Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

REGULAR TREE SEARCH FOR SIMULATION OPTIMIZATION

Du-Yi Wang[1,2], Guo Liang[2], Guangwu Liu[1], and Kun Zhang[2]

[1]Dept. of Decision Analytics and Operations, College of Business, City University of Hong Kong,
Kowloon, Hong Kong SAR, CHINA
[2]Institute of Statistics and Big Data, Renmin University of China, Beijing, CHINA

ABSTRACT

Tackling simulation optimization problems with non-convex objective functions remains a fundamental challenge in operations research. In this paper, we propose a class of random search algorithms, called Regular Tree Search, which integrates adaptive sampling with recursive partitioning of the search space. The algorithm concentrates simulations on increasingly promising regions by iteratively refining a tree structure. A tree search strategy guides sampling decisions, while partitioning is triggered when the number of samples in a leaf node exceeds a threshold that depends on its depth. Furthermore, a specific tree search strategy, Upper Confidence Bounds applied to Trees, is employed in the Regular Tree Search. We prove global convergence under sub-Gaussian noise, based on assumptions involving the optimality gap, without requiring continuity of the objective function. Numerical experiments confirm that the algorithm reliably identifies the global optimum and provides accurate estimates of its objective value.

1 INTRODUCTION

Simulation optimization, also known as optimization via simulation (OvS), refers to a class of optimization problems where the objective function values can only be evaluated through simulation, and the evaluations are subject to stochastic noise. Such problems have wide applications in various domains, including engineering design and hyperparameter tuning. For a comprehensive review of simulation optimization, see Amaran et al. (2016), Fan et al. (2024).

When the decision variables are discrete, the problem is called Discrete OvS (DOvS). Many algorithms for DOvS are based on random search strategies, such as the COMPASS algorithm (Hong and Nelson 2006). When the decision space is finite, the problem can be formulated as a multi-armed bandit problem, with well-established methods like Upper Confidence Bound (UCB) aiming to minimize regret (Auer et al. 2002). Alternatively, when the goal is to accurately estimate the optimal value rather than minimize regret, algorithms such as those proposed by Liu et al. (2019), Kun et al. (2024) are more appropriate. If the goal is to identify the best alternative (or arm), the problem is generally referred to as a Ranking and Selection problem. For comprehensive reviews of this setting, see Chen et al. (2015), Hong et al. (2021).

When the decision variables are continuous, the problem is known as Continuous OvS (COvS). A wide range of algorithms has been developed to address COvS problems. Among these are stochastic approximation algorithms (Kiefer and Wolfowitz 1952; Spall 1992; Wang et al. 2025a), response surface methodologies (Chang et al. 2013), model-based methods (Hu et al. 2007), and random search algorithms (Shi and Ólafsson 2000; Andradóttir and Prudius 2010; Kiatsupaibul et al. 2018; Zhang and Hu 2022; Wang et al. 2025). Different algorithms are suited to different COvS problem settings and offer distinct convergence guarantees. Notably, many algorithms assume global continuity of the objective function, an assumption that may fail in practice. In this paper, our focus is on random search algorithms, with the goal of designing methods that are gradient-free, globally convergent, and capable of handling discontinuous objective functions.

We consider a class of random search algorithms that partition the search space into disjoint regions to explicitly resolve the exploration-exploitation dilemma. This structural decomposition creates isolated promising areas, enabling strategic navigation of the fundamental tension between exploring uncertain regions and exploiting promising ones. The resulting partition structure enables discrete optimization algorithms to be seamlessly embedded within this framework, thereby transforming continuous optimization problems into the identification of optimal subspaces within discretized regions. Tree structures naturally embody this paradigm, as they recursively split the domain according to criteria that progressively refine promising regions. One of the most well-known decision tree algorithms is Classification And Regression Tree (CART), introduced by Breiman et al. (1984). At each node, CART selects axis-aligned split directions to optimize the Gini index for classification or mean squared error (MSE) for regression. Subsequent developments, including random forests (Breiman 2001) and various theoretical advancements (Scornet et al. 2015; Wager and Athey 2018; Mazumder and Wang 2023; Zhang et al. 2024), have improved predictive performance and provided convergence guarantees for these models. However, despite their contributions to predictive modeling, these advances often fail to address the unique challenges of simulation optimization, where the ultimate goal is to identify the global optimum.

A key distinction between prediction and simulation optimization lies in sample usage: prediction utilizes a complete sample set, whereas simulation optimization depends on samples generated iteratively. This fundamental distinction raises a critical question: which point should be simulated next? A central challenge in this setting is selecting the leaf node within a hierarchical tree structure, a key element in effective space partitioning. In addressing this challenge, Monte Carlo Tree Search (MCTS) provides a principled framework that adaptively balances exploration and exploitation. For instance, the Upper Confidence Bounds applied to Trees (UCT) algorithm proposed by Chang et al. (2005), Kocsis and Szepesvári (2006) exemplifies this approach. The practical success of UCT-based MCTS in complex combinatorial environments, as demonstrated by systems like AlphaGo (Silver et al. 2016), highlights the potential of tree-based strategies for tackling challenging optimization problems.

Specialized algorithms have been developed to adapt tree structures for optimization tasks. The optimistic optimization algorithm for noise-free settings proposed by Munos (2011) and its extensions to noisy settings Valko et al. (2013) rely on tree-based partitioning that employs random directional selection combined with uniform splitting. However, due to their reliance on uniform partitioning, these algorithms are not well-suited for adaptive splitting and tend to be less effective in isolating promising regions. In contrast, adaptive partitioning strategies have been introduced for noise-free settings. These include using linear regressors for space partitioning in neural architecture search (Wang et al. 2022), and distinguishing between high- and low-value regions via sample-driven splits (Wang et al. 2020). Despite promising experimental results, these methods lack theoretical guarantees.

The remainder of this paper is organized as follows. In Section 2, we formulate the simulation optimization problem and outline the key assumptions. Section 3 introduces the concept of the Regular Tree and details the criterion for its construction. In Section 4, we describe the Regular Tree Search and explain how the UCT algorithm is integrated into it. Section 5 presents the theoretical foundations that guarantee the global convergence of the algorithms under certain conditions. Finally, Section 6 presents numerical results that validate the proposed algorithm, and Section 7 concludes the paper.

2 PROBLEM FORMULATION

We consider the following simulation optimization problem:

$$\max_{\mathbf{x}\in\mathcal{X}\subseteq\mathbb{R}^d}\{\mu(\mathbf{x})=\mathbb{E}[Y|\mathbf{X}=\mathbf{x}]\}.$$

Here, $\mathbf{x}\in\mathbb{R}^d$ denotes the decision variable, $\mathcal{X}$ denotes the feasible domain, $\mu(\mathbf{x})$ denotes the unknown response surface of the simulation model, and Y represents the random output of a simulation model evaluated at $\mathbf{x}$. Since the conditional distribution of $Y|\mathbf{X}=\mathbf{x}$ is unknown and $\mu(\mathbf{x})$ has no closed form, we rely on running simulation experiments to generate independent samples of $Y|\mathbf{X}=\mathbf{x}$, denoted by $Y(\mathbf{x})$:

$$Y(\mathbf{x}) = \mu(\mathbf{x}) + \epsilon(\mathbf{x}),$$

where $\epsilon(\mathbf{x})$ is the zero mean simulation noise.

Our analysis focuses on the setting where the feasible domain $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$. While our theoretical framework can be adaptable to more general compact sets (as any compact set can be enclosed in a sufficiently large hyperrectangle), we specifically consider the canonical unit hypercube $\mathcal{X} = [0,1]^d$ for analytical tractability.

A fundamental challenge in simulation optimization arises from the black box of μ. Algorithms with minimal structural assumptions are highly desirable. This enhances the algorithm's applicability to a broad range of real-world problems where prior knowledge of the objective landscape is limited or unavailable. Many established algorithms, particularly those within the frameworks of Bayesian optimization and methods ensuring global convergence, rely on the assumption of global continuity. Examples include the approaches discussed in Frazier (2018) and Kiatsupaibul et al. (2018). Although this assumption facilitates theoretical analysis, it is restrictive and may be violated in practical simulation settings. For instance, functions with discontinuities might arise in specific system models. We consider the conditions focused on the optimal point instead of global continuity.

Throughout this paper, we make the following assumptions:

Assumption 1 The simulation noise $\epsilon(\mathbf{x})$ is sub-Gaussian with the variance proxy C_{sg}^2 for all $\mathbf{x} \in \mathcal{X}$, i.e., $\mathbb{E}\left[e^{\lambda\epsilon(\mathbf{x})}\right] \le e^{\lambda^2 C_{\text{sg}}^2/2}, \forall \lambda \in \mathbb{R}$.

Assumption 2 There exists a unique optimal point $\mathbf{x}^* \in \mathcal{X}$ of μ satisfying $\mu(\mathbf{x}^*) = \sup_{\mathbf{x}\in\mathcal{X}} \mu(\mathbf{x})$. The function μ exhibits local smoothness: there exists a constant $C > 0$ such that for any $\mathbf{x} \in \mathcal{X}$:

$$\mu(\mathbf{x}^*) - \mu(\mathbf{x}) \le C \cdot \|\mathbf{x} - \mathbf{x}^*\|.$$

where $\|\cdot\|$ denotes the Euclidean norm.

Assumption 3 For any $\epsilon > 0$, there exists $\delta(\epsilon) > 0$ such that for all $\mathbf{x} \in \mathcal{X}$ satisfying $\|\mathbf{x} - \mathbf{x}^*\| \ge \epsilon$,

$$\mu(\mathbf{x}) < \mu(\mathbf{x}^*) - \delta(\epsilon).$$

The sub-Gaussian noise assumption is a standard condition in the simulation optimization literature. Although the uniqueness condition in Assumption 2 could be relaxed to accommodate multiple optimal points, we retain it to highlight our algorithmic innovation.

It is important to note that continuity does not imply Assumption 2. For example, the one-dimensional function $\mu(\mathbf{x}) = 1 - \sqrt{\mathbf{x}}$ is continuous on $[0,1]$ but has an infinite derivative at its optimal point 0. Assumption 2 is designed to capture the quantitative relationship between the optimal point and other points, without imposing a global smoothness condition.

Assumption 3 is weaker than global continuity. To see this, suppose μ is continuous on $\mathcal{X}$ with unique optimal point $\mathbf{x}^*$. Then, for any $\epsilon > 0$, the set $K_\epsilon = \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x} - \mathbf{x}^*\| \ge \epsilon\}$ is compact by the boundedness and closeness. By the Extreme Value Theorem, the continuous function $\mu(\mathbf{x})$ attains its maximum on the compact set K_ϵ, denoted by $M_\epsilon = \max_{\mathbf{x}\in K_\epsilon} \mu(\mathbf{x})$. Since $\mathbf{x}^*$ is the unique global maximum of $\mu(\mathbf{x})$ and $\mathbf{x}^* \notin K_\epsilon$, it follows that $M_\epsilon < \mu(\mathbf{x}^*)$. Taking $\delta(\epsilon) = \frac{\mu(\mathbf{x}^*) - M_\epsilon}{2} > 0$, then for any $\mathbf{x} \in K_\epsilon$, we have:

$$\mu(\mathbf{x}) \le M_\epsilon = \mu(\mathbf{x}^*) - 2\delta(\epsilon) < \mu(\mathbf{x}^*) - \delta(\epsilon),$$

which directly verifies Assumption 3. Hence, Assumption 3 describes a local property of the optimum and is strictly weaker than global continuity. Assumption 2 captures how the function rises toward the optimum, while Assumption 3 ensures it declines beyond a neighborhood. Both express local characteristics of the optimum without relying on global smoothness.

3 REGULAR TREE

To address the theoretical limitations of the original CART algorithm, particularly the lack of convergence guarantees, Wager and Athey (2018) introduce a set of definitions that ensure favorable statistical properties. In this work, we adapt and extend these definitions into four definitions, honesty, random-split, α-spatial balance, and $f(c)$-sample balance, to align with the specific requirements of simulation optimization.

Definition 1 A tree is **honest** if, for each training sample i, it only uses the response Y_i to estimate the value of the leaf or to decide where to place the splits, but not both.

The concept of honesty ensures that the data used to grow the tree is entirely independent of the data used to estimate the leaf values. Although this separation may lose efficiency since it prevents using each observation for both purposes, it guarantees that the sample mean estimated in each leaf node is an unbiased estimator of the true expected value. In practice, we implement honesty by maintaining two disjoint datasets, $\mathbb{S}_{\mathcal{I}}$ and $\mathbb{S}_{\mathcal{J}}$. The dataset $\mathbb{S}_{\mathcal{I}}$ is used to estimate the values at the leaf nodes, while $\mathbb{S}_{\mathcal{J}}$ is used to grow the tree (i.e., to determine the split direction and split value). Each leaf node L corresponds to a hyperrectangle in the feasible domain. We define the random variable $Y(L)$ through the following two-step sampling procedure:

1. Sample $\mathbf{x}$ uniformly from L.
2. Generate Y according to the conditional distribution $\mathbb{P}_{Y|\mathbf{X}=\mathbf{x}}$.

Let $\bar{Y}_L = \frac{\sum_{i\in\mathcal{I}} \mathbb{1}\{\mathbf{x}_i\in L\}Y_i}{\sum_{i\in\mathcal{I}} \mathbb{1}\{\mathbf{x}_i\in L\}}$ denote the sample mean of responses in L computed from $\mathbb{S}_{\mathcal{I}}$. By construction, we have

$$\mathbb{E}[\bar{Y}_L] = \mathbb{E}[Y(L)].$$

Definition 2 A tree is **random-split** if, at each step of the tree-growing procedure, the probability that the next split occurs along the j-th feature is bounded below by κ/d for some $0 < \kappa \leq 1$, for all $j = 1, \dots, d$.

Definition 3 A tree grown by recursive partitioning is **α-spatially balanced** if, after each split, the Lebesgue measure of the splitting coordinate in each child node is at least an α fraction of that in the parent node.

Random split ensures that every feature has a non-negligible chance of being selected at every split. By imposing a uniform lower bound κ/d on the probability of selecting any given coordinate, we prevent the algorithm from systematically neglecting certain dimensions. This balance in feature selection is essential for deriving probabilistic bounds on how the tree explores the feasible domain and ensuring that all directions are sufficiently considered during the tree's growth. α-spatial balance further controls the geometry of each split. This condition prevents splits from creating one tiny region and one huge region, instead guaranteeing that every split reduces the size of each subregion at a predictable rate. Together, these two definitions ensure that the diameter of a leaf, defined as the Euclidean norm of its longest side, converges to 0 in probability as the leaf's depth increases. Intuitively, random-split spreads the splits across all dimensions, while α-spatial balance prevents any single split from producing excessively imbalanced partitions. Specifically, we have the following result:

Lemma 1 Let T be a random-split, α-spatially balanced tree, and let L be a leaf of T with depth c. Suppose that leaf $L = \bigotimes_{j=1}^{d} \left[r_j^-, r_j^+\right]$, where $0 \leq r_j^- < r_j^+ \leq 1$. Define the diameter of L as $\mathrm{diam}(L) = \sqrt{\sum_{j=1}^{d}(r_j^+ - r_j^-)^2}$. For any $\lambda \in (0,1)$, the following inequality holds:

$$\mathbb{P}\left[\mathrm{diam}(L) \geq \sqrt{d}(1-\alpha)^{(1-\lambda)\frac{\kappa}{d}c}\right] \leq d\exp\left\{-\frac{\lambda^2}{2}\frac{\kappa}{d}c\right\}.$$

Definition 4 A tree is **$f(c)$-sample balanced** if the number of $\mathcal{I}$-samples in the leaf node at depth c is bounded between $\beta f(c-1)$ and $f(c)-1$ for a positive function $f(c)$ and $0 < \beta < 1/2$.

The function $f(c)$ generalizes the concept of k-regularity from Wager and Walther (2015), where $f(c)$ is always a fixed constant. In contrast to the original CART setting, where the entire dataset is provided in advance, simulation optimization involves data generated adaptively as the algorithm progresses. Due to the presence of noise in the observations, accurately identifying the optimal point and estimating the optimal value requires that the number of samples in each leaf increases as the algorithm progresses. Consequently, in our framework, the number of samples in a leaf is linked to the depth of the leaf: deeper leaf nodes are required to contain more samples to ensure statistical reliability.

A tree that satisfies all four definitions is called a *Regular Tree*. To construct such a tree, we need to design a splitting criterion that explicitly enforces these four definitions. We propose a splitting criterion that guarantees compliance with all four definitions, as detailed in Algorithm 1.

Algorithm 1 Regular Tree splitting criterion.

```
Input: Datasets $\mathbb{S}_{\mathcal{I}}$ and $\mathbb{S}_{\mathcal{J}}$, spatially balanced parameter $\alpha \in (0, 0.5]$, random-split parameter $\kappa \in (0, 1]$,
    rectangle $L = \bigotimes_{j=1}^{d} \left[ r_j^-, r_j^+ \right]$ with depth $c$ where $0 \le r_j^- < r_j^+ \le 1$, and sample balanced parameter
    $f(c), \beta \in (0, 0.5)$ .
1: With probability $\kappa$:
2:     Randomly select a direction $j^* \in \{1, 2, \ldots, d\}$.
3:     Select the best split value $z^*$ by optimizing the MSE on $\mathbb{S}_{\mathcal{J}}$ along $j^*$-th direction:
           $$z^* = \arg\min_{z} \sum_{i \in \mathcal{J}} (Y_i - \bar{Y}_1)^2 \mathbb{1}\{\mathbf{x}_i \in L_1\} + \sum_{i \in \mathcal{J}} (Y_i - \bar{Y}_2)^2 \mathbb{1}\{\mathbf{x}_i \in L_2\},$$
       ensuring:
           $$(1-\alpha) r_{j^*}^- + \alpha r_{j^*}^+ \le z^* \le \alpha r_{j^*}^- + (1-\alpha) r_{j^*}^+, \quad (1)$$
       and
           $$\#\{i \in \mathcal{I} : \mathbf{x}_i \in L_l\} \ge \beta f(c), \quad l = 1, 2. \quad (2)$$
       Here, $L_1 = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{x}^{(j^*)} \le z\}$, $L_2 = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{x}^{(j^*)} > z\}$, $\bar{Y}_1, \bar{Y}_2$ are average responses of $\mathbb{S}_{\mathcal{J}}$
       within child nodes $L_1, L_2$.
4: Otherwise:
5:     Select the best split value $z^*$ and direction $j^*$ by optimizing the MSE:
           $$z^*, j^* = \arg\min_{z,j} \sum_{i \in \mathcal{J}} (Y_i - \bar{Y}_1)^2 \mathbb{1}\{\mathbf{x}_i \in L_1\} + \sum_{i \in \mathcal{J}} (Y_i - \bar{Y}_2)^2 \mathbb{1}\{\mathbf{x}_i \in L_2\},$$
       ensuring (1) and (2). Here, $L_1 = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{x}^{(j)} \le z\}$, $L_2 = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{x}^{(j)} > z\}$, $\bar{Y}_1, \bar{Y}_2$ are
       average responses of $\mathbb{S}_{\mathcal{J}}$ within child nodes $L_1, L_2$.
6: Return: Split direction $j^*$, split value $z^*$, leaf nodes $L_1, L_2$.
```

The setting of two disjoint datasets $\mathbb{S}_{\mathcal{I}}$ and $\mathbb{S}_{\mathcal{J}}$ satisfies Definition 1. Moreover, the probability κ for randomly selecting a direction satisfies Definition 2, while constraint (1) satisfies Definition 3 and constraint (2) satisfies Definition 4.

Using this criterion, we can grow a Regular Tree from the given data. The tree growing procedure is presented in Algorithm 2. This algorithm begins by dividing the dataset $\mathbb{S}$ into two disjoint datasets $\mathbb{S}_{\mathcal{I}}$ and $\mathbb{S}_{\mathcal{J}}$, and recursively applies Algorithm 1 to split the space until each node with depth c contains

$\beta f(c-1)$ to $f(c)-1$ samples from $\mathbb{S}_{\mathcal{I}}$. It can be shown that the tree grown by Algorithm 2 satisfies all four definitions.

Algorithm 2 Grow Regular Tree.

```
Input: Dataset $\mathbb{S} = (\mathbf{x}_i, Y_i)_{i=1}^n$, spatially balanced parameter $\alpha$, random-split parameter $\kappa$, and sample
    balanced parameter $f(c), \beta$.
 1: Divide $\mathbb{S}$ into $\mathbb{S}_{\mathcal{I}}$ and $\mathbb{S}_{\mathcal{J}}$ with $|\mathcal{I}| = \lfloor n/2 \rfloor$ and $|\mathcal{J}| = n - \lfloor n/2 \rfloor$.
 2: Repeat: each current node $L \subseteq [0,1]^d$ do
 3:     Use Algorithm1 with $\mathbb{S}_{\mathcal{I}}^L = \{(\mathbf{x}_i, Y_i) : i \in \mathcal{I}, \mathbf{x}_i \in L\}$, $\mathbb{S}_{\mathcal{J}}^L = \{(\mathbf{x}_i, Y_i) : i \in \mathcal{J}, \mathbf{x}_i \in L\}$ to find the
        split direction $j^*$, split value $z^*$. Divide $L$ into $L_1, L_2$.
 4: Until: Each node with depth $c$ contains $\beta f(c-1)$ to $f(c)-1$ samples in $\mathbb{S}_{\mathcal{I}}$.
 5: Return: A Regular Tree.
```

4 REGULAR TREE SEARCH

The Regular Tree Search algorithm involves two stages. In Stage 1, design points are sampled uniformly from the entire feature space $\mathcal{X}$, while in Stage 2, they are selected sequentially. The algorithm is outlined in Algorithm 3.

Algorithm 3 Regular Tree Search.

```
Input: Total sample budget $N$, spatially balanced parameter $\alpha$, random-split parameter $\kappa$, sample balanced
    parameter $f(c), \beta$, and stage 1 sample budget $N_0 < N$.
 1: Uniformly sample $N_0$ points from the space $\mathcal{X}$. Simulate at each point to get $\mathbb{S}$. Set $n = N_0$.
 2: Divide $\mathbb{S}$ into $\mathbb{S}_{\mathcal{I}}$ and $\mathbb{S}_{\mathcal{J}}$ with $|\mathcal{I}| = \lfloor n/2 \rfloor$ and $|\mathcal{J}| = n - \lfloor n/2 \rfloor$. Build up a Regular tree $T$ using
    Algorithm 2. Let $\mathcal{L}$ be the set of leaf nodes of $T$.
 3: While $n < N$ do
 4:     Select a leaf node $\hat{L} \in \mathcal{L}$ using a tree search strategy.
 5:     Uniformly samples a point $\mathbf{x}_{n+1}$ from the space of leaf node $\hat{L}$, simulate at point $\mathbf{x}_{n+1}$ to get $Y_{n+1}$.
 6:     $\mathbb{S}_{\mathcal{I}} = \mathbb{S}_{\mathcal{I}} \cup (\mathbf{x}_{n+1}, Y_{n+1})$.
 7:     If $\#\{i \in \mathcal{I}, \mathbf{x}_i \in \hat{L}\} = f(c)$, where $c$ is the depth of $\hat{L}$ then
 8:         Uniformly sample $\max\{f(c) - \#\{j \in \mathcal{J} : \mathbf{x}_j \in \hat{L}\}, 0\}$ points, simulate at each point once, set
            $n = n + \max\{f(c) - \#\{j \in \mathcal{J} : \mathbf{x}_j \in \hat{L}\}, 0\}$, and add these points to $\mathbb{S}_{\mathcal{J}}$.
 9:         Use Algorithm 1 with $\mathbb{S}_{\mathcal{I}}^{\hat{L}} = \{(\mathbf{x}_i, Y_i) : i \in \mathcal{I}, \mathbf{x}_i \in \hat{L}\}$, $\mathbb{S}_{\mathcal{J}}^{\hat{L}} = \{(\mathbf{x}_i, Y_i) : i \in \mathcal{J}, \mathbf{x}_i \in \hat{L}\}$ to find
            the split direction $j^*$ and split value $z^*$. Divide $\hat{L}$ into $L_1, L_2$.
10:         $\mathcal{L} = \mathcal{L} \cup L_1 \cup L_2 \backslash \hat{L}$.
11:     End if
12:     $n = n + 1$.
13: End while
14: Return: The leaf node $L^* \in \mathcal{L}$ with the largest sample mean
```

$$L^* = \arg\max_{L \in \mathcal{L}} \frac{\sum_{i \in \mathcal{I}} \mathbb{1}\{\mathbf{x}_i \in L\} Y_i}{\sum_{i \in \mathcal{I}} \mathbb{1}\{\mathbf{x}_i \in L\}}, \tag{3}$$

and its corresponding optimal value estimate

$$\bar{Y}_{L^*} = \frac{\sum_{i \in \mathcal{I}} \mathbb{1}\{\mathbf{x}_i \in L^*\} Y_i}{\sum_{i \in \mathcal{I}} \mathbb{1}\{\mathbf{x}_i \in L^*\}}. \tag{4}$$

In Stage 1, we allocate an initial budget of N_0 function evaluations to uniformly sample design points from the entire feasible domain $\mathcal{X}$. Each sampled point is simulated once, and the resulting dataset is split into two disjoint subsets, $\mathcal{S}_{\mathcal{I}}$ for estimating leaf values, and $\mathcal{S}_{\mathcal{J}}$ for determining the split direction and value. A Regular Tree can be grown via Algorithm 2. Stage 1 serves as a warm-up phase, designed to provide an initial exploration of the space. This phase is analogous to the warm-up procedure in Bayesian optimization, see, e.g., Frazier (2018). After this stage, we can construct an initial space partitioning of the feasible domain. Increasing the budget allocated to Stage 1 allows for a finer initial partitioning, providing a better global approximation. However, if the entire budget is exhausted during this phase, the algorithm degenerates into a regression algorithm, aiming to predict the function over the feasible domain rather than to identify optimal regions.

After Stage 1, the remaining sample budget is $N - N_0$. In each iteration, we select the next design point to be simulated using the tree search strategy. A tree search strategy is to select a leaf node $\hat{L} \in \mathcal{L}$ while balancing exploration and exploitation. Once a leaf node $\hat{L}$ is selected, a new point is uniformly sampled from its region and evaluated, and the resulting observation is added to $\mathbb{S}_{\mathcal{I}}$. If the number of $\mathcal{I}$-samples in $\hat{L}$ reaches the threshold $f(c)$, where c denotes the depth of the leaf $\hat{L}$, a refinement of the partition is triggered. We uniformly sample points in $\hat{L}$ and add to $\mathbb{S}_{\mathcal{J}}$ until the number of $\mathcal{J}$-samples in $\hat{L}$ reaches the threshold $f(c)$. Then we partition $\hat{L}$ into two new children, L_1 and L_2, using Algorithm 1. The set of leaf nodes $\mathcal{L}$ is then updated.

Once the budget is exhausted, the algorithm returns the leaf node $L^* \in \mathcal{L}$ with the highest sample mean of responses in $\mathbb{S}_{\mathcal{I}}$.

An effective tree search strategy for Stage 2 employs the UCT algorithm, which recursively selects child nodes from the root (i.e., the feasible domain $\mathcal{X}$) until reaching a leaf node. UCT operates on a selection criterion analogous to the UCB principle in multi-armed bandit problems, which is a highly efficient approach widely adopted in tree search algorithms. At a parent node $\hat{L}$ with children node L_1 and L_2, the UCB is computed for each child:

$$\text{ucb}_l = \bar{Y}_{L_l} + C_p \sqrt{\frac{2 \log n_p}{n_l}}, \quad l = 1, 2,$$

where $\bar{Y}_{L_l}$ is the sample mean of responses in L_l computed from $\mathbb{S}_{\mathcal{I}}$, n_l is the number of samples in L_l within $\mathbb{S}_{\mathcal{I}}$, and n_p is the number of samples in the parent node $\hat{L}$ within $\mathbb{S}_{\mathcal{I}}$. The constant C_p is a tunable hyperparameter that controls the trade-off between exploration and exploitation. Setting $C_p = 0$ results in a purely greedy search, which always selects the node with the largest sample mean. For the larger C_p, the algorithm reduces to a fully exploratory strategy that behaves like uniform sampling, eventually expanding the entire tree. At each step, the child node with the larger UCB value is selected as the new $\hat{L}$, and the process continues until a leaf node is reached.

5 CONVERGENCE ANALYSIS

In this section, we establish the convergence properties of Algorithm 3. Our objective is to show that as the simulation budget N increases, the algorithm identifies a region that concentrates around the true optimal point $\mathbf{x}^*$, and the corresponding estimated optimal value converges to the global maximum $\mu(\mathbf{x}^*)$. Specifically, we prove that:

$$\text{dist}(L_N^*, \mathbf{x}^*) \xrightarrow{P} 0,$$
$$\bar{Y}_{L_N^*} \xrightarrow{P} \mu(\mathbf{x}^*),$$

where L_N^* denotes the zone (leaf node) selected by Algorithm 3 with the largest sample mean when the sample budget is N, and $\text{dist}(L_N^*, \mathbf{x}^*) = \inf_{\mathbf{x} \in L_N^*} \|\mathbf{x} - \mathbf{x}^*\|$. For the complete proof, see Wang et al. (2025b).

We begin with Lemma 2, which establishes that for any region L, the random variable $Y(L)$ is sub-Gaussian. This property plays a crucial role in our analysis, as it enables the use of concentration inequalities to bound the deviation between the sample mean $\bar{Y}_L$ and its expected value $\mathbb{E}[Y(L)]$.

Lemma 2 Suppose that Assumptions 1 and 2 hold. For any hyperrectangle $L \subseteq \mathcal{X}$, $Y(L)$ is sub-Gaussian.

Theorem 1 states that the selected zone L_N^* asymptotically shrinks toward the true optimal point under some conditions.

Theorem 1 Suppose that Assumptions 1, 2, and 3 hold. Consider the tree search strategy that satisfies the following conditions:

- The depth of each leaf is bounded between $h^-(N)$ and $h^+(N)$.
- $\lim_{N\to\infty} h^-(N) = \infty$.
- $\lim_{N\to\infty} \frac{h^+(N)}{f(h^-(N))} = 0$.

Then, $\operatorname{dist}(L_N^*, \mathbf{x}^*)$ converges in probability to 0.

Theorem 1 contains three conditions:

1. During the algorithm, the depth of any leaf node lies between $h^-(N)$ and $h^+(N)$.
2. The tree search strategy is designed so that $h^-(N) \to \infty$ as the total budget $N \to \infty$, ensuring that the partition is refined ever deeper over time.
3. The ratio $\frac{h^+(N)}{f(h^-(N))} \to 0$ as $N \to \infty$ imposes a balance between the maximal tree depth $h^+(N)$ and the number of samples required for splitting. Intuitively, this condition prevents overfitting due to overly deep trees without sufficient data, thereby maintaining a balance between partition refinement and statistical reliability.

These three conditions imply that the sample balanced function $f(c)$, which determines the number of samples required to split a node at depth c, diverges as $c \to \infty$. This ensures deeper leaf nodes receive enough observations to yield statistically reliable estimates.

The exploration-exploitation trade-off in the UCT algorithm ensures that each node is selected infinitely many times, implying that $\lim_{N\to\infty} h^-(N) = \infty$. In practice, if $h^-(N)$ and $h^+(N)$ are of the same order, then setting $f(c)$ to be the order of $c \log c$ satisfies the conditions of Theorem 1. However, designing a tree search strategy that explicitly guarantees the desired behaviors of $h^-(N)$ and $h^+(N)$ remains an open problem and a promising direction for future research.

Building on Theorem 1, Theorem 2 establishes that the sample mean of the selected region, $\bar{Y}_{L_N^*}$, converges in probability to the true optimal value $\mu(\mathbf{x}^*)$.

Theorem 2 Under the conditions of Theorem 1, $\bar{Y}_{L_N^*} - \mu(\mathbf{x}^*)$ converges in probability to 0.

6 NUMERICAL EXPERIMENTS

In this section, we conduct numerical experiments to evaluate the performance of the Regular Tree Search algorithm with UCT. We compare our algorithm with three global convergence algorithms: the ASR algorithm from Andradóttir and Prudius (2010), and the IHR-SO and AP-SO algorithms from Kiatsupaibul et al. (2018).

Consider the Rastrigin function:

$$\mu(\mathbf{x}) = 10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)],$$

where $\mathbf{x} = (x_1, \ldots, x_d)$. The feasible domain is $\mathcal{X} = [-5, 5]^d$. The objective is to find the global minimum at $\mathbf{x}^* = (0, \ldots, 0)$ with $\mu(\mathbf{x}^*) = 0$. This function has multiple local minima. Let $\epsilon(\mathbf{x}) \sim N(0, 1)$.

Table 1: Statistics of the true function value of the optimal solution estimate of four algorithms.

					Quantile of value			
d	Algorithm	Mean	RMSE	Best	25%	50%	75%	Worst
2	Regular Tree Search	3.20	4.17	0.03	1.28	2.11	4.99	10.23
	ASR	5.23	6.16	0.31	3.02	4.78	6.73	20.50
	IHR-SO	36.54	36.74	22.61	34.24	37.18	39.69	43.80
	AP-SO	35.77	36.16	21.06	33.52	37.06	39.77	43.83
5	Regular Tree Search	9.16	10.08	1.82	6.10	8.77	12.22	20.15
	ASR	26.63	28.16	9.80	20.37	25.42	31.36	52.27
	IHR-SO	45.79	47.29	17.41	39.17	43.91	51.62	82.13
	AP-SO	59.87	60.92	19.98	52.52	60.30	67.56	87.37
10	Regular Tree Search	27.86	30.08	9.58	19.02	27.21	35.98	65.51
	ASR	76.68	79.04	39.78	62.34	73.60	88.46	134.56
	IHR-SO	92.28	93.72	43.84	81.50	91.57	103.33	138.44
	AP-SO	83.55	86.23	39.01	68.23	82.47	96.12	137.12

For illustration purposes, we consider a problem formulation that differs from our algorithm design by converting the original maximization problem into a minimization problem. This change is equivalent to solving the problem of maximizing the negative of the Rastrigin function.

The function is tested for different dimensions ($d = 2, 5, 10$) to evaluate the algorithm's performance. The total sample budget N is set to $500d$. For Regular Tree Search, the first stage sample budget is set to $N_0 = 150d$. Smaller value of parameters α, κ and β, combined with larger C_p promote exploration. We adopt the default settings: the spatially balanced parameter $\alpha = 0.1$, the random-split parameter $\kappa = 0.1$, the UCT parameter $C_p = 2$, splitting parameter $\beta = 1/3$. Theorem 2 suggests $f(c)$ on the order of $c \log c$, although theoretical guarantees for the order of $h^-(N)$ and $h^+(N)$ remain incomplete. To balance finite-sample performance against the risk of premature overpartitioning in early iterations, we define the default sample balanced parameter $f(c) = \max(c \log c, 15)$. The ASR, IHR-SO, and AP-SO algorithms use the same parameter settings as Andradóttir and Prudius (2010) and Kiatsupaibul et al. (2018).

Each experiment is repeated 100 times. For each algorithm, we record the *true function value of the optimal solution estimates* and the *optimal value estimates*. Since Regular Tree Search ultimately selects a region, as shown in (3), we use the midpoint of the selected region as the optimal solution estimate to facilitate comparison. The corresponding estimated optimal value is computed as in (4). The true function value at the estimated optimal solution and the estimated optimal value for ASR, IHR-SO, and AP-SO algorithms are computed in the same manner as described in Kiatsupaibul et al. (2018). We report summary statistics over 100 replications, including the mean, Root MSE (RMSE), the best value, 25th, 50th, and 75th percentiles, as well as the worst value. Table 1 presents the performance statistics of the true function value of the optimal solution estimates of four algorithms across different dimensions d, while Table 2 presents the corresponding performance statistics of the optimal value estimates.

For $d = 2$, Regular Tree Search outperforms the competing methods. The algorithm achieves a mean true function value of 3.20 with an RMSE of 4.17, and its best-case performance is exceptionally low at 0.03. The quantile values (1.28 at the 25th percentile, 2.11 at the 50th, and 4.99 at the 75th) exhibit a narrow spread, indicating that Regular Tree Search not only delivers high accuracy but also maintains robust and consistent performance. In comparison, ASR records a higher mean of 5.23 and RMSE of 6.16, while both IHR-SO and AP-SO yield substantially larger mean values (36.54 and 35.77). A similar trend occurs for optimal value estimates. Regular Tree Search attains a mean of 2.72 and RMSE of 3.79 with a best-case value of 0.02, confirming its superior accuracy and consistency in low-dimensional settings.

Table 2: Statistics of optimal value estimates of four algorithms.

					Quantile of value			
d	Algorithm	Mean	RMSE	Best	25%	50%	75%	Worst
2	Regular Tree Search	2.72	3.79	0.02	0.68	1.69	4.29	9.78
	ASR	5.23	6.17	0.34	4.01	4.85	6.74	20.64
	IHR-SO	19.70	19.71	17.89	19.16	19.76	20.25	21.56
	AP-SO	21.43	21.44	18.00	20.99	21.39	21.98	23.89
5	Regular Tree Search	8.97	9.91	1.29	6.07	8.55	11.96	20.22
	ASR	26.60	28.13	9.82	20.37	25.40	31.30	52.19
	IHR-SO	41.51	41.55	37.66	40.26	41.51	42.79	46.40
	AP-SO	48.88	48.92	45.17	47.09	48.94	50.38	55.39
10	Regular Tree Search	31.64	33.12	12.87	23.77	31.14	38.89	59.61
	ASR	76.62	78.98	39.57	62.37	73.72	88.32	134.47
	IHR-SO	84.83	85.02	73.59	80.94	84.19	88.57	101.80
	AP-SO	93.95	94.18	80.46	88.89	94.15	97.91	106.99

When the dimensionality increases to $d = 5$ and $d = 10$, Regular Tree Search remains competitive despite the challenges of higher dimensions. For $d = 5$, it has a true function value mean of 9.16 and an RMSE of 10.08, with quantiles expanding from 6.10 at the 25th to 12.22 at the 75th. At $d = 10$, it maintains a mean true function value of 27.86 and an RMSE of 30.08, with quantile values ranging from 19.02 (25th) to 35.98 (75th). In both cases, the competing algorithms experience sharper performance deterioration (with ASR, IHR-SO, and AP-SO recording mean values of 26.63, 45.79, 59.87 at $d = 5$ and 76.68, 92.28, 83.55 at $d = 10$, respectively), while Regular Tree Search continues to deliver lower error levels and more robust performance. Similar trends are confirmed by the optimal value estimates, where Regular Tree Search maintains a lower mean and error level relative to its counterparts.

When comparing the true function value of the optimal solution estimate with the optimal value estimates, a clear pattern emerges: both Regular Tree Search and SO algorithms exhibit noticeable gaps between their estimated and actual values, whereas ASR shows virtually no discrepancy. This difference arises because ASR performs multiple independent simulations at each design point and directly computes its sample mean, while Regular Tree Search, IHR-SO, and AP-SO each run only one simulation per point and then use different strategies to estimate the function value.

In particular, for $d = 2$ and $d = 5$, IHR-SO and AP-SO tend to underestimate the true function value. This is likely because these algorithms estimate values by averaging over a neighborhood, which can weaken the impact of the true optimum, especially if the function has sharp peaks. If the initial neighborhood size is too large, the estimate may become inaccurate.

Regular Tree Search enforces an honest sampling rule, ensuring that within-leaf sample means are unbiased estimators of the region's average. However, this does not guarantee an unbiased estimate at the center of the region. Consequently, in the $d = 2$ scenario, Regular Tree Search's estimated optimal value falls below the true optimal value for two reasons: first, we report the function value at the region's center rather than the region's actual mean; second, because Regular Tree Search always selects the leaf with the lowest estimated mean, this selection step introduces a negative selection bias in the reported optimal value estimate. As dimensionality increases to $d = 5$ and $d = 10$, the relationship between estimated and true values becomes less predictable. Sometimes, the estimates exceed the true values; sometimes, they fall short. This increased variability can be attributed to the coarser partitioning required in higher dimensions, which results in larger regions. When a region is large, the discrepancy between its center and its true mean is also large, leading to estimation errors.

7 CONCLUSION

This paper introduces Regular Tree Search, a class of random search algorithms for non-convex simulation optimization that integrates adaptive sampling with adaptive space partitioning. The algorithm initiates with a regular tree structure constructed from uniformly sampled points and iteratively selects leaf nodes by dynamically balancing exploration and exploitation through a tree search strategy. A new sample is drawn uniformly within the selected leaf, and when the number of samples in a node exceeds a threshold determined by its depth, an adaptive splitting criterion triggers further partitioning, progressively focusing the search on promising regions. The algorithm ultimately selects the best empirical solution from the leaves when the sampling budget is exhausted. To guide the search process, the framework incorporates the UCT algorithm. From a theoretical view, we establish global convergence under sub-Gaussian noise assumptions, relying on conditions concerning the optimality gap rather than global continuity. Numerical experiments validate the effectiveness of the algorithm, demonstrating its ability to accurately locate optimal solutions and estimate their values in complex, noisy optimization settings.

ACKNOWLEDGMENTS

The research of the second author is supported partially by the National Natural Science Foundation of China and the Research Grants Council of Hong Kong (RGC-HK), under the RGC-HK General Research Fund Projects 11508620 and 11508523 and Theme-based Research Project T32-615/24-R, and NSFC/RGC-HK Joint Research Scheme under project N_CityU 105/21. The research of the fourth author is supported by the National Natural Science Foundation of China (Grant 72471232).

REFERENCES

Amaran, S., N. V. Sahinidis, B. Sharda, and S. J. Bury. 2016, May. “Simulation Optimization: A Review of Algorithms and Applications”. *Annals of Operations Research* 240(1):351–380 https://doi.org/10.1007/s10479-015-2019-x.

Andradóttir, S., and A. A. Prudius. 2010, September. “Adaptive Random Search for Continuous Simulation Optimization”. *Naval Research Logistics (NRL)* 57(6):583–604 https://doi.org/10.1002/nav.20422.

Auer, P., N. Cesa-Bianchi, and P. Fischer. 2002, May. “Finite-Time Analysis of the Multiarmed Bandit Problem”. *Machine Learning* 47(2):235–256 https://doi.org/10.1023/A:1013689704352.

Breiman, L. 2001. “Random Forests”. *Machine Learning* 45(1):5–32 https://doi.org/10.1023/A:1010933404324.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification And Regression Trees*. 1 ed. New York: Chapman and Hall/CRC https://doi.org/10.1201/9781315139470.

Chang, H. S., M. C. Fu, J. Hu, and S. I. Marcus. 2005, February. “An Adaptive Sampling Algorithm for Solving Markov Decision Processes”. *Operations Research* 53(1):126–139 https://doi.org/10.1287/opre.1040.0145.

Chang, K.-H., L. J. Hong, and H. Wan. 2013, May. “Stochastic Trust-Region Response-Surface Method (STRONG)—A New Response-Surface Framework for Simulation Optimization”. *INFORMS Journal on Computing* 25(2):230–243 https://doi.org/10.1287/ijoc.1120.0498.

Chen, C.-H., S. E. Chick, L. H. Lee, and N. A. Pujowidianto. 2015. “Ranking and Selection: Efficient Simulation Budget Allocation”. In *Handbook of Simulation Optimization*, edited by M. C. Fu, 45–80. New York, NY: Springer New York https://doi.org/10.1007/978-1-4939-1384-8_3.

Fan, W., L. J. Hong, G. Jiang, and J. Luo. 2024. “Review of Large-Scale Simulation Optimization”. *arXiv preprint arXiv:2403.15669*.

Frazier, P. I. 2018. “A Tutorial on Bayesian Optimization”. *arXiv preprint arXiv:1807.02811*.

Hong, L. J., W. Fan, and J. Luo. 2021, September. “Review on Ranking and Selection: A New Perspective”. *Frontiers of Engineering Management* 8(3):321–343 https://doi.org/10.1007/s42524-021-0152-6.

Hong, L. J., and B. L. Nelson. 2006, February. “Discrete Optimization via Simulation Using COMPASS”. *Operations Research* 54(1):115–129 https://doi.org/10.1287/opre.1050.0237.

Hu, J., M. C. Fu, and S. I. Marcus. 2007, June. “A Model Reference Adaptive Search Method for Global Optimization”. *Operations Research* 55(3):549–568 https://doi.org/10.1287/opre.1060.0367.

Kiatsupaibul, S., R. L. Smith, and Z. B. Zabinsky. 2018, November. “Single Observation Adaptive Search for Continuous Simulation Optimization”. *Operations Research* 66(6):1713–1727 https://doi.org/10.1287/opre.2018.1759.

Kiefer, J., and J. Wolfowitz. 1952, September. “Stochastic Estimation of the Maximum of a Regression Function”. *The Annals of Mathematical Statistics* 23(3):462–466 https://doi.org/10.1214/aoms/1177729392.

Kocsis, L., and C. Szepesvári. 2006. "Bandit Based Monte-Carlo Planning". In *Machine Learning: ECML 2006*. September 18th-22th, Berlin, Heidelberg, 282–293.

Kun, Z., L. Guangwu, and S. Wen. 2024. "An Upper Confidence Bound Approach to Estimating the Maximum Mean". *arXiv preprint arXiv:2408.04179*.

Liu, G., W. Shi, and K. Zhang. 2019. "An Upper Confidence Bound Approach to Estimating Coherent Risk Measures". In *2019 Winter Simulation Conference (WSC)*, 914–925 https://doi.org/10.1109/WSC40007.2019.9004921.

Mazumder, R., and H. Wang. 2023. "On the Convergence of CART under Sufficient Impurity Decrease Condition". In *Advances in Neural Information Processing Systems*. December 10th-16th, New Orleans, LA, USA, 57754–57782.

Munos, R. 2011. "Optimistic Optimization of a Deterministic Function without the Knowledge of Its Smoothness". In *Advances in Neural Information Processing Systems*. December 12th-14th, Granada, Spain.

Scornet, E., G. Biau, and J.-P. Vert. 2015, August. "Consistency of Random Forests". *The Annals of Statistics* 43(4):1716–1741 https://doi.org/10.1214/15-AOS1321.

Shi, L., and S. Ólafsson. 2000, June. "Nested Partitions Method for Global Optimization". *Operations Research* 48(3):390–407 https://doi.org/10.1287/opre.48.3.390.12436.

Silver, D., A. Huang, C. J. Maddison, A. Guez, and L. Sifre. 2016, January. "Mastering the Game of Go with Deep Neural Networks and Tree Search". *Nature* 529(7587):484–489 https://doi.org/10.1038/nature16961.

Spall, J. 1992, March. "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation". *IEEE Transactions on Automatic Control* 37(3):332–341 https://doi.org/10.1109/9.119632.

Valko, M., A. Carpentier, and R. Munos. 2013, June. "Stochastic Simultaneous Optimistic Optimization". In *Proceedings of the 30th International Conference on Machine Learning*, Proceedings of Machine Learning Research. June 16th-21th, Atlanta, Georgia, USA, 19-27.

Wager, S., and S. Athey. 2018, July. "Estimation and Inference of Heterogeneous Treatment Effects Using Random Forests". *Journal of the American Statistical Association* 113(523):1228–1242 https://doi.org/10.1080/01621459.2017.1319839.

Wager, S., and G. Walther. 2015. "Adaptive Concentration of Regression Trees, with Application to Random Forests". *arXiv preprint arXiv:1503.06388*.

Wang, D.-Y., G. Liang, G. Liu, and K. Zhang. 2025a. "Derivative-Free Optimization via Finite Difference Approximation: An Experimental Study". *Asia-Pacific Journal of Operational Research*:2540005.

Wang, D.-Y., G. Liang, G. Liu, and K. Zhang. 2025b. "Regular Tree Search for Simulation Optimization". *arXiv preprint arXiv:2506.17696*.

Wang, L., R. Fonseca, and Y. Tian. 2020. "Learning Search Space Partition for Black-Box Optimization Using Monte Carlo Tree Search". In *Proceedings of the 34th International Conference on Neural Information Processing Systems*. December 6th-12th, Vancouver, Canada.

Wang, L., S. Xie, T. Li, R. Fonseca, and Y. Tian. 2022. "Sample-Efficient Neural Architecture Search by Learning Actions for Monte Carlo Tree Search". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44(9):5503–5515 https://doi.org/10.1109/TPAMI.2021.3071343.

Wang, X., L. J. Hong, Z. Jiang, and H. Shen. 2025, January. "Gaussian Process-Based Random Search for Continuous Optimization via Simulation". *Operations Research* 73(1):385–407 https://doi.org/10.1287/opre.2021.0303.

Zhang, Q., and J. Hu. 2022, November. "Actor-Critic–Like Stochastic Adaptive Search for Continuous Simulation Optimization". *Operations Research* 70(6):3519–3537 https://doi.org/10.1287/opre.2021.2214.

Zhang, Y., W. Ji, and J. Bradic. 2024. "Adaptive Split Balancing for Optimal Random Forest". *arXiv preprint arXiv:2402.11228*.

AUTHOR BIOGRAPHIES

DU-YI WANG is a Ph.D. student in the program jointly offered by Renmin University of China and City University of Hong Kong. His research interests include simulation optimization and machine learning. His email address is tylzml@ruc.edu.cn.

GUO LIANG is a Ph.D. Candidate in the Institute of Statistics and Big Data at Renmin University of China. His research interests include stochastic simulation, financial engineering, and risk management. His email address is liangguo000221@ruc.edu.cn.

GUANGWU LIU is a Professor in the Department of Decision Analytics and Operations, College of Business at City University of Hong Kong. His research interests include stochastic simulation, business analytics, financial engineering, and risk management. He serves as an Associate Editor of the Asia-Pacific Journal of Operational Research, and Naval Research Logistic. His e-mail address is msgw.liu@cityu.edu.hk and his website is https://www.cb.cityu.edu.hk/staff/guanliu.

KUN ZHANG is an Assistant Professor in the Institute of Statistics and Big Data at Renmin University of China. He holds a PhD in management science from City University of Hong Kong. His research interests include simulation optimization, machine learning, business analytics, financial engineering, and risk management. His email address is kunzhang@ruc.edu.cn.