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

PARTITIONING AND GAUSSIAN PROCESSES FOR ACCELERATING SAMPLING IN MONTE CARLO TREE SEARCH FOR CONTINUOUS DECISIONS

Menghan Liu
Giulia Pedrielli

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

Yumeng Cao

Arizona State University
School of Mathematical
and Statistical Sciences
900 S Palm Walk
Tempe, 85281, USA

2021 Winter Simulation Conference (WSC) | 978-1-6654-3311-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/WSC52266.2021.9715405

ABSTRACT

We propose Part-MCTS for sampling continuous decisions at each stage of a Monte Carlo Tree Search algorithm. At each MCTS stage, Part-MCTS sequentially partitions the decision space and keeps a collection of Gaussian processes to describe the landscape of the objective function. A classification criteria based on the estimation of the minimum allows us to focus the attention on regions with better predicted behavior, reducing the evaluation effort elsewhere. Within each subregion, we can use any sampling distribution, and we propose to sample using Bayesian optimization. We compare our approach to KR-UCT (Yee et al. 2016) as state of the art competitor. Part-MCTS achieves better accuracy over a set of nonlinear test functions, and it has the ability to identify multiple promising solutions in a single run. This can be important when multiple solutions from a stage can be preserved and expanded at subsequent stages.

1 INTRODUCTION AND MOTIVATION

Monte Carlo Tree Search (MCTS) is a general method for sequential decision making and it has been used as the basis to solve a plethora of large scale data driven problems with applications spanning across the fields of robotics and control, biology, gaming and many others (Billings et al. 2019; Fu 2018; Li et al. 2019). The basic idea of MCTS is quite intuitive, to sequentially approximate the solution of a, possibly, non-linear non-convex large scale optimization problem by decomposing the original problem into a stage-based formulation. As a result of the decomposition, at each stage, MCTS samples in the space of stage-decisions and approximately (and efficiently) evaluates the decision. Hence, the search progresses increasingly fixing the value of stage-decisions. This is accomplished through four main steps whose implementation gives rise to a large variety of approaches. These include: (i) *Selection*: chooses a stage-decision, several sampling rules can be adopted; (ii) *Expansion*: equivalent to the concept of transfer function in control, it is responsible to generate the children from a selected node; (iii) *Simulation*: responsible to complete the stage solution and associate an approximate reward to the selected node; (iv) *Backpropagation*: mechanism used to update the reward associated to the selected node. MCTS emerged as a general purpose, easy to implement, methodology to approximately solve rather difficult non linear non convex large scale optimization problems. The generality of the scheme makes it applicable to nearly any optimization problem, but it also represents one of the sources of inefficiency of the algorithm. While for discrete problems MCTS largely relies on the Upper Confidence Bound results to perform sampling, such an approach becomes impractical when (i) even if discrete, the number of decisions at

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

each stage is extremely large; (ii) the decisions at a specific stage are continuous. The two challenges are strongly related and can be tackled by similar techniques so long as the discrete decision space has some ordering. Since we propose to use a partitioning-based approach, another challenge that we directly face is to choose the number of evaluations to allocate to each subregion at each step of the search within a single MCTS stage-iteration. We use Optimal Computing Budget Allocation (OCBA (Chen et al. 2000)) to do so within this new context.

1.1 Background

We researched within the Monte Carlo Tree Search and the Information Theory/Statistical learning literature. We focus on key contributions on, provably, efficient methods to sample large dimensional, continuous spaces. We highlight that, while general techniques such as Bayesian optimization can be applied to continuous problems, the scope of this paper is on decomposed stage-decisions. In fact, an application of this setup is optimization over highly heterogeneous rewards and large scale problems, both settings where traditional Bayesian optimization implementations exhibit poor performance.

Partitioning to model heterogeneous responses Within the statistical learning community, so-called *treed Gaussian processes* have been proposed with main applications in learning from non-stationary and large data sets. This literature does not address optimization, but it is relevant in that it addresses non-stationarity of the response and the noise. In Chipman et al. (2002) a binary tree is used to learn partitions based on Bayesian regression. The difficulty to scale the approach to high dimensional inputs/large data sets led to the computational work in Denison et al. (2002). One drawback of these approaches was identified in the irregularity of the variance associated to the different subregions. In particular, it was observed that some subregions tended to exhibit variance orders of magnitude larger. In Kim et al. (2005), this problem is alleviated by using stationary processes within the several subregions (thus leading to a larger number of subregions, but better control over the variance profile). Differently, Gramacy and Lee (2005) deals directly with the problem of non-stationarity of the response and of the variance (heteroscedasticity) proposing a new form of Gaussian process, the Bayesian Treed Gaussian Process model. The approach combines stationary Gaussian processes and partitioning, resulting in treed Gaussian processes, and it implements a tractable non-stationary model for non-parametric regression. Along a similar line, Liang and Lee (2018) uses a binary tree to iteratively learn different models and, while the method has good fitting results, it also shows good computational performance for large data sets.

Efficient Sampling in continuous action spaces In several problems, continuous decision variables naturally arise. As a result, some research has been developed to extend MCTS to continuous spaces. Methods have been proposed for discretizing or cutting the continuous space based on domain knowledge (Smith 2007; Archibald et al. 2009; Yamamoto et al. 2015). An alternative to this approach is to directly modify the score functions used to approximate the reward and to sample nodes in a continuous space. The most common method in this family is MCTS-UCT (Upper Confidence Bounds Applied to Trees) (Kocsis and Szepesvári 2006) that applies the bandit algorithm UCB1 for guiding selective sampling of actions in rollout-based planning. Yee et al. (2016) proposes a Kernel based UCT algorithm, which makes use of execution uncertainty, using kernel regression to generalize the action value estimates over the entire parameter space, with the execution uncertainty model as its generalization kernel. Hierarchical Optimistic Optimization applied to Trees (HOOT) addresses planning in continuous action Markov Decision Processes and adaptively partitions the action space (Mansley et al. 2011). Kim et al. (2020) proposes the algorithm VOOT for MCTS, which is based on a novel black-box function-optimization algorithm (VOO) using Voronoi partitioning to efficiently sample actions. There are also hierarchical partitioning methods that can be used to achieve similar goals (Kawaguchi et al. 2015; Wang et al. 2014). In Wang et al. (2019), the latent action Monte Carlo Tree Search (LA-MCTS) sequentially focuses the evaluation budget by iteratively splitting the input space to identify the most promising region. Differently from our case, LA-MCTS uses non-linear boundaries for branching regions, and, like us, learns a local model to select candidates. Another

relevant contribution, AlphaX, presented in (Wang et al. 2019), explores the search using distributed Monte Carlo Tree Search (MCTS) coupled with a Meta-Deep Neural Network (DNN) model that guides the search and branching decision, focusing on the sequential selection of the most promising region.

Budget allocation represents an important aspect for MCTS in continuous spaces since exhaustive exploration of stage-decisions cannot be performed. In fact, most of the approaches in the literature assume that the simulation budget at the different stages and locations (decisions) is given as input. Similar to Li et al. (2019), but in a continuous decision space context, we use Optimal Computing Budget Allocation (OCBA) to allocate evaluation effort to candidate subregions interpreting the problem as the one of ordinal optimization (Chen et al. 2000). OCBA distributes the total evaluation budget to candidate designs sequentially according to the mean and variance of simulation results of the evaluated samples. OCBA is a highly efficient way to identify promising designs and allocate resources to them, and has been extended to a wide variety of problems and application contexts (Chen et al. 2008; Zhang et al. 2015; Xiao and Gao 2018; Xiao et al. 2020; Xiao et al. 2020; Zhang et al. 2016; Xiao et al. 2019).

1.2 Contributions

In this paper, we propose a partitioning based, surrogate model driven algorithm, Part, to improve sampling for MCTS for optimization problems with continuous decision variables. The result is Part-MCTS that efficiently samples across continuous decisions spaces at each stage of a Monte Carlo Tree Search algorithm. At each MCTS iteration, Part-MCTS sequentially partitions the decision space and keeps a collection of Gaussian processes to describe the landscape of the objective function, where the number of surrogates equals the number of active subregions. Within the literature in partitioning and MCTS, Part-MCTS contribution is threefold: (i) instead of partitioning the space and using a unique surrogate model to take sampling decisions, Part uses a collection of Gaussian processes. This results in improved flexibility in terms of correlation functions; (ii) different from implicit partitioning schemes, Part uses orthogonal partitions but avoids the explosion of hyperboxes by stopping branching when a region is deemed not interesting; (iii) Part tackles the problem of how to assign simulation budgets to each subregion in the form of an optimal computing budget allocation problem.

2 Methodology

Under the Part sampling perspective, at the τ^{th} MCTS stage, a black-box optimization needs to be performed over the deterministic function $J(x)$. Namely, we want to solve $\min_{\mathbf{x}_\tau} J\left(\mathbf{x}^*_{[1:\tau-1]}, \mathbf{x}_\tau\right)$, where $J\left(\mathbf{x}^*_{[1:\tau-1]}, \mathbf{x}_\tau\right) = \min_{\mathbf{x}_{[\tau+1:D]}} f(\mathbf{x}_1, \ldots, \mathbf{x}_{\tau-1}, \mathbf{x}_\tau, \ldots, \mathbf{x}_D)$ where $\mathbf{x}^*_{[1:\tau-1]}$ is the best partial solution so far from iteration 1 to iteration τ (Zabinsky et al. 2019; Bertsekas et al. 2000). This function, not available in closed form, can be approximated using simulation (rollout), which is not the focus here (the interested reader can refer to (Fu 2018; Li et al. 2019; Bertsekas 2020)). It is important to highlight that the stage-decision, $\mathbf{x}_\tau$, can be of any dimension based on the decomposition dictated by the modeler/problem.

In the context of this paper, we are interested in the problem of finding a good solution to sample at each MCTS stage and we view this as the problem of minimizing a possibly non-linear non-convex objective function $f : \mathbf{x} \in X \subset \mathbf{R}^{d(\tau)} \rightarrow f(\mathbf{x}) \in \mathbf{R}$ over a feasible space $\mathbb{X}$, which we will assume to be compact. To find such solution, MCTS will allow us a total of T evaluations for each decision stage. Importantly, the decision space $\mathbb{X}$ is continuous, knowingly representing a computational challenge for MCTS-like approaches (section 1.1). Our goal is to find a sequence of decisions that minimize the objective function $J(\mathbf{x})$. The algorithm we propose is an adaptive partitioning and sampling approach that provides: (i) a surrogate of the unknown response in each subregion of the partition iteratively formed; (ii) a mechanism to estimate the optimistic (fortified) minimum of the function achieved in each subregion (Bertsekas et al. 2000).

Figure 1 gives an overview of the proposed approach, which we refer to as Part-MCTS (Partitioning based Monte Carlo Tree Search). The algorithm sequentially and adaptively partitions the space (in this work, dimension-aligned binary partitions are used). In this paper, we use the words "cut" and "branch" interchangeably and refer to a subregion as the result of the branching. Inputs are then evaluated and a surrogate for the objective function is estimated for each subregion. In particular, the surrogates we use are Gaussian processes (Santner et al. 2013; Mathesen et al. 2021; Pedrielli et al. 2020), which allows us to define a variety of sampling distributions ($T(\mathbf{x}|\cdot)$ in Figure 1). In the following, we outline, as in the figure, the Part-MCTS procedure at the τ^{th} stage of the MCTS search.

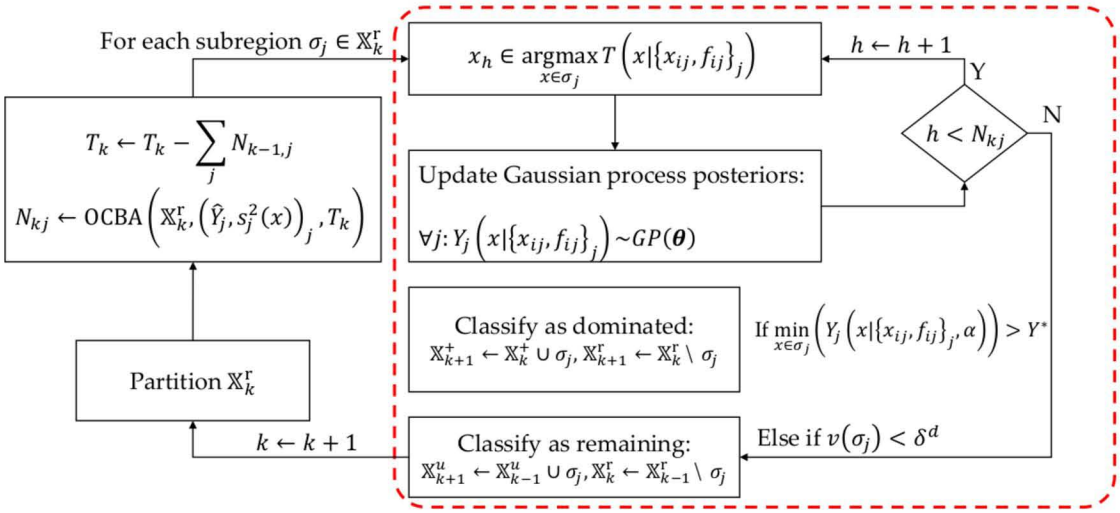


Figure 1: Sampling scheme implemented by the use of Part-MCTS.

At the k^{th} iteration, given a budget $\Delta < T$ of observations, a number N_{kj} of points satisfying $\sum_j N_{kj} \leqslant \Delta$ is sampled in each sub-region σ_j, and the corresponding surrogate model is updated. Specifically, N_{kj} is determined by an optimal budget allocation rule, while Δ is a constant (generalizations have been presented in the context of simulation optimization (Pedrielli et al. 2020)). Then, we decide whether to stop branching a region based on the fact that the posterior α-quantile of the minimum predicted objective value is above the current best estimated function value, which deems an input to be non interesting (dominated). It is important to highlight the complexity associated with the estimation of the minimum α-quantile associated to the Gaussian process $\min_{\mathbf{x}\in\sigma_j}\left[\hat{Y}_j(\mathbf{x}) - Z_{1-\alpha/2}\sqrt{s^2(\mathbf{x})}\right]$. In this work, we use a Monte-Carlo estimate for these quantities. When no longer branched, a region enters the, potentially disconnected, set $\mathbb{X}^+$. In the case the uncertainty associated to the model(s) is large (which is typically the case at the first iterations of the algorithm) no sub-region will be dominated, and all subregions will be branched. If a region reaches the minimum volume $v(\sigma) = \delta^d$ we cannot branch the subregion any more. The algorithm continues until: (i) the maximum number of evaluations is exhausted; (ii) all the subregions have been classified.

The remainder of the section is structured as follows: section 2.1 presents the basic definitions for Gaussian processes, which we use to produce predictions of the cost/reward associated to the problem. Section 2.2, introduces the Part-MCTS scheme to iteratively branch, sample, update subregion models, and decide whether to classify each of the subregions.

2.1 Gaussian process modeling

A Gaussian process (GP) is a statistical learning model used to build predictions for non-linear, possibly non-convex smooth functions (note that other surrogate models may be used in place of the Gaussian process, with minimal impact on the overall algorithm). The basic idea is to interpret the true, unknown function

$y(\mathbf{x})$ as a realization from a stochastic process, the Gaussian process. If we can measure the function without noise, then the Gaussian process will interpolate the true function at the evaluated points, while, conditional on the sampled locations $\mathbf{x}_1,\ldots,\mathbf{x}_n$, a Gaussian process produces the conditional density $P(Y(\mathbf{x}_0)|\mathbf{x})$. In particular, $Y(\mathbf{x}) = \mu + Z(\mathbf{x})$, where μ is the, constant, process mean, and $Z(\mathbf{x}) \sim GP(0,\tau^2 R)$, with τ^2 being the constant process variance and R the correlation matrix. Under the Gaussian correlation assumption, $R_{ij} = \prod_{l=1}^{d} \exp\left(-\theta_l \left(x_{il} - x_{jl}\right)^2\right)$, for $i,j,= 1,\ldots,n$. The d-dimensional vector of hyperparameters θ controls the smoothing intensity of the predictor in the different dimensions. The parameters μ and τ^2 are estimated through maximum likelihood (Santner et al. 2013): $\widehat{\mu} = \frac{1_n^T \mathrm{R}^{-1} f(X_n)}{1_n^T \mathrm{R}^{-1} 1_n}$, $\widehat{\tau}^2 = \frac{(f(\mathbf{X}_n) - 1_n\widehat{\mu}_g)^T \mathrm{R}^{-1}((f(\mathbf{X}_n) - 1_n\widehat{\mu}_g)}{n}$. The best linear unbiased predictor form is (Santner et al. 2013):

$$\widehat{f}(\mathbf{x}) = \widehat{\mu} + \mathbf{r}^T \mathrm{R}^{-1}(f(\mathbf{X}_n) - \mathbf{1}_n\widehat{\mu}) \tag{1}$$

where $\mathbf{X}_n$ is a set of n sampled locations, and $f(\mathbf{X}_n)$ is the n-dimensional vector having as elements the function value at the sampled locations. The model variance associated to the predictor is:

$$s^2(\mathbf{x}) = \tau^2 \left(1 - \mathbf{r}^T \mathrm{R}^{-1}\mathbf{r} + \frac{\left((1 - \mathbf{1}_n^T \mathrm{R}^{-1}\mathbf{r}\right)^2}{\mathbf{1}_n^T \mathrm{R}^{-1}\mathbf{1}_n}\right) \tag{2}$$

where $\mathbf{r}$ is the n-dimensional vector having as elements the Gaussian correlation between location $\mathbf{x} \in \mathbb{X}$ and the n elements of $\mathbf{X}_n$, i.e., $\mathbf{r}_i(\mathbf{x}) = \prod_{l=1}^{d} \exp\left(-\theta_l (x_l - x_{il})^2\right), i = 1,\ldots,n$.

In our application, we use the model in (1) as a surrogate for the unknown stage-cost function. In particular, given a training set of input and associated cost value $\{\mathbf{x}_i, y_i\}_{i=1}^n$, we will predict the cost $\widehat{Y}(\mathbf{x}_{n+1})$ at a new unsampled location $\mathbf{x}_{n+1}$.

2.2 Adaptive Branching and Classification

In this section, the model-based sequential adaptive partitioning is presented. Part-MCTS starts considering the entire input $\mathbb{X}$ and keeps branching until a stopping condition is met: (i) the evaluation budget dedicated to the current MCTS iteration is exhausted (i.e., the maximum number of function evaluations has been performed); (ii) all the non-classified subregions have the a length δ along all the dimensions (i.e., the subregion is unbranchable); (iii) all the subregions have been classified.

Main Notation At each step k of the τ^{th} iteration of Part-MCTS has generated three sets of subregions:

- Dominated set: $\mathbb{X}_k^+$, formed by new W_k^d subregions, represents the union of the subregions classified as dominated (with a level of significance α);
- Remaining set: $\mathbb{X}_k^r$, formed by new W_k^r subregions, represents the union of the subregions that are non classified and have a volume larger than δ^d;
- Unclassified region: $\mathbb{X}_k^u$, formed by new W_k^u subregions, represents the union of the subregions that cannot be classified and have all dimensions at a length of δ forming an hypercube of volume δ^d.

We refer to σ_j as the individual subregion resulting from branching at iteration k, given the previous definitions, at the k^{th} iteration, there will be a number $W = \sum_{i=1}^{k}\left(W_i^d + W_i^r + W_i^u\right)$ of new subregions.

Note that the iteration index k refers to the iterations at a single stage of the MCTS algorithm (see Algorithm 4). Each iteration k has an allocated budget Δ to be shared among all subregions that are active, i.e., $\sigma_j \in \mathbb{X}_k^r$. Looking at the single iteration, we will use t as the index for the sampling within such single iteration at a specific stage (see Algorithm 2). In the attempt to simplify the exposition, the budget allocation algorithm (Algorithm 1) cycles through regions and does not require the iteration index k since the procedure will be called at each of such iterations. The reader will notice that such index is in fact removed from the execution of the algorithms 1, 2, and 3.

Establishing the simulation budget in each subregion At each iteration k, of the τ^{th} stage, Part-MCTS algorithm can use a number of Δ evaluations across all subregions. N^0 samples are required to initiate the procedure in each subregion. Given that n_j observations have been sampled so far in each subregion σ_j, we need to subtract from the Δ observations only in case a subregion does not have enough locations to initialize the procedure. With the current observations or extra samples, we get the mean μ_j and standard deviation ζ_j associated to each subregion σ_j calculated from the samples ($\mu_j = \frac{\sum_{i=1}^{n_j} J(\mathbf{x}_{ji})}{n_j}$ and $\zeta_j = \sqrt{\frac{\sum_{i=1}^{n_j}(J(\mathbf{x}_{ji})-\mu_j)^2}{n_j}}$). We choose $b = \arg\min_{j:\sigma_j \in \mathbb{X}^r} \mu_j$ as the best subregion. Then we solve the following equations to calculate the allocation of Δ to the subregions (Chen et al. 2000): f

$$\frac{N_p}{N_q} = \frac{\zeta_p^2(\mu_b - \mu_p)^2}{\zeta_b^2(\mu_b - \mu_q)^2} \forall p, q \in \mathbb{X}^r, p \neq b, q \neq b \tag{3}$$

$$N_b = \zeta_b \sqrt{\sum_{p=1, p\neq b}^{N} \frac{N_p^2}{\zeta_p^2}}, \sum_{j=1}^{N} N_j = \Delta \tag{4}$$

Algorithm 1 Optimal Computing Budget Allocation (`Part-OCBA`)

Input: Subregion $\sigma_j \subset \mathbb{R}^d : \sigma_j \in \mathbb{X}^r$, W is total number of subregions, objective function $f(\mathbf{x})$, total budget to allocate each iteration Δ, minimum subregion budget $N^0 < \Delta$, locations sampled $(\mathbf{x}_{ji}, f(\mathbf{x}_{ji}))_{i=1}^{n_j}$;

Output: final budget allocation set $\{N_j\} \forall j : \sigma_j \in \mathbb{X}^r$;

for $j : \sigma_j \in \mathbb{X}^r$ **do**

Step 1: Calculate mean μ_j and standard deviation ζ_j for each subregion σ_j using n_j observations (then $N^0 = 0$) if $n_j > N^0$ or sample $N^0 - n_j$ points, Update $\Delta \leftarrow \Delta - N^0$;

end for

Step 2: $b \leftarrow \arg\min_{j:\sigma_j \in \mathbb{X}^r} \mu_j$;

Step 3: Assign the subregion budget solving (3)-(4)

return $\{N_j\} \forall j : \sigma_j \in \mathbb{X}^r$.

Sampling and model estimation At each iteration k, once the number of samples N_{kj} for each subregion has been determined (with procedure `Part-OCBA`), we start the sequential evaluation and update steps. Specifically, given the N_{kj} points to be evaluated in subregion σ_j we start to sequentially select the new locations. The samples are sequentially collected in a way that maximizes the Expected Improvement (Jones et al. 1998). A number N_k^0 of points are used to initialize the estimation of the Gaussian process in each subregion obtaining W models, $\left(\widehat{Y}_j(\mathbf{x}), \widehat{s}_j^2(\mathbf{x}) : x \in \sigma_j\right) \forall j$. At this point, sequential sampling is activated in each subregion for a number $N_k - \left(N^0 - n_j\right)^+$ of iterations. At each iteration, for each subregion, we sample a new location that maximizes the Expected Improvement $\text{EI}_i(\mathbf{x})$, namely:

$$\mathbf{x}_j \in \arg\max_{\mathbf{x}\in\sigma_j} \text{EI}_j(\mathbf{x}) = E\left[\max\left(\left[f^* - \widehat{Y}_j(\mathbf{x})\right]\Phi\left(\frac{f^* - \widehat{Y}_j(\mathbf{x})}{\widehat{s}_j(\mathbf{x})}\right) + \widehat{s}_j(\mathbf{x})\,\phi\left(\frac{f^* - \widehat{Y}_j(\mathbf{x})}{\widehat{s}_j(\mathbf{x})}\right), 0\right)\right]. \tag{5}$$

In (5), f^* is the best value sampled so far. Intuitively, larger regions being less densely sampled, may have larger associated uncertainty, thus contributing to the increased sampling effort. After evaluation, we update the Gaussian process, and proceed until N_{kj} evaluations have been performed. We then proceed verifying the branching conditions and possibly updating the partition. The main procedure for the sampling phase is reported in Algorithm 2.

Algorithm 2 Sequential sampling with Bayesian optimization (`SampleBO`)

Input: Subregion $\sigma_j \in \mathbb{X}^r$, objective function $f(\mathbf{x})$, initialization budget N^0, total budget N^k, locations sampled so far $(\mathbf{x}_{ji}, f(\mathbf{x}_{ji}))_{i=1}^{n_j}$. Budget per iteration per subregion $\{N_{kj}\}_j$;

Output: best location and value $\mathbf{x}^*_{jk} \in \sigma_j$, $f\left(\mathbf{x}^*_{jk}\right)$, final Gaussian process model $\left(\widehat{Y}_j(\mathbf{x}), \widehat{s}^2_j(\mathbf{x})\right)$;

Step 1: Compute the initial required evaluation budget:
if $n_j > N^0$ **then**
  Use the n_j sampled points within the subregion as initializing points for the Gaussian process estimation;
  $t \leftarrow 0$;
else
  Sample $N^0 - n_j$ points using a Latin Hypercube design. Return $\mathbf{x}_{\text{train}} \in \sigma_j$; $f(\mathbf{x}), \forall \mathbf{x} \in \mathbf{x}_{\text{train}}$; $t \leftarrow N^0$;
end if
while $t < N_{kj}$ **do**
  Step 2.1: Estimate the GP using the training data $\{\mathbf{x}_{\text{train}}, \mathbf{y}^i_{\text{train}}\}$, return $\left(\widehat{Y}_j(\mathbf{x}), \widehat{s}^2_j(\mathbf{x})\right)$ for all $\mathbf{x} \in \sigma_j$;
  Step 2.2: Select the next location $\mathbf{x}^*_{\text{EI}} \leftarrow \arg\max_{\mathbf{x}\in\mathbb{X}} \text{EI}_j(\mathbf{x})$; Evaluate and store $f(\mathbf{x}^*_{\text{EI}})$.
  Step 2.3: $t \leftarrow t+1$
end while

Classification Scheme At the end of the sampling stage, we have W Gaussian processes, and we need to estimate the α-quantile for the minimum value of the function in each of the subregions in order to attempt to classify. This problem is generally intractable and several approximations can be used. In this work, without loss of generality, we use a Monte Carlo approach (Algorithm 3).

Algorithm 3 Gaussian process based min quantile estimation (`MCstep`)

Input: Subregion domain $\sigma_j \subset \mathbb{R}^d$, objective function $f(\mathbf{x})$, Gaussian process model $\left(\widehat{Y}_j(\mathbf{x}), \widehat{s}^2_j(\mathbf{x})\right)$, number of Monte Carlo iterations R, number of evaluations per iteration M;

Output: Estimates for the quantile of the minimum function value $\widehat{\underline{\mathrm{Q}}}_j(\alpha), \mathrm{Var}\left(\widehat{\underline{\mathrm{Q}}}_j(\alpha)\right)$;

for $r = 1, \ldots, R$ **do**
  for $m = 1, \ldots, M$ **do**
    Sample uniformly at random a location $\mathbf{x}_{mr}$;
    Evaluate $\left(\widehat{Y}_j(\mathbf{x}_{mr}), \widehat{s}^2_j(\mathbf{x}_{mr})\right)$;
  end for
  minimum α-quantile estimate:

$$\underline{\mathrm{q}}_r(\alpha) \quad = \min_{m=1,\ldots,M} \left(\widehat{Y}(\mathbf{x}_{mr}) - Z_{1-\frac{\alpha}{2}} \sqrt{s^2(\mathbf{x}_{mr})}\right) \tag{6}$$

end for
Build and return the MC-estimates, and confidence interval for the α-quantile of the minimum:

$$\widehat{\underline{\mathrm{Q}}}_j(\alpha) = \frac{1}{R}\sum_{i=1}^{R} q_r(\alpha), \mathrm{Var}\left(\widehat{\underline{\mathrm{Q}}}_j(\alpha)\right) = \frac{\mathrm{Var}(\underline{\mathrm{q}}_r(\alpha))}{R} \tag{7}$$

Once the estimation procedure is complete, we classify a subregion as dominated if:

$$\widehat{\underline{\mathrm{Q}}}_j(\alpha) - Z_{1-\alpha/2}\mathrm{Var}\left(\widehat{\underline{\mathrm{Q}}}_j(\alpha)\right) > \widehat{Y^*}, \widehat{\mathbb{X}^+}_{k+1} \leftarrow \widehat{\mathbb{X}^+}_k \cup \sigma_j \tag{8}$$

Where $\widehat{Y^*}$ is our Monte Carlo estimate of the minimum generated from the Gaussian process model:

$$\underline{\mathrm{Y}}_r = \min_{m=1,\ldots,M}\left(\widehat{Y}\left(\mathbf{x}_{mr}\right)\right), \widehat{Y^*} = \frac{1}{R}\underline{\mathrm{Y}}_r$$

If a subregion is classified as dominated based on the estimate of the minimum, the remaining region ($\mathbb{X}^r$) is updated by removing the dominated subregions. Given the remaining subregions $\widehat{\mathbb{X}^r}_k$, a branching algorithm is called that randomly selects a direction and cuts each subregion along that dimension into B equal volume subregions. In our implementation $B = 2$. We allow subregions to be branched in direction $h = 1,\ldots,d$ only if the size of the hypercube in that dimension is larger than δ, where δ is an input parameter. The sampling phase for the MCTS iteration of Part-MCTS will then terminate, either when the maximum number of function evaluations has been performed, or all the subregions have achieved the minimum branchable volume δ^d, or they have all been classified.

2.3 Algorithm Overview

The procedure in Algorithm 4 summarizes the phases of the proposed approach. In the algorithm, we use the notation $v(\cdot)$ to refer to the volume of a region. Since the regions in Part-MCTS are hyperboxes, volumes are easy to calculate. As already mentioned, algorithm 4 will be repeated at each stage τ of the Monte Carlo Tree Search. It is up to the user to select appropriate values for the total budget evaluation T, and the stage-iteration budget Δ.

Algorithm 4 Part-MCTS sampling approach

Input: Input space $\mathbb{X}$, function $f(\mathbf{x})$, initialization budget N^0. Part sampling budget Δ, number of Monte Carlo iterations R, number of evaluations per iteration M; number of cuts per dimension per subregion B, significance level α, δ;

Set the iteration index $k \leftarrow 1$, Initialize the regions $\widehat{\mathbb{X}^+}_k = \widehat{\mathbb{X}^u}_k = \varnothing$, $\mathbb{X}^r_k \leftarrow \mathbb{X}$;

Output: $\widehat{\mathbb{X}^+}_K \equiv \cup_{j=1}^{\sum_{k=1}^K N_k^d} \widehat{\sigma}_j; \widehat{\mathbb{X}^u}_K \equiv \cup_{j=1}^{\sum_{k=1}^K N_k^u} \widehat{\sigma}_j$, and the associated models $\left(\widehat{Y}_j(\mathbf{x}), \widehat{s}_j^2(\mathbf{x})\right)$;

while $T \geqslant \Delta$ and $\mathbb{X}^r \neq \varnothing$ **do**
  for $j = 1,\ldots,N_k^r$ **do**
    $N_{kj} \leftarrow$ `Part-OCBA`(Δ)
    Execute `SampleBO`$\left(N_{kj}\right)$
    Produce the estimation of relevant quantiles for the minimum function value executing `MCstep`;
    Update the partition:
    if $\widehat{\mathrm{Q}}_j(\alpha) - Z_{1-\alpha/2}\mathrm{Var}\left(\widehat{\widehat{Q}}_j(\alpha)\right) > \widehat{Y^*}$ **then**
      $\widehat{\mathbb{X}^+}_{k+1} \leftarrow \widehat{\mathbb{X}^+}_k \cup \sigma_j, \mathbb{X}^r_{k+1} \leftarrow \mathbb{X}^r_k \backslash \sigma_j$;
    end if
    if $v(\sigma_j) = \delta^d$ **then**
      $\widehat{\mathbb{X}^u}_{k+1} \leftarrow \widehat{\mathbb{X}^u}_k \cup \sigma_j, \mathbb{X}^r_{k+1} \leftarrow \mathbb{X}^r_k \backslash \sigma_j$
    end if
  end for
  $k \leftarrow k+1, T \leftarrow T - \Delta$;
end while

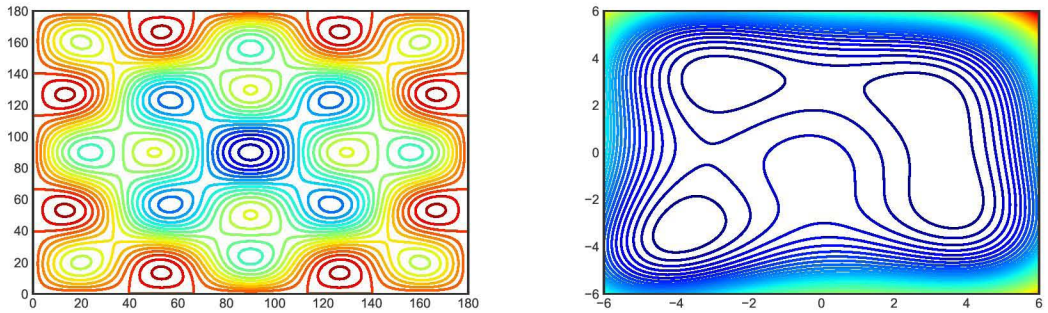
3 Preliminary Numerical Results

In this section, we perform a preliminary analysis of the performance of Part-MCTS when applied to two different objective functions: the Centered Sinusoidal function, and Himmelblau's function. The objective of this analysis

is to show the potential of our algorithm and compare its dynamical sample allocation to the state of the art Kernel-based approach KR-UCT (Yee et al. 2016). Our tests are in a 2-dimensional space, i.e., we emulate a two-step sequential decision making problem since in each stage of the MCTS we take a decision over a single dimension of the function. Note that, at the first iteration, we generate the value on the remaining dimensions using a uniform distribution. Also, for this preliminary analysis, we will assume the stage-decision is one-dimensional.

1. Centered Sinusoidal function: $\mathbb{X} = [0,180] \times [0,180], f(\mathbf{x}) = -2.5 \prod_{i=1}^{2} \sin\left(\frac{\pi x_i}{180}\right) - \prod_{i=1}^{2} \left(\frac{\pi x_i}{36}\right)$, the function has one global minimum and located at $\mathbf{x}^* = (90,90)$ with function value $f^* = -3.5$;
2. Himmelblau's function: $\mathbb{X} = [-6,6] \times [-6,6]$ with $f(\mathbf{x}) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$, the function has four global minima $\mathbf{x}_1^* = (3,2)$, $\mathbf{x}_2^* = (-2.805,3.283)$, $\mathbf{x}_3^* = (-3.779,3.283)$, $\mathbf{x}_4^* = (3.584,-1.848)$, with function value $f^* = 0.0$.

Figure 2 shows the contour plots for both Centered Sinusoidal and Himmelblau's function. We can see clearly from the plots that in Centered Sinusoidal function the minimum is at the center of the feasible space and there is only one optimum, while in the Himmelblau's function there are multiple optimal points and their locations are distributed across the solution space. We performed 30 macro-replications on each test function, a total sampling budget $T = 900, \Delta = 100, N^0 = 10$. We compared our algorithm with KR-UCT (Yee et al. 2016) which we ran using the same T number of observations for each stage (KR-UCT does not require Δ to be defined) and test function.



(a) Centered Sinusoidal function

(b) Himmelblau's function

Figure 2: Contour plots of the test functions.

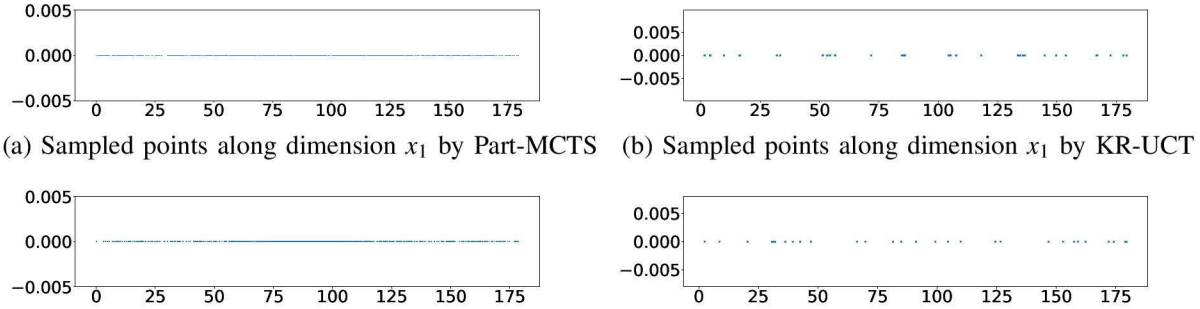
Table 1 shows the statistics about comparison between our algorithm and KR-UCT. In particular, we track four performance measures to characterize our results: $||\widehat{\mathbf{x}}^* - \mathbf{x}^*||$ is the Euclidean distance between the solution resulting from the algorithm and the true optimum, we report the sample average and the standard error associated to this metric. Note that, since the Himmelblau's function has multiple optima, we report as result from each replication the distance of the reported solution to the closest global minimum. The second metric we observe is related to the function value, for each macro-replication, we collect $\widehat{f}^* - f^*$ as the difference between the function value associated to the solution generated by the algorithm and the true minimum (since we are minimizing this difference is non-negative and therefore we do not need the absolute value). Also for this metric, we report the average and the standard error. We can observe that, for both test functions, our algorithm is statistically superior to the benchmark KR-UCT under all considered metrics.

Figure 3 shows the sequence of samples along both dimensions, separately, for both algorithms. Each graph reports the samples from a single macro-replication of the algorithms, and the number of sample points is 900 in both algorithms at both stages. Similar graphs were obtained for the Himmelblau's test function. In the Centered Sinusoidal function case, our algorithm samples densely around the area of the optimum ($x_j^* = 90$), we can see more clear the distinction in Figure 3c. Note that pattern is more evident in general at later stages in the MCTS due to the fact that the sampling is conditioned upon a larger number of variables. The most interesting result certainly comes from observing how dense the samples from KR-UCT are. Being developed for optimization purposes, KR-UCT, which uses a single model to sample across each stage-associated decision space, is strongly biased

Table 1: Comparison between Part-MCTS and KR-UCT

Test	Algorithm	$\|\|\hat{x}-x^*\|\|$		$\widehat{f^*}-f^*$	
		average	std err	average	std err
Centered Sinusoidal	Part-MCTS	1.908	0.413	0.020	0.005
Centered Sinusoidal	KR-UCT	6.054	1.295	0.189	0.048
Himmelblau's	Part-MCTS	0.071	0.018	0.191	0.067
Himmelblau's	KR-UCT	0.487	0.106	8.182	2.211

toward *a* minimum. This explains the higher efficiency in the Himmelblau's test, and also the bias in the centered sinusoidal results, which we analyzed in Table 1.

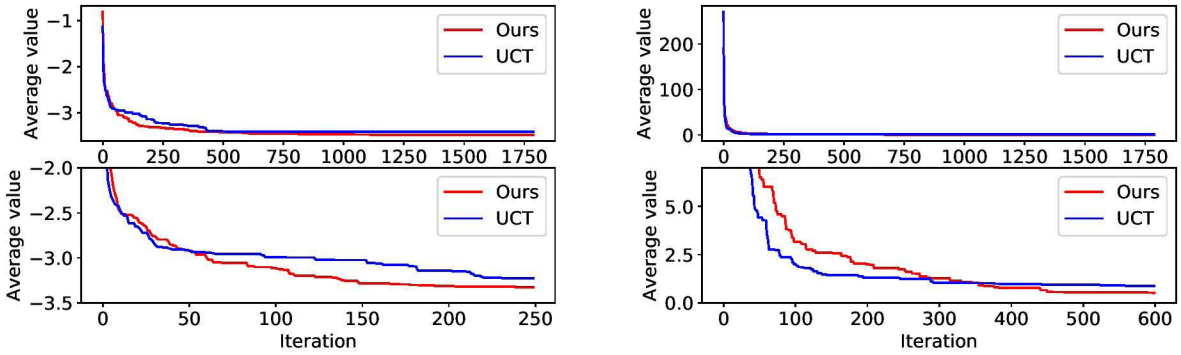


(a) Sampled points along dimension x_1 by Part-MCTS (b) Sampled points along dimension x_1 by KR-UCT

(c) Sampled points along dimension x_2 by Part-MCTS (d) Sampled points along dimension x_2 by KR-UCT

Figure 3: Locations sampled by for the Centered Sinusoidal Test function.

Figure 4 reports the average objective value of the solution from a "fortified" version of the algorithms (averaged across 30 macro-replications) as a function of the sampling effort (i.e., number of sampled locations).



(a) Centered Sinusoidal function (b) Himmelblau's function

Figure 4: Average value along iteration.

Specifically, motivated by the meager performance of KR-UCT in Table 1, we decided to further investigate the reasons for such a difference between the two algorithms with respect to both location and function value. In particular, we ran a fortified version of the MCTS algorithm for both Part-MCTS and KR-UCT. In this version, each algorithm maintains in memory the best solution ever achieved at each stage, and compares the incumbents with it. Such solution is referred to as fortified solution (for more details the interested reader can refer to (Bertsekas

2020), Cfr. Chapter 2, section 2.3.3, pp. 73-76). As expected, we notice that the performance of both algorithms improves in the fortified version. The top graph in Figure 4a shows the evolution of the fortified value for the Centered Sinusoidal test function. The bottom graph shows the zoom for the same test. Looking at the bottom graph, we can see that Part-MCTS converges faster than KR-UCT and appears to steadily achieve better results than KR-UCT around the 50^{th} sample. Both algorithms exhibit convergence to the true minimum around the 500^{th} observation. The situation is different if we observe the bottom graph for the Himmelblau's test function. Initially, Part-MCTS shows slower convergence than KR-UCT. Nonetheless, the algorithm achieves better results starting from the 350^{th} observation. Also in this case, the difference between the algorithms vanishes around the 500^{th} sample. We investigated further the apparent inefficiency of Part-MCTS with respect to the Himmelbalu's function. In fact, this test function has multiple optima. We observed that Part-MCTS tends to spread the samples, increasing the ability to identify, simultaneously, multiple promising regions. On the other hand, KR-UCT is designed to converge toward *an* optimal solution. This may seem like an inefficiency. Nonetheless, many Approximate Dynamic Programming approaches based on tree exploration, can make effective use of a sampler able to suggest multiple solutions by, for example, maintaining multiple branches active. In such a case Part-MCTS can be even more valuable (Bertsekas 2020).

4 Conclusions

We present for the first time the algorithm Part-MCTS for sampling continuous decision spaces within the framework of Monte Carlo Tree Search (MCTS). Specifically, given an input space, a simulation tool, an evaluation budget, and a significance level, Part-MCTS attempts to classify the input space into dominated, undecided, and remaining subregions. The algorithm relies on a collection of Gaussian processes, each defined over a single subregion, used to estimate the unknown reward function in unsampled areas of the input. As a result, the surrogate will have the flexibility to capture complex landscapes of the objective by using a variety of correlation functions. While Part-MCTS does not provide a means to decide the total stage-budget, it uses Optimal Computing Budget Allocation to distribute the stage-budget, divided across stage-iterations, among active subregions. Bayesian optimization is used in each subregion to sample locations. The numerical results demonstrate the ability of Part-MCTS to not only achieve good quality solutions, but also its ability to capture the presence of multiple competing solutions. The next steps in our research will be: (i) extend our MCTS to the case where multiple branches are maintained at each stage; (ii) automate the selection of the stage budget and number of iterations per stage; (iii) extend the algorithm to stochastic systems; (iv) improving accuracy by continuously sampling in already "classified" regions in the partitioning step.

REFERENCES

Archibald, C., A. Altman, and Y. Shoham. 2009. "Analysis of a Winning Computational Billiards Player.". In *IJCAI*, Volume 9, 1377–1382. Citeseer.

Bertsekas, D. P. 2020. *Rollout, Policy Iteration, and Distributed Reinforcement Learning*. Athena Scientific Belmont.

Bertsekas, D. P. et al. 2000. *Dynamic programming and optimal control: Vol. 1*. Athena Scientific Belmont.

Billings, W. M., B. Hedelius, T. Millecam, D. Wingate, and D. Della Corte. 2019. "ProSPr: democratized implementation of alphafold protein distance prediction network". *BioRxiv*:830273.

Chen, C.-H., D. He, M. Fu, and L. H. Lee. 2008. "Efficient simulation budget allocation for selecting an optimal subset". *INFORMS Journal on Computing* 20(4):579–595.

Chen, C.-H., J. Lin, E. Yücesan, and S. E. Chick. 2000. "Simulation budget allocation for further enhancing the efficiency of ordinal optimization". *Discrete Event Dynamic Systems* 10(3):251–270.

Chipman, H., E. George, and R. McCulloch. 2002, 07. "Bayesian Treed Models". *Machine Learning* 48:299–320.

Denison, D., N. Adams, C. Holmes, and D. Hand. 2002. "Bayesian partition modelling". *Computational Statistics & Data Analysis* 38(4):475–485. Nonlinear Methods and Data Mining.

Fu, M. C. 2018. "Monte Carlo tree search: A tutorial". In *2018 Winter Simulation Conference (WSC)*, 222–236. Institute of Electrical and Electronics Engineers, Inc.

Gramacy, R. B., and H. K. H. Lee. 2005. "Bayesian Treed Gaussian Process Models with an Application to Computer Modeling". Technical report.

Jones, D. R., M. Schonlau, and W. J. Welch. 1998. "Efficient Global Optimization of Expensive Black-Box Functions". *Journal of Global optimization* 13(4):455–492.

Kawaguchi, K., L. P. Kaelbling, and T. Lozano-Pérez. 2015. "Bayesian optimization with exponential convergence".

Kim, B., K. Lee, S. Lim, L. Kaelbling, and T. Lozano-Pérez. 2020. "Monte Carlo tree search in continuous spaces using Voronoi optimistic optimization with regret bounds". In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 34, 9916–9924.

Kim, H.-M., B. K. Mallick, and C. C. Holmes. 2005. "Analyzing Nonstationary Spatial Data Using Piecewise Gaussian Processes". *Journal of the American Statistical Association* 100(470):653–668.

Kocsis, L., and C. Szepesvári. 2006. "Bandit based monte-carlo planning". In *European conference on machine learning*, 282–293. Springer.

Li, Y., M. Fu, and J. Xu. 2019. "Monte Carlo tree search with optimal computing budget allocation". In *2019 IEEE 58th Conference on Decision and Control (CDC)*, 6332–6337. Institute of Electrical and Electronics Engineers, Inc.

Liang, W., and H. Lee. 2018, 09. "Bayesian nonstationary Gaussian process models via treed process convolutions". *Advances in Data Analysis and Classification* 13.

Mansley, C., A. Weinstein, and M. Littman. 2011. "Sample-based planning for continuous action markov decision processes". In *Proceedings of the International Conference on Automated Planning and Scheduling*, Volume 21.

Mathesen, L., G. Pedrielli, S. H. Ng, and Z. B. Zabinsky. 2021. "Stochastic optimization with adaptive restart: A framework for integrated local and global learning". *Journal of Global Optimization* 79(1):87–110.

Pedrielli, G., S. Wang, and S. H. Ng. 2020. "An extended Two-Stage Sequential Optimization approach: Properties and performance". *European Journal of Operational Research* 287(3):929–945.

Santner, T. J., B. J. Williams, and W. I. Notz. 2013. *The design and analysis of computer experiments*. Springer Science & Business Media.

Smith, M. 2007. "PickPocket: A computer billiards shark". *Artificial Intelligence* 171(16-17):1069–1091.

Wang, L., S. Xie, T. Li, R. Fonseca, and Y. Tian. 2019, 06. "Sample-Efficient Neural Architecture Search by Learning Action Space". *arXiv preprint arXiv:1906.06832*.

Wang, L., Y. Zhao, Y. Jinnai, Y. Tian, and R. Fonseca. 2019, 03. "AlphaX: eXploring Neural Architectures with Deep Neural Networks and Monte Carlo Tree Search". *arXiv preprint arXiv:1903.11059*.

Wang, Z., B. Shakibi, L. Jin, and N. Freitas. 2014. "Bayesian multi-scale optimistic optimization". In *Artificial Intelligence and Statistics*, 1005–1014. PMLR.

Xiao, H., H. Chen, and L. H. Lee. 2019. "An efficient simulation procedure for ranking the top simulated designs in the presence of stochastic constraints". *Automatica* 103:106–115.

Xiao, H., F. Gao, and L. H. Lee. 2020. "Optimal computing budget allocation for complete ranking with input uncertainty". *IISE Transactions* 52(5):489–499.

Xiao, H., and S. Gao. 2018. "Simulation budget allocation for selecting the top-m designs with input uncertainty". *IEEE Transactions on Automatic Control* 63(9):3127–3134.

Xiao, H., L. H. Lee, D. Morrice, C.-H. Chen, and X. Hu. 2020. "Ranking and selection for terminating simulation under sequential sampling". *IISE Transactions*:1–16.

Yamamoto, M., S. Kato, and H. Iizuka. 2015. "Digital curling strategy based on game tree search". In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, 474–480. Institute of Electrical and Electronics Engineers, Inc.

Yee, T., V. Lisỳ, and M. H. Bowling. 2016. "Monte Carlo Tree Search in Continuous Action Spaces with Execution Uncertainty.". In *IJCAI*, 690–697.

Zabinsky, Z. B., T.-Y. Ho, and H. Huang. 2019. "Integrating heuristics and approximations into a branch and bound framework". In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, 774–779. Institute of Electrical and Electronics Engineers, Inc.

Zhang, J., Z. Li, C. Wang, D. Zang, and M. Zhou. 2016. "Approximate simulation budget allocation for subset ranking". *IEEE Transactions on Control Systems Technology* 25(1):358–365.

Zhang, S., L. H. Lee, E. P. Chew, J. Xu, and C.-H. Chen. 2015. "A simulation budget allocation procedure for enhancing the efficiency of optimal subset selection". *IEEE Transactions on Automatic Control* 61(1):62–75.

AUTHOR BIOGRAPHIES

MENGHAN LIU is a Ph.D. student in the School of Computing, Informatics, and Decision Systems Engineering at the Arizona State University. Her email address is mliu126@asu.edu.

YUMENG CAO is a Ph.D student in the School of Mathematical and Statistical Sciences at the Arizona State University. Her email address is ycao108@asu.edu.

GIULIA PEDRIELLI is Assistant Professor in the School of Computing, Informatics, and Decision Systems Engineering at the Arizona State University. She holds a Ph.D. degree in Mechanical Engineering from the Politecnico di Milano. She is a

member of IEEE and INFORMS. Her email address is giulia.pedrielli@asu.edu.