Tina Memo No. 2002-008
Internal Report

Diagnosis of Dementing Diseases through the Distribution of Cerebral Atrophy: Development of a Multi-Objective Evolutionary Algorithm Optimiser

P. A. Bromiley and N.A. Thacker

Last updated
1 / 7 / 2002

Imaging Science and Biomedical Engineering,
School of Cancer and Imaging Sciences,
University of Manchester, Stopford Building,
Oxford Road, Manchester M13 9PT, U.K.

Diagnosis of Dementing Diseases through the Distribution of Cerebral Atrophy: Development of a Multi-Objective Evolutionary Algorithm Optimiser

P.A. Bromiley and N.A. Thacker
Imaging Science and Biomedical Engineering,
School of Cancer and Imaging Sciences,
University of Manchester, Stopford Building,
Oxford Road, Manchester M13 9PT, U.K.
paul.bromiley@man.ac.uk

Abstract

This report describes the development of a multi-objective genetic algorithm within the TINA machine vision software. This algorithm is intended to be used to optimise the diagnostic capabilities of the dementing disease diagnosis technique described in [23]. Many features previously described in the literature are incorporated: the use of a secondary population, mating restriction, random immigration, and deterministic crowding. However, a novel mating restriction is used in order to prevent global convergence in the primary population, such that the primary population searches the space indefinitely. Convergence is performed exclusively in the secondary population. This division of searching and convergence across two populations avoids the difficulties commonly associated with the trade-off between these two behaviours, such as premature convergence, but without applying features that artificially enforce diversity in the population (e.g. fitness sharing) or require problem-specific adjustments (e.g. annealing schemes). The algorithm is compared to four other algorithms incorporating fitness sharing, and is shown to produce a better estimate of the Pareto front over a range of real-valued multi-objective test problems.

1 Introduction

A previous publication [23] described a technique capable of diagnosing a number of dementing diseases through analysis of the distribution of cerebral atrophy in MRI scans. In brief, the technique divides the interior of the cranium into twelve volumes (anterior, middle and posterior thirds, plus horizontal and vertical dividing planes placed centrally) and counts the number of pixels containing CSF in each. These quantities are then normalised for head size, through division by the total volume, and for age-related atrophy, assuming a simple inverse proportional relationship. Finally, the twelve normalised variables are combined to produce five relative variables. This technique has been applied to a study group incorporating normals, Alzheimers Disease, Fronto-Temporal Dementia and Vascular Dementia. Projected plots of the 5D space show good separation of these groups, and the results from applying a cross-validated Parzen classifier indicated that the four disease groups could be diagnosed with an accuracy of around 80%.

The previous work can be considered as a proof of concept, indicating that relatively crude measures of cerebral atrophy can produce useful diagnostic information, whilst avoiding the complexities of non-rigid co-registration and segmentation that would be introduced in any attempt to produce accurate atrophy maps. Having illustrated the viability of such techniques, it seems natural to attempt to optimise the diagnostic capabilities. In particular, the definition of the twelve boxes used in the CSF counting is sub-optimal, since the box sizes approximate the underlying structure of the brain on the coarsest scale (division into lobes) and yet ignore this information. Therefore, it seems reasonable to attempt to optimise the placement of the decision boundaries to maximise diagnostic capability. The processor time required to compute the relevant cost function over any reasonably sized patient group precludes the use of exhaustive search. Little is known about the form of the cost function: it may be non-smooth and discontinuous, and first differentials will not be available. Since the optimisation must be performed over a number of diseases, and using enough example data sets to guarantee generalisation capability, a multi-objective optimisation routine that does not use differentials is required.

Of the multi-objective optimisation techniques described in the literature, the multi-objective genetic algorithm seems most applicable to the task at hand. Such algorithms are modelled on the process of evolution, using

a population of candidate solutions to explore the space, combining information from the solutions to generate better solutions, and incorporating survival of the fittest to promote convergence. The basic element in this case is the individual: each contains a *chromosome* that encodes the position of a point in the space defined by the cost functions. A population of random individuals is set up, and then pairs of individuals are *selected* for *breeding*. During breeding, genetic information from the parents is recombined in some way to generate children. Random *mutation* is applied to the child chromosomes. The children are then evaluated against their parents (in terms of the cost function) and the weakest are *replaced* (the fittest are retained) in the population ready for the next round of breeding. All evolutionary algorithms incorporate these four features (selection, crossover, mutation and replacement) in some way. At the crudest level, crossover allows the best features from a chromosome pair to be combined, such that the children may inherit the strengths of each. Mutation ensures that some random searching of the space occurs. Replacement ensures convergence. Selection is more subtle and can influence the progress of the algorithm in a number of ways, such as speeding convergence by allowing the fittest individuals to breed more often. It should be recognised that the literature describes a wide range of variations on each of these steps. Since the first genetic algorithms were implemented in the mid-eighties [7, 18], over 450 publications [3] have described various implementations, applications and, to a lesser extent, underlying theory. Several reviews exist in the literature e.g. [5, 8, 25] and so no attempt will be made to provide an exhaustive survey here.

In single objective optimisation, the desired result is usually one or more global and/or local optima. Ideally the cost function should be defined on the basis of quantitative statistical principles, derived from a probabilistic specification of the task [22]. However, in many practical problems we need to take into account several factors that cannot be easily combined into a single simple objective. Such non-commensurate objectives require approaches that can optimise multiple objectives simultaneously and independently. The transition to multiple objectives can be handled in several ways [6]. For instance, a weighting scheme can be used to combine the objectives into a single value. However, the solution then becomes sensitive to the values of the weights. These can be randomly varied, or as an alternative the search can be directed to each objective in turn, or the objectives can be ranked in order of preference and optimised in turn, but in general all such techniques are unsatisfactory or at best problem-specific.

It is possible to demonstrate these relational difficulties using probability theory, in order to arrive at some general conclusions regarding what might be achieved in such problems. To form a conclusion within a hypothesis framework, a single probability relating to the best hypothesis is required. Given a set of objectives a_i and some hypothesis H, the probabilities of the hypothesis given the a_i could be calculated by making certain assumptions A. Then, assuming that these probability functions were available and were statistically independent, their product would give the required single, overall probability

$$P(H|\bar{a}, A) = \prod_i P(H|a_i, A) \tag{1}$$

If, however, the $P(H|a_i, A)$ were kept separate a space of candidate solutions formed, we would expect to find that the space had a bounding surface featuring cusps and ridges corresponding to local extrema in the combined objectives. It would be these locations that would produce the local and global optima of the combined function.

In practice, for the majority of multi-objective problems, the $P(H|a_i, A)$ will not be available, and only the a_i will be known. In addition, the $P(H|a_i, A)$ may not even be independent, producing a rotation of the axes of the space of $P(H|a_i, A)$ (the probability space). This dependency may even vary across the space of the a_i (the objective space), making it very difficult to ever be able to formulate a simple strategy back to the probabilistic interpretation. Also, the required set of assumptions A may be ambiguous or even impossible to define. Therefore, in general, distances and geometry in the objective space are largely meaningless. However, in general the a_i will be monotonically related to the probability space (a one-to-one mapping).

$$P(H|\bar{a}, A) = f_i(a_i) \tag{2}$$

Since we expect diffeomorphic equivalence between the probability and objective spaces, ridges (local peaks in one dimension) and cusps (local peaks in two or more dimensions) in $P(H|a_i, A)$ are still ridges and cusps in a_i, and we would expect to find useful global optima around these positions. These positions in the parameter space define the Pareto-front and it is this front that we wish to locate when we use a multi-objective genetic algorithm. It is clear therefore that the assumptions required to transform between these spaces specify a certain trade-off between the objectives, and that the theoretical global optimum in the probabilistic space corresponds to a point on the Pareto front in the objective space, generally around the cusps (since these represent extrema in one of the a_i on the Pareto front). Therefore, the output from a multi-objective evolutionary algorithm must contain multiple optima, which sample all possible optima that could have been candidates for the theoretical global optimum were the required assumptions available. We will explicitly define this as the goal in our work.

Since the desired goal in multi-objective optimisation is to produce a number of optimal solutions spanning the Pareto front, the use of a genetic algorithm conveys an immediate advantage in that such algorithms necessarily maintain a large population of solutions and so can capture the entire range of the Pareto front in a single run. However, they also convey a second, more subtle, advantage. As long as the population does not become degenerate, several optimisation strategies are available. The most powerful of these is the combination of "building blocks" [9], optimised sub-sections of chromosomes, from individual solutions to produce fitter solutions. If this strategy is not available, the algorithm will decay in turn to stochastic hill-climbing and finally random search. This behaviour guarantees that, given infinite computational resources, global optima will always be found. It is therefore relatively easy to produce a naive genetic algorithm that simply implements crossover, mutation and patricide and locates the global optima eventually. In addition the "No Free Lunch" theorems [28] state that an genetic algorithm must incorporate problem-specific knowledge in order that a formal statement about general effectiveness can be made, and so there is no such thing as a "best" genetic algorithm. However, testing on a broad range of test functions, chosen to exhibit a range of behaviours, should be enough to indicate the potential of an genetic algorithm.

One of the main issues facing any optimisation routine is striking a tradeoff between the diametrically opposed objectives of searching and convergence. An important aspect of this problem in the case of multi-objective optimisation using an genetic algorithm is maintaining the population diversity. When presented with multiple optima a naive genetic algorithm will still converge onto a single solution, due to three effects: selection pressure, selection noise, and operator disruption [14]. This phenomenon, known as genetic drift, has been seen in natural as well as artificial evolution [6]. A wide range of mechanisms known collectively as *niching*, *sharing* and *crowding* methods exist to maintain the diversity of the Pareto front [14]. A popular example is fitness sharing [11], which essentially penalises the fitness values of solutions in close proximity to each other.

The multi-objective genetic algorithm literature describes a wide variety of diversity preservation strategies. However, these modifications also often require additional arbitrary parameters in order to control the search and convergence of the algorithm, in addition to key parameters such as population size and the probabilities governing the application of operators such as mutation and crossover. Unfortunately, these parameters often need tuning to each specific optimisation problem. In addition, the design choices made for the construction of an algorithm will have implications regarding the ideal formulation of optimisation problems, for example the construction of chromosomes. One goal of this work was therefore the specification of a search strategy that efficiently identifies of the Pareto-front with a minimum number of control parameters, and an understanding of efficiency from the point of view of problem formulation. In particular we will show how to eliminate the need for accurate determination of population size and arbitrary parameters to reduce crowding, and the characteristics a chromosome must have for efficient search.

To rephrase the problem, we desire an algorithm that will search the space globally to identify promising regions of the cost function, but converge locally within these regions to find the optima. The analogy of speciation in natural evolution immediately suggests itself, leading to the concept of restricted mating. A wide range of variations on this theme have been used in the past [14], restricting mating so that it only occurs between similar (or dissimilar) individuals. However, unless the population is divided into separate species that are never allowed to interact, convergence will be delayed rather than prevented and collapse to a single point will still occur.

The approach taken here is quite different and relies upon novel use of several features found in previous work: the use of a secondary population to store the non-dominated individuals found so far; random immigration (i.e. introducing new random individuals into the population); and restricted mating, using a novel criterion that allows mating only between neighbouring individuals. The concept is to enforce the differentiation of global searching and local convergence. We consider the primary population as a tessellation of the search space with Voronoi cells [27], and by allowing mating only between Voronoi neighbours enforce the algorithm's own natural definition of local/global. Additionally, we extract individuals from the primary population if they have undergone some number of attempts at mating without producing fitter children ("static extraction"), and place them in a secondary population, which is sorted such that it contains the current non-dominated set of extracted individuals. The extracted individuals are replaced by random immigrants, such that the primary population size remains constant. We demonstrate that this prevents any general convergence of the primary population, thus dividing the processes of searching and convergence over the primary and secondary populations respectively. It also extends any initial definition of the population size in time so that, beyond a minimum size necessary to maintain a breeding population across the space, the population is potentially infinite. This therefore removes this free parameter from the specification of the algorithm. We test the algorithm on a range of single and multi-objective test problems found in the literature, and show its performance to be at least equal to a range of other genetic algorithms previously described.

As a final note, it is important to stress the differences between genetic algorithms that use binary and real-valued representations. In the first case, the analogy with a biological chromosome, and thus the meanings of crossover

and mutation, are more obvious. Crossover can be performed by taking the string of bits encoding each parent, splitting them at some point, and swapping over the information below that point, and mutation can be performed by flipping a single bit. On a more subtle level, the number of solutions within a constrained space is countable, and there exists a simple definition of identicality that can be used in various ways. However, in order to represent a real-valued problem in binary form some loss of accuracy must be accepted. The length of the binary string can always be increased to improve this accuracy, but it has been shown [10] that even for simple problems the required population size is of order of the string length. Conversely, real-valued representations can be arbitrarily accurate, and allow the calculation of distances within the space. However, the definitions of mutation and crossover become problematic. In previous work we defined a genetic algorithm for the design and routing of 3D hardware, which is formulated as a binary search problem [12]. The algorithm presented here can be seen as an extension of that work and uses a real-valued representation, no further investigation of the tradeoffs between binary or real-valued representations will be made.

2 Method: Implementation

The algorithm presented here incorporates the following features. It uses two populations: the primary population and the extracted set. Upon initialisation, the primary population is filled with a fixed number of randomly-generated solutions, which thereafter remains constant. The solutions are held in data structures called genomes, which contain a set of header information and a list of points encoding the position of the solution in the cost function space. The header information contains a number of fields for monitoring the progress of each genome, as well as genetic information such as mutation probabilities.

The algorithm then runs through a loop that includes selection, crossover, mutation, patricide, and extraction management. Each loop breeds one pair of solutions, with children directly replacing their parents (rather than competing with the entire population). Algorithms of this type have been referred to in the literature as steady-state genetic algorithms, as opposed to generational genetic algorithms where the entire population is paired and bred at each step. In order to aid comparisons with such algorithms, the word "generation" will henceforth be used to describe a number of executions of the main loop equal to the population size. In addition, allowing children only to replace their own parents, through a fitness competition, has been referred to as deterministic crowding [14]. Some comments on the limitations imposed by combining deterministic crowding with mating restrictions are included below.

2.1 Selection

Selection is restricted but random within a subset of the population. Considering the population as a tessellation of the search space with Voronoi cells, mating is restricted to within sets of neighbours. A simple restriction criterion was derived to reject pairings that did not involve neighbours [2]. Referring to Fig. 1, if the pairing **AB** was being considered, it was compared to all other pairings involving the same first point e.g. **AC**. The planes bisecting the vectors linking the points are necessarily the Voronoi boundaries if no intermediate cells lie between the individuals. Therefore, if the plane bisecting **AC** intersects **AB** (point I) at less than half of its length (to the mid-point M), **AB** must cross an intermediate cell. The angle **AB**$\angle$**AC** is given by

$$\theta = cos^{-1} \frac{\mathbf{AB}.\mathbf{AC}}{\mid \mathbf{AB} \mid\mid \mathbf{AC} \mid}.$$

Making use of the fact that **AI** is the hypotenuse of a right-angled triangle, the condition for a pairing to be rejected is

$$\mid \mathbf{AM} \mid > \mid \mathbf{AI} \mid \Rightarrow 1 > \frac{\mid \mathbf{AC} \mid^2}{(\mathbf{AB}.\mathbf{AC})}.$$

Selection was then performed by randomly selecting a first parent, and then looping over randomly selected second parents until a Voronoi neighbour was found. The parent pair was then passed on to the later stages (crossover etc.).

2.2 Crossover

For the test problems described here, each genome encoded a single, real-valued point. Therefore, the definition of crossover became problematic, since there was no obvious justification for crossing over sub-sets of the coordinates

of a pair of points to generate children. Several crossover operators for use on such representations have been suggested in the literature (see refs. in [15]). We implemented one of the simplest: child points were placed randomly within a hyper-cuboid with edges parallel to the axes of the space defined by placing the parent points at diametrically opposed corners.

In order to complete each child genome, the header information had to be generated. The only fields over which there was any choice were the mutation probabilities, and no crossover was attempted on these. Each child randomly picked up the entire relevant portion of the header from one of the parents.

2.3 Mutation

In binary representations, mutation can be implemented by flipping a single bit in the genome, thus introducing a limited amount of change. All that is required is a probability of mutation. In real-valued representations, the situation is more complex as a definition of a "limited" amount of change is required in addition to the mutation probability. Although the literature contains suggestions of suitable mutation probabilities (e.g. [1], there seems to be little theoretical justification. Therefore, we took an alternative approach of using the GA to optimise its own mutation parameters. The header for each genome contained five pieces of mutation information: the probability that breeding would occur with no crossover, only mutation; the probability that, after crossover, there would be a mutation in the header; the probability that, after crossover, there would be a mutation in the spatial position of the genome; the maximum distance (as a proportion of the total width of the space) by which a coordinate would be changed if mutated; and the maximum amount (in terms of probability) by which a mutation probability would be changed if mutated. Default values were 0.5, 0.5, 0.2, 0.1 and 0.1 respectively. Since these quantities were placed in the header and could themselves be mutated, they could be optimised by the GA itself.

It should be noted that the mutation probabilities in the header did not influence the fitness of a genome in the same way as its spatial position in terms of the cost function. The latter was a direct effect: the position of a genome in the space implies values of the cost function, which in turn determine whether it survives to breed. The mutation probabilities determined in part the probability that a genome would generate children that would go on to breed, and so were one step removed from directly influencing the fitness. For this reason, optimisation of the header information was slower than optimisation of the cost function. It was expected that optimisation by crossover would be more efficient at points far from optima, since it would allow rapid movement, whereas optimisation by mutation would be more desirable close to optima, where it would allow detailed local random searching. The meaning of "optimal" for header information was therefore linked to the spatial position of the genome in terms of the cost function. Therefore, although the initial random population was assigned initialisation header information entered by the user (typically 0.5 for the probability of breeding by crossover and 0.1 for all other values), once the main loop had begun, randomly generated genomes were given random spatial positions, and then copied header information from the spatially nearest genome in the population, based on a Euclidean distance measure. This encouraged the optimisation of header information. It also had a biological analogue [1].

2.4 Patricide

The patricide stage involves comparison of the children with the parents in terms of the cost function, implementing survival of the fittest. The cost function is evaluated for both children, and their costs compared with the parents. Two versions were implemented. The first involves a "first come, first served" principle. A loop over the parents containing a nested loop over the children was used to compare both children with both parents, replacing a parent with a fitter child as soon as such a combination was found. However, a situation could arise in which the first child was fitter than both parents, but the second child was only fitter than the first parent. In this situation, the first child would replace the first parent before the second child was tested. The second child would not then be able to replace the second parent, and so would be discarded. In order to avoid this behaviour, a maximally child-preserving patricide routine was also implemented that tested all possible combinations of children and parents, and sorted the replacements so that as many occurred as possible. However, this added complexity was found to be unnecessary in practice, and so the simple patricide routine was used.

Since the cost function encoded multiple objectives, a definition of a child being "better than" a parent was required. It was not sufficient to look for children that beat a parent on any single objective, since this could lead to oscillatory behaviour: a child could replace its parent on objective A whilst getting worse on objective B, then the child might breed to regenerate the parent, which would replace the child on objective B whilst getting worse on objective A, and so on. In order to avoid this, a vector of binary direction flags was encoded in the header of

[1] Many species of bacteria are able to exchange genetic information in a similar way, allowing them to rapidly assume genotypes suited to local conditions.

each genome, which listed the objectives on which the genome had beaten its parents (but was set to all zeros for randomly generated genomes). In order to replace its parent, a genome had to optimise on the same objective(s) as the parent, although additional objectives could be added if the child started to optimise on more objectives than the parent had done.

The patricide function also included the possibility of a second pass through the breeding and mutation stages. If the parents had bred by crossover, but had failed to generate children that replaced them, then breeding was repeated through the cloning/mutation of spatial position route, and patricide was repeated on the new children. This was implemented in order to encourage random searching by mutation for solutions close to an optimum, allowing detailed refinement of such solutions. The final step in the patricide routine was the updating of the ages of the genomes. The term "age" here refers to the number of times that a genome has bred without producing children that replaced it, and this quantity was used in the extraction management function described below.

Allowing children to replace only their own parents in the population has been described in the literature as deterministic crowding [14]. Many crowding mechanisms have been proposed in the past, with the aim being to replace the most similar elements in the population. This is a diversity preservation strategy, as it prevents highly fit individuals near the minima of the cost function eliminating less fit individuals elsewhere in the space.

2.5 Extraction management

Following the patricide stage, the breeding pair pointers, which might point to children that had replaced their parents, was passed to the extraction set manager. The age of each genome was tested against a fixed extraction age, which was typically set to 5-10 breeding attempts. If the genome was older than this, it was extracted from the population and replaced with a randomly generated genome. This is similar to the diversity-preserving strategy known as "random immigration", which has previously been described in the literature.

The algorithm was implemented in such a way that the number of objective functions was completely variable. Therefore, although it could be set to any number of multiple objectives, it could also be set to 1 for single-objective optimisation. All of the preceeding stages in the algorithm could cope with either scenario, but the desired output in the extracted set was different in each case. For single-objective optimisation, we desire (at least) a single global optimum, or possibly a list of global optima if more than one exist. Arguably, we might also desire the list to contain all local optima, providing additional insight into the behaviour of the cost function, although the list should be sorted such that the first member is a global optimum. We would not want the list to contain multiple copies of essentially the same solution. Since the algorithm presented here used a real-valued representation, a simple definition of identicality was not available. However, it is inevitable that the representation used would impose some accuracy (typically the root of the smallest number that could be represented as a double type variable) [16] and so we are justified in reducing this accuracy in order to provide a distance measure as a definition of identicality.

In the case of single-objective optimisation, simply discarding "identical" solutions (always keeping the one with the lowest cost) is not sufficient to produce a list containing only local (and global) optima. On adding a new solution to the extracted set, any solutions added in pervious breeding rounds, in the same local minimum, but at a distance greater than the distance used to define identicality, would not be discarded. Therefore, an additional constraint was introduced to determine if a solution lay in the same local "valley" in the cost function as another solution already in the extracted set. The extracted solution was compared to all solutions in the extracted set, and the cost function at the average of the two positions was calculated. If, for any comparison, the cost at the mean was lower than the cost of either of the genomes, then the new genome lay in the same local valley in the cost function as the member of the extracted set against which it had been compared. In that case, the better of the two genomes was placed in the set, and the worse discarded. Of course, a situation could arise in which both genomes lay in different valleys, with a third valley in between at the position of the mean, leading to a genuine new local optima being discarded. However, one of the convenient properties of a GA is that, by the time a given solution is generated, the genetic information required to produce it is usually present in partial form in several individuals in the population, and so the solution will be regenerated multiple times, and will rapidly get added to the population. In addition, on functions that exhibit minima with complex shapes, such as the long, thin, curved minima of the Rosenbrock's Valley function [17], this method will still allow multiple solutions in the valley to be added to the extracted set.

There is one remaining situation in which local optima might never get added to the extracted set, illustrated by the six-hump camelback function described below. This function has six local optima, two of them global. Two of the local optima have costs relatively close to the global optima (compared to the range of the function) but two have relatively high costs. These high-cost local optima were never added to the extracted set, since there are points on the function outside the local valley, but still relatively close, with lower costs. Therefore, the high-cost

local optima tend to get replaced by children before they can be extracted. This behaviour could in theory be prevented by reducing the extraction age to a very low value, but this would degenerate into a random search. However, it is arguable that high-cost local optima would never be desired in the output from an optimisation routine, and so this behaviour was not considered to be a serious drawback.

In the case of multi-objective optimisation, the extracted set should contain a good representation of the Pareto front i.e. solutions spread along the whole length of the front. In the literature, this has sometimes been interpreted as meaning that the solutions should be evenly distributed along the front, and performance metrics measuring the uniformity of this distribution have been suggested [20, 19]. However, it is arguable that this idea contains a fundamental flaw, since it assumes that all non-dominated solutions are equally desirable. In fact, as explained in our discussion of the relationship of the Pareto-front to probability theory, this may not be the case. Solutions close to the ends of the Pareto front, or on cusps in the front (in cost function space) can be more valuable as they represent the best attainable performance on one of the cost functions (at the expense of the others). To paraphrase Orwell, all non-dominated solutions are equal, but some are more equal than others. The GA presented here tends to generate more solutions at these positions than at other regions due to the cost extrema found there. Under some circumstances, it may be desirable to preserve this behaviour, as it provides more choices of solutions with extreme behaviour on one of the costs. To compromise, since it was desired to test this algorithm against others in the literature that implemented clustering, a simple version of clustering was implemented. New solutions were compared to those in the extracted set in terms of Euclidean distance once they had passed the dominance tests described below. If an existing member of the set was closer than some user-defined accuracy variable, then it was discarded. Although this is inferior to clustering in that the number of allowed solutions can vary for different objective functions, it was not anticipated that it would be implemented in practical problems.

The primary reason for implementing clustering in the algorithms described in the literature is to preserve the diversity of the extracted set, since in most cases the general population tends to collapse to a single point as described above. In the algorithm described here, the implementation of the Voronoi mating constraint, the use of the secondary population, and the use of random immigration, served to maintain diversity in the general population. This in turn meant that the extracted set maintained diversity without the need for clustering, and in turn clustering served only to limit the total size of the set.

To determine whether an extracted genome was to be added to the extracted set, it was tested for dominance against all existing members of the extracted set. If the extracted genome dominated any members of the set, it was added to the set and the dominated members removed. If it was itself dominated, it was discarded. If it did not dominate any members of the set, but was itself not dominated, it was simply added to the existing set.

2.6 Summary

To summarise, after initial creation of a random population of fixed length, each loop of the algorithm implements the following steps. Two individuals were selected, the first at random from the whole population, and the second at random from the Voronoi neighbours of the first. These comprised the breeding pair. A random number was generated and compared to the probability of breeding by crossover in the header of the first individual. If breeding occurred by crossover, then one of the crossover operators was applied to generate two spatial positions for the children, and they picked up the complete headers from the parents at random. If no crossover was needed, then the parents were copied. Thus, two new genomes called the child pair were generated.

Mutation was then applied to the child pair. If no crossover had occurred, mutation of a coordinate was forced. Conversely, if crossover had occurred, a random number was generated and compared to the probability of mutation in the spatial position. If mutation was required, then a further random number was generated to determine the coordinate that would be mutated, and another random number was generated, multiplied by the coordinate mutation distance in the header, and the product was multiplied by the range of the space for the relevant dimension. This gave the distance by which the coordinate was to be shifted. An equivalent operation was then carried out for mutations in the header.

The cost functions were then evaluated for both children, and the children compared to the parents in the patricide stage. Any replacements were performed and the spare genomes deleted. The original breeding pair pointers, which may now point to children that have replaced their parents, are passed to the extraction routine that checks their ages and if necessary removes them to the extracted set. The extracted set is checked for dominated solutions, and the next loop initiated.

The subject of termination criteria, the point at which the algorithm determines that the current extracted set is in some sense good enough and terminates, has also received much attention in the literature. Currently, the algorithm presented here uses the simplest form of termination criterion: a set number of loops is executed. There

is some justification for using such a criterion: it relates to the available computing power. However, it is expected that this will be replaced in future with some genuine check for convergence.

3 Method: Testing

The algorithm was implemented on both single-objective and multi-objective forms on a range of test problems commonly used in the literature.

3.1 Single-objective Testing

Since the algorithm presented here was designed to solve a multi-objective optimisation problem, the diagnosis of dementing diseases problem described above, no great effort was made to test its performance on single objective optimisation problems. In fact, only one problem was implemented, the six-hump camelback function [4]

$$f(x_1, x_2) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 \quad where \quad -3 < x_1 < 3 \quad and \quad -2 < x_2 < -2 \qquad (3)$$

The function, shown in Fig. 2 has six local optima, of which two are global. Two more are local with relatively low costs (compared to the range of values exhibited by the function between the spatial limits imposed), and the last two are local with relatively high costs.

x_1	x_2	$f(x_1, x_2)$
0.08984201310	-0.7126564030	-1.031628
-0.088984201310	0.7126564030	-1.031628
1.703606715	-0.796083568	-0.215464
-1.703606715	0.796083568	-0.215464
-1.607104753	-0.568651454	2.104250
1.607104753	0.568651454	2.104250

Table 1: Optima of the six-hump camelback function.

This function was used during the coding to test convergence before the features specific to multiple objectives (specialisation of patricide with the direction vectors, and the alterations to the extraction management function) were incorporated. It also served to illustrate some aspects of the behaviour of the algorithm. However, no tests of the rate of convergence were performed, as the GA cannot compete with other optimisation routines on such a simple function: a crude hill-climber can find the optima in a few tens of evaluations of the cost function, whereas the GA may require this many to set up the initial population, before any convergence is even attempted.

3.2 Multi-objective Testing

A number of real-valued test problems are described in the literature, and a suite of the most popular such functions have been collected in [24]. The same author has tested a number of multi-objective genetic algorithms with these test functions, using a number of performance metrics [26]. In addition, many other authors e.g. [21] use subsets of these functions to test a range of algorithms. For ease of comparison to previously published algorithms, the algorithm presented here was tested on the same functions. The test functions are given in Table 2. Although the no free lunch theorems imply that no guarantee can be given for the general performance of a multi-objective genetic algorithm, and therefore no amount of testing can predict its performance on some previously un-encountered function, testing on functions that exhibit a broad range of characteristics can at least indicate whether an algorithm is likely to perform well on some new function exhibiting a mixture of the same characteristics, and this test suite was chosen to give as broad a range of cost function behaviours as possible.

Van Veldhuizen [24] has collected together a suite of real-valued test functions, and tested a number of algorithms with a number of performance metrics [26]. For ease of comparison to previously published algorithms, the algorithm presented here was tested in the same way. The test functions are given in Table 2. This test suite was chosen to give as broad a range of cost function behaviours as possible.

MOP	Definition	Constraints
MOP1: P_{true} connected, PF_{true} convex	$F = (f_1(x), f_2(x))$ where $f_1(x) = x^2,$ $f_2(x) = (x-2)^2$	$-10^5 \leq x \leq 10^5$
MOP2: P_{true} connected, PF_{true} concave, number of decision vectors scalable	$F = (f_1(\mathbf{x}, f_2(\mathbf{x}))$ where $f_1(\mathbf{x}) = 1 - exp(-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2),$ $f_2(\mathbf{x}) = 1 - exp(-\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{n}})^2)$	$-4 \leq x_i \leq 4; i = 1, 2, 3$
MOP3: P_{true} disconnected, PF_{true} disconnected, 2 Pareto curves	$F = -(f_1(x, y), f_2(x, y))$ where $f_1(x, y) = -[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2],$ $f_2(x, y) = -[(x+3)^2 + (y+q)^2]$	$-3.1416 \leq x, y \leq 3.1416,$ $A_1 = 0.5\sin 1 - 2\cos 1 + \sin 2 - 1.5\cos 2,$ $A_2 = 1.5\sin 1 - \cos 1 + 2\sin 2 - 0.5\cos 2,$ $B_1 = 0.5\sin x - 2\cos x + \sin y - 1.5\cos y,$ $B_2 = 1.5\sin x - \cos x + 2\sin y - 0.5\cos y,$
MOP4: P_{true} disconnected, PF_{true} disconnected, 3 Pareto curves, number of decision vectors scalable	$F = (f_1(\mathbf{x}), f_2(\mathbf{x}))$ where $f_1(\mathbf{x}) = \sum_{i=1}^{n-1}(-10e^{(-0.2)\sqrt{x_1^2 + x_{i+1}^2}})$ $f_2(\mathbf{x}) = \sum_{i=1}^{n}(\lvert x_i\rvert^{0.8} + 5\sin(x_i)^3)$	$-5 \leq x_i \leq 5; i = 1, 2, 3$
MOP5: P_{true} disconnected and un-symmetric, PF_{true} connected, a 3D Pareto curve	$F = (f_1(x, y), f_2(x, y), f_3(x, y))$ where $f_1(x, y) = 0.5(x^2 + y^2) + \sin(x^2 + y^2)$ $f_2(x, y) = \frac{(3x-2y+4)^2}{8} + \frac{(x-y+1)^2}{27} + 15$ $f_3(x, y) = \frac{1}{(x^2+y^2+1)} - 1.1e^{(-x^2-y^2)}$	$-3 \leq x, y \leq 3$
MOP6: P_{true} disconnected, PF_{true} disconnected, 4 Pareto curves, number of Pareto curves scalable	$F = (f_1(x, y), f_2(x, y))$ where $f_1(x, y) = x$ $f_2(x, y) =$ $(1 + 10y)[1 - (\frac{x}{1+10y})^{\alpha} - \frac{x}{1+10y}\sin(2\pi q x)]$	$0 \leq x, y \leq 1,$ $q = 4,$ $\alpha = 2$
MOP7: P_{true} connected, PF_{true} disconnected	$F = (f_1(x, y), f_2(x, y), f_3(x, y))$ where $f_1(x, y) = \frac{(x-2)^2}{2} + \frac{(y+1)^2}{13} + 3$ $f_2(x, y) = \frac{(x+y-3)^2}{36} + \frac{(-x+y+2)^2}{8} - 17$ $f_3(x, y) = \frac{(x+2y-1)^2}{175} + \frac{(2y-x)^2}{17} - 13$	$-40 \leq x, y \leq 400$

Table 2: Multi-objective test functions

Four test metrics are suggested. The first is generational distance,

$$G = \frac{(\sum_{i=1}^{n} d_i^p)^{\frac{1}{p}}}{n} \tag{4}$$

which describes how far in objective space the Pareto front PF_{known} produced by the optimisation algorithm is from the true Pareto front PF_{true}, where n is the number of vectors in PF_{known}, $p = 2$ and d_i is the Euclidean distance in objective space between each vector and the nearest member of PF_{true}. The second is spacing,

$$S = \sqrt{\frac{1}{1-n}\sum_{i+1}^{n}(d_{mean} - d_i)^2} \tag{5}$$

which quantifies how evenly spread the members of PF_{known} are along the Pareto front. The third is overall non-dominated vector generation, ONVG

$$ONVG = |PF_{known}| \tag{6}$$

the number of solutions in PF_{known}, and the fourth is overall non-dominated vector generation ratio,

$$ONVGR = \frac{|PF_{known}|}{|PF_{true}|} \tag{7}$$

the ration of the number of solutions in PF_{known} to the number in PF_{true} at some specified resolution of the space.

Van Veldhuizen compares four multi-objective genetic algorithms: the multi-objective genetic algorithm (MOGA), MOMOGA, the niched Pareto genetic algorithm (NPGA) and non-dominated sorting genetic algorithm (NSGA). The testing procedure requires prior knowledge of the true Pareto front, which can be obtained by exhaustive searching of the space at some resolution for simple MOPs such as the test functions described, although obviously this is not practical for MOPs of any significant difficulty. Van Veldhuizen implements a binary-valued algorithm and defines the computational grid used in both the exhaustive search and the MOGAs such that 2^{24} points in the space are tested. No solution cost specific termination criterion for the algorithms is implemented: instead the search is terminated when the number of cost function evaluations exceeds 2^{16}, such that 0.39% of the space is searched. The exhaustive search is implemented on a high powered parallel architecture. The MOGAs are each run ten times on each MOP, and averages calculated for the four test metrics described above.

Due to lack of available processor time, the same test procedure could not be implemented for the algorithm described here. The exhaustive search was performed at a resolution of 0.001 on all coordinates of the MOPs, resulting in a considerably coarser representation of the space than Van Veldhuizen used. However, the algorithm itself does not restrict solutions to lie on a grid, and so in theory an infinite number of points in the space are available for investigation. Since the algorithm uses a computational resolution finer than the exhaustive search, a minimum bound exists on the generational distance for each MOP, and the test procedure therefore provides a lower limit estimate of the performance of the algorithm. This limits the conclusions that can be drawn from the results: it is possible to say that the algorithm has failed to find the true Pareto front to the accuracy used in the exhaustive search within the number of cost function evaluations imposed, but not that it has improved upon it, or upon the results demonstrated by Van Veldhuizen. Nevertheless, this remains sufficient to demonstrate the feasibility and promise of the algorithm.

Although the generation distance and ONVG provide a good indication of the knowledge of the Pareto front generated by the algorithm, the spacing and ONVGR may be less useful. These relate to the requirement for equal spacing of the solutions along the Pareto front, and each of the four algorithms tested by Van Veldhuizen incorporates fitness sharing in an attempt to enforce even spacing of the solutions. For the reasons outlined above, the algorithm presented here uses no such mechanism. The spatial accuracy variable used to filter the extracted set was set to 0.00001 for all MOPs, a level at which it was found to have no significant effect.

4 Results

4.1 Single-Objective Testing

In order to demonstrate that the inclusion of the Voronoi mating restriction enabled the algorithm to preserve diversity in the general population indefinitely, it was tested on the six-hump camelback function, with a population size of 100, an extraction age of 5, and the default mutation probabilities of, using both random selection of individuals for mating, and Voronoi restricted mating. The spatial positions of the solutions in the general population at each generation for the first 25 generations for both cases are shown in Figs. 3 and 4 respectively. Clearly in the first case both global optima are found, but one global optimum is eliminated from the population after around 15 generations and never recovered. The population collapses onto one of the global optima (which one is random), such that the majority of the individuals in the population are at exactly the same point in the space. Random immigrants are therefore quickly eliminated from the population during breeding with these highly fit individuals, such that further searching of the space is prevented.

In the second case, diversity is preserved and both global optima (and two of the local optima) persist in the general population. Whilst the search is focused in the region of the global optima, such that detailed exploration of these regions can be performed, individuals in these regions cannot breed with random immigrants in distant regions of the space, and so searching of the space continues indefinitely.

Fig. 5 shows the positions of the members of the extracted set after 25 generations with the parameters described above, plotted on a 2D contour plot of the function. The two global minima are found and the two lower-cost local minima are found to within machine precision within 25 generations, all four being captured in a single run. The extracted set never contained more than four entries, due to the check implemented to ensure that multiple solutions in the same local minium were discarded. The remaining two, high-cost local minima are not found, as

so much of the function has lower costs than these regions that solutions here are eliminated in mating before they can age enough to be extracted. Such high-cost local minima may give some information about the behaviour of the function, but it is unlikely that they would ever be desired as the output of an optimisation routine, and so their absence from the extracted set was not considered as a major drawback.

4.2 Multi-Objective Testing

The multi-objective testing regime adopted copied that used by Van Veldhuizen to a large extent, so that results from the algorithm presented here could be compared to the results presented in that paper for the four GAs he studied. It should be noted that the spacing metric can only be applied to two-objective functions, since it depended on being able to define the start and end of the Pareto front in objective space, and so quantitative results are presented only for MOPs 1, 2, 3, 4 and 6. Figs. 6 to 12 show the results of the exhaustive search in cost function space for all seven MOPs, together with the result of a typical run of the algorithm for visual comparison. It can be seen that the algorithm finds a good representation of the Pareto front in all seven cases. Figs. 13 and 14 show plots of the average generational distance and ONVG for MOPs 1, 2, 3, 4 and 6, averaged over ten runs of the algorithm on each, together with results reproduced from Van Veldhuizen. The comments made in the method should be borne in mind at this point: due to differences in the testing methodology adopted here, it is safe to conclude that the algorithm has achieved the same performance as the MOPs presented by Van Veldhuizen, but not safe to conclude that it has surpassed them. Although there are differences between the performance of the five GAs presented over the range of cost MOPs, it can be seen that in general the algorithm presented here performs as well as Van Veldhuizens MOMGA, and considerably better than NPGA, NSGA and MOGA.

5 Conclusions

The algorithm presented here was developed to meet a number of specified objectives. The primary goal was to produce a multi-objective genetic algorithm that would output multiple solutions spanning the Pareto front of the cost function, and was applicable to as broad a range of cost functions as possible. We have shown through the use of probability theory that, in the absence of the specific knowledge required to combine multiple objectives into a single objective measure, the identification of the Pareto front is the only satisfactory output from an optimisation algorithm. In the case of non-commensurate objectives there is no guarantee that such a combination strategy even exists but, assuming that it did, we have shown that optimisation of the resultant single objective would only serve to identify a specific point on the Pareto front identified by the multi-objective optimisation. A secondary goal was to reduce to a minimum the number of control parameters, so as to avoid any need to tailor these parameters to the optimisation problem at hand. Finally, an additional goal was to understand the search strategies employed by the algorithm in order to gain insights into maximising the efficiency of the search, such as details of chromosome construction.

Much of the research into evolutionary algorithms focuses on maintaining diversity in the population. Generally, if all individuals in the population are allowed to interact, the population will collapse onto a single global optimum. The approaches designed to prevent this can be broadly divided into two types. Diversity may be enforced, for instance in the choice of individuals at the replacement stage: approaches such as fitness sharing fall into this category. Alternatively, diversity may be preserved by restricting the range of interactions amongst the population, for instance by restricted mating. In either case, additional control parameters may be required, which in turn may need to be optimised in a problem-specific fashion. In addition, these approaches may slow convergence, since they limit the number of solutions that are allowed to explore any given region of the space.

In more general terms, an optimisation algorithm is required to simultaneously fulfil diametrically opposed objectives: searching and convergence. Striking a balance between these objectives is implicitly linked to the problem of diversity preservation. The approach taken here was to split these two behaviours across two populations: the primary population, which searches the space indefinitely, and the secondary population, which maintains the best current estimate of the Pareto front and so converges. In order to prevent convergence in the primary population, long-range interactions were specifically forbidden, both in the breeding and replacement stages, using mating restriction and deterministic crowding. However, in order to avoid increasing the number of control parameters needed, mating was restricted to occur only between Voronoi neighbours in parameter space. Thus the natural definition of "global" vs. "local" imposed by the algorithm was exploited. These measures alone would slow but not prevent the onset of degeneracy in the primary population, as solutions converged on local minima until the space between them was depleted. Then the local minima would become Voronoi neighbours, allowing breeding between them and so, eventually, collapse of the whole population to a single point. Therefore, solutions that had located a local minimum, identified as those that repeatedly failed to breed and produce children that replaced them,

were extracted to the second population and replaced with random immigrants. This prevented optimal solutions from dominating the population, and introduced enough randomness that searching continued indefinitely without limiting the search behaviour of the individual solutions. The use of extraction/random immigration also conferred a second benefit, in that it eliminated population size as a control parameter, allowing the algorithm to exploit an infinite population over infinite time. To further eliminate control parameters, the parameters controlling random aspects of the algorithm, such as mutation probabilities, were also placed under the control of the algorithm as further objectives to be optimised.

The algorithm has been tested on a number of real-valued test problems for the test suite collated by Van Veldhuizen, allowing comparison with the four algorithms he tested. In general, over the range of test problems and on the three metrics he used (generational distance, spacing, and ONVG), the algorithm at least competes with MOMGA, and is superior to NPGA, NSGA and MOGA. We therefore conclude that, in addition to the advantages outlined above, the algorithm performs at least as well as the state-of-the-art on these simple test problems. However, it should be noted that all of these simple test problems share a common flaw. A GA has two search operators available: random mutation and crossover. Random mutation allows the algorithm to hill climb, but in theory crossover allows much more rapid convergence by forming solutions from building blocks: optimised sub-sections of the genome. Indeed, if the building blocks strategy is not available as a search strategy, there seems little point in using a GA, as no improvements can be made over the performance of other hill climbing algorithms. Therefore, we are forced to ask what kind of problems are amenable to solution in this building block fashion, and how to construct chromosome representations of solutions to these problems that allow the strategy to be applied.

In order that a problem can be solved by the building blocks strategy, it must be possible to optimise sub-sections of the genome independent of the rest of the genome, and combine such sub-sections to form fitter genomes. The definition of independence requires careful treatment, as there are several possible forms of independence. Given a single objective a_i, a function of a set of n parameters c_i, portions of the genome are independent if the cost function can be expressed as a sum over these portions

$$a_i = f(\bar{c}) = \sum_{c_{p,q}} f(c_{p,q}) \tag{8}$$

where the parameters $c_{p,q}$ are some subset (p to q)of the parameters c_{1-n} and the sum is performed over all such subsets. These parameter subsets are commonly referred to in the GA literature as schemata, and "building blocks" are commonly defined as highly fit, low-order, short-length schemata. Schemata correspond to hyperplanes in the search space.

Given that the ability to combine building blocks is a prerequisite of efficient GA use, some conclusions can be drawn concerning the problems to which GAs can be efficiently applied. Clearly, the simple test problems used in this paper, and more generally in the GA literature, do not possess this property and so are not effective tests of GA efficiency, beyond indicating the ability to hill-climb. More realistic test problems would include the Travelling Salesman Problem [13]. This involves planning the shortest path that visits a set of cities exactly once and returns to the starting point. Depending on the geographical layout of the cities, it may be possible to optimise small portions of the path independent of the remainder of the cities. The brain atrophy analysis described in the introduction also possesses this property: each pair of points defines a decision boundary in the brain, and it may be possible to optimise a single decision boundary (possibly by matching its position to some physical structure, such as a boundary between lobes) independent of the remaining boundaries. Note that this does not imply that the problem can be solved by applying optimisation techniques to each portion of the genome in turn. Portions of the genome may still be weakly coupled: in the brain atrophy analysis, the points at which the decision boundaries intersect will be governed by more than two points. In addition, the most efficient division of the genome into schemata may not be available, and the GA can explore multiple possibilities.

We can now draw some conclusions on the types of problems that GAs can solve, the search strategies they can use, and how to most efficiently construct a genome. Firstly, in order for recombination to act efficiently, the problem must be solvable by the combination of building blocks. This requires at least partial independence of schemata within the genome, and in turn dictates how the genome should be constructed (and how the crossover operator should be designed). Optimisation of such schemata implies detection of a highly fit hyperplane within the search space, representing optimisation of a sub-set of parameters. Combination of such schemata then represents detection of the crossover points of the hyperplanes to detect global optima.

It has been stated in the past that the combination of mating restriction and deterministic crowding, as used in the algorithm presented here, would prevent certain types of recombination events from occurring [14]. Specifically, in a deceptive and massively where the only feasible way to locate the deceptive global peak was by combination of schemata from the local optima, and furthermore there were less fit regions between the local and global

optima, the presence of solutions in these less fit regions would prohibit the desired recombination events via the mating restriction. However, the arguments outlined above demonstrate that such problems are somewhat artificial. If the problem is amenable to GA optimisation and the genome is constructed efficiently, the search space will contain local optimal hyperplanes that intersect at global optimal points. The GA will then hill climb or randomly search on schemata to find the hyperplanes, and follow them to detect the global optima. In that case, if the definition of Voronoi neighbours is selected with care, mating will occur between the solutions climbing along different hyperplanes that are closest to the intersection points, and thus the intersection points will be located by recombination. We conclude that the combination of mating restriction with deterministic crowding would not hinder the convergence to global optima on realistic multi-objective functions where the objectives were independent. This is an issue that can be addressed in chromosome construction and objective definition, if only to enhance the possibility of creating building blocks, and therefore the problems typified by deceptive functions should not have undue influence on the design of algorithms. However, it is recognised that additional testing on realistic test problems, such as the Travelling Salesman Problem, will be required to elucidate this point.

References

[1] T Back. Optimal mutation rates in genetic search. In S Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 2–8. Morgan Kaufmann Publishers, 1993.

[2] P A Bromiley, N A Thacker, and P Courtney. Colour image segmentation by non-parametric density estimation in colour space. In T Cootes and C Taylor, editors, *Proceedings of the British Machine Vision Conference*, pages 283–292. British Machine Vision Association, 2001.

[3] C A C Coello. List of references on evolutionary multiobjective optimisation, 1999. Available at: http://www.lania.mx/EMOO.

[4] L C W Dixon and G P Szego. *Towards Global Optimisation 2*. North Holland, Amsterdam, 1978.

[5] L Davis (Ed.). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[6] C M Fonseca and P J Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[7] M P Fourman. Compaction of symbolic layout using genetic algorithms. In J J Grefensette, editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 141–153, Pittsburg, PA, July 1985. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey.

[8] D E Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[9] D E Goldberg, K Deb, and J H Clark. Genetic algorithms, noise, and the sizing of populations. Technical Report IlliGAL Report No. 91090, Illinois Genetic Algorithms Laboratory, http://www-illigal.ge.uiuc.edu, 191.

[10] D E Goldberg, K Deb, and J H Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362, 1992.

[11] D E Goldberg and J Richardson. Genetic algorithms with sharing for multimodal function optimisation. In J J Grefensette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey., 1987.

[12] S P Larcombe. *Implementing Heterogeneous Systems in a Three-Dimensional Packaging Technology*. PhD thesis, Department of Electronic and Electrical Engineering, University of Sheffield, 1996.

[13] P Larranaga, C M H Kuijpers, R H Murga, I Inza, and S Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13:129–170, 1999.

[14] S W Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, Illinois Genetic Algorithms Laboratory, Department of General Engineering, University of Illinois at Urbana-Champaign, 1995.

[15] I Ono, H Kita, and S Kobayashi. A robust real-coded genetic algorithm using unimodal normal distribution crossover augmented by uniform crossover: Effects of self-adaptation of crossover probabilities. In W Banzhaf, J Daida, A E Eiben, M H Garzon, V Honavar, M Jakiela, and R E Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 496–503. Morgan Kaufmann Publishers, 1999.

[16] W H Press, S A Teukolosky, W T Vetterling, and B P Flannery. *Numerical Recipes in C: The Art of Scientific Computing (Second Edition)*. Cambridge University Press, Cambridge, U.K., 1992.

[17] H H Rosenbrock. An automatic method for finding the greatest or least value of a function. *Comput. J.*, 3:175–184, 1960.

[18] J D Schaffer. Multiple objective optimisation with vector evaluated genetic algorithms. In J J Grefensette, editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 93–100, Pittsburg, PA, July 1985. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey.

[19] J R Schott. Fault tolerant design using single and multicriteria genetic algorithm optimisation. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1995.

[20] N Srinivas and K Deb. Multiobjective optimisation using nondominated sortinfg in genetic algorithms. *Evolutionary Computation*, 2(3):221–284, 1994.

[21] K C Tan, T H Lee, and E F Khor. Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimisation. *IEEE Transactions on Evolutionary Computation*, 5(6):565–588, 2001.

[22] N A Thacker, A J Lacey, P Courtney, and G S Rees. An empirical design methodology for the construction of machine vision systems. Technical Report TINA Memo No. 2002-005, Neuroimage Analysis Center (NiAC), http://www.tina-vision.net, 2002.

[23] N A Thacker, A R Varma, D Bathgate, J S Snowden, D Neary, and A Jackson. Quantification of the distribution of cerebral atrophy in dementing diseases. In *Proceeding MIUA*, pages 61–64, London, July 2000. BMVA.

[24] D A Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses and New Innovations*. PhD thesis, Graduate School of Engineering, Air Force Institute of Technology, 1999.

[25] D A Van Veldhuizen and G B Lamont. Multi-objective evolutionary algorithm research: A history and analysis. Technical Report Technical Report TR-98-03, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH 45433-7765, 1998.

[26] D A Van Veldhuizen and G B Lamont. On measuring multiobjective evolutionary algorithm performance. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pages 204–211, 2000.

[27] G Voronoi. Recherches sur les paralléloèdres primitives. *J. reine angew. Math.*, 134:198–287, 1908.

[28] D H Wolpert and L Thiele. No free lunch theorems for optimisation. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

6 Figures

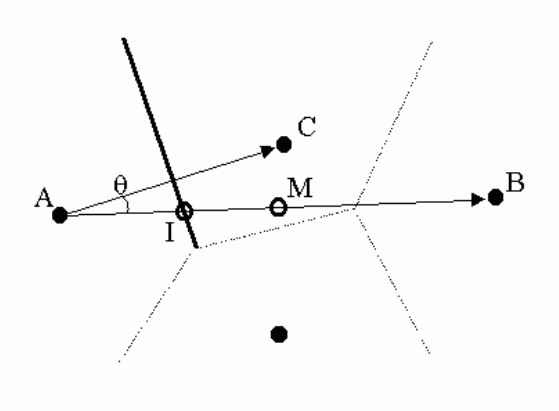


Figure 1: The derivation of the Voronoi cell crossing constraint for mating restriction. The solid points are knot points and the dashed lines are the cell boundaries. The darker line is the boundary crossed by the allowed step **AC**. The constraint tests that the intersection I of this plane with the step **AB** is at less than half the length of **AB**, to the mid-point M.

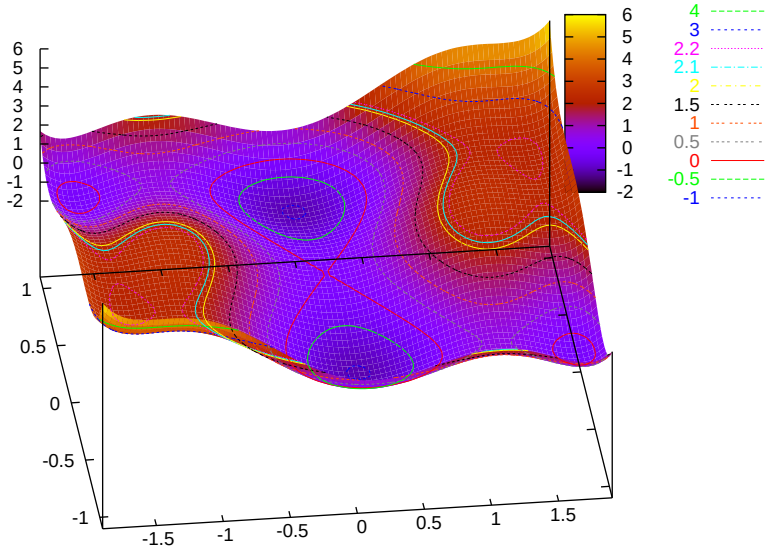


Figure 2: The six-hump camelback function.

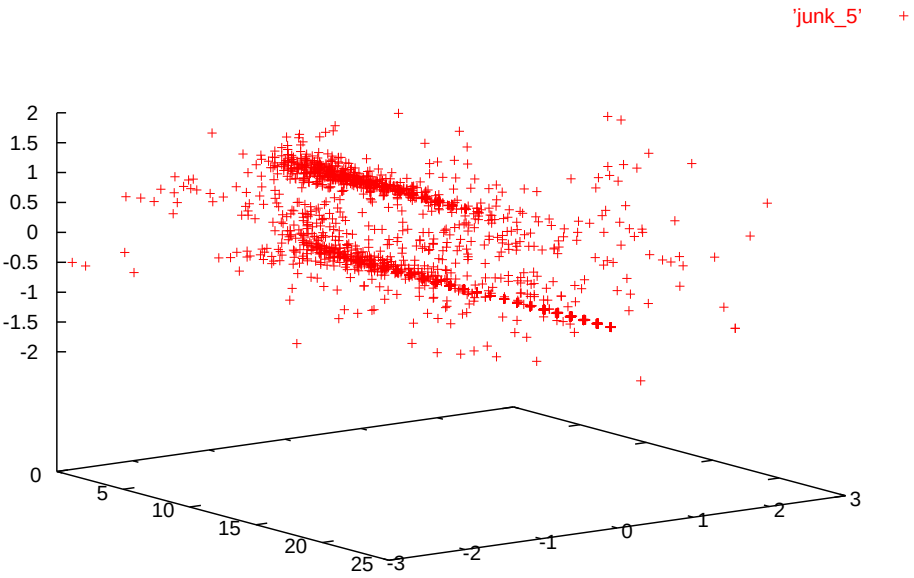


Figure 3: The general population during the first 25 generations of optimisation on the six hump camelback function, for a population size of 100 and extraction age of 5, with random selection for breeding.

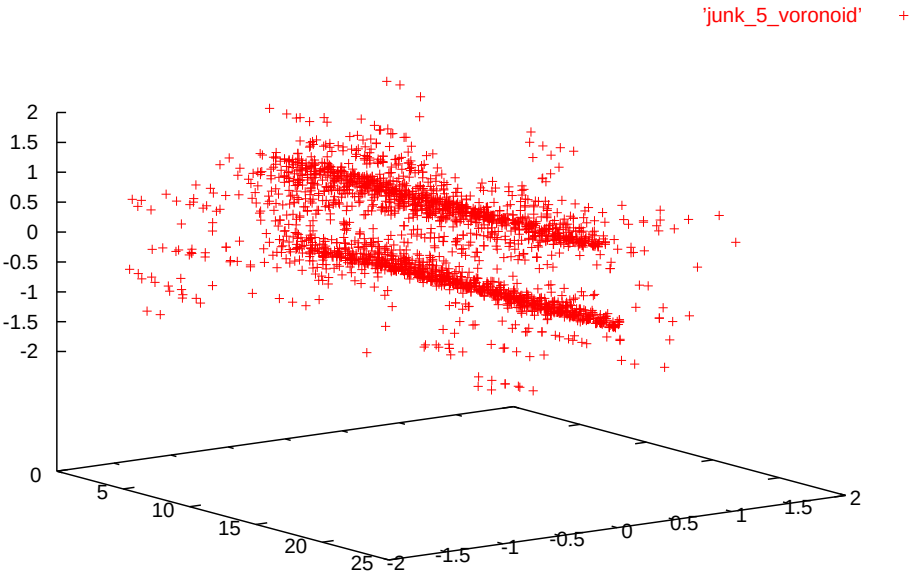


Figure 4: The general population during the first 25 generations of optimisation on the six hump camelback function, for a population size of 100 and extraction age of 5, with Voronoi selection for breeding.

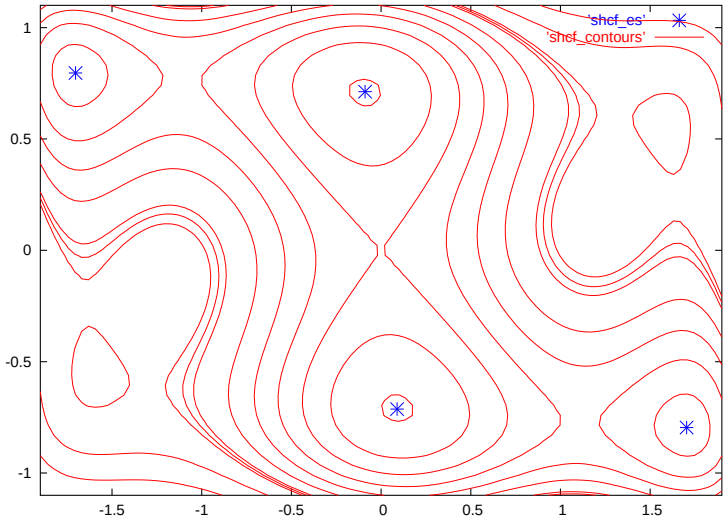


Figure 5: 2D projection of the six-hump camelback function, showing the extracted set solutions found by the GA after 25 generations with a population of 100, extraction age of 5, and default mutation probabilities. Note that the 2 high-cost local optima are not entered into the extracted set.

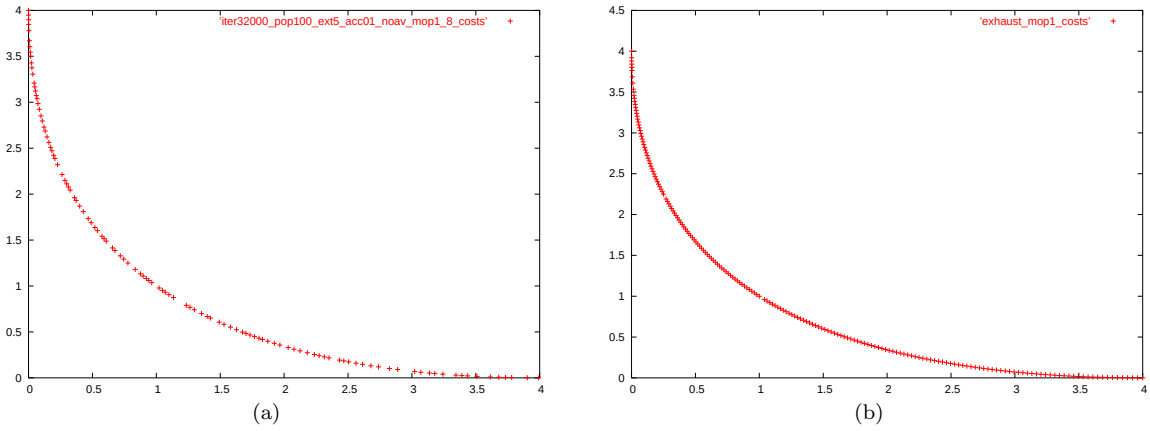


Figure 6: Sample output from algorithm (a) and exhaustive search results (b) for MOP1.

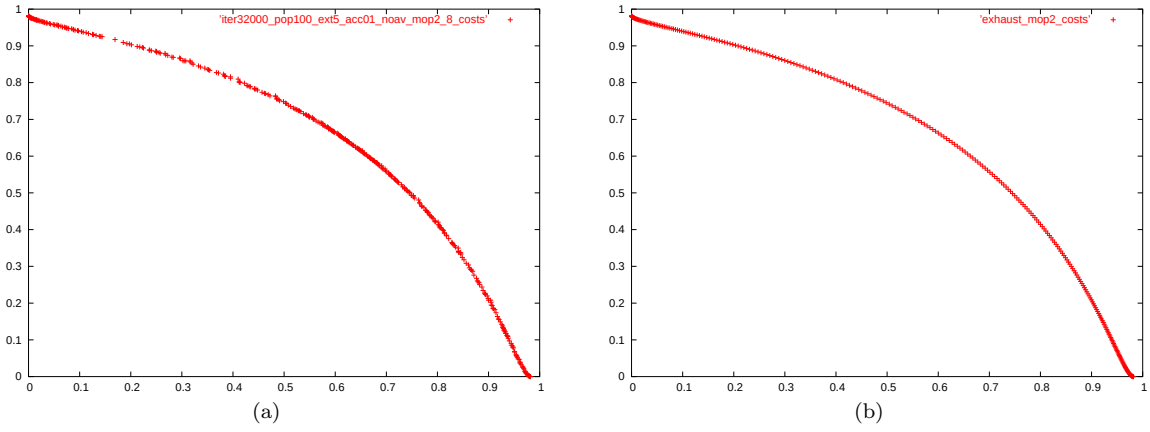
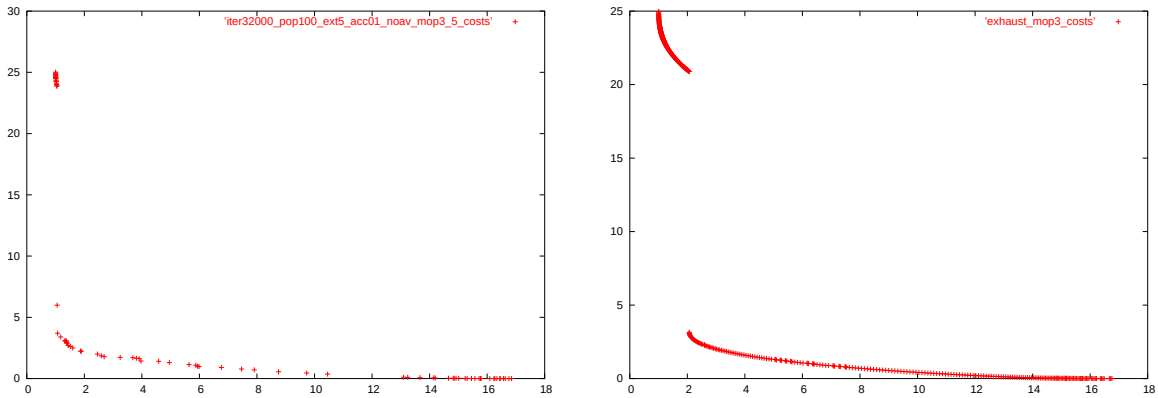
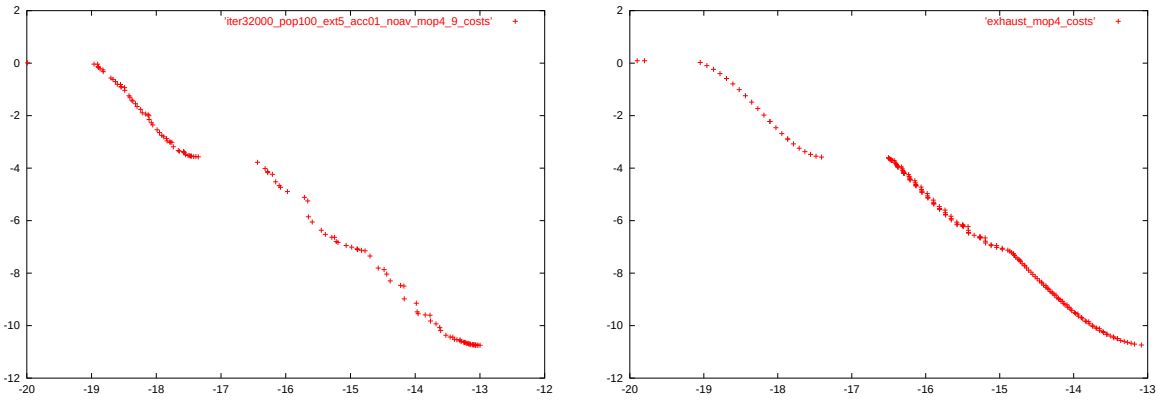


Figure 7: Sample output from algorithm (a) and exhaustive search results (b) for MOP2.



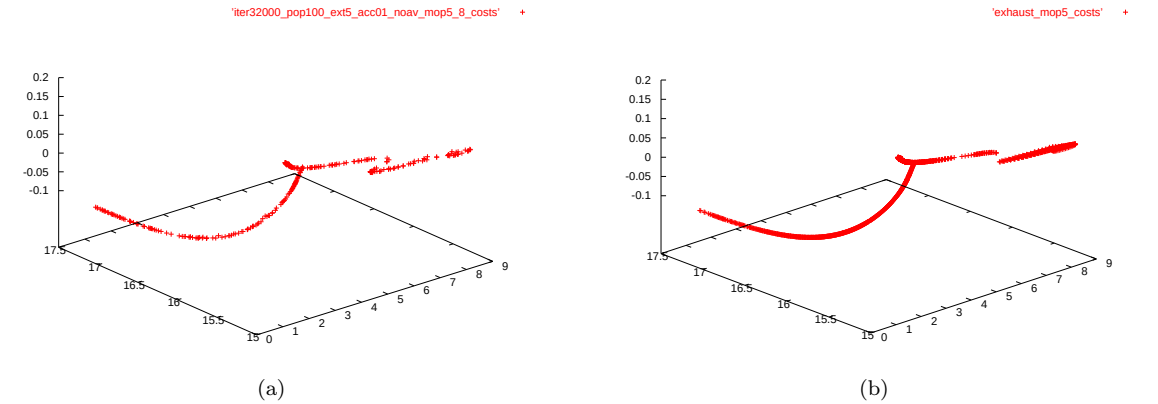
(a) (b)

Figure 8: Sample output from algorithm (a) and exhaustive search results (b) for MOP3.



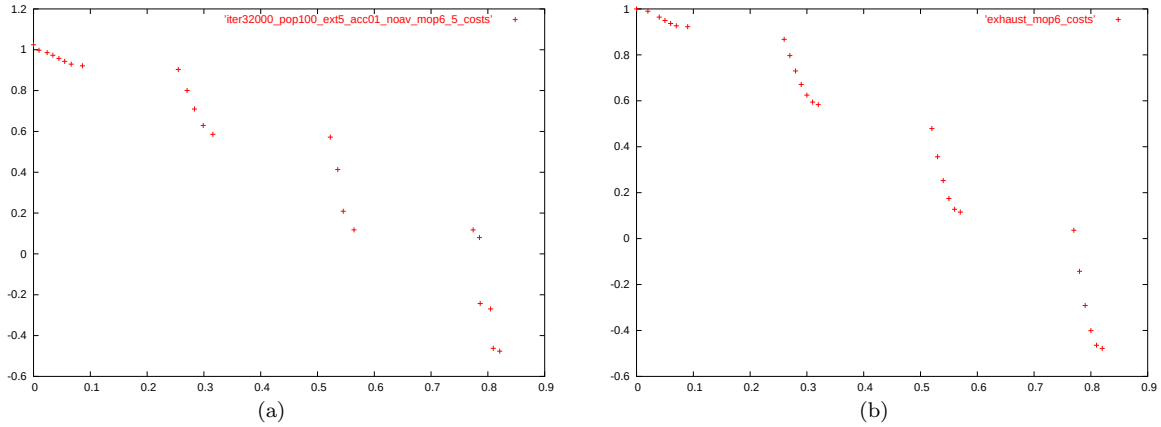
(a) (b)

Figure 9: Sample output from algorithm (a) and exhaustive search results (b) for MOP4.



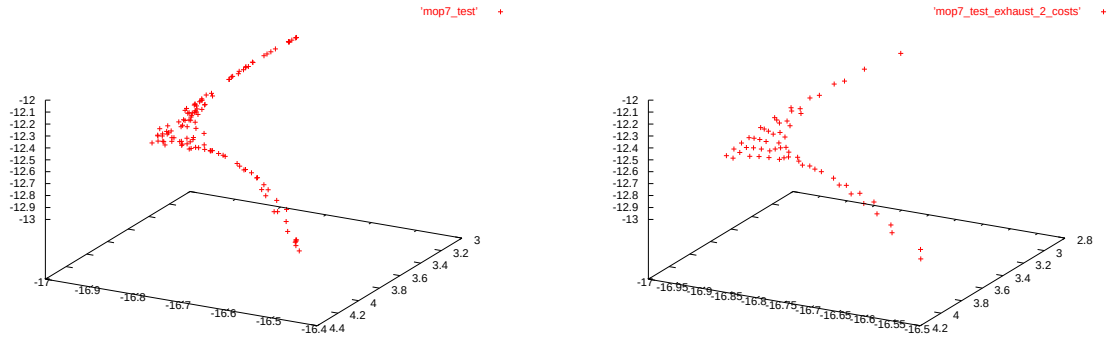
(a) (b)

Figure 10: Sample output from algorithm (a) and exhaustive search results (b) for MOP5.



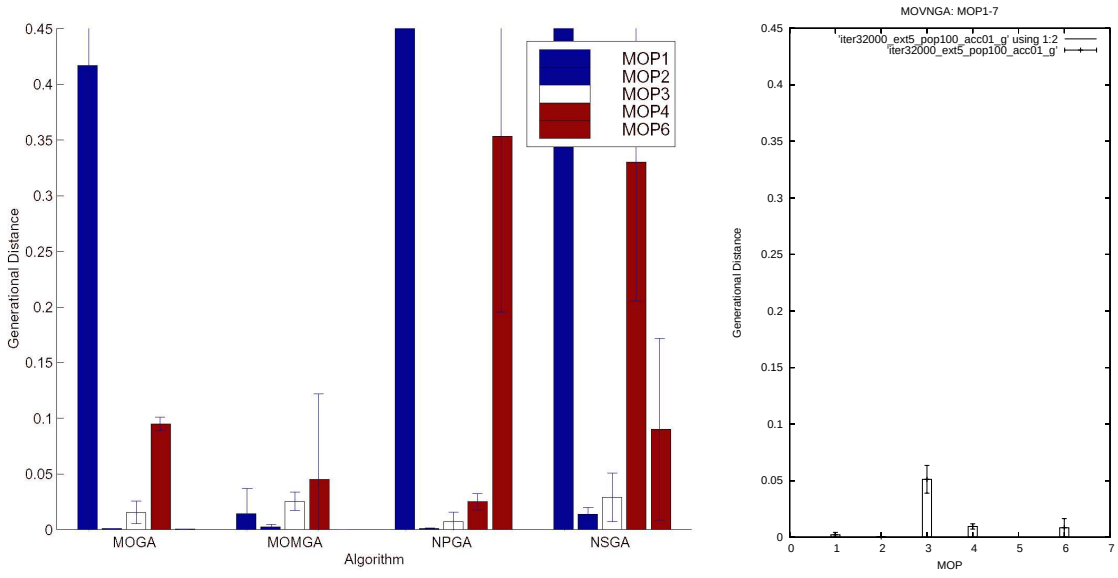
(a) (b)

Figure 11: Sample output from algorithm (a) and exhaustive search results (b) for MOP6.



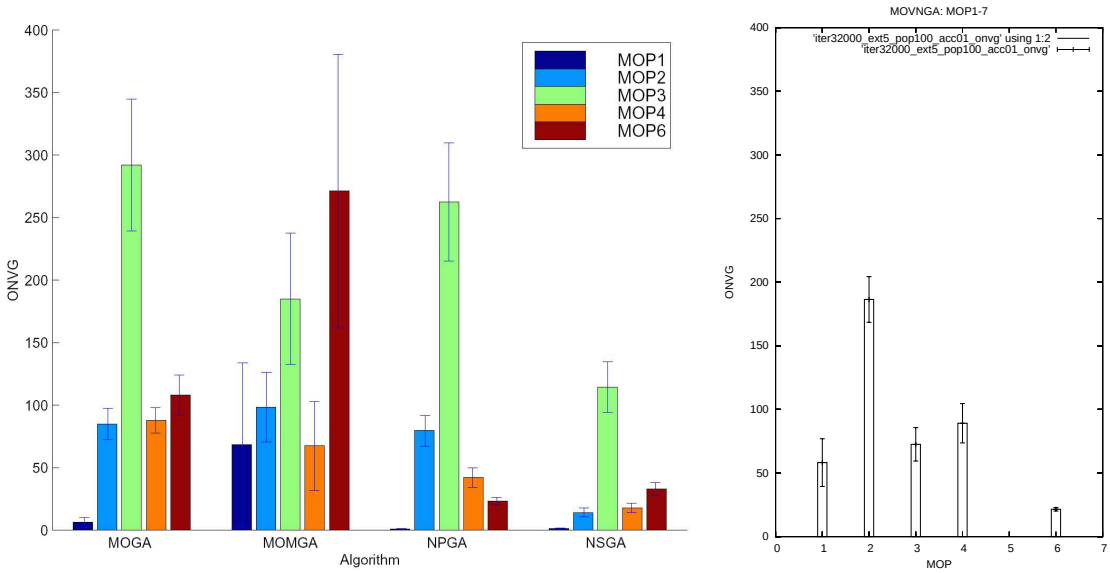
(a) (b)

Figure 12: Sample output from algorithm (a) and exhaustive search results (b) for MOP7.



(a) (b)

Figure 13: Generational distance (lower is better) for the four MOEAs tested by Van Veldhuizen (a) and for MOVNGA (b).



(a) (b)

Figure 14: ONVG (higher is better) for the four MOEAs tested by Van Veldhuizen (a) and for MOVNGA (b).