This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

Efficient multiobjective optimization employing Gaussian processes, spectral sampling and a genetic algorithm

Eric Bradford,*[1,3] Artur M. Schweidtmann[2,3] and Alexei Lapkin[3]

(1) Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, 7491, Norway, ORCID: 0000-0002-0803-0576

(2) Aachener Verfahrenstechnik – Process Systems Engineering, RWTH Aachen University, Aachen, 52062, Germany, ORCID: 0000-0001-8885-6847

(3) Department of Chemical Engineering and Biotechnology, University of Cambridge, Cambridge, CB2 3RA, UK, ORCID: 0000-0001-7621-0889

*E-mail: eric.bradford@cantab.net

Abstract

Many engineering problems require the optimization of expensive, black-box functions involving multiple conflicting criteria, such that commonly used methods like multiobjective genetic algorithms are inadequate. To tackle this problem several algorithms have been developed using surrogates. However, these often have disadvantages such as the requirement of *a priori* knowledge of the output functions or exponentially scaling computational cost with respect to the number of objectives. In this paper a new algorithm is proposed, TSEMO, which uses Gaussian processes as surrogates. The Gaussian processes are sampled using spectral sampling techniques to make use of Thompson sampling in conjunction with the hypervolume quality indicator and NSGA-II to choose a new evaluation point at each iteration. The reference point required for the hypervolume calculation is estimated within TSEMO. Further, a simple extension was proposed to carry out batch-sequential design. TSEMO was compared to ParEGO, an expected hypervolume implementation, and NSGA-II on 9 test problems with a budget of 150 function evaluations. Overall, TSEMO shows promising performance, while giving a simple algorithm without the requirement of a priori knowledge, reduced hypervolume calculations to approach linear scaling with respect to the number of objectives, the capacity to handle noise and lastly the ability for batch-sequential usage.

Keywords Global optimization · Hypervolume · Kriging · Expensive-to-evaluate functions · Response surfaces · Bayesian optimization

1 Introduction

1.1 Expensive black-box optimization

Engineering design problems commonly require the optimization of multiple conflicting criteria. For example in water distribution networks capital, operational, life-cycle and maintenance costs, as well as system reliability and quality of water are the typical goals [1]. It is rarely possible to find a solution that is optimal for all objectives. Instead, the aim is to find a set of points for which, to improve one objective of any set member, the value of at least one other objective needs to be worsened. This set is referred to as Pareto set [2]. The Pareto set is often infinite and cannot be found analytically, such that most algorithms aim to approximate it with a finite number of points [3].

In many applications, the objective functions are black boxes and the derivative information is unavailable. For these problems, deterministic methods cannot be applied. Generally, there are two ways to find points to construct Pareto sets. Either the multiobjective problem is transformed to a single-objective optimization problem known as scalarization or population-based procedures are applied, which work concurrently on all objectives with a set of inputs. One popular scalarization approach uses weighting functions to combine the multiple objectives to a single objective. These single-objective problems are then iteratively solved, while the weights are varied after each iteration to obtain an approximate Pareto set. This is known as weighting sum method. In a similar fashion one can optimize a single objective at a time, while constraining the other objectives by different amounts iteratively. This is known as the ε-constrained method [4]. It is, however, difficult to choose appropriate weightings or constraint values to be able to find points of the Pareto set. It is often impossible to find certain points on a non-convex Pareto front. Therefore, stochastic, population-based methods are commonly applied, such as simulated annealing [5], genetic algorithms [6] and particle swarm algorithms [7]. These improve an initial set of points to approximate the Pareto set by stochastic perturbations.

However, for many problems the evaluations of the objectives are computationally expensive, while the computational time budget is limited. A common instance of expensive problems are high-fidelity computer simulations, which have found extensive use in all areas of engineering [8]. For example a car-crash simulation at Ford took between 36 and 160 hours of computer time for a single run [9]. Modern methodologies for approximating the Pareto set using scalarization or multiobjective stochastic algorithms often require a large number of function evaluations and are therefore highly time-consuming for expensive problems.

For this reason, surrogate-based optimization algorithms have been proposed. These fit a cheap surrogate model to a finite number of points of the expensive objective. The idea is to substitute the objective

with this simple model. Extensive research has been carried out to develop sequential sampling strategies, which do not suffer from the curse of dimensionality like space-filling approaches do [10]. At each instance these approaches use the surrogate to determine the next sampling point(s) and subsequently update the surrogate. Surrogate-based optimization utilizing Gaussian process models (GPs) has shown good performance. This is because GPs provide a predictive distribution of unknown points, which can be used to develop efficient sampling methods. In these, the sampling point is chosen to lie at the maximum of a utility function (a.k.a. acquisition function), which trades off between exploring unknown regions and exploiting regions in which good values have been observed [11]. A prominent single-objective optimization algorithm based on GPs is the "Efficient Global Optimization" (EGO) algorithm [12]. EGO selects the next query point by maximizing the expected improvement (EI) acquisition function [13]. In Łaniewski-Wołłk [14] the EI acquisition function is extended to the relative EI to include observations other than function evaluations, such as derivatives. A review of surrogate-based optimization methods can be found in Jones [15] and Forrester, Keane [16].

1.2 Related work

Several algorithms have been proposed that employ surrogates for multiobjective optimization. A simple idea is to improve existing evolutionary algorithms by inclusion of surrogate models. For example, Voutchkov, Keane [17] have applied GP models in the NSGA-II [18] algorithm instead of the expensive objective function. After running full computations, the surrogates are refined, and the method is continued. Although this approach often works well, it suffers from the fact that it only relies on the prediction provided by the GP and does not actively search in unexplored regions [16].

ParEGO is one of the earliest and simplest extensions of the EGO algorithm to multiple objectives [19]. ParEGO sequentially scalarizes the multiobjective problem with weights that are updated iteratively to explore the Pareto front. The GP is fitted to the transformed data and EI is used to find the next sampling point. There are, however, several disadvantages to the ParEGO algorithm. The scalarization requires accurate knowledge of the limits of the outputs. In addition, the use of uniformly random scalarizing weights does not guarantee a good distribution of non-dominated points, since some Pareto points may be easier to find than others [19]. Lastly, ParEGO uses an augmented Tchebycheff function, which is a discontinuous function due to a max operator and hence violates the continuity assumption of the GP surrogate. ParEGO has been shown to be outperformed by more advanced algorithms on several test problems, which model each objective function individually [20].

Commonly used acquisition functions for single-objective optimization such as the probability of improvement [21] have since been extended for the multiobjective case. For example by considering

the probability that a point augments the Pareto set using GPs for each objective gives a probability of improvement acquisition function for multiobjective problems, which was first proposed by Keane [22]. Weighting the probability of improvement for multiobjective problems with the hypervolume metric, an expected hypervolume (EHV) acquisition function can be obtained [23], which is an extension of EI. The EHV was first proposed in the P.h.D. thesis by Emmerich [24]. Monotonicity properties and formulas for exact computation of EHV have been outlined in Emmerich et al. [25]. A disadvantage of this approach is that the calculation of the EHV is expensive. The output space needs to be divided into several cells, which grow exponentially with the number of objectives. There have been propositions to speed up this process [26]. In particular, the fastest known methods to calculate EHV have been proposed in Emmerich et al. [27] for two objectives and in Yang et al. [28] for three objectives. Another disadvantage of the EHV is the selection of the reference point, which is non-trivial and greatly affects the performance of the method. Apart from EHV, Keane [22] has proposed an acquisition function based on the expected Euclidian distance between Pareto points. This method, however, suffers from similar drawbacks.

1.3 Outline of the paper

This paper proposes an algorithm to approximate Pareto sets in a small number of function evaluations. The algorithm extends the well-known Thompson sampling (TS) method from the multi-armed bandit community [29] to continuous multiobjective optimization. The algorithm was named “Thompson sampling efficient multiobjective optimization” (TSEMO).

The algorithm’s main idea will be commented on using the book by Zhigljavsky, Zilinskas [30] and the paper by Žilinskas [31], which give an overview of the use of different statistical models for global optimization and propose the so-called **P**-algorithm that chooses points based on a probability of improvement for single- and multiobjective problems. In our algorithm each objective function is modelled by an independent GP, which is the same choice as made in Žilinskas [31]. GPs allow to easily incorporate *a priori* knowledge on the objective function by the choice of the covariance function, such as continuity or smoothness. The parameters are adjusted using the data available by maximum likelihood estimation. Given the fitted GPs, a rule is required to choose the next point to evaluate the vector of objectives. In our work we use TS for this decision, which can be reasoned to adhere to the methodology of rational decision making under uncertainty. This methodology states that the method should trade-off the reduction in uncertainty of the objective functions with obtaining a non-dominated point to augment our existing dataset. In TS we determine the Pareto points of the random GP samples, which are taken as possible candidate set for evaluating the expensive functions. The sampled functions will either exploit current knowledge by lying close to the mean functions of the GPs or be considerably different from the mean function particularly in regions with high

variance, which may then lead to a reduction in uncertainty. Lastly, we use the hypervolume criterion to choose the sampling point from the candidate set. Spectral sampling is utilised to efficiently sample the independent GPs to obtain a linear predictor for each objective, which is required for using TS. These linear predictors then allow us to apply a sophisticated multiobjective algorithm, such as NSGA-II to propose the next sampling point. An advantage to this approach is that the linear predictors are cheap to evaluate in the NSGA-II algorithm and only need to be sampled once for each iteration of the algorithm. This contrasts with using a simple linear predictor and variance, given for example by a posterior GP prediction, which can be used in a single-objective problem with a more expensive objective function, such as EHV.

Both spectral sampling of independent GPs and the incorporated NSGA-II algorithm scale linearly, while the hypervolume calculation scales exponentially with respect to the number of objectives. A difference between this algorithm compared to EHV is that the hypervolume calculation is restricted to the output of the NSGA-II algorithm, i.e. the calculation only needs to be carried out once on the output population of the NSGA-II algorithm. In EHV the hypervolume makes up the objective function and hence needs to be calculated for each iteration of the optimization problem, which nearly always necessitates considerably more hypervolume evaluations. In contrast to ParEGO, each objective function is emulated by an independent GP. No prior knowledge is required about the function for this algorithm. In particular, the reference point is estimated within the algorithm, such that no information is needed on the limits of the objective functions. Further, TSEMO can deal with noisy functions. Lastly, a simple heuristic was proposed to add more than one point at each iteration without increasing the computational complexity.

The algorithm proposed in this work has been successfully applied in the development of a decision support tool for chemical manufacturing to trade-off 1) environmental impact measured by a life-cycle assessment and 2) process costs, both of which are estimated by expensive process models [32].

The remainder of the paper is organised as follows. In section 2, GPs are introduced. In section 3, spectral sampling is outlined as a technique to approximately sample functions from GPs. Thereafter, in section 4 the TS method is given for continuous single-objective optimization. In section 5, the proposed algorithm, TSEMO, is outlined. In section 6, the specific implementation of the algorithm is given. Section 7 gives an illustration on how the algorithm works on a simple bi-objective problem. Section 8 presents the performance of TSEMO for both normal and batch-sequential usage with 4 points sampled at each iteration on 9 diverse test problems and compares it to the performance of ParEGO, NSGA-II and EHV. Subsequently, in section 9 a conclusion is given on the proposed algorithm.

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

2 Gaussian processes

This section summarizes the basics about GPs. Further details can be found in [33-35].

2.1 Prior

GPs are an infinite dimensional generalization of a multivariate Gaussian distribution and define a distribution over functions. The GP is fully specified by a mean function $m(\cdot)$ and a covariance function $k(\cdot,\cdot)$. We write y is distributed as a GP with mean function $m(\cdot)$ and covariance function $k(\cdot,\cdot)$, where y is the observation of an underlying function $f(\cdot)$ perturbed by Gaussian distributed noise with variance σ_n^2 [36]:

$$y(\mathbf{x}) \sim GP\big(m(\mathbf{x}), k(\mathbf{x},\mathbf{x}')\big) \tag{1}$$

with

$$m(\mathbf{x}) := \mathbb{E}_f[f(\mathbf{x})] \tag{2}$$

$$k(\mathbf{x},\mathbf{x}') := \mathbb{E}_f\big[(y(\mathbf{x}) - m(\mathbf{x}))\big(y(\mathbf{x}') - m(\mathbf{x}')\big)\big] \tag{3}$$

where $\mathbf{x},\mathbf{x}' \in \mathbb{R}^d$ are arbitrary input vectors and $\mathbb{E}_f(\cdot)$ is the expectation over the function space. The mean function can be interpreted as the 'average' shape of the function, while the covariance function specifies the covariance between any two function values computed at the corresponding inputs [37].

Without loss of generalization, the mean function is set to zero:

$$m(\mathbf{x}) = 0 \tag{4}$$

For GP regression, the covariance function determines the properties of the fitted functions. In this paper, we will focus on stationary covariance functions of the Matérn class. The smoothness of Matérn covariance functions can be adjusted by the parameter ν, such that the corresponding surrogate is $\lfloor \nu/2 - 1 \rfloor$ times differentiable [38]. The squared exponential (SQ-EXP) is the limiting covariance function when $\nu \to \infty$. The following are the most commonly used covariance functions of the Matérn class [33]:

$$k_{\nu=1}(\mathbf{x},\mathbf{x}') = \sigma_f^2 \exp(-r) + \sigma_n^2 \delta(\mathbf{x},\mathbf{x}') \tag{5}$$

$$k_{\nu=3}(\mathbf{x},\mathbf{x}') = \sigma_f^2\big(1+\sqrt{3}r\big)\exp\big(-\sqrt{3}r\big) + \sigma_n^2 \delta(\mathbf{x},\mathbf{x}') \tag{6}$$

$$k_{\nu=5}(\mathbf{x},\mathbf{x}') = \sigma_f^2\left(1+\sqrt{5}r+\frac{5}{3}r^2\right)\exp\big(-\sqrt{5}r\big) + \sigma_n^2 \delta(\mathbf{x},\mathbf{x}') \tag{7}$$

$$k_{\mathrm{SQ-EXP}}(\mathbf{x},\mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}r^2\right) + \sigma_n^2 \delta(\mathbf{x},\mathbf{x}') \tag{8}$$

where $r = \sqrt{(\mathbf{x}-\mathbf{x}')^{\mathrm{T}}\boldsymbol{\Lambda}(\mathbf{x}-\mathbf{x}')}$, $\delta(\mathbf{x},\mathbf{x}')$ is the Kronecker delta function and $\boldsymbol{\Lambda} = \mathrm{diag}\big(\lambda_1^{-2},\ldots,\lambda_d^{-2}\big)$.

The parameter σ_f^2 describes the output variance. The parameters λ_i define the length scales of the input variables. Covariance functions with different length scales for each input dimension are called anisotropic. If an input dimension is not relevant, the corresponding length scale λ_i is large. The hyperparameters which define the GP are jointly denoted by $\boldsymbol{\xi} = \left[\log(\lambda_1), \dots, \log(\lambda_d), \log(\sigma_f), \log(\sigma_n)\right]$, where the hyperparameters are log-transformed for convenience to prevent negative values of the length-scales for the maximum likelihood estimation in in Equation 16.

2.2 Posterior

The prior of the GP is defined in Equation 1, which does not depend on observations. However, it postulates the properties of the functions to be inferred. The next step is to refine the prior by incorporating information from a training data set of n points $X = \{\mathbf{x}_1, \dots \mathbf{x}_n\}$, where each $\mathbf{x}_i$ is a d-dimensional vector $\mathbf{x}_i = [x_{i1}, \dots, x_{id}]^{\mathrm{T}}$, and denoting the corresponding observations at $\mathbf{x}_i$ by y_i to obtain the set $Y = \{y_1, \dots, y_n\}$ and the vector $\mathbf{y} = [y_1, \dots, y_n]^{\mathrm{T}}$. The posterior process can be easily found using Bayes' rule and is given by the following equation:

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')|X, Y) \tag{9}$$

with

$$m(\mathbf{x})|X, Y = \boldsymbol{\Sigma}(\mathbf{x}, X)\boldsymbol{\Sigma}^{-1}\mathbf{y} \tag{10}$$

$$k(\mathbf{x}, \mathbf{x}')|X, Y = k(\mathbf{x}, \mathbf{x}') - \boldsymbol{\Sigma}(\mathbf{x}, X)\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}(\mathbf{x}, X)^{\mathrm{T}} \tag{11}$$

where

$$\boldsymbol{\Sigma} = \left[k\left(\mathbf{x}_i, \mathbf{x}_j\right)\right]_{n\times n} \in \mathbb{R}^{n\times n} \tag{12}$$

$$\boldsymbol{\Sigma}(\mathbf{x}, X) = [k(\mathbf{x}, \mathbf{x}_1), \cdots, k(\mathbf{x}, \mathbf{x}_n)] \in \mathbb{R}^{1\times n} \tag{13}$$

In Equation 11 we can see that the posterior always has a lower variance than the prior, which is expected since more information is available about the function. In Figure 1 two Gaussian processes are shown: on the left the prior and on the right the posterior.

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

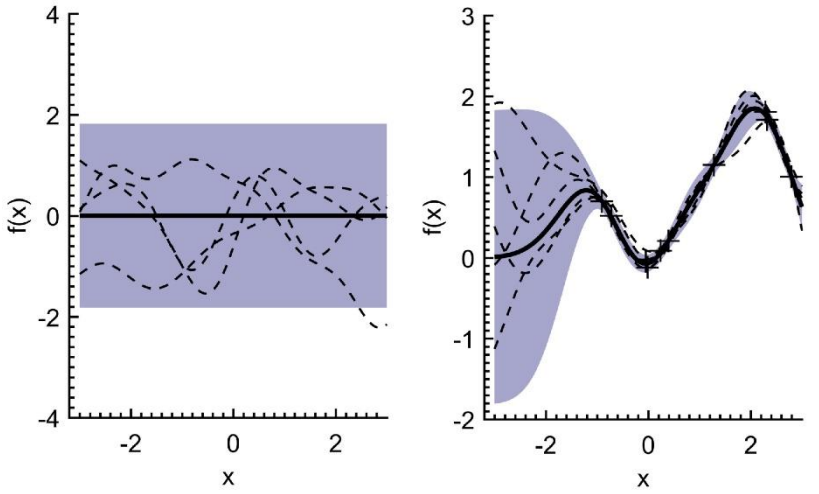

Fig. 1 Illustration of a GP of a 1-dimensional function perturbed by noise. On the left the prior of the GP is shown, while on the right the Gaussian process was fitted to several observations to obtain the posterior. The continuous black-line shows the mean function and the dashed black lines show samples drawn from the GP distribution. The grey area is a 95% confidence area. The black crosses on the right indicate the data used to update the GP.

2.3 Hyperparameter training

Generally, appropriate hyperparameters for a given problem are unknown *a priori*. Therefore, we use the maximum *a posteriori* estimate (MAP) to infer these from data, which has been shown to outperform maximum likelihood estimate for small training sets [39]. In this work, we assume independent Gaussian distributions as prior distributions on the log-transformed hyperparameters:

$$\xi_i \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right) \tag{14}$$

where μ_i and σ_i^2 denote the mean and the variance of the normal distribution of the prior.

The MAP likelihood is then given as follows:

$$\mathcal{L}_{MAP}(\boldsymbol{\xi}) = -\frac{1}{2}\log(|\boldsymbol{\Sigma}|) - \frac{1}{2}\mathbf{y}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{y} - \frac{n}{2}\log(2\pi) + \sum_i \left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\left(\sigma_i^2\right) - \frac{1}{2\sigma_i^2}(\xi_i - \mu_i)^2\right) \tag{15}$$

The MAP hyperparameter estimate is then given by the following optimization problem:

$$\boldsymbol{\xi}_{\mathrm{MAP}} \in \arg\max_{\boldsymbol{\xi}} \mathcal{L}_{\mathrm{MAP}}(\boldsymbol{\xi}) \tag{16}$$

3 Gaussian process spectral sampling

There are no known methods that allow to sample an exact function from a GP. In this section a method is briefly outlined to create an approximate analytical sample of a GP, first proposed by Hernández-Lobato et al. [40]. Given a stationary kernel k, a theorem by Bochner

[41] guarantees the existence of its Fourier dual $s(\boldsymbol{\omega})$, which is known as the spectral density of k. We can write $k(\mathbf{x},\mathbf{x}') = k(\mathbf{x}-\mathbf{x}',\mathbf{0})$ as:

$$k(\mathbf{x},\mathbf{x}') = \int_{\mathbb{R}^d} e^{-j\boldsymbol{\omega}^{\mathrm{T}}(\mathbf{x}-\mathbf{x}')} s(\boldsymbol{\omega})\, d\boldsymbol{\omega} \tag{17}$$

$$s(\boldsymbol{\omega}) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} e^{j\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\delta}}\, k(\boldsymbol{\delta},\mathbf{0}) d\boldsymbol{\delta} \tag{18}$$

The associated normalized probability density can then be expressed as $p(\boldsymbol{\omega}) = s(\boldsymbol{\omega})/\alpha$, where $\alpha = \int s(\boldsymbol{\omega})\, d\boldsymbol{\omega}$ is a proportionality constant. The integral in Equation 17 can be expressed as an expectation:

$$k(\mathbf{x},\mathbf{x}') = \alpha \int_{\mathbb{R}^d} e^{-j\boldsymbol{\omega}^{\mathrm{T}}(\mathbf{x}-\mathbf{x}')} p(\boldsymbol{\omega})\, d\boldsymbol{\omega} = \alpha \mathbb{E}_{\boldsymbol{\omega}}\left[\zeta(\mathbf{x})\overline{\zeta(\mathbf{x}')}\right] \tag{19}$$

where $\zeta(\mathbf{x}) = e^{-j\boldsymbol{\omega}^{\mathrm{T}}\mathbf{x}}$.

Since both probability distribution $p(\boldsymbol{\omega})$ and covariance function $k(\mathbf{x},\mathbf{x}')$ are real, the integral in Equation 19 converges if the complex exponentials are substituted by cosine expressions [42]. Using the sum of angles formula it can be shown that $\zeta(\mathbf{x}) = \sqrt{2\alpha}\cos\left(\boldsymbol{\omega}^{\mathrm{T}}\mathbf{x} + b\right)$ satisfies Equation 19 [40], where $b \sim U(0,2\pi)$.

According to Equation 19, $k(\mathbf{x},\mathbf{x}')$ can be estimated by a Monte-Carlo approach by defining the vector:

$$\boldsymbol{\zeta}(\mathbf{x}) = \sqrt{2\alpha/M}\cos(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{20}$$

where M denotes the number of Monte Carlo samples, $[\mathbf{W}]_i \sim p(\boldsymbol{\omega})$ and $[\mathbf{b}]_i \sim U(0,2\pi)$ are stacked versions of $\boldsymbol{\omega}$ and b respectively.

$k(\mathbf{x},\mathbf{x}')$ can then be approximated by the following inner product:

$$k(\mathbf{x},\mathbf{x}') \approx \boldsymbol{\zeta}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\zeta}(\mathbf{x}') \tag{21}$$

$\boldsymbol{\zeta}$ allows us to approximate the Gaussian process prior given in Equation 1 with a linear model [40]:

$$f(\mathbf{x}) = \boldsymbol{\zeta}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\theta} \tag{22}$$

where $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$.

In the light of data, $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{m},\mathbf{V})$, with

$$\mathbf{m} = \left(\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{y} \tag{23}$$

$$\mathbf{V} = \left(\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \sigma_n^2\mathbf{I}\right)^{-1}\sigma_n^2 \tag{24}$$

where $[\mathbf{Z}]_i = \boldsymbol{\zeta}(\mathbf{x}_i)$ consists of stacked random vectors of $\boldsymbol{\zeta}$ evaluated at the inputs of the data.

Let $\boldsymbol{\zeta}^{(i)}(\mathbf{x})$ and $\boldsymbol{\theta}^{(i)}$ be random vectors corresponding to the probability densities given above, then an approximate posterior sample from Equation 9 is given by:

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

$$f^{(i)}(\mathbf{x}) = \boldsymbol{\zeta}^{(i)}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\theta}^{(i)} \tag{25}$$

This sample can be evaluated at any $\mathbf{x}$. It can be shown that following this procedure the function samples have mean and covariance as given by the Gaussian posterior in Equation 9 [40]. To use the procedure above we need to establish the proportionality constant and the probability density for the covariance functions in question. For the covariance functions introduced in section 2, the proportionality constant is given by [43]:

$$\alpha = \sigma_f^2 \tag{26}$$

The probability density, $p(\boldsymbol{\omega})$, associated with the Matérn covariance functions can be obtained from the Fourier transform in Equation 18 divided by the proportionality constant and takes the form of a multivariate t-distribution with degrees of freedom ν. For the squared exponential kernel, as $\nu \to \infty$, the multivariate t-distribution reduces to the multivariate normal distribution.

$$p_{\text{Matérn}}(\boldsymbol{\omega}) = T(\mathbf{0}, \boldsymbol{\Lambda}, \nu) \tag{27}$$

$$p_{\text{SQ-EXP}}(\boldsymbol{\omega}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}) \tag{28}$$

The algorithm to sample a posterior function $f^{(i)}(\mathbf{x})$ using spectral sampling is summarized in Table 1.

Table 1 Algorithm to draw an approximate sample from the posterior distribution of a GP using spectral sampling.

Algorithm 1

Input: Optimal hyperparameters $\boldsymbol{\xi}_{\text{MAP}}$ and type of covariance function of GP
Training data set of n points $X = \{\mathbf{x}_1, \ldots \mathbf{x}_n\}$
Corresponding set of observations $Y = \{y_1, \ldots y_n\}$

1. Draw M i.i.d. samples $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_M \in \mathbb{R}^d$ from $p(\boldsymbol{\omega})$ and M i.i.d. samples $b_1, \ldots, b_M \in \mathbb{R}$ from the uniform distribution on $[0,2\pi]$ to build $\mathbf{W}^{(i)}$ and $\mathbf{b}^{(i)}$
2. Let $\boldsymbol{\zeta}^{(i)}(\mathbf{x}) = \sqrt{2\alpha/M}\cos\left(\mathbf{W}^{(i)}\mathbf{x} + \mathbf{b}^{(i)}\right)$
3. Draw $M \times n$ i.i.d. samples of $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{M\times n} \in \mathbb{R}^d$ from $p(\boldsymbol{\omega})$ and $M \times n$ i.i.d. samples $b_1, \ldots, b_{M\times n} \in \mathbb{R}$ from the uniform distribution on $[0,2\pi]$ to build $\mathbf{Z}^{(i)}$
4. Let $\mathbf{m}^{(i)} = \left({\mathbf{Z}^{(i)}}^{\mathrm{T}}\mathbf{Z}^{(i)} + \sigma_n^2\mathbf{I}\right)^{-1}{\mathbf{Z}^{(i)}}^{\mathrm{T}}\mathbf{y}$ and $\mathbf{V}^{(i)} = \left({\mathbf{Z}^{(i)}}^{\mathrm{T}}\mathbf{Z}^{(i)} + \sigma_n^2\mathbf{I}\right)^{-1}\sigma_n^2$
5. Draw $\boldsymbol{\theta}^{(i)}$ from the multivariate Gaussian distribution $\mathcal{N}\left(\mathbf{m}^{(i)}, \mathbf{V}^{(i)}\right)$

Output: Approximate parametric sample of posterior GP: $f^{(i)}(\mathbf{x}) = \boldsymbol{\zeta}^{(i)}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\theta}^{(i)}$

4 Single-objective Thompson Sampling

TS for single-objective optimization is outlined in this section. The problem is to find the true global minimizer $\mathbf{x}^*$ of a black-box objective function g:

$$\mathbf{x}^* \in \underset{\mathbf{x}\in\mathcal{X}\subseteq\mathbb{R}^d}{\arg\min}\, g(\mathbf{x}) \tag{29}$$

where $\mathcal{X}$ is the design space and $\mathbf{x}$ is the decision variable.

TS was introduced in 1933 [29] and is one of the oldest heuristics in the multi-armed bandit community, producing both empirical results [44-46] and theoretical results [47,48]. TS achieves its

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

exploration/exploitation through randomness. The basic idea of TS is to choose an action that is matched with the probability that the action leads to the optimum reward. For Bayesian optimization of continuous functions considering minimization, this refers to sampling a function $f^{(i)}$ from the posterior distribution defined by the GP at each iteration i and then minimizing this function to obtain $\mathbf{x}_*^{(i)} = \arg\min_{\mathbf{x}\in\mathcal{X}} f^{(i)}(\mathbf{x})$.

Following this process, $\mathbf{x}_*^{(i)}$ is distributed as $p(\mathbf{x}^*|Y,X)$, which is taken as the sampling point at each iteration. The algorithm is given below in Table 2 and was first suggested by Shahriari et al. [46] to be used for continuous search spaces directly.

Table 2 Algorithm for single-objective optimization using TS.

Algorithm 2
Input: Black-box function $g(\mathbf{x})$
Type of covariance function of GP
Initial training data set of n points $X_0 = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$
Corresponding set of observations $Y_0 = \{y_1, \dots y_n\}$
1. for $i \coloneqq 0, \dots, N$
2. Train GP from current dataset X_i and Y_i
3. Approximately sample $f^{(i)}(\mathbf{x})$ from $GP(m, k|X_i, Y_i)$
4. Determine $\mathbf{x}_{n+1} = \operatorname{argmin}_{\mathbf{x}\in\mathcal{X}} f^{(i)}(\mathbf{x})$
5. Evaluate $y_{n+1} = g(\mathbf{x}_{n+1})$
6. Update dataset $X_{i+1} = \{\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{x}_{n+1}\}$, $Y_{i+1} = \{y_1, \dots y_n, y_{n+1}\}$
7. Update $n \coloneqq n + 1$
8. end for
Output: X_N, Y_N

Empirically, TS has been shown to perform well, for example outperforming the famous EI acquisition function on the Branin test function [46].

5 Thompson sampling efficient multiobjective optimization

5.1 Objective

In this section the TSEMO algorithm is outlined. The algorithm is designed to solve multiobjective optimization problems which can be defined as follows:

$$\underset{\mathbf{x}\in\mathcal{X}\subseteq\mathbb{R}^d}{\text{minimize}}\ \mathbf{G}(\mathbf{x}) = [g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_m(\mathbf{x})] \qquad (30)$$

where $\mathcal{X}$ is the design space, $\mathbf{x}$ is the decision vector and $\mathbf{G}$ is a vector of m scalar objectives $g_i(\mathbf{x})$ to be minimized. It is assumed that each black-box function $g_i(\mathbf{x})$ is continuous and can be evaluated at an arbitrary query point $\mathbf{x}$ in the domain $\mathcal{X}$ and that this process produces noisy outputs such that the noise is unbiased.

The single-objective procedure of TS in Bayesian optimization can be extended to the multiobjective case. For discrete search spaces, TS has been employed for multiobjective problems by applying scalarization and a method based on finding the Pareto set at each iteration. An empirical comparison showed that the method based on the full Pareto

set outperformed the scalarization approach [49]. In this paper, we therefore focus on using TS by finding points to approximate the Pareto front of the objective functions. At each instance the algorithm determines a number of sampling inputs by trading-off exploration and exploitation to improve the current Pareto front in the light of previous observations of the objective functions. The final output is a set of inputs which should contain points close to the true Pareto set.

5.2 Algorithm outline

The algorithm is first outlined for the case where one sampling point is proposed at each iteration. In Figure 2, a flowchart of the overall TSEMO algorithm is given for this case. The extension to batch-sequential operation is then given in the next section.

5.2.1 Initialization

To initialize the algorithm an initial dataset needs to be first provided to build the initial GPs, for example using a Latin hypercube design [50]. Let n equal the size of the initial dataset.

5.2.2 Determine candidate set for sampling

Assume we are at iteration i, such that TSEMO has been employed i-times to find i sampling points. Let $X^{(i)} \coloneqq \{\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{x}_{n+1}, \dots, \mathbf{x}_{n+i}\}$ be the inputs of the data collected and $Y_j^{(i)} \coloneqq \left\{y_j^{(1)}, \dots y_j^{(n)}, y_j^{(n+1)}, \dots, y_j^{(n+i)}\right\}$ the corresponding responses for each objective function $g_j(\mathbf{x})$, with $j = 1, \dots, m$. For each $Y_j^{(i)}$, a corresponding independent GP is trained, that is we find $GP_j^{(i)}\left(m^{(i)}, k^{(i)} \middle| X^{(i)}, Y_j^{(i)}\right)$ for $j = 1, \dots, m$ from the procedure outlined in section 2. Following a similar procedure as for the single-objective TS, we draw m distinct functions from these m independent GPs using spectral sampling, which is outlined in section 3. From this, we obtain a collection of m functions $\left\{f_1^{(i)}(\mathbf{x}), \dots, f_m^{(i)}(\mathbf{x})\right\}$. We now want to match the probability that the next sample leads to a Pareto optimal point. To accomplish this, we find the approximate Pareto set of the sampled functions $\left\{f_1^{(i)}(\mathbf{x}), \dots, f_m^{(i)}(\mathbf{x})\right\}$ at each iteration. Since the GP samples are cheap-to-evaluate functions, a multiobjective algorithm of choice can be used to accomplish this. Common choices would be a genetic algorithm, such as the NSGA-II algorithm [18]. Let $C^{(i)}$ refer to the current candidate set given by the approximate Pareto set of the GP samples $\left\{f_1^{(i)}(\mathbf{x}), \dots, f_m^{(i)}(\mathbf{x})\right\}$. In the case of a multiobjective genetic algorithm such as NSGA-II, the size of the candidate set $C^{(i)}$ is equal to the population size. The number of generations is fixed to allow sufficient convergence of this set to the true Pareto set of the GP samples $\left\{f_1^{(i)}(\mathbf{x}), \dots, f_m^{(i)}(\mathbf{x})\right\}$.

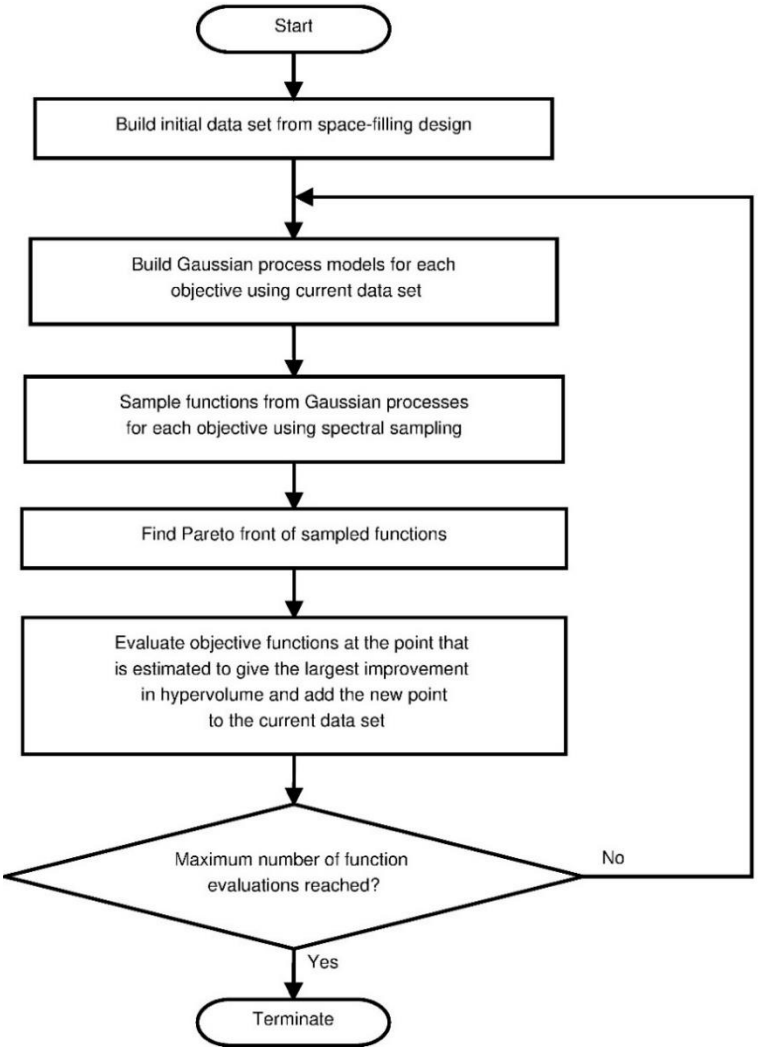

Fig. 2 Algorithm flowchart for TSEMO algorithm

5.2.3 Select point from candidate set

In theory, one could pick a random point from the approximate Pareto set $C^{(i)}$ to match probabilities. One can see that this is a valid approach by noting that once enough data has been sampled the Pareto set of the m independent posterior GPs converges to the true Pareto set of the objective functions, i.e. following this approach eventually a good approximation of the Pareto set is found. Nonetheless, this is not particularly efficient, since the resulting approximate Pareto front might not be well spaced-out due to the randomness involved. Instead, we propose to use the hypervolume criterion to choose the next sampling point.

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

The hypervolume indicator is the *m*-dimensional Lebesgue measure λ_m of a dominated subspace limited above by a reference point and can be defined as [27]:

$$HV(\mathbb{P}, \mathbf{R}) = \lambda_m\left(\cup_{\mathbf{p}\in\mathbb{P}}[\mathbf{p}, \mathbf{R}]\right) \tag{31}$$

where $HV(\mathbb{P}, \mathbf{R})$ is equal to the hypervolume indicator corresponding to the non-dominated Pareto front $\mathbb{P}$ and the reference point **R.**

Define $\mathcal{P}^{(i)}$ as the Pareto front of the current output data set $\left\{Y_1^{(i)}, \dots, Y_m^{(i)}\right\}$ and $\mathbf{r}^{(i)}$ as the current reference point for the hypervolume calculation. We then want to sample at the point that gives the largest hypervolume improvement (ΔHV) once it is added to the current Pareto front $\mathcal{P}^{(i)}$, that is:

$$\mathbf{x}_{n+i+1} \in \arg\max_{\mathbf{x}\in C^{(i)}} \Delta HV\left(\mathbf{y}_C, \mathcal{P}^{(i)}, \mathbf{r}^{(i)}\right) \tag{32}$$

where $\mathbf{y}_C = \left(f_1^{(i)}(\mathbf{x}), \dots, f_m^{(i)}(\mathbf{x})\right)$ and

$$\Delta HV\left(\mathbf{y}_C, \mathcal{P}^{(i)}, \mathbf{r}^{(i)}\right) = HV\left(\mathcal{P}^{(i)} \cup \{\mathbf{y}_C\}, \mathbf{r}^{(i)}\right) - HV\left(\mathcal{P}^{(i)}, \mathbf{r}^{(i)}\right)$$

A fast algorithm to calculate ΔHV is important, since it needs to be carried out for each candidate point in $C^{(i)}$. The efficient computation of ΔHV has been the subject of several papers due to its importance in different indicator-based multi-objective metaheuristics, such as EHV. For two objectives we used an algorithm that was first proposed in Emmerich et al. [27]. The approach could be shown to have asymptotically optimal time complexity of $\mathcal{O}\left(\ell log(\ell)\right)$, where ℓ refers to the number of data-points in the Pareto front approximation. In the case that the Pareto front approximation is sorted by a coordinate, the complexity can be reduced to $\mathcal{O}(\ell)$, which can be easily achieved in Bayesian optimization. In Yang et al. [28] the approach is extended to three objectives with a time complexity of $\mathcal{O}\left(\ell log(\ell)\right)$, which we consequently implemented. A summary for the efficient computation of ΔHV is given in Emmerich, Fonseca [51]. Lastly, for more than three objectives ΔHV is approximated by using a Monte-Carlo approximation [52].

The reference point $\mathbf{r}^{(i)}$ is assumed to be unknown and is instead approximated by the anti-ideal point of the approximate Pareto front of the GP samples $\left\{f_1^{(i)}(\mathbf{x}), \dots, f_m^{(i)}(\mathbf{x})\right\}$ from the candidate set $C^{(i)}$:

$$\mathbf{r}^{(i)} = \left(\max_{\mathbf{x}\in C^{(i)}}\left(f_1^{(i)}(\mathbf{x})\right), \dots, \max_{\mathbf{x}\in C^{(i)}}\left(f_m^{(i)}(\mathbf{x})\right)\right) \tag{33}$$

This choice is made since it gives us a reference point that encloses the extreme values of the candidate set $C^{(i)}$ for the hypervolume calculation in Equation 32. In addition, if the function is well-known, this reference point converges to the value of the anti-ideal point of the true function.

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

In this the Pareto front of the current data set is given by $\mathcal{P}^{(i)} = \{a, b, c, d\}$. We assume that the candidate approximate Pareto set $C^{(i)}$ of the GP samples consists of 6 points with a corresponding Pareto front $\left\{\mathrm{C}_j = \left(f_1^{(i)}(\mathbf{x}_j), f_2^{(i)}(\mathbf{x}_j)\right)\right\}_{\mathbf{x}_j \in C^{(i)}}$, where each C_j represents a separate point on this Pareto front. The reference point $\mathbf{r}^{(i)} = r$ is subsequently equal to the anti-ideal point of this Pareto front, in this case the f_1-value of C_6 and the f_2-value of C_1. In the example, the hypervolume of the current hypervolume shown in red is improved by the blue area if the point C_3 were added. Carrying out this calculation for every combination of C_j, the corresponding sampling point of $C^{(i)}$ is then chosen that yields the largest overall hypervolume improvement.

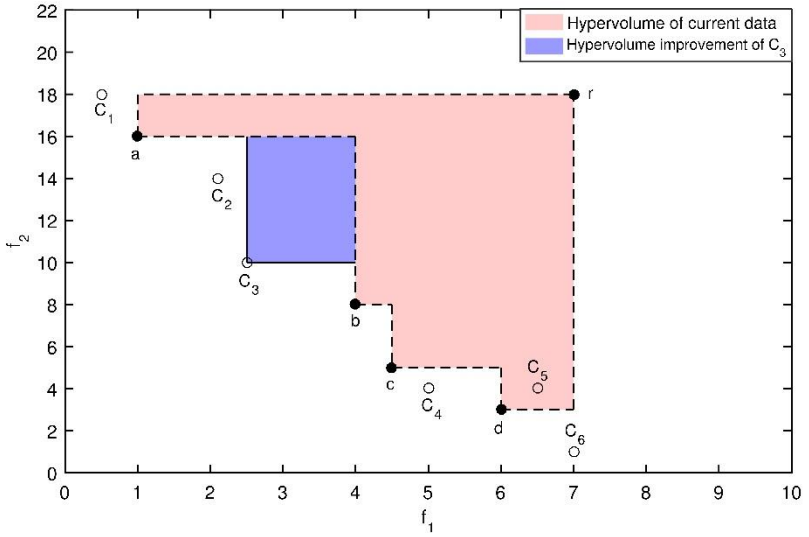

Fig. 3 Illustration of the use of the hypervolume criterion to pick a point to sample. The hypervolume of the Pareto front $\mathcal{P}^{(i)} = \{a, b, c, d\}$ is shown in light red. r is the reference point in this figure picked as the anti-ideal point of the approximate Pareto front of the GP samples. The blue area shows the contribution that adding the point C_3 would make to form the new Pareto front.

5.2.4 Evaluate expensive functions and update data-sets

Lastly, the data sets are updated with the proposed data point: $X^{(i+1)} \coloneqq \{\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{x}_{n+1}, \dots, \mathbf{x}_{n+i}, \mathbf{x}_{n+i+1}\}$ and $Y_j^{(i)} \coloneqq \left\{y_j^{(1)}, \dots y_j^{(n)}, y_j^{(n+1)}, \dots, y_j^{(n+i)}, g_j(\mathbf{x}_{n+i+1})\right\}$ for $j = 1, \dots, m$ and the procedure is repeated with $i \coloneqq i + 1$ until a specified maximum number of function evaluations has been reached.

5.3 Batch-sequential sampling

It is often advantageous to propose multiple sampling points at each iteration known as batch-sequential design. This has become particularly releva31nt due to the advent of parallel computing. Let b denote the number of sampling points added each iteration and i the current iteration. TSEMO has then been employed i-times to add $i \times b$ number of points. Let $X^{(i)} = \{\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{x}_{n+1}, \dots, \mathbf{x}_{n+i\times b}\}$ be the

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

inputs of the data collected and $Y_j^{(i)} = \left\{y_j^{(1)}, \dots y_j^{(n)}, y_j^{(n+1)}, \dots, y_j^{(n+i\times b)}\right\}$ the corresponding responses for each objective function $g_j(\mathbf{x})$, with $j = 1, \dots, m$. The procedure is precisely the same as in section 5.2 using the data $X^{(i)}$ and $Y_j^{(i)}$ except that Equation 32 is replaced with Equations proposing multiple sampling points. A simple heuristic that works well in practice is to pick multiple points from the candidate set $C^{(i)}$ which are predicted to give the highest hypervolume improvement. To accomplish this optimization, we use a greedy approximation. The multiple sampling points at each iteration are then found by the following equations:

$$\mathbf{x}_{n+i\times b+1} \in \arg\max_{\mathbf{x}\in C^{(i)}} \Delta HV\left(\mathbf{y}_C, \mathcal{P}^{(i)}, \mathbf{r}^{(i)}\right) \tag{34}$$

$$\mathbf{x}_{n+i\times b+2} \in \arg\max_{\mathbf{x}\in C^{(i)}} \Delta HV\left(\mathbf{y}_C, \mathcal{P}^{(i)} \cup \left\{\left(f_1^{(i)}(\mathbf{x}_{n+i\times b+1}), \dots, f_m^{(i)}(\mathbf{x}_{n+i\times b+1})\right)\right\}, \mathbf{r}^{(i)}\right) \tag{35}$$

$$\vdots$$

$$\begin{aligned}\mathbf{x}_{n+i\times b+b} \in \arg\max_{\mathbf{x}\in C^{(i)}} \Delta HV\Big(\mathbf{y}_C, \mathcal{P}^{(i)} \cup \\ \left\{\left(f_1^{(i)}(\mathbf{x}_{n+i\times b+1}), \dots, f_m^{(i)}(\mathbf{x}_{n+i\times b+1})\right)\right\} \cup \\ \left\{\left(f_1^{(i)}(\mathbf{x}_{n+i\times b+2}), \dots, f_m^{(i)}(\mathbf{x}_{n+i\times b+2})\right)\right\} \cup \\ \dots\left\{\left(f_1^{(i)}(\mathbf{x}_{n+i\times b+b-1}), \dots, f_m^{(i)}(\mathbf{x}_{n+i\times b+b-1})\right)\right\}, \mathbf{r}^{(i)}\Big)\end{aligned} \tag{36}$$

where $\mathbf{y}_C = \left(f_1^{(i)}(\mathbf{x}), \dots, f_m^{(i)}(\mathbf{x})\right)$

The optimizations are carried out by exhaustively calculating the hypervolume for all points in $C^{(i)}$. Lastly, the data sets are updated with the proposed data points: $X^{(i+1)} \coloneqq \{\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{x}_{n+1}, \dots, \mathbf{x}_{n+i\times b}, \mathbf{x}_{n+i\times b+1}, \dots, \mathbf{x}_{n+i\times b+b}\}$ and $Y_j^{(i)} \coloneqq \left\{y_j^{(1)}, \dots y_j^{(n)}, y_j^{(n+1)}, \dots, y_j^{(n+i\times b)}, g_j(\mathbf{x}_{n+i\times b+1}), \dots, g_j(\mathbf{x}_{n+i\times b+b})\right\}$ for $j = 1, \dots, m$ and the procedure is repeated with $i \coloneqq i + 1$ until a specified maximum number of function evaluations are reached, similar to section 5.2.

6 Implementation of the TSEMO algorithm

This section outlines the implementation of the TSEMO algorithm used. All covariance functions given in section 2.1 can be used. To build the GP model the hyperparameters need to be optimized. This is done by determining the MAP estimate given in Equation 16 with Gaussian priors on the hyperparameters. For this optimization problem the DIRECT search algorithm [53] was used, followed by a local search using MATLAB's *fmincon* function. The various samples from

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

normal distributions and t-distributions were carried out using MATLAB's Machine Learning Toolbox™. The approximate Pareto front of the sampled function was found by employing a NSGA-II [18] implementation in MATLAB [54]. For several GP calculations the Gaussian Process for Machine Learning (GPML) toolbox by Rasmussen, Nickisch [55] was utilised. To determine if a Pareto set point augments the current Pareto front for the hypervolume calculation the function *paretoset.m* was employed [56]. The hypervolume improvement (ΔHV) for the two dimensional and three dimensional case was calculated analytically using implementations that are made available under http://moda.liacs.nl as C++ and Matlab source-code corresponding to the state-of-the-art described in section 5.2 [28,51,27], while for more than three objectives the hypervolume was estimated using a Monte-Carlo approach by utilising the MATLAB function *hypervolume.m* [57]. The default specifications of TSEMO are given in Table 3. Options not given were kept at default for the respective algorithm. The algorithm TSEMO as described here can be found on Bradford, Schweidtmann [58].

Table 3 Default options of TSEMO algorithm.

Option	Value
Number of points added each iteration	1
Type of Matérn, ν	1
Prior variance of $\log \lambda_1 \dots, \log \lambda_d, \log \sigma_f, \log \sigma_n$	10
Prior mean of $\log \lambda_1 \dots, \log \lambda_d, \log \sigma_f$	0
Prior mean of $\log \sigma_n$	-6
Number of direct-algorithm iterations	200/number of variables
Relative tolerance of local search	10^{-12}
Number of Monte-Carlo points for spectral sampling	4,000
NSGA-II population size	100
NSGA-II number of generations	100
Number of Monte-Carlo points for hypervolume	3,000

7 Illustration of the algorithm

Here we show with the aid of a simple one-dimensional test problem how TSEMO works. For this purpose, the bi-objective Schaffer function No. 1 [59] was chosen, which consists of the following optimization problem:

$$\underset{x\in[-10,10]}{\text{minimize}}\ [x^2, (x-2)^2] \tag{37}$$

The Pareto set in the problem above is equal to $[0,2]$. The Matérn 5 covariance function was used for both GPs from which the functions were sampled. The options were otherwise kept at default. The progress of the algorithm is highlighted in Figure 4. Initially, the samples deviate substantially from the function in the first two iterations, which leads to a sampling point outside of the Pareto set. This is expected and corresponds to the exploration action of the

algorithm when uncertainty is high due to limited data. After two iterations, however, the uncertainty is reduced to the point that the function is well identified and the Pareto set found by the multiobjective optimization of the sampled functions is close to the true Pareto set. Consequently, the next sampling point is chosen to exactly lie in the Pareto set to give the highest improvement in hypervolume. Thereafter, all iterations lead to sampling points inside the Pareto set.

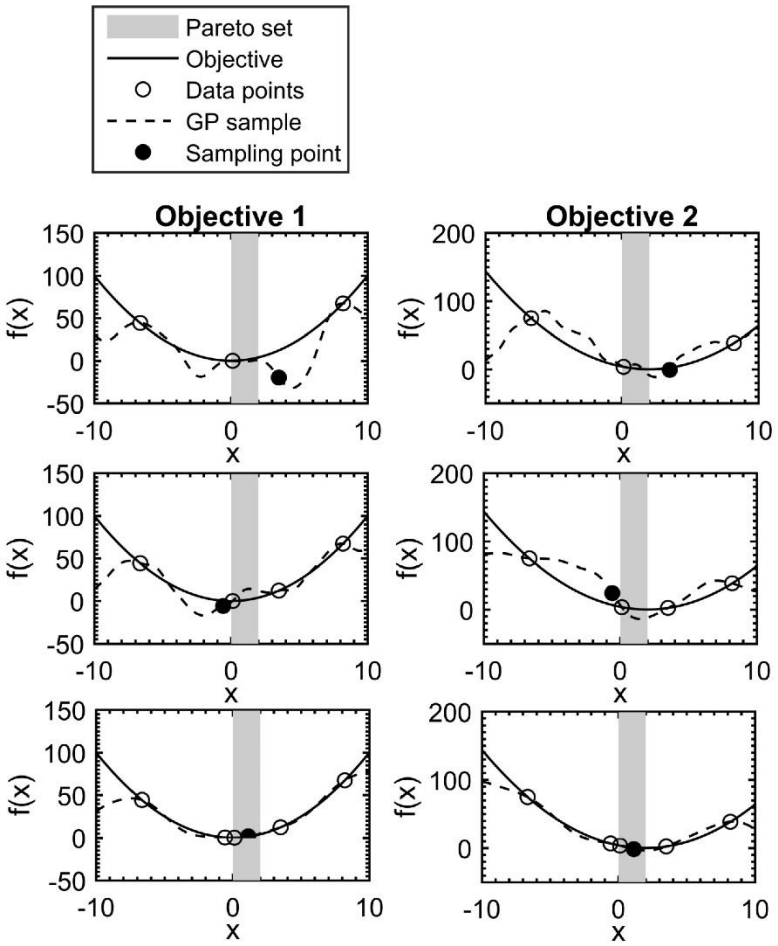

Fig. 4 An example of TSEMO algorithm using Schaffer function No. 1, which is a multiobjective test function of one dimension (solid line), with an initial design consisting of three points (represented by open circles in the first two graphs). The Pareto set corresponds to all $x \in [0,2]$ shown in grey. The graphs on the left correspond to the first objective, while the graphs on the right represent the second objective. Three successive iterations are shown progressing from top to bottom. The dashed lines correspond to the samples drawn by the respective GPs of each objective. Suggested sampling points corresponding to certain values of the GP samples are shown as filled black circles, while the current data set is shown as open circles.

8 Numerical experiments

Numerical experiments were carried out to show the relevance of the algorithm for both simple and batch-sequential usage. These are presented and discussed in this section. Only noiseless cases are treated here, however the algorithm can identify noise through the

noise-term implemented in the GPs and could be used for cases with noise as well.

8.1 Optimization test problems

The algorithm's performance is tested on several diverse optimization test problems and compared to NSGA-II [18] implemented by Lin [54] in MATLAB, ParEGO [19] for which the data is available on GitHub [60], and lastly to an expected hypervolume implementation in the SUMO [61] toolbox in MATLAB. NSGA-II was chosen to compare to a genetic algorithm, which is commonly used to solve derivative-free multiobjective optimization problems. ParEGO is the most sophisticated approach for multiobjective optimization employing GPs. We compare to EHV since it employs multiple GPs without scalarization like the algorithm proposed in this paper. In addition, the TSEMO algorithm was run for the case where a single point is added at each iteration and where 4 points were added at each iteration according to the heuristic given in section 5.2.4 to get a feel for the performance of the batch-sequential case. The batch-sequential case is referred to as "BS-TSEMO". The initial datasets for TSEMO, ParEGO, EHV and BS-TSEMO were generated using a Latin hypercube design with an initial dataset size of 11d-1. Overall the algorithms were tested on 9 test problems taken from Knowles [19] ranging from 2 to 8 input dimensions and 2 to 3 output dimensions. A summary of the test problems with descriptions of the Pareto fronts is given in Table 4. The exact equations for the test problems can be found in Knowles [19]. A budget of 150 total function evaluations was set including the initial dataset. 20 independent runs were carried out for each constellation of algorithm and test problem to compare the average performance. The overall results can be found in section 8.3. The algorithm options for ParEGO, NSGA-II and EHV were kept mostly at default with a few notable exceptions: NSGA-II population size was changed to 20 and the total number of generations to 8 to match the small number of function evaluations allowed, for EHV the sampling criterion was optimized using an implemented particle-swarm algorithm with a population size of 100 and the number of iterations set to 100 to match the NSGA-II algorithm used within TSEMO, since otherwise only a local optimizer would be run.

Table 4 Multiobjective optimization test functions and descriptions from Cristescu, Knowles [62].

Test functions	Number of inputs	Number of objectives	Description of Pareto front [62]
KNO1	2	2	"True Pareto front lies just beyond a locally optimum Pareto front with a much larger basin of attraction"
VLMOP2	2	2	"Pareto front is concave"
VLMOP3	2	3	"Pareto optimal set is disconnected and Pareto front is a curve following a

			convoluted path through objective space"
OKA1	2	2	"The density of the solutions fall away near to the Pareto front"
OKA2	3	2	"Pareto optima lie on a spiral shapes curve and the density of the solutions falls away steeply near to the Pareto front"
DTLZ1a	6	2	"Increased level of function ruggedness"
DTLZ2a	8	3	"Pareto front is a sphere"
DTLZ4a	8	3	"Pareto front is a sphere and the density of solution is biased"
DTLZ7a	8	3	"Pareto front consists of four disconnected regions"

8.2 Performance assessment

In this work we compare the performance of the multiobjective algorithms using two different methods. Firstly, we compare the algorithms using three different unary quality indicators. These return a single scalar value, which allows us to judge the goodness of the approximate Pareto fronts. The second method involves the plotting of so-called empirical worst attainment summary surfaces, which visualize the performance and weaknesses of the algorithms over the different runs.

The quality of the approximate Pareto front is dependent on both the closeness to the true Pareto front referred to as the "convergence", the distribution as well as the spread of points on the approximate Pareto front known as the "diversity" and lastly the number of solutions. These aspects have been used to define various multiobjective performance metrics [63]. Various studies such as Zitzler et al. [64] and Knowles et al. [65] have analysed the various tools available. It could be shown that many commonly used measures give misleading results. For the unary quality indicators, we chose to compare the hypervolume, the modified inverted generational distance [66] and the generalized spread [67].

8.2.1 Hypervolume

The hypervolume indicator defines the volume of the objective space dominated by the respective solution set, which consequently considers both convergence and diversity. For an exact definition and for properties of the hypervolume indicator please refer to Zitzler [68]. The hypervolume can be shown to be dominance compliant [65]. The higher the hypervolume, the better the performance of the respective algorithm. The reference point was chosen by following a procedure in Knowles [19]. First, all sets of non-dominated points after 150 function evaluations from all runs of all algorithms were collected and combined to a single superset. From this superset, the ideal and anti-ideal points were found for each objective. The reference point was

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

simply the anti-ideal point shifted by 0.01 of the difference of the ideal and anti-ideal point. The hypervolume was calculated as described in section 5.2.

8.2.2 Modified inverted generational distance

The inverted generational distance (IGD) is another popular measure that takes into account both convergence and diversity. Let P^* be a set of uniformly distributed points in the objective space along the true Pareto front of the multiobjective problem and let P be the approximate Pareto front obtained from finite runs of the algorithms. The IGD is defined as the average minimum distance from the points in $P^*$to P. A small IGD represents an approximate Pareto front close to the true Pareto front, hence the smaller the IGD the better. In the original IGD the distance is defined as the Euclidian distance, which is Pareto non-compliant [64]. However, we instead use a modified distance as proposed in Ishibuchi et al. [66], which makes the measure weakly Pareto compliant. The modified IGD is then defined as follows:

$$IGD(P, P^*) = \frac{1}{|P^*|}\sum_{\mathbf{v}\in P^*} \min_{\boldsymbol{\zeta}\in P} d^+(\mathbf{v}, \boldsymbol{\zeta}) \tag{38}$$

where $|P^*|$ describes the cardinality of the set P^*, $\min_{\boldsymbol{\zeta}\in P} d^+(\mathbf{v}, \boldsymbol{\zeta})$ is the minimum modified distance between $\mathbf{v}$ and the points in P defined as $d^+(\mathbf{v}, \boldsymbol{\zeta}) = \sqrt{\max(\zeta_1 - v_1, 0)^2 + \cdots + \max(\zeta_m - v_m, 0)^2}$.

8.2.3 Generalized spread

The third measure used is the generalized spread (Δ^*), which was defined by Zhou et al. [67] as a generalization of the Δ metric to more than 2-dimensions. The Δ^* metric should be regarded with caution, since it only measures the diversity of the solutions and is therefore Pareto non-compliant. This metric considers both distribution and spread of the approximate Pareto front P with help of a uniformly distributed set of points of the true Pareto front P^*. Distribution describes how evenly scattered the solutions of P are in the objective space, while the spread describes how close the solutions in P are to the extreme points in P^*. The smaller Δ^*, the better since this indicates a solution set that is close to the extrema in P^* and well-distributed. The Δ^* indicator can be defined as follows [69]:

$$\Delta^*(P, P^*) = \frac{\sum_{i=1}^{m} d(\mathbf{e}_i, P) + \sum_{i=1}^{|P|} |d_i - \bar{d}|}{\sum_{i=1}^{m} d(\mathbf{e}_i, P) + (|P|)\bar{d}} \tag{39}$$

where $d(\mathbf{e}_i, P) = \min_{\mathbf{v}\in P} \|\mathbf{e}_i - \mathbf{v}\|$ is the minimum distance from P to the extreme solutions in P^*, $\mathbf{e}_i \in P^*$ is the extreme solution of the i[th] objective in P^*, $d_i = \min_{\mathbf{v}_j\in P, \mathbf{v}_i \neq \mathbf{v}_j} \|\mathbf{v}_i - \mathbf{v}_j\|$ defines the closest pairwise distances in P and $\bar{d} = \frac{1}{|P|}\sum_{i=1}^{|P|} d_i$ is the average of d_i.

8.2.4 Worst attainment summary surfaces

Empirical attainment functions were also plotted, which give detailed information on where and how the performance of the algorithms differs. We will give a short outline of attainment surfaces here, for more information please refer to Fonseca, Fleming [70], Knowles [19] and Knowles [71]. Attainment surfaces are uniquely defined by a set of Pareto points and divide the objective space into a region dominated by this set and a region not dominated by it. An example of an attainment surface is shown on the left-hand side of Fig. 5 defined by a set of Pareto points. In our case the sets defining the attainment surfaces are the approximate Pareto fronts from the various algorithm runs. Subsequently, each algorithm run gives us an attainment surface, hence we have *20* attainment surfaces for each constellation of algorithm and test-function. The information of these surfaces can be concisely summarised by so-called "attainment summary surfaces", which lie on or between attainment surfaces. From this one can for example define a best, median and worst summary surface. The median summary attainment surface is defined, such that every point on it is weakly dominated in at least 50% of the function runs, while the worst summary attainment surface has the interpretation that every algorithm run weakly dominated the entire surface. Plots of this kind therefore give more information than a scatter plot of non-dominated points from several runs. In Figure 5 on the right-hand side we have schematically plotted *5* attainment surfaces. The summary surfaces can be defined by imagining a diagonal line in the direction of increasing objective values cutting through the *5* surfaces. For illustration *6* diagonal lines are shown. The points on the diagonal lines are part of the worst summary attainment surface, which are weakly dominated by all the other attainment surfaces. By using many of these diagonal sampling lines, we could then graph the full worst summary attainment surface, which concisely represents information of all attainment surfaces. In this paper, the worst summary attainment surfaces were plotted for each test problem for all algorithms since these give a good indication on the weaknesses of the algorithms, i.e. the surfaces shown in section 8.6 are such that all 20 algorithm runs weakly dominated them. For the two-dimensional test problems, these can be given in a single plot for each test function, while for the three-dimensional test problems a separate plot is required for each worst summary attainment surface for visualisation.

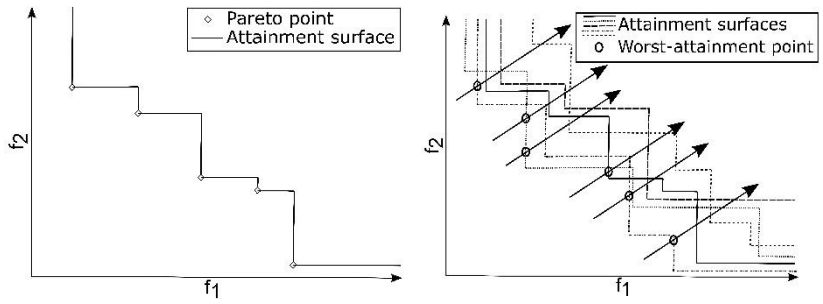

Fig. 5 Illustration of an attainment surface on the left-hand side and on the right-hand side a schematic showing how to plot the worst-attainment surface from 5 attainment surfaces.

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

8.3 Results of hypervolume quality indicator

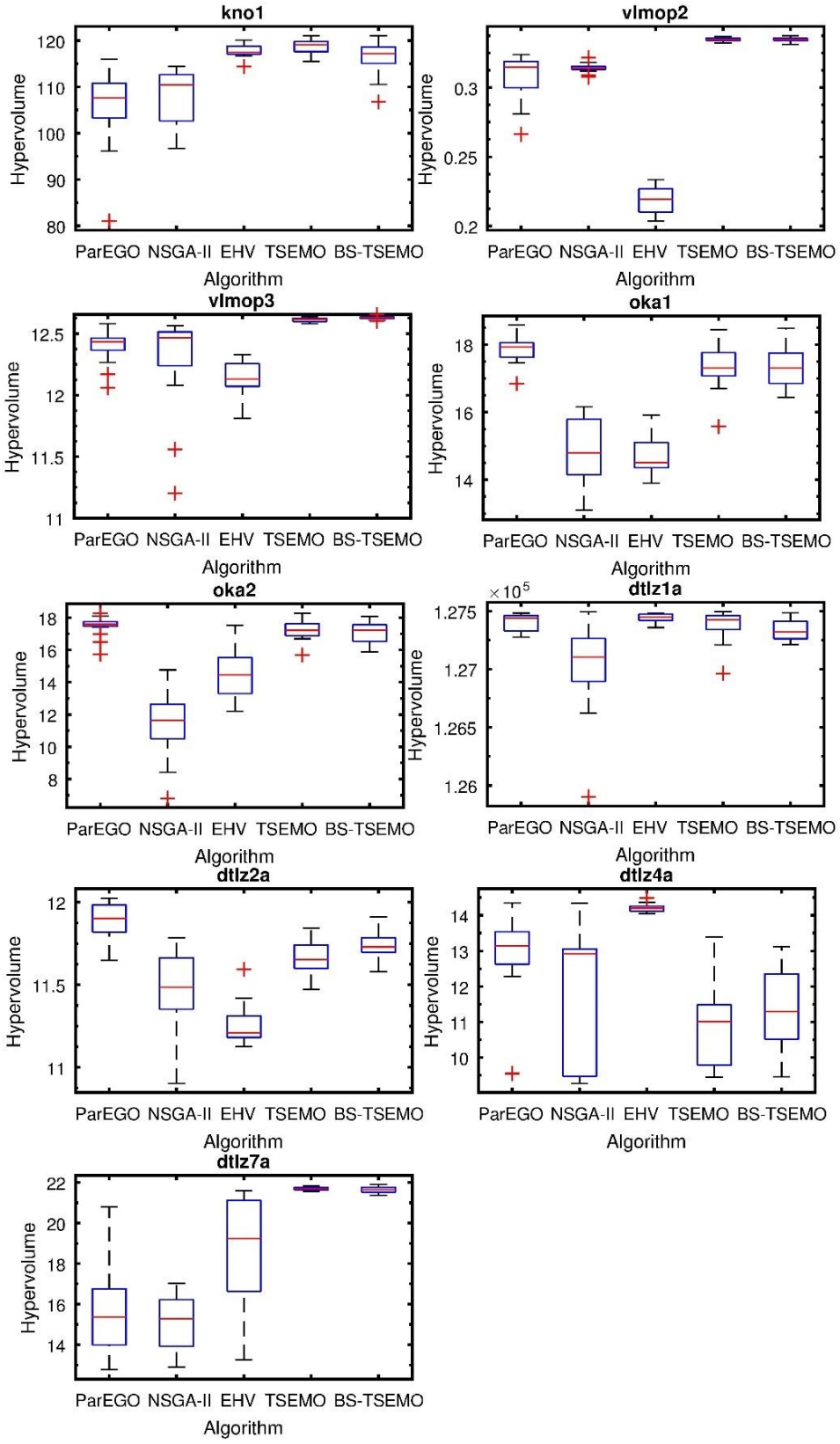

Fig. 6 Optimization test problem results shown in boxplots using the hypervolume performance indicator obtained after 20 runs with 150 function evaluations.

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

8.4 Results of inverted generational distance indicator

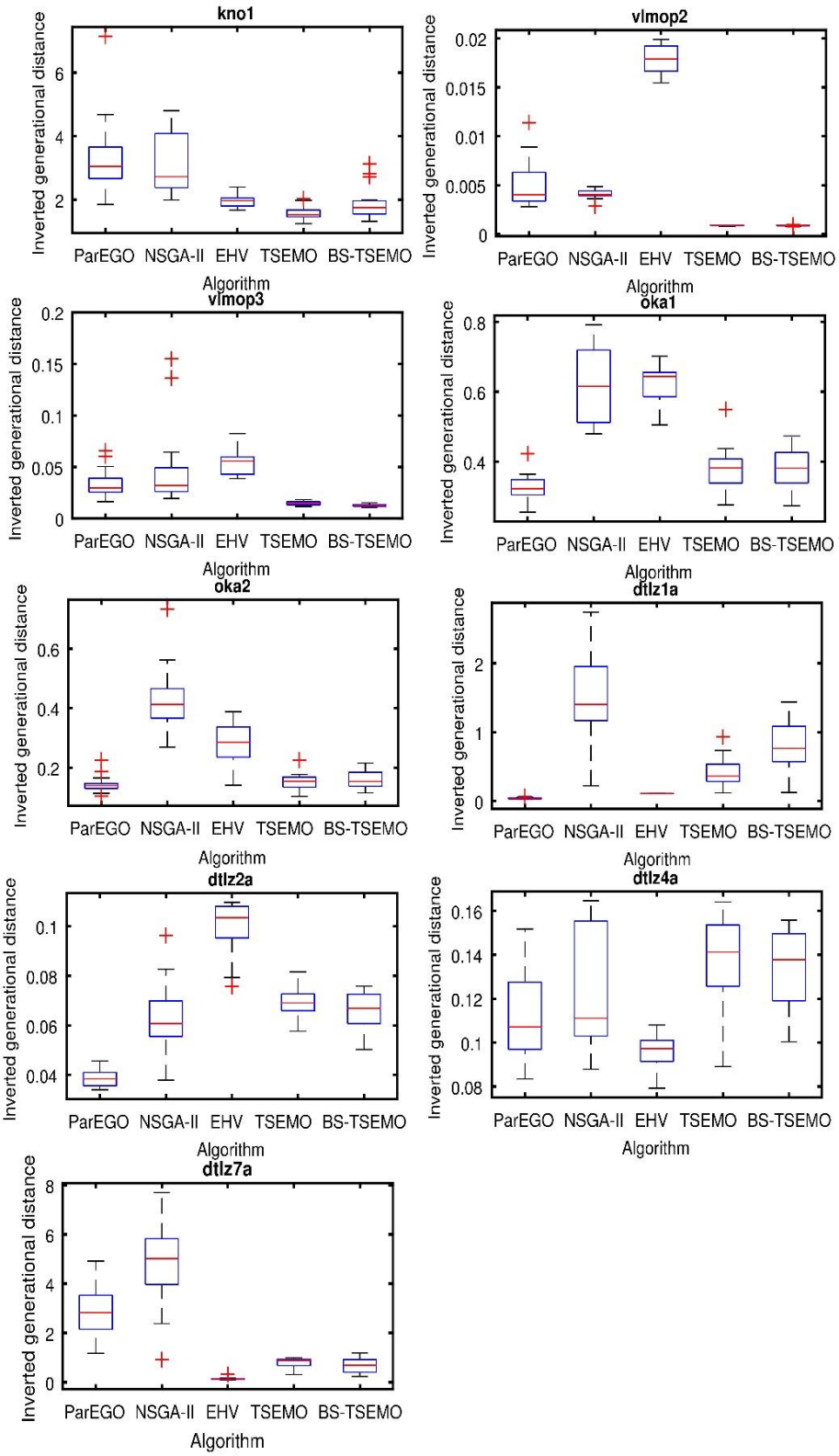

Fig. 7 Optimization test problem results shown in boxplots using the modified inverted generational distance performance indicator obtained after 20 runs with 150 function evaluations.

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

8.5 Results of generalized spread indicator

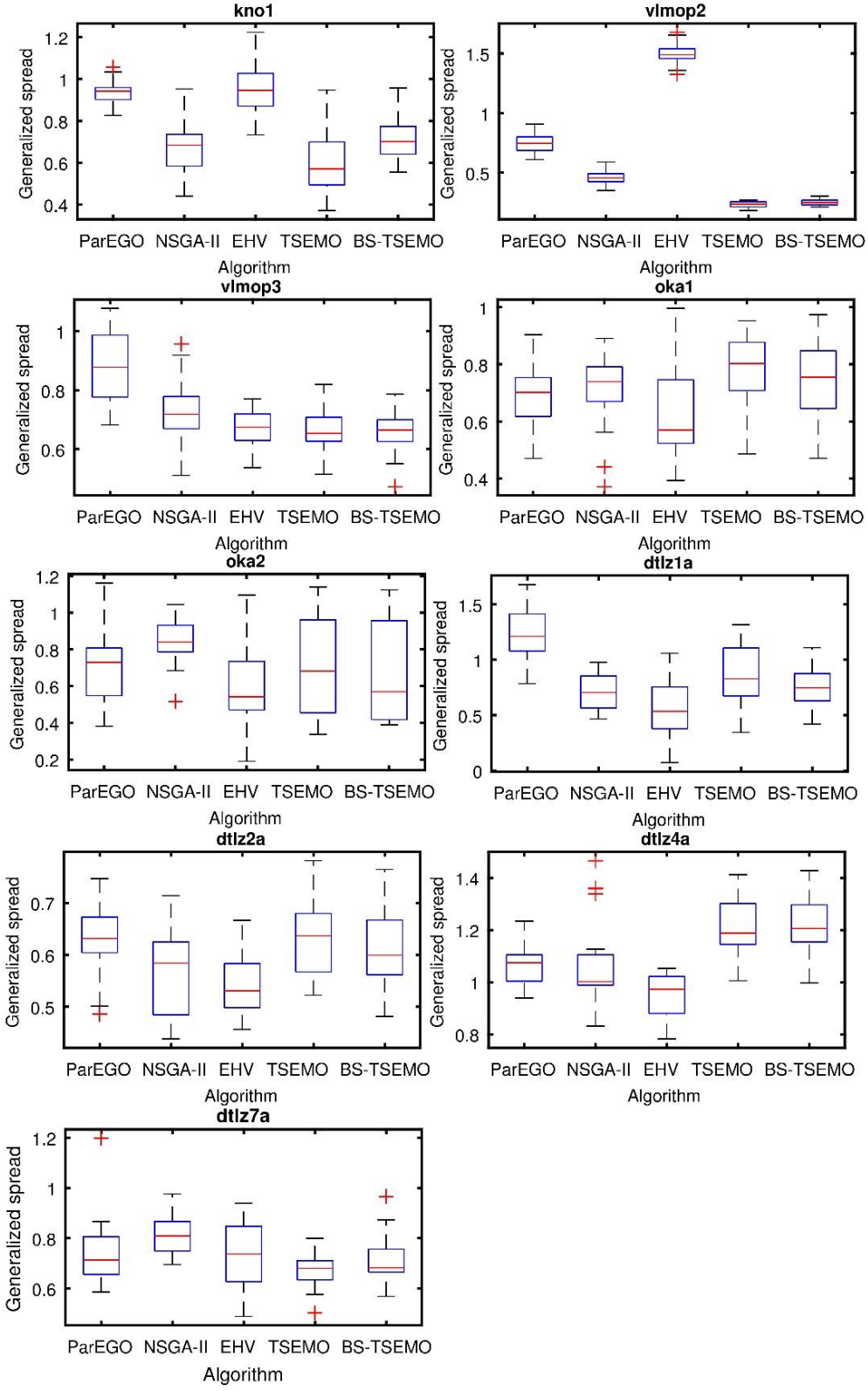

Fig. 8 Optimization test problem results shown in boxplots using the generalized spread performance indicator obtained after 20 runs with 150 function evaluations.

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

8.6 Worst attainment surfaces

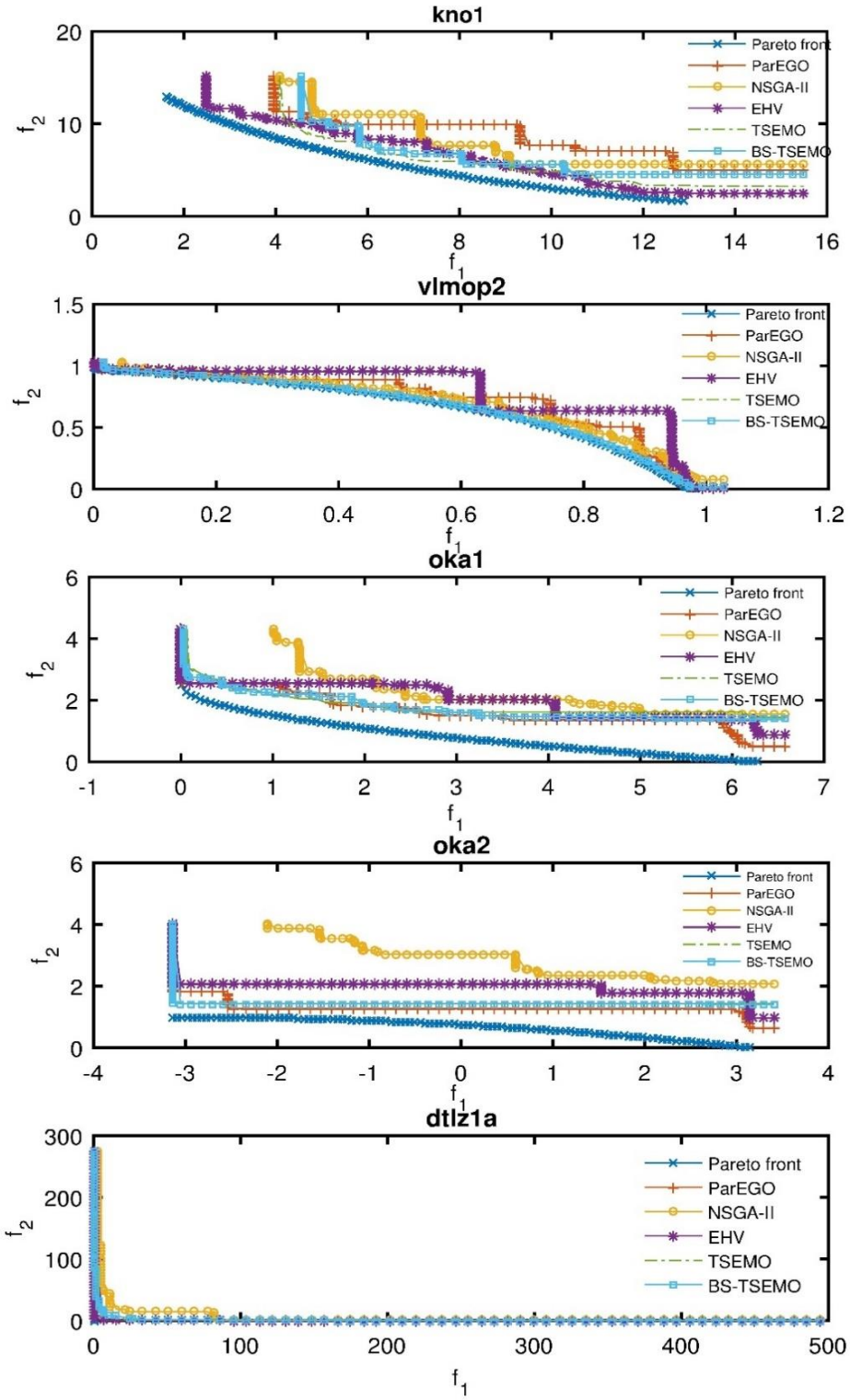

Fig. 9 Worst-attainment plots of algorithms ParEGO, NSGA-II, EHV, TS-EMO and BS-TSEMO for 2-D objective functions: **KNO1, VLMOP2, OKA1, OKA2, DTLZ1a.**

This is a post-peer-review, pre-copyedit version of an article published in Journal of Global Optimization (JOGO). The final authenticated version is available online at: http://dx.doi.org/10.1007/s10898-018-0609-2.

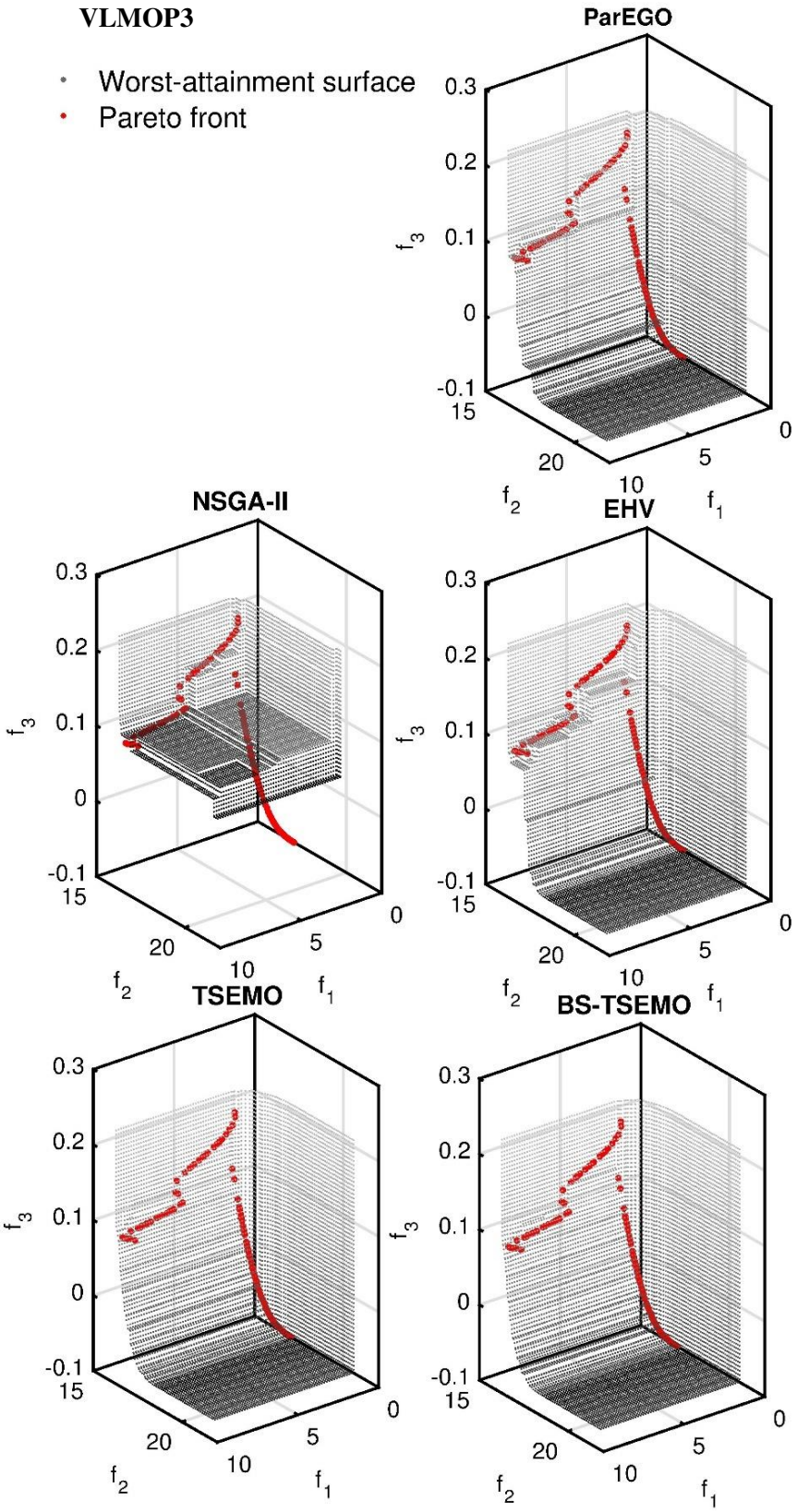

Fig. 10 Worst-attainment surface plots for the test problem **VLMOP3** of algorithms ParEGO, NSGA-II, TSEMO and BS-TSEMO each in separate graphs. The true Pareto front is shown in red and the black dots represent the worst-attainment surface.

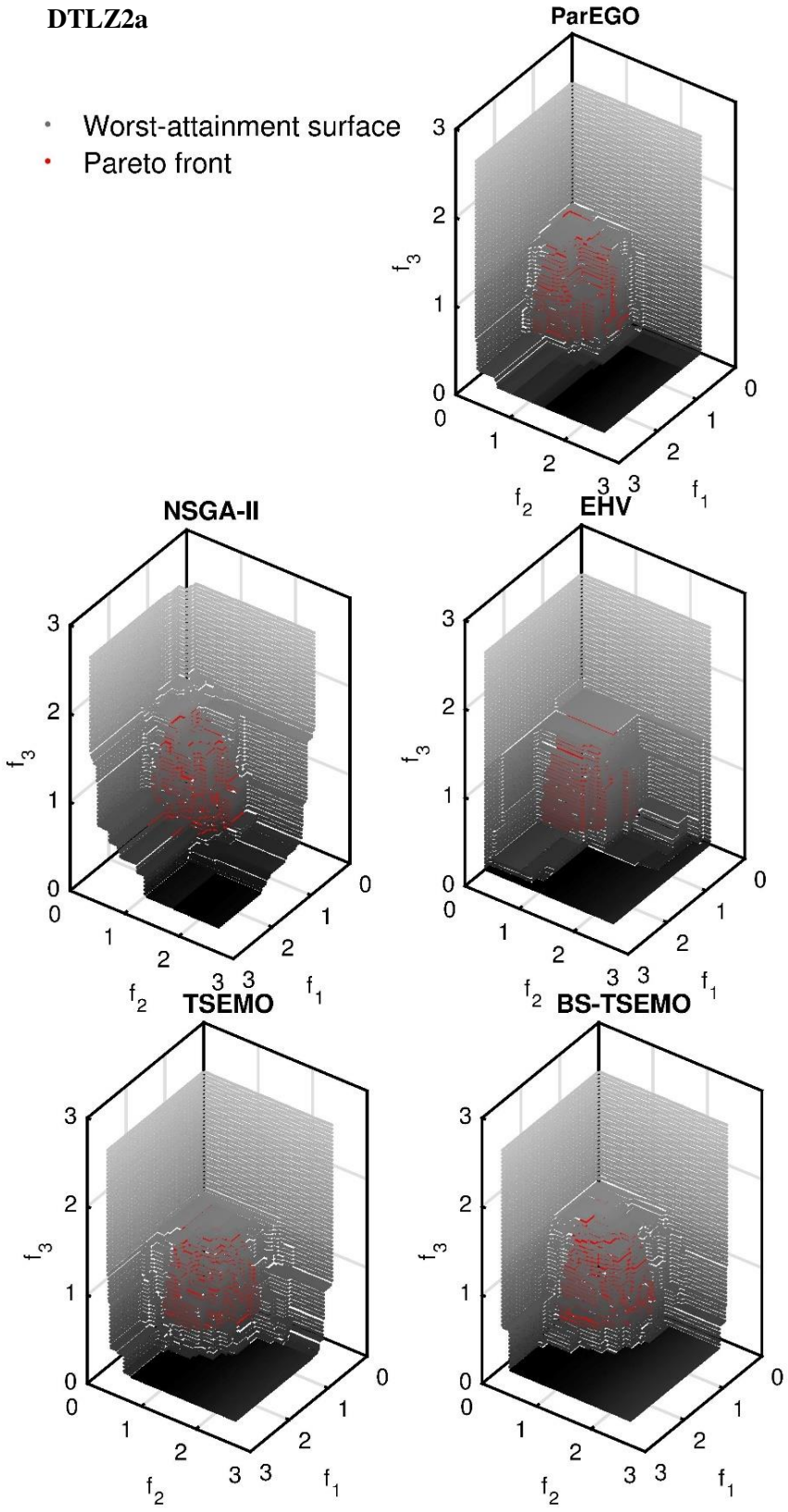

Fig. 11 Worst-attainment surface plots for the test problem **DTLZ2a** of algorithms ParEGO, NSGA-II, TSEMO and BS-TSEMO each in separate graphs. The true Pareto front is shown in red and the black dots represent the worst-attainment surface.

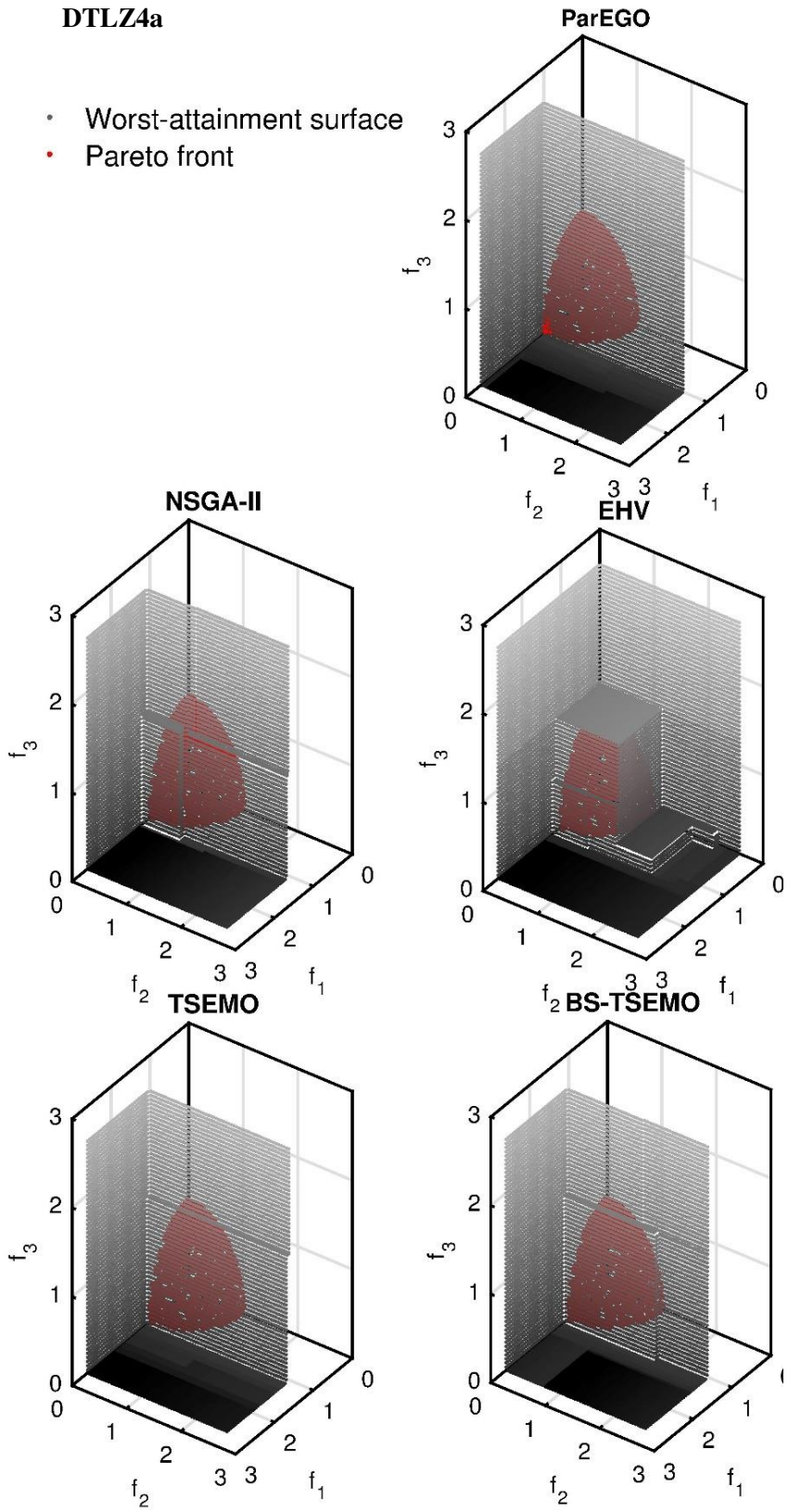

Fig. 12 Worst-attainment surface plots for the test problem **DTLZ4a** of algorithms ParEGO, NSGA-II, TSEMO and BS-TSEMO each in separate graphs. The true Pareto front is shown in red and the black dots represent the worst-attainment surface.

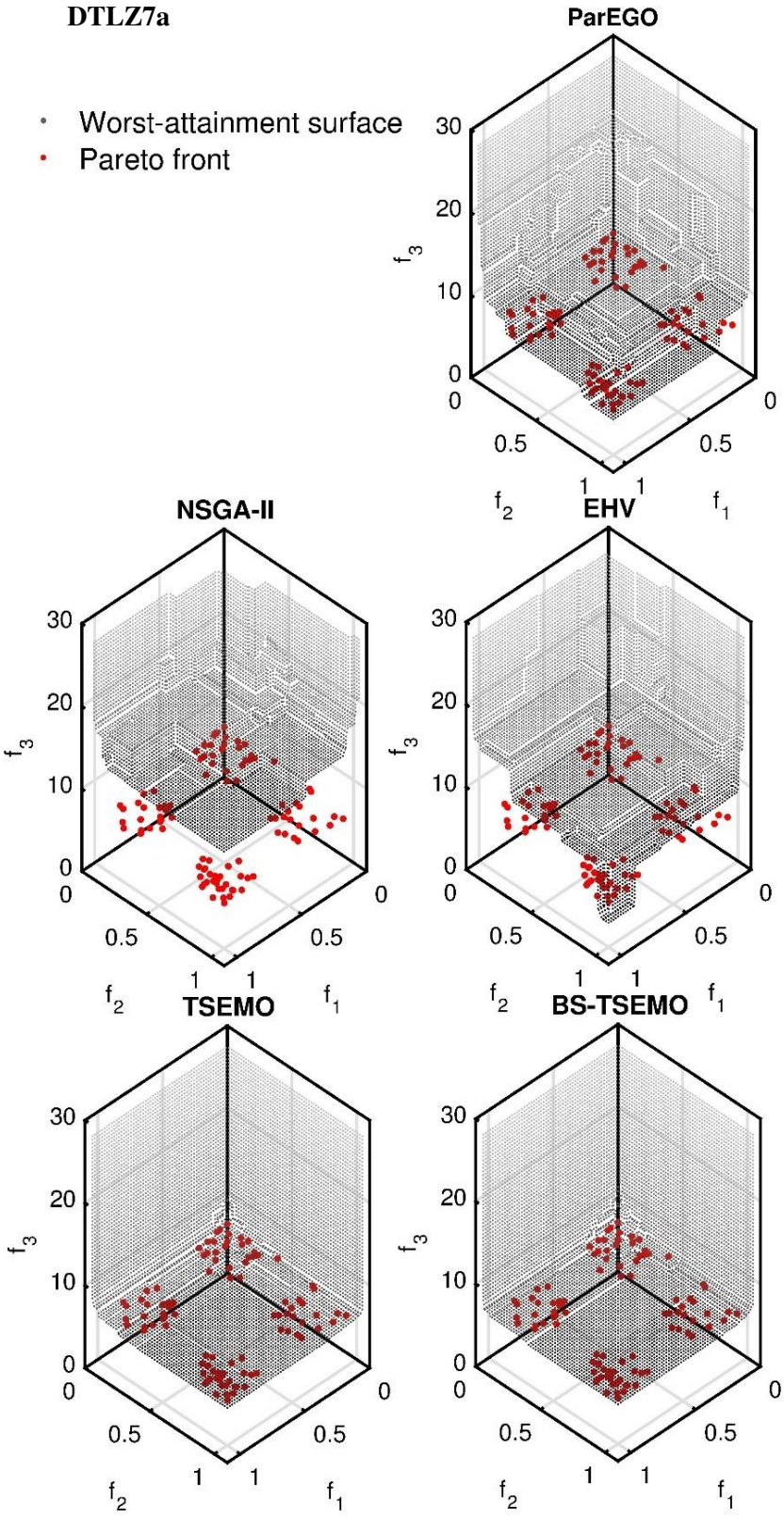

Fig. 13 Worst-attainment surface plots for the test problem **DTLZ7a** of algorithms ParEGO, NSGA-II, TSEMO and BS-TSEMO each in separate graphs. The true Pareto front is shown in red and the black dots represent the worst-attainment surface.

8.7 Discussion of results

Based on the results given in sections 8.3, 8.4, 8.5 and 8.6 the following observations can be made:

- Overall, the worst-attainment plots confirm the results of the hypervolume indicator.
- The worst summary attainment surfaces in three dimensions take a box-like structure if they consist of only a small number of Pareto points, e.g. DTLZ4a for EHV. This is because by definition the objective space is divided into a space weakly dominated by these points and a region not weakly dominated by these points. For a single Pareto point these regions are given by boxes and for a small number of Pareto points box-like regions. For others the worst summary attainment surface consists of many points and takes a complex structure as in DTLZ2a.
- TSEMO outperforms the other 3 algorithms on 4 of the 9 test problems: KNO1, VLMOP2, VLMOP3 and DTLZa7 based on both median and mean of the hypervolume. Further, the standard deviation is also orders of magnitude lower in VLMOP2, VLMOP3 and DTLZa7 compared to the other algorithms, while for KNO1 it is slightly higher than EHV.
- TSEMO comes close second on the test functions OKA1, OKA2 and DTLZ2a based on both median and mean of the hypervolume, while ParEGO attains higher hypervolume values. This may be in part due to the additional information ParEGO has available since it needs to be supplied with accurate knowledge of the limits of the outputs, while TSEMO reference point accuracy is linked to the accuracy of the GPs through Equation 33. In OKA1 and OKA2 in Figure 9 one can see that ParEGO finds Pareto points on the right side from the scaling of the supplied limits which TSEMO is unaware of.
- The modified IGD indicator agrees with the results of the hypervolume indicator nearly exactly on the test-functions KNO1, VLMOP2, VLMOP3, OKA1, OKA2 and DTLZ4a. For DTLZ1a it does allow for further differentiation and shows that EHV and ParEGO perform the best. For DTLZ2a NSGA-II is now seen to outperform TSEMO, however it does show larger variation. Lastly, for DTLZ7a the indicators are in agreeance that TSEMO and EHV are the highest performing algorithms, however EHV is shown to perform slightly better. Overall according to the median of the modified IGD, TSEMO performs the best on KNO1, VLMOP2 and VLMOP3 test-functions and comes close second on OKA1, OKA2 and DTLZ7a.
- The generalized spread indicator gives us further insight into the diversity of the approximate Pareto fronts returned from the algorithms. For the test-functions KNO1, VLMOP2,

VLMOP3 and DTLZ7, TSEMO has the highest median for the generalized spread, which is most likely due to a good identification of the underlying function and hence good placement of Pareto points according to the hypervolume improvement. These results agree with the hypervolume performance indicator. For OKA1 and OKA2 the algorithms seem to have nearly the same diversity of solutions, which may indicate that for these functions diversity of solutions is not important to obtain a good hypervolume value. For DTLZ1a the generalized spread is worse for ParEGO, while it is the highest performing algorithm according to the other indicators. For DTLZ2a and DTLZ4a EHV does particularly well. Overall the generalized spread more or less agrees with the hypervolume indicator showing that TSEMO is able to identify the extrema of the true Pareto front and find a well spread out Pareto front on several test functions.

- TSEMO performs poorly on DTLZ4a. This can be explained by looking at the three competing functions in Knowles [19], each of which is modelled by an independent GP using Matérn-class covariance functions. These covariance functions assume that the underlying function to be fitted by the GP is stationary. In this case two of the three functions, f_2 and f_3, are highly non-stationary and, hence, the GPs yield very poor predictions. EHV and ParEGO achieve higher hypervolume values due to the reference points, which limits the sampling area, while TSEMO's accuracy of the reference is linked to the accuracy of the GPs by Equation 33.
- For KNO1 one can see in Figure 9 that ParEGO struggles to find more than 6 Pareto points, which is the result of a large drawback from the scalarization carried out, i.e. the limited number of scalarizations leads to the same Pareto points repeatedly. Similar behaviour can be observed in VLMOP2.
- NSGA-II often has the worst-attainment surface such as for the test problems KNO1, OKA1 OKA2, DTLZ1a, VLMOP3, DTLZ4a and DTLZ7a. Therefore, it can be broadly said that the surrogate-based optimization algorithms are more robust.
- Apart from DTLZ4a, TSEMO's worst-attainment surfaces show relatively good approximations of Pareto fronts and hence show that the algorithm is robust.
- BS-TSEMO performance is similar to that of TSEMO. It performs well on algorithms that TSEMO performs well on and performs poorly on algorithms that TSEMO performs poorly on. It gives marginally worse results on the test problems KNO1, OKA1 and DTLZ7a, and slightly better result on DTLZ4a, however still worse than the other algorithms.

9 Conclusions

In conclusion, a new algorithm, TSEMO, has been proposed based on individual GPs for each objective, which are then sampled using spectral sampling. This then leads to individual functions from which an approximate Pareto set, and Pareto front can be found using the NSGA-II algorithm. According to a TS scheme a point is then selected from the Pareto set as the next sampling point, which is predicted to give the largest hypervolume. TSEMO was compared to NSGA-II, ParEGO and an EHV implementation utilizing 9 test problems with a limited budget of 150 function evaluations. It was found to outperform the aforementioned algorithms on 4 out of 9 test problems and came close second on 3 out of 9 test problems being outperformed by ParEGO according to the hypervolume criterion. This is thought to be because of the additional information available to ParEGO through the scalarization factors. Further, on 1 test problem all algorithms performed similarly. Based on the modified inverted generational distance TSEMO was determined to outperform 3 out of 9 test problems and came close second on 3 out of 9 test problems. In addition, the generalized spread indicator agreed with the results of the hypervolume criterion and hence showed that TSEMO is able to find a well-spread out Pareto front and the extrema of the true Pareto front on several test functions. TSEMO only performed poorly on 1 test problem, which was pointed out to be highly non-stationary and hence could not be captured by the stationary GPs used in the algorithm. Lastly, a simple heuristic was proposed for a batch-sequential design scheme. This heuristic was tested on the same test problems with a budget of 150 function evaluations adding 4 points each iteration. The comparison showed that this implementation performed very similarly to the nominal TSEMO algorithm. Overall, TSEMO was shown to compare competitively to the state-of-the-art algorithms available in the field with the added advantage that no *a priori* knowledge is required on the scale of the outputs and further can be used for batch-sequential design and for noisy functions.

10 References

1. Farmani, R., Savic, D., Walters, G.: Evolutionary multi-objective optimization in water distribution network design. Engineering Optimization **37**(2), 167-183 (2005).
2. Censor, Y.: Pareto optimality in multiobjective problems. Applied Mathematics and Optimization **4**(1), 41-59 (1977).
3. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE transactions on Evolutionary Computation **3**(4), 257-271 (1999).
4. Ehrgott, M.: Multiobjective optimization. AI Magazine **29**(4), 47 (2009).
5. Hwang, C.-R.: Simulated annealing: theory and applications. Acta Applicandae Mathematicae **12**(1), 108-111 (1988).
6. Davis, L.: Handbook of genetic algorithms. (1991).

7. Kennedy, J.: Particle swarm optimization. In: Encyclopedia of machine learning. pp. 760-766. Springer, (2011)
8. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical science **4**(4), 409-423 (1989).
9. Simpson, T.W., Booker, A.J., Ghosh, D., Giunta, A.A., Koch, P.N., Yang, R.-J.: Approximation methods in multidisciplinary analysis and optimization: a panel discussion. Structural and multidisciplinary optimization **27**(5), 302-313 (2004).
10. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **42**(1), 55-61 (2000).
11. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., de Freitas, N.: Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE **104**(1), 148-175 (2016).
12. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global optimization **13**(4), 455-492 (1998).
13. Močkus, J.: On Bayesian methods for seeking the extremum. In: Optimization Techniques IFIP Technical Conference 1975, pp. 400-404. Springer
14. Łaniewski-Wołłk, Ł.: Relative Expected Improvement in Kriging Based Optimization. arXiv preprint arXiv:0908.3321 (2009).
15. Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. Journal of global optimization **21**(4), 345-383 (2001).
16. Forrester, A.I., Keane, A.J.: Recent advances in surrogate-based optimization. Progress in Aerospace Sciences **45**(1), 50-79 (2009).
17. Voutchkov, I., Keane, A.: Multi-objective optimization using surrogates. In: Computational Intelligence in Optimization. pp. 155-175. Springer, (2010)
18. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation **6**(2), 182-197 (2002).
19. 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 (2006).
20. Ponweiser, W., Wagner, T., Biermann, D., Vincze, M.: Multiobjective optimization on a limited budget of evaluations using model-assisted\ mathcal {S}-Metric selection. In: International Conference on Parallel Problem Solving from Nature 2008, pp. 784-794. Springer
21. Kushner, H.J.: A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. Journal of Basic Engineering **86**(1), 97-106 (1964).
22. Keane, A.J.: Statistical improvement criteria for use in multiobjective design optimization. AIAA journal **44**(4), 879-891 (2006).

23. Emmerich, M., Klinkenberg, J.-W.: The computation of the expected improvement in dominated hypervolume of Pareto front approximations. Rapport technique, Leiden University (2008).
24. Emmerich, M.: Single-and multi-objective evolutionary design optimization assisted by gaussian random field metamodels. PhD thesis, University of Dortmund (2005)
25. Emmerich, M., Deutz, A., Klinkenberg, J.-W.: Hypervolume-based expected improvement: Monotonicity properties and exact computation. In: IEEE Congress on Evolutionary Computation 2011, pp. 2147-2154. IEEE
26. Hupkens, I., Emmerich, M., Deutz, A.: Faster Computation of Expected Hypervolume Improvement. arXiv preprint arXiv:1408.7114 (2014).
27. Emmerich, M., Yang, K., Deutz, A., Wang, H., Fonseca, C.M.: A multicriteria generalization of bayesian global optimization. In: Advances in Stochastic and Deterministic Global Optimization. pp. 229-242. Springer, (2016)
28. Yang, K., Emmerich, M., Deutz, A., Fonseca, C.M.: Computing 3-D Expected Hypervolume Improvement and Related Integrals in Asymptotically Optimal Time. In: International Conference on Evolutionary Multi-Criterion Optimization 2017, pp. 685-700. Springer
29. Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika **25**(3/4), 285-294 (1933).
30. Zhigljavsky, A., Zilinskas, A.: Stochastic global optimization, vol. 9. Springer Science & Business Media, (2007)
31. Žilinskas, A.: A statistical model-based algorithm for 'black-box'multi-objective optimisation. International Journal of Systems Science **45**(1), 82-93 (2014).
32. Helmdach, D., Yaseneva, P., Heer, P.K., Schweidtmann, A.M., Lapkin, A.: A multi-objective optimisation including results of life cycle assessment in developing bio-renewables-based processes. ChemSusChem (2017).
33. Rasmussen, C.E.: Gaussian processes in machine learning. In: Bousquet, O., VonLuxburg, U., Ratsch, G. (eds.) Advanced Lectures on Machine Learning, vol. 3176. Lecture Notes in Artificial Intelligence, pp. 63-71. (2004)
34. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press, (2005)
35. Ebden, M.: Gaussian processes: A quick introduction. arXiv preprint arXiv:1505.02965 (2015).
36. Rasmussen, C.E.: Gaussian processes for machine learning. (2006).
37. Yang, X., Maciejowski, J.M.: Fault tolerant control using Gaussian processes and model predictive control. International Journal of Applied Mathematics and Computer Science **25**(1), 133-148 (2015).
38. Matérn, B.: Spatial variation, vol. 36. Springer Science & Business Media, (2013)
39. Sundararajan, S., Keerthi, S.S.: Predictive approaches for choosing hyperparameters in Gaussian processes. Neural Computation **13**(5), 1103-1118 (2001).

40. Hernández-Lobato, J.M., Hoffman, M.W., Ghahramani, Z.: Predictive entropy search for efficient global optimization of black-box functions. In: Advances in neural information processing systems 2014, pp. 918-926
41. Bochner, S.: Lectures on Fourier Integrals, vol. 42. Princeton University Press, (1959)
42. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: Advances in neural information processing systems 2007, pp. 1177-1184
43. Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C.E., Figueiras-Vidal, A.R.: Sparse spectrum Gaussian process regression. The Journal of Machine Learning Research **11**(6), 1865-1881 (2010).
44. Chapelle, O., Li, L.: An empirical evaluation of thompson sampling. In: Advances in neural information processing systems 2011, pp. 2249-2257
45. Scott, S.L.: A modern Bayesian look at the multi-armed bandit. Applied Stochastic Models in Business and Industry **26**(6), 639-658 (2010).
46. Shahriari, B., Wang, Z., Hoffman, M.W., Bouchard-Côté, A., de Freitas, N.: An entropy search portfolio for bayesian optimization. arXiv preprint arXiv:1406.4625 (2014).
47. Kaufmann, E., Korda, N., Munos, R.: Thompson sampling: An asymptotically optimal finite-time analysis. In: International Conference on Algorithmic Learning Theory 2012, pp. 199-213. Springer
48. Agrawal, S., Goyal, N.: Thompson Sampling for Contextual Bandits with Linear Payoffs. In: 30th International Conference on Machine Learning (ICML-13) 2013, pp. 127-135
49. Yahyaa, S., Manderick, B.: Thompson sampling for multi-objective multi-armed bandits problem. In: 2015 European Symposium on Artificial Neural Networks 2015, pp. 47-52. Presses universitaires de Louvain
50. Stein, M.: Large sample properties of simulations using Latin hypercube sampling. Technometrics **29**(2), 143-151 (1987).
51. Emmerich, M., Fonseca, C.: Computing hypervolume contributions in low dimensions: Asymptotically optimal algorithm and complexity results. In: Evolutionary Multi-Criterion Optimization 2011, pp. 121-135. Springer
52. Bader, J., Deb, K., Zitzler, E.: Faster hypervolume-based search using Monte Carlo sampling. In: Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems: Proceedings of the 19th International Conference on Multiple Criteria Decision Making. pp. 313-326. Springer Berlin Heidelberg, (2010)
53. Finkel, D.E.: DIRECT optimization algorithm user guide. http://www4.ncsu.edu/~ctk/Finkel_Direct/DirectUserGuide_pdf.pdf (2003). Accessed 26/11/2017
54. Lin, S.: NGPM a NSGA-II Program in Matlab. Codelooker. http://www.codelooker.com/dfilec/6987NGPMv11.4/NGPMmanualv1.4.pdf (2011). Accessed 26/11/2017

55. Rasmussen, C.E., Nickisch, H.: The GPML Toolbox version 3.5. http://mlg.eng.cam.ac.uk/carl/gpml/doc/manual.pdf (2014). Accessed 26/11/2017
56. Yi, C.: Pareto Set. Matlab file exchange. https://se.mathworks.com/matlabcentral/fileexchange/15181-pareto-set (2014). Accessed 26/11/2017
57. Yi, C.: Hypervolume Indicator. Matlab file exchange. https://se.mathworks.com/matlabcentral/fileexchange/19651-hypervolume-indicator (2008). Accessed 26/11/2017
58. Bradford, E., Schweidtmann, A.M.: TS-EMO. GitHub. https://github.com/Eric-Bradford/TS-EMO (2017). Accessed 29/11/2017
59. Schaffer, J.D.: Some experiments in machine learning using vector evaluated genetic algorithms. Vanderbilt University, Nashville, United States (1985)
60. Cristescu, C.: ParEGO_Eigen. GitHub. https://github.com/CristinaCristescu/ParEGO_Eigen/graphs/contributors (2015). Accessed 26/11/2017
61. Gorissen, D., Couckuyt, I., Demeester, P., Dhaene, T., Crombecq, K.: A surrogate modeling and adaptive sampling toolbox for computer based design. Journal of Machine Learning Research **11**(Jul), 2051-2055 (2010).
62. Cristescu, C., Knowles, J.: Surrogate-Based Multiobjective Optimization: ParEGO Update and Test.
63. Riquelme, N., Von Lücken, C., Baran, B.: Performance metrics in multi-objective optimization. In: Computing Conference (CLEI), 2015 Latin American 2015, pp. 1-11. IEEE
64. 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).
65. Knowles, J., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers. Tik report **214**, 327-332 (2006).
66. Ishibuchi, H., Masuda, H., Tanigaki, Y., Nojima, Y.: Modified Distance Calculation in Generational Distance and Inverted Generational Distance. In: 2015
67. Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., Tsang, E.: Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, pp. 892-899. IEEE
68. Zitzler, E.: Evolutionary algorithms for multiobjective optimization: Methods and applications. (1999).
69. Jiang, S., Ong, Y.-S., Zhang, J., Feng, L.: Consistencies and contradictions of performance metrics in multiobjective optimization. IEEE Transactions on Cybernetics **44**(12), 2391-2404 (2014).
70. Fonseca, C.M., Fleming, P.J.: On the performance assessment and comparison of stochastic multiobjective optimizers. In: International Conference on Parallel Problem Solving from Nature 1996, pp. 584-593. Springer
71. Knowles, J.: A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective

optimizers. In: 5th International Conference on Intelligent Systems Design and Applications, 2005 2005, pp. 552-557. IEEE

Acknowledgments

The research leading to these results has received funding from the European Research Council under the European Union's H2020 Programme Grant Agreement no. 636820. Artur M. Schweidtmann thanks the Ernest-Solvay-Foundation and the ERASMUS+ program for a scholarship for his exchange at the University of Cambridge.