EngOpt 2012 - International Conference on Engineering Optimization
Rio de Janeiro, Brazil, 1-5 July 2012.

A Simplex-like Search Method for Bi-objective Optimization

T. C. Peachey† M. J. W. Riley‡ D. Abramson † J. Stewart‡

†Caulfield School of Information Technology
Monash University
Caulfield, Vic 3145, Australia
tcp.free@gmail.com, david.abramson@monash.edu
‡Lincoln School of Engineering
University of Lincoln
Brayford Pool, Lincoln, LN6 7TS, United Kingdom
mriley@lincoln.ac.uk , jstewart@lincoln.ac.uk

*

Abstract

We describe a new algorithm for bi-objective optimization, similar to the Nelder Mead simplex algorithm, widely used for single objective optimization. For differentiable bi-objective functions on a continuous search space, internal Pareto optima occur where the two gradient vectors point in opposite directions. So such optima may be located by minimizing the cosine of the angle between these vectors. This requires a complex rather than than a simplex, so we term the technique the "cosine seeking complex". An extra benefit of this approach is that a successful search identifies the direction of the efficient curve of Pareto points, expediting further searches. Results are presented for some standard test functions. The method presented is quite complicated and space considerations here preclude complete details. We hope to publish a fuller description in another place.

Keywords: multi-objective optimization, simplex, complex.

1 Introduction

The proliferation of computational models and their use in engineering design has given rise to a renewed interest in multi-objective optimization. Every design involves trade-offs between many competing demands and desires. Modelling can formalize these choices by quantifying the criteria and produce sets of Pareto parameters representing "best" compromises from which the engineer can make the final decision.

Typically, these models are computationally expensive, a single run may take hours or days on a high-end processor. So the exploration of the design space needs to be parsimonious in terms of the number of runs required. We assume that the number of model runs required will be a prime measure of the efficiency of the exploration algorithm.

We also assume that the model input parameters that are varied in the explorations take a range of values in a continuum of real numbers and that the objective values produced by these models are smooth functions of the input variables. For smooth functions one might expect the natural choice of algorithm to be some gradient based search or perhaps some direct search, such as the simplex method of Nelder and Mead, [14], that uses a general indication of the direction that will improve the objectives and that benefits from a smooth response surface. But current practice is dominated by population based evolutionary algorithms, [7]. We begin by surveying existing alternatives.

A long-standing method, dating back at least to the 1960s, [17], is to optimize a weighted mean of the objectives. If that mean is optimal then the objectives are certainly Pareto optimal. So by optimizing various weighted means, various Pareto points may be determined. There are two well-known difficulties with this approach. If the feasible set in the vicinity of the Pareto front is concave then the method may fail to delineate the full front. Secondly, the sample of points obtained may not show an even distribution along the front. Das and Dennis, [3], have elegantly shown the genesis of these problems. Much research effort has been expended in remediation of these difficulties, well summarised in [8].

*The authors wish to thank Timoleon Kipouros for helpful discussions.

A quite different approach uses the fact that Pareto solutions form manifolds in the search space. Homotopy tracking techniques may be used to determine these sets, [16]. However, these methods require accurate second derivatives of the objective functions, information not available when the objectives are outputs of some computational model. Curiously, the method presented in this paper is able to perform such tracking using only numerically feasible estimates of the first derivatives.

Brown and Smith, [2], demonstrate how quadratic programming may be used to determine a search direction, one which will improve all objectives. This is not presented as a complete algorithm on its own. Rather, it is used to accelerate an evolutionary computation approach by restricting new offspring to those that will dominate existing elements of the population. Independently, [4], Dellnitz *et al.* came to similar conclusions and showed a simple formulation for the search direction in the bi-objective case. They demonstrated an elegant covering method based on this theory, assuming that gradient information is available. Then López *et al.*, [9], hybridized this approach with an evolutionary algorithm.

Direct search is used in multi-objective optimization, in conjunction with other techniques. Jaeggi *et al.* [6], have had success with the Hooke and Jeeves algorithm, [5], combined with a Tabu search. Their method seems able to cope with the very high dimension searches required in some shape optimization. Martínez *et al*, [12], have used a Nelder Mead simplex search, in conjunction with the penalty boundary method of Zhang and Li, [19], to iteratively improve a population of approximations to the Pareto front. This approach appears promising so we have opted to compare our results with theirs.

2 Terminology

We are concerned with optimization over a bounded subset S of a Euclidean space, and will consistently use n for the number of space dimension. Our implementation assumes

$$S = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n],$$

but the method applies more generally. The two objective function will be usually denoted by $f : S \to I\!R$ and $g : S \to I\!R$.

We assume the language of multi-objective optimization, in particular the terms "dominate", "Pareto maxima" and "Pareto minima" which we apply to both points in the search space and in the objective space. These Pareto optima will be termed "local" if they are non-dominated just in some neighbourhood in the search space. The internal local Pareto optima form "efficient curves" in the search space and the global optima form the "Pareto front" in objective space.

3 Fundamental Theory

The proposed algorithm is based on a simple observation.

Theorem 3.1. *Suppose that $f : I\!R^n \to I\!R$ and $g : I\!R^n \to I\!R$ are differentiable, and $u \in I\!R^n$ is a locally Pareto point for jointly maximizing (or jointly minimizing) both f and g. Then at that point, ∇f and ∇g have opposite directions, or one of them is null.*

We have insufficient space for proofs, although in this case the result is obvious. Moreover, if we also assume that the objective functions have continuous derivatives, then the result follows from standard Karush-Kuhn-Tucker (KKT) theory, [13].

Henceforth we use the term "antithetical" (from the Greek αντίϑεση for "in opposition") for a point where ∇f and ∇g have opposite directions. This antithetical condition is necessary, but not sufficient for a local Pareto optimum, as is demonstrated for a two-dimensional search space by Figure 3. Contours for one objective are shown in red and the other in black. At antithetical points contours of the two objectives are mutually tangent. The nature of an antithetical point is related to the values of the objectives in the cusp between these tangent curves. The central point is a Pareto maximum in case (a), a minimum in case (b), and neither in (c).

So if a search finds an antithetical point the question remains as to which of these three cases applies. For real-world simulations this may not be a problem as practical considerations may indicate the nature of the point. KKT theory does supply a test.

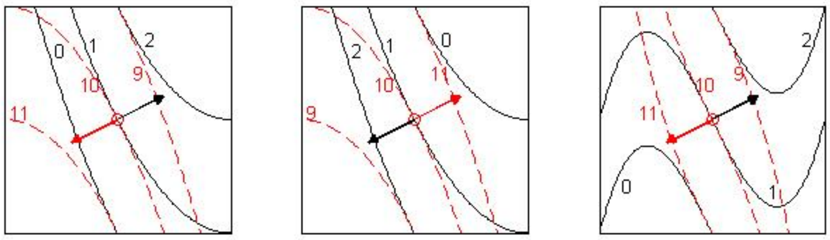

Figure 1: Antithetical points may be (a) Pareto maximum (b) Pareto minimum (c) neither

Theorem 3.2. *Suppose that $f : I\!R^2 \to I\!R$ and $g : I\!R^2 \to I\!R$ have continuous second derivatives at a point v and that there exists $\lambda > 0$ such that at $x = v$,*

$$\nabla(f(x) + \lambda g(x)) = 0, \quad (1)$$

$$w^T \nabla^2 (f(x) + \lambda g(x)) w < 0, \quad (2)$$

where ∇^2 is the Hessian matrix operator and w is any non-zero vector normal to ∇f. Then v is a Pareto maximum for f and g. If the inequality 1 is reversed, then v is a Pareto minimum.

The appearance of $f + \lambda g$ in (1.2) shows that the theory offered here relates to the method of minimizing a linear combination of f and g, discussed above. The method we will present in the next section will produce λ as a by-product of the search, thus obviating the difficulty of choosing its value.

In practice the search for Pareto points is conducted over a finite domain. The theorems above will relate only to an interior point of that domain. Points that are Pareto optimal and lie on the domain boundary need not be antithetical.

4 The Primary Search Algorithm

Theorem 3.1 suggests that a search for Pareto points may be reduced to maximizing the angle θ beween the gradient vectors. Actually, it will be more convenient to compute the cosine of that angle, so we will instead minimize $\cos\theta$. The importance of $\cos\theta = -1$ has been noted in [9] in connection with improving both objectives for a population of solutions. We go further in abandoning a direct attempt to maximize (or minimize) the objectives and concentrate on minimizing $\cos\theta$. to find antithetical points.

4.1 Estimation of the Cosine

Since we assume here that gradient information for the objectives is not directly available, a method of estimating the gradient vectors is required. A single point will not suffice; in n dimensions we require $n+1$ points that do not all lie in some hyperplane. In other words we need the vertices of an n-dimensional simplex. If the objectives are computed for each vertex, then a linear approximation for each objective may be found and this can be used to generate each gradient. We then obtain $\cos\theta \approx \nabla f \cdot \nabla g / \|\nabla f\| \|\nabla g\|$.

Assuming (as we do) that the functions f and g are differentiable, the accuracy of the approximation should increase as the separation of the vertices decreases. So a simplex is sought (i) with small dimensions and (ii) for which $\cos\theta \approx -1$.

4.2 Minimizing the Cosine

A single simplex then will yield just a single estimate of the angle θ. Starting with some initial “core” simplex, the method will repeatedly move to a simplex that yields a better angle. This is done by constructing a set of nearby simplices, evaluating $\cos\theta$ for each, and then selecting the best simplex as the core of the next iteration. If none of the new simplices shows any improvement, then the simplex is deemed to be near an optimal point and a "shrink" operation will be performed yielding a smaller core simplex. As in the Nelder-Mead simplex method, the core simplex will (hopefully) "walk" around the search space and then shrink toward an optimal point.

Iterations will cease when the core simplex is found to be smaller than some specified "spatial tolerance" η. In practical optimization, the search variables may differ greatly in scale, so when calculating the size of a simplex, the separate co-ordinates are normalized. That is, if the domain for the i^{th} coordinate is $[d_i, D_i]$ and the simplex occupies $[s_i, S_i]$ of that domain, then the size of the simplex is taken to be $\sigma = \max_i (S_i - s_i)/(D_i - d_i)$.

It may happen that the iterations converge, the core simplex shrinks to a size $\sigma < \eta$, but that it's $\cos\theta$ does not approach -1. We also specify an "objective tolerance" ϵ. A search will be deemed successful if both $\sigma < \eta$ and $\cos\theta + 1 < \epsilon$.

The description of the algorithm above is intentionally vague. The actual algorithm used should both converge to an antithetical point, and do so with a minimum of computational effort. Here computational effort will be measured by the number of function evaluations of f and g required, since in the scenario described in the introduction, these computations dominate all others.

The crucial decisions to be made are: which alternative simplices to construct at each iteration, how to shrink the simplex, and what to do if the simplices run up against the boundary of the search space. The method presented below represents our solutions to these questions. The method is completely heuristic in that it is the result of many informal experiments with standard test functions. Possibly better schema can be found.

4.3 Moves to Adjacent Simplices

Figure 2 shows the structure that we will use for the case $n = 2$. Starting with a core simplex ABC, more simplices are constructed by reflecting vertices through the centre of the opposite edge. So reflection of A though P to D for example gives simplex BDC. In two dimensions this gives three more simplices making four in all. When $\cos\theta$ has been computed for these simplices, that with the minimum value becomes the new core. For example, suppose that simplex BDC is deemed the best simplex then, in the next iteration, simplices AFB and ACE will be deleted from the complex and new ones BGD and CDH added.

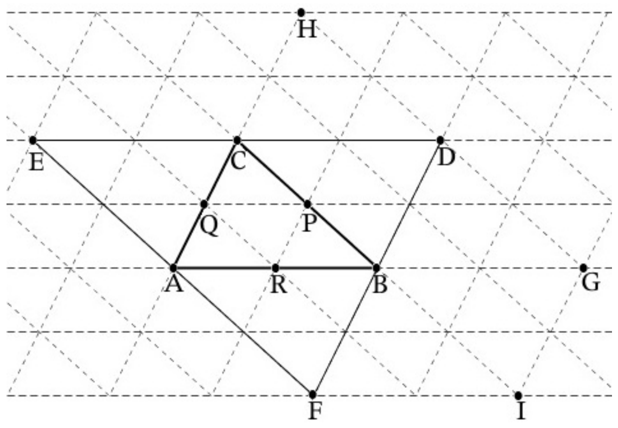

Figure 2: Two-dimensional case of the complex used

In n dimensions, the core simplex will have $n + 1$ vertices and $n + 1$ hyperfaces. Each vertex is reflected through the centroid of the opposite hyperface, producing $n + 1$ new simplices. Such a move will be termed a "face move" as the new simplex shares a hyperface with the previous. This strategy is computationally parsimonious as it produces $n + 1$ simplices from $n + 1$ vertex evaluations, although some of these simplices will normally have been visited before.

The term “complex” is the standard mathematical term for such a connected set of simplices. The name “Complex algorithm” would seem apt for the search method presented here, but this has already been used for Box’s single objective optimization method [1]. We will use the designation CSC for “cosine seeking complex”.

Face moves give an insufficient variety of directions. We also use “corner moves” and “translations moves” which use simplices connected to the core by just a vertex. For a corner move the best vertex of the core simplex is identified as that opposite the worst face move simplex. Then the core simplex is point reflected in that vertex. In the example shown in Figure 2, if the simplex ACE produces the greatest $\cos\theta$, then the simplex BIG will be a potential corner move. For a translation move the best and worst vertices of the core are identified, say B and W respectively, then the core is translated by the vector WB. In Figure 2, if B is best and A worst then the new simplex will be BGD. These two types of moves are computationally expensive compared to face moves. The most efficient strategy that we discovered was to use corner moves only when not near a Pareto optimum, when $\cos\theta + 1 > \eta$, the cosine tolerance. And translations are considered only if other moves fail to improve the cosine.

4.4 Boundary Collision

Collision of the complex with the boundary is to be expected when Pareto optima lie on the boundary. Although we think this unlikely for a well defined practical problem, we tried various techniques to enable the complex to track along that boundary, improving $\cos\theta$.

The method found most effective and finally adopted was to allow the complex to have vertices outside the search space, projecting them onto the boundaries only for the purpose of evaluating the objectives. If a component simplex is squashed flat for the evaluation then the gradient vectors can no longer be evaluated and that simplex is not considered as a new core simplex. Consequently the core simplex always maintains a positive hypervolume. However, the absence of a full complement of face and corner moves hampers progress of the complex along the boundary, we do not consider the problem resolved. Perhaps a better approach would be to confine the search to the boundary involved. That is, to set the relevant parameter to the boundary value and convert the search from n dimensions to $n-1$.

4.5 The Shrink Operation and Search Termination

The shrink operation used involves selection from several smaller simplices. For each vertex of the core simplex the simplex was shrunk to half size toward that corner. In addition, another option was constructed by shrinking all vertices by a factor 1/2 toward the centroid of the core simplex. This provided $n+2$ smaller simplices in total. The objective cosine is computed for each and the best becomes the core simplex for the next generation.

Occasionally convergence occurs very slowly so a limit for the number of iterations is included. This is not a hard limit in that when reached, each subsequent iteration is forced to be a shrink operation. Thus convergence is assured after a finite number of further iterations.

4.6 Testing the Nature of an Optimum

The reader may expect an indication of whether the antithetical point is a maximum or a minimum from scrutiny of the sequence of points that converged to that point. Typically however, the search produces a sequence of non-comparable points, no point dominates another. The CSC method, in taking the shortest route to an antithetical point, does not select moves that simultaneously improve both objectives.

Applying Theorem 3.2 requires knowledge of the second derivatives, we will require at least a quadratic approximation to f and to g. A general quadratic in n dimensions has $(n+2)(n+1)/2$ arbitrary parameters, while the complex we are using, a core simplex and face-adjacent simplices, has $2(n+1)$ vertices. So knowledge of the function at the complex vertices will provide sufficient information to determine the quadratic for the case $n=2$, but not for larger n.

In this two-dimensional case the quadratic approximation to the surface takes the form

$$f(x,y) \approx \phi(x,y) = f_0 + f_1 x + f_2 y + f_3 x^2 + f_4 xy + f_5 y^2.$$

If a complex search has terminated successfully, the final complex vertices will be close together, so ϕ will make a good approximation to f near those points. The known values for f at the vertices will yield six equations to determine the coefficients f_i. Similarly coefficients g_i in a quadratic approximation for g can be determined. Substituting into inequality (2) gives

$$f_2^2(f_3 + \lambda g_3) - f_1 f_2(f_4 + \lambda g_4) + f_1^2(f_5 + \lambda g_5) < 0$$

for a Pareto maximum, the opposite inequality, a minimum. We term the left side of the inequality the "Hessian form".

5 Implementation

The algorithm was coded in C and implemented as an option within the Nimrod/O distributed optimization system [10]. Nimrod/O provides a convenient framework for the types of engineering optimization problems discussed in the introduction and has been extensively used for single objective optimization. By interfacing with the Nimrod/G [11], it can execute batches of jobs concurrently using whatever computatonal resources are available, multiple cores on local machines, clusters, computational grids or clouds. Furthermore, multiple searches will automatically run in parallel, a useful attribute when seeking multiple Pareto points.

6 Tracking Efficient Curves

When the CSC method has determined an antithetical point, it also gives the (opposing) gradient vectors for the two objectives at that point. Assuming reasonable smoothness of the objective functions these directions will be tangent to the efficient curves. Hence the gradient information at an antithetical point can be used to obtain the location of nearby antithetical points. This suggests an iterative procedure for tracing along an efficient curve.

Suppose that at an antithetical point P_0 the gradient of, say, f is $v = \nabla f$. Moving a small distance s in this direction to $Q = P_0 + s\,\widehat{v}$, then Q will normally be near another antithetical point R. Not only will $|Q - R|$ be small, it will be small relative to s. Hence if we use this Q as the starting point of another CSC search, the starting size of the initial complex, say d, may be made small, expediting the convergence of the new search. If convergence is achieved, say to a new antithetical point P_1, the procedure may be iterated with a new estimate of ∇f, producing a sequence of antithetical points P_1, P_2, $\ldots$.

The choice of the starting size d is important in achieving the most efficient algorithm. Ideally d should be large enough so the initial complex encompasses a Pareto point, but not much larger. If d is too small then the algorithm will require many side-steps to reach the Pareto point. If too large then the algorithm may begin with many shrink operations. The choice of d relative to s will depend on the curvature of the Pareto track. This might be estimated by a sophisticated consideration of the second derivatives. However, we have chosen a simpler technique of dynamic adjustment. If, in the previous step along the track, the algorithm began with seven or more side-steps then the new search will double the value of d. Alternately, if it began with seven or more shrinks, then d is halved for the new search. Of course the step size s and the value of d for the first track step, say d_0, still require specification.

When the steps along the efficient curve go past the terminus, one might expect the CSC method to search back to the curve. This would be manifest in the new point P_i being close to an earlier antithetical point, close relative to the distance s. We define a "termination fraction" α such that when P_i is within a normalized distance αs of some previous P_j, tracking with ∇f will terminate. Returning to the initial point P_0, the algorithm will now track in the other direction using ∇g.

To demonstrate the initial search and subsequent tracking, we have applied this method to a bi-objective problem used by Poloni *et al.* [15].

The results are shown in Figure 3 with antithetical points shown as disks, red for Pareto maxima and green for minima as deemed by the Hessian form condition. The initial search started from $(0, 0)$ and converged to P. Tracking with ∇f produced the sequence of 14 antithetical points to the right of

P, approaching C, a critical point for f. Near C, the tracking shows greater volatility and the 14th point is very close to the 12th, sufficiently close to terminate tracking in this direction. So the tracking returned to P and used ∇g to track to the left, producing another 13 points before terminating near A.

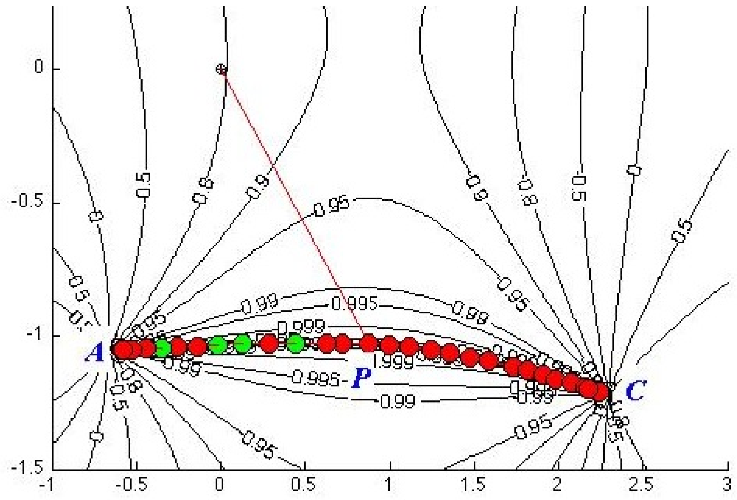

Figure 3: Following the efficient curve for the Poloni function

All points shown are local maxima. Note that the Hessian form test has failed in four cases to correctly distinguish maxima from minima. The Hessian computation is effectively a numerical second derivative, and thus very sensitive to noise.

Table 1: Iterations required for each optimization in a tracking operation

Optimization	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Iterations	25	16	15	14	13	13	11	11	11	14	11	15	18	25
Optimization	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Iterations	32	14	13	79	11	10	17	17	17	15	17	17	24	27

The number of iterations at each step is shown in Table 1, demonstrates the reduction in computational effort in subsequent steps. A notable exception is the 18^{th} optimization which, for reasons unclear, was slow to converge.

7 Hybrid Methods

A CSC search typically makes a sequence of sideways moves (face, corner and translation) until the complex covers an antithetical point and then mixes shrinks with sideways moves . If the CSC method begins it's search already very near an antithetical point then the search effort is greatly reduced. Not only are the original moves unnecessary, the starting core simplex can be made smaller, avoiding the need for some of the shrink operations. Further, the ambiguity in the type of optima found (maximum or minimum) is avoided; if the starting point is known to be near say, a global Pareto maximum then the final antithetical point is likely to also be approximately a global maximum.

This suggests a role for the method in improving the accuracy of Pareto points obtained by other methods. Space restrictions preclude details of our experiments in this vein. We found that an internal non-dominated point did move towards an efficient curve. However the population of internal points tended to bunch together reducing the Pareto front coverage as measured by the hypervolume metric. Further, some searches starting from boundary points failed to converge, so boundary Pareto point were lost from the population. Our solution was to retain all the starting points in the population unless they became dominated by new points.

8 Experiments

8.1 PICS, The Variant Used

Foregoing sections have presented a variety of algorithms with a common basic core. For the purpose of obtaining performance results that can be compared with other methods, we have selected certain attributes from these variants. In particular we desire a method that will search for just Pareto minima, or just maxima, thus avoiding extraneous computational effort.

First a sample of points is selected at random within the search space. These are evaluated and the set of results is winnowed to remove all dominated points. The remaining points are used as starting points for CSC searches. If the sample size is reasonably large then these starting points will be reasonably close to Pareto minima and then, as discussed above, the CSC method will (probably) converge to minimum. If a successful search is achieved, then tracking is used to explore along the efficient curve. However, since several starting points may all converge to the same efficient curve, the various optimizations will pool their tracking point caches and tracking will be stopped prematurely if points are found to be near points found by an earlier optimization. For this combination of approaches we will use the acronym PICS, for population improving complex search. Note that we have the set of starting points $\mathcal{S}$, and the set of points which are successful CSC searches, $\mathcal{P}$. A third set of points, $\mathcal{A}$, the set of all non-dominated points evaluated so far, is also maintained. This is the set actually used for the final solution

In order to achieve reproducible results the multiple CSC searches are performed serially. However a search will be aborted if the starting point is found to be dominated by any point of $\mathcal{A}$.

8.2 The Test Problems

The proposed approach was tested on an engineering design problem and on a selection of synthetic test problems. The test problems used were some deemed similar to a practical problem. The objectives were smooth functions of real variables; we have avoided the type of combinatorial problems more relevant to operations research. We have also preferred problems where the Pareto optima do not lie on the search space boundary. Thus we have not used any of the ZDT problem suite, [18], for example. The test problems used by Martínez *et al.* [12], satisfy these requirements and by using some of those problems we are able to compare results with their NSS-MO method and with MOEA/D, a well-performed decomposition based evolutionary algorithm, [19]. In particular we give results here for the Lis and the Fonseca test problems. We have experimented with higher dimensional search spaces, up to $n = 30$, and with a structural engineering shape optimization problem but space restrictions prevent us giving these results.

8.3 Parameter Settings

The CSC based methods, in particular the PICS variant, require quite a few parameters to be set. Extensive experiments would be required to determine suitable ranges for these. For the current experiments we used: spatial tolerance 0.001, objective tolerance 0.002, starting size 0.05, tracking step 0.001, termination fraction 0.001, starting track simplex size 0.0002, maximum iterations 100, preliminary sample 50. Our informal experiments indicate that the values shown in seem to be roughly optimal for the experiments reported here. We hope to report more fully on how to choose suitable parameters at a later date.

8.4 Results

The PICS algorithm was performed 30 times for experiments with the synthetic test problems. The hypervolume metric was computed using reference points $(1, 1)$ in the Lis case and $(1.1, 1.1)$ for Fonseca. For both the hypervolume metric and the number of evaluations required, tables show the mean , the (unbiased) sample standard deviation s and the standard error $s/\sqrt{30}$.

8.4.1 The Lis Problem

Results are shown in Table 2. Comparison with the results of Martínez *et al.* [12] is complicated by the fact that PICS searches will terminate after a fixed (but unknown) number of iterations, whereas the method used in [12] may be extended to any number of generations. They quote results for 40 generations (4000 evaluations); to make a reasonable comparison we have also given their results after 400 evaluations. These were estimated from the graphs presented.

Table 2: Comparative Performance on the Lis Problem

	PICS	NSS-MO	NSS-MO@400	MOEA/D	MOEA/D@400
$\mathcal{H}$: mean	0.2842	0.3097	0.15	0.2624	0.18
$\mathcal{H}$: s.d.	0.0106	0.0077	NA	0.0088	NA
$\mathcal{H}$: s.e.	0.0019	0.0014	NA	0.0016	NA
Evals.: mean	339.5	4000	400	4000	400
Evals.: s.d.	12.4	0	0	0	0
Evals.: s.e.	2.26	0	0	0	0

Both PICS and NSS-MO give a superior results to those produced by MOEA/D. Ultimately NSS-MO produced a better hypergeometric metric than PICS . However, PICS has completed after about 400 evaluations. At that stage NSS-MO is producing a considerably worse result.

Note also that the concave nature of the Pareto front has not precluded a successful result with the PICS method.

8.4.2 The Fonseca Problem

Table 3 again compares results with those produced by NSS-MO and by MOEA/D, although here a comparison after 800 evaluations is apposite. Conclusions are as for the Lis problem experiment.

Table 3: Comparative Performance on the Fonseca Problem

	PICS	NSS- MO	NSS- MO@800	MOEA/D	MOEA/D@800
$\mathcal{H}$: mean	0.529	0.542	0.49	0.374	0.34
$\mathcal{H}$: s.d.	0.029	0.0015	NA	0.0074	NA
$\mathcal{H}$: s.e.	0.0053	0.00027	NA	0.0014	NA
Evals.: mean	774.8	4000	800	4000	800
Evals.: s.d.	311.7	0	0	0	0
Evals.: s.e.	56.9	0	0	0	0

9 Conclusions

A novel approach to Pareto optimization of two objectives functions of real variables has been presented. The method uses a complex that moves downhill, not downhill in the sense of the two objectives, but downhill for the cosine of the angle between the gradients of those two objectives, hence the name “Cosine Seeking Complex”. The method locates solutions that are locally Pareto, as well as the usual global ones. Further, the solutions may be Pareto maxima or minima; the technique is blind to the distinction. An elaboration of the method, the “Population Improving Cosine Search”, steers convergence to just maxima or just minima, and delineates the relevant efficient curves.

The scenario addressed is that where the objectives are outputs of a computationally intensive model, so the number of objective evaluations required determines the computational effort. Tests with some standard test functions suggests that the method is superior to evolutionary algorithm techniques when the dimensionality of the search space is not great. Comparisons with the NSS-MO algorithm show some advantages and some drawbacks.

References

[1] M. J. Box, A new method of constrained optimization and a comparison with other methods, *The Comput. J.*, 1965, 1, 42–52.

[2] M. Brown and R. E. Smith, Effective use of Directional Information in Multi-objective Evolutionary Computation, *GECCO 2003*, LNCS 2723, 778–789.

[3] I. Das and J.E. Dennis, A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems, *Struct. Optim.*, 1997 , 14, 63–69.

[4] M. Dellnitz, O. Schütze and T. Hestermeyer, Covering Pareto Sets by Multilevel Subdivision Techniques,

[5] R. Hooke and T. A. Jeeves, Direct search solution of numerical and statistical problems, *J. of the ACM*, 1961, 8, 212–229.

[6] D. M. Jaeggi, G. T. Parks, T. Kipouros and P. J. Clarkson, The development of a multi-objective Tabu Search algorithm for continuous optimization problems, *European. J. of Oper. Res.*, 185, 1192–1212.

[7] D. F. Jones, S. K. Mirrazavi, M. Tamiz, Multi-objective meta-heuristics: An overview of the current state-of-the-art, *European. J. of Oper. Res.*, 2002 , 137, 1–9.

[8] I. Y. Kim and O. L. de Weck, Adaptive weighted sum method for multiobjective optimization: a new method for Pareto front generation, *Struct. Multidiscip. Optim., 31, 105–116, 2006*

[9] A. L. López, C. A. Coello Coello, O. Schütze, A Painless Gradient-Assisted Multi-Objective Memetic Mechanism for Solving Continuous Bi-objective Optimization Problems,

[10] Abramson D., Peachey T. and Lewis A., Model Optimization and Parameter Estimation with Nimrod/O *The Internat. Conf. on Comput. Sci., May 2006, University of Reading, UK.*

[11] Abramson, D., Giddy, J. and Kotler, L., High performance parametric modeling with Nimrod/G, Internat. Parallel and Distrib. Processing Symp., Cancun, May 2000, 520–528.

[12] S. Z. Martnez, A. A. Montano and C. A. Coello Coello, A Nonlinear Simplex Search Approach for Multi-Objective Optimization, ???

[13] Nocedal J. and S.J. Wright, *Numerical Optimization* Springer, New York, 1999

[14] J. A. Nelder and R. Mead, A Simplex Method for Function Minimization, *The Comput. J.*, 1965, 7, 308–313.

[15] Poloni C., Giurgevich A., Onesti L., Periroda V., Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimiser for a complex design problem in fluid dynamics, *Comput. Methods Appl. Mech. Engrg.*, 2000, 186, 403–420.

[16] J. Rakowska, R. T. Haftka, and L. T. Watson, Tracing the efficient curve for multi-objective control-structure optimization, *Comput. Systems Engrg.*, 1991, 2 , 461–471.

[17] Zadeh L., Optimality and non-scalar-valued performance criteria, *IEEE Trans. on Automat. Control*, 1963, 8, 59–60.

[18] E. Zitzler, K. Deb, and L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, *Evolutionary Computing*, 2000, 8, 173–195.

[19] Q. Zhang and H. Li, MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition, *IEEE Trans. on Evolutionary Comput.*, 2007, 11, 712–731.