Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

ADAPTIVE RANKING AND SELECTION BASED GENETIC ALGORITHMS FOR DATA-DRIVEN PROBLEMS

Kimia Vahdat
Sara Shashaani

Edward P. Fitts Department of Industrial and Systems Engineering
NC State University
915 Partners Way
Raleigh, NC 27606, USA

ABSTRACT

We present ARGA–Adaptive Robust Genetic Algorithm–to optimize simulation problems with binary variables affected by input uncertainty and Monte Carlo noise. In this method, a surviving population of designs evolves as more information about the high-dimensional problem affected by stochasticity becomes available. In every population, ARGA conducts a ranking and selection with a debiasing mechanism of fitness values using fast iterated bootstraps economized with control variates. Debiasing reduces the model risk induced by input uncertainty bias, leading to a more accurate ranking of the current surviving designs. Given the double loop of function evaluations, we incorporate an adaptive budget allocation throughout the search only if the current population's proximity to optimality signals the need for a smaller standard error. In that case, we identify where to allocate additional replications: the input model of a current surviving design that is most responsible for risk. The empirical results with a fixed optimization budget demonstrate that ARGA obtains significantly better solutions in a feature selection problem in various datasets.

1 INTRODUCTION

Simulation models are widely used in various fields to evaluate, compare, and choose the best system designs or strategies based on their estimated performance. Ranking and selection (R&S) methods are often employed to ensure computationally efficient comparison. R&S distributes simulation efforts among different designs to achieve a predetermined confidence level for choosing the best design. There are different ways to classify R&S procedures, but the boundaries between them are not always clear-cut (Hunter and Nelson 2017; Pasupathy and Ghosh 2013). One way to categorize them is by their approach to ensuring selection quality: fixed-precision or fixed-budget. The fixed-precision procedures run until they meet a guarantee on the optimality gap between the chosen system and the actual best system. The fixed-budget procedures aim to allocate a fixed amount of computational resources to minimize a loss function that penalizes incorrect selection. In this paper, we employ an optimal computing budget allocation (OCBA) during optimization that, though reminiscing the fixed-budget procedures for statistical guarantees, can stop before reaching that maximum budget guided by an adaptive sampling philosophy.

A salient feature of our proposed method is its goal of performing R&S throughout optimization efficiently *while maintaining robustness*, i.e., with consideration for model risk. An important source of model risk is input uncertainty (IU)–the risk of misspecifying the input distribution. Traditionally, R&S methods assume the true input distribution is known, and the inference from simulation outputs is only affected by stochastic uncertainty (SU). However, especially with high-dimensional input data, IU can significantly impact the inference and misguide decision-making (Corlu and Biller 2015; Song and Nelson 2019). A consequence of IU is that it may be impossible to identify the correct best design even with infinite computing effort. In response, we explore an integration of efficient R&S methods to address complex simulation problems affected by both SU and IU *during optimization*.

We compute and reduce the IU bias when input distributions are empirical CDFs of the high-dimensional data. Our *debaising* procedure comes at an increased computational cost. However, the adaptive budget allocation enables economic incorporation of the SU and IU intricacies. In this paper, we use a well-known global binary optimization engine, the genetic algorithm (GA). Given multiple input models (generated from bootstrapped data) for each design in the surviving population of the GA, the adaptive allocation also entails deciding that additional replication to which design with which input model maximizes the overall efficiency of the optimization task. Robust OCBA (R-OCBA), proposed by Gao et al. (2017), suggests a way to link the computation and utilization of multiple input models when allocating budgets for an estimation problem. Our proposed framework integrates R-OCBA with adaptive choice of budget size during optimization. We weave this integrated approach into the inner dynamics of GA for decision-making in noisy simulation environments and call the new stochastic GA–Adaptive Robust GA.

In summary, ARGA leverages (i) the iterative design generation and selection operations within GA, (ii) a variance-reduced fast-iterated bootstrapping (FIB) technique to reduce the IU bias, and (iii) an adaptive sampling scheme that increases the budget only *when* and *where* necessary. To our knowledge, the current work is the first study that enables the interaction between the IU and adaptive sampling inside an optimization regime. This work is a continuation of our previous robust estimation work (Vahdat and Shashaani 2021; Shashaani and Vahdat 2022) to handle the stochasticity of data-driven problems. Beyond general simulation optimization regimes, the application of this method is also on machine learning (ML), where the ML model can be viewed as a black-box simulation. Debiasing with nonparametric input models established in (Vahdat and Shashaani 2023) can reduce the risk when building ML models. In this paper, we use ARGA to minimize the loss of a learner with the right choice of features in a dataset

The organization of the paper is as follows. In Section 2, we review the literature on R&S techniques, GA, and R-OCBA. Section 3 elaborates on ARGA and provides evidence of its applicability. Lastly, numerical results in Section 4 demonstrate the success and shortcomings of ARGA for a feature selection problem with simulation optimization, and Section 5 concludes the paper.

2 PRELIMINARIES AND RELATED WORK

In this section, we review R&S history with budget allocation and IU, the GA processes, and existing work in using R&S within GA. We also introduce the notations used in the remainder of the paper.

2.1 Ranking and Selection with Input Uncertainty

R&S methods are commonly used to compare designs and select the best based on expected performance (Bechhofer 1995). Recent literature studies the implications of parameter uncertainty on subset selection procedures and the effect of IU on identifying the best designs (Fan et al. 2020; Song et al. 2015; Wu and Zhou 2017). Song et al. (2015) consider the impact of IU on the indifference zone parameter, whereas Zhang and Ding (2016) propose various procedures such as the knowledge gradient policy to handle a Bayesian R&S under IU. For a more in-depth review of the current advancements in R&S, refer to surveys by Corlu et al. (2020) and Hong et al. (2021).

With a fixed number of designs $x_t,\ t \in \{1, \cdots, m\}$, we let each design be evaluated under b input distributions $\hat{F}_i^\star,\ i \in \{1, \cdots, b\}$ obtained from the $i-$th bootstrapped dataset, to incorporate the effect of IU on selection. The bootstrapped distributions are ideally generated with common random numbers so that the same uncertainty space is utilized across all designs. Denote the total simulation budget for each design with n, minimum computing budget of each scenario with n_0, and the number of allocated simulation replications to design t of *scenario* (input distribution) i with $n_{t,i}$. Then $n_{t,i}$ runs of the simulation model under design t using scenario i yields simulation outputs $Y_{t,i,j} := Y_j(\hat{F}_i^\star, x_t),\ j \in \{1, \cdots, n_{t,i}\}$. We can then estimate the expected value and variance of each design t's performance under scenario i with

$$\bar{Y}_{t,i}(n_{t,i}) = \frac{1}{n_{t,i}} \sum_{j=1}^{n_{t,i}} Y_{t,i,j}, \text{ and } \hat{\sigma}^2_{t,i}(n_{t,i}) = \frac{1}{n_{t,i}-1} \sum_{j=1}^{n_{t,i}} (Y_{t,i,j} - \bar{Y}_{t,i})^2.$$

We assume that $Y_{t,i,j}$'s for each design t and scenario i follow a normal distribution with mean $\theta_{t,i}$ and variance $\sigma^2_{t,i}$. This assumption is not restrictive as the simulation output often comprises a sum of terms, e.g., in ML, it comprises the sum of squared prediction errors on a test dataset. R&S typically ranks designs based on their overall (across scenarios) average performance $\bar{\bar{Y}}_t = \frac{1}{b}\sum_{i=1}^{b} \bar{Y}_{t,i}(n_{t,i})$. Hence, the selected best design is one with the smallest overall average performance whose index we mark as $t^* := \operatorname{argmin}_{t\in\{1,\ldots,m\}} \bar{\bar{Y}}_t$. The choices b for number of scenarios and m for number of designs will be kept fixed throughout this paper and excluded from notations for simplicity.

R-OCBA suggests a strategy in R&S for optimal allocation of computation budget to different scenarios of each design to *robustly* maximize the probability of correct selection (PCS) of the true best design x^* under a fixed total budget. R-OCBA changes the best design definition as $\operatorname{argmin}_{t\in\{1,\cdots,m\}} \max_{i\in\{1,\cdots,b\}} Y_{t,i}$, i.e., one with the best (smallest) worst-case scenario. Focusing on the worst-case scenario is a common approach in increasing robustness (Ghaoui et al. 2003). The PCS, which we wish to maximize, is defined here as the probability that the best design's worst-case scenario is better than all other designs' worst-case scenarios given b bootstrap distributions:

$$\begin{aligned} \max_{(n_{t,i},\ t=1,\cdots,m,i=1,\cdots,b)} \mathbb{P}\left\{ \max_{i\in\{1,\ldots,b\}} \bar{Y}_{t^*,i}(n_{t^*,i}) < \min_{t\in\{1,\ldots,m\}} \max_{i'\in\{1,\ldots,b\}} \bar{Y}_{t,i'}(n_{t,i'}) \middle| \hat{F}^\star_1, \cdots, \hat{F}^\star_b \right\} \\ \text{s.t.} \sum_{i=1}^{b} n_{t,i} \le n \quad \forall t \in \{1,\cdots,m\}. \end{aligned} \tag{1}$$

Note, unlike the usual practice in R&S, where the total budget across all designs is limited, we limit the budget of each design to provide greater flexibility for the larger optimization task. It will be established later that in our optimization routine, R&S is invoked in every iteration to rank the survivors. Problem (1) seeks $n_{t,i}$ for each design and scenario, within the allowable budget, for each case that attains the highest PCS, conditioning on a set of sampled scenarios, i.e., bootstraps. For ease of exposition, we henceforth drop the sample size as the argument of sample mean and sample variance statistics.

Let $i_t := \operatorname{argmax}_i \bar{Y}_{t,i}(n_{t,i})$ be the index of the worst-case scenario of design t. To have a means of comparison between designs using the normality assumption of the simulation outputs, define the discrepancy between the i-th scenario of the t-th design and that of the i'-th scenario of the t'-th design as

$$R_{[t,i],[t',i']} = \frac{|\bar{Y}_{t,i} - \bar{Y}_{t',i'}|}{\hat{\sigma}_{t,i}/\sqrt{n_{t,i}} + \hat{\sigma}_{t',i'}/\sqrt{n_{t',i'}}}. \tag{2}$$

Using the discrepancy measure in (2), we term the *most sensitive scenario* as the scenario of the best design t^* with minimum discrepancy from all other designs' worst-case scenarios as

$$\bar{i}_{t^*} = \operatorname*{argmin}_{i\in\{1,\ldots,b\}} \min_{t\in\{1,\ldots,m\},t\neq t^*} R_{[t,i_t],[t^*,i]}.$$

Mainly, $\bar{i}_{t^*}$ denotes the best design's most sensitive input model (scenario), which may change the overall competitiveness and rank of the best design with small shifts following an added budget. To better clarify the difference between $\bar{i}_{t^*}$ and i_{t^*}, note that i_{t^*} is the worst scenario of t^* in terms of expected average performance, but $\bar{i}_{t^*}$ is most sensitive using discrepancy in (2). We further define the *sensitive design* $\bar{t}$ as one whose worst-case scenario has the least discrepancy from the best design's worst-case scenario:

$$\bar{t} = \operatorname*{argmin}_{t\in\{1,\ldots,m\},t\neq t^*} R_{[t,i_t],[t^*,i_{t^*}]}.$$

Here too, $\bar{t}$ is a design that is more likely to be affected by an added budget. The R-OCBA procedure proves that an asymptotically approximated version of (1) can be optimized via the following steps:

1. Use n_0 simulation calls for all scenarios and designs to compute $\bar{Y}_{t,i}$ and $\hat{\sigma}_{t,i}$.
2. If the $\sum_{i=1}^{b} n_{t,i} \geq n$ for all $t \in \{1, \cdots, m\}$, go to step 3, otherwise repeat:
   (a) Compute $\hat{A}_1 = \sum_{i=1}^{b} n_{t^*,i}^2 / \hat{\sigma}_{t^*,i}^2$ and $\hat{A}_2 = \sum_{t=1,t \neq t^*}^{m} n_{t,i_t}^2 / \hat{\sigma}_{t,i_t}^2$, derived from the optimality conditions for (1) and representing the partial derivative of the approximated PCS with respect to the $n_{t,i}$'s. At the optimum allocation, we must have $\hat{A}_1 = \hat{A}_2$.
   (b) If $\hat{A}_1 < \hat{A}_2$, allocate budget to the most sensitive scenario of the best design, $\bar{i}_{t^*}$.
   (c) If $\hat{A}_1 > \hat{A}_2$, allocate budget to the worst-case scenario of the sensitive design $i_{\bar{t}}$.
   (d) Update the sample means and variances accordingly.
3. Report the best design, x_{t^*}.

R-OCBA enhances the efficiency of the evaluation process with the goal of maximizing robustness (addressing the best worst-case). We employ a variant of it for allocating additional computing budget in an adaptive way. R&S is an exhaustive search procedure apt for use within *each population* of GA to rank a fixed number of surviving designs. *Therefore, while we conduct simulation optimization across iterations choosing designs with best average performance, we approach the budget allocation in each iteration as a R&S and maintain robustness by using worst-case performance.* We will next describe the GA.

2.2 The Genetic Algorithm

The GA is a popular approach that can help navigate the optimization of complex systems by mimicking the process of natural selection and evolution (Holland 1992). The key idea behind the GA is that the fittest designs are more likely to survive and pass on their genetic structure to the next generation, leading to a gradual improvement of the *fitness* (performance) over time. Since GA is effective in exploring large and complex design spaces, it has been widely used for simulation optimization with successful outcomes reported in qualitative and quantitative case studies (Boesel and Nelson 1998; Azadivar and Tompkins 1999; Nazzal et al. 2012). GA is a heuristic technique that is shown to converge asymptotically, in terms of visiting all solutions infinitely often (Bhandari et al. 1996). Its effectiveness depends on several factors, such as the choice of evaluation metric, the population (generation) size, the stopping criteria, and the characteristics of the problem being solved (Mitchell 1998). Researchers have also combined GA with R&S procedures to enhance the selection procedure of GA with probabilistic guarantees (Gupta 1965; Xiao and Lee 2014; Kou et al. 2021). These techniques have proven effective in increasing the accuracy of GA in solving complex problems (Liu and Cramer 2018). The GA involves five key operations:

Initialization randomly select a population of m designs,

Evaluation evaluate each design and return their mean fitness value,

Selection sample part of the next generation from current generation based on fitness-based ranks,

Crossover combine attributes of two randomly chosen designs to generate new designs,

Mutation add or delete an attribute in a few randomly chosen designs.

Selection, crossover, and mutation occur a certain number of times in each new population based on predefined probabilities (parameters) to form the next population. The algorithm stops when either there is no progress for a certain number of successive generations or the maximum permitted generation count is reached. In standard GA, each individual is represented with a binary vector indicating the inclusion and/or exclusion of attributes. The general formulation of GA and its bit-based definition of designs makes it a suitable optimization engine for high-dimensional binary search. We exploit this characteristic in Section 4 and showcase the performance of GA in a binary space.

The selection process chooses a sub-population for the next generation. At iteration 1, GA initiates its optimization process by uniformly sampling the feasible space. At the later iterations, a selection process randomly chooses a subset of designs with better estimated fitness to generate the next population (Miller and Goldberg 1995). The chosen group is subjected to crossover and mutation methods to explore other possible designs and avoid being stuck in a neighborhood. The design identified as the best survives in the next generation with probability 1. However, due to mutation, there is a nonzero probability that it will be moved out of the population. The Q-tournament method by Schmitt (2001) is commonly used for selection. It selects individuals based on their rank in the current population, where those with higher ranks are more likely to be selected. Therefore, the accuracy of fitness evaluation that ranks the designs plays a critical role in survival probability in each GA iteration.

While successful in deterministic optimization, GA is challenged to determine the best among a set of surviving designs when dealing with stochastic problems. In non-stochastic problems, the fitness value of each solution candidate is precise, and sorting in descending order imposes no risk toward search. However, in stochastic problems, fitness is *estimated* and it needs to be clarified if there is a statistical guarantee that one design is better than another. This guarantee requires considering the bias and variance of IU and SU that can affect the accuracy of pairwise comparisons between their point estimates with a limited budget.

3 THE ADAPTIVE ROBUST GA

To improve the evaluation process for data-driven and stochastic problems, we seek to debias the fitness estimates and use R-OCBA in an efficient manner within GA. Therefore, ARGA has three main components:

1. Implementing a robust R&S within GA via bootstrapped input models statistically guarantees the significance of the surviving design's estimated fitness compared to other designs.
2. Inside the R&S procedure, we devise a debiasing procedure applying an FIB step with control variates to efficiently calculate the induced bias during estimation given a fixed budget.
3. We then use an adaptive sampling rule that examines the current population's proximity to optimality and determines whether the debiased estimated values in the current iteration require more precision. If that is the case, we allocate more budget to a design with an input model that is more likely to strike a balance between the statistical error and optimality gap. We repeat this inspection until obtaining sufficient precision or exhausting the total per-iteration per-design budget n.

Component 1 is, to our knowledge, the first attempt at implementing a nested setting within GA to extract IU information. We adopt the notation introduced for R&S but add an index k for iteration (interchangeably referred to as generation or population) in all the metrics, e.g., $X_{k,t}$ is the t-th design at iteration k of the GA (we use capital letter X to reflect that it is now random and dependent on the random quantities evaluated during one run of the GA algorithm), and $n_{k,t,i}$ number of replications for scenario i of design t in iteration k. The sample means and sample variances in each iteration will be denoted as $\bar{Y}_{k,t,i}$ and $\hat{\sigma}^2_{k,t,i}$ accordingly. The total budget for each design n is fixed for all iterations of GA, while setting the minimum budget to evaluate any design under any scenario as n_0. The rank of designs in population k follows from the estimated expected value and variance of each design's fitness. Given that each design is evaluated under different input models, the ranking will use their average simulation outputs, i.e., $\bar{\bar{Y}}(X_{k,t}) := \bar{\bar{Y}}_{k,t} = \frac{1}{b}\sum_{i=1}^{b}\bar{Y}_{k,t,i}$. We will denote the best design up to iteration k as

$$X_k^* = \underset{X_{k',t}:k'\leq k,\ t\in\{1,\ldots,m\}}{\operatorname{argmin}} \bar{\bar{Y}}(X_{k',t}).$$

Note, this is different from typical robust procedures that label the design with the lowest worst-case performance as the best. This is because we handle robustness differently by debiasing $\bar{Y}_{k,t,i}$ during the evaluation step in Component 2. Following the debiasing and before the selection step, Component 3 conducts a post-fitness evaluation to provide an opportunity to efficiently improve the precision of the population's estimated performance. As in R-OCBA, here the allocation follows a typical worst-case performance to reduce the risk. The following sections provide more details about these two components.

3.1 Evaluation Step: Debiasing

Vahdat and Shashaani (2023) developed a FIB that characterizes the bias in the simulation output due to error in input distributions. The effect of bias in simulation outputs can be significant in smaller datasets (Lam 2016). Recall that at a given iteration k, we aim to compare any two solutions $X_{k,t}$ and $X_{k,t'}$ with their estimated performance and rank them via R&S. Let the true input distribution F be unknown and define $\theta_{k,t}(F) := \mathbb{E}_Y[\bar{Y}(F, X_{k,t})]$ as the true mean performance of design t in iteration k. For the remainder of this section, we drop $X_{k,t}$ and the first two indexes k and t from all notations, as they do not change during debiasing. Recall that the budget $n_i = n_0$ for all scenarios $i = 1, 2, \dots, b$ throughout this step.

On the basis of the bootstrap theory (Efron 1979), we decompose the true mean performance as follows $\theta(F) \approx \theta(\hat{F}) - \beta(\hat{F}) - \gamma(\hat{F})$ with two terms that approximate bias in combination. In particular, the first term $\beta(\hat{F}) := \mathbb{E}_{\hat{F}}[\theta(\hat{F}^{\star})] - \theta(\hat{F})$ approximates the true bias $\beta(F) = \theta(\hat{F}) - \theta(F)$ with approximation error $\gamma(F) = \beta(F) - \beta(\hat{F})$, that is itself approximated with the approximation error between biases in nested bootstrap input distributions, i.e., $\gamma(\hat{F}) := (\mathbb{E}_{\hat{F}}[\theta(\hat{F}^{\star})] - \theta(\hat{F})) - \mathbb{E}_{\hat{F}}[\mathbb{E}_{\hat{F}^{\star}}[\theta(\hat{F}^{\star\star}) \mid \hat{F}^{\star}]]$. Note, $\mathbb{E}_{\hat{F}}[\theta(\hat{F}^{\star})]$ is an expectation of θ values with respect to the sampling distribution of $\hat{F}$, i.e., using datasets $(Y^{\star}_{1,j},\ j = 1, 2, \dots, n_1), (Y^{\star}_{2,j},\ j = 1, 2, \dots, n_2) \dots$ drawn from $\hat{F}$ each forming an empirical distribution denoted by $\hat{F}^{\star}_1, \hat{F}^{\star}_2, \dots$. Similarly, $\mathbb{E}_{\hat{F}}[\mathbb{E}_{\hat{F}^{\star}}[\theta(\hat{F}^{\star\star}) \mid \hat{F}^{\star}]]$ takes an expectation of θ values in an additional nested layer, conditional on the first nested bootstrap's empirical distribution, and then integrated out with respect to $\hat{F}$'s sampling distribution. Vahdat and Shashaani (2023) show that under mild conditions (to allow interchanging of expectations) we can equivalently write a similar expression for the outputs directly, i.e., by fixing the simulation seed that produces the $Y_j(\cdot)$-th output for a given input model:

$$Y_j(F) \overset{\text{d}}{\approx} Y_j(\hat{F}) - (W_j(\hat{F}) + \epsilon_1(\hat{F})) - (V_j(\hat{F}) + \epsilon_2(\hat{F})), \tag{3}$$

where $\overset{\text{d}}{\approx}$ denotes weak approximation (in distribution) with a random variable, $W_j := \mathbb{E}_{\hat{F}}[Y(\hat{F}^{\star})] - Y(\hat{F})$ and $V_j := \mathbb{E}_{\hat{F}}[Y(\hat{F}^{\star}) - Y(\hat{F})] - \mathbb{E}_{\hat{F}}[\mathbb{E}_{\hat{F}^{\star}}[Y(\hat{F}^{\star\star}) \mid \hat{F}^{\star}] - Y(\hat{F}^{\star})]$ are two random variables on the right-hand-side accompanied by $\epsilon_1(\hat{F})$ and $\epsilon_2(\hat{F})$ that represent mean-zero stochastic noise random variables. This means that subtracting the two parentheses in (3) from each simulation output leads to a debiased output value. The point of performing this step for each output value instead of the overall estimator, is to enable use variance reduction techniques such as common random numbers and correlating the deeper layers with the earlier ones. Vahdat and Shashaani (2023) also show that when the problem is purely data-driven such as in machine learning applications, then the estimated bias value from each input model can be quite variable and rather than subtracting fixed bias estimates from the nominal outputs (with the empirical distribution from the original data), it is better to pretend that each of the first layer bootstrapped input models $\hat{F}^{\star}_1, \hat{F}^{\star}_2, \dots$ are the actual nominal input model and repeat the procedure above to compute the bias for each input model separately. This leads to computing the debiased values $Y^{\text{d}}_{i,j} := Y^{\text{d}}_j(\hat{F}^{\star}_i)$ with

$$Y^{\star\text{d}}_{i,j} = Y^{\star}_{i,j} - \hat{W}_{i,j} - \hat{V}_{i,j}, \tag{4}$$

where $\hat{W}_{i,j}$ estimates $W_{i,j}$ using b' inner bootstraps, i.e., with resampled data $(Y^{\star\star}_{i,i',j},\ j = 1, 2, \dots, n_0)$ that form the empirical distributions $\hat{F}^{\star\star}_{i,i'}$ for all $i = 1, 2, \dots, b$ and $i' = 1, 2, \dots, b'$. Concretely, the first term of $W_{i,j} = \mathbb{E}_{\hat{F}}[\mathbb{E}_{\hat{F}^{\star}}[Y(\hat{F}^{\star\star}) \mid \hat{F}^{\star}]] - \mathbb{E}_{\hat{F}}[Y(\hat{F}^{\star})]$ is estimated with $(bb')^{-1} \sum_{i=1}^{b} \sum_{i'=1}^{b'} Y^{\star\star}_{i,i',j}$. Similarly, the second term of $V_{i,j} = \mathbb{E}_{\hat{F}}[\mathbb{E}_{\hat{F}^{\star}}[Y(\hat{F}^{\star\star}) \mid \hat{F}^{\star}] - Y(\hat{F}^{\star})] - \mathbb{E}_{\hat{F}}[\mathbb{E}_{\hat{F}^{\star}}[(\mathbb{E}_{\hat{F}^{\star\star}}[Y(\hat{F}^{\star\star\star}) \mid \hat{F}^{\star\star}] - Y(\hat{F}^{\star\star})) \mid \hat{F}^{\star}]]$ is estimated requiring one more nested layer to obtain $\hat{V}_{i,j}$. However, in this layer we only use one bootstrapped dataset, i.e., $(Y^{\star\star\star}_{i,i',j},\ j = 1, 2, \dots, n_0)$ drawn from $\hat{F}^{\star\star}_{i,i'}$ due to the fact that the error rate of this bias estimation procedure $\mathcal{O}_p((nb')^{-1/2})$ does not depend on repeats in the deeper layer, which is why it is called FIB or the *warp-speed* double-bootstrap (Chang and Hall 2015).

The suggested approach for debiasing the output estimator has limitations, as it raises the possibility of increased variance in the performance estimate. To address this issue, we propose a control variate method for $\hat{Z}_{i,j} = \hat{W}_{i,j} + \hat{V}_{i,j}$, which involves adding a multiplier of a variable that is centered (has mean zero) and correlated to the estimator, i.e., $\hat{Z}^{\mathrm{c}}_{i,j} := \hat{Z}_{i,j} + \alpha(Y^{\star}_{i,n_i+1} - Y^{\star}_{i,j})$. Importantly, $Y^{\star}_{i,n_i+1}$ is the model output independent from simulation outputs $Y^{\star}_{i,j}$ for $j \in \{1, \cdots, n_i\}$, and in the control variate expression we have used the fact that $\mathbb{E}[Y^{\star}_{i,n_i+1} - Y^{\star}_{i,j}] = 0$. This technique aims to minimize and control the estimation variance (Ross 2022). The optimal control variate coefficient $\alpha^* = \mathrm{Cov}(\hat{Z}_{i,j}, Y^{\star}_{i,n_i+1} - Y^{\star}_{i,j})/\mathrm{Var}(Y^{\star}_{i,n_i+1} - Y^{\star}_{i,j})$ can be estimated. We, hence, use the debiased outputs that exploit control variate remedies and denote them by $Y^{\star\mathrm{cd}}_{i,j}$. The ultimate estimated performance that will be used for each design then is computed as $\bar{\bar{Y}}^{\mathrm{cd}} := \sum_{j=1}^{n_i} \sum_{i=1}^{b} (bn_i)^{-1} Y^{\star\mathrm{cd}}_{i,j}$. The proposed control variate technique is a naive attempt to reduce the variance in this paper. More effective variance reduction procedures remain for future research. We also note that when more budget is dedicated for an input model, as will be explained in the next section, only the highest level bootstraps are increased and the budget for deeper level ones remains fixed.

3.2 Evaluation Step: Post-fitness

The evaluation process discussed above requires $n_0 \times (2b' + 1) + 1$ simulation runs to generate one variance reduced debiased estimated value for a given scenario i of design t at iteration k. Therefore, the total budget spent at iteration k is $m \times b \times (n_0 \times (2b' + 1) + 1)$. Furthermore, increasing the number of simulations $n_{k,t,i}$ from n_0 for scenario i of a design t increases the estimation accuracy and reduces the standard error. So the question of how much and at what rate to increase the $n_{k,t,i}$ becomes of interest.

At the beginning of the optimization, spending large computation budgets is unnecessary, as the algorithm is more focused on exploring and screening the feasible space. As the search goes on and the algorithm approaches the optimal region, it becomes critical to more accurately estimate each individual in the population and compare their performance with others to maintain the ability to distinguish the better designs with statistical guarantees. Say, we are in a later iteration of GA and want to increase the simulation budget. How to do this increase, that is, to what design and scenario, can follow the R-OCBA process we explained earlier. The objective of R-OCBA is to enhance the probability of accurate selection by prioritizing the worst-case scenario of the best design among alternatives. While this approach does not precisely align with the sorting process in each genetic algorithm iteration, it offers a more cautious means to integrate uncertainty and determine the crucial solution and bootstrap. Moreover, this allocation significantly influences the average output and serves as an approximation for distributing supplementary budget across diverse input models. The remaining question is *when* should this additional budget allocation be triggered? We suggest an adaptive rule for triggering this additional budget allocation in a stochastic GA. But for practical purposes, we limit the maximum computation budget used for each design and its all scenarios in each iteration to n, i.e., $\sum_{i=1}^{b} n_{k,t,i} \leq n$. Note that here we are excluding the effort needed for debiasing and variance reduction as it is assumed fixed for each simulation run.

Our adaptive rule balances the optimality gap versus the average estimation error in the population. Define the average estimation error in population k as

$$\hat{\sigma}_k := \sqrt{\frac{1}{m}\frac{1}{b}\sum_{t=1}^{m}\sum_{i=1}^{b}\frac{\hat{\sigma}^2_{k,t,i}}{n_{k,t,i}}}. \tag{5}$$

If $\hat{\sigma}_k$ is small relative to the optimality gap for a given iteration k, no additional computation budget is required. We measure the optimality gap of the proposed algorithm with

$$\Pi_k = \left| \frac{\bar{\bar{Y}}(X^*_k)}{\frac{1}{m}\sum_{t=1}^{m} \bar{\bar{Y}}_{k,t}} - 1 \right|, \tag{6}$$

where the numerator is the estimate of the expected performance of the best design up to iteration k, and the denominator is the average performance of all designs in the current iteration. Knowing that in GA approaching the optimal region means the designs in the population will have similar fitness, we track the relative distance of the fitness of X_k^* from the average fitness of all designs in iteration k. $\bar{\bar{Y}}(X_k^*)$ serves as the fitness of a proxy optimal design, and its accuracy and precision gradually improve as the search progresses. We expect Π_k to be small and closer to zero if the current population is near the optimal region and larger otherwise. The adaptive sampling rule here looks like for

$$\min\left\{(n_{k,t,i},\ t=1,\ldots,m, i=1,\ldots,b) : \left(\hat{\sigma}_k \leq \ell\bar{\bar{Y}}(X_k^*)\Pi_k\right) \cup \left(\sum_{i=1}^{b} n_{k,t,i} = n\ \forall t \in \{1,\cdots,m\}\right)\right\},$$

where ℓ is a constant governing our level of conservativeness. A larger ℓ makes the criteria easier to satisfy and could lead to little to no change to the budget allocation prior to post-fitness. The additional account for $\bar{\bar{Y}}(X_k^*)$ is to keep the scales of the right and left-hand side in the same order of magnitude.

```
**Algorithm 1** ARGA
**Given:** population size $m$, number of scenarios $b$, computing budget increment $\delta \geq 1$, adaptive sampling constant $\ell$, maximum allowed budget $n$ and minimum budget $b \times n_0$ ($n_0$ in every scenario) for each design.
**Initialize:** set $k = 1$, randomly select $m$ designs, and compute variance-reduced debiased estimated fitness of each design $\bar{\bar{Y}}_{1,t}^{\star\text{cd}}$ with $b \times n_0$ replications.
**for** *iteration* $k = 2, \cdots, K$ **do**
    Compute $\bar{\bar{Y}}_{k,t}^{\star\text{cd}}$ of each design with $n_0$ replications.
    Update $X_k^*$ with the best solution found up to iteration $k$, in terms of average fitness.
    Calculate the estimation error $\hat{\sigma}_k$ and optimality gap $\Pi_k$ using (5) and (6), respectively.
    **while** $\hat{\sigma}_k > \ell\bar{\bar{Y}}(X_k^*)\Pi_k$ *and* $\sum_{i=1}^{b} n_{k,t,i} + \delta \leq n\ \forall t \in \{1,\cdots,m\}$, **do**
        Add $\delta$ replications to the design and scenario that is identified by R-OCBA (Steps (a)-(d) Section 2.1).
        Update $\bar{Y}_{k,t,i}$ and $\hat{\sigma}_{k,t,i}$ for all $t = 1,2,\ldots,m$ and $i = 1,2,\ldots,b$.
        Update $\hat{\sigma}_k$ and $\Pi_k$.
    **end**
    Complete the selection, crossover, and mutation steps of the standard GA using the updated results.
    If the convergence criteria are met, terminate the search; otherwise set $k = k + 1$ and continue.
**end**
```

When $k > 1$, ARGA applies the post-fitness process once it calculates the debiased fitness of the designs using n_0 replications per design per scenario. If the average standard error $\hat{\sigma}_k$ is reasonably small compared to the optimality gap measure $\ell\bar{\bar{Y}}(X_k^*)\Pi_k$ and the maximum number of runs per design has not been exhausted, the R-OCBA algorithm assigns additional budget to a scenario of a design. Observing $\hat{\sigma}_k > \ell\bar{\bar{Y}}(X_k^*)\Pi_k$ signals one of two possibilities: either the population's average standard error is substantial, making the estimated quantities unstable and misleading, or the optimality gap is small, indicating more effort may be needed to distinguish better designs. In both cases, allocating more budget (with increments of $\delta \in \mathbb{Z}^+$) is advisable to help the progress in optimization. With the added budget, the optimality gap remains nearly constant since the population is not changed. At the same time, the average standard error decreases, ensuring that the loop will terminate (even without reaching the maximum budget). Importantly, the scaling of the optimality gap with $\bar{\bar{Y}}(X_k^*)$ tends to decrease as the optimization algorithm advances, making the adaptive sampling criteria stricter, highlighting the need to minimize the estimation error as much as possible. In the event X_k^* is excluded from the current population due to mutation, its quantity will not change with added budget but it continues to scale the distance to optimality of the current population.

All steps of ARGA are listed in Algorithm 1. The first iteration initiates with a simple population evaluation using minimum simulation effort n_0 and no post-fitness step to obtain a fast measurement of the population's optimality. As a final remark, we emphasize the pivotal role of debiasing prior to increasing the precision. Ideally one would opt for debiasing the designs after choosing the right budget for them (and each of their scenarios). However, debiasing is expensive yet less sensitive to small changes in the budget than the standard error. We leave further investigation on this point to future research.

4 NUMERICAL RESULTS

This section presents the simulated experiments designed to evaluate the performance of the proposed algorithm compared to other benchmarks. The investigations focus on the feature selection problem, a challenging optimization task in both ML and simulation domains (George 2000). Feature selection refers to identifying the most relevant variables that can explain the response effectively. Vahdat and Shashaani (2020) formulated and approached feature selection as a simulation optimization problem.

The selection of features can be a complex problem, particularly because more features in a dataset lead to exponential increase in complexity. To broadly evaluate the performance of competing GAs, we conduct tests on two groups of datasets, namely "small" and "large" datasets. All datasets contain a continuous response variable that we seek to predict, with all features following a normal distribution with varying variances. We use simple linear regression to estimate the response. Each dataset contains a small number of *true features* that contribute to the response. Since the datasets are synthesized, we know which features truly contributed to the response. We compute the ratio of correctly selected features to all true features, or the true positive rate (TPR), for each algorithm as a metric for comparing algorithms.

Identifying the contributing features can become more challenging when there is a correlation between the features. In such cases, the features selection algorithm may mistakenly select correlated variables, further increasing the complexity of the problem. We test the proposed approach with data sets that also contain correlation between features. Table 1 provides detailed information on the characteristics of each dataset. The remaining parameters of ARGA for these experiments are set to $\ell = 0.5$, $\delta = 2$, $n_0 = 5$, $b' = 5$, and $n = 100$. The number of scenarios evaluated for each method is fixed to $b = 10$. The population size m is fixed to the same number of data columns being tested. The input models are generated with bootstrapping the data. The GA search parameters, such as mutation and crossover probability, are fixed among all methods and set to 0.3 and 0.8, respectively. Stopping criteria in a GA affect its convergence, impacting speed, solution quality, and robustness. In our study, GA terminates after 100 iterations without improvement.

Table 1: Description of four synthetic datasets used in the numerical experiments.

Dataset label	Correlated features?	# Columns	# Rows	# True features
SDF	No	15	300	5
SDF-C	Yes	15	300	5
LDF	No	30	300	10
LDF-C	Yes	30	300	10

In Table 2, we evaluate the performance of three different GA methods for feature selection in ML models. The methods compared are standard GA (Shashaani and Vahdat 2022), GA with debiased estimators for model output (Vahdat and Shashaani 2023), i.e., Robust GA (RGA), and ARGA. The evaluation is based on TPR (hoping to be near 1) and the number of selected features (hoping to be near the number of true features). Keeping the total budget constant between these three methods means that GA and RGA use a fixed number of calls in each iteration and will terminate at likely a larger iteration than ARGA, which has a varying number of calls in each iteration. The adaptive number of calls in ARGA results in much fewer simulation runs at the beginning of the search and more extensive searches towards the end. Despite generating fewer populations (GA iterations) ARGA provides better designs. The results also indicate that ARGA outperforms the other methods in retrieving the correct variables while maintaining a high TPR. The computation time for each method, including simulation effort and arithmetic calculations, is also insightful; ARGA indeed relieves some of the added computation for bias calculation by adaptively growing the budget. This relief in computation is evident in the larger datasets with roughly 30% and 25% reduction in the total time in LDF and LDF-C respectively.

Table 2: Three GA methods are test on four synthesized datasets with all metrics over 10 macro-replications; average and standard error for each case are summarized here. In all datasets, ARGA outperforms the other methods in terms of TPR, but more specifically, in the largers datasets (LDF and LDF-C) it shows better performance in finding the right number of features (10) with less time.

Dataset	Method	# Features	TPR	Time (min)
SDF	GA	4.50 ± 0.37	0.66 ± 0.04	2.53 ± 0.31
	RGA	3.80 ± 0.24	0.70 ± 0.04	3.96 ± 0.36
	ARGA	4.40 ± 0.33	**0.82** ± 0.06	4.54 ± 0.86
SDF-C	GA	4.20 ± 0.25	0.64 ± 0.04	2.81 ± 0.19
	RGA	3.80 ± 0.13	0.64 ± 0.04	5.20 ± 0.47
	ARGA	4.00 ± 0.39	**0.74** ± 0.06	5.98 ± 0.93
LDF	GA	7.60 ± 0.31	0.67 ± 0.03	5.49 ± 0.42
	RGA	6.50 ± 0.37	0.61 ± 0.03	8.43 ± 0.52
	ARGA	9.40 ± 0.52	**0.87** ± 0.03	5.93 ± 0.62
LDF-C	GA	7.50 ± 0.31	0.63 ± 0.02	5.02 ± 0.28
	RGA	6.60 ± 0.16	0.61 ± 0.02	10.74 ± 0.54
	ARGA	9.60 ± 0.72	**0.77** ± 0.04	8.06 ± 0.92

Since the simulation effort can be the major burden for feature selection as the dataset grows, we also compare the progress at intermediate simulation budgets spent before termination. Figure 1 depicts the efficiency of ARGA in spending the simulation budget compared to others. We observe that when the budget spent is small at the beginning of the search, the debiased GA has the best performance as it allocates the computation budget to IU bias estimation, thereby enhancing its estimates. In contrast, ARGA outperforms both competitors after spending about half of the budget. It efficiently spends less simulation budget in the initial iterations, and that saved budget helps find better solutions later in the search.

5 CONCLUDING REMARKS

GA is a class of evolutionary solvers widely used in practice. Their volatility makes them one of the few viable choices for complex optimization problems, such as in high-dimensional binary search of functions contaminated by stochastic noise. A significant risk that can misguide the search is IU bias. We propose ARGA, a novel variant of the GA, aiming to accurately evaluate and select the best solution while robustly reducing the computational cost in stochastic optimization. ARGA is equipped with (i) a debiased estimator that uses a variance-reduced fast-iterated bootstrapping method to compute and reduce the IU bias and (ii) an adaptive R-OCBA rule that balances the estimation error and optimality gap at each GA iteration, allocating computation effort to the input models contributing to the variability of the most critical solutions. In principle, the new methodology allows for a middle-ground between the fixed-precision and fixed-budget R&S within an optimization algorithm such as GA. The fundamental unification of finding that middle-ground in R&S-type procedures will be insightful and generalizable for optimization beyond the context of a binary search engine such as GA.

ARGA has the potential to reduce computation costs significantly while still achieving similar or better solutions. This is because its adaptive R-OCBA in each iteration guarantees a predetermined probability of correct selection even without exhausting the predetermined budget. Accurate design comparisons improve GA's exploration in the search space. Our empirical results demonstrate enhanced effectiveness in ARGA applied to a data-driven optimization problem, namely, feature selection, in varying dimensions.

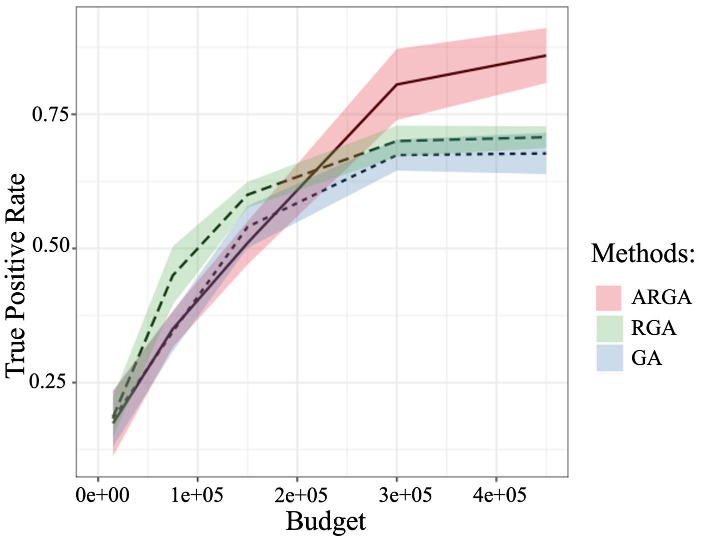

Figure 1: Confidence intervals, computed over 10 macro-replications, compare TPR values for the SDF dataset at intermediate budgets. RGA yields an improvement from the original GA (where no IU information is utilized), but ARGA significantly improves the feature selection.

ACKNOWLEDGMENTS

This work was partially supported by the American Association of University Women (AAUW) Research Publication Grant in Engineering, Medicine and Science, and American Educational Research Association.

REFERENCES

Azadivar, F., and G. Tompkins. 1999. "Simulation Optimization with Qualitative Variables and Structural Model Changes: A Genetic Algorithm Approach". *European Journal of Operational Research* 113(1):169–182.

Bechhofer, R. G. 1995. *Design and Analysis of Experiment for Statistical Selection, Screening, and Multiple Comparisons*. Number 04; QA279, B4.

Bhandari, D., C. Murthy, and S. K. Pal. 1996. "Genetic Algorithm with Elitist Model and its Convergence". *International journal of pattern recognition and artificial intelligence* 10(06):731–747.

Boesel, J., and B. L. Nelson. 1998. "Accounting for Randomness in Heuristic Simulation Optimization". In *Proceedings of the 12th European Simulation Multiconference on Simulation - Past, Present and Future*, 634–638: SCS Europe.

Chang, J., and P. Hall. 2015. "Double-bootstrap Methods that Use a Single Double-bootstrap Simulation". *Biometrika* 102(1):203–214.

Corlu, C. G., A. Akcay, and W. Xie. 2020. "Stochastic Simulation under Input Uncertainty: A Review". *Operations Research Perspectives* 7:100162.

Corlu, C. G., and B. Biller. 2015. "Subset Selection for Simulations Accounting for Input Uncertainty". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, 437–446. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Efron, B. 1979. "Bootstrap Methods: Another Look at the Jackknife". *The Annals of Statistics* 7(1):1–26.

Fan, W., L. J. Hong, and X. Zhang. 2020. "Distributionally Robust Selection of the Best". *Management Science* 66(1):190–208.

Gao, S., H. Xiao, E. Zhou, and W. Chen. 2017. "Robust Ranking and Selection with Optimal Computing Budget Allocation". *Automatica* 81:30 – 36.

George, E. I. 2000. "The Variable Selection Problem". *Journal of the American Statistical Association* 95(452):1304–1308.

Ghaoui, L. E., M. Oks, and F. Oustry. 2003. "Worst-case Value-at-Risk and Robust Portfolio Optimization: A Conic Programming Approach". *Operations Research* 51(4):543–556.

Gupta, S. S. 1965. "On Some Multiple Decision (Selection and Ranking) Rules". *Technometrics* 7(2):225–245.

Holland, J. H. 1992. "Genetic Algorithms". *Scientific American* 267(1):66–73.

Hong, L. J., W. Fan, and J. Luo. 2021. “Review on Ranking and Selection: A New Perspective”. *Frontiers of Engineering Management* 8(3):321–343.

Hunter, S. R., and B. L. Nelson. 2017. “Parallel Ranking and Selection”. In *Advances in Modeling and Simulation*, 249–275. Springer.

Kou, G., H. Xiao, M. Cao, and L. H. Lee. 2021. “Optimal Computing Budget Allocation for the Vector Evaluated Genetic Algorithm in Multi-objective Simulation Optimization”. *Automatica* 129:109599.

Lam, H. 2016. “Advanced Tutorial: Input Uncertainty and Robust Analysis in Stochastic Simulation”. In *Proceedings of the 2016 Winter Simulation Conference*, 178–192. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Liu, M., and A. M. Cramer. 2018. “Computing Budget Allocation in Multi-objective Evolutionary Algorithms for Stochastic Problems”. *Swarm and Evolutionary Computation* 38:267–274.

Miller, B. L., and D. E. Goldberg. 1995. “Genetic Algorithms, Tournament Selection, and the Effects of Noise”. *Complex Systems* 9(3):193–212.

Mitchell, M. 1998. *An Introduction to Genetic Algorithms*. The MIT Press.

Nazzal, D., M. Mollaghasemi, H. Hedlund, and A. Bozorgi. 2012. “Using Genetic Algorithms and an Indifference-zone Ranking and Selection Procedure under Common Random Numbers for Simulation Optimisation”. *Journal of Simulation* 6(1):56–66.

Pasupathy, R., and S. Ghosh. 2013. *Simulation Optimization: A Concise Overview and Implementation Guide*, Chapter 7, 122–150. INFORMS TutORials in Operations Research.

Ross, S. M. 2022. *Simulation*. Academic Press.

Schmitt, L. M. 2001. “Theory of Genetic Algorithms”. *Theoretical Computer Science* 259(1-2):1–61.

Shashaani, S., and K. Vahdat. 2022. “Improved Feature Selection with Simulation Optimization”. *Optimization and Engineering*:1573–2924.

Song, E., and B. L. Nelson. 2019. “Input–Output Uncertainty Comparisons for Discrete Optimization via Simulation”. *Operations Research* 67(2):562–576.

Song, E., B. L. Nelson, and L. J. Hong. 2015. “Input Uncertainty and Indifference-zone Ranking and Selection”. In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, 414–424. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Vahdat, K., and S. Shashaani. 2020. “Simulation Optimization Based Feature Selection, A Study on Data-driven Optimization with Input Uncertainty”. In *Proceedings of the 2020 Winter Simulation Conference*, edited by K.-H. G. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, 2149–2160. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Vahdat, K., and S. Shashaani. 2021. “Non-parametric Uncertainty Bias and Variance Estimation via Nested Bootstrapping and Influence Functions”. In *Proceedings of the 2021 Winter Simulation Conference*, edited by S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, 1–12. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Vahdat, K., and S. Shashaani. 2023. “Robust Prediction Error Estimation with Monte Carlo Methodology”. *arXiv preprint arXiv:2207.13612*.

Wu, D., and E. Zhou. 2017. “Ranking and Selection under Input Uncertainty: A Budget Allocation Formulation”. In *Proceedings of the 2017 Winter Simulation Conference*, edited by V. W. Chan, A. D’Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. H. Page, 2245–2256. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Xiao, H., and L. H. Lee. 2014. “Simulation Optimization Using Genetic Algorithms with Optimal Computing Budget Allocation”. *Simulation: Transactions of the Society for Modeling and Simulation International* 90(10):1146–1157.

Zhang, X., and L. Ding. 2016. “Sequential Sampling for Bayesian Robust Ranking and Selection”. In *Proceedings of the 2016 Winter Simulation Conference*, edited by T. M. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, 758–769. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

KIMIA VAHDAT is a fifth-year Ph.D. candidate at Edward P. Fitts Department of Industrial and Systems Engineering at North Carolina State University. Her research is focused on applications of stochastic simulation in machine learning and data science. Her email is kvahdat@ncsu.edu.

SARA SHASHAANI is an Assistant Professor in the Edward P. Fitts Department of Industrial and System Engineering at North Carolina State University. Her research interests are probabilistic data-driven models and simulation optimization. She is a co-creator of SimOpt. Her email address is sshasha2@ncsu.edu and her homepage is https://shashaani.wordpress.ncsu.edu/.