ParEGO: A Hybrid Algorithm with On-line Landscape Approximation for Expensive Multiobjective Optimization Problems

Joshua Knowles
University of Manchester (formerly UMIST), UK
j.knowles@manchester.ac.uk

Technical report TR-COMPSYSBIO-2004-01

September 2004

Abstract

Many optimization scenarios, particularly those arising in the experimental sciences and engineering, involve fitness evaluations that are financially and/or temporally expensive. In such applications, the computational overhead of the optimization algorithm is effectively irrelevant, whereas the quality of solutions obtained after *realistically small* numbers of function evaluations assumes primary significance, with worst-case performance, as well as average-case, being especially pertinent. These simple considerations apply equally to *multiobjective optimization* as to optimization of a single objective function, yet nearly all multiobjective evolutionary algorithm (MOEA) studies (even some of those relating to expensive fitness functions) report performance after tens of thousands of evaluations, leaving open the question of performance on much shorter runs. In this paper, we concentrate on nine relatively low-dimensional, non-pathological, real-valued functions, such as arise in many applications, and assess the performance of two algorithms at just 100 and 250 function evaluations. The results show that NSGA-II, a popular MOEA, performs well compared to random search, even within the restricted number of evaluations used. A significantly better performance (particularly in the worst case) is, however, achieved on our test set by an algorithm proposed herein — ParEGO — which is an extension of the single-objective *efficient global optimization* (EGO) algorithm of Jones et al. ParEGO uses a design-of-experiments inspired initialization procedure and learns a Gaussian processes model of the search landscape, which is updated after every function evaluation. Overall, ParEGO exhibits a promising performance for multiobjective optimization problems where evaluations are expensive or otherwise restricted in number.

Keywords: multiobjective optimization, expensive black-box functions, EGO, DACE, NSGA-II, landscape approximation, response surfaces, Pareto optima, test suites, performance assessment

1 Introduction

As scientists and engineers become increasingly aware of the potential benefits of applying heuristic search methods in their disciplines, a new and greater variety of real-world optimization problems is gradually coming to light. Many of the problems fall right into the 'backyard' of evolutionary computation (EC) methods — especially those for which conventional techniques are not easily adapted, including non-convex, mixed integer, non-linear, constrained and/or noisy cost functions; for all these problems we may happily look forward to a wealth of EC success stories. But some other problems now appearing seem to pose a particular challenge to evolutionary computation and other heuristic search methods because they combine the requirement of optimizing multiple, incommensurable objectives with the feature of being prohibitively expensive to evaluate. It is on these latter problems that we focus in this paper.

Beyond being expensive and multiobjective, a number of recent optimization scenarios arising in the experimental sciences have a more-or-less common list of features that we can take to loosely specify a class of problems, as follows:

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 locally smooth but multimodal,
8. the dimensionality of the search space is low-to-medium,
9. the problem has multiple, incommensurable objectives.

Problems exhibiting these features include various combinatorial biochemistry and materials science applications [12, 19, 52], as well as instrument set-up optimization [48, 49] in analytical

chemistry, for improving the separation and identification of numerous chemical compounds in biological samples.

For example, [39] reports a a series of experiments, directed at improving *metabolomics*[1], where the settings of a GC-TOF mass spectrometer (GC-MS) were optimized. Three objectives in the optimization were considered: (i) maximizing the signal-to-noise ratio in the chromatogram; (ii) maximizing the number of 'true' peaks in the chromatogram, and (iii) minimizing the processing time — the time for the instrument to analyse one sample. Features 1–3 in the list given above arise in this particular problem because the evaluations are actually 'wet experiments'[2] that must be performed on a physical machine (which is itself very costly and hence only one is available), and may also require costly consumables and/or the use of highly-skilled operators. Feature 4 applies here because it is practically infeasible to simulate accurately the output of such a complicated instrument over a range of chemical inputs. Thus, considering 1-4, it is clear that the EA or other optimization method *must* search the space in a smaller number of evaluations than in many other EA applications.

Fortunately, features 5–8 reduce the difficulty of the task somewhat. Low noise means that individual experiments need not be repeated, and there were only eight real-valued settings of the GC-MS (i.e. search space dimensions). Moreover, it is known from human experts that getting the configuration 'right' yields significant improvement in the chromatograms; since this is usually achieved from hand-tuning by experts the search space must be, at least locally, smooth. Therefore, reasonably large basins of attraction probably exist. However, it is also known that some limited epistasis does exist [48].

In the experiments reported in [39], a multiobjective evolutionary algorithm (MOEA), PESA-II [10], was used to perform the optimization [39] and some 180 GC-MS configurations were assayed in all (taking several days and consuming expensive instrument-time). At the end, a particular configuration that offered the best compromise solution was chosen from the Pareto front obtained. This yielded a threefold increase in the number of peaks observed, coupled with a small increase in the potential throughput (i.e., future samples could be processed more quickly), over the hand-tuned configuration usually employed. The choice of employing an MOEA was motivated primarily

[1] Metabolomics is a science in which chemical samples containing hundreds or thousands of different compounds must be identified efficiently to enable biological processes to be accurately monitored

[2] The term used in biology for laboratory as opposed to computer-based experiments.

by the feature 9 above, and the wish to explore the tradeoffs of the three objectives, enabling the input of expert knowledge in the final choice of GC-MS configuration. The ability to do this from a single optimization run, with little or no knowledge of the cost landscape, makes an MOEA approach a logical choice. Moreover, results reported in [39] show that GC-MS configurations from later experiments were significantly better than earlier configurations, suggesting that the MOEA used was significantly better than a random search would have been.

Nonetheless, the scientific justification for the use of MOEAs in this particular optimization context is not as solid as we would like because, although MOEAs have been applied in a wealth of different contexts and have exhibited competitive performance against several other optimization techniques, their performance when the number of function evaluations is severely limited is largely untested in the literature (though see [30] for some experiments underpinning the use of PESA-II for the GC-MS optimization). *A central aim of this paper* is thus to undertake testing of an MOEA in contexts similar to the GC-MS configuration problem, over a *realistically small number* of function evaluations. To this end, we choose to employ NSGA-II, a well-respected, modern example of an MOEA[3], and test it, using a small population size, over just 100 and 260 function evaluations.

In many engineering applications that share similar features to our scenario above, standard evolutionary algorithms are eschewed because of their low efficiency with respect to function evaluations. A wealth of other global optimization methods (see next section) use more principled methods of exploring the search space under these restricted conditions, most notably those methods based on modeling the landscape on-line during the search. However, to our knowledge, the performance of these methods has never been compared with an MOEA in a straightforward Pareto optimization context. Thus, *a second aim of this paper* is to extend such an algorithm to do multi-objective optimization, and to compare it with the NSGA-II. We choose for this, a frequently cited algorithm from the global optimization literature, specifically designed for expensive functions of low dimension — the *efficient global optimization* (EGO) algorithm [27]. The version that we propose for Pareto optimization is given the designation 'ParEGO' from here on.

To undertake the performance assessment of ParEGO and NSGA-II, we do not use expensive functions here, but employ a suite of test functions enabling us to collect performance data

[3]The IEEE TEC paper describing NSGA-II for multi-objective optimization was judged as the 'Fast-breaking Paper in Engineering' by Web of Science (ESI) in February 2004

over multiple algorithm runs. The selection of problems in the suite includes functions exhibiting a diverse range of problem difficulties, yet is restricted to problems that we think model scientific/engineering contexts like the GC-MS optimization problem. For this reason, the problems are relatively low-dimensional (having up to 8 real-valued dimensions), are not pathologically rugged, and have either two or three objectives. In measuring performance, we give information on the worst-case as well as the average case performance, as this is of particular relevance in scenarios where only one shot at the experiments/optimization is usually possible. The choice of test functions and the methods for measuring performance are such that they should provide a sound basis for future comparison of other algorithms for expensive, low dimensional multiobjective problems. To facilitate this, problems, methods and results are available for download at [29].

The rest of the paper is organized as follows. In Section 2, a review of relevant literature in engineering optimization methods, landscape approximation, and MOEAs, is given. An outline of EGO, a powerful global optimization method for expensive functions, is provided in Section 3, together with results demonstrating our implementation of the approach. Section 4 extends EGO to the multiobjective algorithm, ParEGO, which we propose here. Section 5 describes the choice of test functions, and Section 6 details our methods for performance assessment and comparison. Section 7 sets out the method of comparison, giving all parameter settings, Section 8 presents the results, and Section 9 provides a summary and conclusion.

2 Related work

The rise of evolutionary algorithms for multiobjective optimization in recent years is now well documented, with thorough reviews of the history and current state of the art available in [14, 7, 5]. Much of the success that the field is enjoying relies on the flexibility with which MOEAs can deal with various optimization contexts and features, such as: high dimensionality; integer, real and mixed parameters; lack of knowledge about ranges of fitness functions; non-differentiability of the cost landscape; constraints, and so on. The initial high computational overhead of some early methods has also been overcome by a gradual progression from algorithms such as NPGA [23] and PAES [33], which first attempted to use more efficient niching policies, through to the Micro-GA [6] and the popular NSGA-II [15]. More efficient data-structures that can be used to speed up the

niching, selection and archiving processes that these MOEAs rely upon are also now available [25].

In engineering, MOEAs have found many applications [13], from airplane wing optimization [38] and spacecraft trajectory planning [11] to the design of irrigation networks [4]; several of these obviously entail expensive simulation or experimental steps. Nonetheless, and perhaps due to the focus made on improvements to MOEA's computational overhead *per evaluation*, their effectiveness in applications where very few function evaluations can be afforded has only recently been investigated in a few isolated papers. In [24], for example, two MOEAs that sample the search space by making explicit use of all previously visited points were proposed and evaluated on three simple test functions. These methods appear promising, outperforming more conventional MOEAs on the test set used, and do not seem to be restricted to any particular type of fitness function, so long as it is expensive to evaluate. Nonetheless, it would seem that further savings of precious function evaluations could be made by *learning* the cost landscape from all previously visited points, and by using this to estimate the 'best' place to sample next (either to improve the model the most or to obtain a better solution, or both). Learning a cost landscape from a set of solution/cost pairs is variously called surrogate, approximate or meta- modeling in the literature. The idea of selecting the next solution to sample in order to maximize the improvement in the model is known as *active learning* [8] in the machine intelligence literature. These approaches have received little attention in evolutionary multiobjective optimization but meta-modeling (at least) has a relatively rich history in optimization of a single objective.

In design engineering, meta-modeling is usually known as the response surface method [36], 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 parameterised by maximum likelihood estimation. A particular example of this is known as the Design and Analysis of Computer Experiments (DACE) model [45], which forms the basis of the EGO algorithm, described briefly in the next section. Optimization methods using response surfaces and/or Kriging meta-models have been successfully used in aeronautical design/engineering applications where only tens or hundreds of function evaluations are possible [51]. In the case of EGO, four low-dimensional multimodal test functions have been optimised to within 1% of optimal in the order of 100 function evaluations [27].

In the evolutionary algorithm community, response surface and Kriging methods have also been

used within EAs to reduce the number of expensive function evaluations that need to be carried out during optimization [2, 35, 44]. In addition, similar goals have been obtained using artificial neural networks (either MLPs or RBF networks), or by a rather simpler approach called fitness inheritance [46], where some fraction of offspring are not evaluated, but are assigned a fitness value estimated from the fitness (and proximity) of their parents. Detailed reviews of all these meta-model based EAs can now be found in recent papers [42, 26], along with new approaches that try to manage the meta-models optimally with respect to the conflicting concerns of optimization and the design of experiments (i.e. improving the model) [41], [42]. Finally, a related but somewhat different approach was proposed recently in [9], in which optimization of an expensive function is tackled by modeling fitness transitions using a finite state machine and, offline, running a number of different evolutionary algorithms on the model, in order to choose the most efficient one. This approach may be attractive when one has many optimization tasks to do on the same or very similar landscapes but a certain investment is needed to build up a sufficiently good model.

Returning to the case of multiple objective functions, a handful of papers have considered the use of neural network-based meta-models in MOEAs. The study in [37] demonstrates the use of a neural network approximation combined with the NSGA-II algorithm; the neural net having k outputs, one for each objective function. In this approach, generations using the real evaluation function are alternated with generations using a multi-layer perceptron model of the function derived from the samples collected from the earlier generations. Some speedup is observed over using the original exact fitness function alone, but the study is limited to a single, curve-fitting problem. More recently, [21] and [22] propose and compare two more methods employing neural networks. The first method follows Nain and Deb's approach, though with a different MOEA being used. The second method is a significant departure from most other meta-modeling schemes, however: an inverse neural network is used to map back from a desired point in the objective space (beyond the current Pareto front) to an estimate of the decision parameters that would achieve it. Test function results presented in [22] for the latter look particularly promising, though fairly long runs (of 20000 evaluations) are considered and it is not clear that on much shorter runs the method would offer significant gains. Finally, a different type of neural network, a self-organizing map, has been employed in [1] to replace standard variation operators with adaptive ones. However, so far, tests have not explicitly compared the performance of this approach with other methods.

The concept of fitness inheritance has also been extended to the multiobjective case. In [3], convergence and population-sizing are considered theoretically, with results limited to a simple multiobjective OneMax problem on which it was found only modest savings were possible. A later empirical study [18] revealed performance had a significant test function dependence, with the suggestion that on functions with a convex and continuous Pareto front, large savings might be possible using fitness inheritance, while this may not be the case on non-convex or discontinuous Pareto fronts.

Related to, but somewhat different from, explicit modeling of search landscapes, is the use of Bayesian network and/or other probabilistic model-building algorithms. These methods generate solutions from estimates of the distributions and/or joint distributions of decision parameters observed in highly fit solutions. A number of multiobjective versions have now been proposed [34], [28], [47] and we await their application in real problem domains. However, these algorithms seem to work rather 'tentatively', and require a large number of function evaluations before performance gains are seen; no study has shown that they can be used effectively to increase performance on runs of fewer than 1000 function evaluations.

3 The EGO algorithm

The EGO algorithm for global optimization of expensive black-box functions was first introduced and described in [27]. It makes use of 'Kriging' to model the search landscape from solutions visited during the search. More specifically, it exploits the *design and analysis of computer experiments* (DACE) model of [45], based on Gaussian processes. We do not give a mathematical description of EGO or DACE here, but simply provide some motivation for its choice in this study, and a brief description of the main procedures involved. The reader is referred to [27] for a fuller explanation.

The DACE model, like any other method of supervised learning, is subject to certain no-free-lunch theorems [53], similar to those for optimization, that means its general application cannot be recommended on any theoretical basis. Nonetheless, on functions where we expect a degree of local smoothness, where noise is low, and the number of dimensions is not excessive, the DACE model seems to be a good choice for building the kind of approximation that is needed by a search algorithm restricted to a low number of function evaluations. In particular:

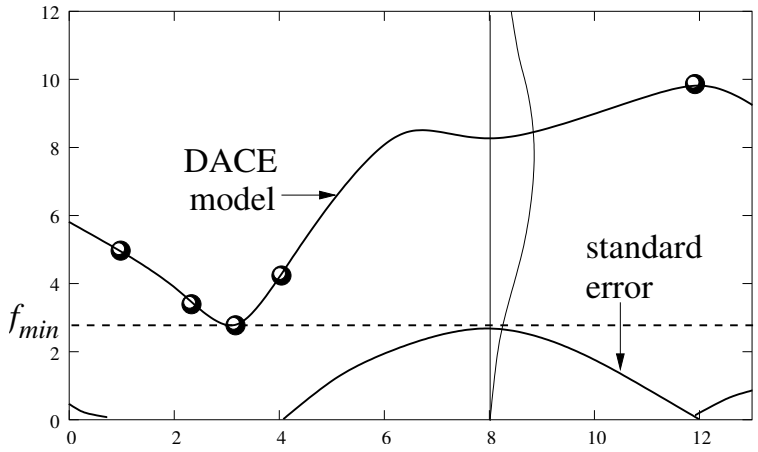

Figure 1: The uncertainty about the function's value at a point (such as x=8 above) can be treated as if the value there were a realization of a normal random variable with mean and standard deviation given by the DACE model and its standard error [Adapted from [27]]

- The model always interpolates the points (independently of the chosen parameters of the model)
- The model has a small, fixed number of parameters: $2d+2$ of them, where d is the dimension of the function to be optimized
- The likelihood of the model given the data has a simple closed form expression from which it is possible to compute the maximum likelihood model — giving the approach a sound statistical interpretation
- The error in the expected cost of a solution also has a simple, closed form expression. Thus, the model estimates its own uncertainty.

The last point above is of particular relevance. Knowledge of the error in the response surface is a useful property when searching a cost landscape, and EGO makes use of this property explicitly, as we explain below.

The EGO algorithm begins by first generating a number of solutions in a latin hypercube (which is similar to a fractional factorial design), and by then finding 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 maximises what Jones et al [27] call "the expected improvement"— the part of the curve of the standard error in the model that lies

below the best cost sampled so far (see Figure 1). This effectively means that EGO weighs up both the predicted value of solutions, *and the error in this prediction*, in order to find the one which 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. Importantly, the EGO algorithm uses a closed form expression for the expected improvement and it is thus possible to search the decision space globally for the solution that maximises this. 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.

Computing the maximum likelihood model at each step of EGO requires several hundred or thousand matrix inversions on a square matrix of dimension equal to the number of solutions so far sampled. As such, it is rather expensive in terms of computation per evaluation, and this overhead increases with time. However, if EGO is efficient with respect to the number of function evaluations needed to reach a given cost, the computational cost can be considered irrelevant in many optimization contexts.

Before we go on to explain how we extend EGO to the multiobjective optimization case, we first give details of our own implementation of the basic algorithm, and some illustrative test results.

3.1 Implementation of EGO

EGO was implemented in C using the open source matpack library[4] for coding the matrix operations. The initial solutions are generated using a latin hypercube routine, following a description in Numerical Recipes [43]. The number of initial solutions is set to $11d - 1$, where d is the dimension of the function to be optimized, as suggested in [27].

In order to optimise the likelihood function, the Nelder and Meads downhill simplex routine from Numerical Recipes [43] has been used. This is restarted twenty times in order to give a more robust search of the model space. In order to find the best solution to visit next, a genetic algorithm is embedded within EGO, to search the decision parameter space globally. We have not used branch

[4] www.matpack.de

and bound to find the best solution to visit, as suggested by Jones et al., simply because of the difficulty of implementing it, particularly for the more complicated case of constrained optimization, which we would like to tackle in future.

Because the computational overhead increases with each function evaluation, we have found it necessary to cap the size of the correlation matrix (on which the model is based) to have a maximum dimension of 80. This means that beyond the 80th function evaluation, 80 solutions are selected at random without replacement from the list of all previously visited solutions — and the DACE model for that iteration is based on these. Potentially, this reduces the accuracy and reliability of EGO (potentially causing it to revisit points). However, beyond matrix sizes of about 80-100, EGO was taking several minutes per evaluation (and this begins to increase dramatically). For testing, this is prohibitive, although in practice a function evaluation can easily take much longer than this. In ParEGO, we also cap the size of the model but use a slightly more advanced selection procedure, as described in Section 4.

3.2 Testing the implementation of EGO

Testing of our implementation of EGO was carried out on multimodal (epistatic) functions, up to a dimension of about 10, taken from [35]. On each function, we ran EGO (51 times) for just $20d$ evaluations (where d is the dimension of the function). In order to get some idea of the quality of these results, we compare with a random search algorithm (Tables 1 and 2), and with the published results of Liang et al [35], who implement an evolutionary algorithm also with landscape approximation (Table 3). The former results are generally more useful: these allow one to estimate the quality of solutions reached by EGO in terms of the number of solutions in the search space that have that cost or a lower one. The results show that EGO is up to 250 times faster than random search (on four of the functions), more than 50 times faster on a further four, and, at worst, about the same as random search on one highly multimodal function.

The results presented in Table 3 are useful only in that they *suggest* EGO is performing very well on *some* functions (far better than the EA of Liang et al); but in general it is hard to make comparisons because Liang et al's algorithm is run for 1 to 3 orders of magnitude longer (in terms of evaluations) than we run EGO for.

Function	d	$\#opt$	EGO ($20d$ evals)	Random search	(#evals)	EGO wins	No diff
f_{14}	2	25	10.7634	54.696730	($20d$)	2X (99%)	5X (90%)
				17.379662	($40d$)		
				10.785819	($100d$)		
				2.054400	($1000d$)		
				1.022432	($5000d$)		
f_{15}	4	∞	0.0313316	0.068178	($20d$)	2X (99%)	5X (90%)
				0.046495	($40d$)		
				0.025082	($100d$)		
				0.004304	($1000d$)		
				0.002271	($5000d$)		
f_{16}	2	6	-1.03149	-0.003015	($20d$)	250X (99%)	–
				-0.247832	($40d$)		
				-0.680195	($100d$)		
				-1.005488	($1000d$)		
				-1.024494	($5000d$)		
f_{17}	2	3	0.400733	2.717814	($20d$)	100X (99%)	250X (90%)
				1.824595	($40d$)		
				1.522864	($100d$)		
				1.411058	($1000d$)		
				1.401838	($5000d$)		
f_{18}	2	4	4.56758	36.718825	($20d$)	5X (95%)	50X (99%)
				22.858280	($40d$)		
				6.779331	($100d$)		
				3.341144	($1000d$)		
				3.112699	($5000d$)		
f_{19}	3	4	-3.86258	-3.556176	($20d$)	250X (99%)	–
				-3.730429	($40d$)		
				-3.763582	($100d$)		
				-3.835820	($1000d$)		
				-3.856405	($5000d$)		

Table 1: Results of running our implementation of EGO on our 13 test functions (6 are shown above — the remaining 7 are in Table 2). Column 1 gives the name of the function, while columns 2 and 3 give, respectively, its dimension d and its number of local optima. Column 4 shows the median best evaluation reached from 51 runs of EGO, where the number of function evaluations used in each run, is just $20d$, where d is the dimension of the function. Columns 5 and 6 show the median evaluation reached by 51 runs of random search for the number of function evaluations indicated. The median value which is closest to EGO's is shown in bold font. Columns 10 and 11 summarise the results of Mann-Whitney rank-sum tests between EGO and the random search algorithm. The "EGO wins" column gives the number of iterations that a random search can be run (in terms of a multiple of EGO's number of iterations), and still be statistically beaten by EGO (with highest associated confidence level). The final column indicates the same thing, but where no difference between the algorithms' performance can be found (and the *lowest* confidence level at which the test *would* have found a difference had there been one). On five test functions, EGO is 250 times faster than random search, on four others it is between 50 and 250 times faster. An interpretation of the results is that, e.g. on f_{17}, EGO finds a function value that is shared or bettered by less than $1/50000d = 0.001\%$ of the entire search space, in just 40 function evaluations. EGO generally performs worst on the functions with very many local optima, though it does well on both the generalised Griewank and Schwefel funtions (here with 6 and 10 dimensions, respectively, rather than the usual 30 — see Table 2)

Function	d	$\#opt$	EGO ($20d$ evals)	Random search	(#evals)	EGO wins	No diff
f_{20}	6	4	-3.30147	-2.016387	($20d$)	250X (99%)	–
				-2.372988	($40d$)		
				-2.607558	($100d$)		
				-3.011547	($1000d$)		
				-3.093685	($5000d$)		
f_{21}	4	5	-2.43167	-0.536953	($20d$)	50X (95%)	100X (90%)
				-0.656933	($40d$)		
				-0.908331	($100d$)		
				-2.296372	($1000d$)		
				-3.356989	($5000d$)		
f_{22}	4	7	-2.51587	-0.762930	($20d$)	50X (95%)	100X (90%)
				-1.029108	($40d$)		
				-1.468311	($100d$)		
				-2.264707	($1000d$)		
				-3.589095	($5000d$)		
f_{23}	4	10	-2.69769	-0.966357	($20d$)	50X (90%)	100X (90%)
				-1.106497	($40d$)		
				-1.463071	($100d$)		
				-2.613725	($1000d$)		
				-3.741701	($5000d$)		
f_{24}	2	∞	0.345967	**0.319360**	($20d$)	–	1X (90%)
				0.203386	($40d$)		
				0.179859	($100d$)		
				0.038538	($1000d$)		
				0.014670	($5000d$)		
Griewank	6	?	1.0205	41.558208	($20d$)	250X (99%)	–
				30.257815	($40d$)		
				24.441566	($100d$)		
				11.717705	($1000d$)		
				6.968934	($5000d$)		
Schwefel	10	$> 10^9$	-4186.97	-3745.924	($20d$)	250X (99%)	–
				-3812.0805	($40d$)		
				-3900.232	($100d$)		
				-4025.030	($1000d$)		
				-4079.477	($5000d$)		

Table 2: The results on the remaining six test functions. See Table 1 caption. The number of local optima in the Griewank function is not known — though for a 30-dimensional version of the same function it is $\simeq 10^{54}$.

Function	EGO			EANA	
	#evals	mean	best	mean #evals	mean
f_{14}	40	20.7267	0.998029	1685	7.36
f_{15}	80	0.0309024	0.00157531	31645	0.000307
f_{16}	40	-1.03019	-1.031628	863	-1.031628
f_{17}	40	0.438083	0.3979	945	0.3979
f_{18}	40	12.8115	3.00007	822	3
f_{19}	60	-3.8624	-3.86278	1297	-3.86
f_{20}	120	-3.2562	-3.32135	8357	-3.3128
f_{21}	80	-2.72465	-6.60786	2958	-9.6512
f_{22}	80	-2.70663	-7.66332	2880	-9.1430
f_{23}	80	-3.32506	-10.276	3118	-9.5767
f_{24}	40	0.330584	0.0393655	1181	0.0297

Table 3: Comparison of EGO with published results of running EANA [35] on the 11 lower-dimensional functions. For EGO, we ran for exactly $20d$ function evaluations (column 2) and quote the mean and best cost found from 51 runs (columns 3 and 4, respectively). For EANA, the algorithm was stopped when it reached either the global optimum (up to the precision of the computer) or some unspecified evaluation limit [35]. Thus, the mean number of evaluations is reported (column 5) and the mean cost of the solution over the 50 runs performed (last column). Comparison is difficult but on some functions, EGO occasionally reaches comparable fitness levels one or two orders of magnitude faster than the mean number of function evaluations quoted for EANA. (EANA is an evolution strategy, also based on building landscape approximations)

4 Extending EGO to the multiobjective case: ParEGO

The EGO algorithm could be extended for use with multiobjective optimization problems in a number of different ways. The simplest approach, which we investigate here, converts the k different cost values of a solution into a single cost via a parameterised scalarizing weight vector. By choosing a different (parameterisation of the) weight vector at each iteration of the search, an approximation to the whole Pareto front can be gradually built up.

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 defined by:

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

with $|\Lambda| = \binom{s+k-1}{k-1}$, so that the choice of s determines how many vectors there are in total. The

scalar cost of a solution is then computed using the augmented Tchebycheff function:

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

where ρ is a small positive value which we set to 0.05. The scalar costs of all previously visited solutions is computed and, using all or a selection of these, a DACE model of the landscape is constructed by maximum likelihood. The solution which maximises 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.

Pseudocode for the entire ParEGO algorithm is given in Algorithm 1. As in our implementation of EGO, 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 which maximises the expected improvement (line 28) is implemented as follows.

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 [16] with probability 0.2, producing one offspring

Mutation: parameter value shifted by $\pm 1/100.\mu.$(parameter range), where μ is drawn uniformly at random from $(0.0001, 1)$, 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[5] 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 parameter space

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

Algorithm 1 ParEGO pseudocode

```
 1: procedure ParEGO(f, d, k, s)
 2:     xpop[] ← LatinHypercube(d)          /* Initialize using procedure defined in 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:     initialise 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
```

In practice, on a very expensive cost function, all solutions evaluated should be used to update the DACE model, at every iteration. However, 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 Table 5.

5 Test function suite

5.1 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 [17, 50, 40] are some of the best. The functions in [17] are very popular, partly because they are scalable in the number of objectives and parameters, and partly because each function is accompanied by a description explaining exactly what difficulty it poses to an MOEA, and also allowing this to be easily tuned. However, while it is tempting to employ this suite wholesale, there is (at least) one drawback to it, which leads us to include functions from [50] and [40] as well.

The drawback in Deb et al's suite is that most of the functions work on a scheme where only $k-1$ parameters control the position *parallel* to the Pareto front in the objective space, with all the remaining parameters controlling distance *to* the front. Thus, a natural bias is given to recombination-based EAs because, once a few points on the true Pareto front are found, recombination tends to propagate and preserve the parameters which are common to them (which is all of them, bar the first $k-1$), while searching over the other parameters, leading to a very fast spreading out over the entire front. Unfortunately, many real-world multiobjective optimization problems are not structured like this; rather, each different optimal tradeoff solution requires changing all or many of the decision parameters.

In [40], exactly the same observation as ours above was made in respect of the Deb functions,

and furthermore, it was also observed that the Pareto fronts of many other test functions consist of piecewise linear curves and/or single points in the parameter space. The paper goes on to propose a scheme for generating functions, that like Deb's, allows some different problem difficulties to be incorporated and tuned, combined with a greater control of the curves/surfaces in parameter space comprising the Pareto front. Two example functions using this general scheme are given, and it is these that we employ in our suite.

The functions we employ from [50] are not part of an overall tunable scheme but have been popular in the literature. VLMOP2 has a concave Pareto front while VLMOP3's Pareto front is both nonlinear and asymmetric.

Overall, the final selection of nine test functions includes functions from two to eight decision parameters; functions with a very low density of solutions at the Pareto front; functions with locally optimal Pareto fronts; functions where the Pareto set follows a complicated curve in the parameter 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 parameter 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.

In the following we describe each of our test functions in turn.

5.2 KNO1

This test function is the only one not borrowed from the existing literature. There are just two parameters and two objectives in the problem:

$$\begin{aligned}
&\text{Minimize } f_1 = 20 - r\cos(\phi)\\
&\text{Minimize } f_2 = 20 - r\sin(\phi)\\
&\qquad r = 9 - [3\sin(5/2(x_1+x_2)^2) + 3\sin(4(x_1+x_2)) + 5\sin(2(x_1+x_2)+2))]\\
&\qquad \phi = \pi/12(x_1 - x_2 + 3)\\
&x_1, x_2 \in [0,3]
\end{aligned}$$

The distance from the Pareto front, controlled by r, is a function of the sum of the two decision parameters, while the location transverse to the Pareto front is controlled by the difference between the two parameters. Thus, each point on the Pareto front is unique in both parameters: the Pareto front is made up of all pairs of parameters that sum to 4.4116. The true Pareto front has a greater extent than, and lies just beyond, a locally optimal Pareto front with a much larger basin of attraction. Altogether, there are fifteen locally optimal fronts in this function.

5.3 OKA1

The OKA1 test function [40] is defined as:

$$\begin{aligned}
&\text{Minimize } f_1 = x_1',\\
&\text{Minimize } f_2 = \sqrt{2\pi} - \sqrt{|x_1'|} + 2|x_2' - 3\cos(x_1') - 3|^{\frac{1}{3}},\\
&\qquad x_1' = \cos(\pi/12)x_1 - \sin(\pi/12)x_2,\\
&\qquad x_2' = \sin(\pi/12)x_1 + \cos(\pi/12)x_2,\\
&\qquad x_1 \in [6\sin(\pi/12), 6\sin(\pi/12) + 2\pi\cos(\pi/12)],\\
&\qquad x_2 \in [-2\pi\sin(\pi/12), 6\cos(\pi/12)].
\end{aligned}$$

The Pareto optima (in the parameter space) lie on the curve, $x_2' = 3\cos(x_1') + 3 \quad (x_1' \in [0, 2\pi])$. In addition the density of solutions falls away near to the Pareto front.

5.4 OKA2

The OKA2 test function [40] is defined as:

$$\begin{aligned}
&\text{Minimize } f_1 = x_1, \\
&\text{Minimize } f_2 = 1 - \frac{1}{4\pi^2}(x_1 + \pi)^2 + |x_2 - 5\cos(x_1)|^{\frac{1}{3}} + |x_3 - 5\sin(x_1)|^{\frac{1}{3}} \\
&\qquad x_1 \in [-\pi, \pi], \quad x_2, x_3 \in [-5, 5].
\end{aligned}$$

The Pareto optima lie on a spiral-shaped curve in the 3-dimensional parameter space. In addition the density of solutions falls away steeply near to the Pareto front.

5.5 VLMOP2

Veldhuizen and Lamont's MOP2 function [50] is scalable in the number of decision variables. We use just two:

$$\begin{aligned}
&\text{Minimize } f_1 = 1 - \exp\left(-\sum_{i=1}^{n}(x_i - 1/\sqrt{n})^2\right), \\
&\text{Minimize } f_2 = 1 + \exp\left(-\sum_{i=1}^{n}(x_i - 1/\sqrt{n})^2\right), \\
&\qquad x_1, x_2 \in [-2, 2], n = 2.
\end{aligned}$$

The Pareto front is concave and the Pareto optima lie on the diagonal passing from $-1/\sqrt{n}, -1/\sqrt{n}$ to $1/\sqrt{n}, 1/\sqrt{n}$ in parameter space.

5.6 VLMOP3

Veldhuizen and Lamont's MOP3 [50] is a three-objective function of two decision variables:

$$\begin{aligned}
&\text{Minimize } f_1 = 0.5(x^2 + y^2) + \sin(x^2 + y^2), \\
&\text{Minimize } f_2 = \frac{(3x - 2y + 4)^2}{8} + \frac{(x - y + 1)^2}{27} + 15, \\
&\text{Minimize } f_3 = \frac{1}{x^2 + y^2 + 1} - 1.1\exp(-x^2 - y^2), \\
&\qquad x, y \in [-3, 3].
\end{aligned}$$

This test function has a disconnected Pareto optimal set, and the Pareto front is a curve 'following a convoluted path through objective space'.

5.7 DTLZ1a

We have selected four of the test functions described in [17], and have made alterations to them in order to keep the number of decision variables small, and/or to reduce the ruggedness to 'reasonable' levels. We append an "a" to the name of these functions to show that we have changed them slightly.

The first function from [17], DTLZ1, is used with two objectives and six decision variables here. This gives the following function, DTLZ1a:

$$\begin{aligned}
\text{Minimize } f_1 &= \frac{1}{2}x_1(1+g), \\
\text{Minimize } f_2 &= \frac{1}{2}(1-x_1)(1+g), \\
g &= 100\left[5 + \sum_{i \in 2..6}(x_i - 0.5)^2 - \cos(2\pi(x_i - 0.5))\right], \\
x_i &\in [0,1], i \in 1..n, n = 6.
\end{aligned}$$

The ruggedness of the function is controlled by the g function. In the original DTLZ1, 20π is used in the cosine term, giving a total of $11^k - 1$ local Pareto optimal fronts. This is excessively rugged for our purposes, hence we use 2π instead. The Pareto optimal set consists of all solutions where all but the first decision variables are equal to 0.5, and the first decision variable may take any value in $[0, 1]$.

5.8 DTLZ2a and DTLZ4a

The next two test problems have three objective functions defined on eight decision variables. DTLZ2a and DTLZ4a are given by:

$$\begin{aligned}
&\text{Minimize } f_1 = (1+g)\cos(x_1^\alpha \pi/2)\cos(x_2^\alpha \pi/2),\\
&\text{Minimize } f_2 = (1+g)\cos(x_1^\alpha \pi/2)\sin(x_2^\alpha \pi/2),\\
&\text{Minimize } f_3 = (1+g)\sin(x_1^\alpha \pi/2),\\
&g = \sum_{i \in 3..8} (x_i - 0.5)^2,\\
&x_i \in [0,1], i \in 1..n, n = 8,\\
&\alpha = \begin{cases} 1, & \text{if function is DTLZ2a} \\ 100, & \text{if function is DTLZ4a.} \end{cases}
\end{aligned}$$

The Pareto front is one eighth of a sphere of radius 1, centred on $0, 0, 0$. The Pareto optimal set consists of all solutions where all but the first decision variables are equal to 0.5, and the first decision variable may take any value in $[0, 1]$. The effect of setting $\alpha = 100$ is to severely bias the density distribution of solutions toward the $f_3 - f_1$ and $f_2 - f_1$ planes.

5.9 DTLZ7a

This problem has four disconnected regions in the Pareto front (in objective space):

$$\begin{aligned}
&\text{Minimize } f_1 = x_1,\\
&\text{Minimize } f_2 = x_2,\\
&\text{Minimize } f_3 = (1+g)h,\\
&g = 1 + 9/6 \sum_{i \in 3..8} x_i,\\
&h = 3 - \sum_{i \in \{1,2\}} \left[\frac{f_i}{1+g}(1 + \sin(3\pi f_i)) \right],\\
&x_i \in [0,1], i \in 1..n, n = 8.
\end{aligned}$$

6 Selected performance analysis techniques

The output of a multiobjective optimizer for a single run is an approximation set: the set of all nondominated points found during the run. Assessing the performance of optimizers thus relies upon some means of measuring or comparing the quality of one or more approximation sets. Many such *performance indicators*[6] exist in the literature but it has been shown in some recent works [32, 31, 55] that several of these are not consistent with the partial ordering of approximation sets that follows logically from the basic principles of Pareto optimization. For example, in the worst case, two sets of nondominated points, A and B may be compared, where *every* point in B is dominated by at least one point in A, and yet B yields a better evaluation under the chosen indicator [32]. This is particularly the case for indicators that try to assess one isolated aspect, such as the diversity, extent or proximity to the Pareto front, of a nondominated set. Fortunately, indicators that simultaneously take into account *all* of these aspects of what it means to be a good nondominated set do exist, and moreover they are compatible and/or complete with respect to the true partial ordering of nondominated sets. Two such 'well-behaved indicators', which we choose to use in our experiments, are the hypervolume indicator and the binary epsilon indicator, described below. For a more detailed discussion of assessment methods, the reader is referred to [55], where a full classification of indicators is given.

In addition to choosing appropriate performance indicators, one must also decide how to collate and interpret the results from multiple runs, and how to present results graphically, if at all. Since, in our optimization scenario, it is likely that there would be just one (or, perhaps, two) shots at performing the optimization, the average-case and worst-case performance are paramount. Hence, for each, test function the following information is given:

- the mean/median results of individual runs (as well as standard deviation/interquartile ranges)
- the statistical significance of differences in the distributions
- for the 2-objective functions only, plots of the median, and worst attainment surfaces achieved
- for the 3-objective functions only, plots of the worst attainment surfaces achieved only (since 3d plots showing other surfaces are visually confusing).

[6] We prefer the term 'indicator', following [55], because the terms 'measure' and 'metric' have reserved meanings in mathematics that not all indicators conform to.

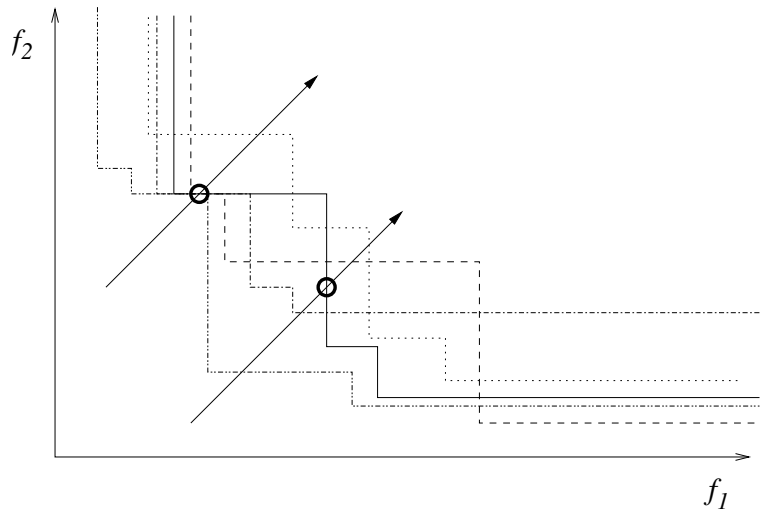

Figure 2: Five attainment surfaces are shown, representing the output of five runs of an optimizer. The two diagonal lines intersect the five surfaces at various points; in both cases, the circle indicates the intersection that weakly dominates at least $5-3+1=3$ surfaces and is also weakly dominated by at least 3 surfaces. Therefore, these two points both lie on the third summary attainment surface

We give all details in the following sections and also make our assessment methods source code available for public use.

6.1 Hypervolume indicator (a.k.a the $\mathcal{S}$ measure) [54]

This indicator assesses the size (hypervolume or Lebesgue integral) of the region dominated by a sample set of points, thus larger values generally indicate better nondominated sets. The region itself 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.

The hypervolume indicator $\mathcal{S}$ is well-behaved in the sense that given two sets, A and B, and using the same bounding point, if $\mathcal{S}(A) > \mathcal{S}(B)$ then it cannot be the case that B is better than A (where better is used in the sense defined in [55]). However, it must be noted that this result does not imply that A is better than B, either. Nonetheless, in the absence of preferences, and with a reasonable choice of bounding point, the $\mathcal{S}$ measure gives results that concur with intuition — proximity to the true PF, extent and evenness of solutions all tend to be rewarded.

In order to choose a bounding point for application of the $\mathcal{S}$ measure, we use the following method. First, the collection of nondominated point sets from all runs of all algorithms are aggre-

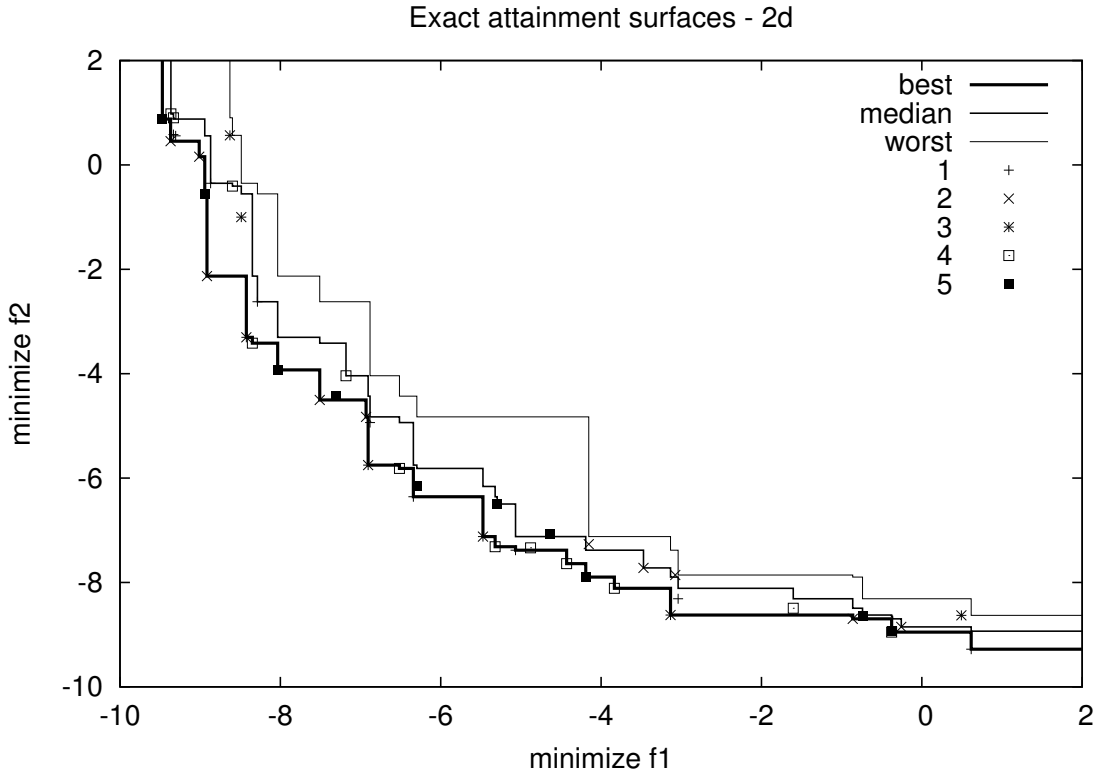

Figure 3: Five sets of nondominated points and the best, median and worst attainment surfaces that they define. The interpretation of the median attainment surface is that, for every point on it, a point (weakly) dominating this was obtained in at least 50% of the nondominated sets. Similarly, the worst 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

gated into a single superset. Then, the ideal and the anti-ideal point of this superset is 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 $\mathcal{S}$ measure of each individual run, and report the mean and the standard deviation of these. Since the distribution of ParEGO and NSGA-II's

results are not necessarily normal, we use the Mann-Whitney rank-sum test to indicate if there is a statistically significant difference in the position of the two distributions.

6.2 Additive binary epsilon indicator

There are two ways in which the hypervolume indicator, used on its own, *can* be misleading. First, there is no way, from looking at $\mathcal{S}$ values in isolation, of inferring whether one set is actually better than another in a strict sense. Secondly, the choice of bounding point is rather arbitrary, and this can affect the ordering of some pairs of sets.

A method that avoids these particular difficulties is the additive binary epsilon indicator [55]. This takes a pair of nondominated sets A and B and returns a pair of numbers, (I_A, I_B) in which I_A (I_B) is the minimum distance that all the points in A (B) must be shifted (positive numbers indicating a shift towards the ideal point) such that A (B) then 'just weakly dominates' every point in B (A). Thus a pair of numbers $(I_A \leq 0, I_B > 0)$ indicates that A is strictly better than B, because the whole of A's attainment surface can be shifted away from the ideal point, and it still weakly dominates all B's points, while B must be shifted *towards* the ideal in order to weakly dominate all of A. A pair of numbers $(I_A > 0, I_B > 0)$ indicates that neither set is strictly better than the other. However, if I_A is less than I_B, then it needs to be shifted less in order to weakly dominate B's points and, in this weaker sense, it is therefore better. In the results we present, unnormalized objective values have been used in the computation of I_A and I_B.

To summarise the results of multiple runs, we compute the binary epsilon indicator for each pair of runs in turn and then report on the median and interquartile range values of I_A and I_B. We prefer the median and IQR to the mean and SD because if the median for either algorithm is less than zero, then this has a precise interpretation — that it was strictly better than the other algorithm on at least 50% of runs. We again use the Mann-Whitney rank sum test to decide if the distribution of I_A values is different from the distribution of I_B values.

6.3 Median and worst attainment surface plots

An *attainment surface* is the surface uniquely determined by a set of nondominated points that divides the objective space into the region dominated by the set and the region that is not dominated by it [20]. Given n runs of an algorithm, it would be nice to summarise the n attainment

Parameter	*setting*
Population size	20
Maximum generations	13 [100]
Crossover probability	0.9
Real-value mutation probability	$1/d$
Real-value SBX parameter	10
Real-value mutation parameter	50

Table 4: NSGA-II parameter settings, where d is the number of decision parameter dimensions

surfaces obtained, using only one or two 'summary surfaces' that everywhere lie on or between the n attainment surfaces. Such summary surfaces can be defined by imagining a diagonal line in the direction of increasing objective values cutting through the n attainment surfaces (see Figure 2). The intersection on this line that weakly dominates at least $n-p+1$ of the surfaces and is weakly dominated by at least p of them, defines one point on the 'pth summary attainment surface'.

Obviously, from the definition of the pth summary attainment surface, it is possible to define a median (summary) surface (for an odd number of approximation sets) and a best and worst (summary) surface. For illustration, we plot five surfaces and their exact sample median, best and worst attainment surfaces in Figure 3. Notice that the interpretation of the median surface is that, for every point on it, a point (weakly) dominating this was obtained in at least 50% of algorithm runs. Similarly, the worst attainment surface has the interpretation that in every algorithm run the entire surface was (weakly) dominated. Thus, plots of these surfaces provide more information than a mere scatter plot of the aggregation of (nondominated) points found from several runs.

For the two-objective problems in this paper, we give the median and worst attainment surface of ParEGO and of NSGA-II on the same plot, with ParEGO's surfaces shown in solid and NSGA-II surfaces shown in dashed lines. For the three-objective problems, we show just the worst attainment surface of the two algorithms, each on separate plots but with the same axes. This gives an indication of the worst-case performance of the two algorithms.

7 Experimental details

To evaluate NSGA-II 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

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

Table 5: ParEGO parameter settings, where d is the number of decision parameter dimensions

evaluations can then be determined and used to estimate performance.

Performance was also compared with a random search run 21 times for up to 10000 function evaluations on all problems.

7.1 NSGA-II parameter settings

The implementation of NSGA-II that we use is the one available for download from Deb's KANGAL webpage. We use default settings (Table 4), as supplied, with one exception. The population size is reduced to 20 in order to maximise the performance over the short run experiments. This size was determined from some runs in which sizes from 10 to 50 were tried. With a population size of 20, 13 generations gives 260 evaluations in all. This is comparable to the 250 evaluations given to ParEGO. Performance of the two algorithms is also compared at 100 evaluations.

7.2 ParEGO parameter settings

The main choices in the parameter settings of ParEGO concern the number of scalarizing vectors to use (specified via the parameter s in Equation 1), and in the parameters of the internal EA. For the former we chose a number which would allow several 'passes' over each scalarizing vector. For the evolutionary algorithm within ParEGO, a small number of preliminary experiments were carried out to determine settings that led to robust optimization of the expected improvement of the DACE model. The full set of parameter settings used in all runs of ParEGO are given in 5.

8 Results

Compared to a random search of the space, NSGA-II does well on most problems, even though only 100 or 260 function evaluations are used. An impression of this can be had from the plots (Figures 4 and 5) showing the median and worst attainment surfaces for NSGA-II, compared with 1000 random solutions, plotted in objective space, on four of the test functions. Plots of the attainment surfaces for the random search algorithm, not shown here, confirm that NSGA-II is better than random search over 260 function evaluations on all the test problems.

Tables 6 and 7 give the results of applying the $\mathcal{S}$ measure to the 21 runs of ParEGO and NSGA-II after, respectively, 100 and 250 (260 for NSGA-II) function evaluations. From these results it is clear that ParEGO consistently dominates a larger region of the objective space (modulo the chosen bound point) at 250 evaluations. Not only is the mean higher, but the standard deviation is consistently lower, sometimes by several orders of magnitude, suggesting a much more reliable, and better worst-case, performance. The Mann-Whitney test shows that the differences between the two populations are significant at the 99% confidence level, on *all* functions.

At 100 function evaluations, the outcome is much the same, except for the functions DTLZ4a and DTLZ7a, where NSGA-II obtains a higher mean value (and is significantly better on DTLZ7a). A possible explanation for the relatively poorer performance of ParEGO on these two functions is that it has actually only 'chosen' 13 of the solutions evaluated — the first 87 solutions being the latin hypercube — since these functions both have 8 decision variables. If one were interested in a strong performance after 100 evaluations on these functions, a sparser latin hypercube may be worth considering.

Tables 8 and 9 give the equivalent results for the additive binary epsilon indicator. The results are largely in agreement with those of the $\mathcal{S}$ measure, despite the fact that the method of measurement is quite different. Once again it is evident that ParEGO has less variation, and that after 250 (260 for NSGA-II) evaluations it is not significantly behind on any problem. Furthermore, on one problem, OKA2, its median indicator score is negative, indicating that its nondominated sets are better in a strict sense than NSGA-II's on more than 50% of runs. On two or three other problems its median indicator value is also very close to zero, and we can confirm that on these problems its nondominated sets are strictly better than NSGA-II's on several of the runs.

Function	ParEGO mean (SD)	NSGA-II mean (SD)	z-value	significance
KNO1	94.5363 (7.28313)	87.7112 (8.28212)	-2.427529	$> 99\%$
OKA1	16.9725 (0.355216)	14.138 (1.25082)	-5.546841	$> 99\%$
OKA2	22.2165 (0.887676)	15.5118 (1.42961)	-5.546841	$> 99\%$
VLMOP2	0.320038 (0.00563326)	0.263637 (0.0387093)	-5.446218	$> 99\%$
VLMOP3	46.3881 (0.157458)	34.8668 (12.5845)	-5.320439	$> 99\%$
DTLZ1a	189262 (209.001)	186855 (1943.03)	-5.270128	$> 99\%$
DTLZ2a	4.40784 (0.0906379)	4.06872 (0.172994)	-4.943103	$> 99\%$
DTLZ4a	0.673329 (0.263526)	0.824647 (0.408542)	-0.943340	– no winner
DTLZ7a	12.9895 (0.655795)	16.8967 (1.00974)	-5.546841	$> 99\%$ NSGA-II wins

Table 6: Values of the $\mathcal{S}$ measure after 100/100 evaluations of ParEGO/NSGA-II. Larger values indicate better performance. The distributions of the $\mathcal{S}$ values are tested using the Mann-Whitney rank sum test. The z values and significance level are indicated. ParEGO is significantly better than NSGA-II unless stated

Function	ParEGO mean (SD)	NSGA-II mean (SD)	z-value	significance
KNO1	108.478 (5.83982)	103.2 (7.92597)	-2.276594	$> 99\%$
OKA1	18.0834 (0.372983)	15.982 (0.681422)	-5.546841	$> 99\%$
OKA2	24.0278 (0.467227)	19.1279 (1.44751)	-5.546841	$> 99\%$
VLMOP2	0.312768 (0.00398254)	0.30501 (0.0119822)	-2.528152	$> 99\%$
VLMOP3	89.9313 (0.113778)	83.0951 (18.8421)	-2.578463	$> 99\%$
DTLZ1a	64869.6 (6.98978)	64682.7 (154.84)	-3.710472	$> 99\%$
DTLZ2a	4.46196 (0.0276682)	4.07554 (0.145971)	-5.546841	$> 99\%$
DTLZ4a	2.14206 (0.478063)	1.42626 (0.809588)	-3.106734	$> 99\%$
DTLZ7a	15.8421 (0.431151)	13.653 (0.930073)	-5.395906	$> 99\%$

Table 7: Values of the $\mathcal{S}$ measure after 250/260 evaluations of ParEGO/NSGA-II. Larger values indicate better performance. The distributions of the $\mathcal{S}$ values are tested using the Mann-Whitney rank sum test. The z values and significance level are indicated. ParEGO is better in all cases

The plots of the median and worst attainment surfaces (Figures 4–9) confirm the outcome of the measures, above. In particular, the worst attainment surface for NSGA-II is much worse than for ParEGO on several of the problems.

Finally, Figure 10 shows the solutions, and corresponding points in objective space, visited by NSGA-II and ParEGO on the first run on function OKA1. This plot indicates the very different ways that ParEGO and NSGA-II sample the space. NSGA-II is much more aggressive at honing in on good points, exploiting them much more than ParEGO. However, despite ParEGO's greater exploration, it still manages to exploit the good solutions it has found very effectively, and so returns both very good and 'very average' points in the objective space.

Function	median values $(I_{ParEGO}, I_{NSGA-II})$	IQR values $(I_{ParEGO}, I_{NSGA-II})$	z-value	significance
KNO1	(1.5261, 2.2245)	(1.2633, 1.3428)	-2.1508	$> 98\%$
OKA1	(0.3581, 0.9120)	(0.2395, 0.3714)	-5.4965	$> 99\%$
OKA2	(-0.0123, 1.3692)	(0.0540, 0.4530)	-5.5468	$> 99\%$
VLMOP2	(0.0313, 0.1029)	(0.0283, 0.2180)	-5.0186	$> 99\%$
VLMOP3	(0.0781, 0.6956)	(0.2167, 1.0073)	-4.3645	$> 99\%$
DTLZ1a	(0.3168, 17.7269)	(0.8050, 17.2216)	-5.5846	$> 99\%$
DTLZ2a	(0.00780, 0.0613)	(0.0151, 0.0969)	-3.458914	$> 99\%$
DTLZ4a	(0.2276, 0.4153)	(0.4253,0.7326)	-1.3207	$> 90\%$
DTLZ7a	(0.7813, 0.0963)	(0.6198, 0.9580)	-3.0564	$> 99\%$ NSGA-II wins

Table 8: Median and interquartile range values of the binary epsilon indicator after 100/100 evaluations of ParEGO/NSGA-II. Lower values indicate better performance. The distributions of the I values are tested using the Mann-Whitney rank sum test. The z values and significance level are indicated. ParEGO is significantly better than NSGA-II unless stated

Function	median values $(I_{ParEGO}, I_{NSGA-II})$	IQR values $(I_{ParEGO}, I_{NSGA-II})$	z-value	significance
KNO1	(1.2906, 1.7204)	(0.8487, 1.4558)	-2.3520	$> 99\%$
OKA1	(0.2674, 0.7440)	(0.1268, 0.1605)	-5.4211	$> 99\%$
OKA2	(-0.0001, 1.3058)	(0.0373, 0.3484)	-5.5468	$> 99\%$
VLMOP2	(0.0467, 0.0491)	(0.0205, 0.0784)	-1.043963	– no winner
VLMOP3	(0.1801, 0.0866)	(0.1256, 0.2509)	-0.6666	– no winner
DTLZ1a	(0.0817, 4.7435)	(0.1699 ,8.5125)	-5.5468	$> 99\%$
DTLZ2a	(0.2425, 0.2141)	(0.1161, 0.3432)	-1.2452	– no winner
DTLZ4a	(0.2674, 0.8682)	(0.3878, 0.2488)	-4.4400	$> 99\%$
DTLZ7a	(0.0152, 3.025)	(0.0516, 2.2221)	-5.5468	$> 99\%$

Table 9: Median and interquartile range values of the binary epsilon indicator after 250/260 evaluations of ParEGO/NSGA-II. Lower values indicate better performance. The distributions of the I values are tested using the Mann-Whitney rank sum test. The z values and significance level are indicated. ParEGO is significantly better than NSGA-II unless stated

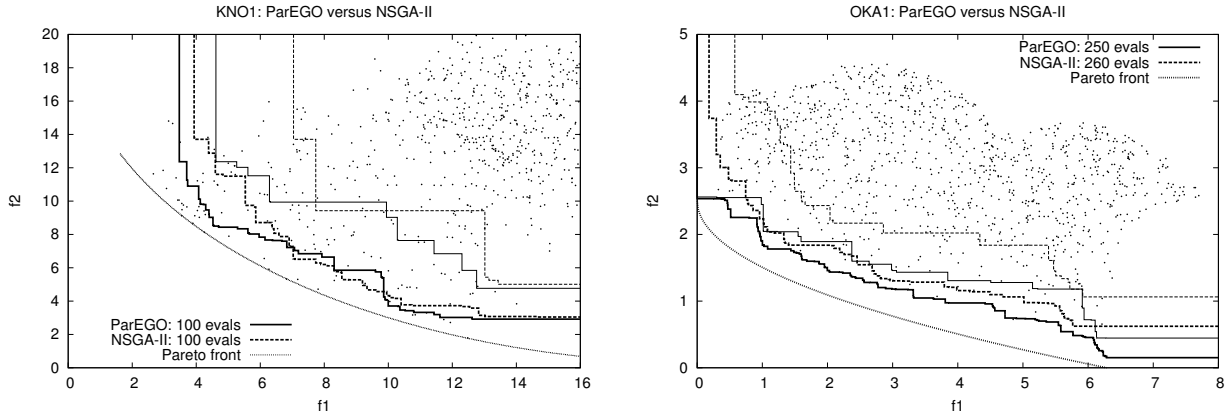

Figure 4: Attainment surface plot with 1000 random search points also shown, on the KNO1 function after 100 function evaluations (left), and on the OKA1 function after 250/260 function evaluations (right)

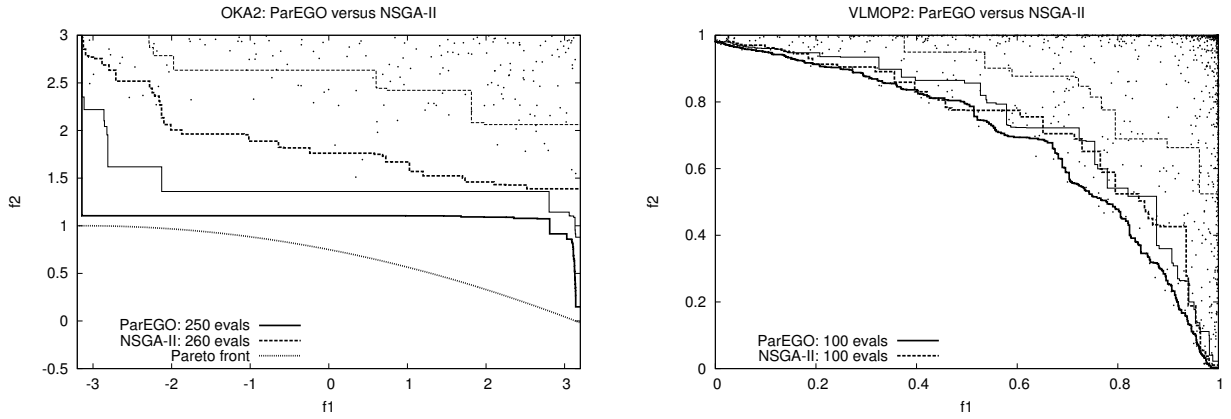

Figure 5: Attainment surface plot with 1000 random points also shown, on OKA2 after 250/260 function evaluations (left), and on VLMOP2 after 100 function evaluations (right)

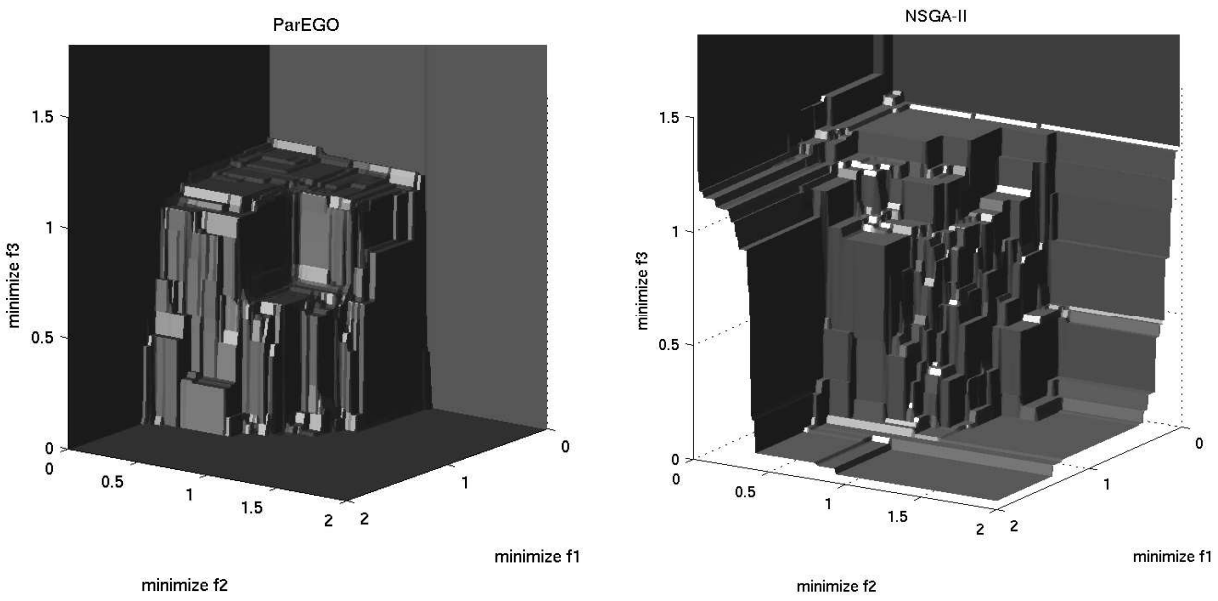

Figure 6: Worst attainment surfaces after 250/260 function evaluations for ParEGO (left) and NSGA-II (right) on the DTLZ2 function

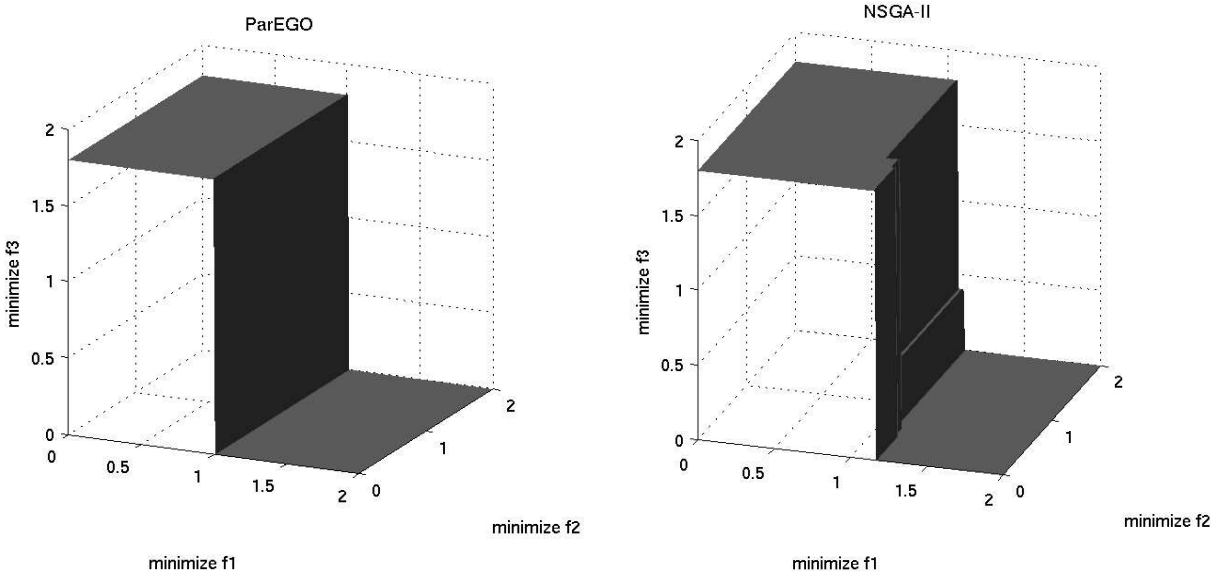

Figure 7: Worst attainment surfaces after 250/260 function evaluations for ParEGO (left) and NSGA-II (right) on the DTLZ4 function

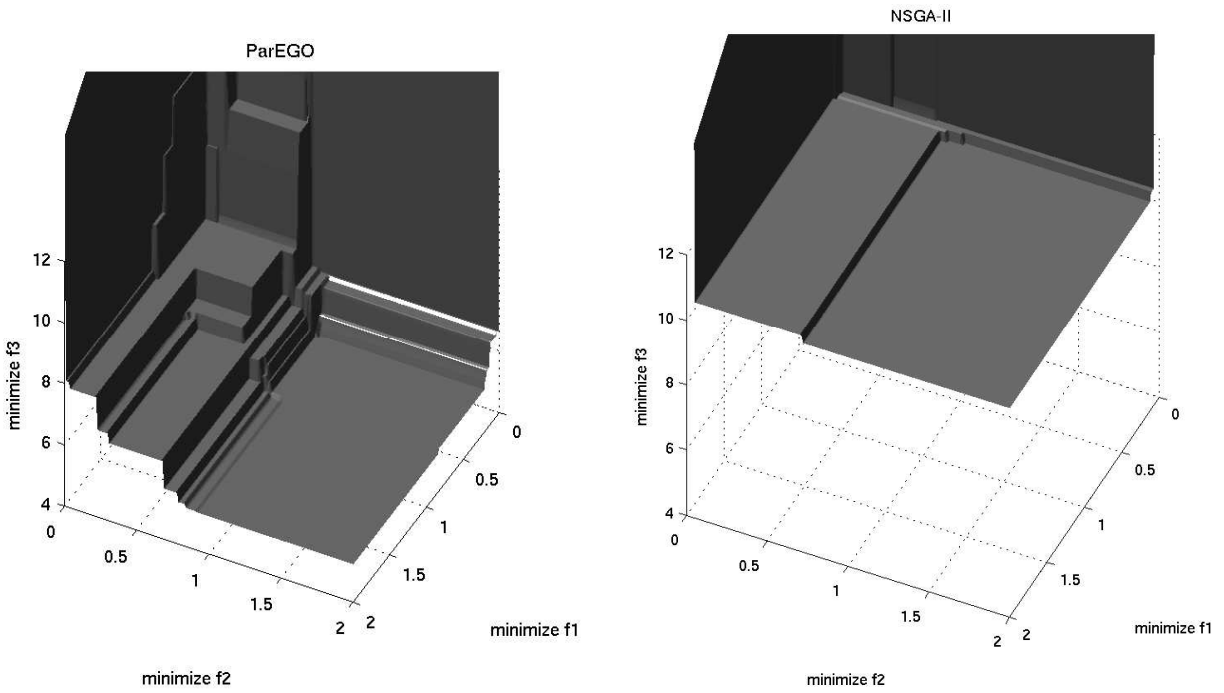

Figure 8: Worst attainment surfaces after 250/260 function evaluations for ParEGO (left) and NSGA-II (right) on the DTLZ7 function

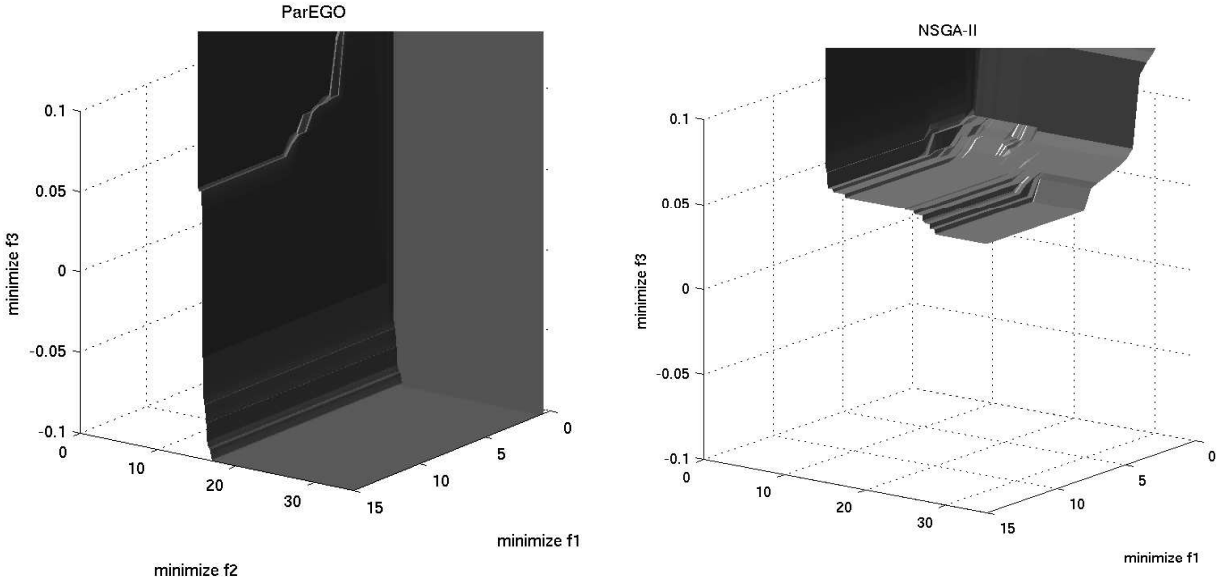

Figure 9: Worst attainment surfaces after 250/260 function evaluations for ParEGO (left) and NSGA-II (right) on the VLMOP3 function

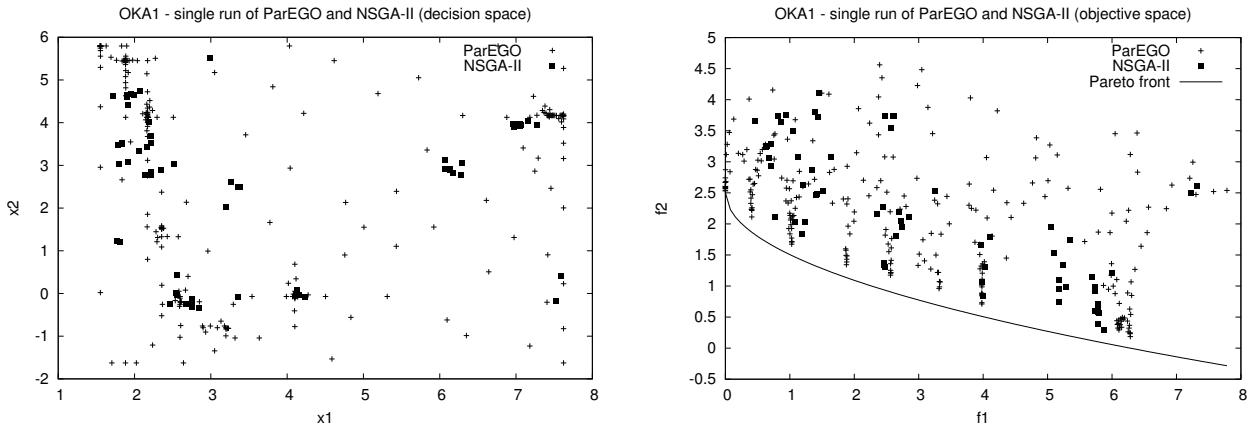

Figure 10: Solutions (left) and their corresponding points in objective space (right) for the first run of ParEGO and of NSGA-II after 250/260 function evaluations on the OKA1 test function. Notice how ParEGO explores the parameter space much more evenly and yet still finds more good points

9 Summary and conclusion

In many optimization scenarios, the number of fitness evaluations that can be performed is severely limited by cost constraints. In this study, the performance of two multiobjective optimization algorithms, ParEGO and NSGA-II, 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 the algorithms, and on these, it was confirmed that both ParEGO and NSGA-II outperformed a random search, even over a very small number of function evaluations.

ParEGO generally outperformed NSGA-II on the tested functions, at both 100 and 250 function evaluations, especially when the worst-case performance was measured. This suggests that ParEGO may offer a more effective search on problems like the instrument set-up optimization problem, described herein, where only one function evaluation can be performed at a time. We are currently evaluating it on this problem as well as some other closely-related 'wet experiment' problems.

Although ParEGO's performance was good on the tested functions, there are at least two potential drawbacks of the current version:

1. Normalization of the cost space relies on knowledge of the cost limits,

2. The use of uniformly random scalarizing vectors does not necessarily result in the best dis-

tribution of nondominated points. Some parts of the Pareto front may be far easier to find than others, so that this may lead to a poor approximation.

In future work, it is our intention to investigate a method based on adapting the scalarizing vectors, as a function of the points visited in the objective space, in order to obtain a better improvement to the current nondominated front at each iteration.

The addition of mechanisms to deal with integer and mixed-integer problems, as well as constraints is also under way.

Acknowledgments

Thanks to Douglas Kell for leading the collaborative work on the instrument set-up optimization problem that led to this paper. JK is supported by a David Phillips fellowship from the BBSRC.

References

[1] D. Büche, G. Guidati, P. Stoll, and P. Kourmoursakos. Self-Organizing Maps for Pareto Optimization of Airfoils. In J. J. Merelo Guervós, P. Adamidis, H.-G. Beyer, J.-L. F.-V. nas, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature—PPSN VII*, pages 122–131, Granada, Spain, September 2002. Springer-Verlag. Lecture Notes in Computer Science No. 2439.

[2] D. Buche, N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with gaussian process fitness function models. *IEEE Transactions on Systems, Man, and Cybernetics C*, 34, 2004. To appear.

[3] J.-J. Chen, D. E. Goldberg, S.-Y. Ho, and K. Sastry. Fitness Inheritance in Multi-Objective Optimization. In W. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. Schultz, J. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 319–326, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[4] P. B. Cheung, L. F. Reis, K. T. Formiga, F. H. Chaudhry, and W. G. Ticona. Multiobjective Evolutionary Algorithms Applied to the Rehabilitation of a Water Distribution System: A Comparative Study. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 662–676, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[5] C. A. Coello Coello. An Updated Survey of GA-Based Multiobjective Optimization Techniques. *ACM Computing Surveys*, 32(2):109–143, June 2000.

[6] C. A. Coello Coello and G. Toscano Pulido. A Micro-Genetic Algorithm for Multiobjective Optimization. In E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 126–140. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[7] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.

[8] D. A. Cohn, L. Atlas, and R. E. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[9] D. Corne, M. Oates, and D. Kell. Landscape state machines: tools for evolutionary algorithm performance analyses and landscape/algorithm mapping. In *Applications of Evolutionary Computing*, volume 2611 of *LNCS*, pages 187–198. Springer, 2003.

[10] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates. PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. In L. Spector, E. D. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 283–290, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[11] V. Coverstone-Caroll, J. Hartmann, and W. Mason. Optimal Multi-Objective Low-Thrust Spacecraft Trajectories. *Computer Methods in Applied Mechanics and Engineering*, 186:387–402, 2000.

[12] Z. S. Davies, R. J. Gilbert, R. J. Merry, D. B. Kell, M. K. Theodorou, and G. W. Griffith. Efficient improvement of silage additives by using genetic algorithms. *Applied and Environmental Microbiology*, pages 1435–1443, April 2000.

[13] K. Deb. Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design. In K. Miettinen, M. M. Mäkelä, P. Neittaanmäki, and J. Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, chapter 8, pages 135–161. John Wiley & Sons, Ltd, Chichester, UK, 1999.

[14] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.

[15] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.

[16] K. Deb and H. Beyer. Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation*, 9(2):197–221, 2001.

[17] 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.

[18] E. I. Ducheyne, B. De Baets, and R. De Wulf. Is Fitness Inheritance Useful for Real-World Applications? In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 31–42, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[19] 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.

[20] C. M. Fonseca and P. J. Fleming. On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature—PPSN IV*, Lecture Notes in Computer Science, pages 584–593, Berlin, Germany, September 1996. Springer-Verlag.

[21] A. Gaspar-Cunha and A. Vieira. A multi-objective evolutionary algorithm using approximate fitness evaluation. In G. Bugeda, J. A-Dsidri, J. Periaux, M. Schoenauer, and G. Winter, editors, *International Congress on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems EUROGEN*, 2003.

[22] 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, Valencia, Spain, 2004.

[23] J. Horn and N. Nafpliotis. Multiobjective Optimization using the Niched Pareto Genetic Algorithm. Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[24] E. J. Hughes. Multi-objective Binary Search Optimisation. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 102–117, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[25] M. T. Jensen. Reducing the run-time complexity of multi-objective eas: The nsga-ii and other algorithms. *IEEE Transactions on Evolutionary Computation*, 7(5):502–515, 2003.

[26] 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.

[27] D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[28] N. Khan, D. E. Goldberg, and M. Pelikan. Multi-Objective Bayesian Optimization Algorithm. In W. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan,

V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. Schultz, J. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, page 684, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[29] Knowles' webpage. http://dbk.ch.umist.ac.uk/knowles/.

[30] J. Knowles. Parameter setting for PESA-II on the closed-loop spectrometer optimization problem: a simulation study using short run experiments. Internal report available on request from j.knowlesumist.ac.uk, April 2004.

[31] J. Knowles and D. Corne. On Metrics for Comparing Nondominated Sets. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 711–716, Piscataway, New Jersey, May 2002. IEEE Service Center.

[32] J. D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization.* PhD thesis, University of Reading, UK, 2002.

[33] J. D. Knowles and D. W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[34] M. Laumanns and J. Ocenasek. Bayesian Optimization Algorithms for Multi-objective Optimization. In *Parallel Problem Solving from Nature—PPSN VII*, pages 298–307, Granada, Spain, September 2002. Springer-Verlag. Lecture Notes in Computer Science No. 2439.

[35] K.-H. Liang, X. Yao, and C. Newton. Evolutionary search of approximated n-dimensional landscapes. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 4(3):172–183, 2000.

[36] R. Myers and D. Montgomery. *Response Surface Methodology.* Wiley, New York, 1995.

[37] 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.

[38] B. Naujoks, L. Willmes, T. Bäck, and W. Haase. Evaluating Multi-criteria Evolutionary Algorithms for Airfoil Optimization. In J. J. Merelo Guervós, P. Adamidis, H.-G. Beyer, J.-

L. F.-V. nas, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature—PPSN VII*, pages 841–850, Granada, Spain, September 2002. Springer-Verlag. Lecture Notes in Computer Science No. 2439.

[39] S. O'Hagan, W. B. Dunn, M. Brown, J. D. Knowles, and D. B. 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) available from http://pubs.acs.org/cgi-bin/asap.cgi/ancham/asap/html/ac049146x.html.

[40] T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff. On test functions for evolutionary multi-objective optimization. In *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature*, LNCS. Springer, 2004. (To appear).

[41] Y. S. Ong, P. B. Nair, and A. J. Kean. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 41(4):687–696, 2003.

[42] Y. S. Ong, P. B. Nair, A. J. Keane, and Z. Z. Zhou. Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In Y. Jin, editor, *Knowledge Incorporation in Evolutionary Computation*, Studies in Fuzziness and Soft Computing Series. Springer Verlag, 2004.

[43] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[44] A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In *Parallel Problem Solving from Nature - PPSN V*, pages 87–96, 1998.

[45] J. Sacks, W. Welch, T. Mitchell, and H. Wynn. Design and analysis of computer experiments (with discussion). *Statistical Science*, 4(409–435), 1989.

[46] K. Sastry, D. Goldberg, and M. Pelikan. Don't evaluate, inherit. In *Proceedings of genetic and Evolutionary Computation Conference*, pages 551–558. Morgan Kaufmann, 2001.

[47] J. Schwarz and J. Ocenasek. Evolutionary Multiobjective Bayesian Optimization Algorithm: Experimental Study. In *Proceedings of the 35th Spring International Conference: Modelling*

and Simulation of Systems (MOSIS'01), pages 101–108, Czech Republic, 2001. MARQ, Hradec and Moravici.

[48] 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.

[49] S. Vaidyanathan, D. Kell, and R. Goodacre. Selective detection of proteins in mixtures using electrospray ionization mass spectrometry: influence of instrumental settings and implications for proteomics. *Analytical Chemistry*, 76:5024–5032, 2004.

[50] D. A. V. Veldhuizen and G. B. Lamont. Multiobjective Evolutionary Algorithm Test Suites. In J. Carroll, H. Haddad, D. Oppenheim, B. Bryant, and G. B. Lamont, editors, *Proceedings of the 1999 ACM Symposium on Applied Computing*, pages 351–357, San Antonio, Texas, 1999. ACM.

[51] A. Vicini and D. Quagliarella. Airfoil and wing design using hybrid optimization strategies. *AIAA Journal*, 37(5), 1999.

[52] D. Weuster-Botz and C. Wandrey. Medium optimization by genetic algorithm for continuous production of formate dehydrogenase. *Process Biochemistry*, 30:563–571, 1995.

[53] D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8:1341–1390, 1996.

[54] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications.* PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[55] 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.