Journal of Optimization in Industrial Engineering

Vol.18, Issue 2, Summer & Autumn 2025, 1-11

Research Article

An artificial neural network metamodel to solve semi-expensive simulation optimization problems: A comparative study

Mohammad Behbahani [1], **S.T.A. Niaki** [2, *], **Mehran Moazeni** [3]

1. Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran
2. Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran
3. Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran

https://doi.org/10.71720/joie.2025.1183309

Received: 08 September 2024
Revised: 25 April 2025
Accepted: 14 May 2025

Keywords:
Semi-expensive simulation problems;
Simulation optimization;
Metamodel-based algorithm;
Artificial neural network

Abstract

Although many problems in the literature involve complex mathematical relationships, many still rely on simplified and unrealistic assumptions. Simulation is one of the most powerful tools for dealing with such problems, as it avoids the restrictive assumptions often required in stochastic systems. Simulation optimization techniques are generally classified into two broad categories: model-based and metamodel-based methods. In the first category, simulation and optimization components interact directly, thereby increasing simulation time and cost. To address this issue, a third component—called a metamodel—is introduced in the second category to estimate the system's relationship between input and output variables. Optimizing semi-expensive simulation problems often requires many simulation runs in model-based methods. However, the cost of validating metamodels also rises rapidly during iterations. A two-phase method has been proposed in the literature to reduce computation time. In the first phase, similar to a model-based algorithm, the simulation output is used directly in the optimization process. In the second phase, a validated metamodel replaces the simulation model. In this paper, an artificial neural network (ANN) is employed as the metamodel, and its performance is compared with that of the original algorithm, which employs a Kriging metamodel, on five well-known test functions and an (s, S) inventory model.

Citation:
Behbahani , M., Niaki, S. T. A., & Moazeni , M. (2025). An artificial neural network metamodel to solve semi-expensive simulation optimization problems: A comparative study. *Journal of Optimization in Industrial Engineering*, 18(2), 1-11. https://doi.org/10.71720/JOIE.2025.1183309

* **Corresponding Author:**
Seyed Taghi Akhavan Niaki
Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran
E-Mail: **niaki@sharif.edu**

1. Introduction

Since its inception over five decades ago, simulation has been a powerful tool for assessing potential risks and guiding managers and practitioners in decision-making under uncertainty. The simulation approach can more accurately anticipate risks and make more robust decisions in the face of uncertainty, ambiguity, and variability. Moreover, the simulation optimization approach identifies the optimal values of the input variables without explicitly evaluating every possible combination. The simulation-optimization approach has a wide range of applications across various problems. For instance, applications of the simulation-optimization approach include liver transplant management, maritime logistics optimization, semiconductor production planning, and parasitology calibration, as noted by Xu et al. (2015).

There are two primary methods to optimize a simulation model: (1) model-based and (2) metamodel-based. In the first method, the simulation and optimization phases are performed iteratively until a stopping criterion is met. In the latter, however, a surrogate model is used to mimic the simulation behavior when a high computational burden, which takes about 36-60 hours per replication, occurs, as in the Ford Motor Company crash simulation (Wang & Shan, 2007). Assuming an average of 50 iterations is required to optimize a two-variable problem, the total computational time would range from 75 days to 11 months, which is unacceptable in practice. Nevertheless, predicting an approximate metamodel (surrogate model) that replaces the simulation model would incur negligible cost relative to the original simulation cost (Wang et al., 2019). The widely used surrogate models in the literature include Kriging, Artificial Neural Networks (ANNs), Support Vector Regression (SVR), multivariate adaptive regression splines (MARS), and Radial Basis Function (RBF) models.

Establishing a surrogate model often goes through three steps as follows (Nguyen et al., 2014):

- Sampling input vectors and calculating corresponding model responses constitute a database for training a surrogate model.
- Constructing the surrogate model based on the database by selecting an appropriate method, e.g., Kriging, SVR, ANN, or RBF.
- Validating the model before it is used as a "surrogate" for the original model.

Both model- and metamodel-based methods have benefits and drawbacks, as summarized in Table 1. Although metamodel-based algorithms have some disadvantages, when a single simulation-model replication takes more than five minutes, one must use a metamodel-based algorithm. This is because, on average, a large number (typically 2000-4000) of replications are required in model-based simulations to evaluate simulation points and incorporate them into optimization. However, the number of simulation replications is minimized in a metamodel-based simulation. However, when a single simulation model run takes less than five minutes, metamodel approaches are inefficient because their validation processes require excessive time due to fitting errors. In these cases, the use of the model-based approach is recommended.

Table 1
Pros and Cons of Simulation Optimization Approaches

Method	Advantages	Disadvantages
Model-Based	• Accurate and numerically efficient for inexpensive simulation problems	• Needs to assess a large number of simulation points • Gives no insight into the objective function
Metamodel Based	• Relieve the computational expense by replacing the simulation with an approximation model.	• Existence of fitness error • Validation time • Gets trapped in the fitting and validating steps

Sometimes, a single simulation model replication takes more than two but fewer than 5 minutes on average. Due to their disadvantages, neither the model-based nor the meta-model-based simulation methods are appropriate. For this type of simulation, Moghaddam & Mahlooji (2017) presented a Semi-Metamodel-Based (SMB) algorithm that employs a metamodel, distinct from standard meta-model-based algorithms. They showed that their algorithm avoids the disadvantages of the model-based approach and mitigates some of the problems of meta-model-based algorithms. In the first phase, the algorithm operates as a model-based method, directly using the simulation output in the optimization phase. The optimization component proceeds in this phase, and metamodel construction is underway in parallel. As the number of simulation points in the experimental design increases at each step, the time required for metamodel validation decreases. The second phase of the algorithm employs the metamodel obtained in the first phase to evaluate solutions during the optimization stage, which begins once the metamodel is validated. Furthermore, they presented an optimization algorithm based on particle swarm optimization (PSO) that employs strategies to enhance its intensification and diversification.

The primary approach to constructing a surrogate model from available simulation points is Gaussian process regression (Kriging). This estimation method aims to obtain a minimum-variance estimate of any unsampled simulation point by smoothing out extreme values in the available simulation points. As such, the objective of the current paper is to compare the performances of the Kriging and an ANN approach used as surrogate models in the SMB algorithm (Moghaddam & Mahlooji, 2017). To this end, simulation points obtained from an (s, S) inventory control model and from five popular test functions are investigated.

The structure of the rest of the paper is as follows. The literature review is given in detail in Section 2. The employed

ANN metamodel is described in Section 3. Section 4 contains the characteristics of the SMB algorithm. The (s, S) inventory and the test functions are introduced in Section 5 to assess the performance of the ANN metamodel in the SMB algorithm and to compare its results with those obtained by the Kriging-based metamodel. Finally, Section 6 concludes with recommendations for future research.

2. Literature Review

In this section, we first survey model- and metamodel-based algorithms used to solve simulation optimization problems. Then, the fundamental differences between Kriging and ANN metamodels in simulation-optimization techniques are discussed. As mentioned above, several model-based approaches have been proposed in the simulation-optimization literature. Wang (2005) developed an optimization approach using a hybrid method of genetic algorithm (GA) and neural network. They employed a neural network to predict the objective function value, then used a GA for its practical, robust evolutionary search to determine the optimal values of the input parameters. Shi & Ólafsson (2000) proposed the Nested Partition method as a global sampling technique that continuously adapts by partitioning the feasible solution region, thereby reducing computational load by selecting the most promising areas. Similarly, Ahmed & Alkhamis (2009) integrated simulation and optimization to design a decision-support tool for a governmental hospital in Kuwait; they evaluated the impact of various staffing levels on service efficiency to determine the optimal number of doctors. Tsai & Fu (2014) employed two GA-based algorithms for a discrete simulation-optimization problem with a single stochastic constraint; they adopted different sampling rules and search mechanisms and thus provided different statistical guarantees.

Table 2
Works on the comparative study of metamodeling methods

Publication	year	surrogate model							Type of performance metric
		Kriging	ANN	RBF	Regression	MARS	SVR	other	
Willmes et al.	(2003)	√	√						Problem dimension, accuracy
Yuan & Guangchen	(2009)	√	√						Four proposed performance measures
Zhao and Xue	(2010)	√	√		√				accuracy, confidence, robustness, efficiency, and robustness
Backlund et al.	(2012)	√		√			√		Problem dimension
Vicario et al.	(2016)	√	√						Accuracy(at a satisfactory cost)
Beşikçi et al.	(2016)		√		√				RMSE (root mean square error)
Díaz-Manríquez et al.	(2017)	√		√			√	√	The budget for computational time
Østergård et al.	(2018)					√	√	√	accuracy, efficiency, ease-of-use, robustness, and interpretability
Amouzegar et al.	(2018)	√	√	√		√	√		A proposed ranking metric, accuracy, and computational efficiency
Cheng et al.	(2020)	√		√			√	√	The accuracy and efficiency

Meta-modeling has been applied to develop simulations for various purposes, including early design decisions, uncertainty and sensitivity analyses, design optimization, and model calibration (Nguyen et al., 2014). Metamodel-based techniques are among the most popular research areas in the field of simulation optimization, and numerous algorithms have been proposed. For instance, Jones et al. (1998) used the Kriging approach within an Efficient Global Optimization (EGO) algorithm to interpolate between function values and select future samples based on an expected improvement metric for simulations with deterministic outputs. In addition, ANNs have been employed in numerous algorithms proposed in the literature to optimize computationally expensive simulations. To name a few, Dengiz et al. (2009) optimized two manufacturing systems utilizing neural network metamodels. A tabu search (TS) metaheuristic was employed to train the ANNs and improve the performance of the meta-modeling approach. Mohammad Nezhad & Mahlooji (2014) presented an ANN metamodel for expensive continuous simulation optimization (SO) with stochastic constraints. Capturing the nonlinear nature of the ANN, the SO problem was iteratively approximated via nonlinear programming problems, whose (near-) optimal solutions yielded estimates of the global optimum. Considering the uncertainty in the random parameters, Roy& Chakraborty (2020) proposed a sequential updating approach that improves the accuracy of the SVR metamodel. A comprehensive review of other metamodels for optimization strategies applied to computationally expensive black-box functions is available in Shan & Wang (2010).

Willmes et al. (2003) compared the performance of feedforward neural networks and Kriging as fitness approximations for evolutionary optimization in both offline and online learning. Yuan & Guangchen (2009) applied four performance measures to evaluate different types of metamodel performance, such as providing good starting

points for gradient-based search and the accuracy of placing optima in the correct location. Furthermore, Vicario et al. (2016) compared the performance of the Kriging model with that of an ANN model to determine which model achieves higher accuracy in predicting results from a four-dimensional computational fluid dynamics system. Moreover, Beşikçi et al. (2016) compared the performance of an ANN metamodel with that of a multiple regression (MR) model, confirming the superiority of the former. Interested readers are referred to Østergård et al. (2018) for other comparison studies among linear regression with ordinary least squares (OLS), random forest (RF), SVR, MARS, Gaussian process regression (GPR), and neural network metamodel. The authors considered five performance indicators for eight test problems and 19 mathematical test functions in their work. Some researchers assessed the enhanced version of these surrogate models. Besides, Cheng et al. (2020) investigated several surrogate metamodels, including Kriging, SVR, RBF, and Low-Rank tensor Approximation (LRA), based on their applications in variance-based global sensitivity analyses.
Table 2 lists published papers that have used metamodels to predict the models' responses.
Neural networks and Kriging approximations are among the most attractive techniques in simulation-optimization meta-modeling, but they have not been surveyed in the context of SMB algorithms. In the following sections, we investigate the accuracy and computational efficiency of these meta-modeling techniques for SMB algorithms using two- and three-dimensional test functions.

3. Neural Network Metamodel

Neural networks (NNs) are powerful tools for approximating unknown nonlinear functions and have gained widespread applications in various fields. ANNs can approximate arbitrary smooth functions and can be trained using noisy response data. ANNs were developed to mimic neural processing and can be implemented on a digital computer using networks of numerical processors whose inputs and outputs are linked according to specific topologies (Barton & Meckesheimer, 2006).
NNs used for function approximation are typically multilayer feed-forward networks. A feedforward ANN is an architecture in which signals flow from the input layer to the output layer. Feedforward neural networks can approximate smooth functions arbitrarily well, provided sufficient nodes and layers are available. Multilayer ANNs are usually capable of modeling more complex problems.
Let the tan-sigmoid $(f_{tan}(x))$ and linear $(f_{lin}(x)$The functions serve as the transfer functions for the hidden and output layers.

$$f_{tan}(x) = \frac{2}{1+exp(-2x)} - 1 \qquad (1)$$

$$f_{lin}(x) = x. \qquad (2)$$

Figure 1 depicts a sample two-layered feedforward ANN. In this network, each neuron in a layer is linked only with neurons of a different layer, and the following two equations determine the outputs:

$$z_s^k(x) = \frac{2}{1+exp\left(-2\left(\sum_{j=1}^{d} v_{js}^0 x_j + b_s^k\right)\right)} - 1 \; ; \quad k = 1, s = 1, \dots, nn_k \qquad (3)$$

$$y(x) = \sum_{s=1}^{nn_{k-1}} v_{s's}^{k-1} z_{s'}^{k-1}(x) + b_s^k) \; ; \quad k = 1, s = 1, \qquad (4)$$

where x_j serves as an input neuron j, v_{js}^0 denotes the weight of the connection link between the input neuron j and a hidden neuron s in the first layer and $v_{s's}^{k-1}$ denotes the weight of the connection link between the neuron s' in the hidden layer $k-1$ and neuron s in the layer k. Moreover, b_1^2 denotes the bias value of the output neuron and $z_s(x)$ is the activation value of the hidden neuron s. A training method defined on a feed-forward ANN to solve such mapping problems modifies the weights and biases in such a manner that the following performance criterion is minimized:

$$MSE = \frac{\sum_{i=1}^{n}(\bar{w}(x_i) - y(x_i))^2}{n}, \qquad (5)$$

where x_i for $i = 1, \dots, n$ stands for the input patterns, $\bar{w}(x_i)$ denotes the target outputs, and n shows the number of input patterns (Mohammad Nezhad & Mahlooji, 2014).

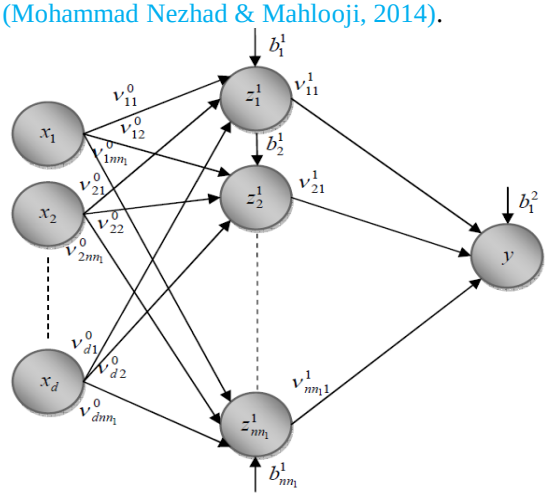

Fig. 1. A sample two-layered feed-forward artificial neural network

3.1. ANN versus kriging

As stated in Section 2, some studies in the literature compare ANN with Kriging and other simulation metamodels (e.g., Wang & Shan, 2007; Willmes et al., 2003; Yuan & Guangchen, 2009; Vicario et al., 2016). The comparisons concern one or some of the efficiency, accuracy, interpretability, and robustness performance measures. Some of the advantages and disadvantages of Kriging and ANN metamodels are reported in Table 3 (Østergård et al., 2018).

Table 3
Comparison of Kriging and ANN metamodels

Metamodel	Advantages	Disadvantages
Kriging	Less sensitive to chosen settings	Among the slowest algorithms, it becomes unstable for large training sets
ANN	Most accurate nonlinear method; efficient for large training sets; less time-consuming for new predictions	Many possible configurations; least interpretable method; accuracy varies across different training runs

As shown in Table 3, determining the optimal combination of neurons in the hidden layer, weights, and biases (i.e., a configuration), as well as selecting a transfer function and training algorithm, is not straightforward in an ANN-based metamodel. Although comparing ANNs with other metamodels is somewhat challenging, several heuristics exist for determining ANN hyperparameters. For example, Rigoni & Lovison (2007) stated that for a network with several training data q time, the number of output variables m, and the hidden layer neurons h could be calculated by:

$$h \leq fix\left(\frac{m(q-1)}{(n+m+1)}\right), \quad (6)$$

where $fix(.)$ is the greatest integer less than or equal to its argument, and n denotes the number of input variables.

4. The Proposed ANN Semi-Meta-Model-Based Algorithm

A semi-meta-model-based algorithm was introduced by Moghaddam & Mahlooji (2017) to address model- and meta-model-based difficulties in semi-expensive simulation optimization problems that take approximately 2–5 minutes. Kriging was the primary metamodel employed in their work to estimate the relationship between the input and output variables of the simulation model. Furthermore, they used a new particle swarm optimization (PSO) algorithm with enhanced exploration and exploitation. In the present study, an ANN-based metamodel is adopted instead of Kriging; the steps of the algorithm are summarized below, along with the differences relevant to this paper.

Simulation outputs are used in metamodel fitting/validation, and in the optimization stage; thus, these components no longer operate independently. If metamodel validity is rejected, several simulation points rather than a single point are added to fit and validate expensive simulation problems. Consequently, processing time can be reduced significantly when an ANN-based metamodel is employed to solve semi-expensive simulation-optimization problems.

The flowchart of the proposed ANN semi-meta-model-based algorithm, shown in Figure 2, consists of two phases. In the first phase, the algorithm has model-based characteristics: the simulation output is used directly for optimization. In the second phase, the algorithm behaves as a metamodel-based method: the validated metamodel obtained from Phase 1 replaces the simulation model. As in Moghaddam & Mahlooji (2017), a spatial hole-PSO (SH-PSO) is used in both phases to draw newly generated particles toward empty regions of the solution space.

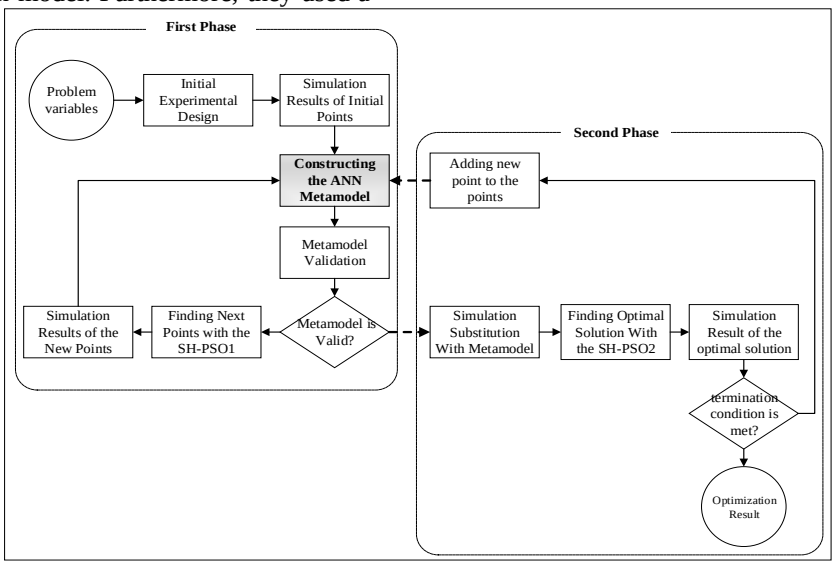

Fig. 2. The flowchart of the ANN semi-meta-model-based algorithm

4.1. Constructing the ANN metamodel

The Levenberg–Marquardt algorithm, one of the fastest back-propagation methods, is used here for network training. Designed to approach second-order training speed without computing the Hessian matrix, it requires more memory than other algorithms. Still, it is well suited to function-approximation problems in which the network may contain several hundred weights and high accuracy is needed (Yuan & Guangchen, 2009). Seventy percent of the available data is allocated to training, and the remaining 30% is used for validation, in accordance with MATLAB recommendations.

4.2. Algorithm specifications

When comparing the proposed ANN-based and the Kriging version of Moghaddam & Mahlooji (2017), most specifications—initial experimental design, PSO variant, metamodel validation, and so on—are kept identical. Key points are listed below.

- Initial sampling. Both methods use Latin hypercube sampling, which maximizes the minimum distance between points.
- Validation scheme. Leave-one-out cross-validation is employed. The studentized prediction error for each left-out point *i* is

$$t_{r-1}^{i} = \frac{\bar{\bar{w}}(x_i) - \bar{y}^*(x_i)}{\sqrt{\widehat{var}(\bar{\bar{w}}(x_i)) + \widehat{var}(y^*(x_i))}}, \quad (7)$$

where $\bar{\bar{w}}(x_i)$ is the mean of the bootstrapped simulation outputs and $\bar{y}^*(x_i)$ represents the mean of the bootstrapped ANN predictors of the left-out point x_i. Furthermore, $\widehat{var}(\bar{\bar{w}}(x_i))$ and $\widehat{var}(y^*(x_i))$ are the variance of the averages of the bootstrapped simulation outputs and the variance of the bootstrapped ANN predictors of x_i, respectively.

Spatial-Hole Particle Swarm Optimization (SH-PSO) extends the canonical PSO by incorporating a hole-detection routine. After each iteration, the search space is scanned for sparsely sampled regions ("spatial holes"); each hole is assigned a weight between 0 and 1 proportional to its potential improvement. New particles are probabilistically attracted toward the highest-weighted holes. This extra drift term enhances global exploration while preserving the local-best and global- best pulls of standard PSO.

In Phase 1 of the SMB framework, solution quality is still evaluated using the simulation model, but SH-PSO steers the swarm into underexplored regions more quickly. In Phase 2, once the ANN metamodel has been validated, the same SH-PSO operates inside the optimization loop, relying on the much cheaper surrogate evaluations. Benchmarks reported by Moghaddam & Mahlooji (2017) show that SH-PSO reached a validated metamodel 12% faster and used ≈approximately 15% fewer function evaluations than canonical PSO on problems of similar size, making it well-suited to the semi-expensive setting studied here. For implementation details of the hole-detection procedure, readers may consult Moghaddam & Mahlooji (2017)

5. Applications and Results

An (s, S) inventory control model, along with five popular test functions, is used as a simulation model to assess the performance of the two semi-meta-model-based algorithms discussed in Section 4.

5.1. The (s, S) inventory as a realistic simulation model

An (s, S) inventory control model is presented in this section to compare the performance of the proposed ANN and the existing Kriging as simulation metamodels within the SMB algorithm. In this model, a replenishment order is placed when the inventory position (on-hand inventory + outstanding orders – backlogs) drops to or below the reorder point *s*. This replenishment order returns the inventory position to the order up to level *S*. Note that *S* consequently denotes the maximum inventory position. The parameters of this (*s*, *S*) inventory control model are as follows (Biles et al., 2007):

- The holding cost is charged at $1 per day per item
- The shortage cost is charged at $5 per day per item
- The simulation period is 4,000 days per replicate
- The ordering cost is $32 plus $3 per unit ordered
- The order arrival time follows an exponential distribution with a mean of 6 days.
- The inventory position is reviewed at the end of each day
- The customer demand is exponentially distributed with a mean of 90
- The number of units demanded per customer is 1.

The parameter values for the optimization processes, ANN-SMB and Kriging-SMB, are set as specified in Table 4. Both algorithms are coded in MATLAB 8.4. Besides, to keep the simulation and optimization models fully compatible, the inventory model is likewise implemented in MATLAB 8.4[1].

Table 4
The parameters of the ANN-SMB and the Kriging-SMB algorithms to solve the (s, S) inventory-control problem.

Parameter	Value	
	ANN-SMB Algorithm	Kriging-SMB Algorithm
Number of initial design points (idp)	20	10
Number of simulation replications (r)	4	4
Particle size (m)	10	10
Maximum number of spatial holes (sh^{max})	5	5
Maximum number of neighbors to be evaluated (N^{max})	15	10
Maximum acceptable error for metamodel validation	0.05	0.1
Number of bootstrap replications for metamodel validation	100	100
The strength of the movement towards the local best (c_1)	2	2
The strength of the movement towards the global best (c_2)	2	2
Terminating condition (T_{min})	0.005	0.005

[1] Complete MATLAB source code and data are openly available at https://github.com/Mbehbahani/ANN-SMB-SemiExpSimOpt

Table 5 and Figure 3 show ten independent replications of the simulation optimization for both the ANN and the Kriging semi-meta-model-based algorithms. The solutions and objective functions obtained from the number of solution evaluations are presented in this table.

Table 5
Comparison of ANN-SMB and Kriging-SMB algorithms for the (s, S) inventory control problem.

Methods	ANN-SMB Algorithm			Kriging-SMB Algorithm		
Run	Solution (s, S)	Objective	No. of Simulation Evaluations	Solution (s, S)	Objective	No. of Simulation Evaluations
1	(741, 845)	604.58	195	(678, 855)	606.97	280
2	(658, 894)	611.75	285	(702, 880)	599.64	260
3	(712, 870)	590.07	240	(692, 813)	613.95	280
4	**(721, 876)**	**584.29**	330	(696, 848)	594.43	240
5	(682, 824)	605.16	285	(628, 796)	635.41	280
6	(736, 873)	587.54	285	(717, 886)	595.96	280
7	(665, 835)	601.58	150	**(749, 895)**	**584.48**	240
8	(740, 887)	578.24	375	(662, 857)	617.38	240
9	(696, 857)	591.68	330	(741, 811)	591.97	260
10	(709, 874)	597.92	150	(730, 848)	596.91	280
Avg.		595.28	262.5		603.71	264
Std.		10.61			14.99	

The best (s, S) and the best objective values are shown in bold.

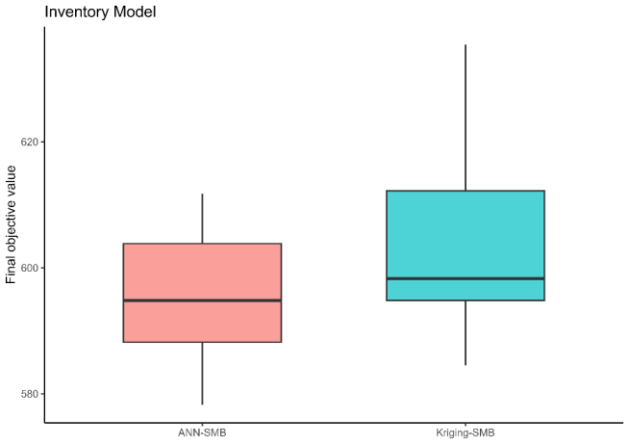

Fig. 3. Box plots of final objective values for the (s, S) inventory-control model over 10 independent replications.

As shown in Table 5 and Figure 3, the mean optimal value of the ANN-SMB algorithm is 595.28, with a standard deviation of 10.61. These values are 603.71 and 14.99 for the Kriging-SMB algorithm, respectively. Here, the null hypothesis $\mu_{ANN-SMB} \leq \mu_{Kriging-SMB}$ A two-sample Student t-test is used to determine if the obtained average optimal value of the ANN-SMB is not significantly greater than that of the Kriging-SMB. As the t-statistic and p-value are -1.4782 and 0.9133, respectively, the null hypothesis cannot be rejected at the 5% significance level. This implies that the two algorithms yield solutions of equal quality on average. Based on the results shown in the fourth and seventh columns of Table 5, a similar hypothesis test is performed here to compare the average number of simulation evaluations required by the two algorithms. As the P-value from the two-sample t-test is 0.953, the null hypothesis of equal means cannot be rejected at the 5% significance level. This implies that the average number of simulation evaluations used in the two algorithms does not differ statistically.

5.2. Analytical test functions

This subsection compares ANN-SMB and Kriging-SMB on five single-objective test functions: Sphere, Griewank, Schaffer's F6, Rastrigin, and Rosenbrock (Molga & Smutnicki, 2005). The solutions obtained using the ANN-SMB algorithm on these test functions, along with a comparative analysis with the Kriging-SMB solutions reported in Moghaddam & Mahlooji (2017), are presented in Table 6. In this table, the values of idp, r, m, and sh^{max} The parameters are set to 20, 5, 10, and 5 for the ANN-SMB algorithm, and to 10, 5, 10, and 5 for the Kriging-SMB algorithm. The other parameter values are the same as the ones used in Table 4. Moreover, while the optimal solutions of the functions are shown in the third column, the fourth and fifth columns contain the sample means and standard deviations (Std.) of the solutions obtained by the proposed ANN-SMB algorithm. Similarly, the sample means and standard deviations of the solutions obtained by the Kriging-SMB algorithm are shown in the seventh and eighth columns of Table 6, respectively. Finally, the number of function evaluations required for the ANN-SMB and Kriging-SMB algorithms to reach the solutions is reported in the sixth and ninth columns, respectively.

Table 6
Numerical results of using the ANN-SMB and the Kriging-SMB algorithms on five test problems.

Function	Formula	Optimal	ANN-SMB Algorithm			Kriging-SMB Algorithm		
			Objective		Number of function evaluations	Objective		Number of function evaluations
			Mean	Std.		Mean	Std.	
Sphere	$y=\sum_{j=1}^{d} x_j^2$ $d=2,-2\le x_j\le 2, j=1,2$	$x^*=[0,0]$ $y^*=0$	0.018	0.024	266	0.189	0.09	244
Griewank	$y=\frac{1}{4000}\sum_{j=1}^{d} x_j^2-\prod_{j=1}^{d}\cos\left(\frac{x_j}{\sqrt{j}}\right)+1$ $d=2,-8\le x_j\le 8, j=1,2$	$x^*=[0,0]$ $y^*=0$	0.024	0.017	415	0.335	0.22	368
Schaffer's F6	$y=0.5-\frac{\left(\sin\sqrt{(x_1^2+x_2^2)}\right)^2-0.5}{\left(1+0.001(x_1^2+x_2^2)\right)^2}$ $d=2,-3\le x_j\le 3, j=1,2$	$x^*=[0,0,0]$ $y^*=0$	0	0	492	0.501	0.05	425
Rastrigin	$y=\sum_{j=1}^{d}\left(x_i^2-10\cos(2\times\pi x_j)+10\right)$ d= $3,-5\le x_j\le 5,\ j=1,2,3$	$x^*=[0,0,0]$ $y^*=0$	4.317	1.122	460	0.772	0.16	457
Rosenbrock	$y=\sum_{j=1}^{d-1}\left(100\left(x_{j+1}-x_j^2\right)\right)^2+\left(x_j-1\right)^2$ $d=3,-3\le x_j\le 3, j=1,2,3$	$x^*=[1,1,1]$ $y^*=0$	2.133	1.142	528	3.103	1.67	483
Avg.					432.2			395.4

As shown in Table 6 and Figure 4, while the proposed ANN-SMB algorithm outperforms the Kriging-SMB algorithm on average across the Sphere, Griewank, Schaffer's F6, and Rosenbrock test functions, the optimization results for the Rastrigin function indicate that Kriging-SMB performs better in this case. Moreover, the results of a two-sample Student's t-test for the equality of the mean number of function evaluations across the two algorithms indicate no significant difference (P-value = 0.570). This implies that the proposed algorithm is generally superior. Besides, the outcome of a one-way blocked (the 5 test problems are used as blocks) analysis of variance (ANOVA), which is used to test the equality of the performance means of the two algorithms in all test problems, is reported in Table 7 based on 20 replications in each block. Once again, because the null hypothesis is not rejected at the p-value of 0.321, the two algorithms do not differ significantly.

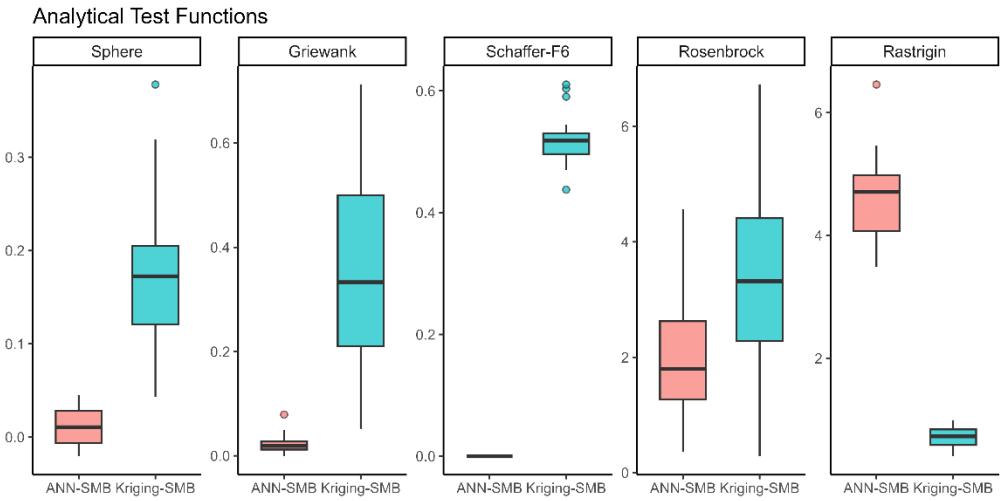

Fig. 4. Box-plots of final objective values across five analytical test functions over 20 independent runs per algorithm. While the blocked ANOVA test (Table 7) indicates no significant overall difference between ANN-SMB and Kriging-SMB (p = 0.321), the plots highlight that ANN-SMB outperforms Kriging-SMB on four of the five functions (Sphere, Griewank, Schaffer-F6, Rosenbrock), whereas Kriging-SMB is superior on Rastrigin.

5.3. Practical implications

The ANN-SMB algorithm is most useful when each simulation run takes a few minutes, which is common in warehouse inventory studies, production-line scheduling, and mid-scale logistics design. By reducing simulation evaluations by roughly 5–10% without sacrificing solution quality, the method shortens optimization turnaround from "overnight" to "same shift." This benefits operations planners in small- to medium-sized manufacturers and retailers that lack high-performance computing capabilities, as well as consultancy firms that deliver what-if analyses under tight client deadlines. Because the metamodel layer is coded in MATLAB, analysts using commercial packages (Arena, AnyLogic, Simio) can integrate the ANN surrogate with minimal changes to the underlying simulation model.

Table 7
Analysis of variance for comparing ANN-SMB and the Kriging-SMB across the five test functions

Source	DF	Adj. SS	Adj. MS	F-Value	P-Value
Treatment	1	59,577	59,577	1.00	0.321
Block	4	247,438	61,860		
Error	94	5,626,298	59,854		
Total	99	5,933,313			

6. Limitations, Conclusion, and Future Works

Although the proposed ANN-SMB algorithm performs well on the selected benchmarks, three constraints should be considered. First, ANN training time and memory requirements increase sharply with the number of hidden-layer neurons and training samples; scalability beyond 10 decision variables was not investigated. Second, solution quality shows moderate hyperparameter sensitivity: Altering the hidden layer size from 10 to 30 neurons changes the final objective value by approximately ±6%. Third, the current study assumes low-noise, continuous decision spaces; effectiveness on highly noisy or discrete problems remains to be verified.

This paper presented a new version of a semi-meta-model-based simulation-optimization algorithm, in which an artificial neural network was used as the primary metamodel. The approach is intended for "semi-expensive" simulations that run for roughly 2–5 minutes per replication, a standard range in industrial scheduling and inventory studies.

Although statistical comparisons with a Kriging-based metamodel did not show a significant overall difference between ANN-SMB and Kriging-SMB on the five analytical

test functions, we demonstrated that ANN-SMB is superior on four of those functions and on an (s, S) inventory-optimisation model. In numerical terms, ANN-SMB achieved the same or better objective values with an average of 5 %–10 % fewer simulation evaluations.

For operations planners in small- to medium-sized manufacturing and logistics firms, ANN-SMB can cut wall-clock optimisation time by several hours to days when each simulation run lasts a few minutes. The method, therefore, enables faster re-tuning of inventory and scheduling policies without access to high-performance computing. Analysts working with commercial simulation packages (e.g., Arena, AnyLogic) can embed the proposed ANN surrogate with minimal code changes, because only the metamodel layer—not the core simulation—needs modification.

Because the choice of metamodel entails a trade-off between accuracy and computational speed, practitioners should weigh the benefits and drawbacks of each method before implementation.

The present study did not test other popular metamodels, such as radial basis functions (RBF) or regression models, within the SMB framework; evaluating those alternatives remains an interesting topic for future work. Modifying the underlying metaheuristic during the optimization steps could further affect metamodel performance. Finally, benchmarking different SMB variants against real-world semi-expensive problems is recommended for future investigation.

References

Ahmed, M. A., Alkhamis, T. M. (2009). "Simulation optimization for an emergency department healthcare unit in Kuwait," *European Journal of Operational Research*, 198 (3): 936-942.

Amouzgar, K., Bandaru, S., & Ng, A. H. (2018). "Radial basis functions with a priori bias as surrogate models: A comparative study, " *Engineering Applications of Artificial Intelligence*, 71, 28-44.

Backlund, P. B., Shahan, D. W., & Seepersad, C. C. (2012). "A comparative study of the scalability of alternative metamodelling techniques," *Engineering Optimization*, *44*(7), 767-786.

Barton, R. R., Meckesheimer, M. (2006). "Metamodel-based simulation optimization," *Handbooks in Operations Research and Management Science*, 13: 535-574.

Beşikçi, E. B., Arslan, O., Turan, O., Ölçer, A. (2016). "An artificial neural network based decision support system for energy efficient ship operations," *Computers & Operations Research*, 66: 393-401.

Biles, W. E., Kleijnen, J. P., Van Beers, W. C., Van Nieuwenhuyse, I. (2007). "Kriging metamodeling in constrained simulation optimization: an explorative study," in 2007 Winter Simulation Conference, pp. 355-362: IEEE.

Cheng, K., Lu, Z., Ling, C., & Zhou, S. (2020). "Surrogate-assisted global sensitivity analysis: an overview. *Structural and Multidisciplinary Optimization*," 1-27.

Dengiz, B., Alabas-Uslu, C., Dengiz, O. (2009). "A tabu search algorithm for the training of neural networks," *Journal of the Operational Research Society*, 60 (2): 282-291.

Díaz-Manríquez, A., Toscano, G., & Coello, C. A. C. (2017). "Comparison of metamodeling techniques in evolutionary algorithms," *Soft Computing*, 21(19), 5647-5663.

Jones, D. R., Schonlau, M., Welch, W. J. (1998). "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, 13(4): 455-492.

metamodel-based method for solving semi-expensive simulation optimization problems," *Communications in Statistics-Simulation and Computation*, 46 (6): 4795-4811.

Mohammad Nezhad, A., Mahlooji, H. (2014). "An artificial neural network metamodel for constrained simulation optimization," *Journal of the Operational Research Society*, 65 (8): 1232-1244.

Molga, M., Smutnicki, C. (2005). "Test functions for optimization needs," https://www.robertmarks.org/Classes/ENGR5358/Papers/functions.pdf

Nguyen, A.-T., Reiter, S., Rigo, P. (2014). "A review on simulation-based optimization methods applied to building performance analysis," *Applied Energy*, 113: 1043-1058.

Østergård, T., Jensen, R. L., Maagaard, S. E. (2018). "A comparison of six metamodeling techniques applied to building performance simulations," *Applied Energy*, 211: 89-103.

Rigoni, E., Lovison, A. (2007). "Automatic sizing of neural networks for function approximation," in 2007 IEEE International Conference on Systems, Man and Cybernetics, pp. 2005-2010: IEEE.

Roy, A., & Chakraborty, S. (2020). "Support Vector Regression Based Metamodel by Sequential Adaptive Sampling for Reliability Analysis of Structures," *Reliability Engineering & System Safety*, 106948.

Shan, S., Wang, G. G. (2010). "Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions," *Structural and Multidisciplinary Optimization*, 41 (2): 219-241.

Shi, L., Ólafsson, S. (2000). "Nested partitions method for global optimization," *Operations Research*, 48 (3): 390-407.

Tsai, S. C., Fu, S. Y. (2014). "Genetic-algorithm-based simulation optimization considering a single stochastic constraint," *European Journal of Operational Research*, 236 (1): 113-125.

Vicario, G., Craparotta, G., Pistone, G. (2016). "Metamodels in computer experiments: Kriging versus artificial

neural networks," *Quality and Reliability Engineering International*, 32 (6): 2055-2065.

Wang, L. (2005). "A hybrid genetic algorithm–neural network strategy for simulation optimization," *Applied Mathematics and Computation*, 170 (2): 1329-1343.

Wang, G. G., Shan, S. (2007). "Review of metamodeling techniques in support of engineering design optimization," *Journal of Mechanical Design*, 129 (4): 370-380.

Wang, X., Wang, G. G., Song, B., Wang, P., Wang, Y. (2019). "A Novel evolutionary sampling assisted optimization method for high dimensional expensive problems," *IEEE Transactions on Evolutionary Computation*, DOI: 10.1109/TEVC.2019.2890818.

Willmes, L., Back, T., Jin, Y., Sendhoff, B. (2003). "Comparing neural networks and kriging for fitness approximation in evolutionary optimization," in the 2003 Congress on Evolutionary Computation, CEC'03., vol. 1, pp. 663-670: IEEE.

Xu, J., Huang, E., Chen, C.-H., Lee, L. H. (2015). "Simulation optimization: A review and exploration in the new era of cloud computing and big data," *Asia-Pacific Journal of Operational Research*, 32 (3): 1550019.

Yuan, R., Guangchen, B. (2009). "Comparison of neural network and Kriging method for creating simulation-optimization metamodels," in 2009 eighth IEEE international conference on dependable, autonomic and secure computing, 2009, pp. 815-821: IEEE.

Zhao, D., & Xue, D. (2010). "A comparative study of metamodeling methods considering sample quality merits," *Structural and Multidisciplinary Optimization*, 42(6), 923-938.