IEICE TRANS. FUNDAMENTALS, VOL.E105–A, NO.9 SEPTEMBER 2022

PAPER

Bayesian Optimization Methods for Inventory Control with Agent-Based Supply-Chain Simulator

Takahiro OGURA[†,††a)], **Haiyan WANG**[†††], **Qiyao WANG**[†††], **Atsuki KIUCHI**[†††], **Chetan GUPTA**[†††], *Nonmembers, and* **Naoshi UCHIHIRA**[††], *Senior Member*

SUMMARY We propose a penalty-based and constraint Bayesian optimization methods with an agent-based supply-chain (SC) simulator as a new Monte Carlo optimization approach for multi-echelon inventory management to improve key performance indicators such as inventory cost and sales opportunity loss. First, we formulate the multi-echelon inventory problem and introduce an agent-based SC simulator architecture for the optimization. Second, we define the optimization framework for the formulation. Finally, we discuss the evaluation of the effectiveness of the proposed methods by benchmarking it against the most commonly used genetic algorithm (GA) in simulation-based inventory optimization. Our results indicate that the constraint Bayesian optimization can minimize SC inventory cost with lower sales opportunity loss rates and converge to the optimal solution 22 times faster than GA in the best case.

key words: *Bayesian optimization, inventory management, simulation-based optimization, agent-based simulator*

1. Introduction

Manufacturers with global supply chains (SCs) have been faced with increasing challenges due to furious competition and dynamically changing global markets. These SCs commonly use multiple tiers, or echelons, of stocking locations to minimize logistics costs, inventory costs, and sales opportunity loss. Besides the complex multi-tier SC structure, other factors have further complicated SC optimization, for example, the highly uncertain and rapidly changing market demand, uncertain lead time between tiers, tremendous variety of product types, and much shortened product lifecycles.

In practice, the aforementioned factors must be considered when optimizing SC inventory to ensure SC efficiency and increase customer satisfaction. The goal of inventory optimization is to determine a replenishment policy (i.e., when and how much to order) that can achieve the best SC efficiency with balanced inventory costs and service levels.

To help address the difficult challenge of SC inventory optimization, there is a large body of academic research and excellent textbooks on inventory management ([1], [2], and references therein). In these classical inventory management theories [3], [4], to facilitate analytical tractability, simplifying assumptions, such as pre-defined analytically solvable demand distribution, lead time distribution, stationary demand pattern over time, and fixed SC structure, are usually necessary to derive managerial insights.

However, these assumptions are easily violated in practice. For example, in many situations, the periodical demand curve will not follow a stationary distribution, especially when different strategies (e.g., price discount) are often used to stimulate demand. Therefore, it is difficult to express performance measures such as the total expected inventory cost and service level in closed-form mathematically.

To solve analytically intractable real world SC inventory optimization problem, researchers and practitioners turned to simulation-based optimization approaches decades ago. In a typical simulation-based optimization method, a simulation model is used to estimate the relationship between SC configurations and performance measures. An optimization procedure is developed on top of the simulation model to find the optimal set of decision variables in terms of a predefined objective function. Meta-heuristic search algorithms such as genetic algorithm (GA) have been popularly used to determine the optimal SC configurations.

In this paper, we propose two Bayesian optimization methods as a Monte Carlo optimization approach to determine the optimal SC configuration and evaluate its efficiency through computational experiments. The rest of the paper proceeds as follows. In Sect. 2, we review relevant literature on simulation-based optimization approaches in inventory control. In Sect. 3, we first formulate the multi-echelon inventory optimization problem. We then introduce the proposed Monte Carlo optimization approach that consists of an agent-based SC simulator and the Bayesian optimization methods. In Sect. 4, we compare the performance of the proposed methods with the most widely used meta-heuristic method, a genetic algorithm (GA) through numerical experiments. Finally, we conclude the paper in Sect. 5.

2. Literature Review

In this chapter, we survey literature on the simulation-based optimization approaches developed and adopted in inventory optimization.

Simulation-based optimization methods can be clas-

Manuscript received September 7, 2021.
Manuscript revised January 16, 2022.
Manuscript publicized February 25, 2022.

[†]The author is with Production System Research Dept., Hitachi, Ltd., Yokohama-shi, 244-0817 Japan.

[††]The authors are with School of Knowledge Science, Japan Advanced Institute of Science and Technology (JAIST), Nomi-shi, 923-1292, Japan.

[†††]The authors are with Industrial AI laboratory, Hitachi America, Ltd., 2535, Augustine Dr, 3rd Floor, Santa Clara, CA 95054, USA.

a) E-mail: takahiro.ogura.cu@hitachi.com

DOI: 10.1587/transfun.2021EAP1110

sified into four categories [5]: 1) gradient-based methods that often utilize a gradient-estimation technique, such as infinitesimal perturbation analysis and finite difference estimation; 2) meta-model-based methods that use meta-models, such as response surface methodology and artificial neural networks, to approximate the objective function; 3) statistical methods, such as multiple comparison procedure; and 4) meta-heuristic methods, including GAs, simulated annealing, and particle swarm optimization. Jalali [6] uses a slightly different terminology. However, the resulting taxonomy is consistent with that of Abo-Hamad and Arisha's [5]. It is noteworthy that both survey papers consider the meta-heuristics methods, especially GA, as the most common optimization approach.

In most simulation-based inventory-optimization research, simulators are usually developed to solve the specific problem under consideration and are not flexible or comprehensive enough to introduce new dynamics such as SC structure changes mandated by the highly dynamic global market and increasing competition.

Recently, the concept of digital SC twins (DSCTs) has gained attention and holds promise to revolutionize supply chain management (SCM) thanks to the advancement of Internet-of-things technology and machine learning algorithms [7]. A DSCT is a virtual model of the physical SC that includes a digital counterpart of every step of the process and simulates complex dynamics and characteristics in a real-world SC that cannot be fully captured by analytical model assumptions. By running simulations and what-if scenarios within the digital twin, a SC manager can make optimized proactive decisions to improve SC efficiency and reduce risks. For example, Kiuchi et al. [8] developed an agent-based SC simulator that has the flexibility to accommodate situations when the SC structure changes, when the SC managers change the inventory-policy structure (e.g., from a single-echelon to a multi-echelon inventory-control policy), when the market-demand pattern changes, etc. However, Kiuchi et al. [8] has proposed just simulation model and has not proposed simulation-based-optimization methods. In this paper, we use the simulator developed by Kiuchi et al. [8] as the DSCT in our proposed simulation-based optimization methods.

While meta-heuristic simulation-based optimization methods may be effective for solving analytically intractable real-world multi-echelon inventory problems, they are usually computationally expensive. This is because for each set of parameters, to evaluate the objective function (i.e., a certain SC performance measure such as expected total inventory cost), we need to run the SC simulator for many scenarios to account for the uncertainties. The objective function is often required to be evaluated under a large number of parameter settings, especially when the decision variables are continuous, if using the traditional meta-heuristic methods. Therefore, when the SC is large and involves multiple sources of stochasticity, the computational inefficiency may hinder the applicability of the DSCT to assist real-time critical decision making in SCM, although it has the capability to accurately replicate the real SC process.

On the other hand, a Bayesian optimization approach has been successfully applied to different areas such as hyperparameter tuning in machine learning algorithms ([9]–[11]) and industrial experimental design that requires expensive resources (materials, money, time, etc.) to evaluate each parameter setting [12].

Bayesian optimization is a powerful global optimization framework for optimizing expensive and black-box objective functions. Compared with other general optimization frameworks, Bayesian optimization often significantly reduces the number of function evaluations.

The fundamental idea of Bayesian optimization is to approximate the objective function by using a surrogate model that can not only accurately approximate the objective function but also be relatively and inexpensively evaluated at a sample parameter setting. Bayesian optimization relies on a surrogate model, hence, falls into the category of meta-model-based methods in accordance with the above taxonomies [5] and [6].

In terms of application of Bayesian optimization to SCM, Kiuchi et al. [13] proposed a Bayesian optimization framework for inventory control problem and verified that the methodology can get optimal solution faster than GA with serial 3-echelon SC model.

Therefore, this study is intended to extend the Bayesian optimization framework [13] to apply it to more general real-world scale of multi-echelon SC and to verify its effectiveness. In order to achieve the purpose, we introduce multi-echelon distribution SC model in the next chapter and formulate an inventory control problem with the SC model.

3. Proposed Method for Supply Chain Inventory Control

In multi-echelon inventory control, in addition to minimizing the total system cost, a certain service level is usually imposed. For example, it is often required that the cumulative demand-fulfill rate over a planning horizon must be greater than a pre-defined threshold. Note that, like the objective function on the inventory-related cost, the service level cannot be analytically expressed and has to be evaluated with the SC simulator. That is, we need to solve a constrained optimization problem where the constraint is similarly expensive to evaluate as the objective function.

The two proposed methods for handling such a constraint are as follows [13]. One involves introducing a high penalty cost whenever the parameter setting falls into an infeasible region (i.e., the constraint is violated). This is called penalty-based Bayesian optimization. The other method involves approximating the constraint function using another surrogate model such as a Gaussian process (GP) model. This method is called constraint Bayesian optimization.

3.1 Problem Setting and Formulation

We consider a finite planning horizon multi-echelon

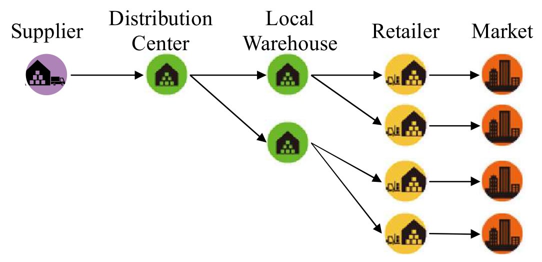

Fig. 1 Example of multi-echelon distribution supply chain (SC).

inventory-optimization problem under uncertainty. The uncertainty may come from the demand in each period during the planning horizon, order-fulfillment lead time from an upstream tier to the next, or both. To take advantage of the agent-based SC simulator, we do not need to specify any probability distribution for the uncertain factors upfront. Instead, we rely on a scenario generator that is independent of the agent-based SC simulator to account for the uncertainties, which significantly improves the flexibility of the simulator. Ideally, the scenario generator can learn from historical data then generate different scenarios that serve as input to the simulator to facilitate optimal and robust planning decisions.

We consider a typical tree structured distribution SC, as illustrated in Fig. 1. We assume that the Supplier has enough supply inventory against any order from the Distribution Center (DC) and solve for the inventory control problems at the DC, all Local Warehouses and Retailers. Also, we assume that there is no order-placement cost and focus on minimizing the total system-inventory-holding cost during the planning horizon while maintaining a minimum demand-fulfill rate at each retailer site.

In practice, a variety of inventory policies are used. For example, a continuous review (R, Q) policy replenishes inventory by ordering Q quantities whenever the inventory position falls below the reorder point R. A periodic review base-stock or order-up-to policy that replenishes the inventory position up to the base-stock level during each review period is also common because it is simple to understand and implement. For the base-stock policy, depending on whether the relevant information is centrally available, and whether the decisions can be centrally made, there are independent single-echelon and multi-echelon inventory policies. Note that the agent-based SC simulator we used [8] has the flexibility to model all these inventory policies.

For a given inventory policy, we need to optimize the corresponding parameters to minimize the total system-inventory cost while maintaining a minimum demand-fulfill rate. For instance, for the single-echelon base-stock policy considered in [9], people optimize the base-stock level at each local echelon. Since both the agent-based SC simulator and our proposed Bayesian optimization methods are not restricted to a particular policy, we use a generic vector $\boldsymbol{\alpha}$ to represent the parameters to be optimized in a policy.

Suppose the scenario generator generates N scenarios in total and let $\boldsymbol{s}_i$ denote scenario $i \in \{1, \ldots, N\}$. Let $Inv(\boldsymbol{\alpha}, \boldsymbol{s}_i), Fr_j(\boldsymbol{\alpha}, \boldsymbol{s}_i)$ be the total system-inventory cost and cumulative demand-fulfill rate at market site j when the inventory policy parameter is $\boldsymbol{\alpha}$ and the scenario is $\boldsymbol{s}_i$. Note that $Inv(\boldsymbol{\alpha}, \boldsymbol{s}_i)$ and $Fr_j(\boldsymbol{\alpha}, \boldsymbol{s}_i)$ are the output of the agent-based SC simulator when we set the input to the simulator as $\boldsymbol{\alpha}$ and $\boldsymbol{s}_i$. Let β be the minimum demand-fulfill rate required at each j. We take the commonly used Monte Carlo approach to estimate the expected inventory cost under uncertainty. The inventory-optimization problem for a given policy is then formulated as

$$\min. \ \frac{1}{N} \sum_{i=1}^{N} Inv(\boldsymbol{\alpha}, \boldsymbol{s}_i) \tag{1}$$

$$s.t. \ \min_{i,j} Fr_j(\boldsymbol{\alpha}, \boldsymbol{s}_i) \geq \beta \tag{2}$$

Constraint (2) is to ensure that the demand-fulfill rate at any market site is not lower than β for any scenario. That is, we aim to maintain a robust service level at all the retailer sites.

The formulation in (1) and (2) show that for a given $\boldsymbol{\alpha}$, the simulator needs to run N times to evaluate the corresponding objective function value and constraint feasibility, which is computationally expensive. When the parameters are in a continuous multi-dimensional space, naive search methods, such as grid search or random search, will be ineffective. To increase optimization efficiency, evolutionary algorithms, such as GAs, have been commonly used in solving practical inventory-optimization problems via simulation ([5], [6], [14]). A GA searches the parameter space by mimicking the natural-selection process where the fittest individuals are selected for reproduction to produce offspring of the next generation. While a GA is used to reduce the number of search iterations by selecting the fittest candidates in each iteration, Bayesian optimization approximates the objective function by using a surrogate model using a few sample points then proposes the most promising next candidate for evaluation by optimizing the surrogate model. Hence, it has the potential to reduce the number of function evaluations. Our experimental results in Chapter 4 indicate that one of the proposed Bayesian optimization methods outperform a GA in both effectiveness (i.e., optimality of the solution) and efficiency (i.e., computational time).

3.2 Bayesian Optimization for Multi-Echelon Inventory Optimization

Bayesian optimization is a framework to solve the optimization problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{3}$$

where $\boldsymbol{x}$ is a vector and $f(\boldsymbol{x})$ is the objective function that is expensive to evaluate.

Bayesian optimization uses a surrogate model that can be evaluated relatively inexpensively to approximate the objective function $f(\boldsymbol{x})$. A common surrogate model for Bayesian optimization is a Gaussian Process (GP). A GP is an extension of the multivariate Gaussian distribution to an infinite-dimensional stochastic process for which any finite sub-collection of random variables has a multivariate Gaussian distribution. For a GP model with a prior distribution of the objective function, denoted as $\tilde{f}(\boldsymbol{x}) \sim GP(\mu(\cdot), \sum(\cdot,\cdot))$, given a set of evaluated function values $D_f = \{(\boldsymbol{x}_1, f(\boldsymbol{x}_1)), \ldots, (\boldsymbol{x_t}, f(\boldsymbol{x_t}))\}$, we can update the posterior belief on the function using linear algebra thanks to the nice properties of the Gaussian distribution. Details on a GP and its use for machine learning can be found in a previous study [15].

At each iteration, the updated GP model $\tilde{f}(\boldsymbol{x})$ is used to select the next most promising candidate $\boldsymbol{x}^*$ in the search space to be evaluated by the expensive function $f(\boldsymbol{x})$. Subsequently, the new data pair $(\boldsymbol{x}^*, \boldsymbol{f}(\boldsymbol{x}^*))$ is added to the known data set D_f and the GP posterior belief is updated. This iterative procedure continues until a stopping criterion is met.

The selection of the next most promising candidate is done via an acquisition function. There are a few common acquisition functions such as maximum probability of improvement, expected improvement (EI), and upper confidence bound [11]. These acquisition functions are used to attempt to strike a balance between exploitation and exploration: exploitation means sampling where the surrogate model predicts a good objective value and exploration means sampling at locations where the prediction uncertainty is high. Among these acquisition functions, EI is the most widely used [16]; we briefly introduce it as follows.

Let $\hat{\boldsymbol{x}}$ be a candidate point and let $\tilde{f}(\hat{\boldsymbol{x}})$ be the GP posterior distribution for $f(\hat{\boldsymbol{x}})$. Let X_f be the set of data points that have been evaluated and $\boldsymbol{x}^+$ be the best point thus far,

$$\boldsymbol{x}^+ = \underset{\boldsymbol{x} \in X_f}{\operatorname{argmin}} f(\boldsymbol{x}) \tag{4}$$

Then the improvement of $\hat{\boldsymbol{x}}$ is defined as the potential decrease of $f(\hat{\boldsymbol{x}})$ against $f(\boldsymbol{x}^+)$, which is a random variable [9], [10]:

$$\tilde{I}(\hat{\boldsymbol{x}}) = \max\left\{0, f\left(\boldsymbol{x}^+\right) - \tilde{f}(\hat{\boldsymbol{x}})\right\} \tag{5}$$

Then the expected improvement acquisition function is the expectation of the random improvement:

$$EI(\hat{\boldsymbol{x}}) = \mathrm{E}\left[\tilde{I}(\hat{\boldsymbol{x}}) \,|\hat{\boldsymbol{x}}\right]. \tag{6}$$

Jones et al. [16] derived an easy-to-compute closed form for the EI acquisition function:

$$EI(\hat{\boldsymbol{x}}) = \begin{cases} (\mu(\hat{\boldsymbol{x}}) - f(\boldsymbol{x}^+))\,\Phi(Z) + \sigma(\hat{\boldsymbol{x}})\,\phi(Z), & \text{if } \sigma(\hat{\boldsymbol{x}}) > 0 \\ 0, & \text{if } \sigma(\hat{\boldsymbol{x}}) = 0 \end{cases} \tag{7}$$

where

$$\tilde{f}(\hat{\boldsymbol{x}}) \sim N\left(\mu(\hat{\boldsymbol{x}}), \sigma^2(\hat{\boldsymbol{x}})\right), \tag{8}$$

$$Z = \frac{\mu(\hat{\boldsymbol{x}}) - f(\boldsymbol{x}^+)}{\sigma(\hat{\boldsymbol{x}})} \tag{9}$$

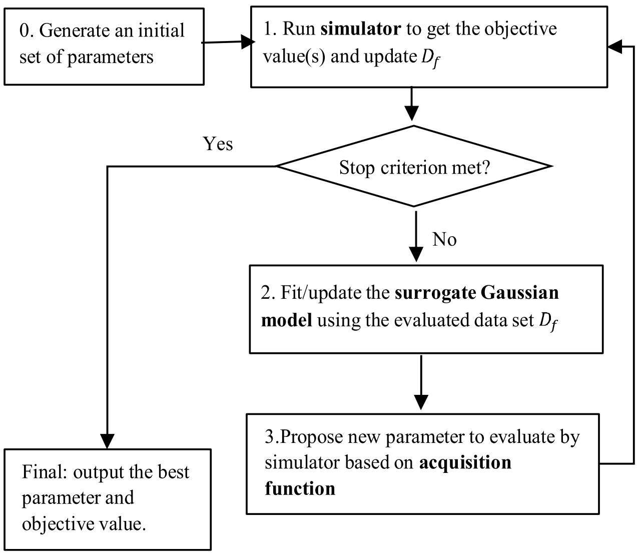
0. Generate an initial set of parameters
1. Run **simulator** to get the objective value(s) and update D_f
Stop criterion met?
Yes
No
2. Fit/update the **surrogate Gaussian model** using the evaluated data set D_f
3.Propose new parameter to evaluate by simulator based on **acquisition function**
Final: output the best parameter and objective value.

Fig. 2 Bayesian optimization framework.

and Φ and ϕ are the distribution function and density function of the standard normal distribution, respectively.

We then select the set of parameters that maximizes the above EI acquisition function as the next most promising candidate. To find the maximum of EI, one may apply any general-purpose non-linear optimization methods such as various quasi-Newton methods, conjugate gradient methods [18]. We used L-BFGS-B in our experiments, which belongs to the family of quasi-Newton methods and is efficient and popularly used.

The overall procedure of the general Bayesian optimization framework is summarized in Fig. 2. A commonly used stopping criterion is to run a certain number of iterations.

3.3 Bayesian Optimization with Constraints

In our problem setting, let $f(\boldsymbol{\alpha}) = \frac{1}{N}\sum_{i=1}^{N} Inv(\boldsymbol{\alpha}, \boldsymbol{s}_i)$, and $g(\boldsymbol{\alpha}) = \min_{i,j} Fr_j(\boldsymbol{\alpha}, \boldsymbol{s}_i)$, then the formulation of (1) and (2) becomes the following problem

$$\min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}) \tag{10}$$

$$s.t.\ g(\boldsymbol{\alpha}) \geq \beta \tag{11}$$

That is, in addition to expensive evaluations of the objective function $f(\boldsymbol{\alpha})$, we have an additional constraint function $g(\boldsymbol{\alpha})$. Note that both $f(\boldsymbol{\alpha})$ and $g(\boldsymbol{\alpha})$ can be evaluated in the same run of the simulator [8].

One of our proposed methods for dealing with the constrained optimization problem is to impose a large penalty to the objective function values in the parameter region where the constraint is violated, then applying the general Bayesian optimization framework in Fig. 2. We call this the penalty-based Bayesian optimization method (PBO). More specifically, let M be a large number and define a new objective function $h(\boldsymbol{\alpha})$ as

$$h(\boldsymbol{\alpha}) = \begin{cases} f(\boldsymbol{\alpha}), & \text{if } g(\boldsymbol{\alpha}) \geq \beta \\ f(\boldsymbol{\alpha}) + M, & \text{if } g(\boldsymbol{\alpha}) < \beta \end{cases} \tag{12}$$

Note that M should be selected large enough relative to the optimal value of the objective function. One practical method may be that we first estimate the upper bound of the optimal value and then multiply this upper bound by a power of 10 factor (e.g., 1e2). Note that, any feasible solution can serve as an upper bound. Hence, we can come up with an upper bound by setting a high inventory level at each location. (In our experiment, we used $M = 1e7$.)

Then, we apply the Bayesian optimization framework to the following optimization problem

$$\min_{\boldsymbol{\alpha}} h(\boldsymbol{\alpha}) \tag{13}$$

Our second method is motivated by the effectiveness of using a GP to approximate the objective function in Bayesian optimization. We approximate the constraint function using another GP then adapt the acquisition function to handle the constraint, as proposed Gardner et al. [13].

Suppose we use another GP model $\tilde{g}(\boldsymbol{x}) \sim GP\left(\mu_g(\cdot), \Sigma_g(\cdot,\cdot)\right)$ to approximate $g(\boldsymbol{x})$. During the optimization procedure, we also maintain a set of evaluated values for the constraint function $D_g = \{(\boldsymbol{x}_1, g(\boldsymbol{x}_1)), \ldots, (\boldsymbol{x_t}, g(\boldsymbol{x_t}))\}$ to update the posterior of $\tilde{g}(\boldsymbol{x})$, in addition to the set of values for the objective function D_f.

The expected improvement acquisition function is then adapted in the following two ways. First, the best point thus far $\boldsymbol{x}^+$ is defined as the best in D_f that is feasible. Second, a constrained improvement function for an $\hat{\boldsymbol{x}}$ is defined as follows:

$$\begin{aligned} \tilde{I}_C(\hat{\boldsymbol{x}}) &= 1_{\{\tilde{g}(\hat{x}) \geq \beta\}} \max\left\{0, f\left(\boldsymbol{x}^+\right) - \tilde{f}(\hat{\boldsymbol{x}})\right\} \\ &= 1_{\{\tilde{g}(\hat{x}) \geq \beta\}} \tilde{I}(\hat{\boldsymbol{x}}), \end{aligned} \tag{14}$$

where $1_{\{g(\hat{x}) \geq \beta\}} = 1$ if $g(\hat{\boldsymbol{x}}) \geq \beta$ and 0 otherwise is the indicator function of feasibility. That is, with the constrained improvement function, zero improvement is assigned if the candidate point $\hat{\boldsymbol{x}}$ is infeasible. The next promising candidate is then selected based on the expected constrained improvement

$$\begin{aligned} E\tilde{I}_C(\hat{\boldsymbol{x}}) &= E\left[\tilde{I}_C(\hat{\boldsymbol{x}})\,|\hat{\boldsymbol{x}}\right] \\ &= E\left[1_{\{\tilde{g}(\hat{x}) \geq \beta\}} \tilde{I}(\hat{\boldsymbol{x}})\,|\hat{\boldsymbol{x}}\right] \\ &= E\left[1_{\{\tilde{g}(\hat{x}) \geq \beta\}} |\hat{\boldsymbol{x}}\right] \mathrm{E}\left[\tilde{I}(\hat{\boldsymbol{x}})\,|\hat{\boldsymbol{x}}\right] \\ &= Pr\left(\tilde{g}(\hat{\boldsymbol{x}}) \geq \beta |\hat{\boldsymbol{x}}\right) EI(\hat{\boldsymbol{x}}). \end{aligned} \tag{15}$$

Note that, $Pr(\tilde{g}(\hat{\boldsymbol{x}}) \geq \beta|\hat{\boldsymbol{x}})$ is a univariate Gaussian complementary cumulative distribution function that can be analytically calculated due the marginal Gaussianity of the GP.

As Gardner et al. pointed out [13], while infeasible points are never considered the best points, they are still useful to add to the already evaluated data sets D_f and D_g to improve the GP posteriors to help estimate the shape of $f(\boldsymbol{\alpha})$ and $g(\boldsymbol{\alpha})$. This is because both of them are smooth functions and more data points will improve the estimation of the function shape. We call this second method the constrained Bayesian optimization method (CBO).

3.4 Agent-Based SC Simulator

In this section, we introduce our agent-based simulator [8] that gives us the flexibility to model complex dynamics in a SC. In a SC system, there are physical, capital, and information flows. Physical and capital flows are triggered by information flows. For instance, the ordering-information flow in inventory control moves products and/or cash along the chain. Information flows are determined by the status of physical objects and cash at relevant sites in accordance with certain rules. We define each decision maker that executes such rules to determine the information flow as an agent. Each agent is specialized in accordance with its intended role in the SC (procurement agent, inventory-control agent, etc.) and consumes information relevant to its role to take action. The behavior of individual agents, as well as the interactions between them, render the behavior of the whole SC.

Figure 3 shows a sample model at a warehouse site to illustrate how different agents coordinate their activities in one business-process instance. The various symbols for nodes and arcs are listed in Table 1 and Table 2, respectively. We use those nodes and arcs to represent the relationship between information, product, and cash flows in the agent-based SC simulator. Let T be the simulation days and K be a set of customer sites to be fulfilled by this warehouse. On each day $t \in T$, the shipping-order agent issues shipping instruction SI_i^t, where $i \in K$, for each received order O_i^t and calculates the backlog quantity $O_i^t - SI_i^t$ if there is a shortage in the stock. Next, the demand-forecast agent works out a demand-forecast plan $DF_{i\bar{t}}^t$, where DF is the demand–forecast value on each date $\bar{t}(> t)$ to destination site i using its logic. Then the inventory-control agent calculates the target inventory level TI^t considering $DF_{i\bar{t}}^t$. Next, the procurement-plan agent refers to the target inventory level, stock level, backlog quantities, as well as the lead time to come up with a procurement plan that specifies when and

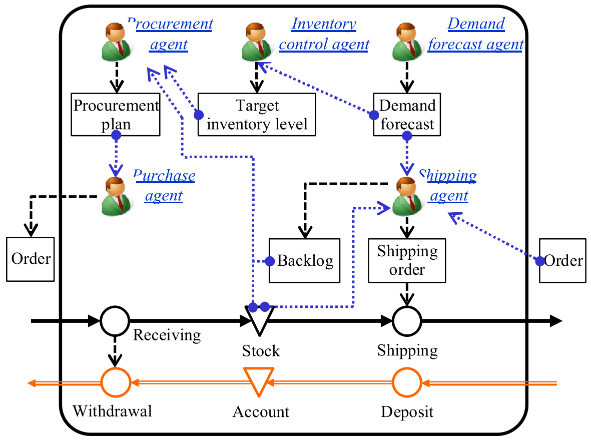

Fig. 3 Example of warehouse site model.

Table 1 Node types

Node Type	Object	Symbol
Physical Objects	Receiving, Production, Transportation, Issuing	○
	Stock	▽
Cash	Deposit, Withdrawal, Payment	○
	Savings (Account)	▽
Information	Decision maker (Agent)	👤
	Data	□

Table 2 Arcs and their symbols.

Arc	Symbol
Physical flow	⟶
Cash flow	⟶
Information flow: reference	●·····>
Information flow: update	---->

how much the warehouse site should procure. Finally, the purchase-order agent executes the procurement plan and issues an order for the warehouse.

To represent a real-world SC, the agents should be able to mimic the decision-making rules adopted either by the personnel or local decision-support systems in the physical world. This is accomplished by defining a set of properties for each agent including decision logic, working priority, and working cycle. The ability to model individual styles and attributes gives us the flexibility to replicate any complex dynamics in the real SC.

4. Computational Experiments

We applied the proposed Bayesian optimization methods to the inventory-optimization problem defined in Eq. (10) and Eq. (11), and compare the performance with widely used baselines, GA.

The considered SC structures, environment, and inventory policies as well as the corresponding parameters to be optimized are presented in Sect. 4.1.

In our experiments, we considered multi-period uncertain non-stationary demand. A scenario generator as described in Sect. 4.2 was used to generate different demand scenarios. Note that besides the optimization algorithms, how well the scenario generator can predict the real demand also affects the optimality of the output solution. To remove the impact of the distributional gap between the demand scenarios and actual demand on performance and focus on the impact of optimization algorithms, we assumed that the underlying stochastic process of the multi-period demand is known.

Section 4.3 briefly discusses the implementation of our two proposed Bayesian optimization methods. Section 4.4 describes the employment of our benchmarking method GA and the corresponding settings. Section 4.5 presents the experimental results. All experiments were run on a Windows server with Intel Xeon E3-1230 v5 3.4 GHz 4 cores 8 threads CPU and 64.0 GB RAM.

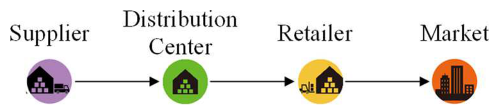

Fig. 4 Experimental simple SC model.

4.1 SC Structures and Variable Settings

We started with a simpler SC structure as shown in Fig. 4 in our experiments. The purpose is twofold: 1) we first evaluate the performances of our proposed methods in solving relatively simpler inventory control problems; 2) we use the simpler SC structure setting to select some hyper parameters of our experiments. For instance, we selected L-BFGS-B over other optimization methods to propose the next most promising candidate based on our initial experiment with the simpler SC structure. We also tested and tuned the parameter of our baseline GA algorithms using the simpler SC structure.

In the simple and complex SC structures shown respectively in Fig. 4 and Fig. 1, the distribution center orders from the supplier, and local warehouse orders from the distribution center to fulfill the demand from the retailer. We assume a deterministic procurement lead time from an upper stage to a lower stage, which can be flexibly set in the agent-based SC simulator. In our experiments, we assumed the procurement lead time for both the distribution center and local warehouse is one week. Due to the uncertain demand and lead time, both the distribution center and local warehouse maintain inventories. We also assumed that the supplier has sufficient supply.

In most of the following experiments if not explicitly specified, we set $\beta = 0.95$, i.e., the demand fill rate must be greater than or equal 95% for all markets.

For inventory replenishment, we assumed weekly review cycle. We assume downstream inventory and market demand information are available to upstream stages and hence use multi-echelon inventory policies for both SC models to achieve higher SC efficiency. In the multi-echelon base-stock policy, inventory decisions are made at the upper stages based on echelon demand and echelon inventory-position information [1]. We implemented the policies to the agent-based SC simulator based on inventory theory derived from stationary stochastic demand due to the elegance of the formula. That is, the base stock is calculated by the average demand during the lead time plus safety stock. Safety stock is calculated by multiplying the standard deviation of lead time demand with a safety stock coefficient. Let I_p denote the base stock inventory-position at a site implemented in our simulator, then

$$I_p = \mu LT + \alpha\sqrt{LT}\sigma \tag{16}$$

where μ, σ are the average demand quantity and standard deviation of all the markets related to each site, respectively.

LT is procurement lead time in each site. α is the safety-stock coefficient, the variable to be optimized in the inventory control problem.

The safety-stock coefficients at the distribution centers, local warehouses, and retailers $\boldsymbol{\alpha}$, were the parameters to optimize in our experiments.

The goal was to select the optimal safety-stock coefficients at the distribution center, local warehouses, and retailers to minimize the total SC-inventory cost while maintaining a minimum cumulative demand-fulfill rate at all the market sites. We assumed that the unit inventory-holding cost rate is the same at each site.

4.2 Demand-Scenario Generation

We assumed the planning horizon is 30 weeks and the weekly demand follows a GP $X(t) \sim 4000 \times \text{Gaussian}(\mu(t), \Sigma(t,t'))$ and for $t \in [0,30]$, the mean function is

$$\mu(t) = 10 + \sum_{p=1}^{4} \gamma_p \phi_p(t) \tag{17}$$

with $\phi_p(t) = \sqrt{2}\sin(2p\pi t/30)$ for $p = 1,\ldots,4$, $\boldsymbol{\gamma} = [\gamma_1,\ldots,\gamma_4]^{\boldsymbol{T}} = [2.133, 1.8, -1.067, -0.267]^{\boldsymbol{T}}$, and the covariance function is

$$\Sigma(t,t') = \sum_{p=1}^{4} \lambda_p \phi_p(t)\phi_p(t') \tag{18}$$

where $\boldsymbol{\lambda} = [\lambda_1,\ldots,\lambda_4]^{\boldsymbol{T}} = [0.5, 0.275, 0.125, 0.05]^{\boldsymbol{T}}$.

The scenario generator uses the defined GP to generate N demand scenarios for each market. To make the demand bounded, we added a cap value at 60,000. For a given market j, the demand scenarios $\{s_{i,j}\}_{i=1}^{N}$ were a set of 30-dimensional random vectors achieved by drawing i.i.d random samples from the specified GP at 30 equally spaced times. Figure 5 shows five simulated scenarios for market $j = 1$ as an example. Demand scenarios $\boldsymbol{s_i}$ consists of generated demands from all markets. Each $\boldsymbol{s_i}$ represents a projected actual demand across markets and serves as an input to the simulator. As we discussed in Sect. 3, for a given parameter vector $\boldsymbol{\alpha}$, the expected inventory cost is estimated as $\frac{1}{N}\sum_{i=1}^{N} Inv(\boldsymbol{\alpha}, \boldsymbol{s_i})$. That is, for each parameter setting, the simulator will run N times to evaluate the corresponding inventory cost.

4.3 Implementation of Proposed Bayesian Optimization Methods

To implement our PBO, we used a well-developed R package called 'rBayesianOptimization' to solve the transformed optimization problem (Eq. (12) and Eq. (13)). We modified the source code of 'rBayesianOptimization' by adding a module to conduct GP fitting for the constraint function to implement our CBO [13]. In our experiments, the number of initializations was 10 and the maximum number of iterations was 30; in total, 40 parameter combinations were evaluated with the agent-based SC simulator. For the kernel function used in the GP, we found that the default exponential function with a power of 2 worked well in our problem setting.

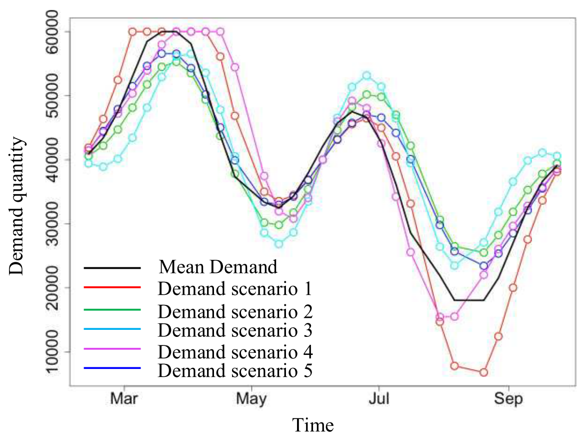

Fig. 5 Example of simulated demand.

For both PBO and CBO, we identify the optimal parameter setting as the one that produces the smallest inventory cost among all feasible solutions. This means that the solution is not necessarily obtained in the last iteration.

4.4 GA Method Settings

The general framework of the GA is designed for unconstrained optimization problems. To handle the demand-fulfill-rate constraint in our inventory-optimization problem (Eq. (10) and Eq. (11)), we use the penalty-based strategy we discussed in Sect. 3.3. That is, we use GA to solve the transformed unconstrained optimization problem in Eq. (12) and Eq. (13).

We used the R package 'GA' to implement the optimal parameter search procedure along with the simulator [8]. We set the number of iterations to 30. In each iteration, ten offspring (i.e., 10 combinations of parameters) were produced by selecting and manipulating the most promising candidates in the parent population. In total, 300 parameter combinations were evaluated with the simulator. The parameter combination at the last iteration is the final output, if it is a feasible solution. Otherwise, no valid solutions have been achieved.

4.5 Experimental Results

The experimental results of the simple and complex SC using the multi-echelon policy are summarized as follows.

1) Applying multi-echelon policy for simple SC (Fig. 4):

Table 3 and Table 4 show the achieved objective function value and corresponding computational time of CBO, PBO, and GA, under the simple SC and multi-echelon policy setting. The number of demand scenarios is N = 5, 20, 50. Large values of N mitigate the impact of randomness to a

Table 3 Objective function value (inventory cost) with simple SC (Fig. 4).

	Method			Ratio	
	CBO	PBO	GA	PBO/CBO	GA/CBO
N=5	19234.16	26103.74	20754.56	1.36	1.08
N=20	19075.14	26654.07	20913.82	1.40	1.10
N=50	19045.41	23529.71	20744.55	1.24	1.09

Table 4 Computational time (sec) comparison with simple SC (Fig. 4).

	Method			Ratio	
	CBO	PBO	GA	PBO/CBO	GA/CBO
N=5	180.00	180.40	1839.80	1.00	10.22
N=20	534.80	541.40	10596.00	1.01	19.81
N=50	1239.20	1234.40	28318.60	1.00	22.85

Table 5 Hyperparameter of GA.

	Objective function value (inventory cost)		
$p_{permutate}$	0.2	0.3	0.4
N=5	21192.36	20754.56	20361.40
N=20	20556.39	20913.82	20102.16
N=50	20856.63	20744.55	20085.85
	Computational time (sec)		
$p_{permutate}$	0.2	0.3	0.4
N=5	1871.00	1839.80	1956.40
N=20	10395.80	10596.00	11136.00
N=50	28494.20	28318.60	29105.80

larger degree, however, the associated running time is longer.

As shown in these Table 3 and Table 4, CBO outperformed PBO in terms of optimality while they have similar computational efficiency. This is because the Bayesian optimization method in PBO cannot approximate the non-smooth penalized cost function $h(\alpha)$ as effectively as it approximates the smooth functions $f(\alpha)$ and $g(\alpha)$ with CBO.

CBO also achieved smaller inventory costs than GA. From the computational efficiency point of view, CBO is after than PBO and GA. And the gain was even more significant when the number of scenarios increased. Remarkably when $N = 50$ case, CBO converged to optimal solution 22 times faster than GA. Note that we considered three different settings for the permutation probability in GA ($p_{permutate} = 0.2, 0.3, 0.4$). The results in Table 3 and Table 4 correspond to the hyperparameter that yielded the best results in terms of optimality. The detailed results are given in a Table 5. It can be seen that although the objective function value and the computational time vary slightly due to the randomness, they are in the same level of magnitude and will not change the conclusion. To save time, in the following experiments, we use $p_{permutate} = 0.3$.

Table 6 Objective function value (inventory cost) with complex SC (Fig. 1), β = 0.95.

	Method			Ratio
	CBO	PBO	GA	GA/CBO
N=5	32846.27	N/A	60295.33	1.84
N=20	32232.52	N/A	61729.16	1.92
N=50	39770.00	N/A	63439.20	1.60

Table 7 Computational time (sec) comparison with complex SC (Fig. 1), β = 0.95.

	Method			Ratio
	CBO	PBO	GA	GA/CBO
N=5	2240.60	N/A	3013.20	1.34
N=20	3494.80	N/A	12705.40	3.64
N=50	5512.60	N/A	31698.00	5.75

2) Applying multi-echelon policy for complex SC (Fig. 1):

Next, we conduct an experiment with the complex SC structure. Table 6 and Table 7 present the achieved objective function values and the computational time. It can be observed that GA can get optimal solutions relatively faster than the previous experiment. However, CBO can get around 1.7 less inventory cost than GA using less computational time.

In this experiment, PBO cannot find a solution in all patterns. To investigate why it is, we show PBO's initial 10 iterations results in Table 8. There are only 2 times PBO got a feasible solution within the initial 10 iterations. This results in a significantly non-smooth penalized cost function $h(\alpha)$, which raises challenges in fitting a valid GP and voting for the next promising candidates. Therefore, the R program generates an error and stops. Hence, it is difficult to find the optimized solution by PBO if the feasible area is small. On the contrary, CBO can solve the optimization problem in such situations by approximation $f(\alpha)$ and $g(\alpha)$ at the same time.

To further investigate whether PBO can produce comparable or even better results than CBO in terms of optimality, we reduced the fulfill rate requirement to 90%. As shown in the last column of Table 8, there are 4 feasible solutions during the initialization phase and PBO was able to solve the optimization problem. The corresponding results are given in Table 9 and Table 10 respectively. It can be seen that, in terms of optimality, CBO still significantly outperforms PBO. As for computational time, we observe that PBO is in general slightly faster than CBO. The gap between these two methods is smaller when there are more scenarios.

5. Conclusion

We proposed two Bayesian optimization methods that use the agent-based supply-chain simulator developed by Kiuchi et al. [8] as the DSCT tool to solve a constrained supply-chain inventory-control problem. Our experimental results

Table 8 Mid-term results of the penalty-based Bayesian optimization with complex SC (Fig. 1).

Round	α_1	α_2	α_3	α_4	α_6	α_6	α_7	Inventory cost	Fr_1	Fr_2	Fr_3	Fr_4	Feasibility ($\beta = 0.95$)	Feasibility ($\beta = 0.90$)
1	1.03	1.33	1.62	1.98	0.96	2.55	2.91	106328.79	92.36	97.47	96.31	95.07	Infeasible	Feasible
2	2.61	1.45	2.81	2.49	2.47	0.69	2.09	77576.81	98.53	98.81	98.57	96.58	Feasible	Feasible
3	1.43	2.10	1.76	2.44	2.20	1.74	2.91	106026.74	96.57	98.69	96.10	97.62	Feasible	Feasible
4	2.18	2.33	0.42	0.99	2.75	2.92	2.56	94207.50	97.80	95.60	93.30	97.98	Infeasible	Feasible
5	1.60	0.19	0.54	1.63	2.54	1.40	2.67	98087.44	96.86	96.56	94.09	89.90	Infeasible	Infeasible
6	2.57	2.17	0.04	1.38	1.15	0.46	0.99	42554.18	97.95	90.96	60.59	94.85	Infeasible	Infeasible
7	2.55	1.08	2.91	2.40	1.78	2.82	2.03	76041.75	98.38	98.58	98.80	91.45	Infeasible	Feasible
8	2.28	0.86	0.35	0.21	1.07	0.85	1.24	50317.83	97.73	89.17	90.86	85.62	Infeasible	Infeasible
9	0.57	0.33	2.22	0.55	0.10	1.74	1.84	71382.25	76.24	94.38	97.16	68.41	Infeasible	Infeasible
10	0.94	1.31	2.15	2.27	2.62	1.26	1.24	51292.39	95.76	92.06	88.44	96.35	Infeasible	Infeasible

Table 9 Objective function value (inventory cost) with complex SC (Fig. 1), β = 0.90.

	Method		Ratio
	CBO	PBO	PBO/CBO
N=5	21854.77	39129.95	1.79
N=20	21030.59	39046.79	1.86
N=50	21322.52	37776.95	1.77

Table 10 Computational time (sec) comparison with complex SC (Fig. 1), β = 0.90.

	Method		Ratio
	CBO	PBO	PBO/CBO
N=5	1009.40	884.40	0.88
N=20	2151.60	2014.00	0.94
N=50	4415.40	4248.60	0.96

indicate that our constrained Bayesian optimization (CBO) outperformed the most widely used optimization algorithm in supply-chain inventory control (i.e., a genetic algorithm (GA)) in terms of both optimality and computational efficiency. CBO achieved lower inventory costs than GA under all settings. And the computational time was 22 times faster than GA in the best case.

As future research, we will improve simulator efficiency and apply our proposed Bayesian optimization methods with the agent-based supply-chain simulator to solve supply-chain optimization problems that do not only for numerical parameters but also non-numerical parameters, such as inventory policy combination, at each site. There is a large amount of searching space if the problem contains non-numerical parameters, and more complex SC network; thus, more samples will be needed to have a good surrogate model approximation and problem formulation. At the same time, optimizing the surrogate model to propose the next most promising sample will also become more challenging. It will be valuable to explore the limit of CBO and the applicability of more advanced Bayesian optimization methods, such as Bayesian optimization with knowledge gradients [17], as the supply-chain optimization problem becomes more complex.

References

[1] P.H. Zipkin: Foundations of Inventory Management, McGraw-Hill Higher Education, 2000.

[2] E.L. Porteus: Foundations of Stochastic Inventory Theory, Stanford University Press, 2002.

[3] A. Gupta and C.D. Maranas, "Managing demand uncertainty in supply chain planning," Comput. Chem. Eng., vol.27, no.8-9, pp.1219–1227, 2003.

[4] A.-P. Hameri and A. Paatela, "Supply network dynamics as a source of new business," Int. J. Prod. Econ., vol.98, no.1, pp.41–55, 2005.

[5] W. Abo-Hamad and A. Arisha, "Simulation optimization methods in supply chain applications: A review," Irish Journal of Management, vol.30, pp.95–124, 2011.

[6] H. Jalali and I. Van Nieuwenhuyse, "Simulation optimization in inventory replenishment: A classification," IIE Trans., vol.47, no.11, pp.1217–1235, 2015.

[7] I. Dmitry, D. Alexandre, D. Ajay, and D. Boris, "Digital supply chain twins: Managing the ripple effect, resilience, and disruption risk by data-driven optimization, simulation, and visibility," Handbook of Ripple Effects in the Supply Chain, pp.209–332, 2019.

[8] A. Kiuchi, T. Ogura, T. Ishii, A. Tsujibe, T. Nomoto, and K. Taguchi, "Development of agent-based supply chain management simulator," Proc. Mechanical Engineering Congress, Japan 2016, S1420102, 2016.

[9] J.R. Gardner, M.J. Kusner, Z. Xu, K. Q, Weinberger, and J.P. Cunningham, "Bayesian optimization with inequality constraints," Proc. 31st International Conference on Machine Learning, 2014.

[10] J. Snoek, H. Larochelle, and R.P. Adams, "Practical Bayesian optimization of machine learning algorithms," NIPS, pp.2960–2968, 2012.

[11] E. Brochu, V.M. Cora, and N. De Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, arXiv preprint arXiv:1012.2599, 2010.

[12] J. Azimi, X. Fern, A. Fern, E. Burrows, F. Chaplen, Y. Fan, H. Jaio, and R. Shaller, "Myopic polices for budgeted optimization with constrained experiments," Proc. AAAI Conference on Artificial Intelligence, vol.24, no.1, pp.388–393, 2010.

[13] A. Kiuchi, H. Wang, T. Ogura, T. Nomoto, C. Gupta, T. Matsui, S. Serita, and C. Zhang, "Bayesian optimization algorithm with agent-based supply chain simulator for multi-echelon inventory management," IEEE 16th CASE, 2020.

[14] J.S.R. Daniel and C. Rajendran, "A simulation-based genetic algorithm for inventory optimization in a serial supply chain," Intl. Trans. Op. Res., vol.12, no.1, pp.101–127, 2005.

[15] C.E. Rasmussen and C.K.I. Williams, Gaussian Process for Machine Learning, MIT Press, 2006.

[16] D.R. Jones, M. Schonlau, and W.J. Welch, "Efficient global optimization of expensive black-box functions," J. Global Optim., vol.13, no.4, pp.455–492, 1998.

[17] J. Wu, M. Poloczek, A.G. Wilson, and P.I. Frazier, "Bayesian optimization with gradients," Advances in Neural Information Processing Systems, pp.5267–5278, 2017.

[18] J. Nocedal and S.J. Wright, Numerical Optimization, Springer, 1999.

Takahiro Ogura is Unit Leader at Supply Chain Management and Logistics research unit, R&D Group, Hitachi, Ltd. He received B.S. and M.S. degrees in Industrial and Management System Engineering from Waseda University in 2003 and 2005, respectively. His research interests include Human-Machine Teaming and Cyber Physical Supply Chain Management System.

Haiyan Wang is Principal Research Scientist at Industrial AI Laboratory, Hitachi America, Ltd. She received Ph.D. degree in Operations Management from Washington University in St. Louis. Her research interests include optimization, machine learning, reinforcement learning as well as the applications in industrial analytics solutions.

Qiyao Wang is Senior Research Scientist at Industrial AI Laboratory, Hitachi America, Ltd. She received Ph.D. degree in Statistics from University of Pittsburgh. Her research interests are time series data analysis, and deep learning techniques.

Atsuki Kiuchi is Researcher at Industrial AI Research Laboratory, Hitachi America, Ltd. He received B.S. and M.S. degrees in Information and Telecommunication Engineering from Tokai University in 2013 and 2015, respectively. His research interests include operation research, simulation-based optimization for supply chain management.

Chetan Gupta is Vice President at Industrial AI Laboratory, Hitachi America, Ltd. He received Ph.D. in Mathematics and M.S. in Mathematical Computer Science and Chemical Engineering from University of Illinois, Chicago, respectively. His research area is development of industrial solutions with artificial intelligence.

Naoshi Uchihira is Professor at School of Knowledge Science of Japan Advanced Institute of Science and Technology (JAIST). He received Dr. Eng. degrees in Information Science and Engineering from Tokyo Institute of Technology, and Ph.D. (Knowledge Science) from JAIST in 1997 and 2010, respectively. He worked at the corporate R&D center of Toshiba Corporation from 1982 to 2013. His current research interests include project management and digital innovation design.