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

NEURAL NETWORK-ASSISTED SIMULATION OPTIMIZATION WITH COVARIATES

Haoting Zhang

Department of Industrial Engineering
and Operations Research
University of California, Berkeley
Berkeley, CA 94720, USA

Jinghai He

Antai College of Economics and Management
Shanghai Jiao Tong University
No. 535, Fahua Town Road
Shanghai, 200030, CHINA

Donglin Zhan

Department of Electrical Engineering
Columbia University
500 West 120th Street, Mudd 1310
New York, NY 10027, USA

Zeyu Zheng

Department of Industrial Engineering
and Operations Research
University of California, Berkeley
Berkeley, CA 94720, USA

ABSTRACT

In real-time decision-making problems for complicated stochastic systems, a covariate that reflects the state of the system is observed in real time and a state-dependent decision needs to be made immediately to optimize some system performance. Such system performances, for complicated stochastic systems, often are not in closed-form and require time-consuming simulation experiments to evaluate, which can be prohibitive in real-time tasks. We propose two neural network-assisted methods to address this challenge by effectively utilizing simulation experiments that are conducted offline before the real-time tasks. One key step in the proposed methods integrates a classical simulation meta-modelling approach with neural networks to jointly capture the mapping from the covariate and the decision variable to the system performance, which enhances the use of offline simulation data and reduces the risk of model misspecification. A brief numerical experiment is presented to illustrate the performance of the proposed methods.

1 INTRODUCTION

Modern stochastic systems often involve a sophisticated structure. Generally, the system performance is not an analytical function of the decision variables, but a complex surface that can only be evaluated at points through noisy samples. Simulation experiments have been widely used to generate these noisy samples, which are used to evaluate the performance of a complex system. These simulation experiments can be time-consuming for sophisticated systems. The system performance typically depends on a set of decision variables, which serve as input parameters to the simulation experiments. In many applications, it is of interest to search the feasible set of the decision variables with the objective to optimize the associated system performance. This need drives the area of *simulation optimization*, also known as optimization via simulation; see Fu et al. (2015), Amaran et al. (2016) for reference. The strategies for solving the simulation optimization problems depend largely on the features of the objective function and the feasible set. If the feasible set is discrete, the methodologies utilized can be found in the broad literature of discrete optimization via simulation; see Andradóttir (2006), Ólafsson (2006), Kim and Nelson (2007), Nelson (2010), Hong et al. (2015), Hong et al. (2020) among others. When the feasible set is continuous, under

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

different circumstances, various methods are developed, including but are not limited to gradient-based methodologies (Fu 2006; Fu 2010) and meta-model based methods (Barton and Meckesheimer 2006; Barton 2013). Our work positions in the domain where the feasible set is continuous and the simulation experiments are time-consuming to run.

In some scenarios, the objective functions and simulation experiments are involved with some additional parameters different from the decision variables. These additional parameters, for example, include the biometric characteristics for cancer treatment, the purchasing preference for marketing, and the ambient information for the self-driving car. These parameters are denoted as the *covariate*, to be distinguished from the decision variables (or, design points, in some literature). The optimal choice of decision variables typically depends on the covariates, leading to the problems of *simulation optimization with covariates*. When it is time-consuming to conduct simulation experiments, it is difficult for the conventional simulation optimization strategies to be directly adopted into the framework for solving the covariate-involved problems. This is especially the case when the problems are required to be solved in real time and a new covariate value is observed that has never been seen before. There may not be enough time to run enough simulation experiments associated with this new covariate in real time. This motivates the need to use simulation experiments that have been run offline to support the real-time choice of optimal decision variable upon observing the new covariate, without running new simulation experiments; see also Hong and Jiang (2019), Gao et al. (2019) for the description of this need. Our work aims at providing new frameworks to facilitate this need. Discrete simulation optimization problems with the covariate have been explored in Shen et al. (2021), while we focus on the continuous simulation optimization problems in this work.

We provide two neural network-assisted frameworks to utilize offline simulation experiments to solve the aforementioned real-time covariate-involved optimization problems. In the first framework, offline simulations are run at selected values of the covariate and various decision variables. For each selected value of covariate, the optimal decision variable is computed offline. Then a neural network is trained through the pairs of covariates and the associated optimal decisions. Therefore, at the real-time stage, when a new value of the covariate is observed, the approximate optimal decision is directly referred through the trained neural network. In the second framework, we approximate the surface of the objective function with respect to the covariate, using the neural network. We incorporate the neural network with the classical meta-model, stochastic kriging model to approximate the performance of the simulation model. To be more specific, given a covariate, the performance of the simulation model is estimated through a traditional stochastic kriging model, while the parameters involved in the stochastic kriging model are approximated with neural networks with respect to the covariate. The training of the neural networks is based on maximizing the likelihood functions of the stochastic kriging models. With the trained neural networks in hand, upon the new covariate is revealed, the approximate objective function is obtained without the need of conducting simulation experiments. Furthermore, the objective function derived from the stochastic kriging model can be optimized efficiently. In this way, the optimization problem is solved in real time.

Besides the neural networks, other machine learning models, e.g. regression models, can be adopted into our methodologies as well. The main reason why neural networks are used in these frameworks is largely due to the ambiguous surfaces with respect to the covariate. We might not have the prior information of the mapping from the covariate. Neural networks preset no structural assumptions on the statistical model, therefore are less risky to suffer the model misspecification. In addition, neural networks are capable of approximating arbitrary functions, provided sufficient nodes and layers. Therefore, the utilization of neural network helps to solve the aforementioned problems accurately. An introduction to neural networks can be found in Masson and Wang (1990).

The advantages of our frameworks are two-fold. First, at the real-time stage, the pre-trained neural network is used to recommend an optimal decision. This not only satisfies the realistic limitation of not having enough time to run additional simulations, but also eliminates the need of temporarily calling all the samples generated offline or executing time-consuming calculations such as high-dimensional matrix inversion. Second, the neural-network assisted approaches have the potential to learn and model non-linear

and complex relationships of the response surface with respect to the covariate, without imposing the specific structual assumptions on the model. Therefore, the accuracy of prediction and robustness against model-misspecification can be improved through the use of neural networks.

Our work is connected to and benefits from the literature of the area of meta-modelling. Meta-models, or models of simulation models are used to approximate the mean performance of the simulation models (Barton 2015). These models adopt the advantage of faster execution, and provide insight on the nature of the model performance. In our work, we utilize the stochastic kriging meta-model to approximate the response surface of the simulation model; see Ankenman et al. (2010) for reference. The stochastic kriging model assumes that the performance of the simulation model is a sample path of a Gaussian process. Compared with the ordinary kriging method, the stochastic kriging method is able to capture both the intrinsic and extrinsic uncertainty in the design. As a nonparametric Bayesian model, the stochastic kriging method imposes less structural assumptions and tends to be more resistant to overfitting than general interpolators. The utilization of the common random numbers and gradient estimators help to enhance the performance of the stochastic kriging; see Chen et al. (2012), Chen et al. (2013), Qu and Fu (2014). The stochastic kriging method has been used in the input uncertainty analysis of the simulation model (Barton et al. 2014; Xie et al. 2014; Xie et al. 2020). Meanwhile, the stochastic kriging-based optimization has been widely used in the simulation optimization field as well, see Sun et al. (2014), Jalali et al. (2017), L. Salemi et al. (2019), Semelhago et al. (2020).

2 METHODOLOGY

Simulation models provide insight into the behavior of stochastic systems, which help to improve the system performance by changing the input parameters. The strategy of using the simulation experiments to facilitate the optimization procedure is denoted as the simulation optimization, formulated as

$$x^* = \arg\min_{x \in \mathscr{X} \subseteq \mathbb{R}^d} \left\{ \mathbb{E}_{\xi \sim P} \left[F(x, \xi) \right] = f(x) \right\},$$

where $x \in \mathscr{X} \subset \mathbb{R}^d$ denotes the controllable design point of the simulation model and serves as the decision variable of the optimization problem as well. In our work, we focus on the continuous optimization problem. $\xi \sim P$ captures the random effects of the system. The objective function $f(x)$ is named as the response surface function of the system. Simulation experiments are conducted to evaluate the response surface and therefore, facilitate the optimization problem. However, it is often the case that simulation experiments are computationally intensive to execute. It is costly to collect sufficient simulation samples for the optimization problem. Therefore, statistical models are often postulated for the response surface with respect to the design points, which are convenient to run and evaluate. The statistical model is named as the *meta-model* in the simulation community.

In some application scenarios, the stochastic systems involve the additional parameters that are observed from the environment. These parameters are named as covariates, and the optimization problem is then reformulated as

$$x^*(\theta) = \arg\min_{x \in \mathscr{X} \subseteq \mathbb{R}^d} \left\{ \mathbb{E}_{\xi \sim P_\theta} \left[F(x, \xi; \theta) \right] = f(x; \theta) \right\}. \tag{1}$$

Here, the covariate $\theta \in \Theta \subseteq \mathbb{R}^q$ influences both the distribution of the random effects P and the form of the function F. The conventional strategy is executing simulation experiments to solve the problem (1) once the covariate is observed. However, this approach might be impractical when the optimal solution should be attained in real-time after a new covariate is revealed. Simultaneously, we are not able to obtain the optimal solution in advance for every potential covariate θ as well, considering the expensive cost of the simulation experiments.

To circumvent the challenge, we propose to utilize the neural network to facilitate the optimization problem (1). We aim at providing the optimal solution $x^*(\theta)$ without executing the simulation experiments at the real-time stage. Therefore, we need to implement simulation experiments in advance. With the data

generated by the simulation models and the data analytics tools, we then build predictive models for solving the problem (1). In our work, the predictive models are neural network-assisted. One brute-force strategy is utilizing the neural network to approximate the optimal solution surface with respect to the covariate, which is named as the *neural network-assisted optimal solution surface* method. We present the complete procedure of this method in Section 2.2. The other methodology embeds the neural network into the existing meta-model, exploiting more structural information of the simulation model. To be more specific, we propose a neural network-assisted stochastic kriging model to approximate the surface of the objective function with respect to the covariate, We name it as the *neural network-assisted objective function surface* method. The details of the training procedure and the optimization procedure are included in Section 2.3.

2.1 Notation of Neural Network

Before we propose our methodologies, we first present the notations of the neural networks we used in our work. Neural networks are computing systems that are used to approximate unknown mappings. In our work, we utilize the fully-connected artificial neural network. For simplicity, we are going to use the term neural network for the remaining parts. A neural network is composed with connected layers of nodes, which are denoted as *neurons*. For each pair of the adjacent layers, the input of the t-th layer is the output of the $(t-1)$-th layer. Assume the t-th layer possesses m_t neurons, which indicates the dimension of the t-th input. (For example, the number of neurons of the first layer (the input layer) is the same as the dimension of the input.) Therefore, suppose the neural network has totally τ layers, the number of neurons of the last layer m_τ is the same with the dimension of the real observations. With the input and the output layers excluded, the remaining layers are denoted as *hidden layers*, which are used to approximate the complex mapping between the input and the output. Traditionally, the neural network is trained through minimizing the loss between the output and the real observations. Other criterion can be adopted to obtain the designated neural network depending on the application scenarios.

To be more specific, suppose the p-dimensional input of the neural network is denoted as

$$\theta_n = \left(\theta_n^{(1)}(1), \theta_n^{(1)}(2), \ldots, \theta_n^{(1)}(p)\right), \quad n = 1,2,\ldots,N,$$

where the superscipt "(1)" indicates the layer of the neural network. For the n-th input, the i-th element of the associated output of the t-th layer is given by

$$\theta_n^{(t+1)}(i) := \varphi\left(v_n^{(t)}(i)\right), \quad v_n^{(t)}(i) := \sum_{j=0}^{m_t} w_{i,j}^{(t)} \theta_n^{(t)}(j), \quad i = 1,\ldots,m_{t+1}, \tag{2}$$

where $\varphi(\cdot)$ is the *activation function*. Meanwhile, it is required that $\theta_n^{(t)}(0) = 1$, representing the constant term. The synaptic weights $w_{i,j}^{(t)}$ are the parameters to be trained. Recall that the neural network has totally τ layers. Therefore, the output vector of the n-th input is $\vec{\theta}_n^{(\tau)} := \left(\theta_n^{(\tau)}(1), \ldots, \theta_n^{(\tau)}(m_\tau)\right)$, while the associated n-th real observation is $\vec{O}_n := (o_n(1), \ldots, o_n(m_\tau))$. Suppose the loss function selected is denoted as L, in order to train the neural network, we aim at minimizing the mean loss

$$\min_w \left\{ \mathscr{L}(w) = \frac{1}{N} \sum_{n=1}^{N} L\left(\vec{\theta}_n^{(\tau)}, \vec{O}_n\right) \right\}, \tag{3}$$

where w is a vector containing all synaptic weights. We select the l_2-norm loss in our work. To solve the optimization problem, the traditional strategy is applying the back-propagation algorithm, along with certain optimization methods. After the training procedure, the neural network is then represented as

$$\phi_{\tau-1}\left(\phi_{\tau-2} \ldots \phi_1(\theta)\right), \tag{4}$$

where ϕ_t denotes the mapping described in (2) from the t-th layer input to the t-th layer output, with the optimal synaptic weights obtained via (3).

2.2 Method I: Neural Network-Assisted Optimal Solution Surface

The first methodology is building the surface of the optimal solution with respect to the covariates. We first select $\theta_1, \theta_2, \ldots, \theta_N \in \Theta$. Then, with each θ_n, simulation experiments are then conducted to find $x^*(\theta_n) \in \mathscr{X}$. For a fixed covariate, the problem reduces to a regular simulation optimization problem. Therefore, our methodology does not require any specific optimization strategies. The selection of the optimization method is supposed to depend on the problem itself. References for solving simulation optimization problems include Fu et al. (2015), Amaran et al. (2016).

Assume that we implement sufficient simulation experiments to obtain the optimal solution for each covariate θ_n. Here, we ignore the bias resulted from the optimization algorithms and the estimation error from the simulation model. At the end of the offline stage, we have the pairs of $\{(\theta_1, x^*(\theta_1)), (\theta_2, x^*(\theta_2)), \ldots, (\theta_N, x^*(\theta_N))\}$, where $x^*(\theta_n)$ denotes the optimal solution under the covariate θ_n. Using these pairs of data, we are then able to approximate the surface of the optimal decision. Since the sophisticated intrinsic mechanism between the covariate and the optimal decision is unknown, we utilize the neural network to construct the surface of the optimal solution with respect to the covariates. The input of the neural network is $\{\theta_1, \theta_2, \ldots, \theta_N\}$. We denote the output as $\{\hat{x}^*(\theta_1), \hat{x}^*(\theta_2), \ldots, \hat{x}^*(\theta_N)\}$. The real observations are $\{x^*(\theta_1), x^*(\theta_2), \ldots, x^*(\theta_N)\}$. By minimizing the loss $\mathscr{L}(w) = \frac{1}{N}\sum_{n=1}^{N} L(\hat{x}^*(\theta_n), x^*(\theta_n))$, we obtain the approximate optimal decision surface, as is described in (4)

$$\hat{x}^*(\theta) = \phi_{\tau-1}(\phi_{\tau-2} \ldots \phi_1(\theta)), \tag{5}$$

which is a mapping from the covariate space to the decision space. In terms of the neural network, our methodology does not require any specific types of the activation function or the loss function.

In this way, at the real-time stage, every time we observe a θ, we could immediately refer to the approximate optimal solution (5). An obvious advantage of this methodology is that the approximate optimal solution is attained via directly referring. Therefore, this methodology is suitable for the problems that require rapid action in different environments, such as the self-driving cars (Katrakazas et al. 2015). An illustrative example and more details for experiments are presented in Section 3. Note that, in general, the neural network approximation does not preserve the feasibility of the mapping from the covariate space to the decision space at all times, unless the simplicity of the feasible region is assumed. Therefore, one drawback of this methodology is the lack of the theoretic support.

2.3 Method II: Neural Network-Assisted Objective Function Surface

In stead of directly approximating the surface of the optimal solution with respect to the covariate, we also propose to build the meta-model of the objective function (response surface of the simulation model) regarding both the decision variable x and the covariate θ. Here, we propose a neural network-assisted stochastic kriging model to fit the response surface of the simulation experiments. Thus, given a fixed covariate, the meta-model reduces to a traditional stochastic kriging model. Meanwhile, the parameters involved in the stochastic kriging are modelled as mappings of the covariate. These unknown mappings are approximated through the neural networks. To be more specific, given the covariate, the output of the simulation model with the input x is represented as

$$Y_m(x;\theta) = \beta(\theta)^\top \mathbf{g}(x) + M(x;\theta) + \varepsilon_m(x;\theta). \tag{6}$$

Here, $\mathbf{g}(x)$ is a vector, of which each element is a preset function. $\beta(\theta)$ denotes the parameter vector, and each element is an unknown function of θ. $M(x;\theta)$ is a mean-zero, second-order stationary Gaussian process, accounting for the spatial dependence of the response surface. Here, the covariate θ controls the covariance of the Gaussian process, and the detailed form will be presented in the following part. $\varepsilon_m(x;\theta)$ denotes the intrinsic uncertainty of the simulation experiments, which are i.i.d. samples drawn from a mean-zero distribution. The covariate θ and the design point x both influence the variance of the intrinsic uncertainty. Thus, $\mathrm{Var}[\varepsilon_m(x;\theta)] = V(x;\theta)$.

In particular, $M(x;\theta)$ is assumed to be a mean-zero, second-order stationary Gaussian process, thus the covariance matrix of two arbitrary points x and x' is represented as

$$\Sigma_{M;\theta}\left(x,x'\right)=\tau^2(\theta)R_M\left(x-x';\gamma(\theta)\right). \tag{7}$$

Here, $\tau^2(\theta)$ denotes the variance of the Gaussian process under the covariate θ. $R_M\left(x-x';\gamma(\theta)\right)$ is the correlation matrix, where $\gamma(\theta)$ is another parameter involved, indicating the decaying of the correlation between design points. It is required that $R_M\left(0;\gamma(\theta)\right)=1$ and $R_M\left(\infty;\gamma(\theta)\right)=0$. Common choices for the covariance matrix include the Gaussian kernel and the Matérn kernel. Although our methodology does not require a certain type of the correlation, here we utilize the Gaussian kernel correlation to illustrate our methodology, which is in the form of

$$R_M\left(x-x';\gamma(\theta)\right)=e^{-\gamma(\theta)\|x-x'\|^2}. \tag{8}$$

The reason for representing the response surface with the meta-model described in (6) is two-fold. First, the predictor of the stochastic kriging is a linear combination of the given basis functions and correlation functions, as is shown in (10). This predictor serves as the objective function at the real-time stage, and can be optimized efficiently. In addition, the stochastic kriging model leads to a sequential, adaptive design of the simulation experiments that jointly considers the placement of design points and simulation efforts. This sequential design helps fit the response surface more efficiently with the limited simulation resources. Second, we propose to use the neural network to fit the parameters involved in the stochastic kriging because the neural network possesses a high accuracy of fitting, along with a low risk of model misspecification. Meanwhile, the utilization of the neural network helps to deal with the high-dimensional covariates and brings a computational efficiency.

2.3.1 Model Training

In this part, we focus on the training procedure of the model (6) with the simulated samples. First, we select N covariates $\{\theta_1,\theta_2,\ldots,\theta_N\}\in\Theta$. We assume that for different covariates, the design points remain the same, as well as the simulation resources. Thus, we select K points of the decision variable $\{x_1,x_2,\ldots,x_K\}$. For each x_k, we perform M_k times simulation experiments to estimate $f(x_k;\theta)$ with the obtained $Y_m(x_k), m=1,2,\ldots.M_k$. Therefore, at the end of the offline simulation stage, we have totally $N\left(\sum_{k=1}^K M_k\right)$ pairs of (θ_n,x_k,Y_m). In addition, we let

$$\bar{Y}(\theta)=\left(\bar{Y}(x_1;\theta),\bar{Y}(x_2;\theta),\ldots,\bar{Y}(x_K;\theta)\right)^\top, \tag{9}$$

where $\bar{Y}(x_k;\theta)=\frac{1}{M_k}\sum_{m=1}^{M_k}Y_m(x_k;\theta)$ denotes the sample mean of the output at (θ,x_k), as shown in (6). We denote $\Sigma_{M;\theta}$ as the covariance matrix of the design points $x_k, k=1,2,\ldots,K$ and $\Sigma_{\varepsilon;\theta}$ as the covariance matrix implied by the intrinsic noise, with (k,k')-th element $\mathrm{Cov}\left[\sum_{m=1}^{M_k}\varepsilon_m\left(x_k;\theta\right)/M_k,\sum_{m=1}^{M_{k'}}\varepsilon_m\left(x_{k'};\theta\right)/M_{k'}\right]$. Assume that we exactly know $\beta(\theta),\Sigma_{M;\theta}$ and $\Sigma_{\varepsilon;\theta}$, the predictor of the response surface at an arbitrary x_0 for the observed θ is

$$\hat{f}\left(x_0;\theta\right)=\beta(\theta)^\top\mathbf{g}(x_0)+\Sigma_{M;\theta}\left(x_0,\cdot\right)^\top\left[\Sigma_{M;\theta}+\Sigma_{\varepsilon;\theta}\right]^{-1}\left(\bar{Y}(\theta)-\mathbf{G}\beta(\theta)\right), \tag{10}$$

where $\Sigma_{M;\theta}\left(x_0,\cdot\right)$ is the vector of the covariance of x_0 with all K design points, defined in (7), and $\mathbf{G}=\left(\mathbf{g}(x_1)^\top,\mathbf{g}(x_2)^\top,\ldots,\mathbf{g}(x_K)^\top\right)^\top$. This predictor possesses an explicit form with respect to x, and therefore can be optimized in real time. Recall that the covariance matrix $\Sigma_{M;\theta}$ is determined by $\tau^2(\theta)$ and $\gamma(\theta)$. Therefore, before the real-time stage, we are supposed to have the estimators of $\beta(\theta)$, $\tau^2(\theta)$, $\gamma(\theta)$ and $\Sigma_{\varepsilon;\theta}$, $\bar{Y}(\theta)$.

In terms of the estimation for $\beta(\theta)$, $\tau^2(\theta)$ and $\gamma(\theta)$, we utilize the Maximum Likelihood Estimation (MLE) method. Before we propose the estimation procedure of the neural network-assisted stochastic kriging model, we first present the existing results of the traditional stochastic kriging model, where the covariate θ vanishes (Ankenman et al. 2010). For an experiment design $(x_k, M_k), k = 1, 2, \ldots, K$, the log-likelihood function of β, τ^2 and γ is

$$\ell\left(\beta, \tau^2, \gamma\right) = -\ln\left[(2\pi)^{K/2}\right] - \frac{1}{2}\ln\left[\left|\tau^2 R_M(\gamma) + \Sigma_\varepsilon\right|\right] - \frac{1}{2}\left(\bar{Y} - \mathbf{G}\beta\right)^\top \left[\tau^2 R_M(\gamma) + \Sigma_\varepsilon\right]^{-1}\left(\bar{Y} - \mathbf{G}\beta\right). \quad (11)$$

Recall that in our neural network-assisted stochastic model, the mappings $\beta(\theta)$, $\tau^2(\theta)$ and $\gamma(\theta)$ are all approximated by the neural networks. Therefore, by substituting β, τ^2 and γ with $\beta(\theta)$, $\tau^2(\theta)$ and $\gamma(\theta)$ in (11) and maximizing the log-likelihood functions, we attain the approximated neural network-assisted stochastic kriging model. Thus, we train the neural networks $\left(\hat{\beta}(\theta), \hat{\tau}^2(\theta), \hat{\gamma}(\theta)\right)$ via solving the optimization problem

$$\max_{w_\beta, w_{\tau^2}, w_\gamma} \frac{1}{N}\sum_{n=1}^{N} \ell_n\left(\beta(\theta), \tau^2(\theta), \gamma(\theta)\right).$$

Here, the synaptic weights $w_\beta, w_{\tau^2}, w_\gamma$ are the parameters to be optimized in the corresponding neural networks. The n-th log-likelihood function is

$$\begin{aligned}\ell_n\left(\beta(\theta_n), \tau^2(\theta_n), \gamma(\theta_n)\right) = &-\ln\left[(2\pi)^{K/2}\right] - \frac{1}{2}\ln\left[\left|\tau^2(\theta_n) R_M(\gamma(\theta_n)) + \Sigma_{\varepsilon;\theta_n}\right|\right] \\ &- \frac{1}{2}\left(\bar{Y}(\theta_n) - \mathbf{G}\beta(\theta_n)\right)^\top \left[\tau^2(\theta_n) R_M(\gamma(\theta_n)) + \Sigma_{\varepsilon;\theta_n}\right]^{-1}\left(\bar{Y}(\theta_n) - \mathbf{G}\beta(\theta_n)\right)\end{aligned} \quad (12)$$

where $\bar{Y}(\theta_n)$ is defined in (9) and $\Sigma_{\varepsilon;\theta_n} \approx \mathrm{Diag}\left\{\hat{V}(x_1;\theta_n)/M_1, \hat{V}(x_2;\theta_n)/M_2, \ldots, \hat{V}(x_K;\theta_n)/M_K\right\}$ with $\hat{V}(x_k;\theta_n) = \frac{1}{M_k - 1}\sum_{m=1}^{M_k}\left(Y_m(x_k;\theta_n) - \bar{Y}(x_k;\theta_n)\right)^2$. We now have the approximated $\left(\hat{\beta}(\theta), \hat{\tau}^2(\theta), \hat{\gamma}(\theta)\right)$, $\theta \in \Theta$.

In order to make the prediction of the response surface, we should know $\bar{Y}(\theta)$ and $\Sigma_{\varepsilon;\theta}$ as well. For the intrinsic uncertainty matrix $\Sigma_{\varepsilon;\theta}$, since it is a diagonal matrix, we then focus on the elements $\hat{V}(x_k;\theta), k = 1, 2, \ldots, K$. Since for each θ_n, we have $\hat{V}(x_k;\theta_n)$. We regard $\{\theta_n, n = 1, 2, \ldots, N\}$ as the input and $\left\{\hat{V}(x_k;\theta_n), n = 1, 2, \ldots, N\right\}$ as the real observations of a neural network. Then, following the procedure presented in Section 2.1, we attain the trained neural networks $\hat{\hat{V}}(x_k;\theta), k = 1, 2, \ldots, K$ and $\hat{\Sigma}_{\varepsilon;\theta}$. Similarly, we train the neural network $\hat{\bar{Y}}(\theta)$ by regarding $\{\theta_n, n = 1, 2, \ldots, N\}$ as the input and $\{\bar{Y}(\theta_n), n = 1, 2, \ldots, N\}$ as the real observations. In this way, we complete the estimation of the neural network-assisted stochastic kriging model.

2.3.2 Real-Time Stage

With the approximated parameters of the stochastic kriging model in hand, during the online application stage, we are then able to make the real-time decision. Assume the observed covariate is θ, with the plugged-in estimators, the approximated objective function is represented as

$$\hat{\hat{f}}(x;\theta) = \hat{\beta}(\theta)^\top \mathbf{g}(x) + \hat{\Sigma}_{M;\theta}(x,\cdot)^\top \left[\hat{\Sigma}_{M;\theta} + \hat{\Sigma}_{\varepsilon;\theta}\right]^{-1}\left(\hat{\bar{Y}}(\theta) - \mathbf{G}\hat{\beta}(\theta)\right) \quad (13)$$

If we assume a Gaussian correlation in the form of (8), the approximated objective function can be further simplified as

$$\begin{aligned}\hat{\hat{f}}(x;\theta) = &\mathbf{g}(x)^\top \hat{\beta}(\theta) + \hat{\tau}^2(\theta)\sum_{j=1}^{K}\sum_{i=1}^{K} a_{i,j;\theta} e^{-\hat{\gamma}(\theta)\|x - x_i\|^2}\hat{f}(x_j;\theta) \\ &- \hat{\tau}^2(\theta)\sum_{j=1}^{K}\sum_{i=1}^{K} a_{i,j;\theta} e^{-\hat{\gamma}(\theta)\|x - x_i\|^2}\mathbf{g}(x_j)^\top \hat{\beta}(\theta).\end{aligned} \quad (14)$$

where $a_{i,j;\theta}$ denotes the (i,j)-th element of the matrix $A \doteq \left[\hat{\Sigma}_{M;\theta} + \hat{\Sigma}_{\varepsilon;\theta}\right]^{-1}$. The arithmetic representation of the function (14) also supports the selection of the model (6). First, in some scenarios, the dimension of the covariate is significantly higher than the dimension of the decision variable. The inverse matrix A is a $K \times K$ matrix, where the simulation sample size K depends on the dimension of the decision variable. Thus, the high-dimensional covariates do not result in the prohibitive computational complexity of inversing the matrix. Second, given a fixed covariate, (14) reduces to a closed-form function. The objective function involves merely linear combinations of simple functions selected in $\mathbf{g}(x)$ and exponential squared terms, which can be optimized efficiently with gradient-based methods. Thus, at the real-time stage, we solve the optimization problem

$$x^*(\theta) = \arg\max_{x \in \mathscr{X}} \hat{\hat{f}}(x;\theta). \tag{15}$$

In terms of the optimization methods, we refer to the meta-model based simulation optimization methodologies; see Section 5.6 of Barton and Meckesheimer (2006). Since the stochastic kriging is a global meta-model, global optimization methodologies including the multistarts methods (Boender and Kan 1987) and the Lipschitzian optimization methods (Jones, Perttunen, and Stuckman 1993) can be adopted. For low-dimensional decision variables, a simpler grid search strategy is also feasible. An illustrative example and detailed experiment descriptions are included in Section 3.

Compared with traditional stochastic kriging model, our neural network-assisted model possesses the merits including the prediction accuracy and the computational efficiency (see online additional experiments part). However, sequential sampling strategies, which are widely employed in stochastic kriging approaches, cannot be naturally adopted into our models, for the reason that we require design points $\{x_k\}_{k=1}^K$ remain the same under different values of covariate.

3 EXPERIMENTS AND EVALUATION

In this section, we implement our two neural network-assisted methodologies with a multi-product newsvendor problem (see Özler et al. (2009)). The multi-product newsvendor problem is a stochastic inventory control problem, where a retailer determines the optimal order quantities for d different products. The aim of the retailer (decision-maker) is maximizing the expected profit with the random demand. Thus, the problem is formulated as

$$\max_{x \in \mathbb{R}^d} \sum_{i=1}^{d} \mathbb{E}\left[-c_i q_i + p_i \min\left(q_i, D_i\right) + s_i \left(q_i - D_i\right)^+\right]. \tag{16}$$

Here $x = (q_1, q_2, \ldots, q_d)^\top$ denotes the ordering quantities for each product. For the i-th product, D_i denotes the random demand, p_i denotes the sales price, c_i denotes the purchasing cost and s_i is the salvage value. It is required that $s_i \le c_i \le p_i$. Besides, we assume that $D_i \sim \mathscr{N}(\mu_i, \sigma_i^2)$, and the demands for different products are independent. In this way, the covariate for the problem is a $4d$-dimensional vector

$$\theta = \left(\mu_1, \sigma_1^2, p_1, s_1, \ldots, \mu_d, \sigma_d^2, p_d, s_d\right).$$

Denoting the cumulative distribution function of the i-th product demand by F_i, the optimal order quantity for the i-th product is

$$q^*(\mu_i, \sigma_i^2, p_i, s_i) = F_i^{-1}\left(\frac{p_i - c_i}{p_i - s_i}\right),$$

which we regard as the benchmark results for our methodologies.

For the two methodologies proposed in Section 2, we adopt the same experiment settings. In our experiments, we set $d = 3$. In terms of the feasible set, we let $\mathscr{X} = [1, 22]^3$. For the covariates, we let $\mu_i \in [8, 13], \sigma_i^2 \in [1, 6], p_i \in [1.1, 1.6], s_i \in [0.1, 0.6]$ for $i = 1, 2, 3$. We first pick 10 points evenly in each dimension for the covariate, and then randomly select $N \approx 100{,}000$ covariates for the simulation

experiments among these 10^{12} points. We divide the samples into the training set and the testing set, taking 70% and 30% of the whole set separately. Thus, the training set contains $N_{\text{train}} \approx 70,000$ samples and the testing set contains $N_{\text{test}} \approx 30,000$ samples.

For the evaluation of the methodologies, we calculate both the mean squared error (MSE) and the mean relative squared error (MRSE) as

$$\text{MSE} = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \|\hat{x}^*(\theta_n) - x^*(\theta_n)\|_2^2; \quad \text{MRSE} = \frac{1}{N_{\text{test}}} \frac{\sum_{n=1}^{N_{\text{test}}} \|\hat{x}^*(\theta_n) - x^*(\theta_n)\|_2^2}{\sum_{n=1}^{N_{\text{test}}} \|x^*(\theta_n) - \bar{x}^*(\theta_n)\|_2^2},$$

where $\bar{x}^*(\theta_n) = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} x^*(\theta_n)$ and $\|\cdot\|_2$ denotes the Euclidean distance (l_2-norm). We also plot the optimal order quantity surface with respect to the sales price and the salvage value, in Figure 1. Since there is no difference between the three products, we only plot the first product order quantity. Besides, we record the execution time for each step. All the experiments are based on running TensorFlow 2 and Python 3.7 on Nvidia GTX 1050 Ti (GPU) and Google Colaboratory.

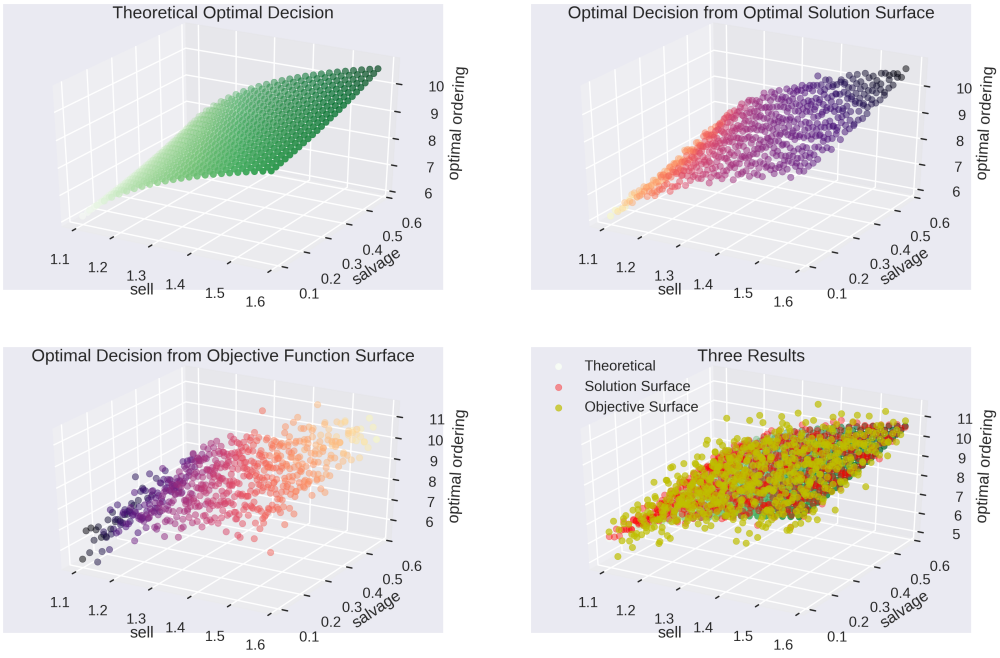

Figure 1: Optimal solution surfaces.

Method I: Neural Network-Assisted Optimal Solution Surface

In this part, we describe the procedure of solving problem (16) using the methodology introduced in Section 2.2. First, we obtain the optimal order quantities for each covariate $\theta_n, n = 1,2,\ldots,N$. The optimization strategy we adopt is the grid search.

Then we train the neural network with the training set, $(\theta_n, x^*(\theta_n)), n = 1,2,\ldots,N_{\text{train}}$. The neural network we apply has 4 hidden layers, containing 64, 32, 16, 8 nodes respectively in each hidden layer. The activation function utilized is the tanh function.

We apply the backward propagation method to minimize the mean squared loss on the training set. The training time is about 623.8791 s for 150 epochs. The MSE and MRSE are included in Tablet 1. At the online stage, referring the approximate optimal decision (5) costs about 0.02422 ms.

Method II: Neural Network-Assisted Objective Function Surface

First, for the design points x, we select evenly in the feasible set with a interval at length of 0.5. Then, for each θ_n in the training set and each selected design points, we perform $M = 10$ times simulation experiments to estimate the response surface. In order to fit the model (6), we adopt the Gaussian kernel (8) and let $\mathbf{g}(x) = (x, x^3, e^x)^\top$. $\beta(\theta)$ is approximated with a 4-hidden-layer neural network with 64, 32, 16 and 8 nodes respectively. $\tau^2(\theta)$ is approximated with a 3-hidden-layer neural network with 32, 16 and 8 nodes respectively. $\gamma(\theta)$ is approximated with a 3-hidden-layer neural network with 32, 16 and 8 nodes respectively. In all three neural networks, we adopt the tanh function as the activation function. The training time of maximizing the likelihood function (12) is about 5879.4987 s (150 epochs), using the backward propagation method.

We test the trained model on the test set. In terms of the optimization method, we adopt a multistart strategy with the truncated Newton method. For the test covariates, the mean time of referring the objective function (14) and then solving the optimization problem (15) is approximately 0.6662 ms. The MSE and MRSE are included in Tablet 1.

Table 1: Optimal decision prediction error.

	Solution Suface Method	Objective Function Method
MSE	0.001492	0.3055
MRSE	0.00036	0.07455

Evaluation

As is shown in Figure 1 and Tablet 1, both Method I and Method II demonstrate the plausible accuracy, while Method I appears to be more accurate for this experiment setting. Since Method I is a brute force fitting of the solution response surface, it is not surprising that given the large size of the offline simulation data, Method I provides a good fitting performance on the solution response surface. However, as indicated before, the neural network-approximated mappings do not preserve the feasibility all the time, and Method I lacks theoretical support. We will do more experiments in the future work to explore the reliance of Method I on the size of the offline simulation data. Meanwhile, we discuss the pros and cons of Method II. The advantages of Method II include:

- Method II incorporates the meta-modelling method, and therefore possesses a more stable structure, inheriting the advantages of classical meta-modelling approaches. Compared with Method I, Method II does not completely depend on the training of the neural network, especially when there are not sufficient offline simulation data.
- In some applications, it is not necessary to find the optimal solution, but only feasible solutions. Method II then possesses the advantage of having an objective function surface, providing useful information.

When the offline simulation data size is sufficiently large, Method II might have a lower prediction accuracy than Method I. We will further explore the pros and cons of Method II in future work, based on more numerical experiments and dedicate theoretical analysis. Besides, additional experiment results for comparing Method II with traditional stochastic kriging models and neural networks are presented in the online supplementary materials.

4 CONCLUSION AND FUTURE WORK

In this work, we propose two neural network-assisted methods to address real-time optimization problems for complicated stochastic systems with covariates, by effectively utilizing simulation experiments that are conducted offline before the real-time tasks. In the first method, we use brute-force neural networks to approximate the optimal solution surface with respect to the covariate, and the real-time tasks are accomplished via referring the approximate optimal decision via the neural network. For the second methodology, we propose a neural network-assisted stochastic kriging model to approximate the objective function surface with respect to the covariate. In this way, the real-time decision is made efficiently based on the explicit objective function. Experiment results indicate that both the methodologies possess the accuracy of solving the real-time optimization problems. Admittedly, it is possible that our methodologies result in a low optimization accuracy when the offline simulation experiments are not sufficient. Meanwhile, the neural network-assisted frameworks might suffer from overfitting the models. We will do more thorough comparisons in future work, and we are also working on providing more theoretic results of our methodologies. Other potential future works include introducing technologies from the neural network to deal with other problems involved in the simulation optimization area.

REFERENCES

Amaran, S., N. V. Sahinidis, B. Sharda, and S. J. Bury. 2016. "Simulation optimization: a review of algorithms and applications". *Annals of Operations Research* 240(1):351–380.

Andradóttir, S. 2006. "An Overview of Simulation Optimization via Random Search". *Handbooks in operations research and management science* 13:617–631.

Ankenman, B., B. L. Nelson, and J. Staum. 2010. "Stochastic Kriging for Simulation Metamodeling". *Operations Research* 58(2):371–382.

Barton, R. R. 2013. "Response surface methodology". *Encyclopedia of Operations Research and Management Science*:1307–1313.

Barton, R. R. 2015. "Tutorial: Simulation Metamodeling". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, V. W. Chan, I.-C. Moon, T. M. Roeder, C. Macal, and M. D. Rossetti, 1765–1779. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Barton, R. R., and M. Meckesheimer. 2006. "Metamodel-based Simulation Optimization". *Handbooks in Operations Research and Management Science* 13:535–574.

Barton, R. R., B. L. Nelson, and W. Xie. 2014. "Quantifying Input Uncertainty via Simulation Confidence Intervals". *INFORMS Journal on Computing* 26(1):74–87.

Boender, C., and A. R. Kan. 1987. "Bayesian Stopping Rules for Multistart Global Optimization Methods". *Mathematical Programming* 37(1):59–80.

Chen, X., B. E. Ankenman, and B. L. Nelson. 2012. "The Effects of Common Random Numbers on Stochastic Kriging Metamodels". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 22(2):1–20.

Chen, X., B. E. Ankenman, and B. L. Nelson. 2013. "Enhancing Stochastic Kriging Metamodels with Gradient Estimators". *Operations Research* 61(2):512–528.

Fu, M. C. 2006. "Gradient estimation". *Handbooks in Operations Research and Management Science* 13:575–616.

Fu, M. C. 2010. "Stochastic Gradient Methods for Simulation Optimization". *Wiley Encyclopedia of Operations Research and Management Science*.

Fu, M. C. et al. 2015. *Handbook of Simulation Optimization*, Volume 216. New York: Springer.

Gao, S., C. Li, and J. Du. 2019. "Rate Analysis for Offline Simulation Online Application". In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K.-H. G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, 3468–3479. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Hong, L. J., W. Fan, and J. Luo. 2020. "Review on Ranking and Selection: A new Perspective". *arXiv preprint arXiv:2008.00249*.

Hong, L. J., and G. Jiang. 2019. "Offline Simulation Online Application: A New Framework of Simulation-Based Decision Making". *Asia-Pacific Journal of Operational Research* 36(06):1940015.

Hong, L. J., B. L. Nelson, and J. Xu. 2015. "Discrete Optimization via Simulation". In *Handbook of Simulation Optimization*, 9–44. Springer.

Jalali, H., I. Van Nieuwenhuyse, and V. Picheny. 2017. "Comparison of Kriging-based Algorithms for Simulation Optimization with Heterogeneous Noise". *European Journal of Operational Research* 261(1):279–301.

Jones, D. R., C. D. Perttunen, and B. E. Stuckman. 1993. "Lipschitzian Optimization without the Lipschitz Constant". *Journal of optimization Theory and Applications* 79(1):157–181.

Katrakazas, C., M. Quddus, W.-H. Chen, and L. Deka. 2015. "Real-time Motion Planning Methods for Autonomous On-road Driving: State-of-the-art and Future Research Directions". *Transportation Research Part C: Emerging Technologies* 60:416–442.

Kim, S.-H., and B. L. Nelson. 2007. "Recent advances in ranking and selection". In *Proceedings of the 1994 Winter Simulation Conference*, edited by R. Barton, S. Henderson, B. Biller, M. hua Hsieh, and J. Shortle, 162–172. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

L. Salemi, P., E. Song, B. L. Nelson, and J. Staum. 2019. "Gaussian Markov Random Fields for Discrete Optimization via Simulation: Framework and Algorithms". *Operations Research* 67(1):250–266.

Masson, E., and Y.-J. Wang. 1990. "Introduction to Computation and Learning in Artificial Neural Networks". *European Journal of Operational Research* 47(1):1–28.

Nelson, B. L. 2010. "Optimization via Simulation over Discrete Decision Variables". In *Risk and Optimization in an Uncertain World*, edited by J. J. Hasenbein, 193–207. Catonsville, Maryland: INFORMS.

Ólafsson, S. 2006. "Metaheuristics". *Handbooks in Operations Research and Management Science* 13:633–654.

Özler, A., B. Tan, and F. Karaesmen. 2009. "Multi-product Newsvendor Problem with Value-at-risk Considerations". *International Journal of Production Economics* 117(2):244–255.

Qu, H., and M. C. Fu. 2014. "Gradient Extrapolated Stochastic Kriging". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 24(4):1–25.

Semelhago, M., B. L. Nelson, E. Song, and A. Wächter. 2020. "Rapid Discrete Optimization via Simulation with Gaussian Markov Random Fields". *INFORMS Journal on Computing*.

Shen, H., L. J. Hong, and X. Zhang. 2021. "Ranking and Selection with Covariates for Personalized Decision Making". *INFORMS Journal on Computing*.

Sun, L., L. J. Hong, and Z. Hu. 2014. "Balancing Exploitation and Exploration in Discrete Optimization via Simulation through a Gaussian Process-based Search". *Operations Research* 62(6):1416–1438.

Xie, W., B. L. Nelson, and R. R. Barton. 2014. "A Bayesian Framework for Quantifying Uncertainty in Stochastic Simulation". *Operations Research* 62(6):1439–1452.

Xie, W., B. L. Nelson, and R. R. Barton. 2020. "Statistical Uncertainty Analysis for Stochastic Simulation". *arXiv preprint arXiv:2011.04207*.

AUTHOR BIOGRAPHIES

HAOTING ZHANG received his M.S. degree in the Department of Industrial Engineering & Operations Research at the University of California Berkeley, and a B.S. degree in Mathematics at Sichuan University. He has research interests in simulation, stochastic modeling, and data analytics. His email address is haoting_zhang@berkeley.edu.

JINGHAI HE is an undergraduate student at Shanghai Jiao Tong University, majoring in finance and economics, and minoring in computer science. He has research interests in simulation, machine learning and financial engineering. His email address is ocean-he@sjtu.edu.cn.

DONGLIN ZHAN is an incoming Ph.D. student in the Department of Electrical Engineering at Columbia University. He has research interests in machine learning and data mining. His email address is dz2478@columbia.edu

ZEYU ZHENG is an assistant professor in the Department of Industrial Engineering & Operations Research at University of California Berkeley. He received his Ph.D. in Management Science and Engineering, Ph.D. minor in Statistics and M.A. in economics from Stanford University, and a B.S. in Mathematics from Peking University. He has done research in simulation, stochastic modeling, data analytics, statistical learning, and over-the-counter financial markets. His email address is zyzheng@berkeley.edu.