Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

Treed Gaussian Process Regression for Solving Offline Data-Driven Continuous Multiobjective Optimization Problems

Atanu Mazumdar atanu.a.mazumdar@jyu.fi
University of Jyvaskyla, Faculty of Information Technology, Finland

Manuel López-Ibáñez manuel.lopez-ibanez@manchester.ac.uk
Alliance Manchester Business School, University of Manchester, UK

Tinkle Chugh t.chugh@exeter.ac.uk
Department of Computer Science, University of Exeter, UK

Jussi Hakanen jussi.hakanen@jyu.fi
University of Jyvaskyla, Faculty of Information Technology, Finland

Kaisa Miettinen kaisa.miettinen@jyu.fi
University of Jyvaskyla, Faculty of Information Technology, Finland

https://doi.org/10.1162/evco_a_00329

Abstract

For offline data-driven multiobjective optimization problems (MOPs), no new data is available during the optimization process. Approximation models (or surrogates) are first built using the provided offline data, and an optimizer, for example, a multiobjective evolutionary algorithm, can then be utilized to find Pareto optimal solutions to the problem with surrogates as objective functions. In contrast to online data-driven MOPs, these surrogates cannot be updated with new data and, hence, the approximation accuracy cannot be improved by considering new data during the optimization process. Gaussian process regression (GPR) models are widely used as surrogates because of their ability to provide uncertainty information. However, building GPRs becomes computationally expensive when the size of the dataset is large. Using sparse GPRs reduces the computational cost of building the surrogates. However, sparse GPRs are not tailored to solve offline data-driven MOPs, where good accuracy of the surrogates is needed near Pareto optimal solutions. Treed GPR (TGPR-MO) surrogates for offline data-driven MOPs with continuous decision variables are proposed in this paper. The proposed surrogates first split the decision space into subregions using regression trees and build GPRs sequentially in regions close to Pareto optimal solutions in the decision space to accurately approximate tradeoffs between the objective functions. TGPR-MO surrogates are computationally inexpensive because GPRs are built only in a smaller region of the decision space utilizing a subset of the data. The TGPR-MO surrogates were tested on distance-based visualizable problems with various data sizes, sampling strategies, numbers of objective functions, and decision variables. Experimental results showed that the TGPR-MO surrogates are computationally cheaper and can handle datasets of large size. Furthermore, TGPR-MO surrogates produced solutions closer to Pareto optimal solutions compared to full GPRs and sparse GPRs.

Keywords

Gaussian processes, Kriging, regression trees, metamodelling, surrogate, Pareto optimality.

Manuscript received: 13 April 2022; revised: 6 November 2022, 23 February 2023, 23 March 2023; accepted: 31 March 2023.

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

1 Introduction

A multiobjective optimization problem (MOP) consists of two or more conflicting objective functions that are optimized simultaneously. The solutions of an MOP are called Pareto optimal when the value of any objective function cannot be further improved without degrading some of the others; thus, tradeoffs exist among the objective functions. Not all real-world MOPs have analytical functions or simulation models available. Instead, the MOP may need to be formulated by utilizing the data acquired from a phenomenon, for example, real-life processes, physical experiments, and sensors. To solve this MOP, first the *underlying objective functions* are approximated by fitting surrogates (also known as metamodels) on the data available. Next, a multiobjective evolutionary algorithm (MOEA) can be used to approximate the set of Pareto optimal solutions by considering the surrogates as objective functions.

Data-driven optimization problems are classified into *online* and *offline* ones (Jin et al., 2019). For online data-driven optimization, new data can be acquired during the optimization process, for example, by conducting further experiments or simulations to update the surrogates (Shahriari et al., 2016). In contrast, for offline data-driven optimization problems, no new data can be acquired, and the optimization has to proceed by using only the existing data (Wang et al., 2019). Ideally, the underlying objective values of the solutions should be better than the objective values in the provided dataset. All the available data should be utilized to build surrogates with good approximation while solving offline data-driven MOPs. However, when the size of the data set is large, for example, in Wang et al. (2016) and Wang and Jin (2020), building certain surrogates, such as Gaussian processes, can become computationally expensive. Because the surrogates cannot be updated with new data while solving offline data-driven MOPs, the approximation accuracy of the surrogates directly influences the closeness of the solutions to the Pareto optimal set. Most of the previous works on offline data-driven optimization, such as Wang et al. (2019, 2016), Wang and Jin (2020), and Yang et al. (2020), considered different surrogate modelling techniques for solving offline data-driven optimization problems. However, these works did not consider the tradeoffs between the underlying objective functions while building the surrogates. Other developed approaches, for example, Mazumdar et al. (2019, 2022), Rahat et al. (2018), and Hughes (2001), relied on the uncertainty in the prediction of Gaussian processes to improve the quality of solutions. The previous works on offline data-driven optimization dealing with large datasets, for example, Wang et al. (2016) and Wang and Jin (2020) did not quantify the uncertainty, which is a critical component in solving offline data-driven MOPs, as shown in Mazumdar et al. (2019, 2020, 2022). This paper proposes an approach to build computationally cheap surrogates with a high approximation accuracy at the region around the Pareto optimal set. The surrogates use a combination of regression trees and Gaussian process regression models and provide uncertainty in the prediction. The proposed surrogates are tailored toward solving offline data-driven MOPs and are capable of handling large datasets (tested for a maximum size of 50,000).

To study offline data-driven MOPs, data is sampled from benchmark problems for which the Pareto optimal set is known. Knowledge of the Pareto optimal set enables quality comparisons of the solutions obtained by different optimization approaches and surrogates. While solving an offline data-driven MOP, generally two types of indicators are used to quantify the quality of the solutions (Mazumdar et al., 2019, 2020). The approximation accuracy is measured in root mean squared error (RMSE) between the approximated objective values of the surrogates and the underlying objective values. In addition, the hypervolume (Zitzler and Thiele, 1998) indicator is used to measure

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

how close the approximated Pareto front (after evaluating with the underlying objective functions) is to the Pareto front of the underlying MOP.

Kriging, or Gaussian process regression (GPR), (Forrester et al., 2008) is a popular choice of surrogate (Rasmussen and Williams, 2006) for solving offline problems because it also provides information about the uncertainty in the approximation. Previous works (Hughes, 2001; Rahat et al., 2018; Mazumdar et al., 2019, 2022) have shown that utilizing the uncertainty prediction from GPR surrogates produces solutions with improved accuracy and hypervolume. The uncertainty prediction from GPR makes it an attractive choice of surrogates for solving offline data-driven MOPs. Most of the approaches for solving offline data-driven MOPs build a global GPR surrogate for each objective using all the provided data. However, full GPRs have high computational and memory complexity when the size of the dataset (or sample size) is large. An alternative to full GPRs is to use sparse GPRs (Snelson and Ghahramani, 2005; Titsias, 2009), which build approximation models with a small subset of data called support or inducing points. In Titsias (2009), a variational method was proposed to select the inducing points by gradient descent or greedy search algorithm. However, this method becomes computationally expensive when the dataset is large. On the other hand, using fewer points for building the surrogates reduces the approximation accuracy of the surrogates compared to full GPRs (Titsias, 2009). Other works, such as Emmerich et al. (2006), proposed building GPR using only K-nearest neighbor samples from the point that is to be predicted.

While performing multiobjective optimization using surrogates, it is desirable to obtain a good approximation accuracy at the region around the Pareto set, which is referred to as the *trade-off region* in this paper. The accuracy of the surrogates in other regions of the decision space (excluding the trade-off region) is not of utmost importance. A possible approach to reduce the computational cost is to build local GPR surrogates for the underlying objective functions exclusively in the trade-off region. Using local GPRs reduces the overall computational cost of building GPRs while providing a good approximation accuracy at the trade-off region. Previous works (Kim et al., 2005; Das and Srivastava, 2010; van Stein et al., 2015) partitioned the decision space into regions and fitted GPRs in each region. Partitioning the decision space is relatively inexpensive, and each GPR utilizes smaller sets of data. This concept was extended by Gramacy and Lee (2008) to fit GPRs in each region or leaf nodes partitioned by a Bayesian treed model previously proposed by Chipman et al. (1998). In Assael et al. (2014), treed GPRs were proposed to deal with cases where the noise is different for different samples. All the previous works build GPRs in all the regions of the decision space. These types of GPRs become computationally expensive, depending on the number of regions and amount of data in each region. Moreover, these surrogates do not consider the trade-off region of the offline data-driven MOP during the building process.

This paper proposes treed GPR surrogates for multiobjective optimization (TGPR-MO) which have a high accuracy around the trade-off region and are tailored for solving offline data-driven MOPs with continuous decision variables. First, computationally inexpensive regression tree surrogates are built using the provided dataset. The regression tree surrogates provide a less accurate approximation of the underlying objective functions (compared to GPRs) (Loh, 2011) and split the decision space into regions. A region is defined as the part of the decision space that is enclosed by the leaf node of the fitted regression trees. The approximation accuracy of the prediction by the leaf node is the accuracy of the region. Next, an MOEA is executed considering the regression trees as objective functions. The solutions of the MOEA are not very accurate but provide the

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

approximate location of the trade-off region. After a certain number of generations of the MOEA, the accuracy of the trees' prediction is improved by building GPRs at leaf nodes within the trade-off region. The GPRs built improve the accuracy of the solutions in the region of the decision space corresponding to the leaf node they replace. The final surrogates consist of regression trees with GPRs at a few leaf nodes providing accurate approximations exclusively in the trade-off region.

The TGPR-MO surrogates were tested on several instances of distance-based visualizable test problems (DBMOPP) with different numbers of decision variables and objective functions. Various sizes of the initial dataset with different sampling strategies were used in the tests. Numerical experiments showed that TGPR-MO surrogates significantly reduced the building time for surrogates when using large size data while solving offline data-driven MOPs. TGPR-MO surrogates also produced solutions with a better hypervolume compared to sparse GPR surrogates.

The rest of the paper is arranged as follows. The background of offline data-driven MOPs, GPRs, and regression trees is given in Section 2. The proposed TGPR-MO surrogates are detailed in Section 3. The experimental results consisting of a comparison of TGPR-MO surrogates with other surrogates are presented in Section 4. Section 5 concludes and discusses the future research perspectives.

2 Background

For an offline data-driven MOP, the starting point for solving the problem is (precollected) data. In this paper, we assume that the available data is the output of a process or phenomenon. The process of generating the data is referred to as the *underlying* objective functions of an MOP. The underlying MOP that has to be solved is considered to be of the following form:

$$\begin{aligned} &\text{minimize} && (f_1(\mathbf{x}), \ldots, f_K(\mathbf{x})) \\ &\text{subject to} && \mathbf{x} \in \Omega, \end{aligned} \tag{1}$$

where $K \geq 2$ is the number of objective functions and Ω is the feasible region in the decision space $\Re^n$. For a feasible decision vector $\mathbf{x}$ (consisting of n decision variables), the corresponding objective vector is $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_K(\mathbf{x}))$, where $f_1(\mathbf{x}), \ldots, f_K(\mathbf{x})$ are the objective (function) values.

A solution $\mathbf{x} \in \Omega$ dominates another solution $\mathbf{x}' \in \Omega$ if $f_i(\mathbf{x}) \leq f_i(\mathbf{x}')$ for all $i = 1, \ldots, K$ and $f_i(\mathbf{x}) < f_i(\mathbf{x}')$ for at least one $i = 1, \ldots, K$. If a solution of an MOP is not dominated by any other feasible solutions, it is called nondominated. Solving an MOP using a multiobjective optimization algorithm, for example, an MOEA typically produces a set of mutually nondominated solutions. The solutions of the optimization problem defined in Equation (1) that are nondominated in the whole set Ω are called Pareto optimal solutions.

A generic way to solve an offline data-driven MOP with an MOEA as the optimizer is shown in Figure 1. As explained in Jin et al. (2019) and Wang et al. (2019), the solution process can be divided into three components: (a) data collection, (b) formulating the MOP and surrogate building, and (c) optimization with surrogates.

The optimization process starts with an offline dataset consisting of N samples. A sample consists of a decision vector $\mathbf{x}$ and its corresponding objective vector $\mathbf{f}(\mathbf{x})$ as a tuple of two matrices: (X, Y) where $X \in \Re^{N \times n}$ and $Y \in \Re^{N \times K}$. Each row in X and Y is a decision vector and its corresponding objective vector, respectively. Next, surrogates are built using all or a subset of the data. The surrogates are considered as objective

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

Figure 1: Flowchart of a generic offline data-driven multiobjective optimization approach.

functions by an MOEA to solve the offline data-driven MOP. For simplicity of describing a surrogate model for a single objective i, let $\mathbf{y}_i \in \Re^{N\times 1}$ be the vector of objective values $f_i(\mathbf{x})$ for all decision vectors $\mathbf{x} \in X$.

2.1 Gaussian Process Regression

In this work, a surrogate model is built for each objective function. For simplicity in terminologies, $\mathbf{y}$ is considered here instead of $\mathbf{y}_i$ to introduce the Gaussian process regression (GPR) and regression trees. GPRs have been widely used in surrogate-assisted optimization and time-series analysis (Jin et al., 2019; Chugh et al., 2019). The major advantage of using a GPR is its ability to provide the distribution about its prediction (or the uncertainty). A GPR is a multivariate normal distribution with a mean $\boldsymbol{\mu}$ and a covariance matrix C:

$$\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, C). \tag{2}$$

For simplicity in calculations, a mean of zero is considered without loss of generality. The covariance matrix C uses a covariance (or kernel) function to define correlation between two samples, $\mathbf{x}$ and $\mathbf{x}'$. In this work, we use the Matérn 5/2 kernel because it does not make the covariance functions unrealistically smooth compared to other kernels such as the Gaussian (or RBF) kernel. Therefore, it is more appropriate for solving practical optimisation problems (Snoek et al., 2012). The kernel is defined as:

$$\kappa(\mathbf{x}, \mathbf{x}', \boldsymbol{\Theta}) = \sigma_{\mathrm{f}}^2 \left(1 + \sqrt{5}\sum_{j=1}^{n} r_j + \frac{5}{3}\sum_{j=1}^{n} r_j^2\right) \exp\left(-\sqrt{5}\sum_{j=1}^{n} r_j\right) + \sigma_{\mathrm{t}}^2 \delta_{\mathbf{x}\mathbf{x}'}$$

where $r_j = \frac{|x_j - x_j'|}{l_j}$ and x_j and x_j' are the jth components of the decision vector of $\mathbf{x}$ and $\mathbf{x}'$, $\boldsymbol{\Theta} = (\sigma_{\mathrm{f}}, l_1, \ldots, l_n, \sigma_{\mathrm{t}})$ is the set of parameters in the GPR model and $\delta_{\mathbf{x}\mathbf{x}'}$ is the Kronecker delta function. The notation $|x_j - x_j'|$ represents the Euclidean distance between x_j and x_j'. The parameters σ_{f}, l_j and σ_{t} represent the amplitude, length scale of the jth variable, and noise in the data, respectively. For more details on the significance of these parameters, see Rasmussen and Williams (2006).

To build a GPR model, the parameters mentioned above can be estimated by maximizing the marginal likelihood function:

$$p(\mathbf{y} \mid X, \boldsymbol{\Theta}) = \frac{1}{\sqrt{|2\pi C|}} \exp\left(-\frac{1}{2}\mathbf{y}^{\top} C^{-1} \mathbf{y}\right). \tag{3}$$

After estimating the parameters, the model defined in Equation (2) can be used for the posterior predictive distribution at a new decision vector $\mathbf{x}^*$. The GPR model provides a posterior predictive distribution which is also Gaussian:

$$p\left(y^* | \mathbf{x}^*, X, \mathbf{y}, \boldsymbol{\Theta}\right) = \mathcal{N}\left(\boldsymbol{\kappa}(\mathbf{x}^*, X, \boldsymbol{\Theta}) C^{-1} \mathbf{y}, \kappa(\mathbf{x}^*, \mathbf{x}^*, \boldsymbol{\Theta}) - \boldsymbol{\kappa}(\mathbf{x}^*, X, \boldsymbol{\Theta})^{\top} C^{-1} \boldsymbol{\kappa}(X, \mathbf{x}^*, \boldsymbol{\Theta})\right). \tag{4}$$

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

The posterior mean in the equation above is $\boldsymbol{\kappa}(\mathbf{x}^*, X, \boldsymbol{\Theta})C^{-1}\mathbf{y}$, and the variance representing the uncertainty is $\kappa(\mathbf{x}^*, \mathbf{x}^*, \boldsymbol{\Theta}) - \boldsymbol{\kappa}(\mathbf{x}^*, X, \boldsymbol{\Theta})^\top C^{-1}\boldsymbol{\kappa}(X, \mathbf{x}^*, \boldsymbol{\Theta})$.

2.2 Regression Trees

A regression tree recursively partitions the decision space such that the provided samples with similar objective function values are grouped together (Loh, 2011). Each node ϕ of the tree contains data $Q^\phi = (X^\phi, \mathbf{y}^\phi)$ with N^ϕ samples, where each row of $X^\phi \in \Re^{N^\phi \times n}$ is a decision vector $\mathbf{x}_i$ and each row of $\mathbf{y}^\phi \in \Re^{N^\phi \times 1}$ is its corresponding objective value $y_i, i = 1, \ldots, N^\phi$. Node ϕ is split into nodes ϕ^{left} and ϕ^{right} according to parameter $\theta = (j, t)$, which specifies a jth decision variable ($1 \le j \le n$) and a threshold t, by partitioning Q^ϕ into two disjoint subsets $Q_\theta^{\phi^{\text{left}}} = \{(X^{\phi^{\text{left}}}, \mathbf{y}^{\phi^{\text{left}}}) \mid (\mathbf{x}_i, y_i) \in Q^\phi \wedge x_{ij} \le t\}$ and $Q_\theta^{\phi^{\text{right}}} = Q^\phi \setminus Q_\theta^{\phi^{\text{left}}}$. That is, split parameter θ partitions the samples (rows) in Q^ϕ according to the value of variable x_j of each decision vector $\mathbf{x} \in X^\phi$.

Given a node ϕ and a split parameter θ, the quality of the split or the total loss is calculated as:

$$G(Q^\phi, \theta) = \frac{N^{\phi^{\text{left}}}}{N^\phi} H(Q_\theta^{\phi^{\text{left}}}) + \frac{N^{\phi^{\text{right}}}}{N^\phi} H(Q_\theta^{\phi^{\text{right}}}), \tag{5}$$

where $N^{\phi^{\text{left}}}$ and $N^{\phi^{\text{right}}}$ are the number of samples in $Q_\theta^{\phi^{\text{left}}}$ and $Q_\theta^{\phi^{\text{right}}}$, respectively, and $H(Q^\phi)$ is a loss function. For instance, the loss function for mean-squared error is:

$$H(Q^\phi) = \frac{1}{N^\phi} \sum_{y_i \in Q^\phi} (y_i - \bar{y}^\phi)^2, \tag{6}$$

where $\bar{y}^\phi$ is the mean objective value in Q^ϕ. An optimal split θ^* of a node ϕ can be found by minimizing the loss function defined in Equation (5) using a single objective optimization algorithm. In the proposed TGPR-MO surrogates, the splitting process is recursed for both $Q_{\theta^*}^{\phi^{\text{left}}}$ and $Q_{\theta^*}^{\phi^{\text{right}}}$ until splitting a node would produce a leaf node with less than a predefined minimum number of samples, $N_{\min}$.

For predicting any given decision variable value, the regression tree is traversed to the respective leaf node. The prediction of the tree at the leaf node l, which contains training subset Q^l with N^l number of samples, is $\bar{y}^l = \frac{1}{N^l} \sum_{y_i \in Q^l} y_i$.

3 Treed GPRs for Multiobjective Optimization (TGPR-MO)

The time complexity of a full GPR is cubic, which is polynomial. A suitable alternative to reduce the computational cost is to split the dataset and build GPRs exclusively in the trade-off region. The proposed TGPR-MO surrogates are based on this modelling approach. The following subsections describe the building process of the proposed TGPR-MO surrogates. A detailed accuracy and complexity analysis and comparison with full GPRs and sparse GPRs are also provided.

3.1 Building the TGPR-MO Surrogates

In a generalized treed GPR surrogate, first a regression tree model is built using the provided data as described in Section 2.2. The splitting at the nodes is done by minimizing the total loss function (Equation 5). The subset of data at the lth leaf node is Q^l, where $l = 1, \ldots, L$ and L is the total number of leaf nodes in the regression tree built. A GPR is fitted at every leaf node of the regression tree using the data $Q^l = (X^l, \mathbf{y}^l)$. The GPRs are built in a similar fashion as described in Section 2.1 by maximizing the marginal

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

likelihood at the leaf node l, as defined in Equation (3). The posterior predictive distribution of the lth GPR is $p\left(y^*|\mathbf{x}^*, X^l, \mathbf{y}^l, \boldsymbol{\Theta}\right)$, as defined in Equation (4). Building GPRs with smaller subsets of data reduces the overall cost of building surrogates compared to building a GPR with the entire dataset. However, building GPRs at all the leaf nodes becomes expensive when there are too many of them, and the data subset at each leaf node is large. Moreover, for solving an offline data-driven MOP, an accurate approximation of the global landscape of the underlying objective functions is not required. A good approximation of the local landscape near the trade-off region is sufficient.

While performing multiobjective optimization using surrogates, the approximation accuracy in the trade-off region is crucial and directly influences the quality of the approximated Pareto optimal solutions. Hence, to obtain better quality solutions, it is desirable to have accurate approximations in the trade-off region. While building the regression trees, the splits at each node divide the decision space. The leaf nodes of the regression trees are the smallest regions the decision space is split into. In addition, the initial approximation of regression tree surrogates (though not highly accurate) provides information about the trade-offs between the objective functions. After building the regression trees with all the provided data, an MOEA is run considering them as objective functions. The solutions found by the MOEA are not accurate, but they provide an approximation of the Pareto front. The accuracy of the trade-off region is the approximation accuracy of the region enclosed by the leaf nodes that predicts the approximated Pareto front. Later, local GPRs are built exclusively in the leaf nodes representing the trade-off region and achieve an accurate approximation only in the neighborhood of the Pareto set. Next, the algorithm to build the treed GPR surrogates for multiobjective optimization (TGPR-MO) tailored to solve offline data-driven MOPs is introduced.

Algorithm 1 starts with a dataset containing N samples with K objective functions and n decision variables. First, K regression trees (one per objective function) are built with all the provided data. Just the parameter $N_{\min}$ is adjusted, and the depth of the trees is not controlled. After building the regression trees, there can be a maximum of $\frac{N}{N_{\min}}$ leaf nodes. Initially, there are no GPRs present at the leaf nodes, and the predictions are from the regression trees (i.e., $\bar{y}^l$). Next, a population is initialized and an MOEA is executed that iteratively builds GPRs at specific leaf nodes. Each iteration consists of running the MOEA for $G_{\max}$ generations (that is a predefined parameter) and building a GPR at one leaf node in every tree. In every generation, offspring individuals are produced using crossover and mutation operators and evaluated using the treed GPRs. Selection of individuals is then performed using MOEA-specific selection criterion, and the process is repeated for $G_{\max}$ generations within an iteration. The inner workings of the proposed algorithm are exemplified in two-dimensional decision spaces in Section 1 and Figure 1 of the supplementary material.

After the MOEA completes $G_{\max}$ generations, the approximation errors of the solutions (for each objective) are compared. The comparison is done by first calculating the loss function value, here the mean squared error, as defined in Equation (6), of the leaf node predicting the objective values of the solutions found by the MOEA. For the treed GPR surrogate corresponding to the jth objective, let $l_1^j, \ldots, l_s^j$ denote the leaf nodes containing the s solutions found by the MOEA and let $H(Q^{l_1^j}), \ldots, H(Q^{l_s^j})$ denote the loss function value of those leaf nodes, then one GRP is built at the leaf node i^* with the maximum loss value, that is, $i^* = \arg\max_{i=1,\ldots,s} H(Q^{l_i^j})$, using its subset of samples $Q^{l_{i^*}^j}$. If multiple solutions fall within the same leaf node, the loss value is calculated once and only one GPR is built for that leaf node. The process of building GPRs is repeated for

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

Algorithm 1: Build process of TGPR-MO surrogates

Input: Offline data of size N with n decision variables and K objectives;

$N_{\min}$ = minimum number of samples at a leaf node;

$I_{\max}$ = maximum number of iterations of building GPRs at leaves;

$G_{\max}$ = maximum number of generations of the MOEA per iteration

Output: TGPR-MO surrogates

For each objective $j = 1, \ldots, K$, initialize a treed GPR surrogate by building a regression tree using the given offline data and $N_{\min}$

Initialize MOEA population

Initialize iteration counter, $I = 0$

while $I < I_{\max}$ **and** *any solution falls in the leaf nodes without GPRs* **do**

  Set counter of generations per iteration, $G = 0$

  while $G < G_{\max}$ **do**

    Perform crossover and mutation and generate offspring

    Evaluate the individuals using the treed GPR surrogates and combine the parents and offspring

    Perform MOEA specific selection

    $G = G + 1$

  For each treed GPR surrogate $j = 1, \ldots, K$, find the leaf node that has the maximum loss function value in the prediction of the population of s solutions: $l^j_{\text{ML}} = \arg\max_{i=1,\ldots,s} H(Q^{l^j_i})$ where $j = 1, \ldots, K$

  Build GPRs using subset $Q^{l^j_{\text{ML}}}$ for $j = 1, \ldots, K$

  Replace the prediction of the trees' leaf nodes with the built GPRs

  $I = I + 1$

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

$j = 1, \ldots, K$ treed GPRs. The process of adding GPRs to the leaf nodes is repeated until any of the following criteria are met:

- Number of iterations has reached the predefined maximum of $I_{\max}$.
- All the solutions in the MOEA's population fall in the leaf nodes that have GPRs built.

The regression trees perform two tasks: (1) they provide an approximation of the underlying objective functions, and (2) they split the decision space into smaller regions. Using regression trees over other clustering algorithms is advantageous since they are optimized for reducing the prediction error. Other clustering algorithms, for example, K-means clustering and mean shift clustering, utilize a loss function in the decision space and not in the objective space. Therefore, regression trees provide a good prediction of the approximated objective functions in addition to splitting the decision space into subregions. In the initial iterations, the MOEA first finds solutions using the approximation provided by the regression tree surrogates (that may have poor accuracy). These solutions have a higher approximation error, but they provide information about the trade-off between the objective functions. To improve the approximation accuracy, GPRs are built for solutions provided by the trees that have the maximum approximation error. This ensures that GPRs are built exclusively in the region of the Pareto set and where the tree's approximation is the worst, simultaneously. In later iterations, if the decision vector in the MOEA's population falls within the path of the leaf node where a GPR is already built, the posterior predictive mean of the GPR is used as the final prediction of the surrogate. Otherwise, the prediction is the mean value at the leaf node of the regression trees. Local GPRs are built sequentially because building a local GPR at a

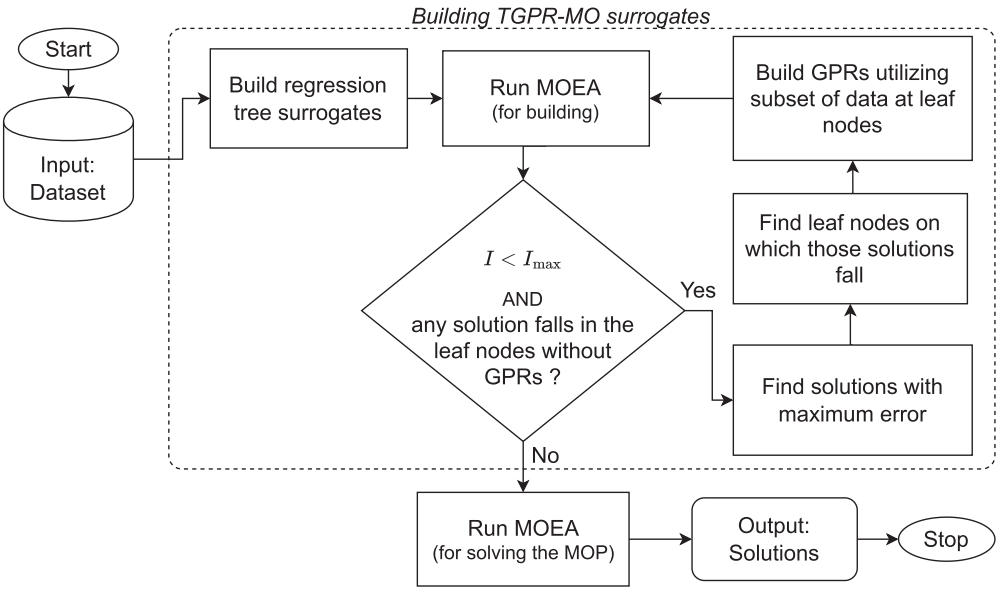

Figure 2: The flowchart for solving offline data-driven MOPs with TGPR-MO surrogates with an MOEA.

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

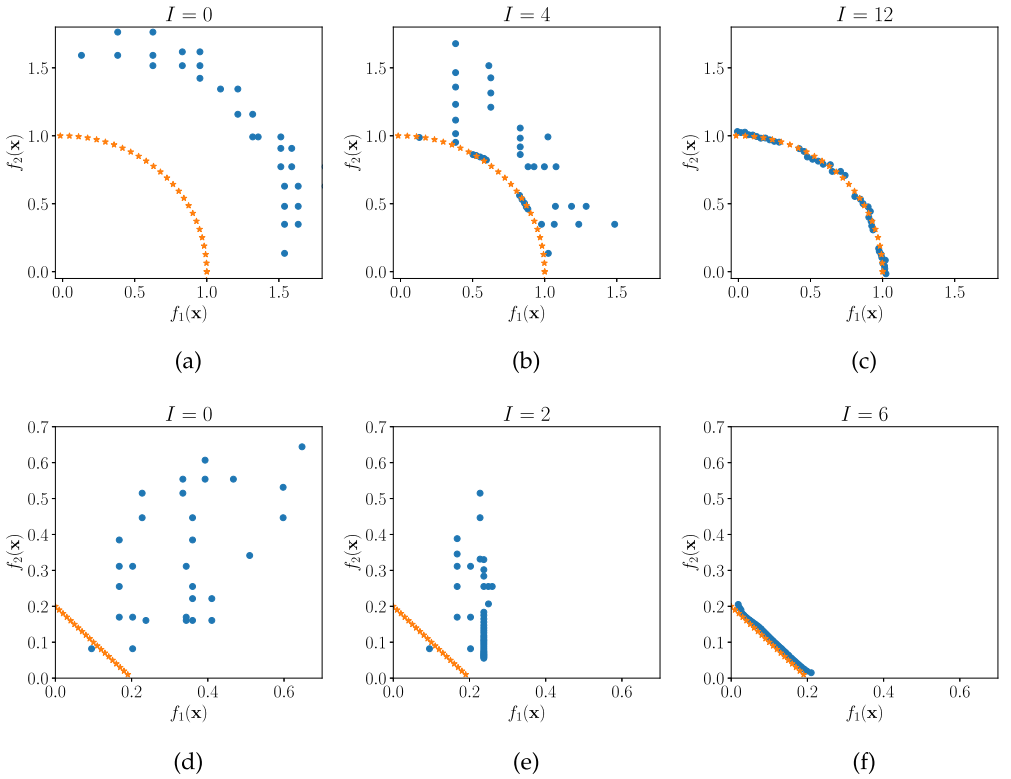

Figure 3: Progress of the building process of TGPR-MO surrogates for a bi-objective DTLZ2 (a)–(c) and a DBMOPP (d)–(f) problem. The plots show the solutions (in blue) of the MOEA (predicted by the treed GPR surrogates) at different iterations I of the building process. The Pareto front of the underlying MOP is shown in orange '⋆'.

leaf node improves the accuracy of the solutions in the following iteration. Thus, many solutions found in the previous iteration (belonging to different leaf nodes of the trees) may be eliminated as newer and better solutions are discovered. Eliminating solutions reduces the number of local GPRs required to be built to approximate the underlying objective functions.

The flowchart for solving offline data-driven MOPs with TGPR-MO surrogates is shown in Figure 2. The approach starts with a dataset, and regression trees are built for each objective. The blocks within the dotted lines represent the building process of TGPR-MO surrogates as described in Algorithm 1. After Algorithm 1 is completed, the surrogates are composed of regression trees with GPRs at some of the leaf nodes. The built TGPR-MO surrogates are then used for multiobjective optimization using a second MOEA. Or the MOEA used in Algorithm 1 may simply continue to run from the last population for an additional number of generations without further modifying the surrogates.

The building process of TGPR-MO surrogates is illustrated in Figure 3. The figure shows the solutions obtained by the MOEA in some of the iterations of Algorithm 1 for two bi-objective test problems. The two problems are DTLZ2 with $n = 10$ (Figure 3a–3c) from the DTLZ (Deb et al., 2005) test suite and a distance-based visualizable test

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

Table 1: Configurations of the DBMOPP problems used.

Problem	Configuration	Dimension (n)	Objectives (K)
P1	*number of disconnected set regions* $= 0$, *number of local fronts* $= 0$, *number of dominance resistance regions* $= 0$, *number of discontinuous regions* $= 0$	2, 5, 7, and 10	3, 5, and 7
P2	*number of disconnected set regions* $= 1$, *number of local fronts* $= 0$, *number of dominance resistance regions* $= 0$, *number of discontinuous regions* $= 0$	2, 5, 7, and 10	3, 5, and 7
P3	*number of disconnected set regions* $= 2$, *number of local fronts* $= 0$, *number of dominance resistance regions* $= 0$, *number of discontinuous regions* $= 0$	2, 5, 7, and 10	3, 5, and 7
P4	*number of disconnected set regions* $= 0$, *number of local fronts* $= 0$, *number of dominance resistance regions* $= 1$, *number of discontinuous regions* $= 0$	2, 5, 7, and 10	3, 5, and 7

problem (DBMOPP) (Fieldsend et al., 2019) with $n = 2$ (Figure 3d–3f). The problem configuration for the DBMOPP problem was the same as problem P1, which can be found in Table 1. For each problem, a dataset of size $N = 2000$ was generated by Latin hypercube sampling (LHS). The MOEA used was RVEA (Cheng et al., 2016) with standard parameter settings. The first column shows the solutions obtained by the MOEA in the first iteration before any GPRs are built at the leaf nodes of the trees. As the objective values of the solutions are predicted by the regression trees, the obtained solutions do not form a smooth Pareto front. In the second column, it can be observed that the solutions are obtained after four more iterations. Few GPRs are built at the leaf nodes, and the approximated Pareto front gradually becomes more smooth. At this iteration, the objective values of the solutions are predicted by the regression trees and GPRs at leaves for either or both the objective functions. The last column shows the final iteration of the surrogate building process, when all solutions in the MOEA's population are predicted by leaf nodes that contain GPRs. Building the surrogates for the DTLZ2 problem consumes more iterations than the DBMOPP problem instance. This behavior is due to the Pareto set of DTLZ problems that is located in a larger region of the decision space. Hence, more GPRs are required to be built at the leaves of the trees to approximate the tradeoff region accurately.

Figure 4 shows an illustration of the leaf node that is considered for building GPRs for two subsequent iterations in a bi-objective minimization problem. The iteration $I \in \{1, 2\}$ and the objective $j \in \{1, 2\}$ are also indicated. The red-colored leaf nodes are chosen for building GPRs at each iteration. In the first iteration, the solutions are predicted by nodes 4 and 5 for the first objective and node 5 for the second objective. Node 4 and node 5 for the first and second objectives' tree have the highest RMSE. Hence, GPRs are built using the subset of data in those leaf nodes. In the second iteration, the solutions predicted by the trees are from node 4 and node 5 (with the GPR built) for the first objective. For the second objective, the solutions are predicted by leaf node 4

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

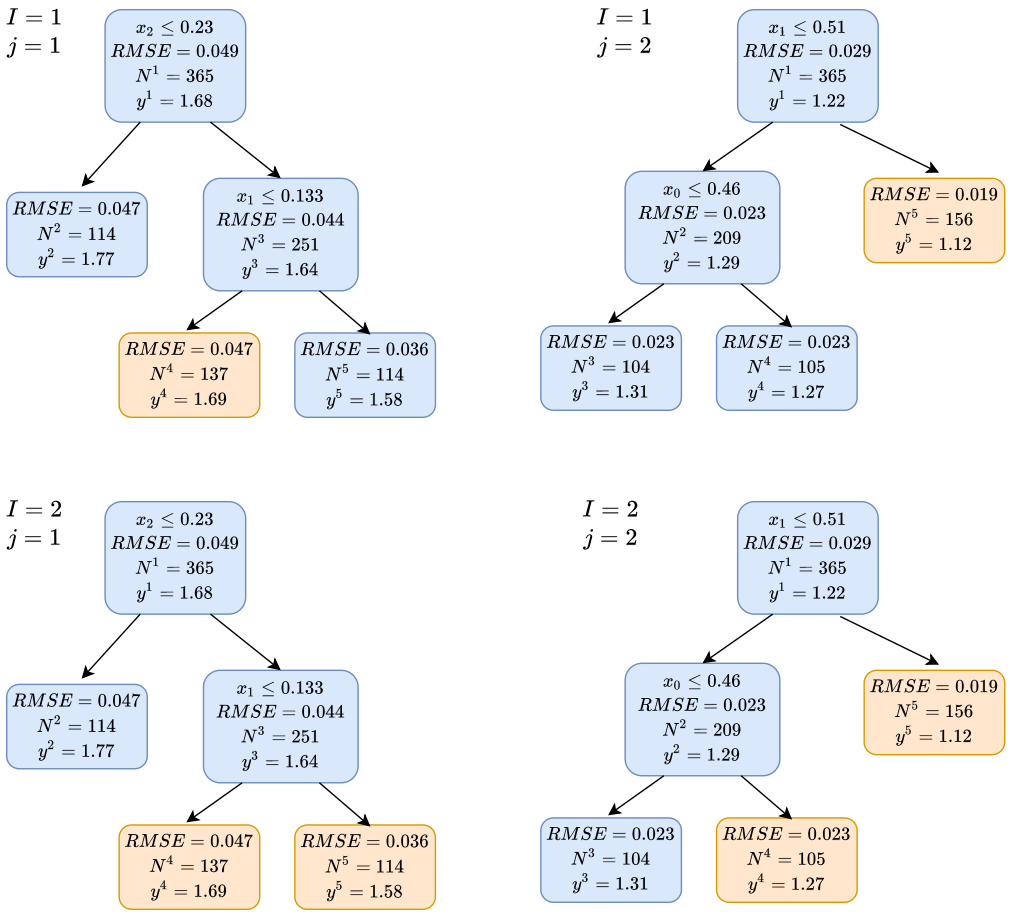

Figure 4: Illustration of the building process of GPRs at leaf nodes of the regression trees in two subsequent iterations. Red leaf nodes denote a GPR was built using the subset of data.

and node 5 (with the GPR built). The process of building GPRs is repeated at node 4 and node 5 for the first and second objective's trees, respectively. The process will be repeated until the stopping criteria mentioned before are satisfied.

3.2 Accuracy Analysis

The fitness landscape of a bi-objective DBMOPP problem with $n = 2$ (with the same configuration as P1 in Table 1) is illustrated in Figure 5. The figure shows the accuracy of the approximation of full GPR, sparse GPR, and TGPR-MO surrogates. The accuracy is measured as the RMSE (described in subsection 4.1) between the approximated objective values of the surrogates and the evaluated values of the underlying objective functions. The number of samples in the dataset was $N = 2000$ with Latin hypercube sampling (LHS). The contour lines represent the underlying objective functions' landscape in Figure 5a and the approximated objective functions' landscape in Figures 5b–5d. A further close-up of the tradeoff regions for the sparse GPR and TGPR-MO surrogates are shown in Figures 5e and 5f, respectively. The color grading represents

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

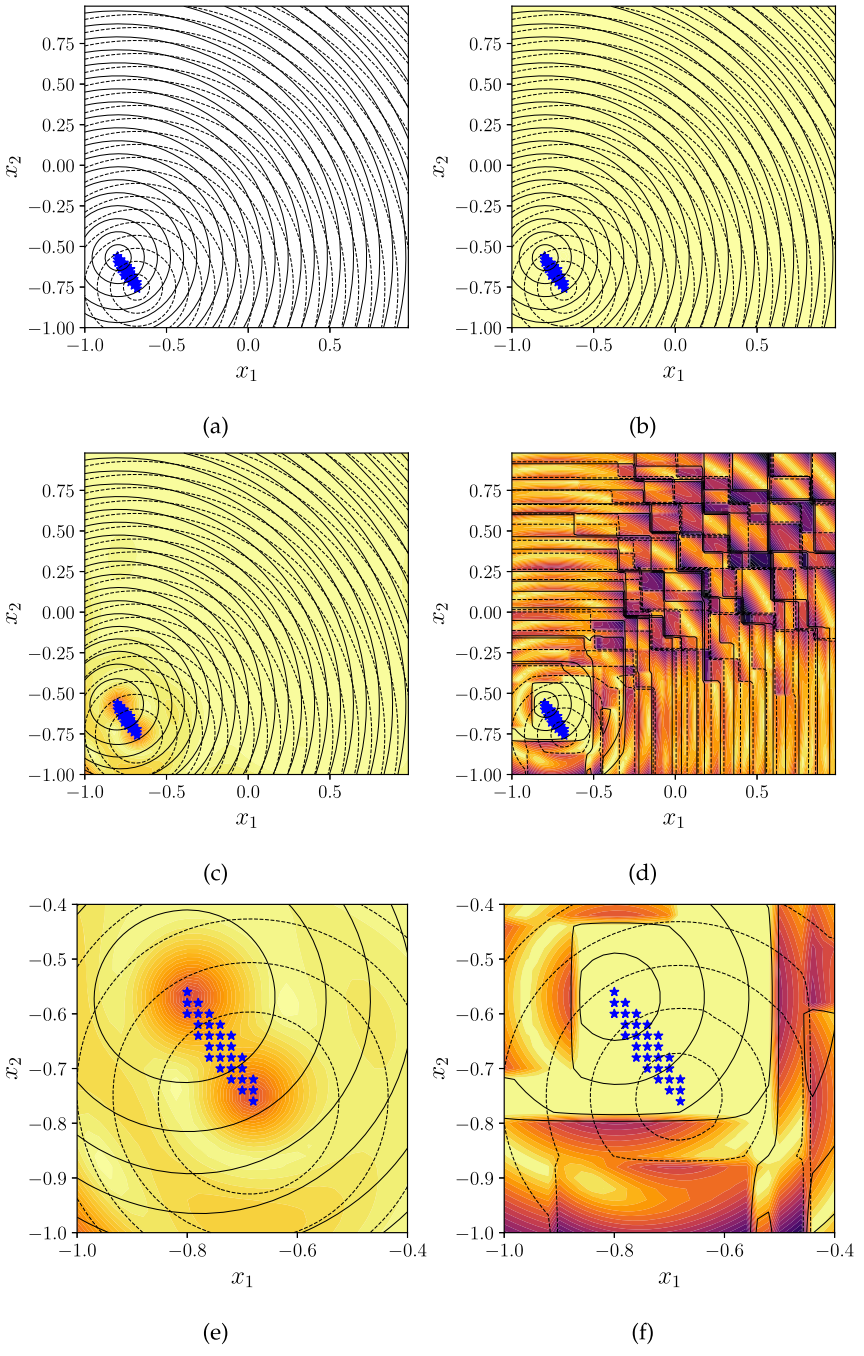

Figure 5: Contour plots of the (a) underlying objective function, landscape approximated by (b) full GPRs, (c) sparse GPRs, and (d) the proposed TGPR-MO surrogates. Close up of the Pareto set for (e) sparse GPRs and (f) the proposed TGPR-MO surrogates. The contour lines (in solid and dotted) indicate the value of objective functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, respectively. Color shade represents the RMSE in the approximation (darker is higher RMSE), and the blue dots show the Pareto set.

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

the RMSE between the approximated and the underlying objective functions, and the blue dots represent the Pareto set.

It can be observed that full GPRs (that use all the data) in Figure 5b have the highest accuracy in all the approximated regions of the decision space. For the sparse GPR surrogates in Figure 5c, the accuracy is good in all the regions with deterioration near the Pareto set. In the case of the proposed TGPR-MO surrogates in Figure 5d, the accuracy is low in most of the regions of the decision space except near the Pareto set. It can also be observed that the contour lines are linear in most regions because of the prediction provided by the leaf nodes of the trees. However, near the Pareto set the GPRs at the leaf nodes are used for predictions, thus making the contours nonlinear. Comparing the accuracy near the Pareto set for sparse GPR and TGPR-MO surrogates in Figures 5e and 5f, it is evident that TGPR-MO surrogates provide a better approximation accuracy compared to sparse GPR surrogates.

3.3 Complexity

While building TGPR-MO surrogates, a maximum of K GPRs are built in every iteration. These GPRs can have $N_{\min}$ to $2N_{\min} - 1$ samples, because a split at the node of a tree occurs when the subset size is larger than or equal to $2N_{\min}$. Thus every leaf node has at least $N_{\min}$ samples and a maximum of $2N_{\min} - 1$ samples. As these local GPRs have a smaller number of samples, the computational cost is low. The complexity of building TGPR-MO surrogates can vary depending on the function landscape, the number of decision variables, and the provided data. The worst-case complexity will be achieved when GPRs are built at all the leaf nodes of the regression tree with $2N_{\min} - 1$ samples at each leaf node. The complexity of building a GPR at a leaf node is $O((2N_{\min} - 1)^3)$. As the total number of leaf nodes (in the worst case) is $\frac{N}{2N_{\min}-1}$, the total complexity of building GPRs at all the leaf nodes is $O(N(2N_{\min} - 1)^2)$. For an MOP with K objective functions, the worst case time complexity is $O(KN(2N_{\min} - 1)^2)$ with a memory complexity of $O(KN(2N_{\min} - 1))$.

Even in the worst case, the computational cost of building TGPR-MO surrogates is significantly smaller than building full GPRs that have time and memory complexities of $O(N^3)$ and $O(N^2)$, respectively. The complexity for building sparse GPRs with M (where $M < N$) inducing points is $O(KNM^2)$. However, as mentioned previously, finding the induction points uses a gradient descent or greedy search algorithm that becomes expensive when the sample size is large, increasing the overall computational cost. In contrast, the complexity of TGPR-MO surrogates during optimization is primarily due to evaluating the individuals using the regression trees and GPRs at the leaves, which is significantly lower. The optimization results and comparisons of the proposed TPGR-MO surrogates with other surrogate models are shown in the next section.

4 Experimental Results

The goal of the proposed TGPR-MO surrogates is to build computationally cheaper surrogates when dealing with large datasets for solving offline data-driven MOPs. This section reports the tests performed to find whether TGPR-MO surrogates have significant improvement over sparse GPRs and full GPRs in computation time and the quality of the solutions (in hypervolume and accuracy). The starting point of the experiments was data generated from distance-based visualizable test problems (DBMOPP) (Fieldsend et al., 2019) for a better understanding of the behavior of the surrogates. For the DBMOPP test problems, the solutions in the decision and objective spaces can be simultaneously visualized. Thus, the search behavior can be observed with the progress

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

of the optimization process. In addition, the complexity of these problems can be controlled by the features (e.g., varying density, number of local fronts, dominance resistance regions, numbers of objective functions, and decision variables). These advantages provide more control over the problems and make them adjustable to reflect the needs of real-world optimization problems. These features of DBMOPP test problems prove to be more advantageous compared to DTLZ (Deb et al., 2005) benchmark problems. A detailed description of the experiment settings, results, and in-depth analysis are provided in the following subsections.

4.1 Experimental Setup

All the approaches for solving the offline data-driven MOP were coded in Python utilizing the DESDEO framework (Misitano et al., 2021) (https://desdeo.it.jyu.fi).[1] The experiments were executed on one node of an HPC cluster equipped with AMD Rome CPUs, each node having 128 cores running at 2.6 GHz, with 256 GiB of memory. Each individual run was executed on one CPU core and allocated a maximum memory usage of 2 GiB. The regression tree was built using the sklearn Python package (Pedregosa et al., 2011). For building the GPRs at the leaf nodes, the GPy (GPy, 2012) Python package was used.

4.1.1 Benchmark Problems

Four DBMOPP problems P1–4, as shown in Table 1, were used. The problem instances and data were generated by the code provided by Fieldsend et al. (2019). All combinations of numbers of objective functions ($K \in \{3, 5, 7\}$) and numbers of decision variables ($n \in \{2, 5, 7, 10\}$) were used for the tests. The total number of problem instances tested was 48, and every problem instance represented a different type of problem characteristic.

4.1.2 Datasets

For generating the data, Latin hypercube sampling (LHS) and multivariate normal sampling (MVNS) (Forrester et al., 2008) were used. In MVNS sampling, the objective functions were considered independent with mean at the mid-point of the decision space, that is, zero for DBMOPP problems. The variance of the sampling distribution was set to 0.1 for all the objective functions. Using MVNS sampling tests the ability of the surrogate models and optimization algorithm to handle biased (or skewed) datasets (Mazumdar et al., 2022). Sample sizes of initial data ($N \in \{2000, 10000, 50000\}$) were chosen for the tests. A total of 288 cases were tested, and 31 sets of data with a random seed were generated for each test case. Each of these datasets was the starting point of the three different surrogate models that were tested. These individual runs were independent, and the results were used to compare the surrogates' performances statistically.

4.1.3 Settings for TGPR-MO Surrogates

The MOEA for building TGPR-MO surrogates was RVEA (Cheng et al., 2016), a decomposition-based MOEA (Zhang and Li, 2007; Deb and Jain, 2014). Decomposition-based MOEAs have shown to be effective in solving offline data-driven MOPs with more than three objective functions. However, the surrogate is not limited to RVEA and one can use any MOEA of choice. The parameters for RVEA were kept the same as suggested by Cheng et al. (2016). The parameter $G_{\max}$ was set to 50. The maximum number of iterations was set to $I_{\max} = \frac{N}{N_{\min}} = \frac{N}{10n}$. Such a value of $I_{\max}$ was chosen considering

[1]Source code available at https://github.com/industrial-optimization-group/TreedGP_MOEA

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

one GPR is built at a leaf node for all the trees in each iteration (as the maximum number of leaf nodes is $\frac{N}{N_{\min}}$). The loss function used for building the trees was MSE as mentioned in Equation (6) with $N_{\min} = 10n$. These parameters were chosen because sufficient design points are necessary for building GPRs at the leaves. Based on recommendations from the GPR literature (Chapman et al., 1994; Jones et al., 1998), at least $10n$ points are required at each leaf node. The kernel used for building the GPRs at the leaf nodes was Mátern 5/2 with automatic relevance determination enabled.

4.1.4 Other Surrogates Tested

The proposed TGPR-MO surrogates were compared with sparse GPR and full GPR surrogates. The same GPy package and kernel were used for building both of these surrogates as for the TGPR-MO surrogates. For sparse GPR surrogates, the number of induction points was set to $M = 10n$. Here, the overall process of solving an offline data-driven MOP with a specific surrogate is referred to as an *approach* for simplicity. It should be noted that random forest was not considered in the tests as the focus of this paper is on GPR surrogates.

4.1.5 Parameter Settings of MOEA (for Solving the Offline Data-Driven MOP)

For solving the offline data-driven MOP with surrogates as objective functions, RVEA with the same default parameter settings was used. The termination criterion for RVEA was 1,000 generations, which was sufficient for all the approaches tested to converge. To generate a uniform Pareto front, the reference vectors are rearranged or adapted after a certain number of generations in RVEA. The reference vector adaptation rate was set to once every 100 generations.

4.1.6 Performance Indicators

The quality of the solutions obtained by the MOEA was measured in terms of their hypervolume (HV) after evaluating them with the underlying objective functions of the test problems. For computing the HV indicator, the reference point of $\mathbf{y}^* = (y_1^*, y_2^*, \ldots, y_K^*)$ in the objective space was chosen, such that it is dominated by all the solutions. In this paper, the reference point used was $\mathbf{y}^* = (2\sqrt{K}, 2\sqrt{K}, \ldots, 2\sqrt{K})$ because this point is always dominated in DBMOPP problems. The multivariate RMSE of the objective values of the solutions obtained by the MOEA with their respective underlying objective values was used to measure the accuracy. The multivariate RMSE is the Euclidean distance between the approximated and evaluated underlying objective function values of the solutions and is given by $\frac{1}{s}\sum_{i=1}^{s}\sqrt{\sum_{j=1}^{K}(\hat{f}_{j,i} - f_{j,i})^2}$, where s is the number of solutions, $\hat{f}_{j,i}$ and $f_{j,i}$ are the approximated and the underlying objective value, respectively, for the ith solution and jth objective. Finally, the computational cost of the various methods was measured as the time taken in seconds to build the surrogates.

In this paper, multivariate RMSE is referred to as RMSE for simplicity. The hypervolume and RMSE indicators are used here to benchmark the performance of TGPR-MO surrogates for solving an offline data-driven MOP. Evaluating the solutions with the underlying objective functions in an offline data-driven MOP may not be possible in real life.

4.2 Results and Discussions

While running the experiments, it was observed that building full GPR surrogates with 10,000 and 50,000 sample sizes consistently gave out-of-memory errors. The storage

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

complexity of full GPRs is $O(KN^2)$, and each element of the array is a 64-bit float. Hence, the memory requirement for three objective functions and sample sizes of 10,000 and 50,000 is 2.3 GiB and 56 GiB, respectively. Thus, building GPR surrogates using all the provided offline data will become almost impossible with readily available computing resources when the sample size is large. Hence, the results for full GPR surrogates are not included for sample sizes of 10,000 and 50,000.

A pairwise Wilcoxon two-tailed significance test was conducted to compare the performance of the different approaches. The calculated p-values were Bonferonni corrected, and $\alpha = 0.05$ was considered for rejecting the null hypothesis (an approach is not significantly better or worse than another approach). The median values were compared to determine whether an approach is significantly better or worse than another one, provided that the p-value is less than α. A pairwise comparison of the approaches with a scoring system was used for ranking the approaches. An approach is given a score of $+1$ if it is significantly better than the other approach. A score of -1 is given to the approach if it is significantly worse than the other approach. If the approach is not significantly better or worse than the other approach, a score of zero is given to both approaches. The sum of the scores is used for ranking all the approaches (a higher score gives a better rank) for the indicator being compared. A rank of "1" indicates that an approach has performed significantly better than all other approaches. Equal ranks indicate that those approaches are not significantly different in their performance.

The performance of the approaches is summarized in Table 2. The ranks of the approaches for three different indicators are color-coded in green, yellow, and red in the order of best to worst. It should be noted that these rankings are categorized with respect to the sample size and sampling strategies. The total number of instances is 96 (48 for LHS and MVNS each) for each sample size. Each instance consists of the combination of the various problem settings, that is, the number of samples (N), sampling strategy, number of objective functions (K), number of decision variables (n), and problem configuration. The number of instances in which full GPRs or sparse GPRs perform better, worse, or not significantly different compared to the proposed TGPR-MO surrogates is denoted by "+," "-," and "$\approx$," respectively. For TGPR-MO surrogates, only the number of instances where it performed significantly better than both full GPRs and sparse GPRs (or it ranked the best) is shown.

The rows measuring "Time" in Table 2 show that the proposed TGPR-MO surrogates were computationally the cheapest compared to the full GPR and sparse GPR for different sample sizes and sampling strategies. The number of instances TGPR-MO performed significantly better than the other two surrogates in each subcategory is close to 48 (that was the total number of instances in each subcategory). The TGPR-MO surrogates performed better than sparse GPR surrogates in hypervolume for all sample sizes and sampling strategies. However, full GPRs performed the best in hypervolume for a sample size of 2,000. For 2,000 samples, sparse GPRs outperformed TGPR-MO surrogates for both sampling strategies in RMSE, and full GPR performed the best. The RMSE of the solutions for TGPR-MO surrogates was better than sparse GPRs for sample sizes of 10,000 and 50,000 for LHS sampling only, whereas for MVNS sampling, the sparse GPR slightly outperformed TGPR-MO surrogates for 10,000 and 50,000 samples.

For a smaller sample size, one may choose full GPRs because they give the best performance in hypervolume and RMSE. However, full GPRs become almost impossible to build due to their high computational cost when the sample size is large. One can use TGPR-MO surrogates for larger sample sizes because they perform better in hypervolume and RMSE and excellently in computation time compared to sparse GPRs.

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

Table 2: Summary of pairwise comparison of the hypervolume (HV), RMSE, and time (seconds) for the different approaches. The number of instances in which full GP and sparse GP surrogates perform better, worse, and not significantly different compared to the TGP-MO surrogates is indicated by "+," "-," and "≈," respectively. The full GP runs failed due to memory overflow in the instances marked "—." The approaches' performance is ranked by the color code green, yellow, and red in the order of best to worst, respectively.

			Surrogate type		
Sample size	Sampling strategy	Indicator	Full GP + / - / ≈	Sparse GP + / - / ≈	TGP-MO +
2,000	LHS	HV	9 / 9 / 30	4 / 38 / 6	9
		RMSE	43 / 2 / 3	19 / 21 / 8	1
		Time (s)	0 / 48 / 0	1 / 46 / 1	46
	MVNS	HV	17 / 14 / 17	14 / 28 / 6	12
		RMSE	34 / 7 / 7	20 / 16 / 12	2
		Time (s)	0 / 48 / 0	1 / 47 / 0	47
10,000	LHS	HV	—	4 / 33 / 11	33
		RMSE	—	19 / 23 / 6	23
		Time (s)	—	0 / 48 / 0	48
	MVNS	HV	—	11 / 29 / 8	29
		RMSE	—	21 / 16 / 11	16
		Time (s)	—	0 / 48 / 0	48
50,000	LHS	HV	—	3 / 35 / 10	35
		RMSE	—	15 / 24 / 9	24
		Time (s)	—	0 / 48 / 0	48
	MVNS	HV	—	10 / 29 / 9	29
		RMSE	—	23 / 19 / 6	19
		Time (s)	—	0 / 48 / 0	48

However, for non-uniform sampling strategies, for example, MVNS, TGPR-MO surrogates suffer slightly in RMSE compared to sparse GPR surrogates. Sparse GPR surrogates have better RMSE than TGPR-MO surrogates because the variational parameters are selected by minimizing the KL divergence. Thus, the inducing inputs selected are not skewed even if the provided dataset is, for example, in MVNS sampling.

The performances of a few selected instances are shown in Table 3. The table shows the median and the standard deviation of hypervolume, RMSE, and time taken to build the surrogates for the three different approaches for 31 runs. The instances shown in the table were chosen based on the maximum difference between the median hypervolume of the best and the second best performing surrogates. One instance was selected from each sample size, sampling strategy, and number of objective functions. The figures in bold represent the best performing approaches. It can be observed that the proposed TGPR-MO surrogates performed the best in building time in the instances shown. It can also be observed that TGPR-MO surrogates produce solutions with an improved hypervolume and RMSE for sample sizes of 10,000 and 50,000 compared to sparse GPR surrogates. It can be observed that the building times of TGPR-MO surrogates are less

Table 3: Comparison of selected test instances showing the median hypervolume, RMSE, and time and their standard deviation (in italics) of the 31 test runs. The best performing approaches are shown in bold.

					Hypervolume			RMSE			Time (s)		
Sample size	Sampling strategy	Problem	K	n	Full GPR	Sparse GPR	TGPR-MO	Full GPR	Sparse GPR	TGPR-MO	Full GPR	Sparse GPR	TGPR-MO
2,000	LHS	P3	3	5	**7.54E+01**	5.49E+01	**7.48E+01**	**4.18E-02**	1.04E+00	**2.67E-01**	1.36E+02	8.64E+01	**1.40E+01**
					4.02E+00	*9.35E+00*	*2.73E+00*	*2.74E-01*	*5.72E-01*	*1.38E-01*	*1.85E+01*	*2.16E+01*	*1.20E+01*
		P4	5	10	1.46E+04	1.27E+04	**1.64E+04**	1.75E+00	1.79E+00	**9.33E-01**	6.16E+02	5.51E+02	**2.62E+01**
					6.73E+02	*1.23E+03*	*5.59E+02*	*3.69E-01*	*4.38E-01*	*1.66E-01*	*1.40E+02*	*1.43E+02*	*1.12E+01*
		P3	7	5	**9.21E+06**	6.55E+06	**9.23E+06**	**5.11E-02**	2.16E+00	1.66E-01	3.20E+02	2.10E+02	**2.83E+01**
					1.00E+05	*1.24E+06*	*9.32E+04*	*8.26E-02*	*9.70E-01*	*1.11E-01*	*5.38E+01*	*3.46E+01*	*1.53E+01*
	MVNS	P3	3	10	5.71E+01	5.65E+01	**6.27E+01**	**1.08E+00**	1.52E+00	**8.55E-01**	3.25E+02	3.56E+02	**1.31E+01**
					4.14E+00	*6.35E+00*	*5.10E+00*	*4.65E-01*	*6.09E-01*	*1.99E-01*	*6.76E+01*	*1.19E+02*	*2.45E+00*
		P2	5	10	**1.26E+04**	1.06E+04	**1.30E+04**	**1.86E+00**	2.76E+00	2.04E+00	5.34E+02	5.51E+02	**1.39E+01**
					2.08E+03	*2.10E+03*	*1.20E+03*	*1.95E-01*	*1.59E-01*	*2.78E-01*	*1.01E+02*	*7.64E+01*	*8.00E+00*
		P2	7	7	7.81E+06	6.42E+06	**8.82E+06**	**7.80E-01**	3.76E+00	9.80E-01	3.57E+02	4.36E+02	**1.27E+01**
					5.36E+05	*2.62E+05*	*5.30E+05*	*2.07E-01*	*8.55E-01*	*2.38E-01*	*4.40E+01*	*6.32E+01*	*1.69E+01*
10,000	LHS	P3	3	5	—	6.26E+01	**7.58E+01**	—	8.95E-01	**1.50E-01**	—	1.45E+03	**2.49E+01**
						9.78E+00	*5.93E-01*		*4.85E-01*	*5.05E-02*		*2.22E+02*	*8.80E+00*
		P4	5	10	—	1.27E+04	**1.65E+04**	—	2.10E+00	**7.09E-01**	—	6.52E+03	**5.93E+01**
						6.09E+02	*8.26E+02*		*2.22E-01*	*2.28E-01*		*1.24E+03*	*2.11E+01*
		P1	7	10	—	6.71E+06	**8.59E+06**	—	1.56E+00	**9.73E-01**	—	9.37E+03	**3.78E+01**
						1.90E+05	*2.61E+05*		*1.84E-01*	*8.82E-02*		*1.38E+03*	*1.32E+01*
	MVNS	P3	3	10	—	4.97E+01	**6.27E+01**	—	1.68E+00	**9.09E-01**	—	4.12E+03	**1.42E+01**
						5.40E+00	*3.31E+00*		*3.40E-01*	*1.52E-01*		*6.69E+02*	*1.52E+00*
		P3	5	5	—	1.23E+04	**1.75E+04**	—	2.32E+00	**1.83E-01**	—	2.38E+03	**4.73E+01**
						2.50E+03	*3.15E+02*		*1.10E+00*	*1.22E-01*		*4.65E+02*	*1.19E+01*
		P3	7	5	—	6.55E+06	**9.24E+06**	—	2.91E+00	**1.22E-01**	—	3.51E+03	**8.94E+01**
						1.22E+06	*1.01E+05*		*1.34E+00*	*1.42E-01*		*6.54E+02*	*3.02E+01*

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

Table 3: Continued.

Sample size	Sampling strategy	Problem	K	n	Hypervolume			RMSE			Time (s)		
					Full GPR	Sparse GPR	TGPR-MO	Full GPR	Sparse GPR	TGPR-MO	Full GPR	Sparse GPR	TGPR-MO
50,000	LHS	P3	3	10	—	6.04E+01	**7.30E+01**	—	8.29E-01	**3.91E-01**	—	2.16E+04	**3.16E+01**
						7.28E+00	*2.62E+00*		*6.08E-01*	*8.63E-02*		*1.77E+03*	*9.25E+00*
		P3	5	10	—	1.45E+04	**1.65E+04**	—	8.64E-01	**6.18E-01**	—	3.59E+04	**3.90E+01**
						4.10E+02	*4.04E+02*		*2.10E-01*	*1.20E-01*		*6.29E+03*	*1.10E+01*
		P1	7	10	—	7.49E+06	**8.65E+06**	—	9.20E-01	**7.44E-01**	—	5.05E+04	**8.77E+01**
						1.12E+05	*1.65E+05*		*9.23E-02*	*8.59E-02*		*9.41E+03*	*3.44E+01*
	MVNS	P3	3	5	—	5.27E+01	**7.59E+01**	—	1.77E+00	**9.73E-02**	—	8.19E+03	**5.29E+01**
						1.14E+01	*4.41E-01*		*9.11E-01*	*5.04E-02*		*6.35E+02*	*2.07E+01*
		P3	5	5	—	1.25E+04	**1.76E+04**	—	2.73E+00	**1.43E-01**	—	1.34E+04	**8.68E+01**
						2.04E+03	*3.03E+02*		*1.13E+00*	*1.49E-01*		*7.96E+02*	*4.14E+01*
		P3	7	5	—	6.55E+06	**9.26E+06**	—	3.75E+00	**9.81E-02**	—	1.87E+04	**1.91E+02**
						1.05E+06	*1.18E+05*		*1.45E+00*	*1.36E-01*		*2.24E+03*	*7.59E+01*

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

than those of sparse GPR by an order of about 10^2 and 10^3 for samples size of 10,000 and 50,000 respectively. The building time also increases with the number of objective functions for sparse GPR surrogates.

The total number of samples utilized to build TGPR-MO surrogates with the number of iterations for six different problem instances with sample sizes of 2,000, 10,000, and 50,000 is shown in Figure 6. The solid line shows the mean number of samples used, and the shaded region denotes the 95% confidence interval of the runs for each treed GPR surrogate (for each objective). The plots have been extended in the iteration axis to the maximum allowed iterations, $I^{\max} = \frac{N}{10n}$. The iteration axis is broken when $I^{\max}$ is large to reduce the width of the plots. It can be observed that the number of samples utilized converges before the maximum iterations and varies with the objective. Certain problem instances required more samples during the building process than others, especially due to the number of decision variables and the characteristics of the problem. If the trade-off region is located in a smaller region of the decision space, the building process converges quickly and therefore consumes fewer samples.

The quantity of data available at the trade-off region affects the accuracy and hypervolume of the solutions obtained that is a general challenge while solving offline data-driven MOPs. The proposed TGPR-MO surrogates split the decision space into subregions and approximate the underlying objective functions at the trade-off region. It is best suitable when the Pareto set is located in a smaller region of the decision space. When the Pareto set is in a larger region, the prediction of TGPR-MO surrogates is from multiple leaf node GPRs and has discontinuities near the splits of the regression trees. Therefore the approximated Pareto front has some discontinuities compared to sparse GPRs or full GPRs. Further tests with the DTLZ (Deb et al., 2005) benchmark problems are given in the supplementary material.

5 Conclusions

This paper proposed tailored surrogate models that can be built with a lower computational cost compared to sparse GPRs and full GPRs for solving offline data-driven MOPs when dealing with large datasets. The proposed TGPR-MO surrogates performed significantly better than sparse GPR surrogates in hypervolume, RMSE, and computation time for most of the instances of the DBMOPP problems. The full GPR surrogates failed to complete the runs for larger sample sizes due to memory restrictions. Thus, it can be concluded that the proposed TGPR-MO surrogates are best suited for solving offline data-driven MOPs with a large size dataset.

A GPR at a leaf node of a tree approximates certain regions of the decision space. This feature can be exploited while solving offline data-driven MOPs with preferences from the decision maker in an a priori or interactive fashion. The provided preferences for objective functions can be utilized to build GPRs at the leaf nodes, and only the solutions that follow the preferences can be approximated. Developing an interactive framework utilizing the TGPR-MO surrogates will be one of the future works.

Utilizing the correlation between the objective functions using multi-target regression trees (Osojnik et al., 2018) and multi-output GPRs (Borchani et al., 2015) is an interesting future research direction. Tests and comparisons of the uncertainty prediction provided by the proposed TGPR-MO surrogates will be conducted for solving offline data-driven MOPs. Selecting a suitable kernel for a given dataset is an active research topic. Integrating automatic kernel selection into the TGPR-MO surrogates will be quite beneficial. One of the major drawbacks of the TGPR-MO surrogates is the discontinuity between the leaf node GPRs. Tackling discontinuity in the prediction (van Stein et al.,

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

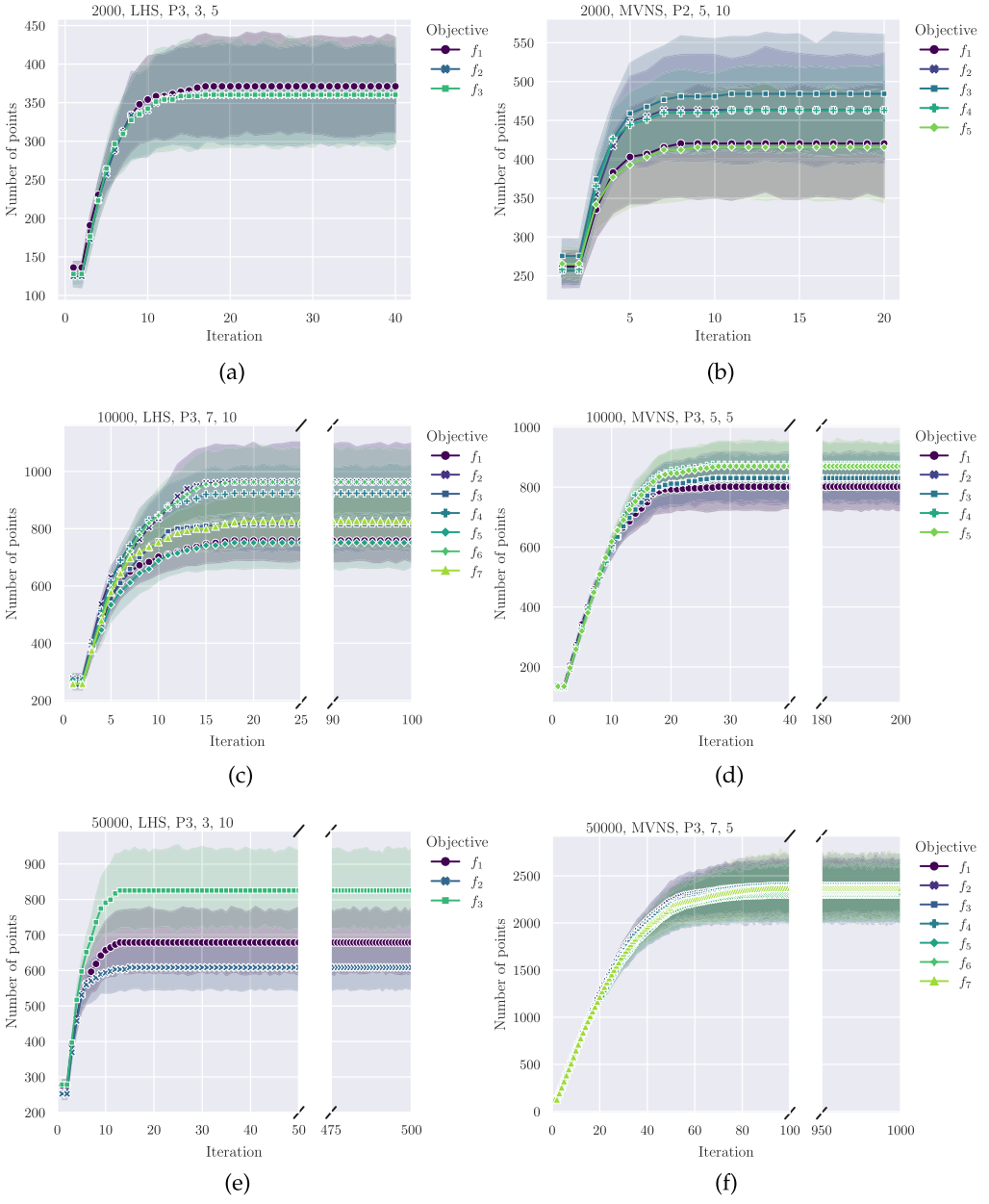

Figure 6: The median and 95% confidence interval of the total number of samples utilized (or the sum of the samples available at the leaf nodes that are considered for building GPRs) with iterations for three different DBMOPP problem instances as displayed according to the following format (N, sampling strategy, problem, K, n). The iteration axis is extended to the maximum possible iteration $\frac{N}{10n}$. The iteration axes for plots 6c–6f are broken to accommodate the plots.

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

2015; Wang et al., 2017) at the partition of the decision space (as provided by the tree) will be a future task. Further improvements can be made in the way the trees partition the decision space. Instead of using a traditional loss function, a nonlinear trade-off criterion can be formulated to split the nodes. Such splitting criteria will enable the TGPR-MO surrogates to approximate the trade-off region with fewer samples. Testing the TGPR-MO surrogates for solving real-life offline data-driven MOPs will be a future task.

Acknowledgments

This research was partly supported by the Academy of Finland (grant number 311877 and 322221) and is related to the thematic research area DEMO (Decision Analytics utilizing Causal Models and Multiobjective Optimization, http://www.jyu.fi/demo) of the University of Jyväskylä. The numerical experiments were performed on Mahti supercomputer provided by CSC (https://www.csc.fi).

References

Assael, J.-A. M., Wang, Z., Shahriari, B., and de Freitas, N. (2014). Heteroscedastic treed Bayesian optimisation. *CoRR*. Retrieved from arXiv:1410.7172.

Borchani, H., Varando, G., Bielza, C., and Larrañaga, P. (2015). A survey on multi-output regression. *WIREs Data Mining and Knowledge Discovery*, 5(5):216–233. 10.1002/widm.1157

Chapman, W., Welch, W., Bowman, K., Sacks, J., and Walsh, J. (1994). Arctic sea ice variability: Model sensitivities and a multidecadal simulation. *Journal of Geophysical Research: Oceans*, 99(C1):919–935. 10.1029/93JC02564

Cheng, R., Jin, Y., Olhofer, M., and Sendhoff, B. (2016). A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):773–791. 10.1109/TEVC.2016.2519378

Chipman, H. A., George, E. I., and McCulloch, R. E. (1998). Bayesian CART model search. *Journal of the American Statistical Association*, 93(443):935–948. 10.1080/01621459.1998.10473750

Chugh, T., Sindhya, K., Hakanen, J., and Miettinen, K. (2019). A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computing*, 23:3137–3166. 10.1007/s00500-017-2965-0

Das, K., and Srivastava, A. N. (2010). Block-GP: Scalable Gaussian process regression for multimodal data. In *2010 IEEE International Conference on Data Mining*, pp. 791–796.

Deb, K., and Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601. 10.1109/TEVC.2013.2281535

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2005). Scalable test problems for evolutionary multiobjective optimization. In A. Abraham, L. Jain, and R. Goldberg (Eds.), *Evolutionary multiobjective optimization: Theoretical advances and applications*, pp. 105–145. Springer.

Emmerich, M., Giannakoglou, K., and Naujoks, B. (2006). Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439. 10.1109/TEVC.2005.859463

Fieldsend, J. E., Chugh, T., Allmendinger, R., and Miettinen, K. (2019). A feature rich distance-based many-objective visualisable test problem generator. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 541–549.

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

Forrester, A., Sobester, A., and Keane, A. (2008). *Engineering design via surrogate modelling*. John Wiley & Sons.

GPy (2012). GPy: A Gaussian process framework in python. http://github.com/SheffieldML/GPy.

Gramacy, R. B., and Lee, H.K.H. (2008). Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130. 10.1198/016214508000000689

Hughes, E. J. (2001). Evolutionary multi-objective ranking with uncertainty and noise. In *Proceedings of Evolutionary Multi-Criterion Optimization*, pp. 329–343.

Jin, Y., Wang, H., Chugh, T., Guo, D., and Miettinen, K. (2019). Data-driven evolutionary optimization: An overview and case studies. *IEEE Transactions on Evolutionary Computation*, 23(3):442–458. 10.1109/TEVC.2018.2869001

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492. 10.1023/A:1008306431147

Kim, H.-M., Mallick, B. K., and Holmes, C. C. (2005). Analyzing nonstationary spatial data using piecewise Gaussian processes. *Journal of the American Statistical Association*, 100(470):653–668. 10.1198/016214504000002014

Loh, W.-Y. (2011). Classification and regression trees. *WIREs Data Mining and Knowledge Discovery*, 1(1):14–23. 10.1002/widm.8

Mazumdar, A., Chugh, T., Hakanen, J., and Miettinen, K. (2020). An interactive framework for offline data-driven multiobjective optimization. In *Proceedings of Bioinspired Optimization Methods and Their Applications*, pp. 97–109.

Mazumdar, A., Chugh, T., Hakanen, J., and Miettinen, K. (2022). Probabilistic selection approaches in decomposition-based evolutionary algorithms for offline data-driven multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 26(5):1182–1191. 10.1109/TEVC.2022.3154231

Mazumdar, A., Chugh, T., Miettinen, K., and López-Ibáñez, M. (2019). On dealing with uncertainties from Kriging models in offline data-driven evolutionary multiobjective optimization. In *Proceedings of Evolutionary Multi-Criterion Optimization*, pp. 463–474.

Misitano, G., Saini, B. S., Afsar, B., Shavazipur, B., and Miettinen, K. (2021). DESDEO: The modular and open source framework for interactive multiobjective optimization. *IEEE Access*, 9:148277–148295. 10.1109/ACCESS.2021.3123825

Osojnik, A., Panov, P., and Džeroski, S. (2018). Tree-based methods for online multi-target regression. *Journal of Intelligent Information Systems*, 50:315–339. 10.1007/s10844-017-0462-7

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rahat, A.A.M., Wang, C., Everson, R. M., and Fieldsend, J. E. (2018). Data-driven multi-objective optimisation of coal-fired boiler combustion systems. *Applied Energy*, 229:446–458. 10.1016/j.apenergy.2018.07.101

Rasmussen, C. E., and Williams, C.K.I. (2006). *Gaussian processes for machine learning*. MIT Press.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. (2016). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175. 10.1109/JPROC.2015.2494218

Evolutionary Computation corrected proof
https://doi.org/10.1162/evco_a_00329

Snelson, E., and Ghahramani, Z. (2005). Sparse Gaussian processes using pseudo-inputs. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, pp. 1257–1264.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959. Curran Associates.

Titsias, M. K. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, pp. 567–574.

van Stein, B., Wang, H., Kowalczyk, W., Bäck, T., and Emmerich, M. (2015). Optimally weighted cluster kriging for big data regression. In E. Fromont, T. De Bie, and M. van Leeuwen (Eds.), *Advances in intelligent data analysis XIV*, pp. 310–321. Springer.

Wang, H., and Jin, Y. (2020). A random forest-assisted evolutionary algorithm for data-driven constrained multiobjective combinatorial optimization of trauma systems. *IEEE Transactions on Cybernetics*, 50(2):536–549. 10.1109/TCYB.2018.2869674

Wang, H., Jin, Y., and Jansen, J. O. (2016). Data-driven surrogate-assisted multiobjective evolutionary optimization of a trauma system. *IEEE Transactions on Evolutionary Computation*, 20(6):939–952. 10.1109/TEVC.2016.2555315

Wang, H., Jin, Y., Sun, C., and Doherty, J. (2019). Offline data-driven evolutionary optimization using selective surrogate ensembles. *IEEE Transactions on Evolutionary Computation*, 23(2):203–216. 10.1109/TEVC.2018.2834881

Wang, H., van Stein, B., Emmerich, M., and Bäck, T. (2017). Time complexity reduction in efficient global optimization using cluster kriging. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 889–896.

Yang, C., Ding, J., Jin, Y., and Chai, T. (2020). Offline data-driven multiobjective optimization: Knowledge transfer between surrogates and generation of final solutions. *IEEE Transactions on Evolutionary Computation*, 24(3):409–423. 10.1109/TEVC.2019.2925959

Zhang, Q., and Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731. 10.1109/TEVC.2007.892759

Zitzler, E., and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—A comparative case study. In *Parallel Problem Solving from Nature*, pp. 292–301.