10.5
Binary	(6)	147	170.0	174	6.7
ES	(6)	74	98.4	122	11.3
Voronoi	(7)	56	65.9	75	3.6
Binary	(7)	13	28.8	37	4.2
ES	(7)	8	17.2	27	3.9

proach allow full, independent control over the exploration and exploitation phases of the search. The exploration phase is designed to give uniform coverage of the search volume by targeting unexplored areas directly. The exploitation phase gives uniform coverage in 'good' areas by targeting unexplored regions close to points that perform well. The algorithms exploit multi-objective techniques developed for evolutionary algorithms and can handle multiple objectives and constraints easily for problems with multiple variables.

The research has also shown that for problems with very few parameters, the Voronoi search algorithm performs the best when compared to the binary search optimisation and a typical evolutionary solution. The computational complexity increases rapidly though for the Voronoi approach when the number of variables is increased. The binary approach scales linearly with an increasing number of dimensions, allowing large problems to be tackled. The research demonstrates how the two new algorithms exploit the information from all the evaluations performed to give much more structure to the location of trial points when compared to a typical evolutionary approach.

9 Acknowledgements

The author would like to acknowledge the use of the Department of Aerospace, Power, and Sensors DEC Alpha Beowulf cluster for the production of the results.

References

1. Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001. ISBN 0-471-87339-X.
2. Joseph O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1993. ISBN 0-521-44592-2.
3. Franz Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Comput. Surveys*, 23:345–405, 1991.
4. Malcolm Sambridge. Geophysical inversion with a neighbourhood algorithm – I. Searching a parameter space. *International Journal of Geophysics*, 138:479–494, 1999.
5. Mark Allen Weiss. *Algorithms, data structures, and problem solving with C++*. Addison–Wesley Publishing Company, Inc., 1996. ISBN 0-8053-1666-3.
6. David A. Van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis. Technical Report TR-98-03, Air Force Institute of Technology, 1 Dec 1998.