Multiobjective Optimization on a Budget of 250 Evaluations

Joshua Knowles[1] and Evan J. Hughes[2]

[1] School of Chemistry, University of Manchester, Faraday Building, Sackville Street, PO Box 88, Manchester M60 1QD, UK. Email:j.knowles@manchester.ac.uk
[2] Cranfield University, Shrivenham, Swindon SN6 8LA, UK

Abstract. In engineering and other 'real-world' applications, multiobjective optimization problems must frequently be tackled on a tight evaluation budget — tens or hundreds of function evaluations, rather than thousands. In this paper, we investigate two algorithms that use advanced initialization and search strategies to operate better under these conditions. The first algorithm, Bin_MSOPS, uses a binary search tree to divide up the decision space, and tries to sample from the largest empty regions near 'fit' solutions. The second algorithm, ParEGO, begins with solutions in a latin hypercube and updates a *Gaussian processes* surrogate model of the search landscape after every function evaluation, which it uses to estimate the solution of largest expected improvement. The two algorithms are tested using a benchmark suite of nine functions of two and three objectives — on a budget of only 250 function evaluations each, in total. Results indicate that the two algorithms search the space in very different ways and this can be used to understand performance differences. Both algorithms perform well but ParEGO comes out on top in seven of the nine test cases after 100 function evaluations, and on six after the first 250 evaluations.
Keywords: multiobjective optimization, expensive black-box functions, ParEGO, DACE, Bin_MSOPS, landscape approximation, response surfaces, test suites

1 Introduction

The vast majority of research effort in developing modern multiobjective evolutionary algorithms (MOEAs) has concentrated on improving algorithm performance and efficiency on runs, typically, of ten thousand function evaluations or more. In this paper, we consider multiobjective problems where a 'budget' of at most 250 evaluations is imposed because of the expensive nature of evaluating candidate solutions. More specifically, we are interested in problems where most or all of the features described in Fig.1 are true.

Features 1–4 limit the numbers of function evaluations possible, while features 5–8 make it reasonable to apply global search techniques rather than either random search or hillclimbing. Problems exhibiting these features include various combinatorial biochemistry and materials science applications [6, 26], as well as instrument set-up optimization in analytical chemistry [20, 24]. In [20], a standard MOEA, PESA-II, was successfully used to substantially improve the settings of a GC-MS spectrometer, using just 180 evaluations. However, it is clear that given such a restricted number of evaluations, and no

1. the time taken to perform one evaluation is of the order of minutes or hours,
2. only one evaluation can be performed at one time (no parallelism is possible),
3. the total number of evaluations to be performed is limited by financial considerations,
4. no realistic simulator or other method of approximating the full evaluation is readily available,
5. noise is low (repeated evaluations yield very similar results),
6. the overall gains in quality (or reductions in cost) that can be achieved are high,
7. the search landscape is multimodal but not highly rugged,
8. the dimensionality of the search space is low-to-medium,
9. the problem has multiple, possibly incommensurable, objectives.

Fig. 1. Features exhibited by problems of interest

particular restriction on computational overhead (since each experiment requires 20 minutes), a search strategy that more carefully considers each evaluation would be more appropriate.

Scanning the optimization literature reveals that a sparse but varied array of different techniques (that were proposed or could be used) for economizing on evaluations in multiobjective optimization has already been examined. One strand in this focuses on the use of *neural networks* for modeling the search landscape during optimization, in order to replace some real function evaluations with approximated ones [19, 8, 9], or to replace standard variation operators with adaptive ones [1]. The simpler concept of *fitness inheritance* has also been investigated in multiobjective optimization to economize on function evaluations [2, 5]. And a third strand is to use Bayesian network and/or other probabilistic model-building algorithms in a multiobjective scenario, e.g. [17].

However, while the above methods may offer some performance gains over standard MOEAs when function evaluations are expensive, not one of the studies above has demonstrated a significant performance advantage within the challenging evaluation budget we are interested in here. In this paper, we present and compare two recently proposed algorithms that take very different approaches to this challenge. The first algorithm, Binary-MSOPS, which is summarized below, is based on two separate pieces of work previously published by the second author [11, 12]. The second algorithm, ParEGO, was first described in a recent technical report [16], and is described here again in some detail. We evaluate these algorithms over a range of problems and, unlike in other studies, we focus explicitly on the first 250 evaluations only.

The rest of the paper is organized as follows. Sections 2 and 3 describe the two algorithms, while 4, 5 and 6 detail the test functions, performance assessment methods and parameter settings of the algorithms, respectively. Section 7 presents results and section 8 discusses findings and concludes.

2 Binary-MSOPS

The Binary-MSOPS algorithm is based primarily on the Binary Search Algorithm [11], summarized below. This method, which can be combined with almost any fitness as-

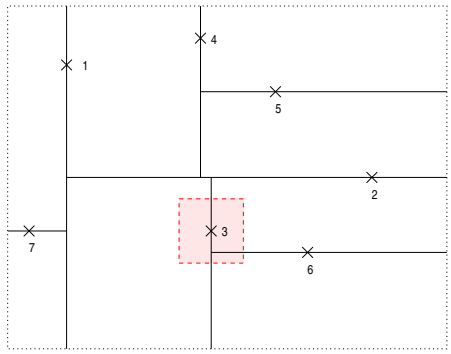

Fig. 2. The Binary Search process illustrated in a two dimensional decision space, with the first seven search points shown. The box around the third point indicates a small distance around a 'fit' point, and how this intersects with several empty regions

signment scheme, attempts to improve decision space sampling to ensure that promising regions are not missed or over-sampled in the early stages of the search, and to explicitly balance exploitation and exploration. Combining this with the MSOPS ranking method [12] —a computationally efficient means of assigning fitness in multiobjective optimization, based on target vectors —, Binary-MSOPS is both efficient and frugal with evaluations.

2.1 Binary Search Algorithm

The overall strategy of Bin_MSOPS uses a binary search tree [11] to divide the decision space into empty regions, allowing the largest empty region to be approximated. The search tree is constructed as shown in Fig. 2 by generating a point at random within a chosen hypercube, then dividing the hypercube along the dimension that yields the most 'cube-like' subspaces.

The basic algorithm for constructing the binary search tree (and generating new solutions) works by repeatedly choosing an exploration or exploitation step:

Exploration: Next point is generated at random within the largest empty region (i.e. global search),

Exploitation: Next point is generated within the largest empty region that is within a small distance of a selected *good* point (i.e. local search),

where the choice is random but biased by a parameter specifying the exploration/exploitation ratio.

The identification of a local region for exploitation is illustrated in Fig. 2. A small offset distance d_p is used to generate a hypercube of interest about the chosen point (chosen with tournament selection). The small hypercube is placed around the point of interest simply to provide an efficient means of identifying large neighbouring regions. A new point is then generated at random using a normal distribution in the largest region that intersects the hypercube.

At each iteration, the tree search to find the largest empty region is at worst $O(d\,P)$, where P is the number of evaluation points so far and d is the number of dimensions. Tree pruning can lead to $O(d\,log(P))$ performance for exploitation, and at worst $O(d\,P)$. Thus a computational explosion is avoided.

2.2 Population ranking by MSOPS

In order to decide which are 'good' points, the entire population is ranked, and a tournament between a random subset of the population, based on rank value, decides on the next solution to update. In order to control the computational complexity, a non-Pareto ranking approach has been applied, that like ParEGO (see next section), is also capable of handling many-objective problems. For this, the Multiple Single Objective Sampling (MSOPS) [12], with $O(P\,log(P))$ time complexity, has been used.

The concept of MSOPS is to generate a set of T target vectors, and evaluate the performance of every individual in the population, of size P, for every target vector, based on a conventional aggregation method. As aggregation methods (e.g. weighted min-max, ϵ-constraint, goal attainment etc.) are very simple to process, the calculation of each of the performance metrics is fast.

Thus each of the P members of the population has a set of T scores that indicate how well the population member satisfied the range of target conditions. The scores are held in a score matrix, S, which has dimensions $P \times T$. Each *column* of the matrix S corresponds to one target vector (each column containing P entries) and is ranked, with the best performing population member on the corresponding target vector being given a rank of 1, and the worst a rank of P. The rank values are stored in a matrix R. Each *row* of the rank matrix R may now be sorted, with the ranks for each population member placed in ascending order. The R matrix now holds in the first column the highest rank achieved for each population member across the set of target vectors. The second column will hold the second highest rank achieved etc. Thus the matrix R may be used to rank the population, with the most fit being the solution that achieved the most scores that were ranked 1, etc.

The flexibility of the approach is such that the target vectors can be arbitrary, either generated using some structure, or generated at random within certain limits. As the ranking method employed is based on the number of target vectors that are satisfied the best, a solution at the edge of the objective space will often satisfy vectors that cannot be attained. Thus the focus of the optimization is naturally drawn to interesting regions of surface such as the boundary of the optimization surface and discontinuities.

3 ParEGO: landscape modeling using Gaussian processes

Learning a cost landscape from a set of solution/cost pairs is variously called surrogate, approximate or meta- modeling in the literature [13]. In design engineering, meta-modeling is usually known as the response surface method [18], and involves fitting a low order polynomial via some form of least squares regression. A closely related approach, deriving from geology, is Kriging, whereby *Gaussian process* models are parameterized by maximum likelihood estimation. A particular example of this is known as the Design

and Analysis of Computer Experiments (DACE) model [22], which forms the basis of the EGO search algorithm [14]. EGO has been designed specifi cally for optimization on a very restricted evaluation budget: e.g. in [14], four low-dimensional multimodal test functions are optimized to within 1% of optimal in the order of 100 function evaluations.

The EGO algorithm begins by fi rst generating a number of solutions in a latin hypercube, and by then fi nding the maximum likelihood DACE model that best explains these solutions (making use of some suitable optimization algorithm). To generate a new solution to evaluate, EGO searches for the solution that maximizes what Jones et al [14] call "the expected improvement" —the expected value of that part of the standard error curve that lies below the best cost sampled so far. This effectively means that EGO weighs up both the predicted value of solutions, *and the error in this prediction*, in order to fi nd the one that has the greatest *potential* to improve the minimum cost. EGO does *not* just choose the solution that the model predicts would minimize the cost. Rather, it *automatically* balances exploitation and exploration: where a solution has low predicted cost and low error, it may not be as desirable as a solution whose predicted cost is higher but whose associated error of prediction is also higher. Once a new solution has been chosen and evaluated (using the true, expensive cost function), the DACE model is updated with this new information, and the next solution is chosen using this updated model.

The EGO algorithm could be extended for use with multiobjective optimization problems in a number of different ways. One simple approach recently proposed by the fi rst author in [16] (and that has the advantage of scaling to many objectives), converts the k different cost values of a solution into a single cost via a parameterized scalarizing weight vector. By choosing a different (parameterization of the) weight vector at each iteration of the search, an approximation to the whole Pareto front can be gradually built up. This multiobjective extension of EGO is called ParEGO.

ParEGO begins by normalizing the k cost functions with respect to the known (or estimated) limits of the cost space, so that each cost function lies in the range [0,1]. Then, at each iteration of the algorithm, a weight vector $\boldsymbol{\lambda}$ is drawn uniformly at random from the set of evenly distributed vectors defi ned by:

$$\Lambda = \left\{ \boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_k) \mid \sum_{j=1}^{k} \lambda_j = 1 \wedge \forall j, \lambda_j = l/s, l \in 0..s \right\}, \qquad (1)$$

with $|\Lambda| = \binom{s+k-1}{k-1}$, so that the choice of s determines how many vectors there are in total [10]. The scalar cost of a solution $f_{\boldsymbol{\lambda}}(x)$ is then computed using the augmented Tchebycheff function [23]:

$$f_{\boldsymbol{\lambda}}(x) = \max_j(\lambda_j \cdot f_j(x)) + \rho \sum_j \lambda_j \cdot f_j(x), \quad j \in 1..k \qquad (2)$$

where f_j is the raw cost value on objective j and ρ is a small positive value, which ensures all minima are proper Pareto optima, and which we set to 0.05. The scalar costs of all previously visited solutions are computed and, using all or a selection of these, a DACE model of the landscape is constructed by maximum likelihood. The solution that maximizes the expected improvement with respect to this DACE model is determined. This becomes the next point, and is evaluated on the real, expensive cost function, completing one iteration of ParEGO.

Algorithm 1 ParEGO pseudocode

```
1:  procedure ParEGO(f, d, k, s)
2:      xpop[] ← LatinHypercube(d)                    /* Initialize using procedure: line 15 */
3:      for each i in 1 to 11d − 1 do
4:          ypop[i] ← Evaluate(xpop[i], f)            /* See line 36 */
5:      end for
6:      while not finished do
7:          λ ← NewLambda(k, s)                       /* See line 19 */
8:          model ← Dace(xpop[], ypop[], λ)           /* See line 22 */
9:          xnew ← EvolAlg(model, xpop[])             /* See line 28 */
10:         xpop[] ← xpop[] ∪ {xnew}
11:         ynew ← Evaluate(xnew, f)
12:         ypop[] ← ypop[] ∪ {ynew}
13:     end while
14: end procedure

15: procedure LatinHypercube(d)
16:     divide each dimension of search space into 11d − 1 'rows' of equal width
17:     return 11d − 1 vectors x such that no two share the same 'row' in any dimension
18: end procedure

19: procedure NewLambda(k, s)
20:     return a k-dimensional scalarizing weight vector chosen uniformly at random from
            amongst all those defined by equation 1
21: end procedure

22: procedure Dace(xpop[], ypop[], λ)
23:     compute the scalar fitness f_λ of every cost vector in ypop[], using equation 2
24:     choose a subset of the population based on the computed scalar fitness values
25:     maximize the likelihood of the DACE model for the chosen population subset
26:     return the parameters of the maximum likelihood DACE model
27: end procedure

28: procedure EvolAlg(model, xpop[])
29:     initialize a temporary population of solution vectors, some as mutants of xpop[] and others
            purely randomly
30:     while set number of evaluations not exceeded do
31:         evaluate the expected improvement of solutions using the model
32:         select, recombine and mutate to form new population
33:     end while
34:     return best evolved solution
35: end procedure

36: procedure Evaluate(x, f)
37:     call the expensive evaluation function f with the solution vector x
38:     return true cost vector y of solution x
39: end procedure
```

Population size: 20 solutions
Population update: steady state (one offspring produced per generation, from either a crossover or cloning event, followed by a mutation)
Generations/evaluations: 10,000 evaluations
Reproductive selection: binary tournament without replacement
Crossover: simulated binary crossover [3] with probability 0.2, producing one offspring
Mutation: decision value shifted by $\pm 1/100.\mu.\rho$, where μ is drawn uniformly at random from $(0.0001, 1)$, ρ is the range of the decision variable, and p_m, the per-gene mutation probability, is $1/d$.
Replacement: offspring replaces (first) parent if it is better, else it is discarded
Initialization: 5 solutions are mutants[a] of the 5 best solutions evaluated on the real fitness function under the prevailing $\boldsymbol{\lambda}$ vector; the remaining 15 solutions are generated in a latin hypercube in decision space

[a] The mutation is carried out as described above except that mutants are checked to ensure they are different than parents.

Fig. 3. The EA used in ParEGO to search for the 'best' next solution

Pseudocode for the entire ParEGO algorithm is given in Algorithm 1. The Nelder and Meads downhill simplex algorithm is used (with 20 restarts) to maximize the likelihood of the DACE model (line 25 of Algorithm 1). The evolutionary algorithm used within ParEGO to search for the solution that maximizes the expected improvement (line 28) is implemented as detailed in Fig. 3.

In practice, on a very expensive cost function, all solutions previously evaluated should be used to update the DACE model, at every iteration. However, to save computational overhead in our experiments (because of the need to do 21 runs on a large number of functions to collect performance data), we used a simple, heuristic method of choosing a subset of the solutions evaluated to update the model, as follows: At each iteration: (i) if the iteration number $iter$ is less than 25, all $11d - 1 + iter$ solutions evaluated so far, are used to update the model; and (ii) if $iter \geq 25$ a subset of $11d - 1 + 25$ solutions is used, where the first half of them are the best j solutions under the prevailing scalarizing vector $\boldsymbol{\lambda}$ and the other half are selected at random without replacement. Further details of the parameter settings used in ParEGO are given in Section 6.

4 Test function suite

4.1 Notes on the selection of functions

A number of good attempts at designing test function suites and/or general schemes for test function generation have been proposed in the multiobjective optimization literature, of which those described in [4, 21, 25] are some of the best. We make a selection of nine test functions, borrowing from these, and adapting some of them slightly for our purposes. Overall, our suite contains functions from two to eight decision variables; functions with a very low density of solutions at the Pareto front; functions with locally

KNO1 [16] Features: Two decision variables; two objectives; Fifteen locally optimal Pareto fronts.

OKA1 [21] Features: Two decision variables; two objectives; Pareto optima lie on curve; density of solutions low at PF.

OKA2 [21] Features: Three decision variables; two objectives; Pareto optima lie on spiral-shaped curve; density of solutions very low at PF.

VLMOP2 [25] Features: Two decision variables; two objectives; concave PF.

VLMOP3 [25] Features: Two decision variables; three objectives; disconnected Pareto optimal set and PF is a curve 'following a convoluted path through objective space'.

DTLZ1a, adapted from [4] Features: Six decision variables; two objectives; local optima on the way to the PF.

DTLZ2a and DTLZ4a, adapted from [4] Features: Eight decision variables; three objectives; DTLZ4a biases the density distribution of solutions toward the $f_3 - f_1$ and $f_2 - f_1$ planes.

DTLZ7a, adapted from [4] Features: Eight decision variables, three objectives; four disconnected regions in the Pareto front (in objective space).

Fig. 4. Summary of the nine test functions

optimal Pareto fronts; functions where the Pareto set follows a complicated curve in the decision space; functions where the Pareto front is disconnected in objective space; and functions where the density of points parallel to the Pareto front is non-uniformly distributed. There is thus a good deal of variety in the difficulties that they pose. We have nonetheless been restrictive in some particular aspects: all functions are unconstrained and while difficult, are not overly high-dimensional (in decision space), and have a reasonable, rather than pathological degree of ruggedness. And, we have kept to functions of two and three objectives only. These restrictions accord with our description (in Section 1) of certain kinds of expensive engineering/scientific problem, where we hope to obtain good results in a very small number of function evaluations. We do not reproduce the equations of all functions here but they can be found in [16] and are summarized in Fig. 4.

5 Selected performance analysis techniques

In accordance with the analyses presented in [27], we choose the hypervolume indicator to assess the approximation sets obtained by Bin_MSOPS and ParEGO. We supplement these tabulated values and significance levels with a visual representation based on *summary attainment surfaces*, for some of the 2-objective functions.

5.1 Hypervolume indicator

The hypervolume indicator assesses the size (hypervolume or Lebesgue integral) of the region weakly dominated by an approximations set, thus larger values indicate better nondominated sets. It is "the only unary indicator we are aware of that is capable of

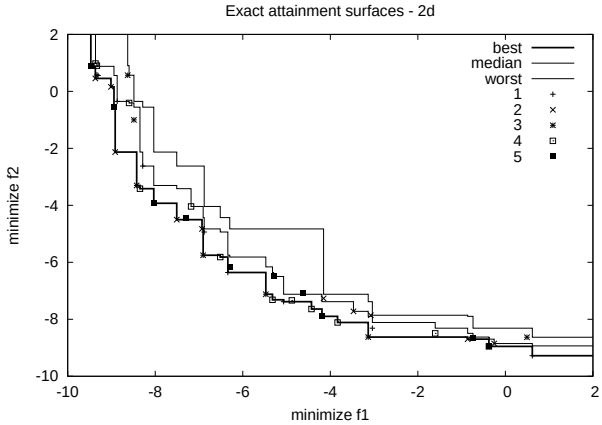

Fig. 5. Five sets of nondominated points and the best, median and worst attainment surfaces that they define. The interpretation of the median attainment surface is that, for every point on it (independently), a point (weakly) dominating this was obtained in at least 50% of the nondominated sets. Similarly, the worst attainment surface indicates the level achieved in 100% of the sets. The best attainment surface indicates the level achieved by the aggregation of all sets. *In this study, the best attainment surface is irrelevant and is* not *included in plotted results*

detecting that A is not worse than B for all pairs $A \rhd$ [better than] B" [27], where A and B are two approximation sets.

The weakly dominated region being measured must be bounded from above in some way, and for this some point b is chosen, which must be itself dominated by every point in the sample set. In order to choose a bounding point for application of the hypervolume indicator, we use the following method. First, the collection of nondominated point sets from all runs of both algorithms (on the relevant function) are aggregated into a single superset. Then, the ideal and the anti-ideal point of this superset are found. The bounding point is then the anti-ideal point shifted by δ times the range, in each objective:

$$\mathbf{b} = (b_1, b_2, \ldots, b_k), \quad \text{with}$$
$$b_j = \max_j + \delta(max_j - \min_j), \quad j \in 1..k,$$

where $\max_j$ and $\min_j$ are the maximum and minimum value, respectively, on the jth objective, found within the superset. We use $\delta = 0.01$ here.

For the analysis of multiple runs, we compute the hypervolume indicator of each individual run, and report the mean and the standard deviation of these. Since the distribution of Bin_MSOPS' and ParEGO's results are not necessarily normal, we use the Mann-Whitney rank-sum test to indicate if there is a statistically signifi cant difference in the position of the two distributions.

5.2 Median and worst summary attainment surface plots

A *summary attainment surface* is a visual way of summarizing a number of runs of a multiobjective optimizer, based on the notion of an attainment surface [7]. For illustra-

Table 1. ParEGO parameter settings, where d is the number of decision variables

Parameter	*setting*
Initial population in latin hypercube	$11d - 1$
Total maximum evaluations	250
Number of scalarizing vectors	11 (for 2 objectives), 15 (for 3 objectives)
Scalarizing function	augmented Tchebycheff
Internal GA evals per iteration	200 000
Crossover probability	0.2
Real-value mutation probability	$1/d$
Real-value SBX parameter	10
Real-value mutation parameter	50

tion, we plot five sets of nondominated points and their exact sample median, best and worst, summary attainment surfaces in Fig. 5.

For the two-objective problems in this paper, we give the median and worst attainment surfaces *only* of ParEGO and of Bin_MSOPS on the same plot, with ParEGO's two surfaces shown in solid and Bin_MSOPS's two surfaces shown with dashed lines. We do not give plots for the three-objective problems here, because of space restrictions.

6 Experimental details

To evaluate Bin_MSOPS and ParEGO on the test suite, each algorithm is run 21 times, and all solutions visited are stored. The nondominated sets achieved after a particular number of function evaluations can then be determined and used to estimate performance.

6.1 Bin_MSOPS parameter settings

Weighted Min-Max was used as the aggregation method within the MSOPS ranking algorithm. The weighted min-max score s of k objectives is calculated using (3),where λ_i is the weight for the ith objective value, f_i.

$$s = \max_{i=1}^{k}(\lambda_i f_i), \tag{3}$$

A set of objective weights constitutes a single target vector.

Thirty target vectors were used, spaced so that the angle to their nearest neighbour was constant across the set of 30. Thus in trials with 3 objectives, the set of weight vectors, although evenly spaced, was non-unique.

To choose a 'good' point, a tournament size with a maximum of 20, without replacement, was used throughout all the experiments.

A search interval (see figure 2) of $d_p = 0.02$ was used and a lower limit was set on the coverage area of allowable cells of 0.005^2 (in normalized decision space with all variables in the range [0,1]). If the cells near to the chosen 'good' point were below the cell area limit, a search was performed to find the nearest cell that is large enough to

split. The lower limit promotes a wider search around interesting points, but does prevent a tight-formation search occurring, which may be detrimental in problems with a very low density at the Pareto set (e.g. test function OKA1).

Initially, the algorithm performed a global exploration search for approximately the first 20 points, then global exploration was performed 6% of the time.

6.2 ParEGO parameter settings

The full set of parameter settings used in *all* runs of ParEGO are given in Table 1. These were determined empirically from a few exploratory trials.

7 Results

Tables 2 and 3 present the results of applying the hypervolume indicator to the 21 runs of Bin_MSOPS and ParEGO after, respectively, 100 and 250 function evaluations. Because it is not possible to use these values in comparisons with other algorithms, we make available the raw results at [15]. Note also that the appearance of the hypervolume decreasing from 100 to 250 evaluations on some problems is only due to the choice of a different bound point (see above).

Fig. 5 and 6 visualize the median and worst summary attainment surfaces of the 21 runs of both algorithms on selected 2-objective problems. Fig. 7 and 8 show the decision space points visited by the first run of the two algorithms on KNO1 and OKA1, respectively.

From these results a number of observations can be made:

- ParEGO is statistically significantly better than Bin_MSOPS on seven of the nine functions at 100 evaluations, and on six of the nine functions after 250 evaluations, under the hypervolume indicator. (cf. [16], where ParEGO was better than a standard setup of NSGA-II with population size 20 on all test functions under two different indicators).
- The standard deviations of the two algorithms are generally comparable, with a large difference evident on only one problem, DTLZ1a, at 250 evaluations (2 orders of magnitude less deviation for ParEGO). Results reported in [16] indicated that ParEGO's standard deviations were frequently one or two orders of magnitude lower than NSGA-II's on these problems, so the results here show that Bin_MSOPS is performing comparatively robustly —an important feature on problems where only one run may be possible.
- Fig. 5 and 6 indicate that ParEGO run for 250 evaluations is superior to a random search of 1000 evaluations on the difficult OKA1 and OKA2 functions, where there is a low density of solutions near the Pareto front. On OKA1, ParEGO's median attainment surface dominates all 1000 randomly generated points and on OKA2, even ParEGO's worst attainment surface does.
- Fig. 7 and 8 demonstrate that the search patterns of Bin_MSOPS and ParEGO are very different, and generally complementary. It is clear that while Bin_MSOPS attempts to get an even coverage of the search space, both globally and locally, ParEGO

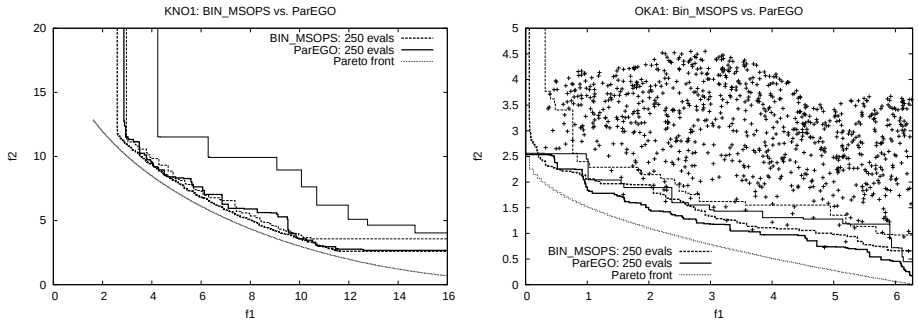

Fig. 6. Attainment surface plot on the KNO1 function after 250 function evaluations (left), and attainment surface plot with PF and 1000 random search points also shown on the OKA1 function after 250 function evaluations (right)

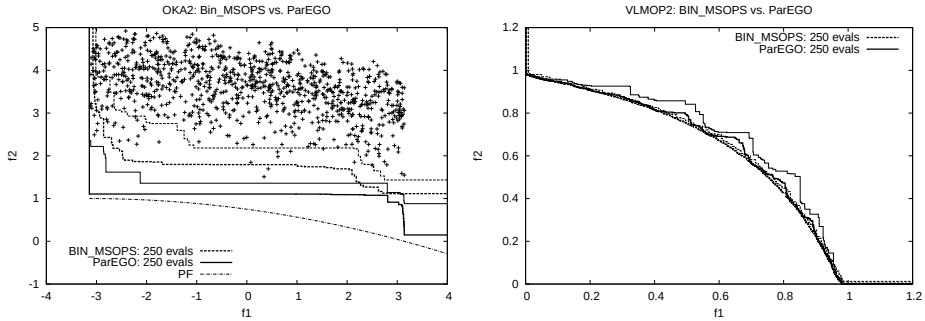

Fig. 7. Attainment surface plot on OKA2 after 250 function evaluations, with the true PF and 1000 random search points also shown (left), and attainment surface plot on VLMOP2 after 250 function evaluations (right)

operates by combining a well-spread global search with full exploitation of local 'niches' (highly fi t regions). This explains the far superior performance of ParEGO on the test functions with either low density at the Pareto front, local Pareto fronts or severe discontinuities. Fig. 7 (right) is a good example, however, of a smoother, more dense function where Bin_MSOPS has provided good even coverage, but ParEGO has focused too much on local niches. Fig. 8 really shows how ParEGO homes in on parts of the true Pareto set more aggressively, but sometimes fails to spread across it.

8 Summary and conclusion

In many optimization scenarios, the number of fi tness evaluations that can be performed is severely limited by cost or other constraints. In this study, the performance of two

Table 2. Mean and SD values of the hypervolume indicator after 100 evaluations of Bin_MSOPS / ParEGO from 21 runs of each. Larger values indicate better performance. The distributions of the values are tested using the Mann-Whitney rank sum test. The z values and significance level are indicated. ParEGO is significantly better than Bin_MSOPS unless stated

Function	Bin_MSOPS mean (SD)	ParEGO mean (SD)	z-value	significance	
OKA1	14.8496 (0.571733)	17.9532 (0.372966)	-5.546841	$> 99\%$	
OKA2	13.6773 (1.53528)	19.7183 (0.887686)	-5.546841	$> 99\%$	
KNO1	86.9879 (5.9303)	78.4856 (7.24012)	-3.735627	$> 99\%$	Bin_MSOPS wins
VLMOP2	0.317661 (0.00440428)	0.307027 (0.00564198)	-4.792169	$> 99\%$	Bin_MSOPS wins
VLMOP3	7.22865 (0.681038)	7.61652 (0.157667)	-1.547078	$> 90\%$	
DTLZ1a	185317 (2573.43)	189262 (209.001)	-5.521685	$> 99\%$	
DTLZ2a	3.85168 (0.139089)	3.97869 (0.0890044)	-3.358291	$> 99\%$	
DTLZ4a	0.533989 (0.0394657)	0.673329 (0.263526)	-2.477840	$> 99\%$	
DTLZ7a	10.5544 (1.7988)	12.9893 (0.655795)	-4.490300	$> 99\%$	

Table 3. Mean and SD values of the hypervolume indicator after 250 evaluations of Bin_MSOPS / ParEGO from 21 runs of each. Larger values indicate better performance. The distributions of the values are tested using the Mann-Whitney rank sum test. The z values and significance level are indicated. ParEGO is significantly better than Bin_MSOPSunless stated

Function	Bin_MSOPS mean (SD)	ParEGO mean (SD)	z-value	significance	
OKA1	14.8169 (0.446187)	15.9849 (0.372917)	-5.219816	$> 99\%$	
OKA2	13.9093 (1.13887)	18.4416 (0.467239)	-5.546841	$> 99\%$	
KNO1	94.972 (1.70367)	84.2247 (5.5432)	-5.521685	$> 99\%$	Bin_MSOPS wins
VLMOP2	0.324363 (0.000664073)	0.310789 (0.00398408)	-5.546841	$> 99\%$	Bin_MSOPS wins
VLMOP3	4.94459 (0.0599138)	4.75726 (0.113527)	-4.892791	$> 99\%$	Bin_MSOPS wins
DTLZ1a	63980.6 (602.738)	64869.6 (6.98978)	-5.546841	$> 99\%$	
DTLZ2a	2.79582 (0.0960357)	3.08818 (0.0276524)	-5.546841	$> 99\%$	
DTLZ4a	0.111108 (0.124952)	1.67242 (0.478067)	-5.521685	$> 99\%$	
DTLZ7a	10.6171 (1.26015)	18.1657 (0.431554)	-5.546841	$> 99\%$	

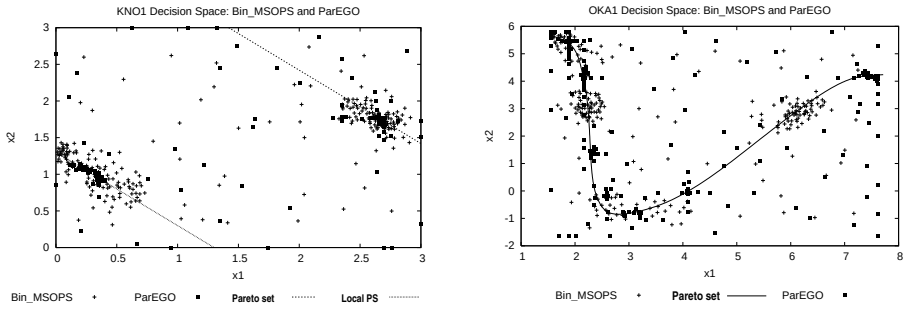

Fig. 8. Decision space points visited by Bin_MSOPS and ParEGO on KNO1 and OKA1. ParEGO fares worse on the former because it focuses too much on getting exactly on the PF instead of spreading out along it, while Bin_MSOPS extends further along it. But ParEGO's strategy does better than Bin_MSOPS's on OKA1 because points a small way off the Pareto set in decision space are far from the PF in objective space

advanced multiobjective optimization algorithms, Bin_MSOPS and ParEGO, was measured on much shorter runs than used in most previous MOEA studies. A suite of nine difficult, but low-dimensional, multiobjective test functions of limited ruggedness were used to evaluate and compare the algorithms. The results of the comparison indicated that ParEGO's use of a surrogate model to establish both the expected multiobjective cost of candidate solutions *and* the uncertainty in these predictions, enables rapid advances in the early phase of the optimization process. The less directed search performed by Bin_MSOPS is good, but can fail to capitalize effectively on previously gathered information when the solution density at the Pareto front is low.

Overall, the experiments reported here can serve as a benchmark for other algorithms aimed at these type of expensive, low-dimensional multiobjective problems. To facilitate the comparison of Bin_MSOPS and ParEGO with other methods, raw results are available at [15]. Comparisons of ParEGO with NSGA-II can already be found in [16].

Future work will focus on three possible extensions. 1. an adaptive update of the scalarizing vectors to get a better distribution on the Pareto front; 2. constraint handling mechanisms; and 3. investigation of a hybrid between Bin_MSOPS and ParEGO that might offer a good combination of computational efficiency and high performance in short-to-medium run lengths (say up to 500 evaluations) for problems where evaluations are faster but still otherwise limited.

Acknowledgments JK is supported by a David Phillips fellowship from the Biotechnology and Biological Sciences Research Council (BBSRC), UK. Thanks to J. Handl for proof-reading and ParEGO implementation advice.

References

1. D. Büche, G. Guidati, P. Stoll, and P. Kourmoursakos. Self-organizing maps for Pareto optimization of airfoils. In *Parallel Problem Solving from Nature—PPSN VII*, pages 122–131, September 2002. Springer-Verlag.
2. J.-J. Chen, D. E. Goldberg, S.-Y. Ho, and K. Sastry. Fitness inheritance in multi-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 319–326, July 2002. Morgan Kaufmann Publishers.
3. K. Deb and H. Beyer. Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation*, 9(2):197–221, 2001.
4. K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multi-objective optimization. Technical Report 112, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2001.
5. E. I. Ducheyne, B. De Baets, and R. De Wulf. Is fitness inheritance useful for real-world applications? In *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 31–42, April 2003. Springer.
6. J. R. G. Evans, M. J. Edirisinghe, and P. V. C. J. Eames. Combinatorial searches of inorganic materials using the inkjet printer: science philosophy and technology. *Journal of the European Ceramic Society*, 21:2291–2299, 2001.
7. C. M. Fonseca and P. J. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In *Parallel Problem Solving from Nature—PPSN IV*, pages 584–593, 1996. Springer-Verlag.

8. A. Gaspar-Cunha and A. Vieira. A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. *International Journal of Computers, Systems, and Signals*, 2004 (in press).
9. A. Gaspar-Cunha and A. S. Vieira. A hybrid multi-objective evolutionary algorithm using an inverse neural network. Hybrid Metaheuristics (HM 2004) Workshop at ECAI 2004, pages 25-30, 2004. http://iridia.ulb.ac.be/ hm2004/proceedings/
10. M. P. Hansen and A. Jaszkiewicz. Evaluating the quality of approximations of the nondominated set. Technical Report IMM-REP-1998-7, Technical University of Denmark, 1998.
11. E. J. Hughes. Multi-objective binary search optimisation. In *Second International Conference on Evolutionary Multi-Criterion Optimisation, EMO'03*, pages 102–117, April 2003. Springer.
12. E. J. Hughes. Multiple single objective Pareto sampling. In *Congress on Evolutionary Computation 2003*, pages 2678–2684, December 2003. IEEE.
13. Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, 2002.
14. D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.
15. Knowles' webpage. http://dbk.ch.umist.ac.uk/knowles/
16. J. Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. Technical Report TR-COMPSYSBIO-2004-01, University of Manchester, UK, 2004. Available from http://dbk.ch.umist.ac.uk/knowles/pubs.html
17. M. Laumanns and J. Ocenasek. Bayesian optimization algorithms for multi-objective optimization. In *Parallel Problem Solving from Nature—PPSN VII*, pages 298–307, September 2002. Springer-Verlag.
18. R. Myers and D. Montgomery. *Response Surface Methodology*. Wiley, New York, 1995.
19. P. K. S. Nain and K. Deb. A computationally effective multi-objective search and optimization technique using coarse-to-fine grain modeling. Technical Report Kangal Report No. 2002005, IITK, Kanpur, India, 2002.
20. S. O'Hagan, W. Dunn, M. Brown, J. Knowles, and D. Kell. Closed-loop, multiobjective optimization of analytical instrumentation: gas chromatography/time-of-flight mass spectrometry of the metabolomes of human serum and of yeast fermentations. *Analytical Chemistry*, 2004 (in press). http://pubs.acs.org/cgi-bin/asap.cgi/ancham/asap/html/ac049146x.html
21. T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff. On test functions for evolutionary multiobjective optimization. In *Parallel Problem Solving from Nature VIII*, pages 792–802, September 2004, Springer.
22. J. Sacks, W. Welch, T. Mitchell, and H. Wynn. Design and analysis of computer experiments (with discussion). *Statistical Science*, 4:409–435, 1989.
23. R. E. Steuer and E.-U. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 25:326–344, 1983.
24. S. Vaidyanathan, D. I. Broadhurst, D. B. Kell, and R. Goodacre. Explanatory optimization of protein mass spectrometry via genetic search. *Analytical Chemistry*, 75(23):6679–6686, 2003.
25. D. A. V. Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm test suites. In *Proceedings of the 1999 ACM Symposium on Applied Computing*, pages 351–357, 1999. ACM.
26. D. Weuster-Botz and C. Wandrey. Medium optimization by genetic algorithm for continuous production of formate dehydrogenase. *Process Biochemistry*, 30:563–571, 1995.
27. E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.