Advanced Multi-objective Evolutionary Algorithms Applied to two Problems in Telecommunications

Joshua Knowles[1], Martin Oates[2], and David Corne[1]
[1]Department of Computer Science, University of Reading, UK
[2]BT Research Labs, Adastral Park, Martlesham Heath, UK
J.D.Knowles@reading.ac.uk, martin.oates@bt.com, D.W.Corne@reading.ac.uk

Abstract

The bulk of research in optimization is aimed at single objective problems, where the aim is to find a solution which maximises or minimises a single quality measure. However, as in nature, many problems in telecommunications are fundamentally multi-objective, particularly where the issues involved are related to quality of service, or cost/reliability tradeoffs, or perhaps both. The results of optimization should ideally provide the network designer or service manager with a wide and high-quality spread of potential solutions which are uniformly spread across the Pareto tradeoff frontier. There has been considerable research in multi-objective optimization, but, until recently, the most prominently known multi-objective optimization algorithms have tended to be rather slow, and there has been no universally accepted way to properly compare the performance of different methods. In this paper, we describe two evolutionary computation based multi-objective optimization methods which have recently been shown to be both considerably faster than the classical set of such methods, and to outperform existing methods on a wide range of test problems, where performance comparisons are done using a sophisticated technique recently extended and developed by the authors. We focus on two application areas in telecommunications: the adaptive distributed database management problem, and the offline-routing problem. The new and classical evolutionary multi-objective optimizers are compared on variants of both problems. The speed and quality of these new methods suggest that their adoption in live applications of these and other telecommunications related problems is feasible.

1 Introduction

Evolutionary computation is well known for providing a family of naturally-inspired algorithms which can successfully address a wide range of optimization and machine learning problems (Holland, 1975; Goldberg, 1989; Fogel, 1995; Back, 1996). The bulk of research in evolutionary optimization (indeed all optimization techniques), however, tends to focus on single-objective problems, in which the aim is to find a solution which optimises a single scalar quantity. In contrast, many problems in both nature and the real world fundamentally *multiobjective*. Rather than finding the best value for a single objective, the goal is to find a collection of solutions which occupy different parts of a tradeoff surface involving two or more objectives.

For some time now, the ability of evolutionary algorithms to capably address *multiobjective* optimization problems has been known, thanks to seminal work by Srinivas and Deb (1994), Horn and Nafpliotis (1994) and others. However, estimation of the relative quality of evolutionary techniques compared with classical methods on multiobjective problems has been hampered by two issues. First, the seminal multiobjective evolutionary algorithms (MOEAs) have tended to be considered rather slow and cumbersome compared with the classical approaches to multiobjective optimization (local search or branch and bound techniques based on mixtures of single-objective weighted measures). As a result, MOEAs have generally been considered a research issue, rather than a competitive approach. Secondly, there has been little in the way of methods which are capable of providing convincing statistical comparisons between rival multiobjective optimizers.

Both of these issues have now (arguably) been successfully addressed. In particular, recent work supported by British Telecommunications PlC has led to the development of two MOEAs which have been found to convincingly outperform the classical MOEAs, and rival methods from the local search and operations research communities, while also being suitably speedy. Also, this work has developed, extended and implemented some powerful ideas for comparing multiobjective optimizers which were originally set out by Fonseca and Fleming (1996).

In this paper, we report on the use of these new MOEAs in the context of two distinct multiobjective problems arising in the telecommunications arena. For each of these two problems, we compare the new MOEAs with classical MOEAs, and/or with a rival modern MOEA developed elsewhere. In particular, we describe and use two algorithms developed with BT support called PAES (Pareto Archived Evolution Strategy) and M-PAES (Multiobjective PAES). These compare highly favourably with the rival techniques, and we are able to conclude that fast, effective multiobjective optimisation can be feasibly used in live applications of these and other problems in telecommunications.

In section 2 we describe the two problems addressed in this paper. The first is the 'offline routing' problem. This has previously been addressed as a single objective optimisation problem, in which the three objectives are combined into a single scalar value. Here we treat it directly as a three objective problem and show how a near-optimal tradeoff surface can be quickly developed with modern MOEA techniques. The second problem is the Adaptive Distributed Database Mangement problem; agai, this is a problem that has previously been treated as a single-objective problem in which two or more raw quality of service measures are combined into a single value. We show here how its treatment as a multiobjective problem can be successfully pursued without undue loss of speed, and with consequent gain in decision quality. In section 3 we briefly review multiobjective evolutionary search, presenting pseudocode for the modern techniques used in experiments later described. The last part of this section briefly describes the analysis technique we use for comparing multiobjective optimizers. Section 4 details experiments involving PAES and M-PAES, and then summarises results on the two problems, and we conclude in section 5.

2 Multiobjective Issues in Telecommunications

There are many optimisation issues in telecommunications which, for various reasons, are best seen as multiobjective. We therefore need to find a set of solutions which approximate an optimal Pareto tradeoff surface (see section 3), rather than a single near-optimal solution. We also need to find this set of solutions quickly. We concentrate here on two problems in particular which are sensibly viewed as multiobjective.

2.1 The Offline Routing Problem

In a traditional telephone network, when a connection is requested the network attempts to assign a circuit of fixed bandwidth between the source and destination. If a circuit is available it is assigned exclusively to the connection for the duration of the call. Otherwise, the caller will receive an engaged signal and access to the network is denied. This system, called circuit switching ensures that quality of service is guaranteed for every call that is connected since each one is effectively on its own circuit. However, the disadvantage of circuit switching is that it is potentially very wasteful of network resources because much of the time circuits may not be transmitting any information (for instance in the pauses between words in ordinary speech). A potentially more efficient system, called packet switching, works by splitting a stream of information into small packets. Each packet contains a small header indicating its destination. Now, each packet can be routed to its destination along cables which it is sharing with many other calls. This 'multiplexing' of information means that more calls can be accepted onto the network because each call takes up less space on average than in a circuit switched scheme. However, there are several disadvantages associated with packet switching which arise from the way in which the packets are switched through the network. When a packet arrives at a switch its header is read and the packet is placed on one of the output lines of the switch. The process of reading the header and moving the packet to the correct output line takes a finite amount of time. Thus, if two packets arrive at a switch simultaneously one of them must be placed in a buffer where it waits until the switch is free to process it. Buffering of packets means that variable delay (jitter) is introduced in the flow of information from a source to its destination. It also opens up the possibility for packets to be lost completely as the buffers have a finite capacity and can become overloaded. These problems can be minimised by routing calls in such a way as to reduce the congestion that they encounter. Routing algorithms for packet switched networks are thus concerned with minimising the mean delay of packets through the network, the variance in the delay (jitter), and the probability of packet loss due to over-utilisation of links.

Routing of calls may be undertaken on different timescales: Dynamic, distributed routing is concerned with the real-time routing of connections that are requested on a moment by moment basis. In modern models of real-time routing each node uses local congestion information to update self-contained routing tables and uses these to select the next node with which to connect (Munetomo et al, 1997). Off-line routing works on a slower timescale, and would be used where large amounts of data are being transferred. In this domain it has advantages for both the network provider and the user. Because connections are booked in advance by users, the network provider knows the traffic profile and can find a tailored, near-optimal set of paths for the traffic. This means that the quality of service can be guaranteed and the network provider can maximise the profit it makes because fewer calls have to be turned away. Off-line routing is currently used in the synchronous digital hierarchy (SDH) networks (Tanenbaum, 1996) for data transfer, and it may become one means of routing in future asynchronous transfer mode (ATM) networks (Clark, 1996).

There will always be a requirement for dynamic routing but the advantages of off-line routing ensure that it will also have a useful part to play in network management systems of the future. How useful it becomes may depend on the timescale with which good routing solutions can be found. The faster that a set of near-optimal paths can be found, the more information will be available to customers booking connections. Network providers stand to gain from fast optimisation because they could then load up the network as much as possible, maximising the profit that they make, whilst still guaranteeing quality of service.

We adopt the problem definition used by Mann and Smith (1996) with only minor modifications, as follows. We are given a telecommunications network over which we must route multiple traffic requests in such a manner as to achieve a feasible routing assignment, i.e. no link is over-capacitated (hard constraint). This is the primary objective. In addition, we impose a secondary objective that link utilisations should all be below a specified, fixed target utilisation. Finally, the routing assignment attempts to minimise the communications costs, assuming that costs are associated with the use of each link.

Specifically, we are given a network $G = (N, E)$, where N is the set of n nodes and E is the set of m edges. Associated with each edge $e \in E$ is a bandwidth, $b(e)$, and a cost, $c(e)$. The network is always bidirectional, there is only a single type of traffic, n is in the range 30 to 60 and m is between 40 and 150.

The bandwidths, $\{b(e)|e \in E\}$, lie in $\{16, 64\}$, that is there are two link types, a 'backbone' type of capacity 64 units and a 'local' type of capacity 16 units. For each distinct $v, w \in N$, there is an amount of network traffic, $t(v, w)$, which must be routed from v to w in the network. This traffic must all be routed on the same path $P(v, w)$, which has to be determined. The total traffic, $f(e)$, on any edge $e \in E$ is given by

$$f(e) = \sum_{v,w \in N} \{t(v, w) \mid e \in P(v, w)\} \tag{1}$$

This must not exceed the bandwidth of that edge, i.e.

$$for\ all\ e \in E, f(e) \leq b(e) \tag{2}$$

This can be achieved by minimising the deviation $f(e) - b(e)$, so long as $f(e) > b(e)$, i.e.

$$minimise \sum_{e \in E} max\{f(e) - b(e), 0\} \tag{3}$$

A second objective is to find a minimum cost allocation of traffic through the network, satisfying constraint (3). The cost of routing all traffic, $t(v, w)$, on the path, $P(v, w)$, between v and w is given by

$$t(v, w) \times \sum_{e \in P(v,w)} c(e) \tag{4}$$

This is summed over all possible source/destination pairs to yield the second objective

$$minimise \sum_{v,w \in N} \{t(v, w) \times \sum_{e \in P(v,w)} c(e)\} \tag{5}$$

bearing in mind the bidirectional constraint so we are summing only over $n(n-1)/2$ possible pairs.

The third and final objective specified is to minimise the deviation from a target utilisation, u , for each link in the network, i.e.

$$minimise \sum_{e \in E} max \left\{ f(e) - \frac{u \times b(e)}{100}, 0 \right\} \tag{6}$$

Thus, so long as $f(e) > (u \times b(e)/100)$, for at least one link $e \in E$, there will exist some pressure on the optimisation process to find a more balanced solution. For our experiments we set $u = 50$.

When approached as a single objective problem, The full and final objective is therefore:

$$\begin{aligned} minimise \qquad & p \times \sum_{e \in E} max\{f(e) - b(e), 0\} \\ & + \ w_1 \times \sum_{v,w \in N} \{t(v, w) \times \sum_{e \in P(v,w)} c(e)\} \\ & + \ w_2 \times \sum_{e \in E} max \left\{ f(e) - \frac{u \times b(e)}{100}, 0 \right\} \end{aligned} \tag{7}$$

where, p, w_1 and w_2 are weights applied to the various terms in the evaluation function. These can be adjusted as necessary to prioritise any of the objectives in a single objective approach. For example, Mann and Smith (1996) report that through extensive testing they found values of $p = 10$, $w_1 = 1$ and $w_2 = 5$ gave well balanced, feasible routing strategies. By viewing the problem as multiobjective, however, we are freed from any need to prioritise these objectives, and can demand that our optimizer deliver a broad range of solutions spanning an optimal (or near optimal) tradeoff surface defined by these objectives.

2.2 The Adaptive Distributed Database Management Problem

The Adaptive Distributed Database Management problem (ADDMP) is a problem faced by distributed database service providers (DDSPs), such as video-on-demand, genome databanks, and so forth. Oates and Corne (1998) gives a succinct description, while Oates and Corne (2000) provides considerable detail. Also, C source code for the evaluation function can be found via the first author's website[1]. Here, we provide a brief description and some basic details of the ADDMP, aimed at furnishing the reader with an appreciation of its multiobjective nature.

A DDSP offers a database service to a collection of clients. Both the database itself and the clients will typically be globally distributed. That is, the database will be mirrored and/or distributed over a number of servers on different parts of the globe, and these servers will be accessed by a globally distributed client base. For convenience in problem formulation, we tend to speak of this situation as a network of 'nodes', where each node may be a client, a server, or both. The typical interpretation of a node is that it represents the entry point to a LAN (or WAN, depending on the scale of ADDMP instance concerned), which contains a server provided by the DSSP, one or more clients, or both.

The DDSP needs to regularly ensure that database users (clients) are receiving adequate quality of service (QoS). Indeed, clients' subscription to the database may involve guarantees from the DDSP of distinct levels of QoS, perhaps varying with subscription cost. A key factor in QoS is the delay (or response time) experienced by a client for a typical database query. In maximising QoS, the DDSP aims to minimize the delay for each client. The DSSP is able to attempt to control this by from time to time redirecting particular clients accesses to different servers. Clearly, the aim of maximising QoS for all occurs in the context of load balancing. That is, we may be able to minize the delays experienced by certain clients by routing their queries to the fastest server which contains the required data; however, the extra load on this server will degrade the delays. So, the optimal solution will involve a careful balancing of clients across servers.

The ADDMP is hence the problem of finding the best client/server connection configuration, given a particular scenario which specifies details of the underlying communications network, server speeds, and access rates for each client. What counts as 'best' depends on many things, but a single-objective QoS measure will typically involve combining the *worst* client delay with the mean or median delays. However, such QoS measures are growing increasingly inadequate as distributed database service provision becomes more widespread and complex as regards the range of service gaurantees on offer. For example, consider two potential solutions to a 5-client ADDMP in which the vectors of client delays (in milliseconds) are, respectively: Solution 1 (155, 130, 140, 140, 140), Solution 2 (350, 80, 90, 90, 90).

In solution 1, the worst delay is 155 ms, and the mean delay is 141 ms. In solution 2, the worst delay is far worse than in solution 1 at 350 ms, but the the mean delay is slightly better than solution 1 at 140 ms. Also, the median delay is considerably better in solution 2 than in solution 1. In a single-objective approach, which of solution 1 or 2 is preferred depends very much on the relative weightings given to the worst and mean (or median) components. It is hence complex, and perhaps impossible, to derive 'correct' relative weighting for these components, especially considering the widely different kinds of ADDMP scenarios which exist.

A multiobjective approach therefore seems more sensible and flexible. Client 1, for example, may have paid for QoS guarantee which indicates that their delay will always be below 200 ms. Client 2, on the other hand, may have been given a guarantee that their delay would be always within 20time. With varied sets of factors like this, the task of an optimizer addressing an ADDMP would be to quickly produce a good and diverse spread of solution configurations, leaving it to a later decision process to then choose from these on the basis of the various QoS guarantees in operation for the clients currently using the service.

The problem we address in this paper is therefore that of quickly providing a good set of diverse ADDMP configurations, from which a second decision-making process can then choose the best according to prevailing QoS issues.

3 Evolutionary Multiobjective Search

The general (unconstrained) multiobjective optimization problem can be expressed as:

$$\begin{array}{rl} \text{"minimize"} & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_k(\mathbf{x})) \\ \text{subject to} & \mathbf{x} \in X \end{array} \tag{8}$$

where $\mathbf{x}$ is represents a solution, and X is a finite set of feasible solutions. The objective function $\mathbf{f}(\mathbf{x})$ maps X into $\Re^k$, where $k \geq 2$ is the number of objectives. The term minimize appears in quotation marks because, in general, there does not exist a single solution that is minimal on all objectives. Therefore, one may seek to find a set of solutions $X^* \subseteq X$, called the Pareto optimal set, with the property that :

$$\begin{array}{c} \forall x^* \in X^* \not\exists\, x \in X \text{ such that } x \succ x^* \\ \text{where } x \succ x^* \text{ iff } \forall i \in \{1, ..., k\}\ f(x_i) \leq f(x_i^*) \ \wedge\ \exists i \in \{1, ..., k\} : f(x_i) < f(x_i^*) \end{array} \tag{9}$$

in which $x \succ x^*$ is read as x *dominates* x^*, and solutions in the Pareto optimal set are also known as efficient or admissible solutions. For example, as we saw above in the ADDMP, the first objective might be worst delay,

[1] http://www.reading.ac.uk/~ssr97jdk

and the second might be mean delay. Very typically, there is no solution with objective values (x, y) such that all other solutions are either equal or worse on both objectives. Rather, there will be a wide collection of diverse solutions such that no one dominates another in the set. this collection is the Pareto tradeoff surface.

Many single objective combinatorial optimization problems that can be solved in polynomial time, become $\mathcal{NP}$-hard when formulated as corresponding multiobjective problems (Ulungu and Teghem, 1994). This makes the need for good approximate methods for multiobjective optimisation especially important.

In recent years, there has already been growing interest in good approximate methods for such problems. The approaches taken so far can be roughly split into two kinds: local search methods, including tabu search and simulated annealing, e.g. Czyżak and Jaszkiewicz (1998), Gandibleux et al (1996), Hansen (1996), and evolutionary algorithm based methods, such as Srinivas and Deb (1994), Horn and Nafpliotis (1994), Horn et al (1994), and more recently Zitzler and Thiele (1999) and Knowles and Corne (2000).

In particular, a recent clutch of algorithms developed in the evolutionary multiobjective search community have been found both to improve significantly in speed and quality over the seminal such methods, and have also been seen to outperform recent techniques from the local search multiobjective community. These algorithms are the Strength Pareto Evolutionary Algorithm (SPEA) developed by Zitzler and coworkers (Zitzler, 1999; Zitzler and Thiele, 1999), the Pareto Archived Evolution Strategy (PAES), and the Memetic Pareto Archived Evolution Strategy (M-PAES), both of the latter developed with BT support by Knowles and co-workers (Knowles and Corne, 2000; 2000a). In the following, we describe PAES and M-PAES.

3.1 PAES

The PAES algorithm is outlined via pseudocode in Figure 1. PAES maintains a single 'current' solution, and, via essentially a straightforward hillclimbing procedure, it searches the space of solutions in an aggressive manner by continually generating and testing mutants of the current solution. Under certain circumstances, a mutant becomes the 'new' current solution and search continues from there. In the meantime, an archive is maintained which comprises a representative collection of all of the nondominated points so far found by the algorithm.

```
1    Generate initial random solution c and add it to the archive
2    Mutate c to produce m and evaluate m
3        if (c dominates m) discard m
4        else if (m dominates c)
5            replace c with m, and add m to the archive
6        else if (m is dominated by any member of the archive) discard m
7        else apply test(c,m,archive) to determine which becomes the new
             current solution and whether to add m to the archive
8    until a termination criterion has been reached, return to line 2
```

Figure 1: Pseudocode for $(1+1)$-PAES

The archived collection of nondominated points is 'representative', rather than a complete set of nondominated points found, since we necessarily need to limit its size. Hence, this brings in complications such as how to update the archive when it is already full, but we have a new non-dominated solution which could be included. Such decisions are settled by considering the uniformity of spread of solutions along the current approximation to the non-dominated Pareto surface. Basically, we prefer the archive to have a uniform spread of solutions, and hence an archive update for a new solution tends to only occur if it would increase the uniformity of this spread. Similar considerations apply when it comes to deciding whether or not a mutant of the current solution should become the new current solution. The pseudocode in Figure 2, which specifies the procedure `test` at line 7 of Figure 1, details the decision processes involved.

The crowding strategy in PAES works by forming an implicit hypergrid which divides (normalised) phenotype space into hyperboxes. In Figure 3, this is illustrated by the thick horizontal and vertical lines; the problem is two-dimensional and hence these hyperboxes are simply squares. Each chromosome in the archive is associated with a particular hyperbox in phenotype space. At points in the algorithm where information is needed about the degree of crowding in different regions of the space (such as lines 8 and 9 in Figure 2, this information is simply provided by returning the number of occupants of relevant hyperboxes. For example, in Figure 3, chromosome A is in a more crowded region of the space than chromosome B.

3.2 M-PAES

Memetic algorithms are hybridisations of evolutionary and local search which have achieved significant success recently on a wide range of important optimization problems (Moscato, 1999). M-PAES is a recently pro-

```
if the archive is not full
    add m to the archive
    if (m is in a less crowded region of the archive than c)
       accept m as the new current solution
    else maintain c as the current solution
else
    if (m is in a less crowded region of the archive than x for
     some member x on the archive)
       add m to the archive, and remove a member of the archive from
        the most crowded region
       if (m is in a less crowded region of the archive than c)
          accept m as the new current solution
       else maintain c as the current solution
    else
       if (m is in a less crowded region of the archive than c)
          accept m as the new current solution
       else maintain c as the current solution
```

Figure 2: Pseudocode for test(c,m,archive)

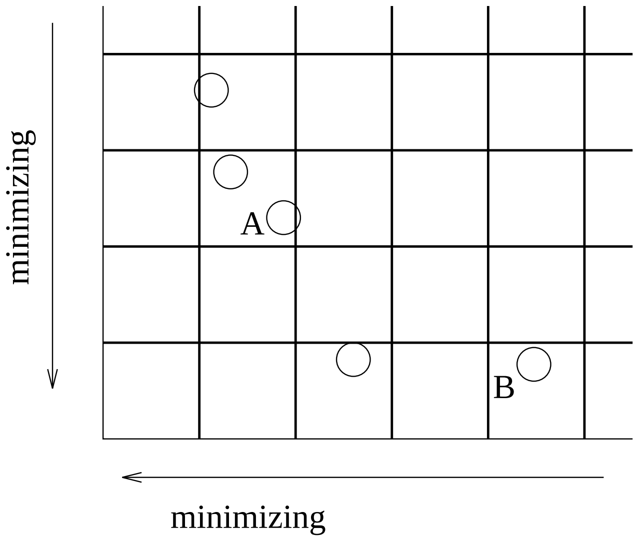

Figure 3: PAES' crowding strategy

posed memetic multiobjective algorithm, which has already been found to outperform sophisticated alternative approaches on some difficult multi-dimensional knapsack problems (Knowles and Corne, 2000a).

The M-PAES algorithm is described by the pseudocode in Figure 4. Essentially, the idea is to run the PAES algorithm $|P|$ times, once for each of a population P of initial current solutions. Following this, P is updated by replacing each chromosome in it with the final current solution from the PAES algorithm which 'improved' it. Then, a recombination phase occurs (lines 7–14 in Figure 4, in which crossovers of parents in P (and also the global archive G – see below) are sampled, yielding a further updated population P. The cycle then repeats, with PAES runs on each member of P, and so forth. Note that there are now two archives, G and H. G is the global archive, which plays the same role as the normal archive in the simple PAES algorithm, and simply maintains a representative record of all nondominated solutions found during the course of the algorithm. In M-PAES, however, we also need local archives called H. When a PAES algorithm is run as part of M-PAES, the individual PAES algorithm uses H as its archive in the normal way, but each new candidate generated during such a PAES run is also treated as a candidate for the global archive G also.

The individual runs of PAES appear in line 5 of Figure 4, and are parameterised by c, G, and H. G and H are the global and local archives, as explained above, while c is the member of P which is to be used to initialse this PAES run.

3.3 A Note on Statistical Analysis of Multiobjective Optimizers

Proper comparison of the results of two multiobjective optimisers is highly non-trivial. A recent ingenious approach was described by Fonseca and Fleming (1995), which we have adopted, implemented and extended. An

```
1     Generate and evaluate initial population of random solutions, P
2     Place each nondominated member of P into global archive G
3     Repeat the following until a termination criterion is met:
3        For each candidate solution c ∈ P
4           Initialise local archive H to contain c, and members of G which do not dominate c
5           Apply PAES(c,G,H)
6           Place c (final current solution from step 5) into P.
7        Set intermediate population I to be empty
8        Repeat until intermediate population I is full:
9           Repeat until a suitable child is found or max trials exceeded:
10             Choose two parents randomly from P ∪ G and recombine to produce c.
11             Update global archive G with c if necessary.
12          If c is not worthy of the archive G, select c by binary
            tournament selection from G.
13          place c into I
14       Update population P by overwriting it with I.
15    Return global archive G.
```

Figure 4: Pseudocode for M-PAES

appreciation of the comparison method is necessary to support interpretation of the results presented in later, so we include a concise description here.

The result of a multiobjective optimizer M is an *attainment surface* D_M, as illustrated in Figure 5. If algorithm A returned the points P_A – the set of nondominated and non-duplicated solutions found by A (p1, ..., p5 in the figure), then the attainment surface, D_A, returned by A is the (solid) lines joining the points. Notice that D_A divides the phenotype space into two regions. All points in region 1 either dominate A's results or are nondominated by them, and all points in region 2 are dominated by A's results.

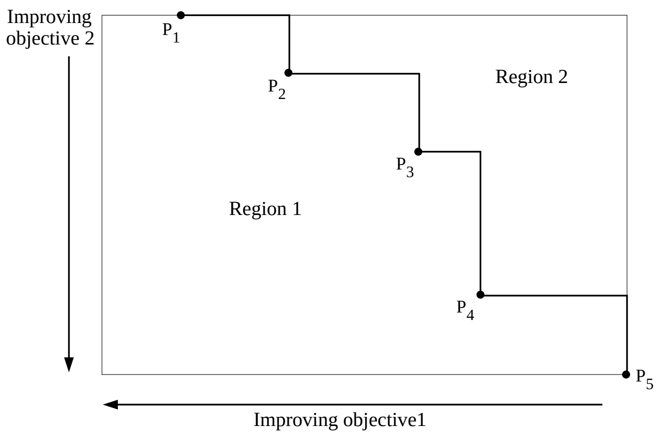

Figure 5: How a set of nondominated points divide up the phenotype space

When comparing P_A and P_B, we therefore compare the two surfaces D_A and D_B. Consider Figure 6, in which attainment surfaces are drawn for a set of results from a single trial run of A, and a single trial run of B. The statistical comparison process is engineered by implicitly drawing a collection of lines, perhaps originating from the origin, which sample the Pareto frontier in different regions. Line L_1, for example, intersects D_A at point I_1, and will intersect with D_B somewhere above the figure at a place much more distant from the origin than I_1. Line L_2 intersects D_A at I_2, and D_B at I_3; again, D_A's intersection is the closer to the origin.

With such a collection of lines, we can compare the two algorithms in different regions of the Pareto frontier by comparing the intersections on these lines. As long as we use enough equally-spread lines, a comparison of two surfaces can therefore return two numbers: the number of lines in which D_A's intersection was closer to the origin (and hence dominated D_B's intersection), and the similar number for D_B. Notice however that the number of lines is important; in Figure 6, the regions in which B is better than A (between lines L_2 and L_3, and below L_5) are not picked up by this five-line sample.

Next, notice that for (say) 20 trial runs of A and 20 trial runs of B, we will have generated 40 different surfaces. For each line, we can therefore find 40 intersections (20 from algorithm A and 20 from algorithm B). These intersections, since they are defined entirely by a distance from the origin on that line (or of course we

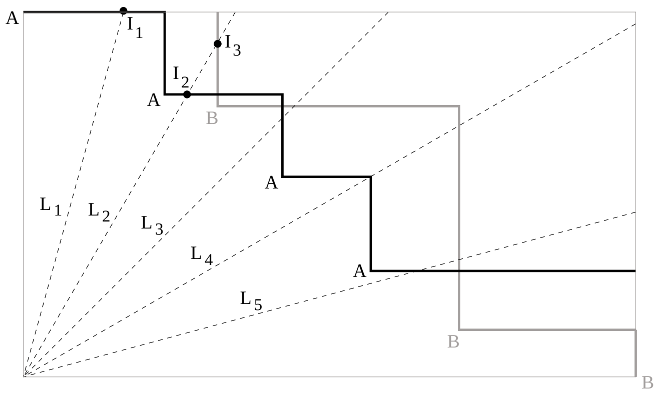

Figure 6: Sampling the Pareto frontier using lines of intersection

can simply take the x co-ordinate of the intersection) provide a univariate distribution. A statistical test on this distribution of 40 intersections can then provide, in the usual way, a verdict as to which algorithm did better in the region of space represented by that line, backed up by a confidence measure.

We find that a good way to present the results of a comparison is in the form of a pair [a,b], where a gives the percentage of the space (ie: the percentage of lines) on which algorithm A was found statistically superior to B, and b gives the similar percentage for algorithm B. Typically, if both A and B are 'good', $a+b < 100$. the result $100-(a+b)$ then of course gives the percentage of the space on which results were statistically inconclusive. We present all of our results in this form. In some cases we also present the results by plotting the *median* dividing surfaces for each algorithm. This is very useful for illustrating the results, and is done by simply having the comparison code return each algorithm's median intersection point for each line. This is what Fonseca and Fleming do, but we note that, unlike the simple paring [a,b], plotting median surfaces cannot easily be done for problems of 4 or more dimensions.

4 Approach and Experiments

4.1 Representations and Operators for the Offline Routing Problem

Off-line routing involves finding a set of routes for a given network/traffic combination such that communications costs and congestion are minimised. A chromosome must therefore somehow represent the complete route through the network for each of a set of calls defined by the problem data. That is, if there are c calls, the chromosome must somehow specify c paths through the network, one for each call.

As done by Mann and Smith (1996), our approach is to pre-computea number of paths for each call, so that the chromosome only needs a single gene (an integer) to represent the choice of path for a call. The gene for call i, for example, will be an index into a table of pre-computed paths for call i.

Mann and Smith (1996) pre-computed the K shortest paths for each call, where path-length is identified with total path cost (ie: the 'length' of a link, for the purpose of the shortest-path calculations, is its cost). We do the same, with $K = 15$, pre-calculating them using an algorithm due to Yen (1971). In addition, however, pre-compute a separate sets of K-shortest path tables where cost is based on a heuristic estimate of the congestion that would be caused by the path in question. We hence pre-compute two tables of 15 paths for each call. Although there may be some overlap and similarity between these tables, in general they may contain very different paths.

As reported in Knowles and Corne (2000), various methods were experimented with for making use of these pre-computed path tables. The best technique was found to be the use of a 'double' chromosome for the representation, and biased operators for crossover and mutation, which we now describe.

We use a 'double' chromosome structure, in which the first chromosome epresents the choice of path for each call in terms of providing an index into a path-table. The second chromosome then indicates *which table* is used for the corresponding gene in the first chromosome. For example, the following chromosome would be valid for a 25-call problem:

(1,1,2,3,1,3,1,3,5,1,1,2,1,1,1,1,2,4,1,3,1,1,8,1,2)
(0,0,1,1,0,1,0,0,1,1,1,0,0,1,0,1,0,0,1,0,1,1,0,1,1)

The top chromosome indicates the path index, and the bottom chromosome indicates the table index. So, if table 0 contains paths ordered by cost, and table 1 contains paths ordered by the congestion heuristic. this chromosome indicates that the path chosen for call 1 is the best path in its cost-cordered list, the path chosen for call 2 is also the best path in its cost-ordered lsit, the path chosen for call 3 is the 2nd-best path in its congestion-ordered list, and so on.

Both chromosomes are subject to random initialisation, mutation and recombination. Thereby, effectively employing analogues of the concepts of dominance and multiploidy in natural genetics, a representation of sorts will be evolved along with the solution itself. That is, over time, the genes in the second chromosome will fixate, indicating, for example, that for certain calls it is best to choose a path from the cost-ordered list, and for others it is best to choose from the congestion-ordered list.

It is worth noting that 'good' solutions may be expected to contain many alleles with the value 1 in the path-index chromosome, indicating in many places the choice of 'local' best path, either according to cost or congestion, for individual calls. This intuition is employed in a simple scheme for biasing the mutation and crossover operators, as well as the population initialisation scheme. To this end, gene values are initialised randomly according to an exponential distribution with the shape in figure 7.

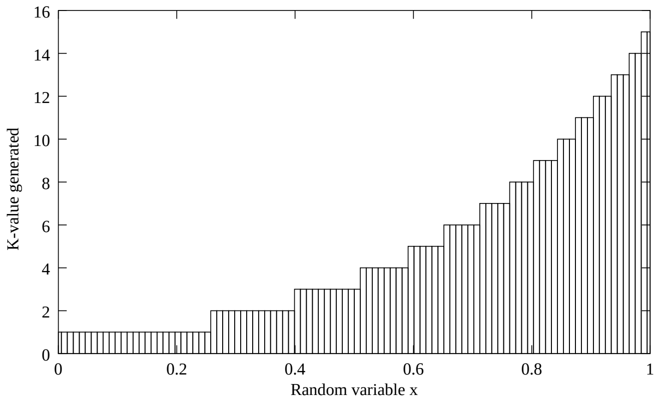

Figure 7: The Exponential function used in the initialisation and mutation of genes in the offline routing problem.

The function illustrated in Figure 7 introduces a bias towards generating low allele values. In implementation, the distribution is altered along the length of the chromosome, by linking the tuning parameter p to the position in the chromosome, such that it is more strongly biased for the genes representing high bandwidth calls and less so for low bandwidth calls.

Finally, Figure 8 shows our crossover technique for this problem, which also makes use of the biasing function.

The crossover operator selects two parents via binary tournament solution, and then considers their 'reduced surrogate' (ie: the collection of loci at which the parents have different allele values). Then, using tournament selection among the genes in the reduced surrogates, a locus then chosen whose allele in at least one of the parents is relatively high. The child is then formed by replacing the lower valued allele at this locus with the higher one, with that child otherwise being the same as the parent which contained the lower-valued allele. In this way, recombination concentrates on propogating the 'interesting' building blocks. That is, locally poor choices of best-cost or best-congestion path which have been found to work very well in combination with locally good choices elsewhere.

4.2 Representation and Operators for the ADDMP

In stark contrast to the case of the offline routingproblem, we have found a simple, direct representation and corresponding simple, standard operators to be suitable for the ADDMP. Given an ADDMP involving n nodes (each of which can be either a client or server or both), the chromosome is a string of n genes, where the alleles of a gene range through the possible servers to which a client can be connected. In all the cases investigated later, each node is considered to be both a client and a server, and each of n genes can therefore take any of n values independently. Standard uniform crossover is used, and the mutation function picks a gene at random, and changes it to a random new value.

We will also briefly discuss the evaluation function for the ADDMP. This needs to take into consideration several factors, including the basic performance of each node, the effect of loading a particular node with transactions from several clients, the effect of combining both retrievals and updates from clients, the need to perform multiple updates and the delays imposed by inter-node communications overheads. The evaluation method we use is based on Edwards' model (Edwards, 1997), developed at British Telecom Labs. Whilst the model is by no means exact, incorporating a number of intentional generalisations and assumptions, it appears sufficiently refined to address the issues with which we are concerned, specifically allowing customisation of the network

Figure 8: Selection of a Gene from *Parent 2* to Generate Offspring from *Parent 1*.

Parent 1	1 2 1 1 1 3 1 5 1 1 1 2 1 4 1 1 1 2
Parent 2	1 2 **4** 1 **2** 3 1 **1** 1 1 **9** 2 1 **7** 1 1 1 **1**
Reduced Surrogate	4 2 1 9 7 1
Tournament Select	4 $\boxed{2}$ 1 $\boxed{9}$ $\boxed{7}$ 1
Winners	9 7
Break Ties	9
Offspring	1 2 1 1 1 3 1 5 1 1 **9** 2 1 4 1 1 1 2

topography, individual node performance, communications costs, contention rates and individual client loading profiles.

The key to this model is to make good estimates of the 'effective transaction rate' of any particular server based on the raw connection speed data and various aspects of the load on that server. Space restrictions make it impossible to fully describe the evaluation function here, but full details (in particular, executables for evaluating candidate solutions for the 10 node/10 server problem considered here) will be made available at the ECTELNET[2] WWW site, or can be obtained from the authors.

4.3 Experiments: Offline Routing

To generate test problems for offline routing we use a random network generator which is based on a suboptimal degree-constrained minimum spanning tree construction algorithm due to Narula and Ho (1980). We use two edge bandwidths, a backbone spanning tree of edges which have a bandwidth of 64 units, through which traffic from distal nodes can be routed, and a set of local edges of bandwidth 16 units. The algorithm begins by generating B points in a two-dimensional space: these are the nodes in the backbone network. The Euclidean distance between each pair is calculated and a degree 4 minimum length spanning tree is generated, each edge being assigned a bandwidth of 64. Next, a further $N - B$ points are generated to be the remaining network nodes. The d-MST algorithm continues until all of these nodes are connected, using edges of bandwidth 16. At this point, we have a tree network which is close to minimising the delay/cost of the edges and which meets the degree constraint. Now, extra edges are added to give alternate routes between node pairs. This adds reliability and allows the load in the network to be spread thus reducing congestion, our original aim. The extra edges are added between pairs of nodes which currently have a connectivity of only one or two in a way which ensures that we continue to be parsimonious as regards edge length (delay/cost).

In order to ensure that the points making up the original backbone network are spread evenly in the plane and span the width of it, and the further points fall between fairly evenly we use a sub-random sequence generator, usually finding use in Monte Carlo simulation, due to Halton (see Press et al, 1988). The nodes in our 'random' networks are not placed truly randomly in the plane but by changing the initialisation conditions of the sequence we can generate different networks which 'appear' random. Figure 9 shows one of the 45 Node Networks generated using the above method. The larger circles represent nodes on the backbone, and are connected by the bold edges, representing the 64 unit bandwidth links. The smaller circles represent the remaining nodes, and are connected by feint edges representing the 16 unit bandwidth links.

To generate the traffic matrix simply requires randomly choosing source-destination pairs from the $n(n-1)/2$ that exist, and assigning them a bandwidth requirement, also randomly. We use bandwidth requirements in the range $(1, 4)$.

For the multiobjective experiments, a network of size $n = 45$ was generated, and three different traffic scenarios were generated for this network. The traffic matrices were all challenging to an extent, but the first one

[2] ECTELNET is the Telecommunications subgroup of the European Network of Excellence in Evolutionary Computation, at http://www.dcs.napier.ac.uk/evonet/ .

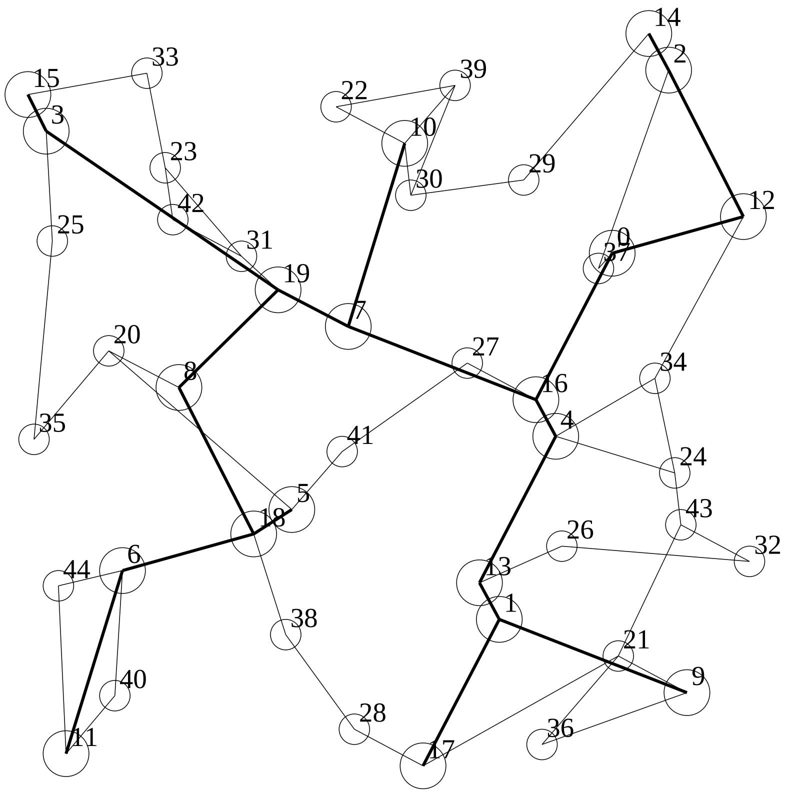

Figure 9: A 45 Node Network.

PAES vs. NPEA		
scenario 1	scenario 2	scenario 3
[82.7, 1.3]	[23.6, 59.0]	[58.2, 24.6]

Table 1: The routing problem with three traffic scenarios

has slightly less traffic than the second, and the third has the most traffic, sometimes more than can be routed without violating the bandwidth constraint of at least one link.

To investigate the performance of multiobjective optimizers on the offline routing problem, we compared PAES with the Niched Pareto Evolutionary Algorithm (Horn et al, 1994; Horn and Nafpliotis, 1994). Comparison of a wider range of methods on this problem was rather confounded by the fact that various specializations were required to the code to achieve the pre-processing and specialised representation and operators employed. For the offline routing problem, this has only been done so far for NPEA and for PAES. NPEA is a classic evolutionary multiobjective algorithm, which is improved in our implementation by the employment of a steady-state reproduction strategy.

Results are presented for three different traffic scenarios over the 45 node network. PAES, has only two parameters which must be set. These are the archive size, here set to 100, and the number of divisions of the objective dimension used in the hypergrid crowding method (ie: the number of lines per dimension in the grid illustrated by Figure 3), set here to 32.

The NPEA algorithm uses tournament selection and a comparison set as described in Horn and Napfliotis (1994). It is optimized via preliminary experiments to have a population size of 150, tournament size of 20, comparison set size of 70 and a σ_{share} niching parameter of 0.01, where the range of each objective is scaled to lie between 0 and 1. It uses steady state regeneration with a replace-worst replacement strategy.

In Table 1 we show the raw results when comparing NPEA and PAES on the 45-node offline routing problem with three distinct traffic scenarios.

As can be seen, PAES conclusively outperforms NPEA on almost all of the space on scenario 1, and over half of the space on scenario 3, however NPEA seems best on scenario 2. In further analysis of the results, we concentrated on visualising the surfaces generated, especially for scenario 2.

We use three different methods of visualising the surfaces generated by these algorithms and discuss the merits and drawbacks of each. First, all the points from 20 runs of each algorithm separately are collected together and all of the nondominated points are plotted. This gives us the combined nondominated surface for each algorithm. This was done for scenario 2, and the plots are shown in Figure 10. An important lesson can be learned here. Although, by using our statistical technique, we know that the NPEA has superior performance overall, combining the distributions shows that it was PAES which on the combination of twenty runs found the best solutions. This underlines the danger of blindly using either of these techniques in isolation.

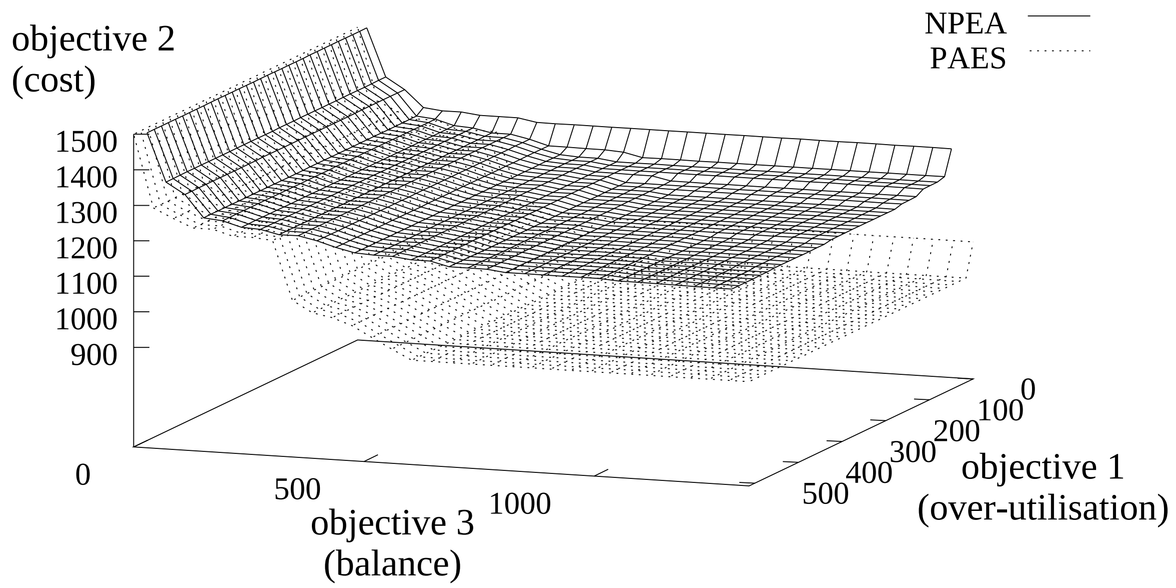

Figure 10: Best combined attainment surfaces obtained on scenario 2

The second method employed uses the twenty surfaces generated from the twenty runs of each algorithm. As in our statistical method, j intersecting lines are then drawn through these surfaces, giving j distributions, for each algorithm. The median of these distributions was then plotted for each algorithm. Three views of the resultant plots are shown in Figure 11, again for scenario 2. These plots are more in agreement with the results of the statistical tests presented in Table 1. The plotted NPEA surface lies closer to the origin than the PAES surfaces in the majority of points, although PAES clearly covers more of the phenotype space.

Finally, the surface generated by a single run of each algorithm is shown in Figure 12. This technique is not as powerful as the previous two described but in this case it does shed further light on the statistical test results presented in Table 1. The plot shows that, again, PAES covers more of the phenotype space, spreading solutions across a large range on all the objectives but that a large portion of the NPEA surface lies below the surface generated by PAES.

From these results it is difficult to be conclusive about which of these two algorithms is "better" on the off-line routing task, although the indicators generally favour PAES. Certainly, in terms of speed, PAES is far superior to NPEA (Knowles and Corne, 2000). It seems that PAES is at least competitive on two of the traffic scenarios considered, while its performance when the traffic load was lightest seems to be clearly superior.

4.4 Experiments: The ADDMP

ADDMP instances can occur in great variety. The numbers of clients and servers can range typically between 2 and 20, and the number of servers between 10 and several thousand. Database access patterns can vary equally dramatically. Eg, access to share price and similar financial databases may be very frequent with constantly changing global activity, and hence re-optimization of client/server access configurations may need to occur every few minutes. In other scenarios, re-optimization may only need to occur every few hours involving a small number of clients.

In this paper we look at ADDMP scenarios involving 10, 20, and 40 clients, and in each case we consider 5 separate problems which reflect possible changes in access patterns over time. Therefore, our comparisons of algorithms are in the context of a wide but representative range of potential instances. We are interested particularly in ADDMPs which need constant, and hence fast, re-optimization. Ie, results must arrive quickly. Hence, in increasing order of the problem sizes, the maximum allowed number of evaluations is 500, 2000 and 5000.

We consider both 2-objective and 3-objective versions of each problem. In the 2-objective version, the objectives are the worst delay figure and the median delay figure. In the 3-objective version, the objectives are the

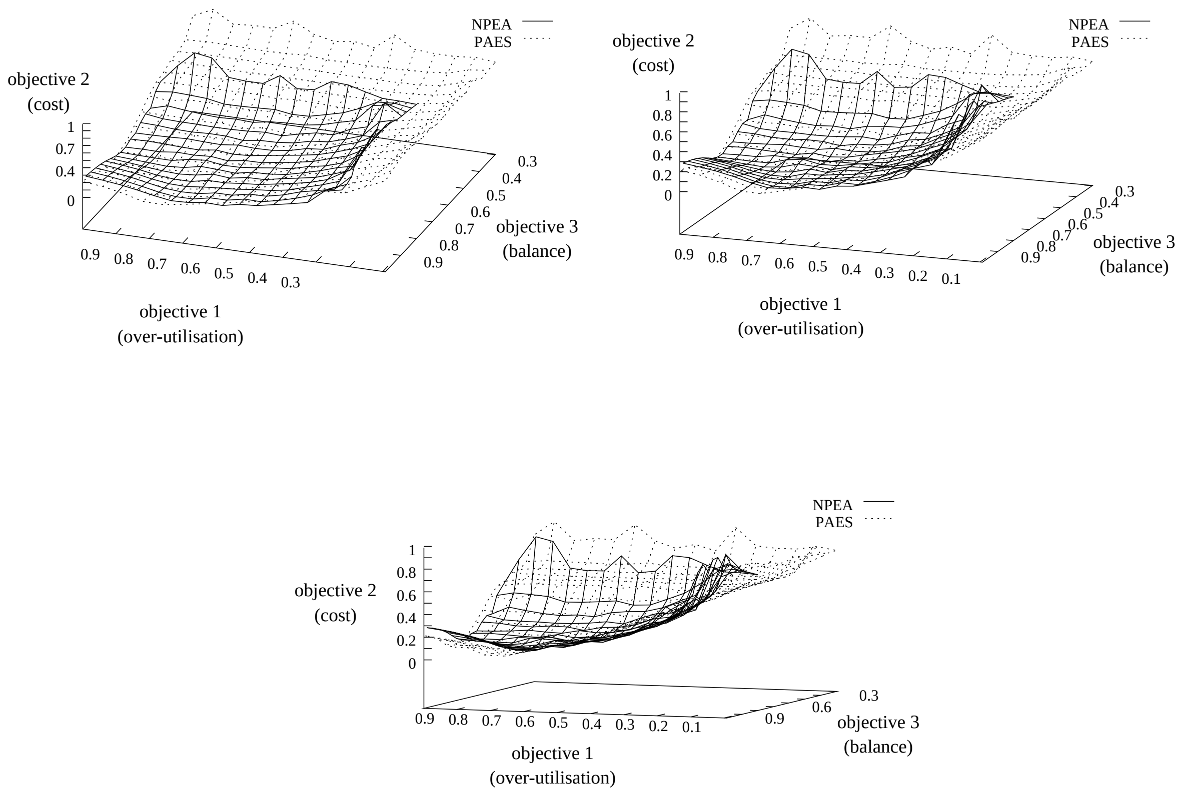

Figure 11: Median attainment surfaces obtained on scenario 2

90% quartile delay figure (ie: 90% of clients will have a better delay figure than this), the 80% quartile delay figure, and the median figure.

We also note that the fitness function can be expected to break down (and hence give inaccurate estimates of delays) for configurations which highly overload certain servers. We deal with this by building in a conservative cutoff of 1000 ms; ie: when the model estimates more than 1000ms for the worst client delay, we return the result '1000' for all the objectives rather, forcing the point to be rejected. In this way, we avoid troubling the Pareto frontier with inaccurate phenotypes

Or experiments on this problem used PAES and M-PAES, and compared both of these with the Strength pareto Evolutionary Algorithm (Zitzler and Thiele, 1999), which is perhaps the most competent rival to PAES and M-PAES yet described in the evolutionary multiobjective algorithm literature.

The following parameter settings were fixed throughout the trials: crossover type, crossover rates and the per gene mutation rate. Other parameters settings were investigated on an ad hoc basis. With SPEA, which uses both an 'internal' and an 'external' population (for details see Zitzler (1999), the size of the internal population was altered for different problems. With M-PAES, the two parameters controlling the length of the local search phases were altered, also to give good performance on different problems. With PAES, no changes in parameter settings were necessary, with the exception that the number of hypergrid subdivisions per dimension was different

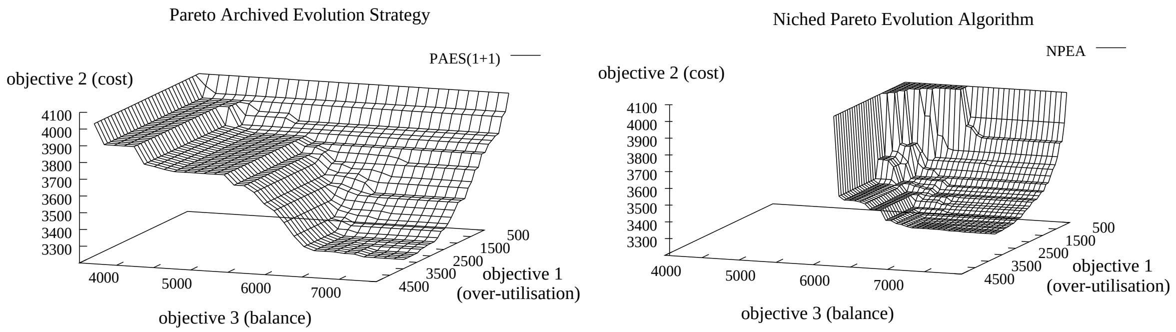

Figure 12: Nondominated surfaces obtained from a single run on scenario 3

for the 2-objective and 3-objective problems. Table 13 summarises the parameter settings.

algorithm	p_c	crossover type	p_m	mutate type	internal pop.	external pop.	l
M-PAES	N/A	**uniform**	**1/L**	**flip**	5–20	80–95	5 / 3
SPEA	**0.8**	**uniform**	**1/L**	**flip**	5–20	80–95	N/A
1+1-PAES	N/A	N/A	**1/L**	**flip**	**1**	**99**	5 / 3

Figure 13: Parameter settings for the three algorithms. Bold indicates a fixed value in all experiments. The ranges used for the free parameters is shown. The two values for l, the number of hypergrid divisions, refer to the two and three objective problems, respectively.

To obtain meaningful statistical results, each algorithm is run thirty times on each problem instance considered. The full set of results obtained from the three-way comparison are provided in Table 14. The different problems are labelled by the number of nodes and the number of the scenario. E.g. The third scenario of the twenty node problem is labelled 20-3.

ADDMP instance	statistic	2-objective M-PAES	2-objective SPEA	2-objective PAES	3-objective M-PAES	3-objective SPEA	3-objective PAES
10-1	*unbeaten*	93.8	83.1	99.2	98.0	81.3	100
	beats all	0.8	0	5.0	0	0	2.0
10-2	*unbeaten*	100	99.8	94.9	68.5	98.2	100
	beats all	0	0	0	0	0	1.8
10-3	*unbeaten*	100	98.7	95.2	77.3	78.4	100
	beats all	0	0	0	0	0	8.3
10-4	*unbeaten*	100	98.9	100	64.3	51.8	100
	beats all	0	0	0	0	0	34.1
10-5	*unbeaten*	99.8	49.7	100	35.3	54.9	100
	beats all	0	0	0.2	0	0	16.3
20-1	*unbeaten*	100	48.7	100	95.8	24.1	100
	beats all	0	0	0	0	0	3.0
20-2	*unbeaten*	65.0	67.2	100	63.7	63.7	99.8
	beats all	0	0	32.8	0	0	36.3
20-3	*unbeaten*	100	0	100	98.9	67.0	100
	beats all	0	0	0	0	0	1.1
20-4	*unbeaten*	52.1	0.5	100	95.2	41.0	100
	beats all	0	0	47.9	0	0	4.8
20-5	*unbeaten*	49.9	50.0	100	23.8	24.7	100
	beats all	0	0	50.0	0	0	74.1
40-1	*unbeaten*	92.0	15.6	99.9	77.4	31.0	70.8
	beats all	0.1	0	8.0	9.9	0	22.6
40-2	*unbeaten*	69.3	7.9	93.4	76.4	26.2	87.6
	beats all	4.6	0	30.7	0.5	0	23.6
40-3	*unbeaten*	68.7	10.3	93.8	71.7	15.0	77.4
	beats all	4.2	0	31.3	10.9	0	28.3
40-4	*unbeaten*	100	11.2	99.5	78.2	0.1	93.1
	beats all	0	0	0	6.9	0	21.8
40-5	*unbeaten*	68.2	12.9	94.7	69.1	57.7	73.0
	beats all	4.9	0	31.8	0	0	30.9

Figure 14: The *unbeaten* and *beats all* statistics for the combined space inhabited by the solutions found. Two forms of the problem were investigated: The 2-objective case, where the median response time and the worst response time are minimized. And the 3-objective case, where the median response time, the response time bettered by 80% of requests, and the response time bettered by 90% of requests are minimized.

In almost all cases, we can see that either M-PAES or PAES is superior to SPEA. On the smaller problems, performance of all three algorithms tends to be similar on the two objective case, although SPEA performs relatively poorly on one of the scenarios. On the three-objective case, M-PAES performance is rather disappointing, although PAES is clearly superior to SPEA. On the moderately sizes problems, PAES is clearly superior to the other two, and SPEA seems worst, although occasionally M-PAES performs quite poorly. Finally, on the larger problems, M-PAES and PAES provide fairly similar performance, both being distinctly superior to SPEA.

5 Conclusion

In this paper we have looked at two distinct telecommunications optimisation problems in a multiobjective context, and looked at the performance of advanced multiobjective optimization algorithms on examples of these problems.

We refer to previously published work (Zitzler, 1999; Knowles and Corne, 2000 2000a) in establishing the fact that the modern MOEAs discussed here are generally considered superior in both speed and solution quality to seminal methods such as those pioneered by Horn and Napfliotis (1994) and Srinivas and Deb (1994), and also highly competitive with methods arising from the operations research and local search communities (Zitzler, 1999; Knowles and Corne, 2000a; 2000b). Meanwhile, in this paper we have shown that PAES (in one case) and both PAES and M-PAES (in the other case) provide excellent results on two distinct multiobjective problems in telecommunications.

Confidence in the experiments reported, as well as the claims derived from previously published work, is attained via use of a sophisticated procedure for statistical comparison of multiobjective optimizers. As we saw with the results on the offline routing problem, it is important not to take these statistical indicators in isolation, however the method clearly provides a strong guide when it comes to choosing an algorithm for use in live application.

In general, PAES and M-PAES are two of a small family of algorithms which are beginning to make feasible the fast generation of near-optimal tradeoff surfaces for complex multiobjective problems. Owing to the slow and relatively ineffective results from earlier work on multiobjective optimization, many complex problems are in practice simplified and treated as single objective. However, the performance of new algorithms such as PAES and M-PAES suggests that such problems can be revisited in their more natural form, and real-world optimization packages can be designed which quickly deliver high-quality tradeoff surfaces for further processing and decision making.

References

Back, T. (1996) *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New Yourk, NY, USA.

Clark, M.P. (1996) *ATM networks: principles and use*, John Wiley and Sons Ltd, Chichester, UK.

Czyżak, P. and Jaszkiewicz, A. (1998), Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization, *Journal of Multi-Criteria Decision Analysis*, **3**(1), pp. 34–47.

Edwards, D. Performance Adaption Algorithm (draft 4x4c), British Telecommunications Advanced Networks and Systems Project Document, Ipswich, UK.
Fogel, D.E. (1995) *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, NJ, USA.

Fonseca, C.M. and Fleming, P.J. (1996) On the Performance Assessment and Comparison of Stochastic Multi-objective Optimizers, in Voigt, H.-M., Ebeling, W. Rechenberg, I. and Schwefel, H.-P. (eds.), *Parallel Problem Solving from Nature—PPSN IV*, Springer Verlag Lecture Notes in Computer Science, Springer, Berlin, Germany, pp. 584–593.

Gandibleux, X., Mezdaoui, N. and Fréville, A. (1996), A tabu Search Procedure to Solve Multiobjective Combinatorial Optimization Problems, in Caballero, R. and Steuer, R. (eds.), *Proceedings volume of MOPGP'96, Springer-Verlag, Berlin, Germany, pp. 291–300.*

Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimisation, and Machine Learning*, Addison-Wesley, Reading, MA, USA.*

Hansen, M.P. (1996) Tabu Search in Multiobjective Optimisation : MOTS, in Stewart, T. and van den Honert, R. (eds.), Proceedings of 13th International Conference on Multiple Criteria Decision Making*, Springer-Verlag, Berlin, Germany.*

Holland, J.H. (1975) Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, USA.*

Horn, J. and Nafpliotis, N. (1994) Multiobjective Optimisation Using The Niched Pareto Genetic Algorithm, IlliGAL Report 93005, Illinois Genetic Algorithms Laboratory, University of Illinois, Urbana, IL, USA.

Horn, J., Nafpliotis, N. and Goldberg, D.E. (1994) A niched pareto genetic algorithm for multiobjective optimisation, in Proceedings of the First IEEE Conference on Evolutionary Computation, *IEEE World Congress on Computational Intelligence, Volume 1. Piscataway, NJ: IEEE Service Centre, pp. 67–72.*

Knowles, J.D. and Corne, D.W. (2000) Approximating the nondominated front using the Pareto Archived Evolution Strategy, Evolutionary Computation **8***(2):149–172.*

Knowles, J.D. and Corne, D.W. (2000a) M-PAES: A Memetic Algorithm for Multiobjective Optimisation, submitted to the 2000 Congress on Evolutionary Computation.

Knowles, J.D. and Corne, D.W. (2000b) A Comparison of Diverse Approaches to Memetic Multiobjective Combinatorial Optimization, submitted to WOMA: The Workshop on Memetic Algorithms at the 2000 Genetic and Evolutionary Computation Conference.

Mann, J.W. and Smith, G.D. (1996) A Comparison of Heuristics for Telecommunications Traffic Routing, in Rayward-Smith, V.J., Osman, I.H., Reeves, C.R. and Smith, G.D. (eds.), Modern Heuristic Search Methods, *John Wiley and Sons Ltd, Chichester, UK.*

Moscato, P. (1999) Memetic Algorithms: An Introduction, in Corne, D., Dorigo, M. and Glover, F. (eds.), New Ideas in Optimization, McGraw-Hill, London, UK.

Munetomo, M., Takai, Y. and Sato, Y. (1997) An Adaptive Network Routing Algorithm Employing Path Genetic Operators, in Proceedings of the 11th International Conference on Genetic Algorithms, *Morgan Kuafmann, pp. 643–649.*

Narula, S.C. and Ho, C.A. (1980) Degree-Constrained Minimum Spanning Tree. Computers and Operations Research **7***(4):39–49.*

Oates, M.J. and Corne, D.W. (1998) Investigating Evolutionary Approaches for Self-Adaptation in Large Distributed Databases, in Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, *IEEE Neural Networks Council, pp. 452–457.*

Oates, M.J. and Corne, D.W. (2000) Exploring Evolutionary Approaches to Distributed Database Management, in Corne, D.W., Oates, M.J. and Smith, G.D. (eds.), Telecommunications Optimisation: Heuristic and Adaptive Techniques, *John Wiley and Sons Ltd, Chichester, UK, pp. 235–264.*

Srinivas, N. and Deb, K. (1994) Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, Evolutionary Computation **2***(3):221–248.*

Tanenbaum, A.S. (1996) Computer Networks. *(third edition), Prentice-Hall Inc, USA.*

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1988) Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press.

Ulungu, E.L. and Teghem, J. (1994) Multi-objective Combinatorial Optimization Problems: A Survey, Journal of Multi-Criteria Decision Analysis **3***, pp. 83–104.*

Yen, J.Y. (1971) Finding the K Shortest Loopless Paths in a Network, Mangement Science **17***(11).*

Zitzler, E. (1999) Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, PhD Thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

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