Surrogate Strategies for Scalarisation-based Multi-objective Bayesian Optimizers

Qingyu Mo, João A. Duro, and Robin C. Purshouse

University of Sheffield, Sheffield, UK
`{qmo2,j.a.duro,r.purshouse}@sheffield.ac.uk`

Abstract. Scalarisation-based approaches to multi-objective Bayesian optimization, such as the seminal ParEGO algorithm, may be either single-surrogate or multi-surrogate. In the former case, a single surrogate model is built of the scalarised function; in the latter case, separate surrogates are built for each objective function. A recent study argued that the multi-surrogate approach should be preferred and presented empirical findings supportive of this case. However, these findings were based on an outdated approach to benchmarking algorithm performance and were limited to low-dimensional problems. In this study, we use the modern COCO benchmarking framework to analyse the performance of single-surrogate and multi-surrogate ParEGO algorithms and compare these to random sampling, Sobol space-filling, and the high-performing optimizer known as TPB. Our findings broadly support the original findings for low-dimensional problems, but we find that multi-surrogate ParEGO performs comparatively poorly in higher dimensions. TPB tends to outperform both ParEGOs, suggesting that initial budget investment in ideal and nadir point identification is a favourable strategy.

Keywords: Bayesian Optimization · Multi-objective Optimization · Benchmark Problems · Surrogate Modelling.

1 Introduction

Multi-objective optimization deals with problems with multiple performance objectives, where there is usually no single solution that satisfies all objectives simultaneously [13]. Without loss of generality, a multi-objective optimization problem (MOP) can be defined as follows:

$$\min_{\mathbf{x}} \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})) \text{ subject to } \mathbf{x} \in \mathcal{S}, \tag{1}$$

where a solution $\mathbf{x}$ is a vector of n decision variables defined in the domain $\mathcal{S} \subseteq \mathbb{R}^n$. $\mathcal{S}$ is known as the feasible region or decision space, the mapping $\mathbf{F} : \mathcal{S} \to \mathbf{Z}$ consists of $m \geq 2$ objective functions and $\mathbf{Z} \subset \mathbb{R}^m$ is the objective space. Let $\mathbf{a}, \mathbf{b} \in \mathbf{Z}$, then $\mathbf{a}$ is said to *Pareto dominate* $\mathbf{b}$ (denoted by $\mathbf{a} \prec \mathbf{b}$) if and only if $\mathbf{a}$ is not worse than $\mathbf{b}$ in all objectives (i.e. $a_i \leq b_i$ for every $i \in \{1, \ldots, m\}$) and $\mathbf{a}$ is better than $\mathbf{b}$ is at least one objective (i.e. $\exists j \in \{1, \ldots, m\}$ such that $a_j < b_j$). A solution $\mathbf{x}^* \in \mathcal{S}$ is said to be Pareto-optimal to (1) in case there is no other $\mathbf{x} \in \mathcal{S}$ such that $\mathbf{x} \prec \mathbf{x}^*$. The set that contains all Pareto-optimal solutions is called the *Pareto set* (PS) and its image in the objective space is called the *Pareto front* (PF).

The focus here is on closed-box MOPs with expensive-to-evaluate objective functions. Many real-world problems fall into this category, particularly in engineering design, where there is increasing reliance on computationally expensive high-fidelity simulation models [9]. In practice it may be only possible to afford a limited number of function evaluations (say 1000 or fewer), potentially due to the time it takes to conduct a single evaluation, licensing limitations, or costs.

Multi-objective *Bayesian optimization algorithms* (BOAs) a popular choice for solving these type of problems, since they are capable of providing an approximate set of Pareto-optimal solutions using a limited number of function evaluations. BOAs replace the expensive function with a cheap-to-evaluate surrogate model. *Gaussian process* (GP) models [14] are often used since they provide not only the predictions but also the corresponding uncertainty. The latter is important because the search over a surrogate model does not simply imply finding the location that improves the estimated fitness, since the model's accuracy needs to be taken into account. Acquisition functions (e.g. *Expected Improvement* (EI)) offer a criterion that promotes a balance between exploration and exploitation, where exploration is associated with areas in the search space with high uncertainty, whereas exploitation corresponds to areas with better fitness.

Various acquisition functions have been proposed and adapted for MOPs. One approach is to use scalarisation functions that transform a MOP into a series of single-objective optimization problems that are solved simultaneously. Building a single surrogate of the scalarised objective vectors was popularised by Knowles' ParEGO [11] algorithm, and is known here as the *single-surrogate* approach. However, Chugh [5] recently questioned if it would be more advantageous to instead build a surrogate model of each individual objective function and afterwards apply scalarisation to the objective vectors obtained by the surrogate models, which is known as the *multi-surrogate* approach. One clear disadvantage of the latter approach is the need to build multiple surrogate models as opposed to just one, which increases the computational complexity. However, Chugh argued that in the single-surrogate approach the fitness landscape of the scalarisation function and the objective functions can differ, and that this issue is not experienced by the multi-surrogate approach. The scalarising function built by the multi-surrogate approach is not Gaussian, meaning a closed-form expression for EI is not available. Instead, Chugh approximated the scalarising function using a Gumbel distribution, with EI estimated by Monte Carlo sampling of this distribution. Chugh's experimental results confirmed that the multi-surrogate approach was more effective than the original original single-surrogate method. However, these findings were limited to three DTLZ problems and one real-world problem, and considered no more than five decision variables.

In the present paper, we seek to validate Chugh's findings by testing single and multi-surrogate scalarising strategies using a best practice approach for contemporary MOP benchmarking:

1. We consider all 55 bi-objective problems from the BBOB test suite that have been designed to address issues with older problems like DTLZ [1].
2. We adopt the COCO benchmarking framework that can be customized for comparing optimization algorithms on a budget, with empirical cumulative distribution functions (ECDFs) providing a straightforward visual approach for comparing optimization algorithms [10].
3. We do a comparison with a high-performing alternative algorithm for expensive problems, TPB, guided by existing COCO benchmarking findings [16].

The rest of the paper is structured as follows. The single-surrogate and multi-surrogate approaches are described in Section 2. Details about the COCO performance assessment, the bi-objective BBOB problems, and experimental setup are in Section 3. The experimental results along with a discussion are in Section 4, and the paper concludes with Section 5.

2 Multi-objective Bayesian optimization: Single-surrogate and Multi-surrogate Approaches with Scalarisation

This section starts by describing the general procedure followed by a generic multi-objective Bayesian optimization algorithm that relies on the scalarisation technique. Following this, the differences between the single-surrogate and multi-surrogate approaches will be described in more detail.

The first step is to generate a space-filling sampling plan $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of N solutions, and to use the expensive-to-evaluate objective functions to generate the initial data $\{(\mathbf{x}_i, \mathbf{z}_i = \mathbf{F}(\mathbf{x}_i))\}_{i=1}^{N}$. One approach to generate sampling plans is to use *Sobol sequences* [15]. The second step is to generate a set of well distributed weight vectors, where each vector targets the PF from a different direction. One approach to generate weight vectors is to use a *simplex design*.

In an iterative approach starting from the first weight vector, the next step is to use statistical surrogate model(s) in order to estimate where best to sample the next solution. GP surrogate models are a popular choice here due to their uncertainty estimation capabilities and the EI is an acquisition function that uses the GP properties to create a balance between exploration and exploitation. The next solution to be sampled is found by maximising the EI function, and is then added to $\mathcal{X}$. Following the evaluation of the new solution, this process repeats iteratively where the next weight vector is chosen, the surrogate model(s) are updated, and a new solution is found again by conducting a search over the EI. More details about the scalarisation technique, including how the weight vectors are integrated, are provided below.

2.1 Single-surrogate Approach

Assuming initial data, the next step is to calculate a scalar fitness value for each solution by using a scalarisation function, and this provides the data

$$\{(\mathbf{x}_i, s(\mathbf{z}_i, \mathbf{w}, \boldsymbol{\xi}))\}_{i=1}^{N} \tag{2}$$

that is to be used for constructing a single surrogate model. A generic scalarisation function $s(\mathbf{z}, \mathbf{w}, \boldsymbol{\xi})$ maps an objective vector $\mathbf{z}$ corresponding to solution $\mathbf{x}$ with respect to some weight vector $\mathbf{w} = (w_1, \ldots, w_m)^T$, and the vector $\boldsymbol{\xi}$ contains parameters that are used to customize the given scalarisation function. An example is the weighted Chebyshev augmented function given by:

$$s(\mathbf{x}) = \max_{1 \leq i \leq m} (w_i z_i(\mathbf{x})) + \rho \sum_{i=1}^{m} w_i z_i(\mathbf{x}), \tag{3}$$

where ρ is a small positive number set to 0.05 as suggested by [11].

A surrogate model is constructed with training data from (2), and both the prediction and uncertainty of the model are considered together in a normally distributed random variable $S(\mathbf{x}) \sim \mathcal{N}(\hat{s}(\mathbf{x}), \hat{\varepsilon}_s^2(\mathbf{x}))$, where $\hat{s}(\mathbf{x})$ and $\hat{\varepsilon}_s^2(\mathbf{x})$ are respectively the prediction mean and prediction variance. The EI criterion is related to the notion of *improvement*, and assuming minimisation, the improvement at $\mathbf{x}$ is given by $I(\mathbf{x}) = \max(s(\mathbf{x}^+) - S(\mathbf{x}), 0)$, where $\mathbf{x}^+$ is the location of the current best known scalar fitness value from amongst the evaluated solutions, that is, $\mathbf{x}^+ = \arg\min_{\mathbf{x} \in \mathcal{X}} \; s(\mathbf{x})$. The EI is obtained by taking the expected value of this improvement as given by:

$$E[I(\mathbf{x})] = \begin{cases} (s(\mathbf{x}^+) - \hat{s}(\mathbf{x}))\Phi\left(\frac{s(\mathbf{x}^+) - \hat{s}(\mathbf{x})}{\hat{\varepsilon}_s(\mathbf{x})}\right) + \hat{\varepsilon}_s(\mathbf{x})\phi\left(\frac{s(\mathbf{x}^+) - \hat{s}(\mathbf{x})}{\hat{\varepsilon}_s(\mathbf{x})}\right) & \text{if } \hat{\varepsilon}_s(\mathbf{x}) > 0 \\ 0 & \text{if } \hat{\varepsilon}_s(\mathbf{x}) = 0 \end{cases} \tag{4}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ are the cumulative and probability density functions of the standard normal distribution, respectively.

2.2 Multi-surrogate Approach

The multi-surrogate approach constructs a separate surrogate model for each objective function, meaning that m surrogate models are created from the data

$$\{(\mathbf{x}_i, f_1(\mathbf{x}_i))\}_{i=1}^{N}, \{(\mathbf{x}_i, f_2(\mathbf{x}_i))\}_{i=1}^{N}, \ldots, \{(\mathbf{x}_i, f_m(\mathbf{x}_i))\}_{i=1}^{N} \tag{5}$$

Given that there is no analytical closed-form expression to determine the EI in the multi-surrogate approach, an alternative is to use Monte Carlo integration by producing N_C samples from the surrogates [8, 6]. Consider $\mathbf{x}$ to be a solution that we wish to estimate the expected improvement, let $q_{ij}(\mathbf{x})$ be the jth sample for the ith objective from the normal distribution with predictive mean $\hat{f}_i(\mathbf{x})$ and predictive variance $\hat{\varepsilon}_{f_i}^2(\mathbf{x})$, where $i = 1, \ldots, m$ and $j = 1, \ldots, N_C$. These samples are then scalarised, and we show an example with the weighted Chebyshev augmented function in (6), but note that any other scalarisation function could have been used instead.

$$s_j(\mathbf{x}) = \max_{1 \leq i \leq m} (w_i q_{ij}(\mathbf{x})) + \rho \sum_{i=1}^{m} w_i q_{ij}(\mathbf{x}) \tag{6}$$

The improvement at $\mathbf{x}$ w.r.t. the jth sample is $I_j(\mathbf{x}) = \max(s(\mathbf{x}^+) - s_j(\mathbf{x}), 0)$, and an approximation to the expected improvement across all samples is

$$E[I(\mathbf{x})] \approx \frac{1}{N_C} \sum_{j=1}^{N_C} I_j(\mathbf{x}). \tag{7}$$

3 COCO Performance Evaluation and Experimental Setup

3.1 Performance Assessment in COCO

We use the indicator I_{HV}^{COCO} provided by the the COCO library, which is to be minimised [3]. This indicator is applied to an archive of non-dominated solutions, and takes the negative hypervolume of the (normalised) objective-vectors of the solutions that are inside the region of interest, defined by the ideal ($\mathbf{z}_{\text{ideal}}$) and nadir ($\mathbf{z}_{\text{nadir}}$) vectors, that is, $[\mathbf{z}_{\text{ideal}}, \mathbf{z}_{\text{nadir}}]$. In case there are no solutions that dominate $\mathbf{z}_{\text{nadir}}$, then it calculates the (normalised) Euclidean distance of the closest solution to the region of interest.

During the optimization run of each algorithm we record the number of function evaluations ("runtime") to reach certain I_{HV}^{COCO} indicator values, known as targets. Two anchoring targets are chosen, corresponding to the best performance achieved across all the optimization algorithms involved in the experiment for two budgets: $50 \times n$ (yielding the hardest target) and $0.5 \times n$ (yielding the easiest target). An intermediate 29 further targets are then defined equidistantly on a log-scale between the two anchors. The runtimes to these targets are displayed in the form of empirical cumulative distribution functions [10, 1].

3.2 The Bi-objective BBOB Problems and Algorithm Settings

The performance of the multi-objective optimization algorithms is shown on all 55 bi-objective BBOB problems that are found in the *bbob-biobj* test suite, part of the COCO library [1], named $F_1 \dots F_{55}$. The first and second objective functions from these problems are different combinations of the 24 single-objective noiseless BBOB functions. These problems are scalable in the number of decision variables, and we consider n to be: 2, 3, 5, 10 and 20. The term *instance* is used refer to a given parametrisation of a problem-dimension pair, and we consider the first 15 instances since the COCO library provides hypervolume reference values for these cases. Each optimization algorithm conducts a single run per instance, and a different random seed is used for each run. The search space is defined by the hyperbox $[-100, 100]^n$ for all problems.

The single-surrogate and multi-surrogate approaches are referred to as ParEGO and MParEGO, respectively. For these algorithms the size of the sampling plan is set to $N = 11n - 1$ and the number of scalarising vectors is set to 11, as suggested in [11]. Based on our own set of experiments the number of solutions used to train the surrogate model is set to 100 and, specifically for MParEGO, the number of samples to conduct the Monte Carlo integration is $N_C = 200$. The surrogate model adopted by ParEGO and MParEGO follows the Kriging modelling approach from [9], and uses a squared exponential kernel, where the length scales are treated as hyperparameters to be learned. We use the single-objective evolutionary algorithm ACROMUSE [12] to learn the kernel hyperparameters and search the surrogate model.

Other algorithms considered for comparison are TPB [16], Sobol [15], and uniform random search (known here as Random). TPB is a high-performing optimizer for expensive problems according to previous COCO findings. Sobol is an anytime space-filling DoE technique that relies on quasi-random low-discrepancy sequences. The budget for TPB is set to $50 \times n$, and given that this is not an anytime algorithm its performance can only be compared with the other optimization algorithm at this particular point. This is also inline with how the ECDF targets are chosen, that is, the hardest target is chosen at $50 \times n$. For the other algorithms the budget is set to 1000 function evaluations, and their performance can be compared at any point during the optimization run since they are all anytime algorithms. In the description of the TPB algorithm in [16], it is mentioned that the first solution to be evaluated is the search space origin, and this follows the suggestion by the COCO library authors in [2], where the search space origin is a specially good search point, and should be evaluated first in order to avoid disfavouring any algorithm that does not evaluate this solution. However, we believe that by first evaluating this solution it has a strong bias on the performance of the algorithms since in some cases the solution set is already very close to the region of interest. Therefore, for TPB we replace this solution by a random location inside the hyperbox $[-100, 100]^n$. To conduct the optimization runs, we use a Python implementation of TPB provided by its authors,[1] while the other algorithms are our Tigon C++ implementation [7].

4 Experimental Results

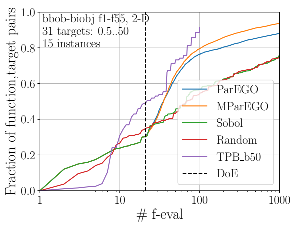
4.1 Single-surrogate versus Multi-surrogate

This section presents the experimental results for the single- and multi-surrogate approaches, namely ParEGO and MParEGO, respectively, when applied to the optimization problems from the bbob-biobj test suite.

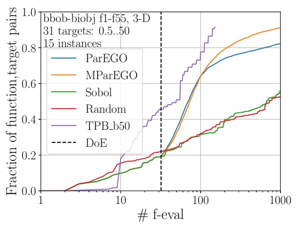
Fig. 1 shows the ECDF graphs aggregated over all functions, for a maximum budget of 1000 functions evaluations. The vertical dashed line is shown at N function evaluations, indicating the last evaluation of the initial sampling plan. Since ParEGO and MParEGO use the same DoE technique (i.e. Sobol) to generate the sampling plan, they show equal performance for the first N evaluations, and this matches with the performance shown by Sobol. Notably, for lower dimensions (i.e. 2, 3 and 5) MParEGO outperforms ParEGO, but for higher dimensions (i.e. 10 and 20) ParEGO attains better performance than MParEGO. As the number of dimensions increases from 2 to 5, the performance gap between MParEGO and ParEGO widens, and the same can be said, but reversely, between 10 and 20 dimensions.

We now consider ECDF graphs aggregated over groups of functions defined by five subgroups: separable, moderate, ill-conditioned, multimodal weakly structured, and multimodal with global structure. Fig. 2 and 4 show a selection of different groups of functions in dimension 5 and 20, respectively. In problems

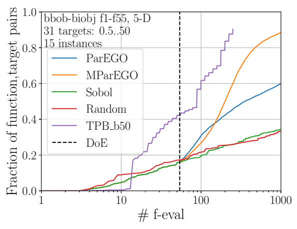
[1] https://github.com/ryojitanabe/tpb



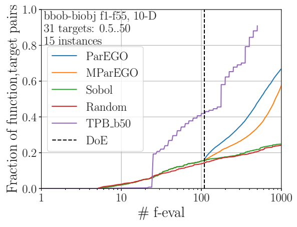
(a) 2-D



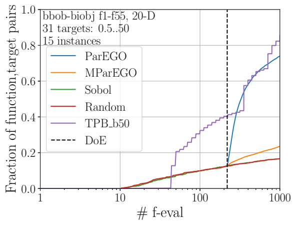
(b) 3-D



(c) 5-D

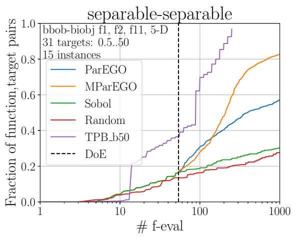


(d) 10-D

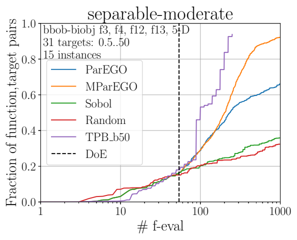


(e) 20-D

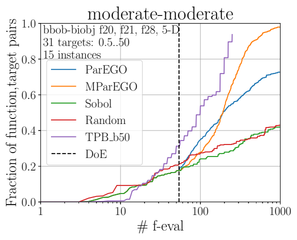
Fig. 1: Empirical runtime distributions aggregated over all bbob-biobj functions in dimensions 2, 3, 5, 10 and 20, comparing the algorithms ParEGO, MParEGO, Sobol, Random, and TPB.



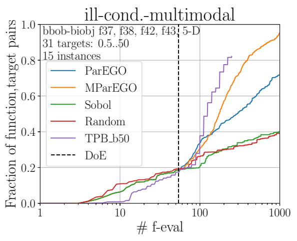
(a)



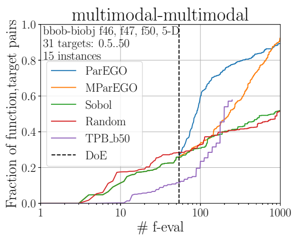
(b)



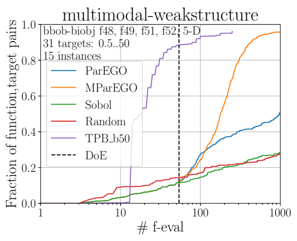
(c)



(d)



(e)



(f)

Fig. 2: Empirical runtime distributions per function group of the bbob-biobj suite for algorithms ParEGO, MParEGO, Sobol, Random, and TPB in dimension 5.

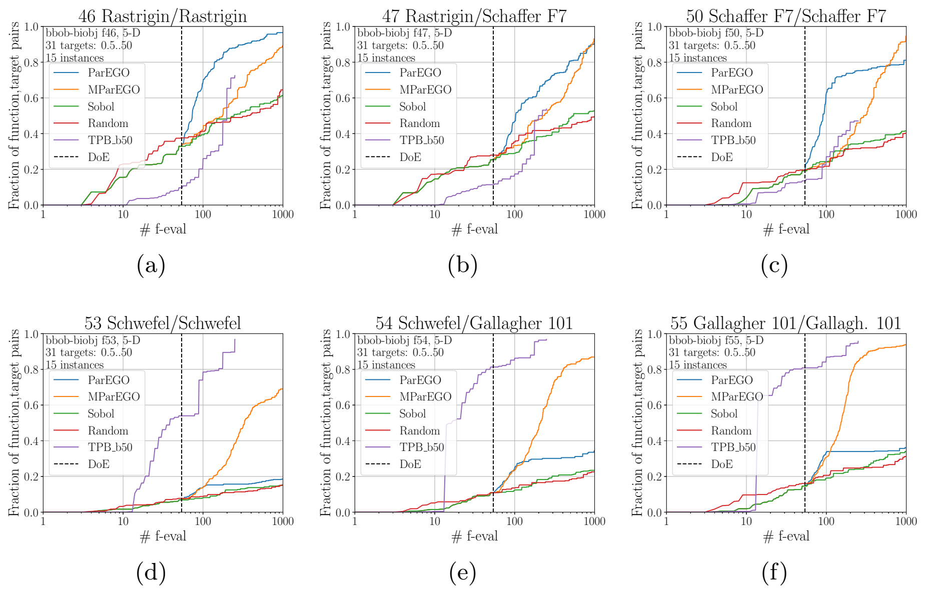
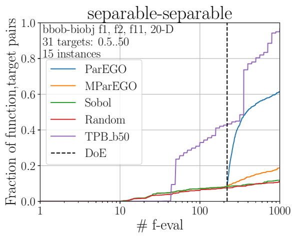


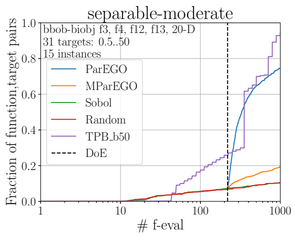
Fig. 3: Empirical runtime distributions for the algorithms ParEGO, MParEGO, Sobol, Random, and TPB on single functions from the bbob-biobj suite that form part of the multimodal with global structure group (first row), and multimodal weak global structure (second row) in dimension 5.

with 5 dimensions, MParEGO performs better than ParEGO for most problems, and the only exception is when the two objectives are from the multimodal with global structure group (multimodal-multimodal) (Fig. 2e). The ECDF graphs of the individual functions from this group are shown in Fig. 3 (first row), where ParEGO attains better performance for most function evaluations, and at 1000 functions evaluations the performance of ParEGO in comparison with MParEGO is: better for F_{46} (Fig. 3a), similar for F_{47} (Fig. 3b), and worse for F_{50} (Fig. 3c). Multi-modality is not necessarily an issue for MParEGO since as shown in the Fig. 3 (second row) all the problems from the multimodal with weak global structure group show MParEGO outperforming ParEGO. There are some cases where ParEGO does not perform much better than random search, e.g., problems F_{53}, and F_{55} as shown in Fig. 3d and 3f, but for these problems the performance of MParEGO is considerably better than random search.

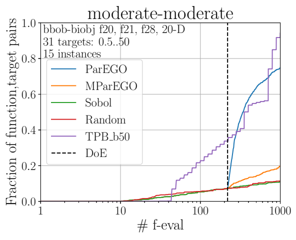
In problems with 20 dimensions (Fig. 4), ParEGO performs considerably better than MParEGO. In some cases, the performance of MParEGO is not much better (or even worse) than random search, as shown in Fig. 5. These problems have in common the Rastrigin function (f_{15}) which is know to cause discontinuities in both the PS and PF [1], and the other objective comes from other groups (except the multi-modal with weak global structure group).



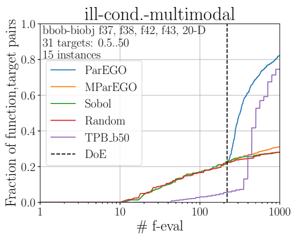
(a)



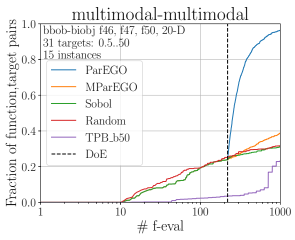
(b)



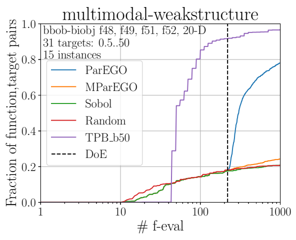
(c)



(d)



(e)



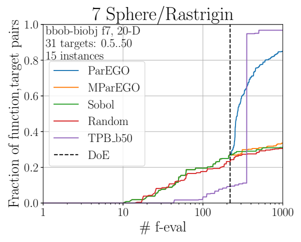
(f)

Fig. 4: Empirical runtime distributions per function group of the bbob-biobj suite for algorithms ParEGO, MParEGO, Sobol, Random, and TPB in dimension 20.

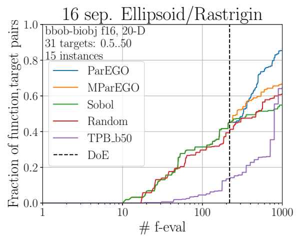
4.2 Comparison with a State-of-the-art Optimization Algorithm for Expensive Problems

In this section, a comparison is conducted with TPB. The TPB optimization runs are for a budget of $50 \times n$ function evaluations, and this implies that for $n \in \{2, 3, 5, 10, 20\}$ the number of functions evaluations are 100, 150, 250, 500, and 1000, respectively.

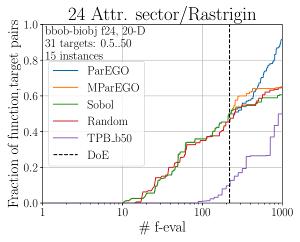
For the case where all functions are aggregated, and across all dimensions, TPB performs better than the other algorithms as shown in Fig. 1. For the different groups with $n = 5$ (Fig. 2), TPB attains better performance for most cases, and the only exception is for the multimodal with global structure group (Fig. 2e) where ParEGO attains better performance. For $n = 20$ as shown in Fig. 4, TPB does better than both ParEGO algorithms in 11 out of the 15 groups, and in the four cases that ParEGO does better, it is ParEGO (not MParEGO) that is able to outperform TPB. TPB performs particularly well on the separable group (Fig. 4a), and on the separable plus multimodality with weak structure group (Fig. 4b). However, TPB performance is worse than random search in the multimodal with global structure group (Fig. 4c), and in some individual cases where the multimodal with global structure is combined with other functions as in F_{24} (Fig. 5c). Considering the functions from the multimodal with global structure group in Fig. 6, the performance of TPB is worst than random search in F_{47} (Fig. 6b) and F_{50} (Fig. 6c), while being marginally better than both random search and MParEGO in F_{46} (Fig. 6a).



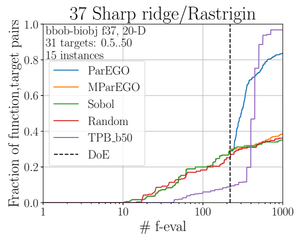
(a)



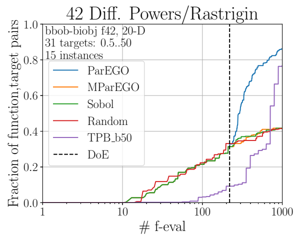
(b)



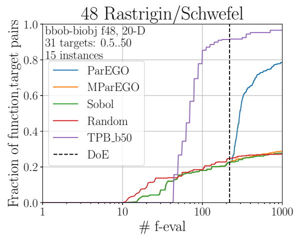
(c)



(d)

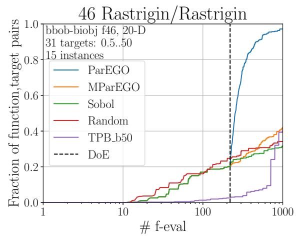


(e)

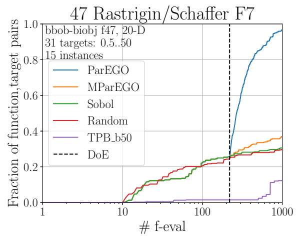


(f)

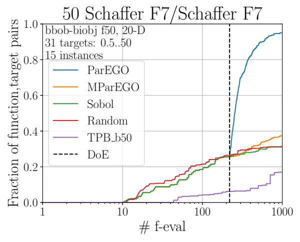
Fig. 5: Empirical runtime distributions for algorithms ParEGO, MParEGO, Sobol, Random, and TPB on single functions from the bbob-biobj suite that shows MParEGO not performing much better (or even worse) than random search in dimension 20.



(a)



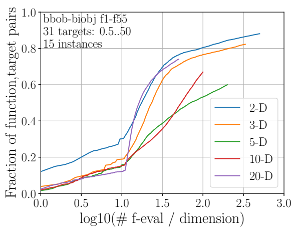
(b)



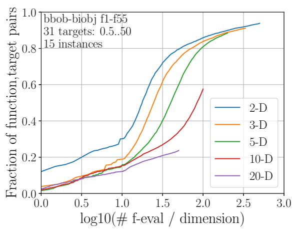
(c)

Fig. 6: Empirical runtime distributions for the algorithms ParEGO, MParEGO, Sobol, Random, and TPB on single functions from the bbob-biobj suite that form part of the multimodal with global structure group in dimension 20.

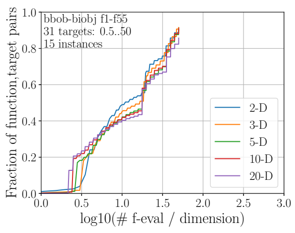
In terms of how the optimization algorithm's performance scale with the number of dimensions, Fig, 7 shows the ECDF aggregated over all bbob-biobj functions, and indicates that the performance of TPB is less affected by an increase in the number of dimensions when compared with ParEGO and MParEGO. Moreover, MParEGO is the most affected algorithm by the increase in the number of dimensions, and in comparison ParEGO seems to do particularly well



(a) ParEGO



(b) MParEGO



(c) TPB

Fig. 7: Empirical runtime distributions aggregated over all bbob-biobj functions showing how the algorithms ParEGO, MParEGO, and TPB scale with the number of dimensions.

on high-dimensional problems (i.e. $n = 20$). The results shown above are a subset of the total conducted runs. A supplementary file with all the experimental results, and also the source code to reproduce these results, is available in https://github.com/jaduro/ssmbo_emo2025.

4.3 Discussion

The single-surrogate and multi-surrogate approaches were previously compared in [5] by using DTLZ problems (2, 3 and 7) with the number of decision variables being 4 and 5, and the number of objectives being 2 and 3. The budget to run the optimization algorithms was set to $n \times 30$ in addition to the initial data set of $n \times 10$ solutions. The experimental results in [5] show that the multi-surrogate approach outperformed the single-surrogate approach on DTLZ2 and DTLZ3, and that the multi-surrogate approach did not perform well on DLTZ7, which is a problem with a disconnected PF. These findings are in agreement with the results presented here, where for low-dimensional problems (i.e. up to $n = 5$) MParEGO outperforms ParEGO in most cases, but MParEGO was outperformed by ParEGO in disconnected problems (e.g. the double Rastrigin function F_{46}, see Fig. 3a). In addition, our results suggest that the multi-surrogate approach does not scale well with the number of dimensions when compared with the single-surrogate approach. However, if the exact EI calculation for ParEGO is replaced by Monte Carlo estimation then ParEGO also suffers this failure mode (see supplementary materials for these experiments). This issue could be arising from an interaction between the quality of the estimated EI and the quality of the GP model in higher dimensions, and warrants further investigation.

A potential reason why TPB shows better performance relative to both ParEGO and MParEGO for the BBOB problems is its ability to find solutions inside the region defined by the nadir and ideal vectors. For this, TPB initially uses some of its budget to estimate the nadir and ideal vectors, by conducting single-objective optimization of each objective function. This is particular advantageous when using the performance indicator I_{COCO}, because a solution set will always have better performance when at least one solution is inside the region of interest $[\mathbf{z}_{\text{ideal}}, \mathbf{z}_{\text{nadir}}]$, as opposed to a solution set where no solution is able to dominate $\mathbf{z}_{\text{nadir}}$. Note that the I_{COCO} relies on the distance between the closest solution to the region of interest when there are no solutions that dominate $\mathbf{z}_{\text{nadir}}$, and only applies the hypervolume indicator to those solutions that are found inside the region of interest. There could be other reasons that give an advantage to TPB over ParEGO, for instance, the use of the Bézier simplex-based interpolation method might be a better approach when compared with the more classic simplex designs.

5 Conclusion

This paper has studied a fundamental question in scalarisation-based multi-objective Bayesian optimization, that is, whether it is better to rely on a single surrogate model or multiple surrogate models. For this, the performance of two algorithms (ParEGO and MParEGO) that are representative of each case have been applied to a wide range of bi-objective BBOB problems (55 in total), where each problem had the number of dimensions scaled up-to 20. The experimental results have revealed that on low-dimensional problems (up to 5), it is better to rely on multiple surrogate models (MParEGO), but as the number of dimensions increases up to 20, then an approach with a single surrogate model (ParEGO) performs better. In addition, we have also conducted a comparative analysis with a state-of-the-art optimization algorithm for expensive problems, called TPB. It has been revealed that TPB is capable of outperforming both ParEGO and MParEGO, but that ParEGO attain better performance on some problems with multimodality. Future work should study the scalability of these approaches with the number of objectives, compare with other optimization algorithms for expensive problem (in particular that are known perform well on problems with multimodality), consider other acquisition function besides EI (as in [4]), and apply these algorithms to real-world problems.

Acknowledgements

This paper was inspired by the Lorentz Center workshop *BeMCO: Benchmarking in Multi-Criteria Optimisation*, 15–19 April 2024. JAD and RP were supported by SIPHER (MR/S037578/1), a UK Prevention Research Partnership funded by the UK Research and Innovation Councils, the Department of Health and Social Care (England) and the UK devolved administrations, and leading health research charities https://ukprp.org/. The authors also acknowledge Dimo Brockhoff for his helpful advice in using COCO.

References

1. Brockhoff, D., Auger, A., Hansen, N., Tušar, T.: Using well-understood single-objective functions in multiobjective black-Box optimization test suites. Evolutionary Computation **30**(2), 165–193 (2022). https://doi.org/10.1162/evco_a_00298

2. Brockhoff, D., Hansen, N.: The impact of sample volume in random search on the bbob test suite. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 1912–1919. GECCO '19, ACM (2019). https://doi.org/10.1145/3319619.3326894
3. Brockhoff, D., Tušar, T., Tušar, D., Wagner, T., Hansen, N., Auger, A.: Biobjective performance assessment with the COCO platform (2016), https://arxiv.org/abs/1605.01746
4. Chugh, T.: Scalarizing functions in Bayesian multiobjective optimization. In: 2020 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8 (2020). https://doi.org/10.1109/CEC48606.2020.9185706
5. Chugh, T.: Mono-surrogate vs multi-surrogate in multi-objective Bayesian optimisation. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. GECCO '22, ACM (Jul 2022). https://doi.org/10.1145/3520304.3533972
6. Chugh, T., Evans, A.: Integrating Bayesian and evolutionary approaches for multi-objective optimisation. In: Smith, S., Correia, J., Cintrano, C. (eds.) Applications of Evolutionary Computation. pp. 391–406. Springer Nature Switzerland (2024)
7. Duro, J.A., Yan, Y., Giagkiozis, I., Giagkiozis, S., Salomon, S., Oara, D.C., Sriwastava, A.K., Morison, J., Freeman, C.M., Lygoe, R.J., Purshouse, R.C., Fleming, P.J.: Liger: A cross-platform open-source integrated optimization and decision-making environment. Applied Soft Computing **98**, 106851 (2020). https://doi.org/10.1016/j.asoc.2020.106851
8. Emmerich, M.T.M., Giannakoglou, K.C., Naujoks, B.: Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. IEEE Transactions on Evolutionary Computation **10**(4), 421–439 (2006). https://doi.org/10.1109/TEVC.2005.859463
9. Forrester, A.I., Keane, A.J.: Recent advances in surrogate-based optimization. Progress in Aerospace Sciences **45**, 50–79 (2009). https://doi.org/10.1016/j.paerosci.2008.11.001
10. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D.: COCO: A platform for comparing continuous optimizers in a black-box setting. Optimization Methods and Software **36**, 114–144 (2021). https://doi.org/10.1080/10556788.2020.1808977
11. Knowles, J.: ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation **10**(1), 50–66 (2005). https://doi.org/10.1109/TEVC.2005.851274
12. McGinley, B., Maher, J., O'Riordan, C., Morgan, F.: Maintaining healthy population diversity using adaptive crossover, mutation, and selection. IEEE Transactions on Evolutionary Computation **15**(5), 692–714 (2011). https://doi.org/10.1109/TEVC.2010.2046173
13. Miettinen, K.: Nonlinear Multiobjective Optimization, International Series in Operations Research & Management Science, vol. 12. Springer (1999)
14. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press (2006)
15. Sobol, I.M.: On the distribution of points in a cube and the approximate evaluation of integrals. U.S.S.R. Comput. Math. Math. Phys. **7**(4), 87–112 (1967)
16. Tanabe, R., Akimoto, Y., Kobayashi, K., Umeki, H., Shirakawa, S., Hamada, N.: A two-phase framework with a Bézier simplex-based interpolation method for computationally expensive multi-objective optimization. In: Proceedings of the Genetic

and Evolutionary Computation Conference. p. 601–610. GECCO ’22, ACM (2022). https://doi.org/10.1145/3512290.3528778