Noisy Multiobjective Optimization on a Budget of 250 Evaluations

Joshua Knowles[1], David Corne[2], and Alan Reynolds[2]

[1] School of Computer Science, University of Manchester, UK
j.knowles@manchester.ac.uk
http://dbkgroup.org/knowles/TOMO/
[2] School of Mathematics and Computer Science, Heriot-Watt University, UK

Abstract. We consider methods for noisy multiobjective optimization, specifically methods for approximating a true underlying Pareto front when function evaluations are corrupted by Gaussian measurement noise on the objective function values. We focus on the scenario of a limited budget of function evaluations (100 and 250), where previously it was found that an iterative optimization method — ParEGO — based on surrogate modeling of the multiobjective fitness landscape was very effective in the non-noisy case. Our investigation here measures how ParEGO degrades with increasing noise levels. Meanwhile we introduce a new method that we propose for limited-budget and noisy scenarios: TOMO, deriving from the single-objective PB1 algorithm, which iteratively seeks the basins of optima using nonparametric statistical testing over previously visited points. We find ParEGO tends to outperform TOMO, and both (but especially ParEGO), are quite robust to noise. TOMO is comparable and perhaps edges ParEGO in the case of budgets of 100 evaluations with low noise. Both usually beat our suite of five baseline comparisons.

1 Introduction

Real-world optimization problems often involve solutions that are expensive to evaluate, either financially or in time, thus imposing a budget on the number of evaluations that can be done during an optimization procedure. Sometimes, the expense is so acute that only a 'handful' of evaluations is feasible, so that using a latin hypercube design, other experimental designs (DoE approaches), or even a random search may yield better results than iterative approaches, particularly on multiobjective problems. When thousands or more evaluations may be done, however, we would expect iterative sampling methods such as evolutionary algorithms (EAs) to generally outperform random search and/or DoE. But between these two regimes there may lie a third where EAs are ineffective, yet there is the potential to outperform static 'designs' or random search.

There is evidence of this regime in the work done on EAs combined with surrogate modeling, and in the statistics/DoE, direct search and machine learning (ML) literature. Although DoE traditionally concerns itself with static designs, modern techniques also include iterative approaches that augment an initial design, based on the values observed. The EGO method [19] is such a technique: starting from an initial latin hypercube design, it proceeds point by point, always using all previous points to (fit a model and) estimate the point of maximum expected improvement. A similar iterative method

M. Ehrgott et al. (Eds.): EMO 2009, LNCS 5467, pp. 36–50, 2009.

(PB1 [1]) was also proposed for optimization on a very limited evaluation budget, and seems fairly robust to noise — a very common phenomenon in real-world applications.

Here we investigate optimization given the combined difficulties of a very limited evaluations budget and the existence of noise. We aim towards an understanding where MOEAs, random search, DoE and advanced iterative approaches might stand relative to one another, under these conditions. We continue in Section 2 with an account of prior and related research. Then in Section 3 we introduce TOMO in some detail, and briefly outline our set of comparison and baseline methods: ParEGO, DoE, random search, a simple multiple trajectory hillclimber, what we call a 'simple Gaussian model learner', and PESA-II. Section 4 describes our experimental setup, results are presented in Section 5, and Section 6 discusses the results and concludes.

2 Background

Expensive optimization problems are now rather common. Optimizing structural form, guided by the use of accurate simulators, is perhaps the most familiar domain in which such problems arise (e.g. [16,17]), while they also occur in biochemistry and materials science [6,12], robotics (e.g. [14]), and instrument configuration (e.g. [10]).

Typically, expensive fitness functions involve computational fluid dynamics (CFD) or similar simulations or finite element grids. E.g. [16] uses CFD in evaluating candidate shapes for the combustion chamber of a diesel engine, aiming to minimize Nitrous Oxide emissions. Often, real-world testing rather than simulation is involved. E.g. an instrumentation setup problem ([10]) which formed the motivation for ParEGO [22], concerns the efficiency of instruments used to test and monitor biological samples; [10] reports that it took several days to perform 180 evaluations, each of which required manual configuration and testing of an instrument. Similar time may be needed for evolving locomotion controllers for robot gaits, in which physical setup and testing of a configuration is preferable to simulation [14]. In [14], they observed the (very common) complication of noise: some configurations may score well, but be non-robust to slight changes in the evaluation regime.

The effects of noise and ways to deal with it in evolutionary computation have been analysed much (e.g. [2,3]). Unfortunately no comfort is yet to be found in this for those with a very limited evaluations budget. It turns out that effects of noise are highly problem dependent, while the key questions relate to whether increasing exploration (e.g. larger population sizes) or increased resampling (multiple evaluations to better characterize individual solutions) are best. Either way, there is little help here for practitioners on a very limited evaluation budget.

Even ignoring noise, it seems that, particularly in the context of multiobjective optimization, the published work relevant to limited evaluation budgets is sparse. One approach is to attempt to learn a model of the search landscape with neural networks [26,13], enabling predicted fitnesses to replace the need for evaluation in specific phases of an overall control algorithm. The simplest approach to 'guessing' fitness is actually fitness inheritance, first proposed for multiobjective optimization in [4] and later tested by [9]. Meanwhile, the research literature in general is becoming richer in suggestions of modelling techniques to underpin these guesses and consequently provide guidance towards the most promising next point to evaluate, and in multiobjective optimization

specifically, this work (some of which is reviewed in [23]) has included the use of Bayesian model-building (e.g. [24,33]), support vector regression [27]) and the use of and the adoption of a Gaussian process model in ParEGO [22], by Emmerich and co-authors [11], and also in EGOMOP [16].

Such models, which attempt to learn to predict fitness via neural networks, Gaussian processes, or otherwise, are generally known as 'meta-models'. Detailed reviews of meta-model based evolutionary algorithms include [30,18]. It seems intuitively right that, when the evaluation budget is limited, algorithms that incorporate sophisticated meta-models should be promising. This is borne out in published work so far. It seems clear that ParEGO's employment of a metamodel leads to significantly better performance in this context than traditional MOEAs, random search, and a few additional alternatives (using less sophisticated models) against which it has so far been tested [21,22].

3 Optimization Methods

3.1 The Tau-Oriented Multiobjective Optimizer (TOMO)

TOMO is based closely on the principles and some of the procedures used in the single-objective optimizer, PB1.

Principles of PB1. This method [1] is based on the assumption that near an optimum of a function there should be a negative correlation between the (single-objective) fitness of points and their distance from the optimum. This leads to a basic procedure for estimating the location of an optimum: given a cloud of previously evaluated points, search for a 'test point' where the correlation reaches its maximum absolute value. PB1 iterates this, as illustrated in Fig. 1a. NB: throughout the paper fitnesses are actually costs, i.e. we assume minimization.

On non-convex problems as well as functions with plateaus or ridges, the correlation between fitness and distance from the optimum may only hold locally. To account for this, PB1 attempts to identify a subset of the points previously evaluated, forming a convex region in decision space, for which the correlation is observed to occur (see below for details). It is this subset of points which is subsequently used to estimate a new test point. Correlations in PB1 are measured using Kendall's Tau (τ), a nonparametric method based on the ranks of the distances and fitnesses.

Empirical tests of PB1 [1] indicate that it can successfully optimize low-dimensional functions in a small number of steps, including multimodal, ridge and plateau functions, and it is relatively robust to noise. On the one hand, PB1's ability to aggressively search a space can be attributed to the fact that it exploits information on the topology of the search landscape gleaned from all previous points. In this regard, PB1 works similarly to EGO (see ParEGO, below). On the other hand, its robustness to noise can be attributed to the fact that the test it uses to 'reason' about the topology is quite weak: a nonparametric correlation value is not disturbed much when noise is added to the points. See Fig. 1 to see how this contrasts with EGO.

Adapting PB1 to the Multiobjective Case. There are two main hurdles to making an effective multiobjective algorithm based on PB1 and its use of Kendall's correlation

Algorithm 1. High-level Algorithm Pseudocode for TOMO and ParEGO

input: a multiobjective optimization problem with k objectives
require: a sequence of scalarizing weight vectors $\langle \boldsymbol{\lambda} \rangle$
distribute initial E points in a latin hypercube design; evaluate each one
while evaluation limit not reached **do**
    draw the next scalarizing weight vector and use it to scalarize all previously evaluated points
    construct a model of the scalarized search landscape based on a subset of (or all) previous points
    search the model iteratively to find a single new candidate point; evaluate this point on the real multiobjective function
end while
output: all visited solutions

measure to orient search. The first concerns how to convert the basic principle of using fitness-distance correlations to work in the multiobjective case. This, we achieve by taking a simple scalarizing approach, very similar to that used in ParEGO, where at each step of the algorithm the next weight vector from a sequence is used to scalarize the fitnesses of all points for that step (see Algorithm 1). The second hurdle derives from the fact that although PB1 can cope with some multimodality it is not designed to find multiple optima. Our initial testing of PB1 showed that it strongly favours one optimum in a multimodal function, and has limited ability to escape local optima. To overcome this, TOMO was equipped with a parameter-space niching method and intermittent generation of explorative (random) search points.

Latin hypercube initialization. The initial solutions are generated in a space-filling design using a latin hypercube routine following a description in [31]. The number of initial solutions is set to $E = 11d - 1$, where d is the parameter space dimension of the function to be optimized, as suggested in [19]. This procedure in TOMO is adopted directly from ParEGO.

The scalarizing weight vectors. TOMO 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 from the set of evenly distributed vectors defined by:

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

with $|\Lambda| = \binom{s+k-1}{k-1}$ (so that the choice of the parameter s determines how many vectors there are in total). The scalar cost of a solution is then computed using the augmented Tchebycheff function (see [25] pp. 100–102) :

$$f_{\boldsymbol{\lambda}}(x) = \max_{j=1}^{k} (\lambda_j . f_j(x)) + \rho \sum_{j=1}^{k} \lambda_j . f_j(x), \quad (2)$$

where ρ is a small positive value which we set to 0.05. The weight vectors are arranged in a sequence using a Gray coding. To select the 'next' vector, an index into this sequence is incremented mod $|\Lambda|$ so that the sequence wraps around.

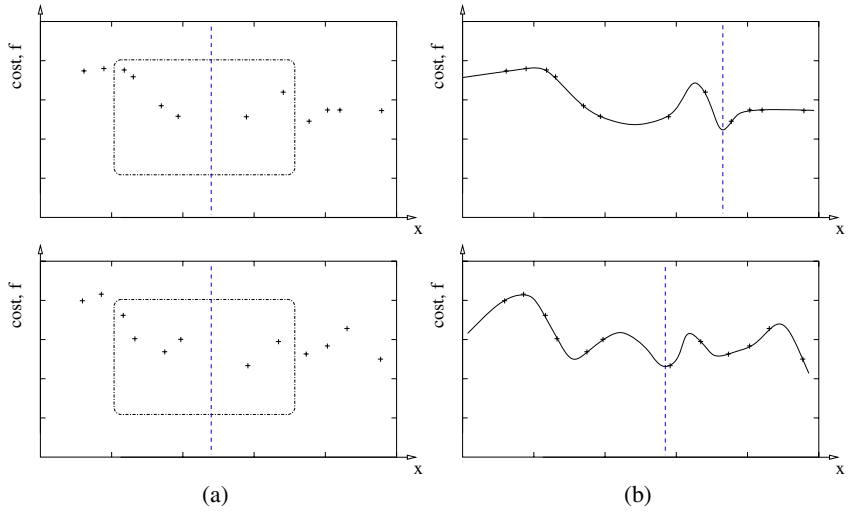

Fig. 1. Schematic illustrating the principles of (a) TOMO and (b) ParEGO on a 1-parameter 1-objective cost function. The upper and lower plots indicate different noisy measurements of the same points. (a) TOMO identifies a region (dashed rectangle) where a statistically significant negative correlation seems to occur between fitness difference and parameter-space distance from a 'test' point (the vertical line). This region and the test point are little changed as a result of noise (compare top and bottom). (b) ParEGO fits a model which interpolates the set of previous points. The model may move significantly under different noisy instances of the same set of measurements (compare top and bottom). The minimum of the (mean) model is shown by the vertical line, however ParEGO does not necessarily move to this minimum, but rather to a point of maximum *expected improvement*, which accounts also for the variance in the model (not shown).

Using niching and random explorative moves. In the original PB1, each main iteration begins by finding a subset of the previously evaluated points possessing a high τ value. This is done by starting with the set of all the points and iteratively removing points on the exterior of the convex hull in decision space, one by one. More precisely, the exterior point furthest from the centroid of the current point cloud is removed in each of these mini-steps. For each subset visited along the way, τ is calculated, using the current centroid as the point to compute τ from, and these values are stored. The subset with the smallest number of points that has a statistically significant τ is then selected for subsequent use. This ensures that τ is calculated over a region from within which no points have been removed, and over which the required fitness-distance correlation holds. To generate a new point for real evaluation, PB1 then searches over a region defined by this subset (specifically, the union of Voronoi regions pertaining to these points) for a point yielding a strong fitness-distance correlation.

We replace PB1's method of obtaining a subset by an approach less likely to converge on a local optimum. Instead of using the centroid of the point cloud during this whittling down process, we use either (i) tournament selection, based on parameter-space niched fitness [7]; or (ii) a randomly generated point. An exploitation (i) step is used with probability ν and a random explore (ii) step with probability $1 - \nu$. We tuned this on a single-objective test function with two local optima (see Fig. 2), which easily

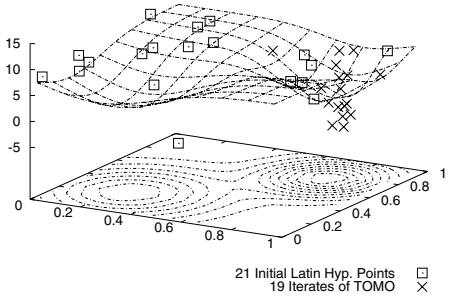

Fig. 2. A single objective two-peak noisy function used to test and tune TOMO. Twenty-one initial points are shown, followed by the nineteen next points TOMO visits. The true objective value of points are indicated by the surface.

misleads the original PB1. The parameters we found to give both rapid convergence and the ability to escape the local optimum were: tournament size 10, Niche radius set according to [7] assuming 10 peaks, and rate of exploitation $\nu = 0.8$. The remaining procedures and parameters of TOMO follow the detailed specification of PB1 in [1].

3.2 ParEGO

The ParEGO (Pareto EGO) algorithm used here is identical to that described in [22], which is essentially a multiobjective translation of EGO [19], making use of scalarizing weight vectors at each step. The high-level pseudocode is given in Algorithm 1. ParEGO works by fitting a Gaussian process stochastic model called DACE [32], to the previously evaluated points, and using this to estimate interesting new points to visit subsequently. The mean of the DACE model interpolates the points, which might make it sensitive to noise, as shown in Fig. 1b. However, because ParEGO uses expected improvement, defined as

$$E[I(\boldsymbol{x})] \equiv E\{\max(y^* - Y_p(\boldsymbol{x}), 0)\},$$

where y^* is the best cost sampled so far, and $Y_p(\boldsymbol{x})$ is a Gaussian distributed random variable representing the model through the point $\boldsymbol{x}$, — a calculation that is based on the *variance* of the model as well as its expected value, this may counteract the problem of interpolating evaluated points to some degree.

3.3 Latin Hypercube and Random Search

As our first two baseline algorithms for comparison, we consider (i) random search (RS) and (ii) the latin hypercube [31] (LH) method used, for initialization, in both TOMO and ParEGO. In the case of random search, a single 'run' corresponds to generating and evaluating *numevals* random points in the parameter space, where *numevals* is the maximum number of fitness evaluations allowed in the experiment (i.e. either 100 or 250). Each solution is independently generated in the standard fashion, by choosing a uniform random value from the range of each parameter. In the case of LH, a single run simply applies one iteration of the method of [31] to generate either 100 or 250 solutions.

3.4 Simple Multiple Trajectory Hillclimber

We also considered baselines that allow us to test alternative yet simple strategies. The first of these attempts a multiobjective search by spreading the evaluation budget (almost) equally among $k + 1$ hillclimbers, where k is the number of objectives. Each objective is assigned to one hillclimber, whose fitness is that objective alone; the $k + 1$th hillclimber uses an equally weighted sum of objectives as its fitness function. A hillclimber maintains a 'current' solution c initialized uniformly at random, and then iterates the following: generate mutant m by copying c and then choosing a random parameter and applying a Gaussian perturbation to it with standard deviation s. If m is no worse than c, then c becomes m.

When the overall evaluations budget is n, the k hillclimbers each have a budget of $n/(k + 1)$, maintaining whole numbers by allowing the sum-of-objectives hillclimber to use the surfeit after division by $k + 1$. The result of SMH is then taken to be the nondominated set of all solutions visited. We test three parameterizations of SMH, namely with $s = 0.1, 0.3, 1.0$.

3.5 Simple Gaussian Model Learner

Our second alternative yet simple baseline is best described as a type of estimation of distribution algorithm for real-valued parameters, although it can equally well be described as a standard type of evolution strategy with multi-parent uniform crossover and a Pareto-oriented truncation selection scheme. Its inspiration comes originally from the considerably growing body of work that finds combinations of learning and exploration to be highly valuable in accelerating progress per evaluation, even when the learning mechanism or model is very simple.

The simple Gaussian model learner (SGM) operates as follows, with a population size P and a standard deviation s. The key part of the algorithm is the way a new point is generated from the current nondominated set of points visited, S. Given S, we generate a new point by doing the following for each parameter j of the new point: choose a member c of S uniformly at random; let parameter j of c be $c[j]$; let parameter j of the new point be drawn from a Gaussian distribution with mean $c[j]$ and standard deviation s. After initializing the population uniformly at random, we evaluate these P points and find their Pareto Set S; then we continue as follows until our evaluation budget is used up. (i) generate and evaluate a new population of P points using the procedure above; (ii) update the Pareto Set S, and return to (i).

So, in SGM, a simple probabilistic model is learned, based on the current approximation to the Pareto Front. Considering only the PF points, each parameter is modelled independently as an equally weighted mixture of Gaussians each with standard deviation s, with one Gaussian per point in the current Pareto Set approximation, centred on that point's value for the parameter in question. In our experiments we set $P = 10$ and use $s = 0.1, 0.3, 1.0$ as for SMH.

3.6 PESA-II

The Pareto Envelope based Selection Algorithm (PESA-II) [5] is one of the several multiobjective evolutionary algorithms that emerged in the resurgence of interest in this

field in the late 90s. PESA-II attempts to find good approximations to the Pareto front by maintaining a datastructure that keeps track of the density of solutions across its current Pareto front approximation. The objective space is divided into 'hyperboxes', and selection of points for further exploration is guided by the relative crowding of hyperboxes, preferring to explore (i.e. use as parents for crossover or mutation) areas that currently have low density. We use it here as a convenient example of one of the several MOEAs that, in noise-free unlimited-evaluations budget scenarios at least, is quite proficient. [5] explains PESA-II in detail. We set the key parameters of PESA-II in our experiments as follows: population size $IP = 10$; archive size $EP = 100,\ 250$; number of hyperboxes=10^k, where k is number of objectives; binary representation of parameters (30 bits per parameter); $1/L$ bit-flip mutation rate, where L is bitstring length; uniform crossover applied at a rate of 0.2.

4 Testing Regime and Procedures

Our testing regime and procedures are informed by both real-world problems/ applications of interest to us, and what is available and best practice in the MOEA literature. The numbers of function evaluations available, the number of real-valued variables that we consider (less than 9) and the noise levels are all typical of real problems in mass-spectrometer optimization (e.g. [28]) as well as the optimization of chemical mixtures [6,12] and process optimization problems such as the one used in [1].

Test Functions. Originally from four sources, our test functions are those used and described fully in [22][1]. They range in dimension from 2 to 8 decision variables, and are all 2- or 3-objective problems. Note that the DTLZa functions we use are derived from the popular DTLZ ones, but we have reduced the number of parameters commensurate with the limited number of evaluations we are using, and also further reduced the difficulty of DTLZ1 by lessening the ruggedness of the function. (This lessening of the dimension/difficulty of the functions was done entirely independently and before any optimization was begun.)

Noise Model. We apply additive Gaussian noise to the objective function values before passing the values to the optimization algorithm. Repeated evaluation of the same point would therefore yield different results. We test three noise levels, 0%, 10% and 30%. 10% noise, for example, indicates that the objective value is perturbed by a Gaussian with mean zero and standard deviation of 10% of the cost function's range. When the output of the optimization algorithm is measured and compared, we use the true underlying (noiseless) objective values. This makes sense in the case that the noise represents just (unbiased) measurement error, but that underlying differences are important or, equally, the case that the noise represents natural variation in the measured objective, but we are interested in the expected value of this, e.g. the average yield that a chemical process would give over the long term, given some setting (see [3], section 3.1 Type C uncertainty).

[1] We consider here 8 of the 9, having dropped VLMOP2, due only to space limitations.

KNO1 [22] Features: Two decision variables; two objectives; Fifteen locally optimal Pareto fronts.
OKA1 [29] Features: Two decision variables; two objectives; Pareto optima lie on curve; density of solutions low at PF.
OKA2 [29] Features: Three decision variables; two objectives; Pareto optima lie on spiral-shaped curve; density of solutions very low at PF.
VLMOP3 [34] 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 [8] Features: Six decision variables; two objectives; local optima on the way to the PF.
DTLZ2a and DTLZ4a, adapted from [8] 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 [8] Features: Eight decision variables, three objectives; four disconnected regions in the Pareto front (in objective space).

Fig. 3. Summary of the eight test functions

Performance Assessment. Performance assessment of multiobjective optimizers is well known to be a nontrivial task [20,35] owing largely to the fact that the result of an optimization is a *set* of points, defining an approximation to a Pareto surface, and pairs of such surfaces (e.g. from the results of different algorithms) are commonly incomparable. Following the analysis in [20], we use Jaszkiewicz and Hansen's R metrics, which tend to dominate alternatives in terms of their profile of desirable properties. They tend to avoid being biased in favour of a particular property of a Pareto set approximation, (such as cardinality or uniformity), they do not rely on knowledge of the true Pareto front, and they are relatively scalable to many-objectives. They require using, however, a (relatively arbitrary) reference set of nondominated points for any given problem. Given the reference set and a set of points S output from an optimization run, an R metric provides a single scalar value that estimates the 'utility' of S. We mainly use R_3, but resort to R_2 in two cases where the R_3 measure led to excessive standard deviations, arising from vagaries of the relationship between certain result sets and the chosen reference sets.

5 Results

We compared ParEGO, TOMO, SGM, SMH, PESA-II, LH and RS, with budgets of 100 and 250 evaluations. All parameters of the algorithms have been given in Section 3, but recall SGM and SMH are each tried with three values of their standard deviation parameter: 0.1, 0.3 and 1.0; the other algorithms have no free parameters.

For every (algorithm, test-function, max-evals, noise-level) tuple, 21 independent runs were done. The use of an odd number of runs allows for plots of median attainment surfaces, although space precludes that here. Results tables show the mean and standard deviation of the R_3 metric values for each tuple. Our reference sets[2] and R-metric code are available from the first author's web space, so the tabulated values allow others to directly compare their algorithms with those tested here.

[2] Reference sets were generated using PESA-II runs of 50,000 evaluations.

Table 1. Results for function DTLZ1a, DTLZ2a, DTLZ4a and DTLZ7a - each entry provides mean and standard deviation of Hansen's R_3 metric (R_2 for DTLZ1a) based on 21 runs per algorithm

	100 evals	250 evals	100 evals 10% noise	250 evals 10% noise	100 evals 30% noise	250 evals 30% noise
Function	DTLZ1a					
RS	0.033 (0.01)	0.024 (0.01)	0.033 (0.01)	0.024 (0.01)	0.033 (0.01)	0.024 (0.01)
LH	0.030 (0.01)	0.024 (0.01)	0.030 (0.01)	0.024 (0.01)	0.030 (0.01)	0.024 (0.01)
SMH-0.1	0.013 (0.01)	0.006 (0.00)	0.035 (0.02)	0.021 (0.01)	0.050 (0.02)	0.040 (0.02)
SGM-0.1	0.008 (0.00)	0.003 (0.00)	0.036 (0.02)	0.024 (0.02)	0.071 (0.05)	0.063 (0.04)
PESA2	0.010 (0.01)	**0.000** (0.00)	0.050 (0.02)	0.045 (0.02)	0.112 (0.05)	0.101 (0.06)
ParEGO	**0.001** (0.00)	0.004 (0.00)	**0.002** (0.01)	**0.012** (0.05)	**0.006** (0.00)	**0.006** (0.00)
TOMO	0.011 (0.00)	0.001 (0.00)	0.022 (0.01)	**0.012** (0.01)	0.026 (0.01)	0.017 (0.01)
ParEGO vs TOMO	100/0, 100/30, 250/30 ParEGO wins 99.95; 250/0 TOMO wins 99.95					
SGM-0.1 vs SMH-0.1	100/0, 250/0 SGM wins 97.5; 100/30, 250/30 SMH wins 99.75					
Function	DTLZ2a					
RS	0.367 (0.21)	0.157 (0.33)	0.367 (0.21)	0.157 (0.33)	0.367 (0.21)	0.157 (0.33)
LH	0.232 (0.01)	0.227 (0.11)	0.232 (0.01)	0.227 (0.11)	**0.232** (0.01)	0.227 (0.11)
SMH-0.3	0.305 (0.06)	0.254 (0.04)	0.297 (0.07)	0.294 (0.20)	0.295 (0.07)	0.235 (0.10)
SGM-0.3	0.317 (0.63)	0.057 (0.10)	0.282 (0.09)	0.298 (0.38)	0.443 (0.31)	0.309 (0.05)
PESA2	0.547 (0.53)	0.509 (0.68)	0.535 (0.52)	0.468 (0.67)	0.683 (0.36)	0.646 (0.40)
ParEGO	0.511 (1.14)	-0.032 (0.02)	0.317 (0.63)	**0.057** (0.10)	0.290 (0.26)	0.195 (0.10)
TOMO	**0.182** (0.46)	**-0.137** (1.26)	**0.141** (0.73)	0.289 (0.62)	0.352 (0.32)	**0.136** (0.43)
ParEGO vs TOMO	250/10 ParEGO wins 90					
SGM vs SMH	100/0, 250/0 SGM wins 99.95 ; 100/30, 250/30 SMH wins 99.95					
Function	DTLZ4a					
RS	0.524 (0.08)	0.574 (0.50)	0.524 (0.08)	0.574 (0.50)	0.524 (0.08)	0.574 (0.50)
LH	0.534 (0.11)	0.438 (0.12)	0.534 (0.11)	0.438 (0.12)	0.534 (0.11)	0.438 (0.12)
SMH-0.3	0.264 (0.06)	0.212 (0.04)	0.257 (0.04)	0.221 (0.03)	**0.265** (0.04)	0.245 (0.05)
SGM-0.3	**0.259** (0.07)	0.201 (0.02)	**0.235** (0.04)	**0.211** (0.03)	0.311 (0.09)	0.255 (0.05)
PESA2	0.659 (0.37)	0.549 (0.10)	0.582 (0.10)	0.564 (0.10)	0.598 (0.11)	0.594 (0.04)
ParEGO	0.508 (0.20)	**0.148** (0.13)	0.557 (0.05)	0.223 (0.11)	0.445 (0.13)	**0.201** (0.28)
TOMO	0.616 (0.44)	0.423 (0.25)	0.529 (0.10)	0.529 (0.07)	0.677 (0.43)	0.428 (0.22)
ParEGO vs TOMO	100/30, 250/0, 250/10, 250/30 ParEGO wins 99.5					
SGM-0.3 vs SMH-0.3	100/10 SGM wins 95 ; 100/30 SMH wins 97.5					
Function	DTLZ7a					
RS	2.480 (8.83)	1.940 (9.20)	2.480 (8.83)	1.940 (9.20)	2.480 (8.83)	1.940 (9.20)
LH	0.58 (0.05)	0.489 (0.17)	0.58 (0.05)	0.489 (0.17)	0.58 (0.05)	0.489 (0.17)
SMH-1.0	0.553 (0.09)	0.490 (0.05)	0.613 (0.13)	0.510 (0.19)	0.616 (0.08)	0.508 (0.10)
SGM-1.0	0.538 (0.06)	0.545 (1.12)	0.622 (0.07)	0.580 (0.07)	0.656 (0.08)	0.633 (0.08)
PESA2	0.517 (0.16)	0.212 (0.63)	0.650 (0.32)	0.570 (0.22)	0.681 (0.09)	0.332 (0.19)
ParEGO	0.573 (0.04)	0.232 (0.07)	0.535 (0.13)	0.307 (0.42)	0.561 (0.05)	0.403 (0.15)
TOMO	0.501 (0.11)	**-0.442** (2.71)	**0.497** (0.27)	**0.136** (0.53)	**0.451** (0.87)	**0.323** (1.23)
ParEGO vs TOMO	100/0 TOMO wins 99					
SGM-1.0 vs SMH-1.0	100/30, 250/10, 250/30 SMH wins 90					

We compare ParEGO and TOMO directly, using t-tests (assuming unequal variances) on the R values. Similarly, we compare a selected pair of parameter variants of SGM and SHM per test function (we choose those that performed best with noise). Further statistical comparisons (e.g. ParEGO vs LH) without generating further independent sets of results would amount to multiple testing and be statistically invalid (we could additionally compare LH and RS, but we omit that for reasons of space and salience). Instead, we calculate 'naive' rank orderings of the algorithms for each (test-function,

Table 2. Results for functions VLMOP3, KNO1, OKA1 and OKA2 - each table entry provides mean and standard deviation of Hansen's R_3 metric based on 21 runs per algorithm

	100 evals	250 evals	100 evals 10% noise	250 evals 10% noise	100 evals 30% noise	250 evals 30% noise
Function	VLMOP3					
RS	0.290 (0.15)	0.132 (0.07)	0.290 (0.15)	0.132 (0.07)	0.290 (0.15)	0.132 (0.07)
LH	0.213 (0.137)	0.146 (0.08)	0.213 (0.137)	0.146 (0.08)	0.213 (0.137)	0.146 (0.08)
SMH-0.3	0.269 (0.15)	0.133 (0.08)	0.238 (0.18)	0.173 (0.12)	0.271 (0.17)	0.158 (0.07)
SGM-0.1	0.095 (0.08)	0.042 (0.05)	0.157 (0.11)	0.101 (0.24)	0.277 (0.22)	0.164 (0.16)
PESA2	0.169 (0.22)	0.086 (0.17)	0.260 (0.24)	0.156 (0.22)	0.256 (0.25)	0.181 (0.22)
ParEGO	**0.019** (0.01)	**0.013** (0.00)	0.058 (0.05)	0.029 (0.01)	0.080 (0.08)	**0.033** (0.01)
TOMO	0.024 (0.01)	0.026 (0.03)	**0.031** (0.01)	**0.017** (0.00)	**0.078** (0.09)	0.034 (0.05)
ParEGO vs TOMO	100/0, 250/0 ParEGO wins 90; 100/10, 250/10 TOMO wins 99					
SGM-0.1 vs SMH-0.3	100/0, 100/10 250/0 SGM wins 90					
Function	KNO1					
RS	0.012 (0.08)	-0.126 (0.07)	0.012 (0.08)	-0.126 (0.07)	0.012 (0.08)	-0.126 (0.07)
LH	-0.033 (0.1)	-0.142 (0.08)	-0.033 (0.01)	-0.142 (0.08)	-0.033 (0.1)	-0.142 (0.08)
SMH-0.3	-0.107 (0.10)	**-0.282** (0.08)	**-0.130** (0.09)	-0.258 (0.09)	-0.117 (0.11)	**-0.268** (0.09)
SGM-0.3	-0.106 (0.13)	-0.224 (0.09)	-0.001 (0.14)	-0.096 (0.12)	0.189 (0.18)	0.107 (0.16)
PESA2	0.112 (0.14)	0.085 (0.16)	0.177 (0.13)	0.138 (0.15)	0.299 (0.11)	0.272 (0.14)
ParEGO	-0.049 (0.10)	-0.137 (0.11)	-0.128 (0.11)	**-0.290** (0.07)	**-0.120** (0.11)	-0.265 (0.07)
TOMO	**-0.127** (0.12)	-0.191 (0.14)	-0.129 (0.11)	-0.216 (0.12)	-0.074 (0.10)	-0.196 (0.12)
ParEGO vs TOMO	100/0, 250/0 TOMO wins 90 ; 100/30, 250/10, 250/30 ParEGO wins 90					
SGM-0.3 vs SMH-0.3	100/10, 100/30, 250/0, 250/10, 250/30 SMH wins 97.5					
Function	OKA1					
RS	0.376 (0.04)	0.310 (0.04)	0.376 (0.04)	0.310 (0.04)	0.376 (0.04)	0.310 (0.04)
LH	0.380 (0.03)	0.302 (0.38)	0.380 (0.02)	0.302 (0.38)	0.380 (0.03)	0.302 (0.38)
SMH-0.3	0.339 (0.04)	0.289 (0.03)	0.348 (0.06)	0.302 (0.05)	0.380 (0.06)	0.301 (0.06)
SGM-1.0	0.351 (0.03)	0.321 (0.02)	0.396 (0.05)	0.375 (0.05)	0.440 (0.06)	0.437 (0.05)
PESA2	0.354 (0.07)	0.266 (0.05)	0.447 (0.07)	0.417 (0.08)	0.562 (0.08)	0.540 (0.08)
ParEGO	**0.071** (0.03)	**0.071** (0.03)	**0.211** (0.09)	**0.086** (0.03)	**0.302** (0.04)	**0.195** (0.04)
TOMO	0.273 (0.05)	0.219 (0.06)	0.303 (0.04)	0.198 (0.06)	0.330 (0.12)	0.296 (0.06)
ParEGO vs TOMO	100/0, 100/10, 250/0, 250/10, 250/30 ParEGO wins 99.95					
SGM-1.0 vs SMH-0.3	100/10, 100/30, 250/0, 250/10, 250/30 SMH wins 99.5					
Function	OKA2					
RS	0.458 (0.03)	0.410 (0.03)	0.458 (0.03)	0.410 (0.03)	0.458 (0.03)	0.410 (0.03)
LH	0.440 (0.04)	0.411 (0.03)	0.440 (0.04)	0.411 (0.03)	0.440 (0.04)	0.411 (0.03)
SMH-1.0	0.303 (0.04)	0.241 (0.04)	0.304 (0.06)	0.248 (0.06)	**0.316** (0.05)	0.263 (0.06)
SGM-1.0	0.271 (0.05)	0.237 (0.05)	0.335 (0.08)	0.285 (0.07)	0.410 (0.08)	0.382 (0.05)
PESA2	0.364 (0.09)	0.251 (0.10)	0.488 (0.06)	0.456 (0.07)	0.590 (0.13)	0.555 (0.14)
ParEGO	**0.146** (0.05)	**0.058** (0.04)	**0.245** (0.06)	**0.070** (0.04)	0.330 (0.07)	**0.251** (0.06)
TOMO	0.354 (0.07)	0.307 (0.07)	0.414 (0.04)	0.331 (0.03)	0.444 (0.04)	0.391 (0.04)
ParEGO vs TOMO	100/0, 100/10, 250/0, 250/10, 250/30 ParEGO wins 99.95					
SGM-1.0 vs SMH-1.0	100/0, SGM wins 97.5 ; 100/10, 100/30, 250/10, 250/30 SMH wins 90					

max-evals, noise-level) triple, based only on mean R values, and we build a summary table of mean ranks for each algorithm in different scenarios. This leads to a series of overall indicative observations, and which we feel provide valuable insight and pointers to further work.

Tables 1 and 2 present summary results on each of the test functions. In each case, Random Search (RS) results for the noise cases are shown for convenience, though they are necessarily identical to the non-noise cases. To save space, only the 'best' of the three variants each of SGM and SMH are shown in the tables. To support

Table 3. A broad summary of the effects of number of evaluations and of levels of noise on an algorithm's mean naive rank over the test functions studied

mean rank	RS	LH	SGM	SMH	PESA2	ParEGO	TOMO
overall	5.4	4.6	3.8	3.4	6.0	**2.1**	2.6
100 evals / no noise	6.1	5.3	**2.4**	3.5	5.4	2.6	2.8
100 evals / 10% noise	5.5	4.1	3.4	3.4	6.7	2.8	**2.1**
100 evals / 30% noise	5.4	3.6	4.9	2.6	6.6	**1.8**	3.1
250 evals / no noise	6.3	6.0	3.0	3.5	4.5	**2.3**	2.5
250 evals / 10% noise	4.6	4.5	3.9	3.9	6.6	**1.5**	2.9
250 evals / 30% noise	4.6	4.1	5.3	3.5	6.4	**1.6**	2.4

understanding the tables, we interpret parts of Table 1 as follows: On DTLZ1a, we see that SGM, with standard deviation 0.1, achieved a mean R_2 value of 0.063, (with R values, lower is always better) with a standard deviation of 0.04, in the 250-evaluations limit case with 30% noise. When we compare ParEGO and TOMO on DTLZ1a, we find the following scenarios in which ParEGO outperformed TOMO with statistical confidence at least 90%: 100 evals at no noise and 30% noise, and 250 evaluations at 30% noise - among these cases, the lowest level of confidence was 99.95%. Meanwhile TOMO outperforms ParEGO in the 250-evaluations no-noise case, with confidence 99.95%; in the cases not mentioned (100/10, 250/10) the comparisons were not significant with $\geq 90\%$ confidence.

Table 3 provides a broad summary of the observations that we can make on the basis of the naive rank orderings of the algorithms for each (test-function, max-evals, noise-level) scenario. Naive rank orderings are based only on mean R value; in general, they either have no statistical significance, or have significance but at a low confidence level. For a given scenario (e.g. 100 evals, no noise) we rank algorithms from 1 to 7, considering, for any particular problem, only the best of the three SGM variants, and the best of the three SMH variants for that problem. Hence, for example, in the 250-evaluations 30%-noise case on problem OKA1 (table 2), ParEGO is best with rank 1, TOMO has rank 2, SMH has rank 3, and so on, until PESA-II has rank 7. The table indicates the mean ranks for each scenario over the eight test problems.

Running Times. It is worth noting that both TOMO and ParEGO do consume significant resources to compute each solution to evaluate next, and that this time grows with each iteration. On a single-core Pentium III 2.8GHz desktop machine, they require of the order of 10s per evaluation at the end of a 250 evaluation run. Although limited budgets tend to arise when the time to evaluate a solution is considerably larger than this, one can certainly envisage some budget-constrained scenarios where such a lag would be unacceptable. For this reason, the performance of the baselines, which all have negligible runtimes in comparison to TOMO and ParEGO, are of more than incidental interest.

6 Concluding Discussion

As we suggested towards the end of Section 2, it is not surprising that a metamodel-based technique should do well in the limited-evaluations regime. However, the

question of performance in the presence of noise is rather less clear *a priori*. Metamodels repeatedly rely on the positions and fitnesses of samples previously visited (each evaluated only once) in order to build a picture that guides choice of the next sample point. Noise can be expected to mislead this model, and with few evaluations available there is little opportunity for recovery from this. ParEGO / EGO is no exception to this, as it does not explicitly account for uncertainty in evaluated points [15], indeed using a model that interpolates between these points and a sampling method which ensures they will *never* be re-evaluated.

As it turns out, however, ParEGO stands up to noise much more successfully than the other techniques tested; from the indications in Table 3, especially so when given at least the luxury of 250 evaluations, and especially at the higher noise levels. In contrast, SGM and PESA-II get very confused by noise. Both may have done a little better using a larger population size (and hence fewer generations) in the noise cases, but it seems that the strategies inherent in both of these techniques place undue trust in the accuracy of points visited so far. TOMO is clearly the second-best technique tested overall, regularly outperformed by ParEGO, although TOMO seems to have the edge over ParEGO at 100 evaluations and 10% noise. This suggests various ways forward for coping with such severely-limited budgets, such as tweaking the EGO model to account for noisy evaluations (as in [15] for single-objective optimization), or basing sampling decisions on evidence obtained from both ParEGO's and TOMO's strategies, weighting them appropriately given the number of evaluations so far. Meanwhile, many further parametric and design variants of TOMO can be explored, perhaps most pressingly an evaluation of the many ways scalarization could be done more adaptively, e.g. using goal programming.

As for SGM and SMH, herein they have fulfilled a need to provide further alternative strategies, continuing to investigate whether the sophistication inherent in ParEGO (and TOMO) can be undermined by a simpler (and possibly faster) alternative. As it turns out, it seems that the SGM approach appears quite useful when noise is absent, at least in the 100-evaluations regime. Parameter dependence limits this observation, however it could be suggested that an adaptive version of SGM may extend its niche of good performance toward 250 evaluations.

Finally, we note that PESA-II's performance seems generally awful; this adds weight to findings elsewhere (e.g. with NSGA-II in [22]) that standard modern MOEAs, designed with perhaps '0,000s or '000,000s of evaluations in mind, are simply inapplicable when much more limited evaluation numbers are available. However, it must be said that PESA-II's parameterization was not optimized here and the use of a binary representation is almost certainly unfair. We mention in passing that a fitness-inheritance based version of PESA-II was tested in preliminary work, and found to work fine when many '000s of evaluations were available, however the beneficial effect of fitness inheritance simply failed to be present below $\sim$ 1,000 evaluations.

References

1. Anderson, B., Moore, A., Cohn, D.: A nonparametric approach to noisy and costly optimization. In: Langley, P. (ed.) Proc. 17th ICML, pp. 17–24. Morgan Kaufmann, San Francisco (2000)
2. Beyer, H.-G.: Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice. Computer Methods in Applied Mechanics and Engineering 186(2-4), 239–267 (2000)

3. Beyer, H.-G., Sendhoff, B.: Robust optimization: A comprehensive survey. Computer Methods in Applied Mechanics and Engineering 196(33-34), 3190–3218 (2007)
4. Chen, J.-J., Goldberg, D.E., Ho, S.-Y., Sastry, K.: Fitness inheritance in multi-objective optimization. In: Proc. GECCO 2002, pp. 319–326. Morgan Kaufmann, San Francisco (2002)
5. Corne, D., Jerram, N., Knowles, J., Oates, M.: PESA-II: Region-based selection in evolutionary multiobjective optimization. In: GECCO 2001, pp. 283–290. Morgan Kaufmann, San Francisco (2001)
6. Davies, Z.S., Gilbert, R.J., Merry, R.J., Kell, D.B., Theodorou, M.K., Griffith, G.W.: Efficient improvement of silage additives by using genetic algorithms. In: Applied and Environmental Microbiology, pp. 1435–1443 (2000)
7. Deb, K., Goldberg, D.: An Investigation of Niche and Species Formation in Genetic Function Optimization. In: Proc. 3rd International Conference on Genetic Algorithms, pp. 42–50. Morgan Kaufmann, San Francisco (1989)
8. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: 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)
9. Ducheyne, E.I., De Baets, B., De Wulf, R.: Is fitness inheritance useful for real-world applications? In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 31–42. Springer, Heidelberg (2003)
10. Dunn, E., Olague, G.: Multi-objective Sensor Planning for Efficient and Accurate Object Reconstruction. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 312–321. Springer, Heidelberg (2004)
11. Emmerich, M., Naujoks, B.: Metamodel Assisted Multiobjective Optimisation Strategies and their Application in Airfoil Design. In: Parmee, I. (ed.) Adaptive Computing in Design and Manufacture VI, pp. 249–260. Springer, Heidelberg (2004)
12. Evans, J.R.G., Edirisinghe, M.J., Eames, P.V.C.J.: Combinatorial searches of inorganic materials using the inkjet printer: science philosophy and technology. Journal of the European Ceramic Society 21, 2291–2299 (2001)
13. Gaspar-Cunha, A., Vieira, A.S.: A hybrid multi-objective evolutionary algorithm using an inverse neural network. In: Hybrid Metaheuristics (HM 2004) Workshop at ECAI 2004, pp. 25–30 (2004), http://iridia.ulb.ac.be/~hm2004/proceedings/
14. Hornby, G.S., Takamura, S., Yamamoto, T., Fujita, M.: Autonomous evolution of dynamic gaits with two quadruped robots. IEEE Transactions on Robots 21(3), 402–410 (2005)
15. Huang, D., Allen, T.T., Notz, W.I., Zeng, N.: Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models. Journal of Global Optimization 34(3), 441–466 (2006)
16. Jeong, S., Minemura, Y., Obayashi, S.: Optimisation of combustion chamber for diesel engine using kriging model. Journal of Fluid Science and Technology 1(2), 138–146 (2006)
17. Jeong, S., Suzuki, K., Obayashi, S., Kirita, M.: Improvement of nonlinear lateral characteristics of lifting-body type reentry vehicle using optimization algorithm. In: Proc. of AIAA Infotech-Aerospace Conference 2007, pp. 1–15. AIAA (2007)
18. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing-A Fusion of Foundations, Methodologies and Applications 9(1), 3–12 (2005)
19. Jones, D., Schonlau, M., Welch, W.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13, 455–492 (1998)
20. Knowles, J., Corne, D.: On metrics for comparing nondominated sets. In: Congress on Evolutionary Computation (CEC 2002), Piscataway, New Jersey, vol. 1, pp. 711–716. IEEE Service Center, Los Alamitos (2002)
21. Knowles, J., Hughes, E.J.: Multiobjective Optimization on a Budget of 250 Evaluations. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 176–190. Springer, Heidelberg (2005)

22. Knowles, J.: ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans. Evol. Comp. 10(1), 50–66 (2006)
23. Knowles, J., Nakayama, H.: Meta-Modeling in Multiobjective Optimization. In: Branke, D., Deb, K., Miettinen, S., Słowiński, R. (eds.) Multiobjective Optimization: Interactive and Evolutionary Approaches. LNCS, vol. 5252. Springer, Heidelberg (2008)
24. Laumanns, M., Ocenasek, J.: Bayesian optimization algorithms for multi-objective optimization. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 298–307. Springer, Heidelberg (2002)
25. Miettinen, K.M.: Nonlinear Multiobjective Optimization. Kluwer, Dordrecht (1999)
26. Nain, P.K.S., Deb, K.: 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)
27. Nakayama, H., Yun, Y.: Multi-objective Model Predictive Optimization using Computational Intelligence. In: Artificial Intelligence in Theory and Practice II, pp. 319–328. Springer, Heidelberg (2008)
28. O'Hagan, S., Dunn, W., Knowles, J., Broadhurst, D., Williams, R., Ashworth, J., Cameron, M., Kell, D.: Closed-loop, multiobjective optimization of two-dimensional gas chromatography/mass spectrometry for serum metabolomics. Analytical Chemistry 79(2), 464–476 (2007)
29. Okabe, T., Jin, Y., Olhofer, M., Sendhoff, B.: On Test Functions for Evolutionary Multiobjective Optimization. In: Parallel Problem Solving from Nature - PPSN VIII, pp. 792–802. Springer, Heidelberg (2004)
30. Ong, Y.S., Nair, P.B., Keane, A.J., Zhou, Z.Z.: Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In: Jin, Y. (ed.) Knowledge Incorporation in Evolutionary Computation. Springer, Heidelberg (2004)
31. Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press, Cambridge (1992)
32. Sacks, J., Welch, W., Mitchell, T., Wynn, H.: Design and analysis of computer experiments (with discussion). Statistical Science 4, 409–435 (1989)
33. Bosman, P.A.N., Thierens, D.: Multi-objective Optimization with the Naive MIDEA. Studies in Fuzziness and Soft Computing 192, 123–157 (2006)
34. van Veldhuizen, D.A., Lamont, G.B.: Multiobjective Evolutionary Algorithm Test Suites. In: Proc. 1999 ACM Symposium on Applied Computing, pp. 351–357. ACM, New York (1999)
35. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)