Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

TREED-GAUSSIAN PROCESSES WITH SUPPORT VECTOR MACHINES AS NODES FOR NONSTATIONARY BAYESIAN OPTIMIZATION

Antonio Candelieri

University of Milano-Bicocca
Department of Economics,
Management and Statistics
Piazza Ateneo Nuovo 1
Milan, 20126, ITALY

Giulia Pedrielli

Arizona State University
School of Computing Informatics
& Decision Systems Engineering
699 S Mill Ave
Tempe, 85281, USA

ABSTRACT

A large family of black box methods rely on surrogates of the unknown, possibly non linear non convex reward function. While it is common to assume stationarity of the reward, many real-world problems satisfy this assumption only locally, hindering the spread application of such methods. This paper proposes a novel nonstationary regression model combining Decision Trees and Support Vector Machine (SVM) classification for a hierarchical non-axis-aligned partition of the input space. Gaussian Process (GP) regression is performed within each identified subregion. The resulting nonstationary regression model is the Treed Gaussian process with Support Vector Machine (SVMTGP), and we investigate the sampling efficiency from using our a model within a Bayesian optimization (BO) context. Empirically, we show how the resulting algorithm, SVMTGP-BO never underperforms BO when this is applied to an homogeneous Gaussian process, while it shows always better performance compared to the homogeneous model with nonlinear functions with complex landscapes.

1 INTRODUCTION

1.1 Motivation and Background

Gaussian Process (GP) modelling (Rasmussen and Williams 2006; Gramacy 2020) is usually based on the assumption of *stationarity*, meaning that the same covariance function (*kernel*), is used throughout the entire input space. In many real-world problems such an assumption could be not desirable because the modelled process might exhibit a different variability from one region to another of the input space. Fully non-stationary kernels have been used in the literature to overcome this challenge (Hebbal et al. 2021; Higdon et al. 1999; Schmidt and O'Hagan 2003). However, the complexity of the model makes these approaches computationally intractable. To face this problem, the idea of partitioning of the input space into subregions, and fitting separate stationary GP models within each subregion was proposed, leading to a single nonstationary model (Gramacy and Lee 2008; Kim et al. 2005). The combined adoption of partitioning and Gaussian processes (sometimes referred to as treed-GP) results in computationally tractable algorithms. Although first papers on *Treed-GP* modelling date back to more than 10 years ago (Gramacy et al. 2007; Gramacy and Le Digabel 2011), a renewed interest on this topics has been recently emerging, for instance with respect to building engineering applications (Civera et al. 2017; Civera et al. 2020).

Partitioning is also used in Adaptive Random Search methods for global optimization, such as Probabilistic Branch and Bound (PBnB) (Zabinsky et al. 2019; Zabinsky and Huang 2020), and level set estimation (Shekhar and Javidi 2019). Partitioning is usually based on a simple procedure consisting into

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

dividing a region of the input space equally along the longest side. Methods differ on the criteria used to choose the region to split, also depending on the specific task (e.g., function learning, level set estimation or global optimization).

Differently, *Treed-GP modelling considers splitting as a part of the learning process itself*. For instance, if one uses a regression tree learning algorithm like CART (Classification And Regression Tree), regions are not split equally along the longest side but the best split is identified to minimize regression error on the available observations. In all these cases, splits are always parallel to the dimensions of the input space, thus implicitly assuming that the nonstationarity of the function can be expressed through splits into axis-aligned hyper-rectangles, throughout the input space. Warping is considered in Marmin et al. (2018) for multivariate outputs, but the approach relies on low effective dimensionality concerning the direction of largest variation of the variance. Also, the local GP models approach proposed in Gramacy and Apley (2015), overcomes axis-aligned partitioning underlying TreedGP. Local-GPs do not consider partitioning and fitting, on-the-fly, a GP model on the observations within the neighbourhood of a *target* location. Our SVM-TGP replaces axis-aligned with a more flexible and general partitioning technique and does not assume the existence of a low dimensional manifold.

To overcome these modelling limitations, more sophisticated regression and classification tree algorithms have been proposed within the Machine Learning community, with nodes of the decision tree consisting of classification models allowing for non-orthogonal splits of regions of the input space. A typical example consists in using linear Support Vector Machine (SVM) classifiers as nodes of a decision tree (Fei and Liu 2006; Chang et al. 2010; Cohen and Fernández 2012; de Boves Harrington 2015; Chen and Ge 2019). Figure 1 shows the resulting divisions of the input space into regions provided by (a) a simple partitioning based on equally dividing split, (b) a regression decision tree (i.e., a common CART), and (c) a decision tree with linear SVM classifiers as nodes.

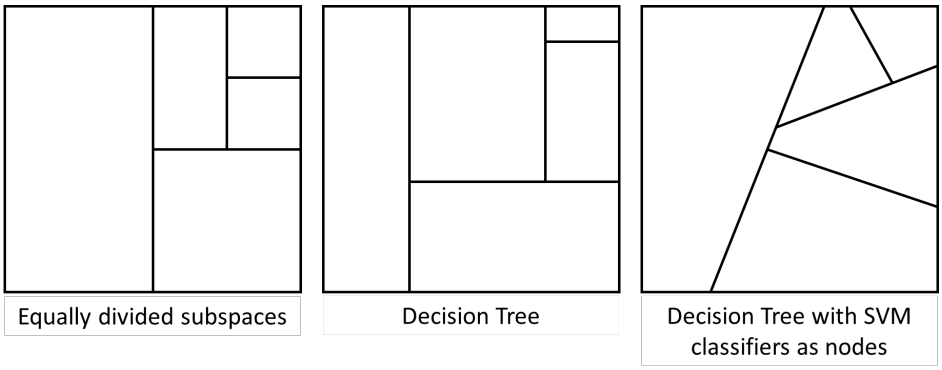


Figure 1: Different separations into subspaces for three different classes of methods.

1.2 Contributions

Our approach tackles the non-stationarity challenge with non linear boundaries while maintaining computational efficiency. Our algorithm, contributes to the literature by:

- proposing a new Treed-GP having SVM classifiers as intermediate nodes and GP regression models as leaves. We refer to the resulting nonstationary model as *SVMTGP*.
- providing a specific implementation of SVMTGP for Bayesian optimization (BO), with a simple and computational efficient criterion (i.e., median) for organizing, into two classes, the observations falling within each partition.
- empirically comparing the advantagaes/drawbacks in using linear or non-linear SVM classifiers in SVMTGP. Figure 2 compares the splitting of the input space obtained from a decision tree with linear and nonlinear SVM classifiers as nodes.
- providing preliminary empirical evidence of the SVMTGP-based BO performance.

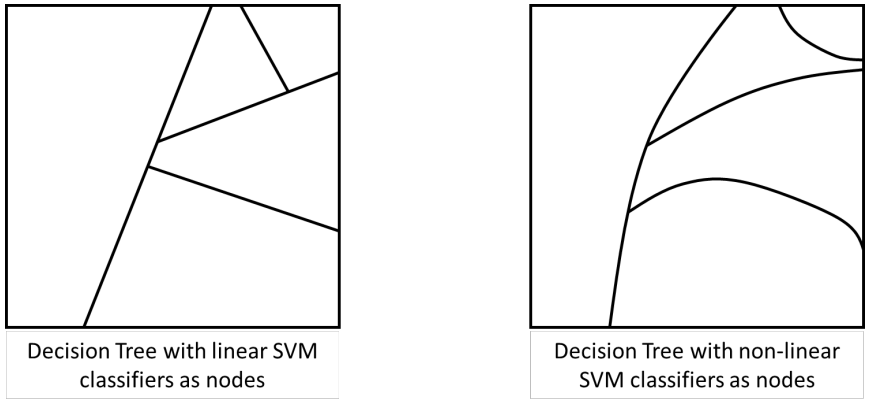


Figure 2: Separation induced by SVM-based Decision Tree: linear vs non-linear SVM classifiers.

The remainder of the paper is structured as follows: Section 2 provides a description of the proposed SVMTGP modelling strategy, with training and inference algorithms provided, separately. Section 3 briefly recaps the basics of Bayesian optimization, focusing on its specialization in the case that an SVMTGP is used as probabilistic surrogate model of the black-box objective function. Section 4 details the experimental setting, and the relevant numerical results are commented. Finally, conclusions about the advantages and limitations of the proposed approach are provided, with a perspective on ongoing and future works.

2 SVMTGP - SUPPORT VECTOR MACHINES TREED GAUSSIAN PROCESS

2.1 Training an SVMTGP and Making Inference

Denote the set of available observations (i.e., function evaluations in the case of a global optimization task) with $D_{1:n} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1,\dots,n}$, where $\mathbf{x}^{(i)} \in \Omega \cup \Re^d$ and $y^{(i)} = f(\mathbf{x}^{(i)}) + \varepsilon$ with ε a zero-mean Gaussian noise, $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. Let $\mathscr{L}$ be a list whose elements are subsets of evaluated locations belonging to a specific region in Ω. Finally, assume to have a binary decision tree where each node j refers to a specific model $h_j(x)$. In particular, $h_j(x)$ is a Support Vector Machine (SVM) classifier at each branching (intermediate) node (circle in Figure 3) or a Gaussian Process Regression (GPR) model in the case of a leaf node (square in Figure 3). Just for simplicity in the notation, we use the following enumeration for the tree's nodes – and consequently models: $h_1(x)$ is the root, $h_j(x)$ is a generic branching node with children $h_{2i}(x)$ and $h_{2i+1}(x)$. The example in Figure 3 does not show nodes indexes $\{8,9,12,13,14,15\}$ because the associated parent nodes are leaves (i.e., nodes $4,6,7$). While tree-based models having SVM classifiers as nodes have been already proposed in the ML literature, their application is mostly related to classification problems, leading to learn the tree structure depending on the misclassification error on a given dataset. On the other hand, in Treed-GP – generating axis-aligned splits – the structure is obtained via regression tree learning. Our attempt is to get the best from the two worlds by introducing a criterion to split observations at each intermediate node (classification) while function inference is done at the leaves (regression).

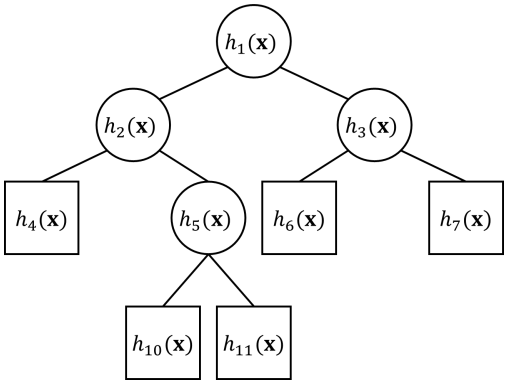


Figure 3: An example of a SVM-Treed GP.

Since we use Bayesian optimization at the core of sampling and optimization, in this paper, we propose a simple optimization-oriented splitting criterion: observations at an intermediate node are divided into two classes, i.e., observations above (+1) and below (-1) the median observation at that node. This simple

criterion leads to the following properties: (i) at each node of the partitioning tree, the observations are divided into two balanced classes, avoiding issues associated to unbalanced data which should be explicitly addressed, for instance via cost-sensitive classification; (ii) there is no need to perform operations such as pruning or rotations of the decision tree nodes, which are instead required for regression tree learning algorithms; (iii) along the BO process, new observations are collected leading to changes of the median values within – potentially all – the intermediate nodes. First, we summarize the SVM-tree GP training procedure in Algorithm 1.

Algorithm 1: Training an SVM-Treed GP

Result: $H = \{h_j(x)\}$: list of nodes (i.e., models) of the SVM-treed GP
$\mathscr{L} = \{D_{1:n}\}$, $J = \{1\}$, $\tau > d$, with τ points per subregion, and d being the dimensionality;
while $|\mathscr{L}| > 0$ **do**
  $\Delta \leftarrow \mathscr{L}[1]$ # *first element in* $\mathscr{L}$;
  $\mathscr{L} \leftarrow \mathscr{L} \setminus \mathscr{L}[1]$;
  $j \leftarrow J[1]$, $J \leftarrow J \setminus J[1]$, $m \leftarrow \underset{(\mathbf{x}^{(i)}, y^{(i)}) \in \Delta}{\text{median}} \{y^{(i)}\}$;
  $\forall i = 1, ..., |\Delta|$ set $\ell^{(i)} = \begin{cases} -1 & \text{if } y^{(i)} < m \\ +1 & \text{otherwise} \end{cases}$
  $h_j(x) \leftarrow train\_SVM\_classifier\left(\left\{(\mathbf{x}^{(i)}, \ell^{(i)})\right\}_{i=1,...,|\Delta|}\right)$;
  $\Delta^- \leftarrow \{(\mathbf{x}^{(i)}, y^{(i)}) \in \Delta : sign(h_j(\mathbf{x}^{(i)} < 0))\}$;
  $\Delta^+ \leftarrow \{(\mathbf{x}^{(i)}, y^{(i)}) \in \Delta : sign(h_j(\mathbf{x}^{(i)} > 0))\}$;
  if $\left(|\Delta^+| > \tau$ ***AND*** $|\Delta^-| > \tau\right)$ **then**
    $J \leftarrow J \cup \{2j\}$;
    $\mathscr{L} \leftarrow \mathscr{L} \cup \{\Delta^-\}$;
    $J \leftarrow J \cup \{2j+1\}$;
    $\mathscr{L} \leftarrow \mathscr{L} \cup \{\Delta^+\}$;
  else
    $h_j(x) \leftarrow fit\_GP(\Delta)$;
  end
end

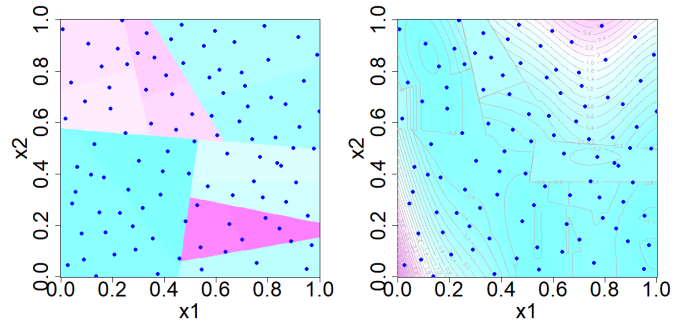
Algorithm 2 gives the overview of the inference algorithm.

Algorithm 2: Predicting by SVM-Treed GP

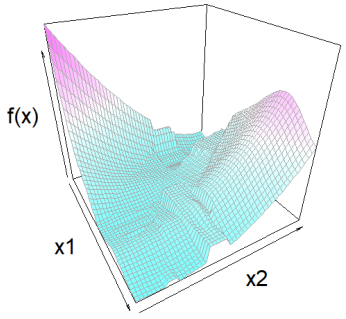
Result: $(\mu(\mathbf{x}), \sigma(\mathbf{x}))$
$\mathbf{x} \in \Omega$;
$j \leftarrow 1$;
while $h_j(\boldsymbol{x})$ *is an SVM classifier* **do**
  if $sign(h_j(\boldsymbol{x})) < 0$ **then**
    $j \leftarrow 2j$ # left child;
  else
    $j \leftarrow 2j+1$ # right child;
  end
end
$(\mu(\mathbf{x}), \sigma(\mathbf{x})) \leftarrow h_j(\mathbf{x})$ # by construction, $h_j(\mathbf{x})$ is a GPR (i.e., node j is a leaf);

2.2 An example of Learning with SVMTGP

We present an example of inference performed via an SVM-Treed-GP (Figure 4). The model has been trained on 100 function evaluations, chosen via Latin Hypercube Sampling (LHS), of the Branin rescaled test function (Picheny et al. 2013). On the left hand side of Figure 4a, we show the partition of Ω into 13 subregions according to the tree intermediate nodes (i.e., the hierarchically organized SVM classifiers). The right panel shows the mean of the GP regression models, each one fitted by using the function evaluations belonging to the associated subregion of the input space. Although this could result in large values of $\sigma(\mathbf{x})$, when few observations are in that region, this is not a drawback in the BO setting. Indeed, the acquisition function combines $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ to balance exploitation and exploration in sampling the next point. Moreover, increasing τ increases the number of minimum observations for each partition, allowing to manually reduce the predictive uncertainty of all the GPs, if needed. A 3D representation of the approximation provided by the SVM-Treed-GP is reported in Figure 4b. The discontinuities introduced by the hierarchical organization into regions of the input space are clearly visible.



(a) An example of prediction provided by an SVM-Treed-GP on the Branin rescaled test function: the separation into regions (left) and the final prediction mean (right). Dots refer to 100 locations sampled via LHS whose function evaluations have been used to train the SVM-Treed-GP model.



(b) A 3D representation of the approximation provided by the SVM-Treed-GP for the Branin rescaled test function.

Figure 4: SVM-Treed-GP model applied to the Branin rescaled test function.

3 NONSTATIONARY BAYESIAN OPTIMIZATION WITH SVMTGP

Bayesian Optimization (BO) (Shahriari et al. 2015; Frazier 2018; Archetti and Candelieri 2019) is a sample efficient and sequential method for global optimization of black-box, expensive and multi-extremal functions. In this paper, we consider, without loss of generality, the minimization setting:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\Omega} f(\mathbf{x}) \tag{1}$$

BO is based on two key components: (*i*) a *probabilistic surrogate model* approximating the objective function, depending on n function evaluations already performed, and (*ii*) an *acquisition function* (aka *utility function* or *infill criterion*) driving the selection of the next location $\mathbf{x}^{(n+1)}$ where to evaluate the objective function, depending on the current surrogate's prediction and associated uncertainty. Choosing $\mathbf{x}^{(n+1)}$ requires to solve an *auxiliary* optimization problem whose cost is usually negligible compared to that incurred for evaluating $f(\mathbf{x})$, more precisely:

$$\mathbf{x}^{(n+1)} = \arg\max_{\mathbf{x}\in\Omega} \alpha(\mathbf{x}) \tag{2}$$

with $\alpha(\mathbf{x})$ the acquisition function. Being a sequential method, BO has to choose, at each iteration, the next $\mathbf{x}^{(n+1)}$ by dealing with the *exploration-exploitation dilemma*, in order to provide global convergence. Several acquisition functions have been proposed in the literature offering different mechanisms to balance between exploration and exploitation; here we consider the (Lower) Confidence Bound, which is optimistic in face of uncertainty. Denote with $\mu(\mathbf{x})^{(n)}$ and $\sigma(\mathbf{x})^{(n)}$, respectively, the predictive mean and standard deviation, at location $\mathbf{x}$, given by the probabilistic surrogate model trained on the available set of n function evaluations (i.e., $D_{1:n}$, as denoted in the Section 2.1). Then, the Lower Confidence Bound (LCB) is computed as $\text{LCB}(\mathbf{x})^{(n)} = \mu(\mathbf{x})^{(n)} - \sqrt{\beta^{(n)}}\sigma(\mathbf{x})^{(n)}$, with a proper no-regret schedule for $\beta^{(n)}$ given in (Srinivas et al. 2012). While $\sigma(\mathbf{x})$ could be large at the boundaries of each partition, this does not necessarily lead to select a "boundary point", due to the fact that the acquisition function considers both $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$. To be compliant with notations in (2) we have to set $\alpha(\mathbf{x}) = -LCB(\mathbf{x})$.

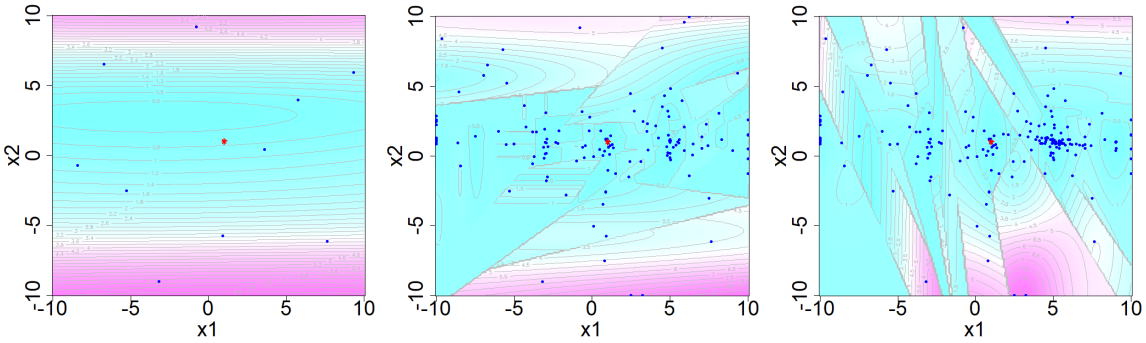
A schematic representation of the overall SVMTGP-BO algorithm is given in Algorithm 3.

Algorithm 3: SVMTGP-BO

Result: (x^+, y^+), that is the function evaluation associated to the lowest observed value;
set N as the maximum number of function evaluations;
set n_0 as the number of function evaluations required to initialize the SVMTGP;
choose $\left\{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(n_0)}\right\} \in \Omega$, for instance via Latin Hypercube Sampling;
observe $\left\{y^{(1)}, ..., y^{(n_0)}\right\}$;
store function evaluations into $D_{1:n_0}$;
$n \leftarrow n_0$;
while $n \leq N$ **do**
  $(\mu(\mathbf{x}), \sigma(\mathbf{x})) \leftarrow$ train the SVMTGP as in **Algorithm 1** and make inference as in **Algorithm 2**;
  $\mathbf{x}^{(n+1)} \leftarrow \arg\max_{\mathbf{x}\in\Omega} \left\{ - \left[\mu(\mathbf{x})^{(n)} - \sqrt{\beta^{(n)}}\sigma(\mathbf{x})^{(n)} \right] \right\}$;
  observe $y^{(n+1)} = f\left(\mathbf{x}^{(n+1)}\right) + \varepsilon$;
  $D_{1:n+1} \leftarrow D_n \cup \left\{ \left(\mathbf{x}^{(n+1)}, y^{(n+1)}\right) \right\}$;
  $n \leftarrow n + 1$;
end
$i^+ = \arg\min_{i=1:N} \left\{ y^{(i)} \right\}$;
$(x^+, y^+) = D_{i^+}$

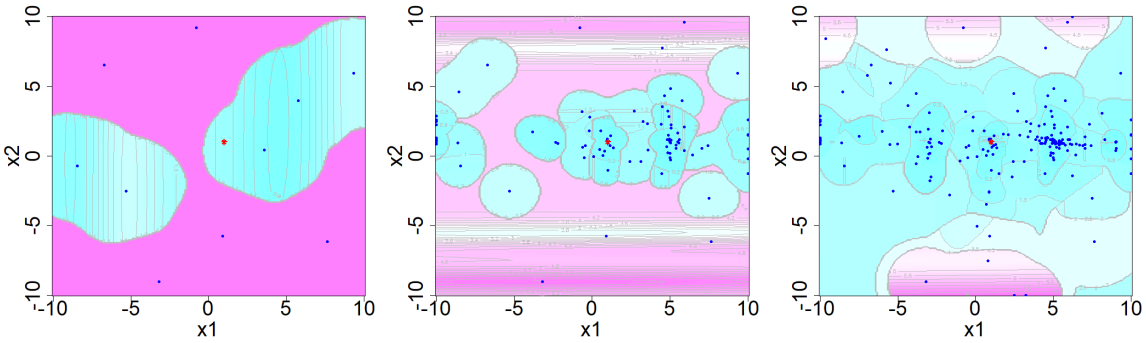
It is important to highlight that the overall BO framework holds independently from the specific acquisition function and, even more importantly in the scope of this paper, the framework is robust to the specific surrogate model being used. Although GP regression is the most common choice (GP-based BO or GP-BO), alternative modelling strategies have been proposed, with Random Forest (RF) regression among the most successful especially when the considered input space Ω is complex, in the sense that it is spanned by mixed (i.e., continuous, integer, nominal) and/or conditional decision variables. A relevant example of RF-based BO is given by Automated Machine Learning (AutoML) (Hutter et al. 2019; He et al. 2021). These cases are becoming increasingly relevant: often complex dynamical systems behavior is the result of interplay of discrete variables (e.g., switching dynamics based on a binary condition) and continuous variables that for example describe the differentiable dynamics within a specific class. The difficulty is that the way the discrete and continuous components interact is not known, hence, the surrogate implicitely plays the role of capturing the interaction among these different components. Cyberphysical systems represent a perfect application for this kind of problem, but also biological applications such as molecular design where residues concentrations, and orientation, interplay with residues choices and bonds among residues (discrete).

In our approach, the Gaussian process is replaced with our SVMTGP with no substantial changes to the Bayesian optimization framework, resulting into a *nonstationary GP-based BO* approach. An important aspect is that the SVMTGP is learned at each BO iteration, leading to a completely different division of the search space and nonstationary regression model from one iteration to the next, depending on the observations sequentially collected (examples are reported in Figures 5-6). This is quite different from partitioning based algorithm, such as PBnB (Zabinsky and Huang 2020), where each region is iteratively refined by further splits, but never changing the axis-aligned hyperrectangles defined at previous iterations.



(a) Partition after 10 evaluations. (b) Partition after 40 evaluations. (c) Partition after 100 evaluations.

Figure 5: Partition obtained with linear SVM.



(a) Partition after 10 evaluations. (b) Partition after 40 evaluations. (c) Partition after 100 evaluations.

Figure 6: Partition obtained with non linear SVM.

Due to the nonstationarity of the SVMTGP, and the hierarchical splitting it performs, the resulting final model is not continuous, as well as the associated acquisition function. Thus, analogously to the RF-BO, derivative free methods are used to solve (2), instead of gradient-based algorithms typically adopted in GP-BO. It is important to remark that the cost for evaluating the acquisition function is negligible compared to $f(\mathbf{x})$, so (Adaptive) Random Search as well as Evolutionary approaches can be successfully applied, taking advantage of their amenable parallelism and the low cost of evaluation of the infill function.

4 PRELIMINARY ANALYSIS

All the experiments have been performed on a Microsft Azure virtual machine, H8 (High Performance Computing family) Standard with 8 vCPUs, 56 GB of memory, Ubuntu 16.04.6 LTS. Code has been developed in R and it is freely available at: https://github.com/acandelieri/SVM-Treed-GP. Section 4.1 describes the experimental setting of our study, specifically (*i*) the set of 2-dimensional test functions

considered, (*ii*) the BO's settings with respect to the different models adopted (i.e., stationary GP, TreedGP and the proposed SVMTGP), and (*iii*) the computational settings. Section 4.2 discusses the obtained results.

4.1 Experimental Settings

Test Functions. We considered eight 2D test functions, with different levels of homogeneity and, consequently, difficulty (i.e., simple and hard). While in general the concept of function homogeneity is related to its multiplicative scaling behaviour, when Gaussian processes are used as processes, we are specifically interested in the ability of the hyperparameters in a subregion of the input to reflect the behavior of the function in distant regions. Equivalently, the homogeneity of the function is reflected by the ability of a single covariance function to reflect the output landscape. Two examples are shown in Figure 7. All the test functions are available at https://www.sfu.ca/~ssurjano/optimization.html and http://infinity77.net/global_optimization/index.html. All the functions are considered noise-free in the experiments.

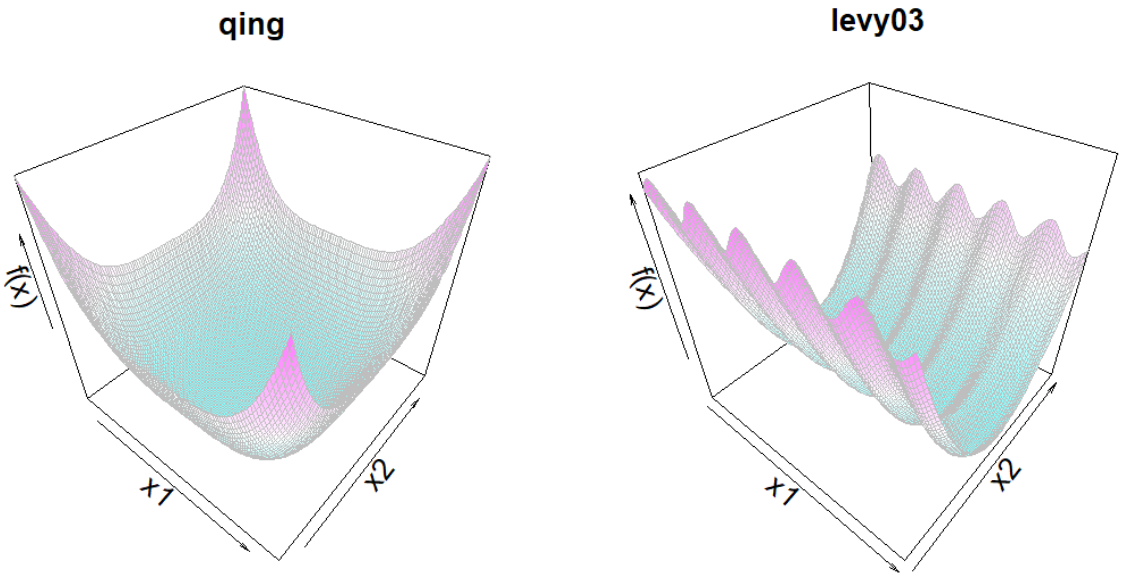


Figure 7: Examples of homogeneous (left) and non-homogeneous (right) test functions.

- *Branin rescaled* (simple): $f(\mathbf{x}) = \frac{1}{51.95}\left[\left(\bar{x}_2 - \frac{5.1\bar{x}_1^2}{4\pi^2} + \frac{5\bar{x}_1}{\pi} - 6\right)^2 + \left(10 + \frac{10}{8\pi}\right)\cos(\bar{x}_1) - 44.81,\right]$, where $\bar{x}_1 = 15x_1 - 5$ and $\bar{x}_2 = 15x_2$. $\Omega = [0,1]^2$, $f(\mathbf{x}^*) = 0.397887$ $\mathbf{x}^* = (-\pi, 12.275), (\pi, 2.275)$ and $(9.24478, 2.475)$;
- *Cosine mixture* (simple): $f(\mathbf{x}) = -0.1\sum_{i=1}^{2}\cos(5\pi x_i) - \sum_{i=1}^{2} x_i^2$, where $\Omega = [-1,1]^2$, $f(\mathbf{x}^*) = -0.2$ and $\mathbf{x}^* = (0,0)$;
- *Levy03* (hard): $f(\mathbf{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2[1 + 10\sin^2(\pi w_{i+1})] + (w_d - 1)^2$, where $w_i = 1 + \frac{x_i - 1}{4}$, $\Omega = [-10,10]^2$, $f(\mathbf{x}^*) = 0$ and $\mathbf{x}^* = (1,1)$;
- *Qing* (simple): $f(\mathbf{x}) = \sum_{i=1}^{2}(x_i^2 - i)^2$, where $\Omega = [-500,500]^2$, $f(\mathbf{x}^*) = 0$ and $\mathbf{x}^* = (\pm\sqrt{1}, \pm\sqrt{2})$;
- *Rosenbrock modified* (hard): $f(\mathbf{x}) = 74 + 100(x_2 - x_1^2)^2 + (1 - x_1)^2 - 400\exp - \frac{(x_1+1)^2 + (x_2+1)^2}{0.1}$, where $\Omega = [-2,2]^2$, $f(\mathbf{x}^*) = 34.37$ and $\mathbf{x}^* = (-0.9, -0.95)$;

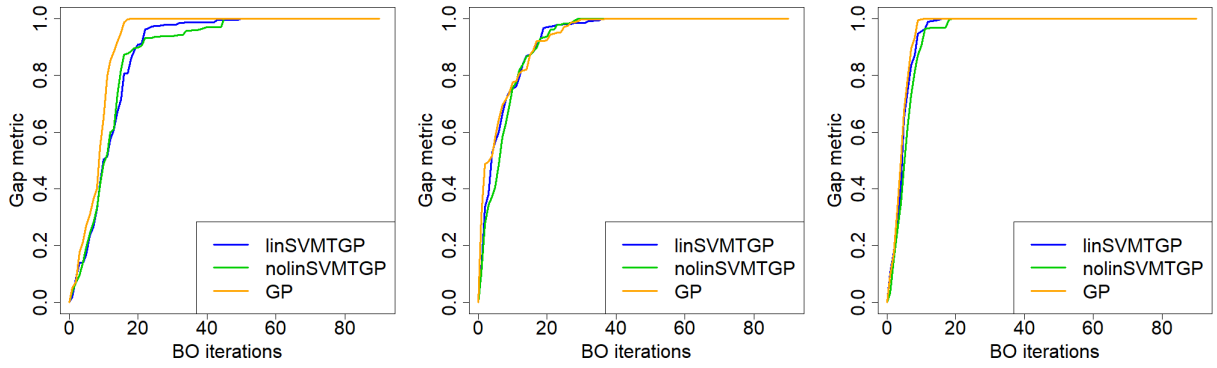
- *Tripod* (hard): $f(\mathbf{x}) = p(x_2)[1+p(x_1)] + x_1 + 50p(x_2)[1-2p(x_1)] + x_2 + 50[1-2p(x_2)]$, where $p(x_i) = 1$ **iff** $x_i >= 0$, $p(x_i) = 0$ otherwise, $\Omega = [-100,100]^2$, $f(\mathbf{x}^*) = 0$ and $\mathbf{x}^* = (0,-50)$;
- *Ursem01* (simple): $f(\mathbf{x}) = -\sin(2x_1 - 0.5\pi) - 3\cos(x_2) - 0.5x_1$, where $\Omega = [-2.5,3] \times [-2,2]$, $f(\mathbf{x}^*) = -4.81680$ and $\mathbf{x}^* = (1.69714, 0)$;
- *UrsemWaves* (hard): $f(\mathbf{x}) = -0.9x_1^2 + (x_2^2 - 4.5x_2^2)x_1x_2 + 4.7\cos[2x_1 - x_2^2(2+x_1)]\sin(2.5\pi x_1)$, where $\Omega = [-0.9,1.2] \times [-1.2,1.2]$, $f(\mathbf{x}^*) = -8.5536$ and $\mathbf{x}^* = (1.2,1.2)$.

Bayesian Optimization setting In the following, we provide the information related to the compared approaches and their settings:

- Initial set of function evaluations (aka initial design): 10 samples through LHS-maximin;
- Total budget: 100 function evaluations overall (including the initial design)
- Approaches: GP-BO (i.e., no partitioning), and the proposed SVMTGP-BO
- 2 types of SVMTGP-BO: (i) with linear SVM classifiers, namely *linSVMTGP-BO*; (ii) with nonlinear SVM classifiers, namely *nolinSVMTGP-BO*. SVM classifiers kernel: Radial Basis Function (RBF) $k_{RBF}(\mathbf{x},\mathbf{x}') = e^{-\gamma||\mathbf{x}-\mathbf{x}'||^2}$ (default value: $\gamma = 1/|\Omega|$)
- GP's kernel: Squared Exponential (SE) $k_{SE}(\mathbf{x},\mathbf{x}') = e^{-\frac{||\mathbf{x}-\mathbf{x}'||}{\theta}}$
- Macro-replications: 30 for each test function and each approach (given a run, the initial design is the same for all compared methods)
- metric: (i) Best seen: $y^{+(n)} = \min\{y^{(i)}\}_{1:n}$ (for tabular results) (ii) Gap metric: $G^{(n)} = (y^{+(n_0)} - y^{+(n)})/(y^{+(n_0)} - y^*)$ (for charts), with $y^{+(n_0)}$ the best so far and y^* the actual optimum
- Statistical test for comparison: Mann-Whitney's U test, the hypothesis is that the best seen distributions of two approaches are similar with p-value=0.05.

4.2 Results Discussion

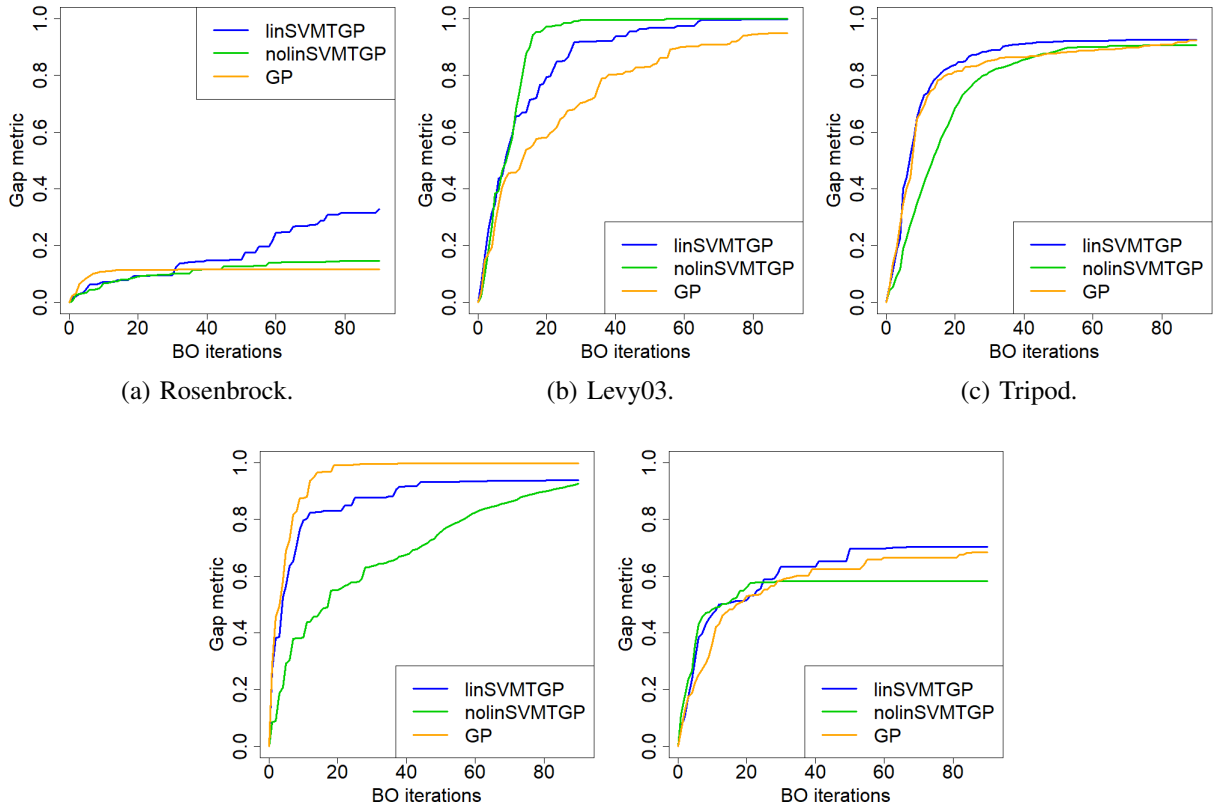
Figures 8-9 compare the average gap metrics of the three approaches over function evaluations.



(a) Rescaled Branin. (b) Cosine Mixture. (c) Ursem01.

Figure 8: Cases with no difference between approaches.

From Figure 9, it is possible to notice that GP-BO outperforms the proposed SVMTGP-BO approach only on the Qing test function, while (at least one the implementations of) SVMTGP-BO outperforms GP-BO on non-homogeneous functions, specifically, *Rosenbrock_modified*, *Levy03*, and *Usermwaves*. On the remaining test functions, the three approaches are equivalent in terms of gap metric. It is however apparent that the complexity of non linear partitioning not always results in increased performance. Nonetheless, the new algorithms show clear advantage in performance when applied to the most complex functions (Levy03 and modified Rosenbrock).



(a) Rosenbrock. (b) Levy03. (c) Tripod.

(d) Qing. (e) UrsemWaves.

Figure 9: Cases with difference among approaches.

Table 1: Best seens of the approaches: average and standard deviation over 30 independent runs. Symbols: ▼for significantly underperforming, ▲for significantly outperforming and △for outperforming on average.

Test Function	**GP-BO**	**linSVMTGP-BO**	**nolinSVMTGP-BO**
Branin rescaled	$-1.0474(0.0)$	$-1.0474(0.0)$	$-1.0474(0.0)$
Cosine mixture	$-0.2(0.0)$	$-0.2(0.0)$	$-0.2(0.0)$
Rosenbrock modified	$74.0002(0.00032)^{\blacktriangledown}$	$64.5709(15.72908)$	$72.7417(7.30998)$
Levy03	$0.0029(0.00747)^{\blacktriangledown}$	$0.0002(0.00046)$	$0.0(0.0)$
Tripod	$1.0198(0.71705)$	$0.9337(0.78939)^{\triangle}$	$1.1362(0.77615)$
Qing	$16666.06(24163.19)^{\blacktriangle}$	$528872.3(1703224)$	$27505369(72340940)$
Ursem01	$-4.8168(0.0)$	$-4.8168(0.0)$	$-4.8168(0.0)$
Ursemwaves	$-7.4906(1.03481)$	$-7.7367(0.93083)^{\triangle}$	$-7.2739(1.27227)$

Table 1 summarizes the results, reporting the best seen, y^+, at the end of the optimization process in terms of mean (standard deviation) on 30 independent runs. *Branin rescaled*, *Cosine mixture* and *Ursem01*, GP-BO and SVMTGP-BO (in both the linear and the nonlinear implementations) perform the same. These are the simplest test functions among those considered, and all the three BO approaches converge to the optimum. On 2 out of 8 test functions, namely *Tripod* and *Ursemwave*, linSVMTGP outperforms the other two approaches, on average, but this difference in performances is not statistically significant (p-value > 0.05). Then, on 2 out of 8 test functions, namely *Rosenbrock modified* and *Levy03* GP-BO significantly underperforms the two SVMTGP-BO implementations (p-value≤ 0.05). These are,

basically the two *hardest* – in terms of stationarity – test functions among those considered. Finally, GP-BO significantly outperforms the two SVMTGP-BO implementations only on the *Qing* test function (p-value ≤ 0.05). The intuitive reason why SVMTGP-BO performs worse on this test function is that the SVMTGP model introduces discontinuities, due to the underlying partitioning process, which are end up being detrimental in cases with high-order smooth functions such as the Qing case.

SVMTGP-BO does not underperform BO while outperforms it on nonstationary test functions.

5 CONCLUSIONS

We propose, for the first time, the algorithm SVMTGP-BO. SVMTGP-BO addresses an important challenge arising when black box optimization methods are applied to general non linear non convex problems that may exhibit heterogeneity of the reward. In such cases, the kernels normally used to construct the predictor for the function will typically show degrading performance due to the inability of the correlation structure to capture the landscape. Nonetheless, several applications require to consider rewards with heterogeneity. As an example, dynamical systems with switching dynamics will typically exhibit sharp transitions of the reward for inputs in different subregions. Our preliminary analysis shows that partitioning tends to improve the performance. While the shape of the subregions impacts such performance, partitioning-based approaches beat vanilla BO, especially in the case of non-stationary, non-smooth functions.

More analysis is required to understand how to automate the choice of the partitioning model, and scaling to higher dimensions. In fact, the cost of estimating the partition may not be worth the accuracy gain. We are currently investigating different criteria for partitioning, and focusing on examples where a "vanilla" Gaussian process fails to capture the reward with focus on real applications such as automated testing of cyberphisical systems. Also, an extended testing bench of competing algorithms is being developed.

ACKNOWLEDGMENTS

We greatly acknowledge the DEMS Data Science Lab, Department of Economics Management and Statistics (DEMS), University of Milano-Bicocca, for supporting this work by providing computational resources. The research presented in this work was partially supported under grants NSF#1829238-#2007861-#2026860.

REFERENCES

Archetti, F., and A. Candelieri. 2019. *Bayesian optimization and data science*. Springer.

Chang, F., C.-Y. Guo, X.-R. Lin, and C.-J. Lu. 2010. "Tree decomposition for large-scale SVM problems". *The Journal of Machine Learning Research* 11:2935–2972.

Chen, G., and Z. Ge. 2019. "SVM-tree and SVM-forest algorithms for imbalanced fault classification in industrial processes". *IFAC Journal of Systems and Control* 8:100052.

Civera, M., G. Boscato, and L. Z. Fragonara. 2020. "Treed Gaussian Process for manufacturing imperfection identification of pultruded GFRP thin-walled profile". *Composite Structures* 254:112882.

Civera, M., C. Surace, and K. Worden. 2017. "Detection of cracks in beams using treed Gaussian processes". In *Structural Health Monitoring & Damage Detection, Volume 7*, 85–97. Springer.

Cohen, D. A., and E. A. Fernández. 2012. "SVMTOCP: A binary tree base SVM approach through optimal multi-class binarization". In *Iberoamerican Congress on Pattern Recognition*, 472–478. Springer.

de Boves Harrington, P. 2015. "Support vector machine classification trees". *Analytical chemistry* 87(21):11065–11071.

Fei, B., and J. Liu. 2006. "Binary tree of SVM: a new fast multiclass training and classification algorithm". *IEEE transactions on neural networks* 17(3):696–704.

Frazier, P. I. 2018. "Bayesian optimization". In *Recent Advances in Optimization and Modeling of Contemporary Problems*, 255–278. Institute For Operations Research and Management Sciences.

Gramacy, R. B. 2020. *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. Chemical Rubber Company Press.

Gramacy, R. B. et al. 2007. "tgp: an R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed Gaussian process models". *Journal of Statistical Software* 19(9):6.

Gramacy, R. B., and D. W. Apley. 2015. "Local Gaussian process approximation for large computer experiments". *Journal of Computational and Graphical Statistics* 24(2):561–578.

Gramacy, R. B., and S. Le Digabel. 2011. *The mesh adaptive direct search algorithm with treed Gaussian process surrogates*. Groupe d'études et de recherche en analyse des décisions.

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

He, X., K. Zhao, and X. Chu. 2021. "AutoML: A Survey of the State-of-the-Art". *Knowledge-Based Systems* 212:106622.

Hebbal, A., L. Brevault, M. Balesdent, E.-G. Talbi, and N. Melab. 2021. "Bayesian optimization using deep Gaussian processes with applications to aerospace system design". *Optimization and Engineering* 22(1):321–361.

Higdon, D., J. Swall, and J. Kern. 1999. "Non-stationary spatial modeling". *Bayesian statistics* 6(1):761–768.

Hutter, F., L. Kotthoff, and J. Vanschoren. 2019. *Automated machine learning: methods, systems, challenges*. Springer Nature.

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

Marmin, S., D. Ginsbourger, J. Baccou, and J. Liandrat. 2018. "Warped gaussian processes and derivative-based sequential designs for functions with heterogeneous variations". *SIAM/ASA Journal on Uncertainty Quantification* 6(3):991–1018.

Picheny, V., T. Wagner, and D. Ginsbourger. 2013. "A benchmark of kriging-based infill criteria for noisy optimization". *Structural and Multidisciplinary Optimization* 48(3):607–626.

Rasmussen, C. E., and C. K. Williams. 2006. "Gaussian processes for machine learning".

Schmidt, A. M., and A. O'Hagan. 2003. "Bayesian inference for non-stationary spatial covariance structure via spatial deformations". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 65(3):743–758.

Shahriari, B., K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. 2015. "Taking the human out of the loop: A review of Bayesian optimization". *Proceedings of the IEEE* 104(1):148–175.

Shekhar, S., and T. Javidi. 2019. "Multiscale gaussian process level set estimation". In *The 22nd International Conference on Artificial Intelligence and Statistics*, 3283–3291. Proceedings of Machine Learning Research.

Srinivas, N., A. Krause, S. M. Kakade, and M. W. Seeger. 2012. "Information-theoretic regret bounds for gaussian process optimization in the bandit setting". *IEEE Transactions on Information Theory* 58(5):3250–3265.

Zabinsky, Z. B., and H. Huang. 2020. "A partition-based optimization approach for level set approximation: Probabilistic branch and bound". In *Women in Industrial and Systems Engineering*, 113–155. Springer.

Zabinsky, Z. B., G. Pedrielli, and H. Huang. 2019. "A Framework for Multi-fidelity Modeling in Global Optimization Approaches". In *International Conference on Machine Learning, Optimization, and Data Science*, 335–346. Springer.

AUTHOR BIOGRAPHIES

ANTONIO CANDELIERI, Ph.D., is Assistant Professor for the Department of Economics, Management and Statistics at the University of Milano-Bicocca, Italy. His research activities are focused on Machine Learning and Bayesian Optimization. His email address is antonio.candelieri@unimib.it. His website is https://www.unimib.it/antonio-candelieri.

GIULIA PEDRIELLI, Ph.D., is Assistant Professor for the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University. She develops her research activity in the area of stochastics and simulation with a particular interest in simulation based optimization. Her e-mail address is Giulia.Pedrielli@asu.edu. Her website is https://www.gpedriel.com/.