MOI-MBO: Multiobjective Infill for Parallel Model-Based Optimization

Bernd Bischl[1(✉)], Simon Wessing[2], Nadja Bauer[1], Klaus Friedrichs[1], and Claus Weihs[1]

[1] Department of Statistics, TU Dortmund, Dortmund, Germany
{bischl,bauer,friedrichs,weihs}@statistik.tu-dortmund.de
[2] Department of Computer Science, TU Dortmund, Dortmund, Germany
simon.wessing@tu-dortmund.de

Abstract. The aim of this work is to compare different approaches for parallelization in model-based optimization. As another alternative aside from the existing methods, we propose using a multi-objective infill criterion that rewards both the diversity and the expected improvement of the proposed points. This criterion can be applied more universally than the existing ones because it has less requirements. Internally, an evolutionary algorithm is used to optimize this criterion. We verify the usefulness of the approach on a large set of established benchmark problems for black-box optimization. The experiments indicate that the new method's performance is competitive with other batch techniques and single-step EGO.

1 Introduction

Efficient optimizers that work on a strictly reduced budget of function evaluations are crucial for parameter optimization of expensive black-box functions, such as industrial simulators or time-consuming algorithms. Classical optimization methods in design of experiments are based on the assumption of a simple (often linear or quadratic) relationship between input parameters and performance output. In this case, the optimal set of evaluation points to fit such a model can usually be specified in advance.

However, for computer experiments, these simple models often do not suffice, as their assumptions are often severely violated, leading to unsatisfying results if they are employed nevertheless. Therefore, general sequential model-based optimization (MBO) is a standard technique for cost expensive simulations nowadays. Here, evaluation points are proposed sequentially using an appropriate surrogate model that allows for nonlinear relationships. After defining an initial set of evaluation points, e.g., a space-filling latin hypercube design, the basic procedure of MBO is an iterating loop of the following steps: firstly, a model is fitted on the evaluated points; secondly, a new evaluation point is proposed by an infill criterion; and lastly, its performance is evaluated.

In the last decade, many MBO procedures were proposed and compared relying on kriging models. Kriging is usually employed when only continuous

P.M. Pardalos et al. (Eds.): LION 2014, LNCS 8426, pp. 173–186, 2014.
DOI: 10.1007/978-3-319-09584-4_17

inputs are available. But particularly in the context of algorithm configuration, although kriging has been applied successfully in this domain, see [1] for an example, recently random forest surrogate models received attention, because of their capability to handle categorical parameters [2]. For an example how to integrate model selection into MBO see [3].

Instead of just evaluating one point in each iteration, a batch-sequential extension, which enables parallel evaluation of several points, is natural because of modern multi-core architectures. While there are already some recent approaches for multi-point MBO, the idea proposed in this paper is to use a new multiobjective perspective by considering multiple infill criteria simultaneously without aggregating them into a single criterion. Instead of searching for one optimal evaluation point, an approximate Pareto front of q points is generated, which contains a spectrum of trade-offs between the different criteria. Possible single-objective infill criteria are mean prediction or local uncertainty of the surrogate model, and we also consider distances of the points in the current batch to each other to ensure diverse new evaluations.

In Sect. 2, MBO is defined more formally and state-of-the-art approaches are described, including several ones for parallel MBO. Our proposal using multiobjectivization is explained in Sect. 3. In Sect. 4, the conducted comparison experiments are described, and in Sect. 5, their results are presented and interpreted. Finally, in Sect. 6, the most important findings are summarized.

2 Model-Based Optimization

2.1 The Basic Sequential Algorithm

Let us assume that we aim to minimize an expensive black-box function $f : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$, $f(\boldsymbol{x}) = y$, $\boldsymbol{x} = (x_1, \ldots, x_d)^T$. Each x_i is a continuous parameter with box constraints $[\ell_i, u_i]$, $\mathcal{X} = [\ell_1, u_1] \times \ldots \times [\ell_d, u_d]$ is the parameter space of $\boldsymbol{x}$, and y is the target value. An important distinction is whether we are observing a deterministic output y or one that is corrupted by noise, i.e., whether our observed target values are actually realizations of a random variable. In this paper, we will only study the noiseless, deterministic case. With $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$, we will later denote an indexed set (design) of n different points $\boldsymbol{x}_i \in \mathcal{X}$ and $\boldsymbol{y} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))^T$, the vector of associated target values.

The main idea in model-based optimization is to approximate the expensive function $f(\boldsymbol{x})$ in every iteration by a regression model, which is much cheaper to evaluate. This is also called a meta-model or surrogate. Such a regression model often not only provides a direct estimation $\hat{f}(\boldsymbol{x})$ of the true function value $f(\boldsymbol{x})$ but also an estimation of the prediction standard error $\hat{s}(\boldsymbol{x})$, also called a local uncertainty measure. This value allows to assess the "trustworthiness" of the prediction, or, in a more Bayesian terminology, the spread of the posterior distribution of $\hat{f}(\boldsymbol{x})$.

An outline of sequential model-based optimization (MBO) is given in Algorithm 1. We start by exploring the parameter space with an initial design, often

Algorithm 1. Sequential model-based optimization

```
Generate an initial design 𝒟 ⊂ 𝒳;
Compute 𝒚 = f(𝒟);
while total evaluation budget is not exceeded do
    Fit surrogate on 𝒟 and obtain f̂, ŝ;
    Get new design point 𝒙* by optimizing the infill criterion based on f̂, ŝ;
    Evaluate new point y* = f(𝒙*);
    Update: 𝒟 ← (𝒟, 𝒙*) and 𝒚 ← (𝒚, y*);
return y_min = min(𝒚) and the associated 𝒙_min.
```

constructed in a space-filling fashion. The main sequential loop can be divided into two alternating stages: Fitting the response surface to the currently available design data, then optimizing the so-called infill criterion to propose a new promising point $\boldsymbol{x}^*$ for the next expensive evaluation $f(\boldsymbol{x}^*)$.

Quite a few infill criteria exist, both for the deterministic and noisy case. They are usually constructed point-wise by combining $\hat{f}(\boldsymbol{x})$ and $\hat{s}(\boldsymbol{x})$ in a certain way. For the former, lower values are more promising, as they indicate a low true function value $f(\boldsymbol{x})$. For the latter, higher values indicate less explored regions of the search space, as our model is less certain about the true landscape, usually because it lacks training points nearby. The task of many infill criteria is to balance these two conflicting criteria into one numerical formula.

Probably the simplest infill criterion, which can be used even if no local uncertainty estimator is available, is just considering the mean prediction $\hat{f}(\boldsymbol{x})$. This results in a greedy behavior, where promising regions are exploited at once and can quickly result in only local convergence.

In their seminal paper, Jones et al. [4] recommended to use Kriging [5], i.e., a Gaussian process, for regression. This model can fit multimodal landscapes with satisfying quality, even when only a low amount of data points is available. As a kernel method, it also offers flexibility, and roughness information regarding the target function can be encoded into the model via the covariance kernel. The posterior distribution of $\hat{f}(\boldsymbol{x})$ is now a univariate $N(\hat{f}(\boldsymbol{x}), \hat{s}(\boldsymbol{x})^2)$ one. Based on this, the now standard *expected improvement* (EI) criterion was proposed, which supposedly ensures global convergence [6–8]. It is defined as

$$\begin{aligned}\mathrm{EI}(\boldsymbol{x}) &= \mathbb{E}[\max\{0, y_{min} - \hat{f}(\boldsymbol{x})\}] \\ &= (y_{min} - \hat{f}(\boldsymbol{x}))\ \Phi\left(\frac{y_{min} - \hat{f}(\boldsymbol{x})}{\hat{s}(\boldsymbol{x})}\right) + \hat{s}(\boldsymbol{x})\ \phi\left(\frac{y_{min} - \hat{f}(\boldsymbol{x})}{\hat{s}(\boldsymbol{x})}\right),\end{aligned}$$

where ϕ and Φ are the density and cumulative distribution function of the standard normal distribution, respectively. Hence, the sought point is $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \mathrm{EI}(\boldsymbol{x})$.

One further infill criterion which is relevant for this paper is the *lower confidence bound* (LCB) criterion. LCB combines the predicted mean response with the estimated standard error through a weighted sum:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathcal{X}} \text{LCB}(\boldsymbol{x}) = \arg\min_{\boldsymbol{x} \in \mathcal{X}} \ \hat{f}(\boldsymbol{x}) - \lambda \hat{s}(\boldsymbol{x}) \,.$$

The weighting parameter λ has to be selected by the user and wrong choices will not guarantee global convergence. Obviously, the LCB criterion has to be minimized. If $\lambda = 0$, LCB coincides with predicted mean value. The larger λ is chosen, the more attractive the unexplored regions of the search space become. Further infill criteria are discussed in [7].

2.2 Review of Parallel MBO Strategies

In the standard MBO procedure, only one point $\boldsymbol{x}^*$ is proposed in each sequential step (see line 5 of Algorithm 1). However, often dozens or hundreds of CPU cores are simultaneously available nowadays. In some cases, these processors can be used to parallelize a single function call of the simulator $f(\boldsymbol{x}^*)$, but in many cases this is not possible and such an evaluation has to be considered "atomic". It is hence essential to exploit this computational potential in a reasonably efficient way, suggesting a batch-sequential approach. The aim is to propose q (instead of only 1) new points $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_q^*$ in each iteration so that q expensive function evaluations can be performed in parallel. Some possibilities have already been proposed to achieve this parallelization. However, they all introduce drawbacks that we want to avoid.

Arguably the mathematically most intuitive way to tackle the problem is to directly extend the EI-criterion for q points (q-EI) [9]. While for the 2-EI case an analytic solution is provided [9], for $q > 2$ an expensive Monte-Carlo simulation was implemented in [9] and [10]. Recently, Chevalier and Ginsbourger [11] proposed an analytic approach to more efficiently compute the q-EI for moderate values of $q \leq 10$.

A simple alternative for multi-point proposal using the EI-criterion is given by [9], where in the first step the kriging model is fitted based on the real data and $\boldsymbol{x}_1^*$ is calculated according to the regular EI-criterion. Then, for $i = 2, \ldots, q$, a simple "guess" for $f(\boldsymbol{x}_i^*)$ is used to update the model in order to propose the subsequent point. This estimation could be $\hat{f}(\boldsymbol{x})$ or even a constant like y_{min} or y_{max}. The first option is called *kriging believer*, and the constant estimation of $\hat{f}(\boldsymbol{x})$ is called *constant liar*. One should note that although the expensive re-estimation of the covariance kernel parameters is not performed during the batch generation but only after its batch evaluation (although without formal justification), the EI still needs to be optimized sequentially q times, which can in itself be very time consuming for higher values of d and q.

Hutter et al. [12] introduce another strategy. They use the LCB criterion and sample a new λ value from an exponential distribution with mean 1 for each point in the batch. This defines q different infill objective functions, one for each desired new point, each one – in principle – encoding a different trade-off between mean and standard error prediction of the model. The simple, independent optimization of the single-objective LCB criteria (which can also be performed in parallel) hence leads to a batch of q promising points. While this

approach scales computationally very well with increasing q and is also extremely simple to implement, there is no guaranteed diversity in the generated batch, so different values for λ can still lead to very similar global optima locations in the LCB landscape.

The aim of this article is to propose and compare other strategies for multi-point MBO, based on multicriteria evolutionary algorithms. The next section will detail our approach.

3 Proposal

Multiobjective optimization considers itself with the task of finding Pareto-optimal solutions to a set of objective functions. This methodology has become a standard tool under the belt of many researchers in black-box optimization, and a lot of excellent overview literature exists. As an introduction, we would like to refer the reader to [13].

Infill criteria for model-based optimization often consist of two conflicting parts: the mean response $\hat{f}(\boldsymbol{x})$ and the local uncertainty estimation $\hat{s}(\boldsymbol{x})$. In the case of LCB, it is very obvious that these two functions actually constitute a bicriteria problem. They have simply been scalarized into a weighted sum, where setting the actual weighting parameter is left up to the experimenter. This is usually a hard task if no further information is provided, and the best setting might not even be a constant value, but instead depend on the stage of the optimization process. For EI one might argue that a formal motivation exists which dictates the specific formula, but this derivation assumes that all model assumptions of the Gaussian process hold, which might not be true in practice. Furthermore, one might want to transfer the principle to other non-parametric regression models, e.g., because of discrete parameters in the input space or the disadvantageous runtime scaling behavior of the kriging model fits. This is already done in the case of SMAC, where a random forest instead of a kriging model is used in combination with an EI criterion, but there is no reason to assume that the posterior distribution of forest predictions is really normally distributed or that the derivation of the EI criterion is still valid.

Instead of dealing with the hassle of appropriately aggregating two or more objective functions into a single one (as, e.g., for EI or LCB), we simply accept the multiobjective nature of the problem and use a multiobjective optimizer. As we want to employ parallelization, we focus on a posteriori methods that return not only a single point, but a set of points $\mathcal{P}$ that approximate the Pareto-set. We therefore choose $|\mathcal{P}| = q$, the number of points we want to process in parallel. In this case, it also suggests itself to use evolutionary algorithms (EA), as they are population-based approaches and can be used easily for multiobjective optimization.

So far, we have not discussed the fact that we need a set of q *distinct* points. Therefore, we add the distance to neighboring solutions as another objective, to respect the influence of the points on each other. By maximizing this distance, points are rewarded for being dissimilar to other solutions. The objective function

is thus dynamic, i.e., it depends on the EA's population and may change in each iteration of the optimization. If the distance is added as an artificial, secondary objective to a single-criterion problem, this approach is known under the name of multiobjectivization. The reason to proceed in such a fashion is that one might want to obtain a whole spectrum of promising, diverse solutions in the decision space, which is exactly what we want to achieve for multipoint proposal in MBO. This approach does not guarantee to result in a simpler problem, as shown by Brockhoff et al. [14], but gave decent results in practical applications similar to ours [15–17]. Summing up, the following objective functions are available for building our multiobjective infill (MOI) criterion:

- `mean`: mean model prediction $\hat{f}(\boldsymbol{x})$,
- `se`: standard error / model uncertainty $\hat{s}(\boldsymbol{x})$,
- `ei`: expected improvement $\mathrm{EI}(\boldsymbol{x})$,
- `dist.nn`: distance to the nearest neighbor of $\boldsymbol{x}$ in the current population

$$dist_{\mathrm{nn}}(\boldsymbol{x}, \mathcal{P}) = \min\{d(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \mid \tilde{\boldsymbol{x}} \in \mathcal{P} \setminus \{\boldsymbol{x}\}\},\ \mathcal{P} \subset \mathcal{X},$$

- `dist.nb`: distance to the nearest better neighbor of $\boldsymbol{x}$,

$$dist_{\mathrm{nb}}(\boldsymbol{x}, \mathcal{P}) = \min\{d(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \mid f(\tilde{\boldsymbol{x}}) < f(\boldsymbol{x}) \wedge \tilde{\boldsymbol{x}} \in \mathcal{P}\},\ \mathcal{P} \subset \mathcal{X}.$$

The latter distance definition is considered because it proved successful in experiments by Wessing et al. [17]. The distance measure $d(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ used throughout this paper is simply the euclidean distance $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2$. As there is in every set at least one solution $\boldsymbol{x}^\star$ that has no nearest better neighbor, we define $dist_{\mathrm{nb}}(\boldsymbol{x}^\star, \mathcal{P}) := \infty$.

Our multi-point MBO approach is geared to the evolutionary multiobjective optimization algorithm proposed by Beume et al. [18]. Algorithm 2 introduces the resulting optimization procedure.

The Function $crossover(\boldsymbol{x}, \tilde{\boldsymbol{x}}, \mathcal{X}, \eta_c, p_c)$ is the simulated binary crossover operator with a distribution index η_c and application probability p_c. The latter parameter specifies the probability to apply the operator to each variable. *crossover* produces one offspring from two parents. The function $mutate(\boldsymbol{x}, \mathcal{X}, \eta_m, p_m)$ is the polynomial mutation operator, which is applied to the offspring. Again, η_m denotes the distance parameter of the distribution and p_m the mutation probability for each variable. In this work, we fix these parameters to $\eta_c = \eta_m = 15$ and $p_c = p_m = 1$.

Lines 9–14 of Algorithm 2 describe the survivor selection. After the distance values have been incorporated, a non-dominated sorting of all individuals is conducted. Then, the worst individual of the worst non-dominated front is identified, according either to the hypervolume contribution (`hv`) [18] or according to the first single objective (`first`). The first single objective (see also experimental setup in Sect. 4,) will always be either `mean` or `ei`, as these are obviously most important to guide the optimization. The individual identified as worst is then removed from the population.

Table 1 introduces our multiobjective infill (MOI-)MBO strategies. Note, that the two last approaches simply implement a multiobjectivization of the

Algorithm 2. Evolutionary optimization of multiobjective infill criteria

1 Generate an initial population $\mathcal{P} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu\} \subset \mathcal{X}$;
2 Evaluate $\mathcal{P}$;
3 **while** *number of iterations is not exceeded* **do**
4 Sample two individuals (parents): $\boldsymbol{x}_{p1} \in \mathcal{P}$ and $\boldsymbol{x}_{p2} \in \mathcal{P}$;
5 Generate a new individual (child): $\boldsymbol{x}_{ch} = \mathit{crossover}(\boldsymbol{x}_{p1}, \boldsymbol{x}_{p2}, \mathcal{X}, \eta_c, p_c)$;
6 Mutate the new individual: $\boldsymbol{x}_{ch} := \mathit{mutate}(\boldsymbol{x}_{ch}, \mathcal{X}, \eta_m, p_m)$;
7 Evaluate selected infill criteria (except distance) for $\boldsymbol{x}_{ch}$;
8 Update the current population: $\mathcal{P} := \mathcal{P} \cup \{\boldsymbol{x}_{ch}\}$;
9 **for** $\boldsymbol{x} \in \mathcal{P}$ **do**
10 Calculate $\mathit{dist}(\boldsymbol{x}, \mathcal{P})$ and append to objective values;
11 Compute non-dominated fronts $\mathcal{F}_1, \ldots, \mathcal{F}_k$ of $\mathcal{P}$;
12 Sort $\mathcal{F}_k$ by a selection criterion;
13 $\boldsymbol{x}_{worst}$ = last element of $\mathcal{F}_k$;
14 Update the current population: $\mathcal{P} := \mathcal{P} \setminus \{\boldsymbol{x}_{worst}\}$;
15 **return** $\mathcal{P}$;

LCB criterion and hence do not use any distance measure. In this case, lines 9–10 of Algorithm 2 should be ignored.

4 Comparison Experiments

In order to study the performances of all variants of our proposed MOI-MBO and the existing strategies for parallel multi-point infill, we conduct an extensive benchmark study. We also compare against the usual EGO 1-step algorithm – which of course works in a purely sequential, non-parallel fashion and is therefore able to exploit more information during the optimization. Finally, we also include a simple random search as a baseline comparison method.

Problem Instances and Budget. As problem instances for the benchmark, we selected all 24 test functions of the black-box optimization benchmark (BBOB) noise-free test suite [19]. It covers simple unimodal, ill-conditioned and multimodal functions. Some of the landscapes of these functions exhibit a strong global structure which a model-based optimizer can potentially exploit, while other functions do not have this characteristic. We study these functions in the dimensions $d \in \{5, 10\}$. For every function (and each dimension) we create 10 initial designs of size $5 \cdot d$. We then run each candidate optimizer 10 times, using the mentioned designs. Hence, this results in 10 statistical replications for each problem instance with fair and equal starting conditions for all competing optimizers. The optimizers are given an additional budget of $40 \cdot d$ function evaluations on top of the initial design. Parallel optimizers always propose batches of size $q = 5$ in our experiments, resulting in $8 \cdot d$ sequential iterations for them. As a quality measure, we choose the difference between the function values of the best obtained point during the optimization and the known global minimum.

Table 1. Proposed MOI-MBO approaches

Single Infill Criterion					Sel. Criterion		Abbreviation
mean	se	ei	dist		first	hv	
			nn	nb			
×	×		×		×		moi_mean.se.dist_nn_first
×	×			×	×		moi_mean.se.dist_nb_first
×	×		×			×	moi_mean.se.dist_nn_hv
×	×			×		×	moi_mean.se.dist_nb_hv
		×	×		×		moi_ei.dist_nn_first
		×		×	×		moi_ei.dist_nb_first
		×	×			×	moi_ei.dist_nn_hv
		×		×		×	moi_ei.dist_nb_hv
×	×				×		moi_mean.se_first
×	×					×	moi_mean.se_hv

EGO and Constant Liar Variants. To also study the effect of infill optimization methods on the final performance outcome – in EGO this is usually a sophisticated combination of a gradient-based and an evolutionary/restart mechanism, while in multiobjective optimization often a much simpler EA is used –, we reimplemented EGO with a simple $(\mu+1)$ evolutionary algorithm with simulated binary crossover and polynomial mutation as variation operators for optimization of the infill criterion. Here, we set $\mu = 20$. We also include an EGO variant where we use the mean value $\hat{f}(\boldsymbol{x})$ as a simple infill criterion.

Software. All of our experiments are conducted in the statistical programming language R. The BBOB test functions are made available in the R package `soobench` [20]. We have implemented all our code regarding model-based optimization (including the below mentioned evolutionary variants of 1-step EGO and constant liar, all multicriteria methods and parallel LCB) in the experimental R package `mlrMBO`[1]. The toolbox allows a generic combination of regression models and optimization strategies and builds upon the `mlr` R package for machine learning from Bischl [21]. The kriging models are fitted via the `DiceKriging` package and we compare against the EGO and constant liar implementations of the popular `DiceOptim` package implementation. Both Dice packages are published by Ginsbourger et al. [9].

Summary. In order to provide a succinct overview, we again list all compared approaches in Table 2.

[1] See https://github.com/berndbischl/mlrMBO

Table 2. Overview of compared model-based optimization strategies.

Method	Abbreviation	R Package	Infill optimizer
EGO	`ego`	`DiceOptim`	gradient-based `rgenoud`
EGO	`ego_ea_ei`	`mlrMBO`	simple EA
EGO Infill $\hat{f}(\boldsymbol{x})$	`ego_ea_mean`	`mlrMBO`	simple EA
Constant liar	`par_cl`	`DiceOptim`	gradient-based `rgenoud`
Constant liar	`par_cl_ea`	`mlrMBO`	simple EA
Parallel LCB	`par_lcb`	`mlrMBO`	simple EA per LCB function
Multiobj. Infill (see Table 1)	`moi`	`mlrMBO`	multicrit EA
Random search	`random_search`	–	–

5 Results

As the test functions differ w.r.t. their respective characteristics, we divide the BBOB function into the following three sets, using the same numbering as in [19]:

set1 Unimodal functions: 1, 2, 5–14,
set2 Multimodal functions with adequate global structure: 3, 4, 15–19,
set3 Multimodal functions with weak global structure: 20–24.

To interpret the results, we first rank the considered optimization approaches among each other for each dimension / test function / replication by their performance and subsequently calculate the mean rank for each approach across test functions and replications (per dimension). Thus, the lower the mean rank, the better the approach. We also conduct statistical sign tests to analyze whether differences in performance between candidate algorithms are significant. Here, we compute for each pair of optimization approaches the performance differences over all replications for a set of test functions (per dimension). The sign distribution of this difference vector is then used to decide whether one approach significantly outperforms the other, which is in essence a binomial test, see [22]. Each test is conducted at the 5 significance level without further adjustment for multiple testing, as our aim is to use the test as an exploratory tool to provide a descriptive visualization of the stochastic results.

We illustrate the results using preference relation graphs, containing two kinds of information: the mean ranks of the optimization approaches on the one hand and the decisions of pairwise sign tests on the other hand. The results are presented in Fig. 1, 2, 3 and 4. Each colored node represents an optimization approach, where its mean rank is given in braces. The main goal is to compare the different MOI-MBO strategies (yellow nodes) with other parallel approaches (red nodes). Of these, our most relevant competitor is LCB, as it scales a lot better with increasing sizes of q than constant liar. For comparison, also non-parallel approaches are considered (green nodes). But these purely sequential optimizers are expected to perform better than the parallel ones, as they benefit from a

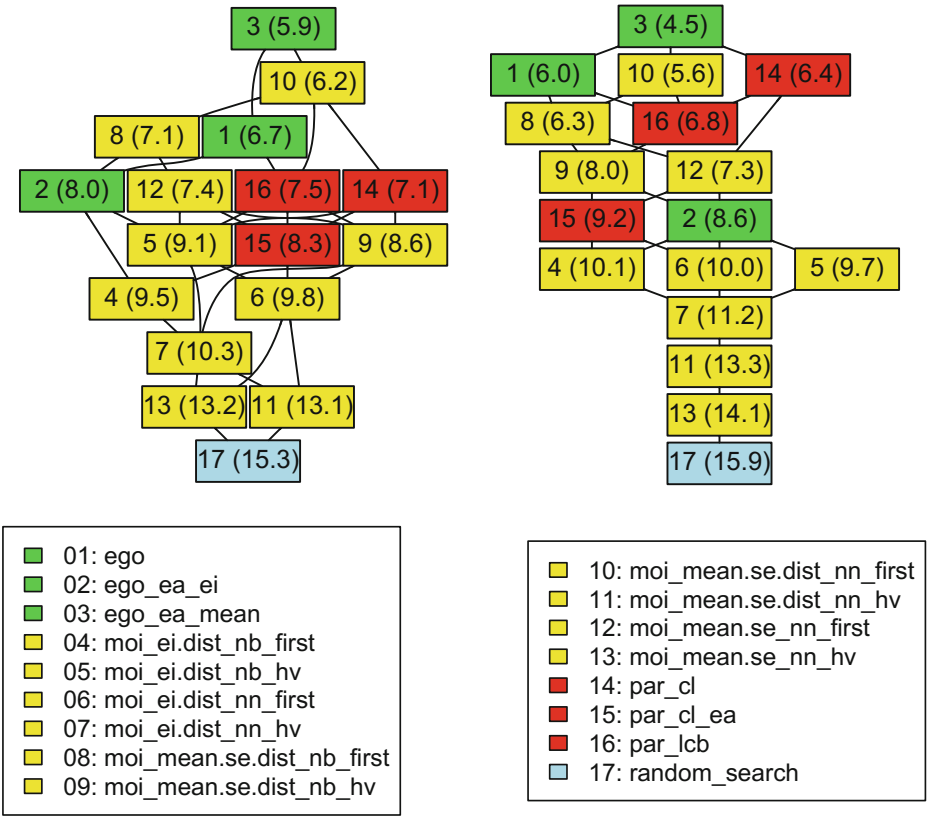

Fig. 1. Comparison over all test functions ($d = 5$ and $d = 10$) (Color figure online).

higher number of surrogate model fits and can incorporate more information into the models. Two nodes are connected with an edge if one approach (the upper one) is significantly better than the other (the lower one). Note that it is possible that one approach is significantly better than another one, although it has a (slightly) worse mean rank.

Figure 1 illustrates the results regarding all test functions. Overall, the best strategy is `ego_ea_mean`, even outperforming `ego` and `ego_ea_ei`. This is somehow surprising, particularly as the same holds true for the set of multimodal functions (Figs. 3 and 4), where exploration of the parameter space is expected to be more beneficial than just exploitation of the surrogate function.

Over all test functions, the best parallel approach seems to be `moi_mean.se.dist_nn_first`, which has the best mean rank and – with one exception – outperforms all other parallel strategies according to the sign test. Only slightly worse perform the strategies `moi_mean.se.dist_nb_first` and `moi_mean.se_first`, which outperform all other considered MOI-MBOs. Since all three methods just differ in the applied distance criterion, this seems to only have a relatively weak influence. In

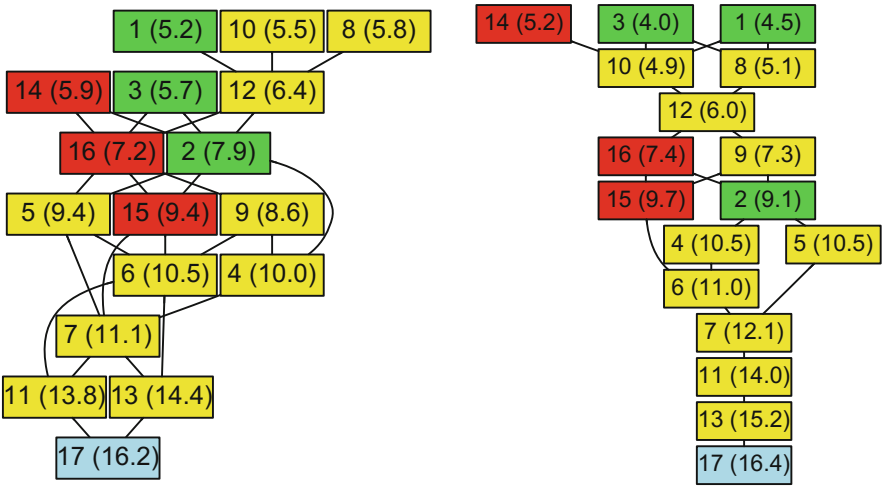

Fig. 2. Comparison over test functions of set1 ($d = 5$ and $d = 10$) (Color figure online).

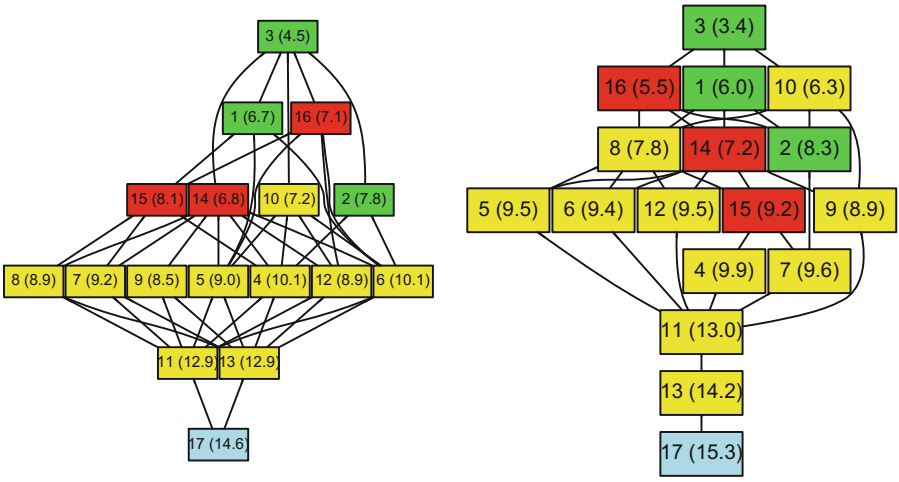

Fig. 3. Comparison over test functions of set2 ($d = 5$ and $d = 10$) (Color figure online).

contrast, the graphs indicate significant improvements by using the selection criterion `first` – which focuses more on the first criterion `mean` or `ei` – instead of `hv`. Regarding our experiments, the best state-of-the-art parallel MBO is `par_cl`, although this holds in a strong sense (significance of the sign test when we compare it to `par_lcb`) only for $d = 10$. While `par_cl` performs approximately as good as the best MOI-MBO `moi_mean.se.dist_nn_first` for the unimodal and the multimodal functions with adequate global structure (Figs. 2 and 3), our new approach

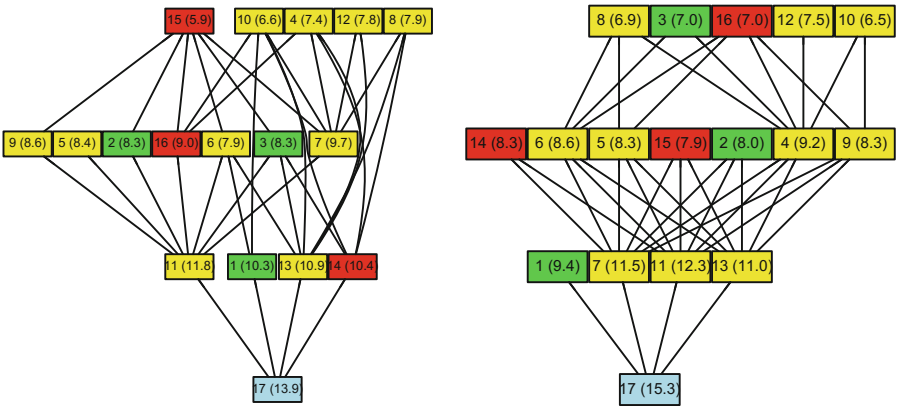

Fig. 4. Comparison over test functions of set3 ($d = 5$ and $d = 10$) (Color figure online).

performs better on multimodal functions with weak global structure (Fig. 4). Contrary, `par_cl_ea` and `par_lcb` – the other two considered state-of-the-art parallel MBOs – perform quite weakly on unimodal functions (Fig. 2).

6 Conclusion

In this paper, a multiobjective approach for parallel model-based optimization was introduced. Therefore, ten different strategies – each relying on a reasonable subset of five infill criteria – were compared with several state-of-the-art approaches including EGO, parallel LCB and constant liar. While three of the infill criteria have been applied before, the concept of multiobjectivization and the consideration of the distance to neighboring points is a new approach in the context of MBO. Regarding the 24 considered test functions in five and ten dimensions, a MOI-MBO strategy, which relies on the mean model prediction $\hat{f}(\boldsymbol{x})$, the model uncertainty $\hat{s}(\boldsymbol{x})$ and the distance to the nearest neighbor as infill criteria, performs best on average. As shown in the previous section it even outperforms existing parallel methods in many situations. Additionally, its runtime behavior is not significantly inhibited if the number of cores q is increased, as this only results in a larger population size of the EA. Furthermore, the experiment shows a bias in favor of more exploitative methods versus more explorative ones, although this might be an artefact of the considered benchmark set.

In future comparison studies, also the recent approach of Chevalier and Ginsbourger [11] should be considered once their code is released. While in the experiments above all approaches are applied on only five CPU cores ensuring acceptable run times even for the most complex ones, in a next step the influence of the number of cores regarding the performance should be analyzed. Furthermore, we also would like to investigate the applicability of MOI-MBO in the noisy case.

Acknowledgements. This paper is based on investigations of the projects B3 and C2 of the Collaborative Research Center SFB 823, which are kindly supported by Deutsche Forschungsgemeinschaft (DFG). It is also partly supported by the French national research agency (ANR) within the Modeles Numeriques project NumBBO. The authors also thank Tobias Wagner for fruitful discussions of multiobjective infill criteria.

References

1. Koch, P., Bischl, B., Flasch, O., Bartz-Beielstein, T., Weihs, C., Konen, W.: Tuning and evolution of support vector kernels. Evol. Intel. **5**(3), 153–170 (2012)
2. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) LION 5. LNCS, vol. 6683, pp. 507–523. Springer, Heidelberg (2011)
3. Hess, S., Wagner, T., Bischl, B.: PROGRESS: progressive reinforcement-learning-based surrogate selection. In: Nicosia, G., Pardalos, P. (eds.) LION 7. LNCS, vol. 7997, pp. 110–124. Springer, Heidelberg (2013)
4. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Global Optim. **13**(4), 455–492 (1998)
5. Krige, D.: A statistical approach to some basic mine valuation problems on the witwatersrand. J. Chem. Metall. Min. Soc. S. Afr. **52**(6), 119–139 (1951)
6. Locatelli, M.: Bayesian algorithms for one-dimensional global optimization. J. Global Optim. **10**(1), 57–76 (1997)
7. Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. J. Global Optim. **21**(4), 345–383 (2001)
8. Vazquez, E., Bect, J.: Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. J. Stat. Plann. Infer. **140**(11), 3088–3095 (2010)
9. Ginsbourger, D., Le Riche, R., Carraro, L.: Kriging is well-suited to parallelize optimization. In: Tenne, Y., Goh, C.-K. (eds.) Computational Intel. in Expensive Opti. Prob. ALO, vol. 2, pp. 131–162. Springer, Heidelberg (2010)
10. Janusevskis, J., Le Riche, R., Ginsbourger, D., Girdziusas, R.: Expected improvements for the asynchronous parallel global optimization of expensive functions: potentials and challenges. In: Hamadi, Y., Schoenauer, M. (eds.) LION 6. LNCS, vol. 7219, pp. 413–418. Springer, Heidelberg (2012)
11. Chevalier, C., Ginsbourger, D.: Fast computation of the multi-points expected improvement with applications in batch selection. In: Nicosia, G., Pardalos, P. (eds.) LION 7. LNCS, vol. 7997, pp. 59–69. Springer, Heidelberg (2013)
12. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Parallel algorithm configuration. In: Hamadi, Y., Schoenauer, M. (eds.) LION 6. LNCS, vol. 7219, pp. 55–70. Springer, Heidelberg (2012)
13. Coello, C.A.C., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007)
14. Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., Zitzler, E.: Do additional objectives make a problem harder? In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07, pp. 765–772. ACM (2007)
15. Tran, T.D., Brockhoff, D., Derbel, B.: Multiobjectivization with NSGA-II on the Noiseless BBOB Testbed. In: GECCO (Companion), Workshop on Black-Box Optimization Benchmarking (BBOB'2013), Amsterdam, Pays-Bas, July 2013

16. Ulrich, T., Thiele, L.: Maximizing population diversity in single-objective optimization. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11, pp. 641–648. ACM, New York (2011)
17. Wessing, S., Preuss, M., Rudolph, G.: Niching by multiobjectivization with neighbor information: Trade-offs and benefits. In: IEEE Congress on Evolutionary Computation (CEC), pp. 103–110 (2013)
18. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. Eur. J. Oper. Res. **181**(3), 1653–1669 (2007)
19. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: noiseless functions definitions. Technical report RR-6829, INRIA (2009). http://hal.inria.fr/inria-00362633/en
20. Mersmann, O., Bischl, B., Bossek, J., Judt, L.: soobench: Single Objective Optimization Benchmark Functions. R package version 1.1-164
21. Bischl, B.: mlr: Machine Learning in R. R package version 1.2
22. Conover, W.: Practical Nonparametric Statistics, 2nd edn. Wiley, New York (1980)