2024 32nd Mediterranean Conference on Control and Automation (MED)
June 11-14, 2024 | Chania, Crete, Greece

2024 32nd Mediterranean Conference on Control and Automation (MED) | 979-8-3503-9544-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/MED61351.2024.10566222

Scalable approximate optimization of objective functions represented by Random Forests

Marco Leonesio[1] and Lorenzo Fagiano[2]

Abstract— **The problem of global optimization of an objective function represented by a Random Forest (RF) is considered. A method to obtain an approximate solution at low computational complexity is proposed, resorting to the inherent structure of an RF, which is a non-parametric model that partitions the feature space in convex polytopes according to the training data. The approach selects the optimal solution inside the polytopes corresponding to the best data points. It is shown that the proposed approximate method is significantly more efficient, thus applicable at large scale, than extensive global search algorithms, such as gridding and Mixed Integer Linear Programming (MILP), which in turn provide exact solutions. The efficiency and sub-optimality of the approach are evaluated on RFs trained on a dataset generated by sampling a bivariate, discontinuous and non-convex benchmark function from the literature.**

I. Introduction

Data-driven optimization can be pursued following a model-free approach [1], exploiting reinforcement learning [2], or by identifying black-box or gray-box models as surrogate objective functions in optimization programs [3]–[5]. Focusing on the latter case, in this study, we are interested in finding an efficient approach to minimize objective functions represented by Random Forests (RFs).

An RF is an example of an ensemble method that combines predictions from several machine learning models. In the case of RFs, these algorithms are Regression Trees (RTs) (see Breiman's seminal article [6] and the survey [7]). The use of an RF in regression tasks offers numerous benefits. RFs excel in capturing unknown non-linear, discontinuous and intricate feature interactions with minimal feature engineering. Besides, they are generally characterized by robust predictive capabilities, even in the case of a relatively small training set. In particular, an RF avoids over-fitting problems, as it can be viewed as a non-parametric model that behaves like an adaptive nearest-neighbor prediction method [8]. These qualities have contributed to RFs popularity and extensive utilization in machine learning [9].

A large amount of literature deals with RTs and RFs training, seeking the smallest generalization error possible (optimal estimation). For this purpose, many different algorithms have been proposed, such as ID3 [10], C4.5 [11], and CART [12]. Indeed, our analysis does not tackle the issue of optimizing over a known function by a proper surrogate RF, nor do we investigate if an RF is the best model for fitting experimental data. We assume that a given RF already represents a suitable cost function to be minimized, and want an approach to do it efficiently, with a trade-off between accuracy and computational cost. Namely, our goal is focused on choosing the best "input feature vector" yielding the unconstrained global minimum of the RF within a short time and with a limited sub-optimality.

Even though the goal above can be achieved by a standard general approach for global optimization [13]–[15], a "dedicated" approximate method is herein proposed, with higher efficiency. It exploits the unique structure of an RF, whose RTs induce a countable partition of the input space. Relying on the same observation, other authors demonstrated that an RF minimization problem can be formulated as a Mixed Integer Linear Program (MILP), whose solution is the exact global optimum [16]. However, depending on the size of the RF, the computational effort to solve the MILP becomes quickly unaffordable in most applications. The loophole proposed in [16] consists of optimizing a subset of RF trees after demonstrating that the consequent sub-optimality gap is usually limited and under control.

In the described framework, a new approximate approach is proposed for unconstrained RF minimization, based on the direct identification of the partition sets in the feature space that may contain the global minimum. It is shown that the complexity of the proposed algorithm is linear with the training set cardinality and number of trees in the RF, while in MILP it increases exponentially. On the other side, under weak assumptions, the payback in terms of sub-optimality is likely limited.

II. Problem statement and preliminaries

We consider the following optimization problem:

$$y^* := \min_{\mathbf{x}} R(\mathbf{x}), \quad \mathbf{x} \in \Omega \tag{1}$$

where $R : \Omega \to \mathbb{R}$ is a RF and $\mathbf{x}$ is a feature vector belonging to the input space $\Omega \subseteq \mathbb{R}^d$. Next, we provide details on RFs and solution methods from the literature, which are instrumental to introduce our approach.

A. Random forest properties

An RF regressor is a set of Regression Trees (RTs) whose outputs are averaged to compute the overall prediction. An

[1]Marco Leonesio is a researcher at the Institute of Intelligent Industrial Technologies and Systems for Advanced Manufacturing - National Research Council of Italy (STIIMA - CNR), via Corti 12, 20133 Milano, Italy marco.leonesio@stiima.cnr.it

[2]Lorenzo Fagiano is a professor at Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano Piazza Leonardo da Vinci 32, 20133 Milano, Italy lorenzo.fagiano@polimi.it

This research has been funded by Regione Lombardia and Tech Fast Innovation program (CUP E79J22000010007) and by Fondazione Cariplo under grant n. 2022-2005, project "NextWind - Advanced control solutions for large scale Airborne Wind Energy Systems".

RT provides a regression model in the form of a tree structure: the input (or feature) vector is routed along the tree branches to the final leaf containing the output (or "target") result. In the training phase, a splitting procedure divides the input feature vectors into partitions defined by subsequent inequalities on any single feature. At the end of the procedure (leaf level), each partition set contains a few feature vectors (in our case, only one) belonging to the dataset, and the leaf is mapped to the corresponding average target value.

Let $\mathcal{D} := \{(\mathbf{x}_i, y_i) \in \Omega \times \mathbb{R},\ i = 1, 2, ..., N\}$ be a training dataset of N samples, composed of pairs of features $\mathbf{x}_i \in \Omega$ and targets $y_i \in \mathbb{R}$, where $\Omega \subseteq \mathbb{R}^d$ is the *input space* of dimension d. We denote with $x_i^{(j)}$ the j^{th} entry of the feature vector $\mathbf{x}_i$. According to the CART algorithm [12], the steps to construct an RF can be outlined as follows:

Repeat for $j = 1, 2, ..., N_T$ (number of trees):

1) Sample with equal-probability draws with-replacement to obtain a new bootstrap training set $\mathcal{D}_j := \{(\mathbf{x}_i, y_i),\ i \in \{1, 2, ..., N\}\}$ (sample repetions are possible).
2) Create a RT (denoted by T_j) by recursively splitting the set $\mathcal{D}_j$ as follows:
   a) Start from the *root node* $\mathcal{N} \equiv \mathcal{D}_j$.
   b) Draw a random subset $\mathcal{S}$ of $p_1 \leq d$ features (p_1 is a model hyperparameter also known as *features try*);
   c) Split the node $\mathcal{N}$ in 2 sets, $\mathcal{N}_1(j, h)$ and $\mathcal{N}_2(j, h)$, on the basis of a feature index $\ell \in \mathcal{S}$, and a threshold $h \in \mathbb{R}$ as follows:
   $$\begin{aligned} \mathcal{N}_1(\ell, h) &:= \{(\mathbf{x}_i, y_i) \in \mathcal{N} : x^{(\ell)} \leq h\} \\ \mathcal{N}_2(\ell, h) &:= \{(\mathbf{x}_i, y_i) \in \mathcal{N} : x^{(\ell)} > h\} \end{aligned}$$
   where ℓ and h are chosen by minimizing the sum of the output variances obtained on the two corresponding partitions:
   $$(\ell, h) = \arg \min_{\bar{\ell} \in \mathcal{S}, \bar{h} \in \mathbb{R}} \sum_{k=1}^{2} \sum_{i \in \mathcal{N}_k(\bar{\ell}, \bar{h})} (y_i - \bar{y}_k)^2 \quad (2)$$
   where $\bar{y}_k$ is the average value of the targets in the partition $\mathcal{N}_k$.
   d) For each new subnode $\tilde{\mathcal{N}}$ with more than a given number of samples denoted by p_2 (a hyperparameter called *node size*) and with some variation in the associated target values (namely, the variance must not be 0) and features (the $\mathbf{x}_i$ must be distinct), repeat the steps from 2b to 2d with $\tilde{\mathcal{N}}$ instead of $\mathcal{N}$. Any subnode with no variation in targets or feature vectors is not subject to a further split procedure, and it is called *leaf* of the tree T_j.
3) Create the set of trees $R := \{T_1, ..., T_j, ..., T_{N_T}\}$, that constitutes the random forest. The number of trees N_T is the 3rd hyperparameter of the model.

The tree prediction $\hat{y}_j := T_j(\mathbf{x})$ is produced by applying, to the feature vector $\mathbf{x}$, the splitting rules defined for T_j to identify one leaf: then, $\hat{y}_j$ is the average of the leaf target values. If the node size equals 1, the leaf contains a unique target value. In this paper, we assume $p_2 = 1$, such that there exists a bijective function that maps distinct training samples to leaf nodes. Moving now to the RF, its prediction $\hat{y}$ for the same $\mathbf{x}$ is given by:

$$\hat{y} = R(\mathbf{x}) = \frac{1}{N_T} \sum_{j=1}^{N_T} \hat{y}_j = \frac{1}{N_T} \sum_{j=1}^{N_T} T_j(\mathbf{x}) \quad (3)$$

Namely, the random forest averages the output of its regression trees.

Another way to consider a regression tree is to observe that T_j defines a labeled partition over the input space Ω. In fact, T_j associates each vector $\mathbf{x}_i$ in the dataset to a convex polytopic set $X_{ji} \subseteq \Omega$, defined by the intersection of all the half spaces identified by the bounds created during the splitting procedure. Using the described CART algorithm, the derived partitions are d-orthotopes, because each split represents a plane perpendicular to one coordinate direction in $\mathbb{R}^d$. Formally, for each $j = 1, \dots, N_T$, the partition $\mathcal{P}_j$ induced by the tree T_j, trained on $\mathcal{D}_j := \{(\mathbf{x}_i, y_i)\}$, can be written in the form:

$$\mathcal{P}_j := \{X_{ji}, y_i\},\ i = 1, 2, ..., N \quad (4)$$

such that

$$\mathbf{x}_i \in X_{ji} \subset \Omega\, \forall i \quad \text{and} \quad X_{jk} \cap X_{jr} = \emptyset \quad \forall k \neq r \quad (5)$$

Then, each RT can be seen as a function $T_j : \Omega \to \mathbb{R}$ that returns the output estimate $\hat{y}_j$ corresponding to a given vector $\mathbf{x}$ as follows:

$$\hat{y}_j = T_j(\mathbf{x}) = y_k \ : \ \mathbf{x} \in X_{jk},\ (X_{jk}, y_k) \in \mathcal{P}_j \quad (6)$$

This implies that T_j is a piecewise constant function, since $T_j(\mathbf{x}_i) = T_j(\mathbf{x}_\ell) = y_k\, \forall \mathbf{x}_i, \mathbf{x}_\ell \in X_{jk}$.

Note that the CART algorithm returns different trees T_j, $j = 1, \dots, N_T$ (with the associated partitions $\mathcal{P}_j$) from the same $\mathcal{D}$, since the training process is stochastic, because of both the bootstrap sampling of the training sets $\mathcal{D}_j$, and the random selection of the features' subset used in nodes split. Therefore, in general, for each data point $\mathbf{x}_i$, $i = 1, \dots, N$, the sets X_{ji}, $j = 1, \dots, N_T$ are different, depending on the tree index j. For the sake of analyzing the RF, we are interested in the intersection $\bigcap_{j=1}^{N_T} X_{ji}$. The set $X_{ji} \cap X_{ki}$ is surely non-null when both $\mathcal{D}_j$ and $\mathcal{D}_k$ contain the data point $(\mathbf{x}_i, y_i)$, while this is generally not true for trees that have been trained without such a point. Compactly, we can state:

$$(\mathbf{x}_i, y_i) \in (\mathcal{D}_j \cap \mathcal{D}_k) \Rightarrow X_{ji} \cap X_{ki} \neq \emptyset \quad (7)$$

$$(\mathbf{x}_i, y_i) \notin (\mathcal{D}_j \cap \mathcal{D}_k) \nRightarrow X_{ji} \cap X_{ki} = \emptyset \quad (8)$$

An example of RF partition for $d = 2$ (input space dimension) and $N_T = 2$ (number of trees) is illustrated in Fig. 1. Different possible cases can be observed: an intersection containing a data point belonging to the training set of both trees (case 1); an intersection of sets, related to different

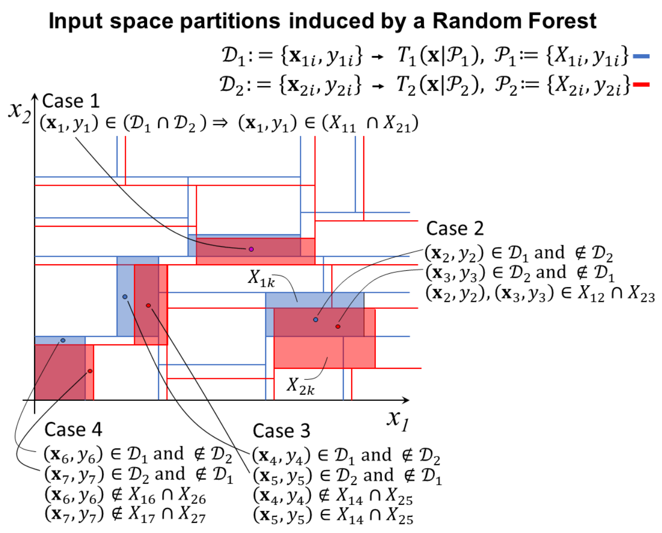
Input space partitions induced by a Random Forest
$\mathcal{D}_1 := \{\mathbf{x}_{1i}, y_{1i}\} \rightarrow T_1(\mathbf{x}|\mathcal{P}_1),\ \mathcal{P}_1 := \{X_{1i}, y_{1i}\}$
$\mathcal{D}_2 := \{\mathbf{x}_{2i}, y_{2i}\} \rightarrow T_2(\mathbf{x}|\mathcal{P}_2),\ \mathcal{P}_2 := \{X_{2i}, y_{2i}\}$
x_2
Case 1
$(\mathbf{x}_1, y_1) \in (\mathcal{D}_1 \cap \mathcal{D}_2) \Rightarrow (\mathbf{x}_1, y_1) \in (X_{11} \cap X_{21})$
Case 2
$(\mathbf{x}_2, y_2) \in \mathcal{D}_1$ and $\notin \mathcal{D}_2$
$(\mathbf{x}_3, y_3) \in \mathcal{D}_2$ and $\notin \mathcal{D}_1$
$(\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3) \in X_{12} \cap X_{23}$
X_{1k}
X_{2k}
x_1
Case 4
$(\mathbf{x}_6, y_6) \in \mathcal{D}_1$ and $\notin \mathcal{D}_2$
$(\mathbf{x}_7, y_7) \in \mathcal{D}_2$ and $\notin \mathcal{D}_1$
$(\mathbf{x}_6, y_6) \notin X_{16} \cap X_{26}$
$(\mathbf{x}_7, y_7) \notin X_{17} \cap X_{27}$
Case 3
$(\mathbf{x}_4, y_4) \in \mathcal{D}_1$ and $\notin \mathcal{D}_2$
$(\mathbf{x}_5, y_5) \in \mathcal{D}_2$ and $\notin \mathcal{D}_1$
$(\mathbf{x}_4, y_4) \notin X_{14} \cap X_{25}$
$(\mathbf{x}_5, y_5) \in X_{14} \cap X_{25}$

Fig. 1. Example of partitions induced by an RF in the input space. Each tree induces its own partition sets containing the feature vector associated with each training set sample.

samples, that contains both samples (case 2); an intersection that contains only one of the two samples (case 3); finally an intersection containing no samples (case 4). Note that the RF output is piecewise constant over each intersection, being it the average of the output values pertaining to the sets that generated it (this is evident also in Fig.3 of the case study section).

B. Mixed Integer reformulation

Consider now problem (1). Is it possible to conceive an efficient minimization algorithm that exploits the structure of the RF, described above? In [16], the authors demonstrated that problem (1) can be formulated as a Mixed Integer Linear Programming (MILP), whose solution can be computed using the Bender cuts method or Interior Point method. However, the size of the MILP grows very quickly with the number of samples, input space dimension, and the number of trees in the RF. In fact, it can be shown that the number of real variables is d and that of integer (Boolean) auxiliary variables is:

$$N_v = \sum_{j}^{N_T} N_{nodesT_j} \tag{9}$$

where N_{nodesT_j} is the number of nodes of the tree T_j in the RF. In turn, N_{nodesT_j} depends on the data set cardinality N. If the node size $p_2 = 1$ (see point 2-d in the CART Algorithm in Section II-A), it can be shown that:

$$N_{nodesT_j} = 2N_j - 1 \tag{10}$$

with N_j being the number of distinct samples in the training set $\mathcal{D}_j$. Since the bootstrap procedure is done with equal-probability draws, for a sufficiently large dataset it can be shown that $N_j \approx 0.63N$ (see [17]). For instance, for an RF of 100 RTs, trained with 1000 samples, and an input space dimension of 2, the number of integer MILP variables is about 133900.

At the same time, solving globally a MILP is NP-hard: it generally requires exponential time by branch and bound. As a matter of fact, an exact solution to the original problem becomes intractable for common RFs. The loophole proposed in [16] consists in performing the optimization on a subset of RF trees, after demonstrating that the consequent sub-optimality gap is usually limited and under control.

To obtain a scalable solution approach, differently from [16], we propose here to compute an approximate solution by considering directly the partitions X_{ji} associated with the best targets y_i, as described next.

III. Proposed Method

For a given random tree T_j, let us consider an unconstrained optimization program aimed at finding the set of input vectors that minimize its prediction:

$$X_j^* = \arg\min_{\mathbf{x}} T_j(\mathbf{x}). \tag{11}$$

Recalling that $p_2 = 1$ (one sample for each leaf), the solution is:

$$X_j^* = X_{ji} : (X_{ji}, y_i) \in \mathcal{P}_j \ \wedge \ y_i = \min_{(\mathbf{x}_\ell, y_\ell) \in \mathcal{D}_j} y_\ell \tag{12}$$

i.e. the feature set(s) associated to the best data point(s) in the bootstrapped dataset of $\mathcal{D}_j$.

When considering the whole RF, the situation becomes more complex, because the RF output associated with each set of the overall partition is generally different from the output value of any single data point, as described in Section II-A and visualized in Fig. 1. To solve problem (1), one would have to compute all intersections of all sets X_{ji}, $j = 1, 2 \ldots, N_T$, $i = 1, 2 \ldots, N_j$, compute the corresponding piecewise constant outputs, and take the best one, resulting again in exponential computational complexity. The approach we propose consists instead in focusing on a small number of promising data points $(\mathbf{x}_i, y_i)$, and intersecting the corresponding sets X_{ji} for all trees that contain such points in their leaves.

At first, the best set X_j^* for each T_j is identified as in (12). Let us denote with $\mathbf{x}_k^*$, $k = 1, \ldots, N^*$, the distinct data points contained in at least one set X_j^*, $j = 1 \ldots, N_T$, and y_i^* the corresponding output values. Clearly, it holds $N^* \leq N_T$, but we will show that in practice $N^* \ll N_T$ (see Theorem 3.1). Now, we can group the indices $j = 1, \ldots, N_T$ as follows

$$I_k = \{j : \mathbf{x}_k^* \in X_j^*\},\ k = 1, \ldots, N^* \tag{13}$$

Then, we compute the intersection of sets sharing the same best data point:

$$\hat{X}_k^* = \bigcap_{j \in I_k} X_j^* \tag{14}$$

By construction, $\hat{X}_k^* \neq \emptyset$, because all the feature sets associated with a target y_k^* contain at least the feature vector $\mathbf{x}_k^*$ (see Fig.1). To trade off optimality and computational complexity, one can consider the first $M \leq N^*$ sets $\hat{X}_k^*$, ordered by the goodness of the corresponding targets y_k^*.

Finally, the proposed approach computes candidate minimizers for the RF by taking the central points of sets $\hat{X}^*_k$, $k = 1, \dots, M$. This corresponds to considering the expected value assuming a uniform probability distribution of the target over each set. Let us denote with $\mathbf{x}^c_k$ such central points. The final approximate solution is found by computing $R(\mathbf{x}^c_k)$, $k = 1, \dots, M$ and taking the one that yields the minimum prediction:

$$\hat{\mathbf{x}}^* = \arg \min_{k=1,\dots,M} R(\mathbf{x}^c_k), \tag{15}$$

$$\hat{y}^* = \min_{k=1,\dots,M} R(\mathbf{x}^c_k) \tag{16}$$

The rationale of the proposed method derives from considering that the optimal solution of each random tree is provided by the optimal sample of its training set. As the RF just averages the prediction of all the trees, the optimal feature vector of the whole RF is likely to be close to the optimal samples appearing in the training sets of the trees, $\mathcal{D}_j$. The next result quantifies the number of trees that share the same best point, and provides a theoretical foundation to taking a rather small number M of candidates with respect to the total number of trees.

Theorem 3.1: Consider a finite set of distinct target values $\mathcal{Y} := \{y_i\}, i = 1, 2, ..., N, \quad y_i \in \mathbb{R}$ and other N_T finite sets $\mathcal{Y}_j := \{y_i\}$, $i \in \{1, 2, ..., N\}$, obtained by sampling element y_i from $\mathcal{Y}$ with equal-probability draws with-replacement (bootstrap sampling), so that $\mathcal{Y}_j$ can contain repeated samples (namely, index i can be repeated); let y_i be ordered, such that $y_1 < y_2, ..., < y_i <, ..., y_N$; finally, let $\mathcal{T}_k$, $k = 1, \dots, N^*$ be the collection of sets $\mathcal{Y}_j$ such that $\min_{y_i \in \mathcal{Y}_j} y_i = y_k$. Then, the expected cardinality of $\mathcal{T}_k$ is given by:

$$\mathbb{E}(|\mathcal{T}_k|) = \begin{cases} qp^{k-1}N_T & \text{for } k = 1, 2, ..., N^* - 1 \\ p^{k-1}N_T & \text{for } k = N^* \end{cases} \tag{17}$$

where $p := (1 - 1/N)^N$ and $q := 1 - p$.

Note that, for large N, we have:

$$p \simeq e^{-1} \simeq 0.37; \qquad q \simeq 0.63 \tag{18}$$

Therefore, as a consequence of Theorem 3.1, the best optimal sample, which coincides with the absolute minimum of the original dataset $\mathcal{D}$, appears on average in $\approx 63\%$ of the random trees; when this absolute minimum is not present, the "2nd best" point appears in $\approx 23\%$ of trees on average, then the "3rd best" for trees that don't have the first two in their training sets ($\approx 8\%$), and so on. One can see that, in expectation, the number M of data points that must be considered to deplete all the candidates offered by the various trees is a small fraction of the number of trees N_T.

Remark 1: It is also possible to associate the expected value of (17) with a confidence interval. To do that, we exploit the Central Limit Theorem. Defining for simplicity $\mu_i := \mathbb{E}(|\mathcal{T}_i|)$ and letting $\bar{\mu}_i$ be the corresponding sample average coming from a particular realization of bootstrapped sets $\mathcal{D}_j$, it yields:

$$\bar{\mu}_i \sim \mathcal{N}(\mu_i, \frac{\sigma_i}{\sqrt{N_T}}) \tag{19}$$

where σ_i is the variance of $\bar{\mu}_i$ distribution. By definition, the variance σ_i is given by:

$$\sigma_i = \mathbb{E}\left(\left(\mathbb{E}\left(|\mathcal{T}_i|\right) - |\mathcal{T}_i| \right)^2 \right) = \mu_i \left(\mu_i - 1\right)^2 \tag{20}$$

The overall approach is summarized in the pseudo-code of Algorithm 1. Tasks at lines 2, 3, 9 and 10 are the most time-consuming. However, their complexity is linear with the product $N \times N_T$, namely, linear with the overall number of nodes characterizing the RF.

Algorithm 1 Compute $\hat{y}^*$ and $\hat{\mathbf{x}}^*$

```
Require: $R := \{T_1, ..., T_j, ..., T_{N_T}\}$,
    $\mathcal{D}, \mathcal{D}_j \ \forall j = 1, 2, ..., N_T$
    $M \leftarrow$ User input   $(M \in \mathbb{N})$
Ensure: $\hat{y}^* = R(\hat{\mathbf{x}}^*) \approx \min_{\mathbf{x} \in \Omega} R(\mathbf{x})$
 1: for $j = 1$ to $N_T$ do
 2:     identify labeled partitions $\mathcal{P}_j := \{X_{ji}, y_i\}$
 3:     compute $X^*_j$ and $y^*_j$ by Eq. (12)
 4: end for
 5: $\mathcal{Y}^* \leftarrow$ all the $N^*$ distinct values of $y^*_j$
 6: Order $y^*_k \in \mathcal{Y}^*$ such that $y^*_k < y^*_{k+1}$
 7: $M \leftarrow \min(M, N^*)$
 8: for $k = 1$ to $M$ do
 9:     $I_k \leftarrow \{j : \mathbf{x}^*_k \in X^*_j\}$
10:     $\hat{X}^*_k \leftarrow \bigcap_{j \in I_k} X^*_j$
11:     $\mathbf{x}^c_k \leftarrow$ central point of $\hat{X}^*_k$
12:     $\hat{y}^*_k \leftarrow R(\mathbf{x}^c_k)$
13: end for
14: $k^* \leftarrow \arg\min_k \hat{y}^*_k$
15: $\hat{y}^* \leftarrow \hat{y}^*_{k^*}$
16: $\hat{\mathbf{x}}^* \leftarrow \mathbf{x}^c_{k^*}$
```

IV. Case study: Tripod function

The proposed approach is demonstrated by resorting to a Montecarlo analysis, which consists of minimizing a set of 100 RFs trained on a dataset generated by sampling a bivariate benchmark function and, then, comparing the approximation sub-optimality with the exact solution provided by fine a gridding. In fact, each RF will result in a different structure due to the inherent stochasticity of the generation algorithm (see Section II), thus enabling statistical evaluations. The advantage of focusing on a simple 2-dimensional feature space is two-fold: it allows the computation of the exact solution with MILP in a reasonable amount of time (for the same reason, we decided to limit the number of trees for each RF) and enables an expressive visualization of the objective function, highlighting its peculiar characteristics.

The selected benchmark, used to test evolutionary optimization algorithms, is the so-called Tripod function [18]. It is defined as $f : \mathbb{R}^2 \rightarrow \mathbb{R}^+$:

$$\begin{aligned} f(\mathbf{x}) = & \, p\left(x_2\right)\left(1 + p\left(x_1\right)\right) + \\ & \left| \left(x_1 + 50p\left(x_2\right)\left(1 - 2p\left(x_1\right)\right)\right) \right| + \\ & \left| \left(x_2 + 50\left(1 - 2p\left(x_2\right)\right)\right) \right| \end{aligned} \tag{21}$$

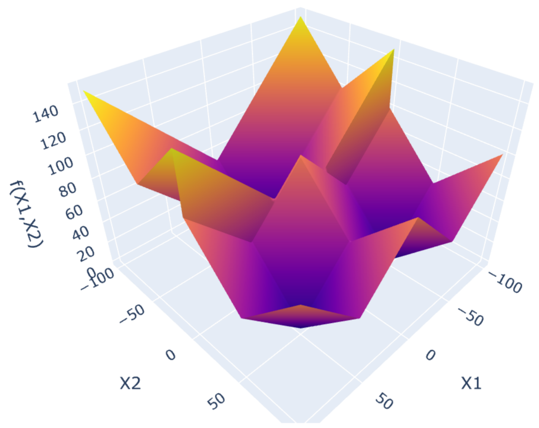

Fig. 2. Tripod function plot (browsable html: http://tinyurl.com/49zsw28a)

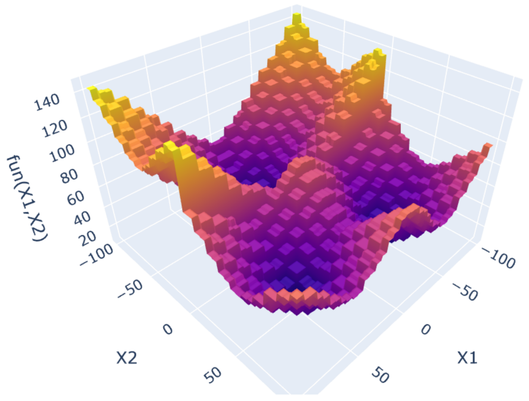

Fig. 3. Example of RF trained on samples of the Tripod function (browsable html: http://tinyurl.com/4w95ecy8)

with $\mathbf{x} := [x_1, x_2]$ and $p(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$.

It can be verified that $\min f = f(\mathbf{x}^*) = 0$, with $\mathbf{x}^* = [0, -50]$. The Tripod function exhibits some properties that make its minimization challenging: it is discontinuous, non-separable (a decomposition in the form $f(\mathbf{x}) = f_1(x_1) f_2(x_2)$ is not possible), non-scalable (it is defined only for $\mathbf{x} \in \mathbb{R}^2$), multimodal (with multiple local minima). A 3D plot of the function is provided in Fig. 2.

The samples' mesh, used to generate the dataset $\mathcal{D}$ for RFs training, is a regular grid of 20x20 points in $\Omega := [-100, 100] \times [-100, 100]$, extremes included. It can be observed that, due to the particular sampling, the minimum target sample in $\mathcal{D}$ is $\{\mathbf{x}^*_{\mathcal{D}}, y^*_{\mathcal{D}}\} := \{[-47.3, 47.3], 6.26\}$, which is different from the global minimum of the underlying Tripod function, although close to another local minimum.

Then, 100 RFs with 25 trees each ($N_T = 25$) have been trained on $\mathcal{D}$, according to the procedure outlined in Section II, with hyperparameters $p_1 = 2$ and $p_2 = 1$. Regarding p_1 choice, due to the limited number of features, we decided that all of them had to be ranked to search for the best splitting by Eq. (2). Fig. 3 illustrates one of the resulting RF. The piecewise nature of the RF can be appreciated: the output values are constant for feature vectors that belong to the same polytope in the overall partition.

An implementation of Algorithm 1 in Python 3.10.11 has been developed[1]. has been developed and applied to each RF in the Montecarlo set. The identification of the first $M = 4$ best targets in the bagged dataset $\mathcal{D}_j$ has been carried out for each RT of the RFs, together with the corresponding partition sets. Fig. 4 shows the result of best targets identification for RF #1.

The best candidate, i.e., the sample with the minimum target in $\mathcal{D}$, appears in 14 out of 25 training sets $\mathcal{D}_j$. It can be noted that the 2nd, 3rd and 4th best candidates appear with a frequency that is compatible with Theorem 3.1. In case of RF #1, 4 candidates are enough to deplete all the distinct minimum samples in the $\mathcal{D}_j$ sets, i.e. in this case

[1]https://github.com/MarLeonesio/RF_Optimization

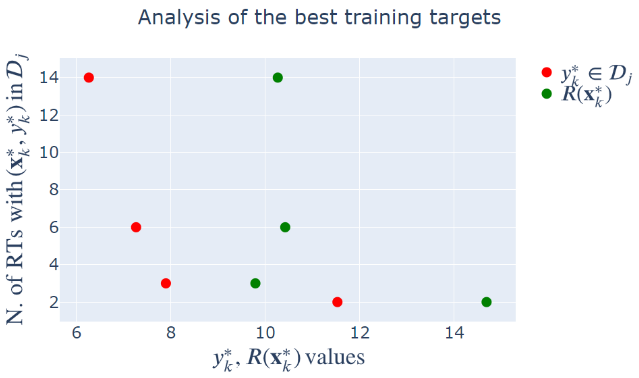

Fig. 4. Analysis of the best targets (frequency and values) for the RF #1.

$M = N^* = 4$ (see line 7 of Algorithm 1). Interestingly, the RF prediction for the input vector associated with the 1st best candidate is not the 1st best prediction, which is instead associated with the 3rd one. This is because the best targets have similar values, which are comparable to the variance introduced by the inherent stochasticity in RTs identification.

The exact absolute minimum of each RF, needed to evaluate the approximation error, has been computed by an equally spaced fine gridding, i.e., evaluating them over a 1000×1000 grid in Ω.

The overall Montecarlo performance analysis is illustrated in Fig. 5, which reports the empirical distribution of the relative and absolute error between the exact global minimum and the approximate solution for the 100 RFs. In 14 cases, the approximate approach yields the exact solution; the overall average relative error is about 5.5% (with a standard deviation of 4.4%), and the average absolute error is 0.47 (with a standard deviation of 0.35). In some cases, the deviations can be explained by the fact that the approximate solution refers to a different local minimum. This circumstance is highlighted in Fig. 6, which shows the exact and approximate minimizers, represented at their position in the input space by circles, whose radius is proportional to the corresponding frequency in the Montecarlo solutions set. When the Euclidean distance is small (lower than 20), the exact and approximate $\mathbf{x}^*$ refers to the same minimum

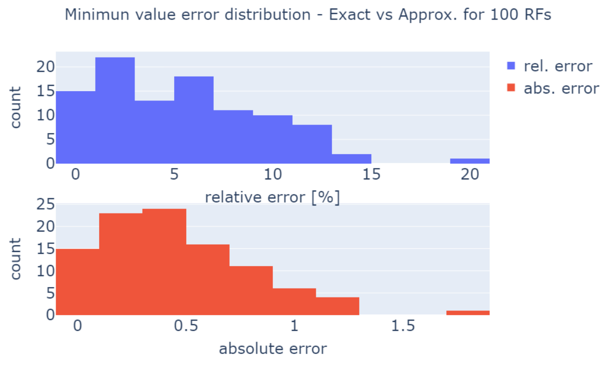

Fig. 5. Comparison between the exact and approximate function minimum for 100 instances of RF training: relative and absolute error

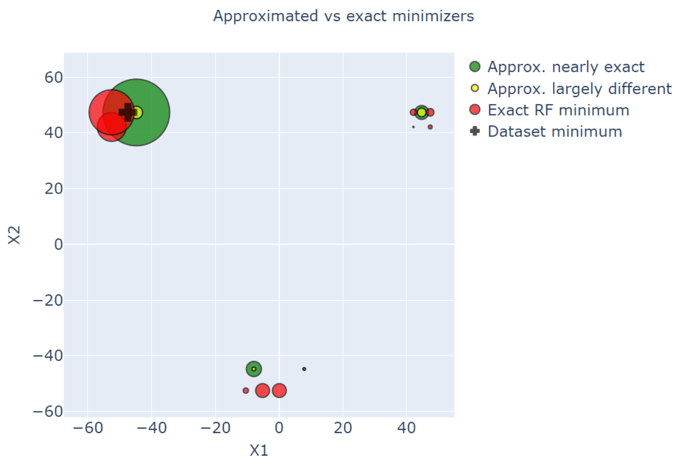

Fig. 6. Comparison between minimizers position in the input space. The circle radius is proportional to the frequency of that minimizer among the solutions of the 100 RFs optimization problems.

(approximation nearly exact, depicted in green); otherwise, they pertain to different minima: this happens in 22 cases out of 100 (approximation largely different, depicted in yellow). Even when the exact minimum output coincides with the approximate one, the corresponding arguments may differ slightly. This is because the best point returned by the gridding approach does not generally coincide with any of the central points $\mathbf{x}_k^c$. Finally, it can be noticed that most of the solutions are close to the best sample in dataset $\mathcal{D}$ (indicated by a black cross), both for exact and approximate cases. Sometimes, the RF seems to correct the sampling bias by placing its global minimum close to the global minimum of the underlying Tripod function (i.e., $[0, -50]$). Such a varied behavior, comprising the mismatch between exact and approximate solutions, can be ascribed to the trickiness of the Tripod function, which presents three local minima that are very close in the output space: $f([0, -50]) = 0$, $f([-50, 50]) = 1$ and $f([50, 50]) = 2$.

V. Discussion and Conlcusions

A novel approach has been proposed to perform the global minimization of an objective function represented by an RF. It has been shown that its efficiency outperforms other approaches providing an exact solution, in particular, those based on MILP formulation, which entails using auxiliary optimization variables and algorithms characterized by exponential complexity growth with the number of data. Conversely, the complexity of the proposed approximation is linear w.r.t. the number of data, while its sub-optimality (evaluated by a Monetcarlo approach on a hard-to-optimize problem) appears to be acceptable (5.5% of relative error and 0.47 of absolute error in a function with image set ranging from 0 to 100). It can be noticed that in approximately 20% of cases, the approximation fails in providing the exact solution: the mismatch is fundamentally due to the presence of other 2 local minima in the RF with very close function values.

Future developments will concern the extension of the approach to constrained problems. Further investigations will be aimed at predicting the approximation sub-optimality. Finally, a prediction interval approach will be investigated to allow the computation of a confidence interval around the approximate solution.

References

[1] Sabug, L., Ruiz, F., & Fagiano, L. (2021). SMGO: A set membership approach to data-driven global optimization. Automatica 133 109890.
[2] Mattera, G., Caggiano, A. & Nele, L. (2024). Reinforcement learning as data-driven optimization technique for GMAW process. Weld World 68, 805–817.
[3] Koziel, S., & Leifsson, L. (Eds.). (2013). Surrogate-Based Modeling and Optimization: Applications in Engineering. Springer.
[4] Alexander I. J. Forrester, András Sóbester, Andy J. Keane (2008). Engineering Design via Surrogate Modelling: A Practical Guide. John Wiley & Sons.
[5] Nestor V. Queipo, Raphael T. Haftka, Wei Shyy, Tushar Goel, Rajkumar Vaidyanathan, P. Kevin Tucker. (2005). Surrogate-based analysis and optimization. Progress in Aerospace Sciences 41(1) 1-28.
[6] Breiman, Leo. (2001). Random forests. Machine learning 45(1) 5-32.
[7] Gilles Louppe (2014). Understanding Random Forests - from theory to practice. PhD Dissertation, University of Liège - Faculty of Applied Sciences, arXiv:1407.7502v3.
[8] Lin, Y., and Jeon, Y. (2006). Random Forests and Adaptive Nearest Neighbors. J of the American Statistical Association 101 578–590.
[9] Biau, Gerard, Erwan Scornet. (2016). A random forest guided tour. Test 25(2) 197-227.
[10] Quinlan, J. Ross. (1986). Induction of decision trees. Machine learning 1(1) 81-106.
[11] Salzberg, Steven L. (1994). Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. Machine Learning 16(3) 235-240.
[12] Breiman, Leo. (1996). Bagging predictors. Machine learning 24(2) 123-140.
[13] Marco Locatelli, Fabio Schoen, (2021). (Global) Optimization: Historical notes and recent developments, EURO Journal on Computational Optimization 9 100012.
[14] Horst, R., & Pardalos, P. M. (Eds.). (1995). Handbook of Global Optimization. Springer.
[15] Floudas, C. A., & Pardalos, P. M. (1999). Introduction to Global Optimization. Springer.
[16] Biggs, M., Hariss, R., and Perakis, G. (2023). Constrained optimization of objective functions determined from random forests. Production and Operations Management 32(2) 397–415.
[17] Haozhe Zhang, Joshua Zimmerman, Dan Nettleton & Daniel J. Nordman. (2020). Random Forest Prediction Intervals. The American Statistician. 74(4) 392-406.
[18] Shahryar Rahnamayan, Hamid R. Tizhoosh, Magdy M.A. Salama. (2007). A novel population initialization method for accelerating evolutionary algorithms. Computers & Mathematics with Applications 53(10) 1605-1614.