Fast Simulation-Based Order Sequence Optimization Assisted by Pre-Trained Bayesian Recurrent Neural Network

Issei Suemitsu, Hanoz Kaiwan Bhamgara, Kei Utsugi, Jiro Hashizume, and Kiyoto Ito

***Abstract*—This paper presents a fast optimization method for the picking order sequence of automated order picking systems in logistics warehouses. In this order sequencing problem (OSP), the fulfillment sequence of the given picking order set is determined to optimize the performance measures such as makespan and deadlock occurrence. Simulation is generally necessary to evaluate these measures for complex automated systems. However, their order sequence cannot be optimized quickly due to the long calculation time. It may make the system productivity and flexibility lower than expected because its picking schedules cannot be updated frequently. We, therefore, propose a fast optimization method to solve these simulation-based OSPs by taking a pre-trained surrogate-assisted optimization approach. Firstly, we utilized a Bayesian recurrent neural network (BRNN) as a surrogate model to accurately learn the relationship between picking order sequence and performances. Secondly, we developed the surrogate-assisted optimization method based on simulated annealing (SA) and BRNN. Numerical experiments show that the surrogate model can evaluate about 10000 times faster than the simulation. The proposed method also obtains an optimized solution 8.9 times faster than simulation-based optimization by the original SA.**

***Index Terms*—Logistics, Planning, Scheduling and Coordination, Optimization and Optimal Control, Deep Learning Methods, Model Learning for Control.**

I. Introduction

SEQUENCE optimization problem (SOP) or sequencing problem is a combinatorial optimization problem (COP) to find an appropriate sequence of tasks to minimize or maximize the given objective function. SOP can be seen in various practical fields such as traveling salesman problem (TSP), vehicle routing problem (VRP), and other job scheduling problems. In logistics warehouses, SOP can take the form of the scheduling of order picking activity, where the largest workforce is allocated. For example, in the most common order picking system (picker-to-parts), order pickers physically walk into a warehouse to pick items [1], and the associated labor costs account for about 55% of the total fulfillment operation cost [2]. Recently, various robotic systems have been developed to automate order picking. Among these, automated storage and retrieval systems (AS/RS) [3], automated guided vehicle (AGV) picking systems [4], [5] and piece picking robots [6], [7] are commonly employed.

Manuscript received 1 March 2022; accepted 7 June 2022. Date of publication 23 June 2022; date of current version 5 July 2022. This letter was recommended for publication by Associate Editor J. Morrison and Editor J. Yi upon evaluation of the reviewers' comments. *(Corresponding author: Issei Suemitsu.)*

The authors are with the Center for Technology Innovation, Hitachi, Ltd. Research & Development Group, Kokubunji 185-8601 Tokyo, Japan (e-mail: issei.suemitsu.rj@hitachi.com; hanoz_kaiwan.bhamgara.dg@hitachi.com; kei.utsugi.nz@hitachi.com; jiro.hashizume.hu@hitachi.com; kiyoto.ito.kp@hitachi.com).

Digital Object Identifier 10.1109/LRA.2022.3185778

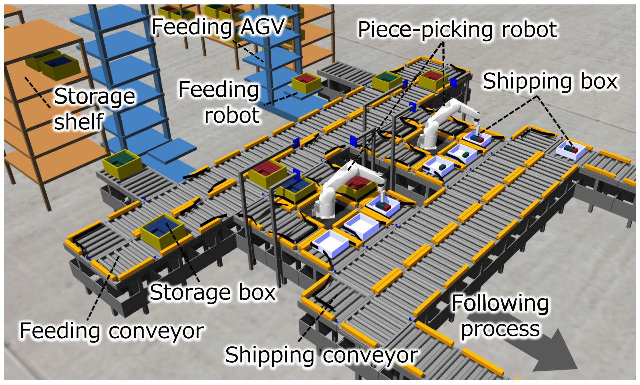

Fig. 1. Pick-Place-Carry-Pick-Place (PPCPP) order picking system.

In this paper, we consider the AGV picking system with robot arms and conveyor modules shown in Fig. 1, which is entitled Pick-Place-Carry-Pick-Place (PPCPP) system. Fig. 2 shows the detailed layout. The PPCPP system contains multiple piece-picking robots that can deal with various picking orders in parallel. When the warehouse management system (WMS) sends picking order information to the PPCPP system, the picking order is allocated to one of the available piece-picking robots. Then AGVs start transferring the movable storage shelves where ordered items are stored inside the storage boxes. The feeding robots *pick* the transferred storage boxes from the shelves and *place* them on the feeding conveyor. Then the feeding conveyor *carries* the storage boxes in front of the piece-picking robots. The piece-picking robots *pick* items from storage boxes on the feeding conveyor and *place* them into shipping boxes on the shipping conveyor. When a shipping box contains all the requested items, the shipping conveyor transfers the box to the following process, and a new picking order is immediately assigned to the piece-picking robot. PPCPP systems can achieve high scalability since the number of AGVs, robot arms, and other components can be customized easily.

For achieving high productivity, appropriate job scheduling is vital [8]. In the PPCPP system, the operation sequence of the picking orders from the WMS strongly affects the *makespan*,

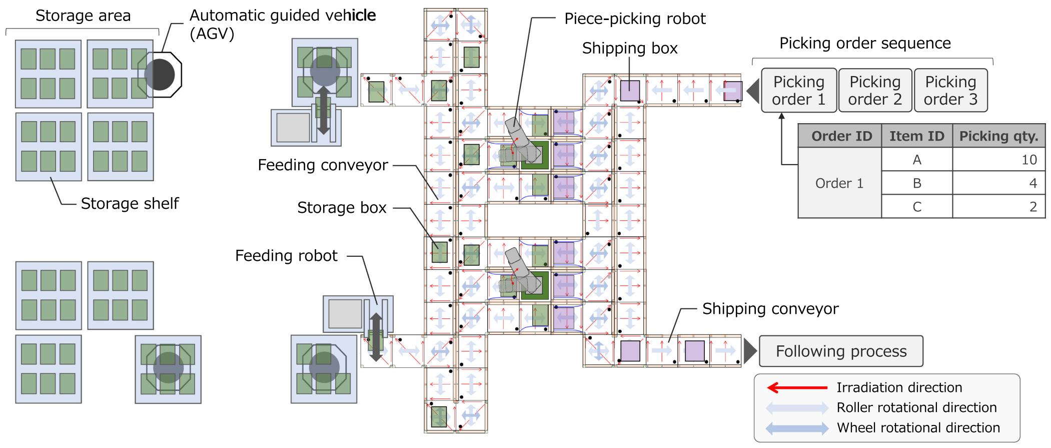

Fig. 2. PPCPP system layout. The system consists of AGVs, feeding robots, piece-picking robots, and conveyors. The system input (picking orders sequence) strongly affects the performances, such as the makespan and the deadlock occurrence.

defined as the duration from the initial order placement to the end of the final order shipping. Generally, the makespan can be reduced by continuously picking similar orders with overlapped items since transportation of these items' storages by AGVs and conveyors can be aggregated. The order sequence can also affect the occurrence of *deadlock*, which is a situation where no task can be achieved because of stacked system modules. An example situation is when inputting picking orders that require more kinds of items than the feeding conveyor capacity at the same time. All feeding conveyors can be occupied before completing these orders in this situation. Once such a situation happens, it increases the operation time due to system stoppage and is hard to restore automatically. Therefore, the order sequence that can lead to deadlock must be avoided.

In this paper, we focus on the order sequencing problem (OSP) of PPCPP systems that determines the optimal order sequence $\mathbf{X}$ of N picking orders $\mathbf{O} = \{O_n | n = 1, 2, \ldots, N\}$ for minimizing the makespan under the deadlock avoidance constraint. For example, if $N = 3$, the possible $\mathbf{X}$ is $\{O_1, O_2, O_3\}, \{O_1, O_3, O_2\}, \ldots, \{O_3, O_2, O_1\}$. Simulation is essential for solving the real SOPs (such as the OSP of PPCPP systems) because the performance measures are hard to formulate mathematically. Therefore, many researchers and practitioners have solved SOPs with simulation-based optimization (SBO) methods using metaheuristics [9]–[13]. However, these approaches are usually computationally expensive because they require iterative simulation runs, which generally take a few seconds to several minutes for one execution.

Surrogate-assisted optimization (SAO) is an alternative approach to solving computationally expensive SBO problems. Fig. 3 shows the flowchart of SAO, where the fundamental idea is to approximate the performance measures using a *surrogate model* or *metamodel* whose computational cost is relatively lower than simulation. Especially, Bayesian optimization (BO) is the most popular one, and several BO variants for binary optimization problems have been developed recently [14], [15]. However, it remains challenging for these methods to solve SOPs because enormous numbers of binary variables and constraints must be considered. Additionally, BO requires many simulations to train an accurate surrogate model *during optimization*. Thus, since conventional approaches cannot solve simulation-based SOPs in a short amount of time, flexible updating of picking schedules is impossible to achieve.

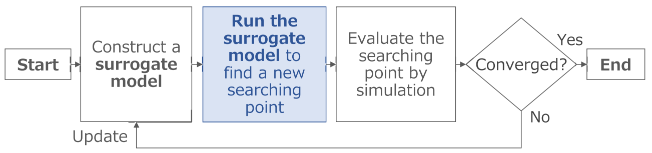
Start
Construct a surrogate model
Run the surrogate model to find a new searching point
Evaluate the searching point by simulation
Converged?
Yes
No
End
Update

Fig. 3. Flowchart of surrogate-assisted optimization. It can solve the simulation-based optimization problem efficiently by approximating simulation with a low-computational-cost surrogate model.

For quickly solving simulation-based SOPs, we have developed a new sequence optimization method that uses a surrogate model trained *before optimization (pre-trained)*. In practical OSPs, most picking system configurations cannot be changed frequently, and the distribution of future picking orders can be forecasted using historical data. Therefore it is possible to train the surrogate model with a dataset randomly generated from the forecasted distribution. We utilize a Bayesian recurrent neural network (BRNN) as a surrogate model to learn the sequential nature of OSPs. Our proposed method, BRNN-SA, combines simulated annealing (SA) as the basic optimization strategy and BRNN as the surrogate model in the optimization.

The purpose of this paper is to answer the following research questions.

- RQ1: Which surrogate model is most suitable for the OSPs of PPCPP systems?
- RQ2: How quickly can the proposed optimization method (BRNN-SA) solve the OSPs of PPCPP systems compared to conventional metaheuristics?

The contributions of this paper are as follows.

- Proposal of the Bayesian recurrent neural network-assisted simulated annealing (BRNN-SA) method.
- Investigation and selection of the proper surrogate model for the OSPs of PPCPP systems.
- Evaluation of BRNN-SA demonstrating its superiority in solving the OSPs of PPCPP systems compared with conventional metaheuristics.

In Section II, we explain the background of this study. Section III describes the BRNN-SA for OSPs. In Section IV, we evaluate the performance of the proposed BRNN-SA. We conclude in Section V with a brief summary and mention of future work.

II. Related Work

A. Order Sequencing Problem

The order sequencing problem (OSP) is a job scheduling process [16] that is categorized as a sequence optimization problem (SOP), which is a type of combinatorial optimization problem (COP). As OSP is a vital decision-making task for efficient picking operations in a logistics warehouse, many studies have investigated OSP for order fulfillment systems. Boysen *et al.* [17] formulated the OSP of the consolidation and packing area as a mixed-integer linear programming (MILP) model and solved it with a heuristic approach and dynamic programming. Pinto *et al.* [18] solved the order batching and sequencing problem by an iterative method of two Genetic algorithms (GA) that solve order batching and order sequencing separately.

In these OSPs and other ordinary job scheduling problems, the time needed for each operation and transportation is given to the planner as constant input values. However, in our problem setting, these values are difficult to determine mathematically since they can change depending on the AGV and conveyor status at a given time. In PPCPP systems, storage feeding for multiple picking orders can be aggregated in one AGV transportation, and it is nearly impossible to calculate the operation and transportation times in all possible situations. Therefore, instead of defining them as constant input values, we evaluate the makespan and the deadlock occurrence with the PPCPP simulator, whose input is an order sequence $\mathbf{X}$.

B. Simulation-Based Combinatorial Optimization

SOPs like those above have been solved as simulation-based optimization (SBO) problems, where simulation models evaluate the system performance under a given configuration. Metaheuristic methods are widely utilized in this context, especially for solving simulation-based combinatorial optimization problems (SBCOP). Rouky *et al.* [10] proposed an ant colony optimization (ACO) hybridized with a variable neighborhood descent local search to solve the task sequences of quay cranes. Uman *et al.* [11] tackled the flow shop scheduling problem and proposed the tabu search (TS) process with a genetic algorithm (GA) to minimize makespan. Another popular metaheuristic to solve SOPs is simulated annealing (SA). Cheng *et al.* [12] proposed an extended SA algorithm to minimize the makespan of a permutation flowshop scheduling problem (PFSSP). Khurshid *et al.* [13] also tackled PFSSP by using an improved evolution strategy that harnesses local search abilities of SA.

While these SBO methods may effectively solve analytically intractable SOPs, they are usually computationally expensive because iterative simulation runs (generally at least 100 or more) are essential to calculate the performance measures of each set of parameters. Realistic simulations like the PPCPP simulator take anywhere from a few seconds to several minutes for one execution, which limits the application of metaheuristics for OSPs to situations with enough calculation time.

TABLE I
Notations and Sets

Notation	Description
$\mathbf{X}$	Decision variables, which indicate an order sequence.
x_i	i-th picking order in order sequence $(i = 1, 2, .., N)$.
O_n	n-th customer's picking order $(n = 1, 2, .., N)$.
$F(\mathbf{X})$	Makespan to finish given order sequence $\mathbf{X}$, where F is implemented as a simulator.
$G(\mathbf{X})$	Deadlock occurence of order sequence $\mathbf{X}$ (1 if a deadlock occurs, 0 otherwise), where G is implemented as a simulator.
$f(\mathbf{X})$	Predicted makespan of order sequence $\mathbf{X}$ by surrogate model f.
$g(\mathbf{X})$	Predicted deadlock occurence of order sequence $\mathbf{X}$ by the surrogate model g (1 if a deadlock occurs, 0 otherwise).

C. Surrogate-Assisted Optimization

Surrogate-assisted optimization (SAO) is an alternative approach to solving computationally expensive SBO. The fundamental idea of SAO is to approximate the performance measures by a *surrogate model* whose computational cost is relatively lower than simulation. As surrogate models, linear regression (LR) [19], support vector machine (SVM) [20], gradient boosting trees (GBT) [21], artificial neural networks (NN) [22], and Gaussian process (GP) [23] are utilized. Bayesian optimization (BO), which uses GP as a surrogate model, is one of the most popular SAO methods. Although the original BO can solve only continuous optimization problems, more recent BO variants for solving SBCOP have been developed [24]. For example, COMBO (Oh *et al.* [14]) and MerCBO (Deshwal *et al.* [15]) are state-of-the-art BO methods to solve binary optimization problems.

However, it is still challenging to use these methods for solving simulation-based SOPs. SOP can be reformulated as a binary optimization problem, but the reformulated problem typically contains an enormous number of binary variables and constraints to eliminate overlaps or sub-tours. Generally, such problems are difficult to solve with metaheuristics and BO. Additionally, when solving a large-scale problem, BO requires many simulations to train the surrogate model accurately *during optimization*. Thus, conventional approaches cannot quickly solve simulation-based SOPs enough to update picking schedules flexibly. For reducing optimization runtime, we propose a new SAO approach that uses a surrogate model trained *before optimization*.

III. Methodology

A. Problem Setting and Formulation

We investigate the OSP of PPCPP systems, which determines the optimal sequence of the given N picking orders $\mathbf{O} = \{O_n | n = 1, 2, \ldots, N\}$ for minimizing the makespan without system deadlock. The makespan and deadlock occurrence are obtained as simulation results. The notations and decision variables are listed in Table I.

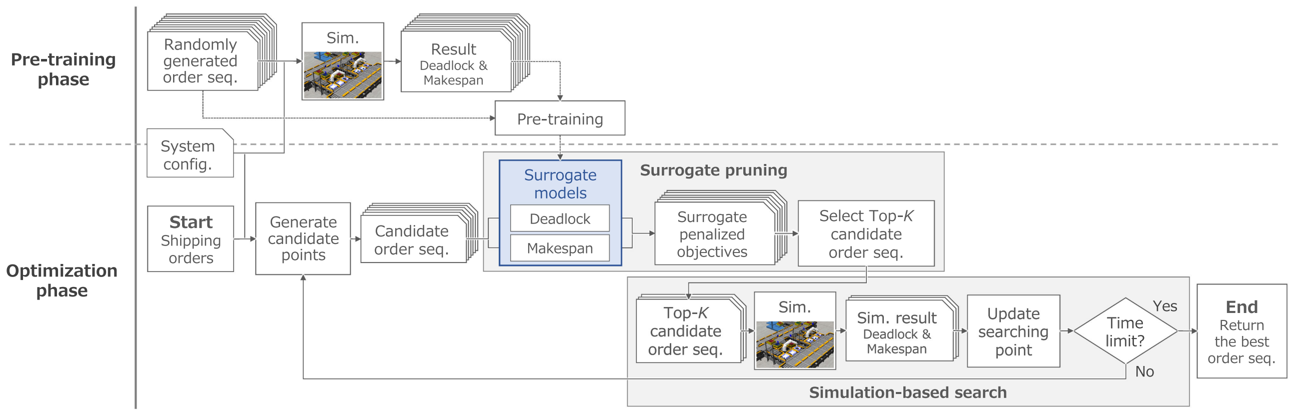
Pre-training phase
Randomly generated order seq.
Sim.
Result Deadlock & Makespan
Pre-training
System config.
Surrogate models
Deadlock
Makespan
Surrogate pruning
Start Shipping orders
Generate candidate points
Candidate order seq.
Surrogate penalized objectives
Select Top-K candidate order seq.
Optimization phase
Top-K candidate order seq.
Sim.
Sim. result Deadlock & Makespan
Update searching point
Time limit?
Yes
No
End Return the best order seq.
Simulation-based search

Fig. 4. Flowchart of surrogate pruning optimization framework. Surrogate models of the performance measures are trained in the pre-training phase and utilized to choose the top-K candidate solutions in the surrogate pruning of the optimization phase.

The mathematical formulation of the OSP is as follows:

$$\min_{\mathbf{X}} \quad F(\mathbf{X}) \tag{1}$$

$$\text{s.t.} \quad G(\mathbf{X}) = 0 \tag{2}$$

$$\mathbf{X} = \{x_1, x_2, \ldots, x_N\} \tag{3}$$

$$x_i \in \{O_1, O_2, \ldots, O_N\} \quad \forall i \tag{4}$$

$$x_i \neq x_j \quad \forall i \neq j \tag{5}$$

(1) shows the objective function $F(\mathbf{X})$, which is the minimization of the makespan to finish the given order sequence $\mathbf{X}$. (2) describes the constraint to avoid deadlock $G(\mathbf{X})$. (3), (4), and (5) ensure that the decision variable $\mathbf{X}$ is a non-overlapped sequence of each customer's picking order $\mathbf{O}$. As described in Section I, $F(\mathbf{X})$ and $G(\mathbf{X})$ are difficult to define as mathematical equations. We, therefore, calculate them with a PPCPP simulator.

In this paper, we investigate practical OSPs in logistics warehouses. In general, OSPs are solved with only the target OSP information, which is available just before solving the problem. In this case, the target problem information is the picking orders from customers (shipping orders) and the PPCPP system configuration. However, in real situations, we can utilize other information to solve the target OSP, such as the historical picking order information and the current system configuration. In this work, we assume the following two conditions;

- Rather than assuming that the picking orders from customers are only available just before the picking operation begins, we assume that the distribution of historical picking orders is always available.
- The system configurations do not change from the preparation to the picking operations because the configurations of practical industrial systems cannot be changed so frequently.

In the following sections, we discuss how to solve the OSPs of PPCPP systems under these assumptions.

B. *Surrogate Pruning Optimization Framework*

The overall flowchart of the proposed surrogate pruning optimization framework is shown in Fig. 4. This method consists of two phases: pre-training and optimization. In the pre-training phase, random picking orders are generated in accordance with the distribution obtained by the historical picking orders. Then, the surrogate models of the performance measures, namely, the makespan $f(\mathbf{X})$ and the deadlock $g(\mathbf{X})$, are trained to predict the simulated makespan $F(\mathbf{X})$ and deadlock $G(\mathbf{X})$ respectively. In the optimization phase, surrogate models are used to choose *good* candidate solutions quickly without simulations. The selected candidate solutions are then evaluated by simulation. This *surrogate pruning* enables us to reduce computational time to evaluate performance measures by simulation. These steps are repeated until the runtime reaches the time limit.

C. *Selection of Surrogate Models*

There are two requirements for the surrogate models for OSPs ($f(\mathbf{X})$ and $g(\mathbf{X})$) as follows.

Requirement 1: The surrogate models must find the appropriate features to infer the performance measures automatically since manual feature engineering of a complex system such as PPCPP systems is complicated.

Requirement 2: For optimizing the order sequence, the surrogate models must learn how the performance measures change depending on the order sequence.

To meet these requirements, we opted to use a multi-layer perceptron (MLP), which is a powerful machine learning method that automatically extracts features from input, which meets Requirement 1. For dealing with Requirement 2, we use a recurrent neural network (RNN), one of the MLPs that utilizes a recurrent layer to learn the sequential relationship in input features, such as order sequences. We also utilize a Bayesian neural network (BNN) to improve the model generalization. BNN is an MLP model that imposes a prior distribution on the weight parameters and aims to infer a posterior distribution instead of point estimates. Although the output of the PPCPP simulator is deterministic, we utilize BNN because it performs ensemble learning by marginalizing over this posterior for prediction.

Based on the above discussion, we select a Bayesian recurrent neural network (BRNN), which is a BNN consisting of a recurrent layer, for the surrogate models, as shown in Fig. 5. However,

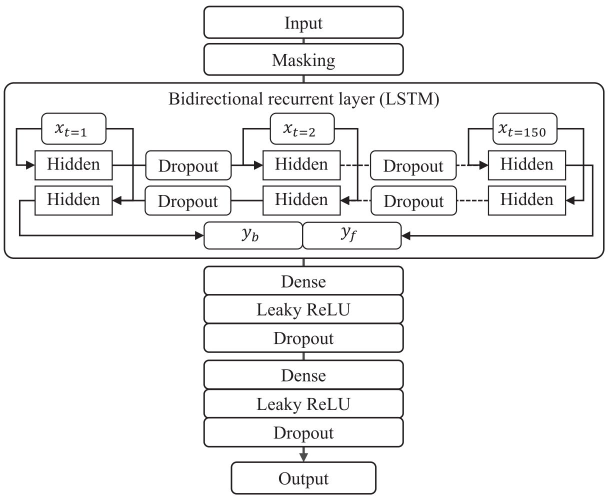

Fig. 5. The proposed architecture of the Bayesian recurrent neural network. It contains a bidirectional recurrent layer to learn the sequential relationship in input and Dropout layers for the Dropout variational inference.

precisely computing the posterior of non-linear BRNNs is infeasible, so we utilized the Dropout variational inference [25] to train BRNN. It interprets Dropout regularization as approximate inference in BRNN models and learns a variational Dropout posterior over the weight parameters. Prediction results of BRNN are obtained by averaging N_{pred} time inferences. In this work, we use $N_{pred} = 1$ for training and $N_{pred} = 10$ for prediction and optimization. In addition, we use the Adam optimizer [26] to train BRNN. The bidirectional long short-term memory (LSTM) architecture with 64 neurons (n_r) is utilized as the recurrent layer and connected to two dense layers, whose neurons are connected to all neurons in the next layer. Each layer consists of 192 neurons (n_d) with Leaky ReLU as the activation function and has a Dropout layer whose Dropout rate is set to 10% (r_d). All mentioned parameters ($n_r \in [32, 48, 64, 128]$, $n_d \in [64, 128, 192, 256]$, and $r_d \in [5\%, 10\%, 30\%]$) were analyzed and the most accurate model on the test set was picked.

D. Proposed Optimization Method: BRNN-SA

In this section, we propose BRNN-SA, which implements the surrogate pruning optimization by combining the simulated annealing (SA) strategy and BRNN as surrogate models. The detailed algorithm is shown in Algorithm 1.

This optimization procedure starts when the target shipping orders from customers are confirmed. Let $\mathbf{X}_t$ be the searching point in iteration t. First, we generate M_0 searching points randomly and select the initial searching point $\mathbf{X_1}$ that has the minimum surrogate penalized objectives $h(\mathbf{X}) = f(\mathbf{X}) + \omega g(\mathbf{X})$, where $f(\mathbf{X})$ and $g(\mathbf{X})$ are surrogate models and ω is the penalty parameter. Then the M candidate points are sampled uniformly at random from the *neighborhood* $\mathbf{S}(\mathbf{X}_t)$, which contains all points with a Hamming distance of at most one from $\mathbf{X}_t$. In the *surrogate pruning*, the top-K candidate points are chosen from M candidate points according to the surrogate penalized objectives $h(\mathbf{X})$. This *surrogate pruning* can reduce the computational time by choosing *good* candidate solutions quickly without time-consuming simulations.

For the selected K candidate points, the penalized objectives $H(\mathbf{X}) = F(\mathbf{X}) + \omega G(\mathbf{X})$ are evaluated by using their simulation results ($F(\mathbf{X})$ and $G(\mathbf{X})$). Then, the best candidate point $\hat{\mathbf{X}}_t$ is chosen from the K candidate points. If $\hat{\mathbf{X}}_t$ has a lower penalized objective than the best solution $\mathbf{X}_{\mathrm{best}}$, $\mathbf{X}_{\mathrm{best}}$ is replaced with $\hat{\mathbf{X}}_t$. The searching point is updated in the same way as SA. If $H(\hat{\mathbf{X}}_t)$ is smaller than $H(\mathbf{X}_t)$, let $\mathbf{X}_{t+1} \leftarrow \hat{\mathbf{X}}_t$. Otherwise, $\hat{\mathbf{X}}^t$ is adopted as $\mathbf{X}_{t+1}$ with the following probability P:

$$P = \exp\Big((H(\hat{\mathbf{X}}_t) - H(\mathbf{X}_t))/T\Big), \tag{6}$$

where T is the temperature parameter. T starts with a high T_0 to encourage exploration and then cools down by multiplying the cooling rate $\alpha \in [0, 1]$ to zoom in on a good solution.

These processes continue until the computational time reaches the time limit. Finally, the best searching point is returned as the solution order sequence. In this paper, we set $M_0 = M = 10000$, by which the surrogate runtime in a loop is almost equal to one simulation runtime, and $\omega = 10^6$ so that the deadlocked candidates always become worse than any candidates without deadlock. We also selected $K = 10$. In the preliminary experiments, we found that the optimization performance was excellent when $5 \leq K \leq 30$, and it became unstable when $K = 1$. It implies that selecting multiple candidates can improve the robustness of finding *good* solutions under prediction errors. Additionally, we selected $T_0 = 30$ and $\alpha = 0.95$ that showed the best optimization performance according to the preliminary experiments with the following parameter ranges; $T_0 \in [10, 30, 100]$ and $\alpha \in [0.9, 0.95, 0.99]$.

IV. Evaluation

We conducted numerical experiments to investigate RQ1 and RQ2 and compared the proposed optimization method with conventional metaheuristic methods.

A. Conditions

In our experiments, the sequence of the given picking orders is optimized for minimizing the makespan without the deadlock of the PPCPP system (Refer back to Fig. 2 for the system configuration.) All picking orders used in the experiments are generated randomly. One picking order set consists of 1 to 30 picking orders. Each picking order contains 1 to 5 items and the shipping quantity of each item is 1 to 700. The pre-training dataset is the PPCPP simulation results of 50,000 randomly generated picking order sets whose order sequence s are randomly sorted.

We formulate the deadlock occurrence prediction as a binary classification problem and the makespan prediction as a regression problem. In the following experiments, BRNN shown in Fig. 5 is compared with linear regression (LR), Gaussian process regression (GP), gradient boosting trees (GBT), and the Bayesian neural network without recurrent layer (BNN). Each picking order set is converted to the input features shown in Fig. 6. BRNN uses the sequential feature whose shape is

Algorithm 1: Optimization procedure of BRNN-SA.

```
Input: Picking orders $\mathbf{O} = \{O_1, O_2, \ldots, O_N\}$, Penalty $\omega$.
Output: Best order sequence $\mathbf{X}_{\text{best}} = \{x_i\}$.
1:  Initialize $t \leftarrow 1$ and temperature $T \leftarrow T_0$.
2:  Generate $M_0$ candidate points $\mathbf{X}_{0j}$ $(j = 1, 2, \ldots, M_0)$
3:  Select an initial searching point $\mathbf{X}_1$ from $\mathbf{X}_{0j}$ for
    minimizing $h(\mathbf{X}_{0j}) = f(\mathbf{X}_{0j}) + \omega g(\mathbf{X}_{0j})$.
4:  while Until reaching the time limit do
5:    Select $M$ candidate points $\mathbf{X}_{tj}$ $(j = 1, 2, \ldots, M)$
      from the neighborhood $\mathbf{S}(\mathbf{X}_t)$ uniformly at random.
6:    Evaluate $h(\mathbf{X}_{tj}) = f(\mathbf{X}_{tj}) + \omega g(\mathbf{X}_{tj})$.
7:    Select the top-$K$ candidate points $\hat{\mathbf{X}}_{tk}$,
      $(k = 1, 2, \ldots, K)$ from $\mathbf{X}_{tj}$ according to $h(\mathbf{X}_{tj})$.
8:    Evaluate $H(\hat{\mathbf{X}}_{tk}) = F(\hat{\mathbf{X}}_{tk}) + \omega G(\hat{\mathbf{X}}_{tk})$.
9:    $\hat{\mathbf{X}}_t \leftarrow \hat{\mathbf{X}}_{tk^*}$, where $k^* \leftarrow \text{argmin}_k\, H(\hat{\mathbf{X}}_{tk})$.
10:   if $H(\hat{\mathbf{X}}_t) < H(\mathbf{X}_{\text{best}})$ then
11:     $\mathbf{X}_{\text{best}} \leftarrow \hat{\mathbf{X}}_t$.
12:   Generate a random number $0 \le r \le 1$.
13:   $P \leftarrow \exp((H(\hat{\mathbf{X}}_t) - H(\mathbf{X}_t))/T)$.
14:   if $H(\hat{\mathbf{X}}_t) < H(\mathbf{X}_t)$ or $r \le P$ then
15:     $\mathbf{X}_{t+1} \leftarrow \hat{\mathbf{X}}_t$.
16:   $T \leftarrow \alpha T$.
17:   $t \leftarrow t + 1$.
18: return $\mathbf{X}_{\text{best}}$
```

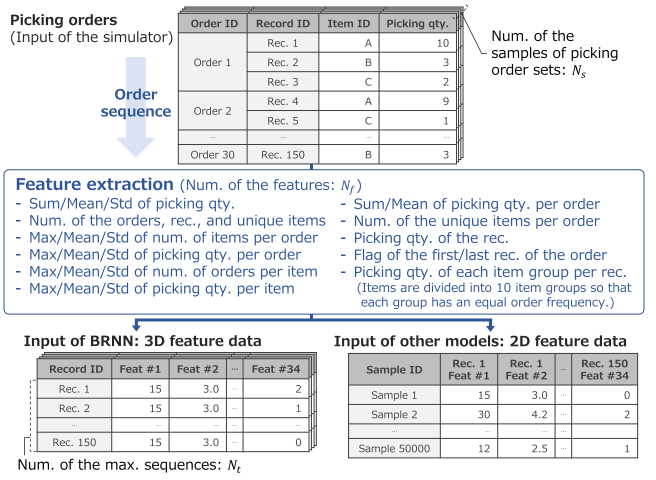
Picking orders
(Input of the simulator)
Order sequence
Order ID | Record ID | Item ID | Picking qty.
Order 1 | Rec. 1 | A | 10
Rec. 2 | B | 3
Rec. 3 | C | 2
Order 2 | Rec. 4 | A | 9
Rec. 5 | C | 1
Order 30 | Rec. 150 | B | 3
Num. of the samples of picking order sets: N_s
Feature extraction (Num. of the features: N_f)
- Sum/Mean/Std of picking qty.
- Num. of the orders, rec., and unique items
- Max/Mean/Std of num. of items per order
- Max/Mean/Std of picking qty. per order
- Max/Mean/Std of num. of orders per item
- Max/Mean/Std of picking qty. per item
- Sum/Mean of picking qty. per order
- Num. of the unique items per order
- Picking qty. of the rec.
- Flag of the first/last rec. of the order
- Picking qty. of each item group per rec. (Items are divided into 10 item groups so that each group has an equal order frequency.)
Input of BRNN: 3D feature data
Record ID | Feat #1 | Feat #2 | … | Feat #34
Rec. 1 | 15 | 3.0 | … | 2
Rec. 2 | 15 | 3.0 | … | 1
Rec. 150 | 15 | 3.0 | … | 0
Num. of the max. sequences: N_t
Input of other models: 2D feature data
Sample ID | Rec. 1 Feat #1 | Rec. 1 Feat #2 | … | Rec. 150 Feat #34
Sample 1 | 15 | 3.0 | … | 0
Sample 2 | 30 | 4.2 | … | 2
Sample 50000 | 12 | 2.5 | … | 1

Fig. 6. Feature formats of surrogate models. Picking orders are converted to 3D feature data for BRNN and 2D feature data for others.

(N_s, N_t, N_f), where N_s is the number of samples of the picking order sets, N_t is the maximum sequence number, and N_f is the number of the features. Other models, which cannot handle the sequential feature, use the 2D feature consisting of horizontally connected sequential features. This data shape is $(N_s, N_t \times N_f)$. In this experiment, we used $N_f = 34$ features and set $N_t = 150$ to cover all training dataset. In the pre-training of GP, only 200 randomly selected samples were used to shorten the training time. The prediction accuracies of makespans and deadlock occurrence were evaluated by 4-fold cross-validation. The computing environment is a Ryzen 3950X CPU (3.70 GHz, 32 threads), 128 GB RAM, and NVIDIA RTX3080 GPU.

B. Prediction Accuracy Comparison

In this experiment, we evaluate the prediction accuracies of the pre-trained surrogate models. These models are compared based on accuracy, defined as the number of correctly predicted data points out of all the data points and the area under the ROC curve (ROC-AUC) for the deadlock occurrence prediction. Mean absolute percentage error (MAPE) and mean absolute error (MAE) are used as metrics for the makespan regression.

The evaluation results of each metric are shown in Table II. For the deadlock occurrence prediction, the classification accuracy is highest for GBT (93.7%), BRNN (93.2%), and BNN (93.2%) in that order. However, the differences in classification accuracies between all models are not so significant (92.7–93.7%), which implies that the order sequence has little effect on the deadlock occurrence. On the other hand, for the makespan regression, BRNN shows a lower MAPE (2.68%) than GBT (3.45%), BNN (5.15%), and other methods. Additionally, we confirmed the BRNN inference time (0.89 msec/sample) was over 10000 times faster than the PPCPP simulation runtime (11.7 sec/sample).

We also evaluated how accurately each model can predict the makespan changes according to the order sequences of one order set. Fig. 7 shows the simulated and predicted makespans of all combinations of the seven randomly generated orders (5040 combinations). The horizontal axis represents each order sequence sorted by its simulated makespan. The vertical axis represents the makespan of each order sequence obtained by simulation and prediction with the pre-trained surrogate models. We can see that the BRNN prediction (blue line) is closer to the simulation result (orange line) than any other surrogate model. These results suggest that BRNN is suitable as a surrogate model to predict performances strongly affected by the sequential feature such as the order sequence, which is the decision variable of OSPs.

Based on these findings, we answer the RQ1 as follows. **RQ1**: Which surrogate model is most suitable for the OSPs of PPCPP systems?

A1: In the performance measures of OSPs, makespan is strongly affected by the order sequences. To infer such performance measures precisely, surrogate models that can consider the sequential input, such as BRNN, are suitable.

C. Optimization Runtime Comparison

Next, we evaluated the total runtimes to find the optimal order sequence by the proposed surrogate pruning optimization method. In this experiment, we used three problem instances consisting of 20 picking orders that are randomly generated from the same distribution with training data. These problems were solved by the proposed optimization methods with each surrogate model (BRNN, BNN, GBT, GP, LR) and the original SA. All the experiments were executed ten times each, and their time limits were 900 sec. We then compared the averages of the objective values of each final solution (Final obj.) and the runtimes to find the solution whose objective value is equal to or lower than the original SA's final solution (LE runtime). Table III shows the optimization results. As we can see, BRNN-SA can find the best solution to two problems and show the shortest

TABLE II
PREDICTION RESULT COMPARISON OF MAKESPAN AND DEADLOCK OCCURRENCE

Surrogate model	Deadlock classification		Makespan regression		Inference time
	Accuracy [%]	ROC-AUC	MAPE [%]	MAE	[msec/sample]
BRNN	93.2 ± 0.3	$\mathbf{0.987 \pm 0.001}$	$\mathbf{2.68 \pm 0.21}$	$\mathbf{53.4 \pm 2.2}$	0.89
BNN	93.2 ± 0.6	0.985 ± 0.001	5.15 ± 0.43	98.9 ± 1.4	0.96
GBT	$\mathbf{93.7 \pm 0.2}$	$\mathbf{0.987 \pm 0.000}$	3.45 ± 0.04	63.7 ± 1.2	0.02
GP	92.8 ± 0.3	0.982 ± 0.001	7.73 ± 0.61	123.8 ± 13.5	0.84
LR	92.7 ± 0.3	0.983 ± 0.001	7.87 ± 1.68	156.9 ± 22.7	0.002
Simulation	(100.0)	(1.0)	(0.0)	(0.0)	11752

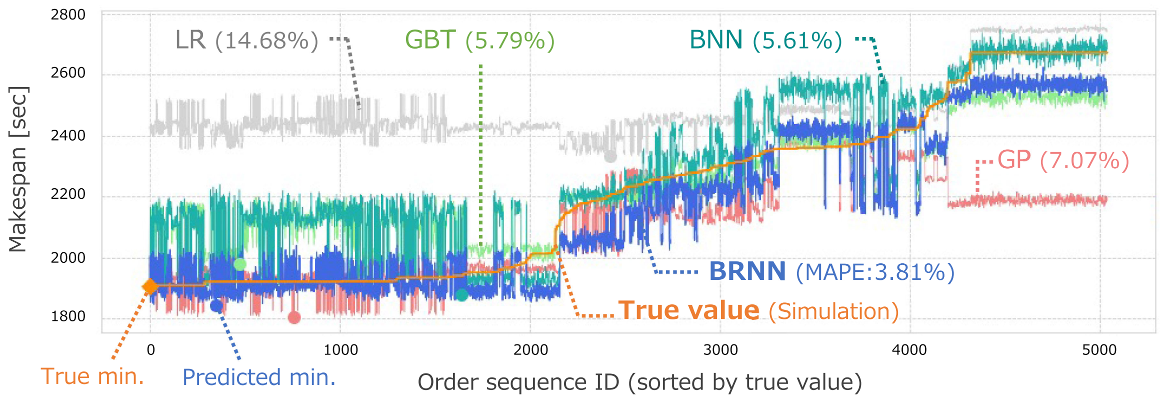

Fig. 7. Simulated and predicted makespans (y-axis) of all sequence combinations of seven orders (x-axis) sorted by simulated makespans. The orange line indicates the simulated true makespans, and the others are the predicted makespans of each surrogate model annotated with MAPE. If the prediction results are always close to the simulation results, the surrogate model can predict the makespan changes according to the order sequence (decision variables).

TABLE III
OBJECTIVES AND RUNTIMES OF THE OPTIMIZATION METHODS ASSISTED BY EACH SURROGATE MODEL

	Problem 1			Problem 2			Problem 3			Average
Method	Final obj.	LE run-time [s]	Speed-up ratio	Final obj.	LE run-time [s]	Speed-up ratio	Final obj.	LE run-time [s]	Speed-up ratio	Speed-up ratio
BRNN-SA	$\mathbf{3169 \pm 236}$	**106.4**	$\mathbf{\times 8.3}$	2174 ± 127	**67.1**	$\mathbf{\times 11.4}$	$\mathbf{1805 \pm 74}$	**101.4**	$\mathbf{\times 7.9}$	$\mathbf{\times 8.9}$
BNN-SA	4090 ± 239	–	$\times 0.0$	$\mathbf{2171 \pm 119}$	84.4	$\times 9.1$	1853 ± 88	397.5	$\times 2.0$	$\times 1.8$
GBT-SA	3305 ± 228	160.7	$\times 5.5$	2204 ± 119	85.6	$\times 8.9$	1823 ± 89	188.9	$\times 4.2$	$\times 5.6$
GP-SA	3537 ± 247	268.0	$\times 3.3$	2194 ± 80	92.9	$\times 8.2$	1929 ± 72	896.9	$\times 0.9$	$\times 1.9$
LR-SA	3442 ± 265	402.9	$\times 2.2$	2311 ± 106	280.9	$\times 2.7$	1818 ± 72	197.2	$\times 4.1$	$\times 2.8$
Original SA	3706 ± 232	883.1	$\times 1.0$	2699 ± 468	763.9	$\times 1.0$	2064 ± 308	798.2	$\times 1.0$	$\times 1.0$

LE time of all methods (8.9 times faster than the original SA on average). BRNN-SA shows the best optimization performances since BRNN can predict the makespan changes according to the order sequence more accurately than other surrogate models as shown in Fig. 7.

Fig. 8 shows the optimization trajectories of each optimization method in Problem 1. The horizontal axis represents the optimization runtime, and the vertical axis represents the makespan of the best solution $\mathbf{X}_{\text{best}}$ at that moment. We can see here that BRNN-SA quickly finds a solution whose objective is lower than SA's final solution. This demonstrates that surrogate pruning by pre-trained BRNN can choose good candidate points more precisely than other surrogate models even at the beginning of the optimization.

On the basis of these findings, we answer RQ2 as follows;

RQ2: How quickly can the proposed optimization method (BRNN-SA) solve the OSPs of PPCPP systems compared to conventional metaheuristics?

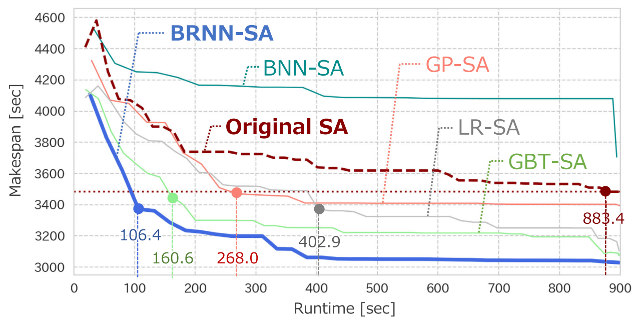

Fig. 8. Makespan trajectories of Problem 1 optimization. The dashed line shows the original SA's trajectory, and the other lines are those of surrogate models. The horizontal dotted line shows the makespan of the original SA's final solution, and the vertical dotted lines indicate the LE runtimes.

A2: BRNN-SA can solve the 20 order OSPs by 8.9 times faster than the original SA. This is presumably because the surrogate pruning with the pre-trained BRNNs can find good order sequences successfully by inferring their performance measures without simulation.

V. Conclusion

This paper proposed the surrogate pruning optimization framework and its implementation (BRNN-SA) to quickly solve the order sequencing problem (OSP) of PPCPP systems, the automated order picking system in logistics warehouses. The proposed method utilizes the Bayesian recurrent neural network (BRNN) as surrogate models to infer sequence-dependent makespans and deadlock occurrences. Then we proposed BRNN-SA, which combines simulated annealing (SA) with pre-trained BRNNs to choose *good* candidate solutions without simulation (*surrogate pruning*). Our experimental results demonstrated the proposed method can streamline solving OSPs thanks to surrogate pruning with the pre-trained BRNNs. For future work, we plan to implement the incremental learning step of surrogate models in this framework for solving larger-scale problems and apply the proposed method to other sequence optimization problems such as job-shop scheduling problems.

Acknowledgment

AI Bridging Cloud Infrastructure (ABCI) of National Institute of Advanced Industrial Science and Technology (AIST) was used for parameter optimization.

References

[1] R. de Koster, T. Le-Duc, and K. J. Roodbergen, "Design and control of warehouse order picking: A literature review," *Eur. J. Oper. Res.*, vol. 182, no. 2, pp. 481–501, 2007.

[2] J. J. Bartholdi and S. T. Hankman, *Warehouse and Distribution Science Release 0.98.1*. Atlanta, GA, USA: Georgia Inst. Technol., 2019, p. 337.

[3] P. Stibinger *et al.*, "Mobile manipulator for autonomous localization, grasping and precise placement of construction material in a semi-structured environment," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 2595–2602, Apr. 2021.

[4] V. Digani, L. Sabattini, and C. Secchi, "A probabilistic Eulerian traffic model for the coordination of multiple AGVs in automatic warehouses," *IEEE Robot. Automat. Lett.*, vol. 1, no. 1, pp. 26–32, Jan. 2016.

[5] H. Yoshitake, R. Kamoshida, and Y. Nagashima, "New automated guided vehicle system using real-time holonic scheduling for warehouse picking," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1045–1052, Apr. 2019.

[6] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, "A dataset for improved RGBD-Based object detection and pose estimation for warehouse pick-and-place," *IEEE Robot. Automat. Lett.*, vol. 1, no. 2, pp. 1179–1185, Jul. 2016.

[7] S. D. Han, S. W. Feng, and J. Yu, "Toward fast and optimal robotic pick-and-place on a moving conveyor," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 446–453, Apr. 2020.

[8] A. R. A. Keshavarz, D. Jaafari, M. Khalaj, and P. Dokouhaki, "A survey of the literature on order-picking systems by combining planning problems," *Appl. Sci.*, vol. 11, no. 22, 2021, Art. no. 10641.

[9] E. M. González-Neira and J. R. Montoya-Torres, "A simheuristic for bi-objective stochastic permutation flow shop scheduling problem," *J. Project Manage.*, vol. 4, pp. 57–80, 2019.

[10] N. Rouky, M. N. Abourraja, J. Boukachour, D. Boudebous, A. E. H. Alaoui, and F. El Khoukhi, "Simulation optimization based ant colony algorithm for the uncertain quay crane scheduling problem," *Int. J. Ind. Eng. Computations*, vol. 10, no. 1, pp. 111–132, 2019.

[11] M. S. Umam, M. Mustafid, and S. Suryono, "A hybrid genetic algorithm and tabu search for minimizing makespan in flow shop scheduling problem," *J. King Saud Univ. Comput. Informat. Sci.*, to be published, doi: 10.1016/j.jksuci.2021.08.025.

[12] C. Y. Cheng, S. W. Lin, P. Pourhejazy, K. C. Ying, and J. W. Zheng, "Minimizing total completion time in mixed-blocking permutation flowshops," *IEEE Access*, vol. 8, pp. 142065–142075, 2020.

[13] B. Khurshid, S. Maqsood, M. Omair, B. Sarkar, I. Ahmad, and K. Muhammad, "An improved evolution strategy hybridization with simulated annealing for permutation flow shop scheduling problems," *IEEE Access*, vol. 9, pp. 94505–94522, 2021.

[14] C. Oh, J. M. Tomczak, E. Gavves, and M. Welling, "Combinatorial Bayesian optimization using the graph cartesian product," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 2910–2920.

[15] A. Deshwal, S. Belakaria, and J. R. Doppa, "Mercer features for efficient combinatorial Bayesian optimization," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 7210–7218.

[16] E. L. Lawler, J. K. Lenstra, A. H. Rinnooy Kan, and D. B. Shmoys, "Sequencing and scheduling: Algorithms and complexity," *Handbooks Operations Res. Manage. Sci.*, vol. 4, pp. 445–522, 1993.

[17] N. Boysen, K. Stephan, and F. Weidinger, "Manual order consolidation with put walls: The batched order bin sequencing problem," *EURO J. Transp. Logistics*, vol. 8, no. 2, pp. 169–193, 2019.

[18] A. R. F. Pinto and M. S. Nagano, "An approach for the solution to order batching and sequencing in picking systems," *Prod. Eng.*, vol. 13, no. 3–4, pp. 325–341, 2019.

[19] Z. Zhou, Y. S. Ong, M. H. Nguyen, and D. Lim, "A study on polynomial regression and Gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm," in *Proc. IEEE Congr. Evol. Computat.*, Sep. 2005, vol. 3, pp. 2832–2839.

[20] Z. Pitra, L. Bajer, and M. Holena, "Comparing SVM, Gaussian process and random forest surrogate models for the CMA-ES," *ITAT*, pp. 186–193, 2015.

[21] N. V. Queipo and E. Nava, "A gradient boosting approach with diversity promoting measures for the ensemble of surrogates in engineering," *Struct. Multidisciplinary Optim.*, vol. 60, no. 4, pp. 1289–1311, 2019.

[22] G. Briffoteaux *et al.*, "Parallel surrogate-assisted optimization: Batched Bayesian neural network-assisted GA versus q-EGO," *Swarm Evol. Computat.*, vol. 57, 2020, Art. no. 100717.

[23] A. Kiuchi *et al.*, "Bayesian optimization algorithm with agent-based supply chain simulator for multi-echelon inventory management," in *Proc. IEEE Int. Conf. Automat. Sci. Eng., CASE*, 2020, pp. 418–425.

[24] J. Stork *et al.*, "Open issues in surrogate-assisted optimization," *Stud. Comput. Intell.*, vol. 833, pp. 225–244, 2020.

[25] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.

[26] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.