Complex & Intelligent Systems (2024) 10:825–846
https://doi.org/10.1007/s40747-023-01179-0

ORIGINAL ARTICLE

Data-driven evolutionary computation for service constrained inventory optimization in multi-echelon supply chains

Ziang Liu[1] · **Tatsushi Nishi**[1]

Received: 21 August 2022 / Accepted: 7 July 2023 / Published online: 9 August 2023

Abstract
Supply chain digital twin has emerged as a powerful tool in studying the behavior of an actual supply chain. However, most studies in the field of supply chain digital twin have only focused on what-if analysis that compares several different scenarios. This study proposes a data-driven evolutionary algorithm to efficiently solve the service constrained inventory optimization problem using historical data that generated by supply chain digital twins. The objective is to minimize the total costs while satisfying the required service level for a supply chain. The random forest algorithm is used to build surrogate models which can be used to estimate the total costs and service level in a supply chain. The surrogate models are optimized by an ensemble approach-based differential evolution algorithm which can adaptively use different search strategies to improve the performance during the computation process. A three-echelon supply chain digital twin on the geographic information system (GIS) map in real-time is used to examine the efficiency of the proposed method. The experimental results indicate that the data-driven evolutionary algorithm can reduce the total costs and maintain the required service level. The finding suggests that our proposed method can learn from the historical data and generate better inventory policies for a supply chain digital twin.

Keywords Evolutionary algorithm · Inventory management · Data-driven · Supply chain · Digital twin

Introduction

Supply chain design, planning, and operation decisions are essential for the success of a company [1]. A typical supply chain may involve suppliers, manufacturers, retailers, and customers. Inventory management is one of the functional elements of supply chain management [2]. Inventory has impacts on various aspects of a supply chain including assets, costs, responsiveness, and material flow time [1]. An important challenge in inventory management is how to minimize costs and improve customer service in a supply chain [3].

Mathematical programming has been an essential tool in inventory management. An inventory optimization problem can be formulated as a mathematical programming model and solved by heuristic, metaheuristic or exact algorithms. Many studies have applied mathematical programming approaches to optimize inventory planning problems in supply chains [4–6]. However, since a supply chain is a complex system that consists of multiple players and processes, it is difficult to fully reflect the dynamics and complexities in inventory management of a supply chain by analytical approaches. Furthermore, mathematical programming models are usually simplified with many assumptions to ensure solvability [7].

Another important tool in inventory management of a supply chain is the simulation approach. Compared to the mathematical approaches, simulation can further capture the complexities and dynamics of supply chains [8]. Simulation can provide valuable insights for inventory management through conducting what-if analysis. However, for inventory optimization problems with a large search space, there are obvious difficulties in finding the optimal solutions by simulation. To address this challenge, simulation-based optimization provides a promising framework for the optimization of inventory policy in supply chain simulations. A number of studies have shown that the inventory policy can be improved by simulation-based optimization [7–9]. A major

✉ Ziang Liu
liu.ziang@okayama-u.ac.jp

[1] Faculty of Environmental, Life, Natural Science and Technology, Okayama University, 3-1-1 Tsushima-naka, Kita-ku, Okayama, Okayama 700-8530, Japan

problem with these studies was that the simulation models are oversimplified and cannot be used to simulate the actual supply chain.

Recently, there has been a growing number of studies focusing on supply chain digital twins [10–13]. A supply chain digital twin is a digital dynamic simulation model of a real-world logistics system [14], which can reflect a physical object in real-time [15]. Previous researches have implemented digital twins to analyze various supply chains, such as healthcare [10], pharmaceutical [12], and food supply chains [16]. These studies have compared the performance of supply chains under different scenarios by conducting a what-if analysis. However, the optimization of supply chain digital twins remains a major challenge. First, through what-if analysis, the supply chain digital twin can provide simulation results for a given input as a black-box function. A recent study [17] have indicated that the digital twin is a black box for the multiple stakeholders in a supply chain. An inventory planning problem usually has a large search space which involves decision variables such as reorder point and order quantity for each facility. This method clearly is not efficient for optimizing inventory systems. Second, the objective of inventory management is to minimize costs and improve customer service. Since the supply chain digital twin can only provide an output for a given input, the relationships between the decision variables and the costs, and customer service, are still unclear. Third, simulation-based optimization provides a general framework for solving such problems, and metaheuristic algorithms such as differential evolution (DE), particle swarm optimization (PSO) and genetic algorithm (GA), can provide near-optimal solutions. However, metaheuristic algorithms require considerable number of function evaluations, which is very time-consuming for a complex system such as a supply chain digital twin. Consequently, one of the major challenges is to develop an efficient algorithm to optimize the performance of supply chain digital twins.

Recent developments in the field of machine learning have led to a growing interest in data-driven evolutionary optimization. In data-driven evolutionary optimization, machine learning models are trained from the collected data to approximate the objective functions and/or constraints functions [18]. By equipping evolutionary algorithms with machine learning models, data-driven evolutionary optimization can efficiently solve expensive optimization problems. Previous studies have implemented data-driven evolutionary algorithms in various real-world problems such as, blast furnace optimization [19], trauma system design optimization [20, 21], and airfoil design optimization [22, 23], etc. However, there remains a research gap concerning the optimization of inventory policies for supply chain digital twins using data-driven evolutionary algorithms. The key questions that remain unanswered are whether machine learning models can learn the intricate relationships between inventory policies and supply chain performance and whether data-driven evolutionary optimization can improve supply chain performance.

This study proposes a framework of data-driven evolutionary optimization for inventory management optimization problems in a multi-echelon supply chain. The objective of this paper is to efficiently optimize the inventory policies of multiple facilities in supply chain digital twins based on collected data samples. First, this study proposes a general formulation for the service constrained inventory optimization problem which aims to minimize the total costs while not violating the required service level. The decision variables are the inventory policy for each facility in the supply chain. Second, the random forest algorithm is applied to estimate the total costs and service level for a given inventory policy. Third, an ensemble approach-based differential evolution algorithm (EDE) that can dynamically use different search strategies during the computation process is proposed as an optimizer to explore the optimal solutions on the surrogate models. It should be emphasized that the proposed method is a versatile framework, and it can be easily adapted to incorporate other evolutionary algorithms. To examine the efficiency of the proposed EDE algorithm, we conducted a comparative analysis with other well-established algorithms, including PSO and several variants of DE, all of which have demonstrated superior performance when combined with data-driven approaches in previous studies [18, 24, 25]. Finally, a three-echelon supply chain digital twin on the GIS map in real-time that developed by a multimethod modeling approach is used to evaluate the efficiency of our proposed method. The experimental results indicate that the proposed method can reduce the total cost by 0.33% on average compared with the training data and not violate the required service levels without further access to the supply chain digital twin. This finding shows that data-driven evolutionary optimization is a promising method for optimizing the inventory systems in supply chain digital twins.

This study makes several contributions to the current literature. First, this study presents a theoretical framework of data-driven evolutionary optimization for service constrained inventory management problems in multi-echelon supply chain, which is integrated by evolutionary computation, machine learning, a general formulation of inventory optimization problems and digital twin technique. Second, a novel ensemble approach-based data-driven DE algorithm is proposed to efficiently search for a better inventory policy in surrogate models. The proposed ensemble approach can dynamically adopt different search strategies based on the improvement in current and historical objective values and constraints violations. Also, three constraint-handling methods are used to generate feasible solutions for the proposed algorithm. Third, the proposed frame is examined by

an inventory optimization problem in a supply chain digital twin on the GIS map in real-time. This is the first study that applied data-driven evolutionary algorithms for the service constrained inventory optimization problem of supply chain digital twins.

The remaining part of the paper proceeds as follows: Section "Literature review" gives a review on simulation and digital twin in supply chains and data-driven evolutionary optimization. Section "Data-driven supply chain optimization" illustrates the problem statement and introduces the data-driven evolutionary optimization for inventory management in supply chains. In Section "Experiments", the proposed approach is applied to a case study and the experimental results are reported. Section "Discussion" discusses the experimental results and managerial insights. Finally, Section "Conclusion" summarizes the conclusions.

Literature review

Simulation and digital twin in supply chains

Simulation has been an important approach in the study of supply chain management. Using this approach, researchers have been able to capture the complexities of the real supply chain networks [7]. A large and growing body of literature has investigated the applications of discrete-event simulation and multi-agent systems in supply chain management. Discrete-event simulation is one of the most widely used simulation approaches in supply chain management [26]. Using this approach, Prinz et al. [27] have studied the impacts of new chipper and transport vehicles on the cost and energy efficiency of the forest supply system. Simulation experiments are conducted under different scenarios on the supply chain model, and the results indicate that the new vehicle types with higher capacity can save costs and improve energy efficiency. A typical supply chain involves multiple entities such as suppliers, manufacturers, retailers, and customers. A Multi-agent system is suitable for the study of supply chain simulation, since it can capture the complex interactions between these players [28]. Dai et al. [29] have proposed a framework of supply chain simulation by multi-agent system to address the complexities in supply chains. Their system is used to simulate a supply chain example with different scenarios. Recent studies have been conducted using the multi-agent system approaches to simulate the behaviors of supply chain members [30–32]. The behavior models for different types of supply chain members are predefined, and a virtual supply chain system can be constructed as a multi-agent system.

Recently, researchers have shown an increased interest in supply chain digital twins. A supply chain digital twin can be defined as a digital dynamic simulation model of a real-world logistics system [14]. It allows simulations on the supply chain that is close to reality [33]. Many researchers [10, 12, 34] have utilized the anyLogistix software to implement the digital twins. Since anyLogistix supports what-if analysis to evaluate various scenarios including supply chain disruptions, it has been widely used to investigate the impacts of COVID-19 on supply chains [11, 13, 16].

Although the digital twin in supply chains can reflect the real system's behaviors, it cannot provide optimal solutions for decision-making. Most studies in the field of supply chain digital twin [10–12] have only focused on what-if analysis by conducting experiments under different parameter settings. These studies have failed to analyze how to optimize the performance of supply chain digital twins. This study proposes an efficient data-driven evolutionary algorithm to optimize the inventory policy using the historical data that is generated by supply chain digital twins.

Data-driven evolutionary optimization

Data-driven evolutionary optimization offers an efficient approach for optimization problems with time-consuming objective functions or constraints, optimization problems that are difficult to mathematically formulate, and real-world problems which can only be optimized by collected data [18]. Data-driven evolutionary algorithms can be classified into offline and online algorithms [35].

In offline data-driven evolutionary optimization, algorithms build surrogate models by using the given dataset, and cannot obtain new data during the optimization process [18]. Many offline data-driven evolutionary algorithms have been proposed and have demonstrated excellent performance on benchmark problems [36–38]. Mazumdar et al. [36] have proposed probabilistic selection approaches for solving multi-objective data-driven optimization problems. The superiority of proposed algorithms has been examined on the several benchmark problems. Liu et al. [37] have developed a novel surrogate-assisted indicator-based evolutionary algorithm designed for tackling offline data-driven multi-objective problems. The proposed algorithm has shown its effectiveness on offline data-driven optimization problems with decision variables ranging from 20 to 30. Recently, Huang and Gong [38] have proposed contrastive learning-based approach, where classification model is used to build surrogate model. The experiment results show that their proposed algorithm has good performance on high-dimensional problems. Moreover, offline data-driven evolutionary optimization has been applied to solve real-world problems. Chugh et al. [19] have applied a data-driven evolutionary algorithm to a blast furnace optimization problem with 12 decision variables and 8 objectives. 210 operational data have been used to build the surrogate models for all objective functions. The optimized solutions can dominate the

values of objective functions in the collected data. However, their results have not been verified in practice. Yang et al. [39] have proposed a new data-driven evolutionary algorithm and applied it to an operational indices optimization problem in beneficiation processes. However, they were unable to validate the obtained solutions since the objective function is not available. Guo et al. [40] have studied the optimization of fused magnesium furnaces, where only historical data is available. An offline data-driven evolutionary algorithm is proposed to optimize the furnaces performance in magnesia production. The experimental results show that the proposed algorithm has good performance with limited computational expense. Offline data-driven evolutionary optimization has also been applied to the airfoil design problems [22, 23]. In [22], the objective value of this problem is evaluated by computational fluid dynamic simulation, which is very time-consuming. In their study, 70 simulation results are used to build the surrogate model. The results indicate that data-driven evolutionary algorithms can generate better solutions than the baseline design. Li et al. [23] have proposed a data-driven evolutionary algorithm with perturbation-based ensemble surrogates. The experiment results show that only about 2% computational budgets are required for generating promising solutions compared to non-data-driven approaches.

Online data-driven evolutionary algorithms can actively collect new data samples and update the surrogate models during the optimization process [18]. Since the prediction error of a surrogate model is inevitable, it is important to improve the accuracy of the surrogate model. By adding new data to the surrogate model, online data-driven evolutionary algorithms have more chance to obtain a better solution than offline data-driven evolutionary algorithms. Many online data-driven evolutionary algorithms have been proposed and proven to exhibit excellent performance on large-scale benchmark problems [41–46]. Previous studies have also applied online data-driven evolutionary algorithms to solve real world problems. Long et al. [47] have proposed a data-driven evolutionary algorithm for wind farm layout optimization problems. Since it is time-consuming to evaluate the objective value of a solution in the wind farm layout optimization problems, the surrogate model is used to approximate the objective value. In their study, the general regression neural network has been used to build the surrogate model. The experimental results suggest that their surrogate-assisted evolutionary algorithm has better performance than the other evolutionary algorithms in terms of the wind farm power output. Fu et al. [48] have proposed an online data-driven Harris Hawks constrained optimization algorithm. In their study, the Kriging model is used to generate surrogate models. Their proposed algorithm has been applied to structural optimization of the internal components of a real underwater vehicle, and the algorithm can achieve 18.7% weight reduction. Song et al. [49] have proposed a surrogate sample-assisted PSO algorithm for feature selection problems. Their proposed algorithm can obtain good feature subsets with small computational cost.

In recent years, there has been growing interest in the application of surrogate models in discrete optimization problems. A literature review concluded by Bartz-Beielstein and Zaefferer [50] indicates that although most of the studies have focused continuous optimization problems, the surrogate models can also be applied to discrete optimization problems. Han and Wang [51] have proposed online date-driven evolutionary algorithm using competitive neighborhood search. The random forest algorithm has been used to build the surrogate model and the performance of the proposed algorithm has been examined on expensive constrained combinatorial optimization problems. The experimental results suggest that using the surrogate model can significantly improve the performance of competitive neighborhood search. Wang and Jin [21] have proposed a random forest-assisted data-driven evolutionary algorithm for the optimization of trauma systems. They have compared the performance of the random forests and radial basis function networks in data-driven evolutionary algorithms for discrete optimization problems. Their results suggest that random forests have better performance in the experiments. Gu et al. [52] have proposed a surrogate-assisted evolutionary algorithm for expensive constrained multi-objective discrete optimization problems. The surrogate model is constructed by the random forest algorithm and an individual-based model management strategy has been used to update the model.

Previous studies have applied data-driven evolutionary algorithms to various fields. However, to the best of our knowledge, this is the first study to explore the usefulness of data-driven evolutionary algorithms in service constrained inventory optimization for multi-echelon supply chains. Furthermore, few studies have explored the universality and robustness of evolutionary algorithms. It has been reported that evolutionary algorithms may perform differently for different problems. In data-driven evolutionary optimization, an objective function is estimated by a surrogate model using collected data. The landscapes of the surrogate models are varying stem from the changes in the training data. Therefore, it is important to improve the performance of evolutionary algorithms for various problems. In this study, an ensemble approach is proposed to adaptively select the suitable search strategy in differential evolution during the computation process.

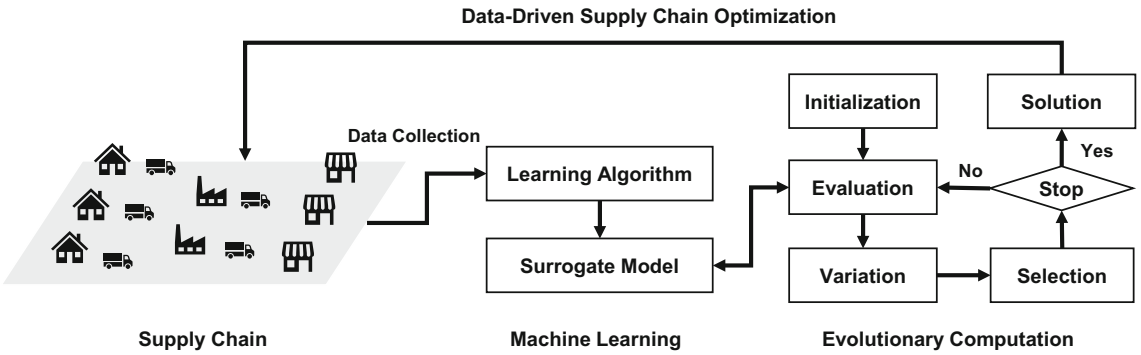


Fig. 1 Data-driven evolutionary optimization for supply chains

Data-driven supply chain optimization

Proposed framework

The framework of the data-driven evolutionary optimization for supply chain digital twins is shown in Fig. 1. The proposed framework consists of three components, which are: supply chain, machine learning, and evolutionary computation. The supply chain data is the input for the proposed method, the data-driven evolutionary algorithm can generate promising solutions for the inventory system in a supply chain digital twin.

Over the past decades, simulation has always played an important role in supply chain management. The recent simulation methods allow for building a detailed supply chain simulation model, also known as the supply chain digital twin. First, the supply chain digital twin is used to generate the training data. The data samples gathered from the supply chain can be used to train surrogate models by machine learning algorithms. This study aims to optimize the inventory policy of each facility to minimize the total costs and not violate the service level. The training dataset in this study includes the inventory policies, total costs, and service level.

Second, the data collected from the supply chain digital twin is used to train the surrogate models. This paper addresses the question of whether the performance of a supply chain can be improved by using historical data. The machine learning algorithm can be used to construct the surrogate models to describe the relationship between inventory policies and performance indicators. Note that there may be various objective functions and constraints for a single supply chain optimization problem. The surrogate models are built separately for different objective functions and constraints. In this study, two surrogate models are trained to predict the total cost and the service level for a given inventory policy.

Third, evolutionary algorithms are used to search for better solutions on the surrogate models. A general computation process of most evolutionary algorithms often consists of the following components: initialization, evaluation, variation, and selection. The first step for an evolutionary algorithm is the initialization, which generates one or more initial solutions for an optimization problem. New solutions can be generated by the variation based on the current solutions. Then, the fitness of each solution is evaluated, and some solutions will be selected and used in the next generation. For data-driven evolutionary algorithms, the fitness values are estimated by the surrogate models.

Problem statement

A major challenge in inventory management is how to minimize the costs and improve customer service in a supply chain [3]. In this section, a general formulation for the service level constrained inventory optimization problem in a multi-echelon supply chain is developed. The objective of this problem formulation is to minimize the total costs while not violating the required service level. The facilities are operated their inventory levels by the (s, S) policies where s is the re-order level, and the S is the order-up-to-level. In an (s, S) policy, the inventory level is reviewed every period; if the inventory is lower than the re-order level s, an order is placed to bring the inventory level to the order-up-to level S. Note that this research aims to propose a data-driven optimization approach for general inventory management problems. The specific features of a certain supply chain should be learned from data. Therefore, a general formulation without many assumptions for the inventory optimization problem is discussed in this study.

Let $I = \{1, 2, \ldots, n\}$ be the set of facilities, where n is a positive integer. For a facility $i \in I$, let x_i be the re-order level, and x_{i+n} be the order-up-to level. The inventory policy $\mathbf{x}$ is a vector of $2n$-dimensional non-negative integers as

$$\mathbf{x} = (x_1, \ldots x_n, x_{n+1}, \ldots, x_{2n}) \tag{1}$$

Let $J = \{1, 2, \ldots, m\}$ be the set of products. For product $j \in J$ and facility $i \in I$, let d_{ij} be the number of orders

placed for product j in facility i, and s_{ij} represents the successful orders for product j in facility i. The service level for a facility is calculated by the number of successful orders and all orders placed for this facility. For facility i, the service level is defined as:

$$r_i(\mathbf{x}) = \frac{\sum_{j=1}^{m} s_{ij}(\mathbf{x})}{\sum_{j=1}^{m} d_{ij}} \tag{2}$$

The service level for the supply chain is defined as:

$$r(\mathbf{x}) = \sum_{i=1}^{n} r_i(\mathbf{x}) \tag{3}$$

Let α represent the minimum desired service level for the supply chain. For each facility $i \in I$, let b_i represents the capability. The total costs $c(\mathbf{x})$ include the sum of initial site cost, inventory carrying cost, inventory cost, processing cost, and transportation cost. The total costs $c(\mathbf{x})$ and service level $r(\mathbf{x})$ are obtained by supply chain simulations. The inventory optimization problem is formulated as follows:

$$\min \quad c(\mathbf{x}) \tag{4}$$

$$\text{s.t. } r(\mathbf{x}) \geq \alpha \tag{5}$$

$$x_i \leq x_{i+n}, \forall i \tag{6}$$

$$x_{i+n} \leq b_i, \forall i \tag{7}$$

$$\mathbf{x} \in \mathbb{Z}_{+}^{2n} \tag{8}$$

The objective function (4) aims to minimize the total cost $c(\mathbf{x})$ for all facilities in a supply chain. Constraints (5) state that the minimum service level should be satisfied. Constraints (6) represent the re-order level should be lower than the order-up-to-level for each facility. Constraints (7) ensure the inventory level of a facility is less than the capacity b_i. Constraints (8) indicate the inventory variables should be non-negative integers.

Surrogate modelling

With our proposed framework, machine learning algorithms can be implemented to build the surrogate model as shown in Fig. 1. An overview of the data-driven evolutionary optimization that conducted by Jin et al. [35] have shown that machine learning algorithms such as polynomial regression, Kriging model, artificial neural networks, and radial basis function networks have been employed in surrogate-assisted evolutionary algorithms.

However, when applying these algorithms to build surrogate models for optimization problems with discrete variables, a disadvantage is that the feature space of the surrogate model may have large areas of redundancy [50]. The tree-based models, such as random forests are discrete in their design, and the random forest algorithm may be the first choice for discrete problems [50]. Recently, random forest algorithm has been widely used in data-driven optimization problems with discrete decision variables [21, 51, 52]. In this study, since the decision variables should be non-negative integers as shown in constraints (8), the random forest algorithm is used to train surrogate models for approximating the objective function and constraints.

The random forest algorithm can be used to solve both classification and regression problems. Since the surrogate model is built to approximate the total cost and the service level in this study, the random forest algorithm for regression is introduced in this section.

A random forest can be considered as an ensemble of a number of decision trees [53]. The flowchart of a random forest is shown in Fig. 2. The main idea of the random forest algorithm is that although a single decision tree may suffer from high variance, a random forest can have better robustness by combining the predictions of different decision trees. In a random forest, each decision tree is built from a random bootstrap sample, which is prepared by randomly choosing a certain number of examples from the training dataset with replacement. Due to the randomness in sampling, decision trees may have different structures [21]. At each node of a decision tree, the mean square error (MSE) between the true target value and the predicted target value is calculated to split the node. The best split, which minimizes the MSE, is used to branch the node. Each built decision tree can make prediction for a given input. The prediction of a random forest model is made by averaging prediction over all decision trees as illustrated in Fig. 2.

This study aims to minimize the total cost while satisfying the required service level based on the collected data from the supply chain digital twin. To approximate the total cost and service level for a given inventory, two random forest models are built separately. Consider a supply chain dataset D that has L samples. For a sample $l \in \{1, 2, \ldots, L\}$, let $\mathbf{x}^l$ be the inventory policies, c^l be the corresponding total cost, and r^l be the corresponding service level. The training dataset can be represented as $D = \{(\mathbf{x}^1, c^1, r^1), (\mathbf{x}^2, c^2, r^2), \ldots, (\mathbf{x}^N, c^N, r^N)\}$, where N represents the number of data points in the dataset. Using the supply chain data D, surrogate model M_c and M_r can be constructed as shown in Algorithm 1. Surrogate model M_c is used to estimate the relations between $\mathbf{x}$ and c as $\widehat{c} = f_c(\mathbf{x})$, and surrogate model M_r is used to estimate the relations between $\mathbf{x}$ and r as $\widehat{r} = f_r(\mathbf{x})$. In the surrogate models, $\widehat{c}$ and $\widehat{r}$ are the estimated total cost and service level for a given inventory policy $\mathbf{x}$.

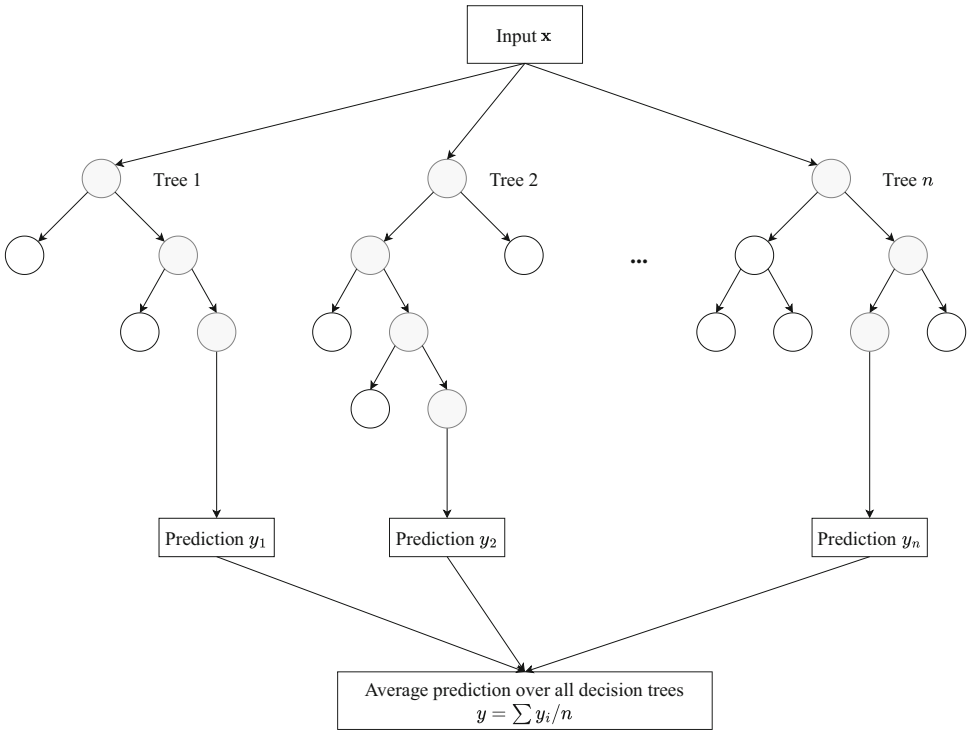


Fig. 2 Flowchart of a random forest for regression

Algorithm 1: Surrogate Modeling for Estimating $\hat{c}$ and $\hat{r}$

Input: Training dataset D
Output: Surrogate model M_c and M_r
1: extract $\mathbf{x}^l$ and c^l for $l \in \{1,2,\ldots,L\}$ from D to construct dataset D_c
2: extract $\mathbf{x}^l$ and r^l for $l \in \{1,2,\ldots,L\}$ from D to construct dataset D_r
3: train a surrogate model M_c using training dataset D_c
4: train a surrogate model M_r using training dataset D_r

Ensemble differential evolution

Previous studies have revealed that the search strategy has a major influence on the performance of evolutionary algorithms [54, 55]. Therefore, it is extremely important to select a suitable search strategy for a given problem. The data-driven evolutionary algorithms evaluate their fitness on the surrogate model. The surrogate models may have different features for different problems. Also, for a complex supply chain system with big data, the surrogate model may become very complex.

DE is a population-based, derivative-free evolutionary algorithm used to solve global optimization problems [56, 57]. A recent study [58] has indicated that DE algorithm and its variants have better performance than the other meta-heuristic algorithms on the benchmark problems and the real-world problems. DE generally consists of the following four steps: initialization, mutation, crossover, and selection [59].

This study provides an ensemble differential evolution (EDE) to adaptively select the suitable search strategy during the computation process. Three classical mutation operators of differential evolution (DE) are used in the proposed algorithm. Also, different constraint-handling techniques are used to generate feasible solutions.

Differential evolution

In this study, DE uses the (s, S) inventory policies in Eq. (1) as a solution vector. Let $K = \{1, 2, \ldots, NP\}$ be the set of solutions, where NP is the population size. For solution $k \in K$ at generation t, the vector of solution $\mathbf{x}_{k,t}$ is represented as

$$\mathbf{x}_{k,t} = (x_{1,k,t}, \ldots x_{n,k,t}, x_{n+1,k,t}, \ldots, x_{2n,k,t}) \tag{9}$$

where $x_{i,k,t}$ is the re-order level for facility i in solution k at generation t, while $x_{i+n,k,t}$ is the order-up-to level. The initial

solutions should be generated before starting the iterations. Since the facility capacity is given and the inventory policies should be non-negative values, the decision variables are defined to be in a given range: $\mathbf{x}_{\min} = (x^1_{\min}, \ldots x^n_{\min}, x^{n+1}_{\min}, \ldots, x^{2n}_{\min})$, and $\mathbf{x}_{\max} = (x^1_{\max}, \ldots x^n_{\max}, x^{n+1}_{\max}, \ldots, x^{2n}_{\max})$. An initial solution is randomly generated within the given range $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$ as:

$$\mathbf{x}_{k,0} = \mathbf{x}_{\min} + \text{rand}_{k,0}[0,\ 1](\mathbf{x}_{\max} - \mathbf{x}_{\min}) \tag{10}$$

where $\text{rand}_{k,0}[0,\ 1]$ is a random vector that uniformly distributed in the range $[0,\ 1]^{2n}$ for solution k before starting the iteration.

After the initialization, mutation operation can be conducted by different mutation strategies. In this step, a mutant vector $\mathbf{v}_{k,t}$ is generated from the existing solutions. Five commonly used mutation strategies are listed below.

$$\text{DE/rand/1} \quad \mathbf{v}_{k,t} = \mathbf{x}_{r1,t} + F\left(\mathbf{x}_{r2,t} - \mathbf{x}_{r3,t}\right) \tag{11}$$

$$\text{DE/rand/2} \quad \mathbf{v}_{k,t} = \mathbf{x}_{r1,t} + F\left(\mathbf{x}_{r2,t} - \mathbf{x}_{r3,t}\right) + F\left(\mathbf{x}_{r4,t} - \mathbf{x}_{r5,t}\right) \tag{12}$$

$$\text{DE/best/1} \quad \mathbf{v}_{k,t} = \mathbf{x}_{best,t} + F\left(\mathbf{x}_{r1,t} - x_{r2,t}\right) \tag{13}$$

$$\text{DE/best/2} \quad \mathbf{v}_{k,t} = \mathbf{x}_{best,t} + F\left(\mathbf{x}_{r1,t} - \mathbf{x}_{r2,t}\right) + F\left(\mathbf{x}_{r3,t} - \mathbf{x}_{r4,t}\right) \tag{14}$$

$$\text{DE/current} - \text{to} - \text{pbest/1}$$
$$\mathbf{v}_{k,t} = \mathbf{x}_{k,t} + F\left(\mathbf{x}_{pbest,t} - \mathbf{x}_{k,t}\right) + F\left(\mathbf{x}_{r1,t} - \mathbf{x}_{r2,t}\right) \tag{15}$$

Let K_{-k} represents the set of all solutions except for solution k. The indices r_1, r_2, r_3, r_4, r_5 are randomly selected from the solution set K_{-k}. The parameter F is often referred to as the scaling factor. In practice, $F \in [0,\ 1]$ is more efficient and stable [56]. In Eqs. (13) and (14), $\mathbf{x}_{best,t}$ is the best solution in the solution set at generation t. In Eq. (15), $\mathbf{x}_{pbest,t}$ is randomly selected from the top $NP \times p$ solutions, where p is a control parameter that determines the percentage of the population that can be chosen as $\mathbf{x}_{pbest,t}$ [60]. Parameter p controls the greediness DE/current-to-pbest/1.

Then, the crossover is conducted to generate the trial vector $\mathbf{u}_{k,t}$ based on the obtained $\mathbf{x}_{k,t}$ and $\mathbf{v}_{k,t}$. In the DE algorithm, the crossover can be implemented by two methods: the binomial or the exponential crossover. The binomial crossover is commonly used in the DE algorithm:

$$u_{i,k,t} = \begin{cases} v_{i,k,t} \text{ if rand}_{i,k,t}[0,\ 1] \le C_r \text{ or } i = i_{\text{rand}} \\ x_{i,k,t} \text{ otherwise} \end{cases} \tag{16}$$

where parameter $C_r \in [0,\ 1]$ controls the crossover rate, and i_{rand} is randomly selected from the facility set I. The random integer $i_{\text{rand}} \in I$ is used to ensure that at least one crossover is implemented. The order-up-to level $u_{i+n,k,t}$ can be generated in the same way.

Finally, the selection operation updates the population by comparing the current vector $\mathbf{x}_{k,t}$ and the trial vector $\mathbf{u}_{k,t}$. For an unconstrained single objective optimization problem, the selection operation can be described as

$$\mathbf{x}_{k,t+1} = \begin{cases} \mathbf{u}_{k,t} \text{ if } f\left(\mathbf{u}_{k,t}\right) \le f\left(\mathbf{x}_{k,t}\right) \\ \mathbf{x}_{k,t} \text{ otherwise} \end{cases} \tag{17}$$

However, the inventory optimization problem in this study consists of several constraints. The comparison method between two solutions is introduced in Section "Constraint-handling techniques".

Constraint-handling techniques

The inventory optimization problem consists of three types of constraints. Different techniques are used to handle these constraints.

First, each facility has capacity constraints, and the inventory level should be nonnegative values. We use $\mathbf{x}_{\max}$ and $\mathbf{x}_{\min}$ to represent the range of possible inventory levels. The vector $\mathbf{v}_{k,t}$ is generated by the mutation operation of the EDE algorithm. For facility i, $v_{i,k,t}$ represents the re-order level and $v_{i+n,k,t}$ represents the order-up-to level. When the mutant vector $\mathbf{v}_{k,t}$ violates the possible range of inventory levels. The variable $v_{i,k,t}$ is reset to a random value as follows:

$$v_{i,k,t} = x^i_{\min} + \text{rand}_{i,k,t}[0,\ 1]\left(x^i_{\max} - x^i_{\min}\right) \quad \text{if } v_{i,k,t} < x^i_{\min} \text{ or } v_{i,k,t} > x^i_{\max} \tag{18}$$

where $\text{rand}_{i,k,t}[0,\ 1]$ is a random value that is uniformly distributed in the range $[0,\ 1]$ for the i th element of solution k at iteration t. Similarly, the order-up-to level $v_{i+n,k,t}$ can be reset in the same way.

Second, the re-order level should be lower than the order-up-to level for the solution $\mathbf{x}_{k,t}$ and trial vector $\mathbf{u}_{k,t}$. In some situations, the $\mathbf{x}_{k,t}$ and $\mathbf{u}_{k,t}$ may violate these constraints. Repair strategies can transform an infeasible solution into a feasible one and they are specific to the optimization problem [61]. In this study, a repair strategy is proposed to repair $\mathbf{x}_{k,t}$ and $\mathbf{u}_{k,t}$ such that $x_{i,k,t} \le x_{i+n,k,t}$ and $u_{i,k,t} \le u_{i+n,k,t}$ can be satisfied. In detail, the proposed repair strategy first check whether $\mathbf{x}_{k,t}$ satisfy $x_{i,k,t} \le x_{i+n,k,t}$ for facility $i \in I$. If $x_{i,k,t} > x_{i+n,k,t}$, the repair strategy is conducted to swap the values of $x_{i,k,t}$ and $x_{i+n,k,t}$. The same process is also applied to $\mathbf{u}_{k,t}$.

Algorithm 2: Selection Operation

Input: current vector $\mathbf{x}_{k,t}$, trial vector $\mathbf{u}_{k,t}$, service level α,
estimated costs $f_c(\mathbf{x}_{k,t})$, $f_c(\mathbf{u}_{k,t})$,
estimated service level $f_r(\mathbf{x}_{k,t})$, $f_r(\mathbf{u}_{k,t})$

Output: new vector $\mathbf{x}_{k,t+1}$

$g(\mathbf{x}_{k,t}) \leftarrow f_r(\mathbf{x}_{k,t}) - \alpha$
$g(\mathbf{u}_{k,t}) \leftarrow f_r(\mathbf{u}_{k,t}) - \alpha$
if $g(\mathbf{x}_{k,t}) \geq 0$ and $g(\mathbf{u}_{k,t}) \geq 0$ and $f_c(\mathbf{u}_{k,t}) < f_c(\mathbf{x}_{k,t})$ **then**
  $\mathbf{x}_{k,t+1} \leftarrow \mathbf{u}_{k,t}$
else if $g(\mathbf{x}_{k,t}) < 0$ and $g(\mathbf{u}_{k,t}) < 0$ and $g(\mathbf{u}_{k,t}) > g(\mathbf{x}_{k,t})$ **then**
  $\mathbf{x}_{k,t+1} \leftarrow \mathbf{u}_{k,t}$
else if $g(\mathbf{u}_{k,t}) \geq 0$ and $g(\mathbf{x}_{k,t}) < 0$ **then**
  $\mathbf{x}_{k,t+1} \leftarrow \mathbf{u}_{k,t}$
else
  $\mathbf{x}_{k,t+1} \leftarrow \mathbf{x}_{k,t}$
end if

Third, the minimum service level should be satisfied as shown in constraints (5). The superiority of feasible solution strategy [62] is used to deal with these constraints and conduct environmental selection. In the proposed algorithm, the environmental selection is implemented by considering both the objective function and the service level constraints. Algorithm 2 shows the pseudocode of the selection operation by using the concept of the superiority of feasible solutions.

In Algorithm 2, the vector $\mathbf{x}_{k,t+1}$ is updated by the trial vector $\mathbf{u}_{k,t}$ if one of the following three criteria is satisfied:

- The current vector $\mathbf{x}_{k,t}$ is infeasible while the trial vector $\mathbf{u}_{k,t}$ is feasible.
- All the solutions are feasible, while $\mathbf{u}_{k,t}$ has better objective value than $\mathbf{x}_{k,t}$.
- All the solutions are infeasible, while $\mathbf{u}_{k,t}$ has smaller constraint violation than $\mathbf{x}_{k,t}$.

By implementing the rules above, the computation process is divided into three stages. In the first stage, most of the solutions are infeasible, and the solutions with lower constraint violations are selected. This method encourages the solutions to approach the feasible regions. In the next stage, both feasible and infeasible solutions exist in the solution set, and this constraint-handling technique will select the feasible solutions. Finally, when all the solutions are feasible, the superiority of solutions is compared by the objective values.

Ensemble approach

In this study, an ensemble differential evolution (EDE) is proposed to adaptively use different mutation strategies during the computation process. For the ensemble approaches in metaheuristic algorithms, it is important to use search strategies with different features [55].

Three DE mutation strategies, DE/rand/1, DE/best/2, and DE/current-to-pbest/1, are used in the proposed algorithm. First, DE/rand/1 is a basic and one of the most widely used mutation strategies in the DE algorithm. Second, DE/best/2 explore the neighbor solutions of the best solution $\mathbf{x}_{best,t}$. This strategy has a strong exploitation ability while it tends to be trapped in the local optima. The third strategy, DE/current-to-pbest/1, uses the information of an elite solution $\mathbf{x}_{pbest,t}$ to guide the current solution $\mathbf{x}_{k,t}$. This strategy has a better exploitation ability than the DE/rand/1 strategy and has more randomness than the DE/best/2 strategy.

The adoption rate of each mutant strategy is calculated based on the solution improvement of each strategy. Let $H = \{1, 2, 3\}$ be the strategy set, where 1, 2, 3 denote DE/rand/1, DE/best/2, and DE/current-to-pbest/1, respectively. For each strategy $h \in H$, binary variable $z_{k,h,t}$ takes 1 if mutation strategy h is used to generate the trial solution $\mathbf{u}_{k,t}$ for $\mathbf{x}_{k,t}$, and 0 otherwise. The solution improvement is calculated based on the improvement of the objective function and service level.

In every iteration, the improvement of the objective function $\Delta c_{k,h,t}$ is recorded as follows:

$$\Delta c_{k,h,t} = \begin{cases} \max\{z_{k,h,t}[f_c(\mathbf{x}_{k,t}) - f_c(\mathbf{u}_{k,t})], 0\}, \\ \qquad \text{if } f_r(\mathbf{x}_{k,t}) \geq \alpha \text{ and } f_r(\mathbf{u}_{k,t}) \geq \alpha \\ 0, \qquad \text{otherwise} \end{cases} \tag{19}$$

When $z_{k,h,t}$ takes 1 and the service level constraints in (5) are satisfied for both $\mathbf{x}_{k,t}$ and $\mathbf{u}_{k,t}$, the fitness improvement $\Delta f_{k,h,t}$ is recorded as the difference between $f_c(\mathbf{x}_{k,t})$ and $f_c(\mathbf{u}_{k,t})$.

Similarly, the improvement of the service level $\Delta r_{k,h,t}$ is recorded as follows:

$$\Delta r_{k,h,t} = \begin{cases} \max\{z_{k,h,t}[\alpha - f_r(\mathbf{x}_{k,t}) - \max\{\alpha - f_r(\mathbf{u}_{k,t}), 0\}], 0\}, \\ \qquad \text{if } f_r(\mathbf{x}_{k,t}) < \alpha \\ 0, \qquad \text{otherwise} \end{cases} \tag{20}$$

where $\max\{\alpha - f_r(\mathbf{u}_{k,t}), 0\}$ represents the level of constraint violation. When $\mathbf{x}_{k,t}$ violates the service level constraints and $z_{k,h,t}$ takes 1, the improvement of the service level $\Delta r_{k,h,t}$ will be calculated as the difference between the level of constraint violation of $\mathbf{u}_{k,t}$ and $\mathbf{x}_{k,t}$.

Then both $\Delta r_{k,h,t}$ and $\Delta c_{k,h,t}$ are normalized to a range of [0, 1]. In iteration t, The normalized value, $\Delta c^{norm}_{k,h,t}$ and $\Delta r^{norm}_{k,h,t}$ can be calculated as follows:

$$\Delta c^{norm}_{k,h,t} = \frac{\Delta c_{k,h,t} - \min\limits_{k\in K, h\in H} \Delta c_{k,h,t}}{\max\limits_{k\in K, h\in H} \Delta c_{k,h,t} - \min\limits_{k\in K, h\in H} \Delta c_{k,h,t} + \epsilon} \tag{21}$$

$$\Delta r^{norm}_{k,h,t} = \frac{\Delta r_{k,h,t} - \min\limits_{k\in K, h\in H} \Delta r_{k,h,t}}{\max\limits_{k\in K, h\in H} \Delta r_{k,h,t} - \min\limits_{k\in K, h\in H} \Delta r_{k,h,t} + \epsilon} \tag{22}$$

where $\Delta c_{k,h,t}$ is the value that needs to be normalized, $\min\limits_{k\in K, h\in H} \Delta c_{k,h,t}$ is the smallest value in iteration t and $\max\limits_{k\in K, h\in H} \Delta c_{k,h,t}$ is the largest value, ϵ denotes a small value to prevent the zero division. In this study, ϵ is set to 10^{-8}.

At the end of iteration t, the solution improvement $\Delta f_{k,h,t}$ of strategy h on solution k is calculated as the summation of $\Delta c^{norm}_{k,h,t}$ and $\Delta r^{norm}_{k,h,t}$ as follows:

$$\Delta f_{k,h,t} = \Delta c^{norm}_{k,h,t} + \Delta r^{norm}_{k,h,t} \tag{23}$$

Let G be a certain number of iterations called the learning period. When the learning period is finished. The average improvement of strategy h at g th learning period is calculated as follows:

$$\overline{\Delta f}_{h,g} = \frac{\sum_{k=1}^{NP} \sum_{t=Gg}^{(g+1)G-1} \Delta f_{k,h,t}}{\sum_{k=1}^{NP} \sum_{t=Gg}^{(g+1)G-1} z_{k,h,t}} \tag{24}$$

Let γ be the decay parameter within the range [0, 1]. The parameter γ indicates how much the historical fitness improvements are worth at the current learning period g. If γ is set to 0, the historical fitness improvements will not affect the strategy selection. In the first learning period, we set the weighted historical fitness improvement $w_{h,0} = 0$ for $h \in H$. The weighted historical fitness improvement $w_{h,g}$ can be updated as the summation of the average improvement $\overline{\Delta f}_{h,g}$ and the weighted historical fitness improvement in learning period $g-1$, as follows:

$$w_{h,g} = \overline{\Delta f}_{h,g} + w_{h,g-1}\gamma \tag{25}$$

This equation can also be expressed as follows:

$$w_{h,g} = \overline{\Delta f}_{h,g} + \overline{\Delta f}_{h,g-1}\gamma + \overline{\Delta f}_{h,g-2}\gamma^2 + \overline{\Delta f}_{h,g-3}\gamma^3 + \ldots \tag{26}$$

Observe this equation, we can find that the recent average improvement has more impact on the results when $\gamma \in (0, 1)$.

The adoption rate of strategy h at the g th learning period is calculated as follows:

$$p_{h,g} = \frac{w_{h,g}}{\sum_{h=1}^{3} w_{h,g}} \tag{27}$$

The adoption rate $p_{h,g}$ should be within the range [0, 1]. At the end of each learning period, the adoption rate $p_{h,g}$ is updated. Then, the roulette wheel selection is conducted based on $p_{h,g}$ to assign different strategies to the solution set K.

Framework of the proposed algorithm

Figure 3 shows the flowchart of the EDE algorithm. The first step in this algorithm is to input the parameters and dataset, and then two surrogate models, M_C and M_r, can be constructed using Algorithm 1. The initial solutions are generated by Eq. (10). Then, we initialize the iteration number t, the learning period g, and the weighted historical fitness improvement $w_{h,g}$ for all strategies in H to 0. For each individual $k \in K$, a mutation strategy $h \in H$ is randomly allocated and $z_{k,h,t}$ is initialized accordingly. It is important to note that individual k can have one mutation strategy h in each iteration, which can be expressed as $\sum_{k\in K} z_{k,h,t} = 1$.

The proposed algorithm conducts mutation, crossover, and selection operations iteratively to generate new candidate solutions after initialization. For solution $k \in K$ at

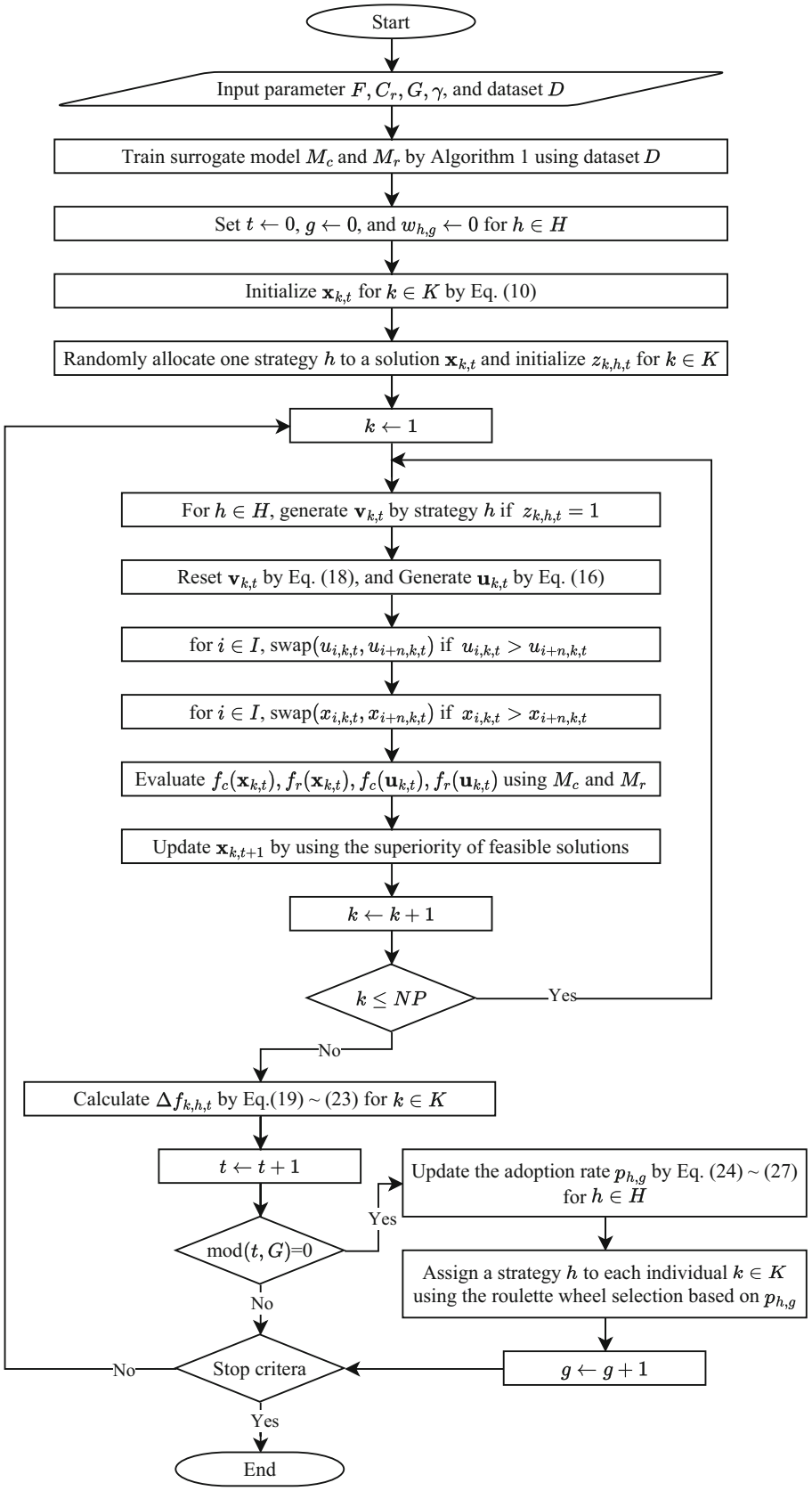


Fig. 3 The flowchart of the EDE algorithm

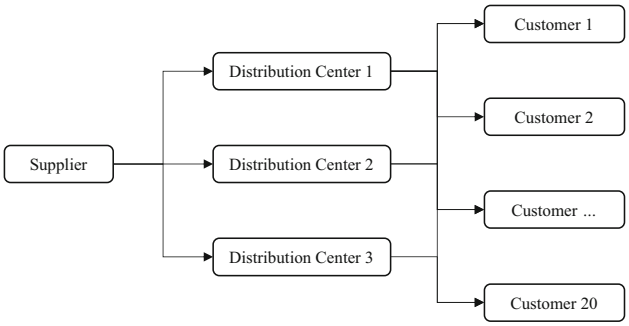


Fig. 4 Supply chain structure

iteration t, a mutant vector $\mathbf{v}_{k,t}$ is generated from $\mathbf{x}_{v,t}$ using the allocated mutation strategy h. If $\mathbf{v}_{k,t}$ violates the range of inventory level, Eq. (18) is used to reset $\mathbf{v}_{k,t}$. Then, the binomial crossover is applied to generate $\mathbf{u}_{k,t}$ by Eq. (16). To generate feasible solutions that satisfy the constraints (6), the repair strategy introduced in Section "Constraint-handling techniques" swap the values of $x_{i,k,t}$ and $x_{i+n,k,t}$ if $x_{i,k,t} > x_{i+n,k,t}$ and values of $u_{i,k,t}$ and $u_{i+n,k,t}$ if $u_{i,k,t} > u_{i+n,k,t}$. The total costs and service level of $\mathbf{x}_{k,t}$ and $\mathbf{u}_{k,t}$ are estimated by the surrogate model M_c and M_r. The selection operation in Algorithm 2 is then conducted to update the population.

After all the solutions in the population K are updated, we calculate the solution improvement $\Delta f_{k,h,t}$ for $k \in K$ using Eqs. (19–23). If the current generation number t is divisible by G, we update the adoption rate $p_{h,g}$ using Eqs. (24–27) for $h \in H$. Then, the roulette wheel selection is conducted to assign a strategy h in the strategy set H to the solution set K based on the adoption rate $p_{h,g}$. The algorithm will terminate if the stop criteria are met.

Experiments

Simulation model

In this study, a typical three-echelon supply chain is considered. This supply chain involves a supplier, three distribution centers, and twenty customers. The supply chain structure is shown in Fig. 4.

The supply chain digital twin is developed by anyLogistix software. It allows simulating the actual supply chain on the GIS map in real-time. We build the supply chain model based on the tutorial example of anyLogistix [63]. Consider the supplier is in Los Angeles, three distribution centers are in Wilkes-Barre, Vicksburg, and Elko, and twenty customers are in different cities. The three-echelon supply chain simulation model on the GIS map developed by anyLogistix is shown in Fig. 5.

The supplier provides products and sells those items to the distribution centers. Then the distribution centers send the products to the customers based on the order quantities. We assume the supplier has unlimited inventory that can always fulfill the orders from the distribution centers. This assumption is commonly used in supply chain simulations [7, 10]. Thus, the supplier's inventory management is not considered.

The distribution centers control their inventory levels by the (s, S) policies. A distribution center orders products from the supplier when its inventory level is lower than the replenishment point s, and S means the order-up-to-level. Before the simulation, each distribution center has a certain level of initial stock. The inventory carrying cost is set at 0.01 USD per product, per day. The shipment processing costs occur when a distribution center receives or sends a shipment. Table 1 shows the shipment processing costs and decision variables for the three distribution centers.

Fig. 5 Three-echelon supply chain digital twin on the GIS map developed by anyLogistix

Table 1 Processing costs and decision variables

Distribution center	Location	Processing cost (USD, per product)	(s, S) policy
Distribution center 1	Wilkes-Barre	0.52	(x_1, x_4)
Distribution center 2	Vicksburg	0.58	(x_2, x_5)
Distribution center 3	Elko	0.56	(x_3, x_6)

Table 2 Demand data of 20 customers

Customer	Location	Order interval (day)	Order quantity
Customer 1	Washington DC	5	118
Customer 2	Phoenix	2	108
Customer 3	Dallas	5	225
Customer 4	Boston	4	80
Customer 5	Nashville	3	68
Customer 6	Orlando	2	18
Customer 7	San Antonio	3	152
Customer 8	Denver	3	70
Customer 9	Seattle	2	47
Customer 10	Philadelphia	3	165
Customer 11	Los Angeles	1	138
Customer 12	Chicago	3	278
Customer 13	Detroit	2	48
Customer 14	Jacksonville	1	30
Customer 15	San Jose	5	179
Customer 16	Kansas City	5	83
Customer 17	Atlanta	3	48
Customer 18	New York	1	300
Customer 19	Salt Lake City	5	34
Customer 20	San Diego	1	50

A deterministic demand model is considered in this simulation. Customers periodically order a certain quantity of products during a certain time. Customers have different order quantities and periods. The demand data of 20 customers are summarized in Table 2. The expected lead time is set to 30 days. The backorder policy is not allowed. The order is canceled if the products are not shipped within the expected lead time. The number of successful orders is recorded to calculate the service level in Eq. (2).

Transportation costs are calculated based on the number of shipments and the distance between the starting location and the ending location. The travel distance is calculated according to the GIS map as shown in Fig. 5. Also, the truck speed is set to 50 km/h. The FIFO (first-in, first-out) policy is used to decide the order priority. The planning horizon is set to 365 days.

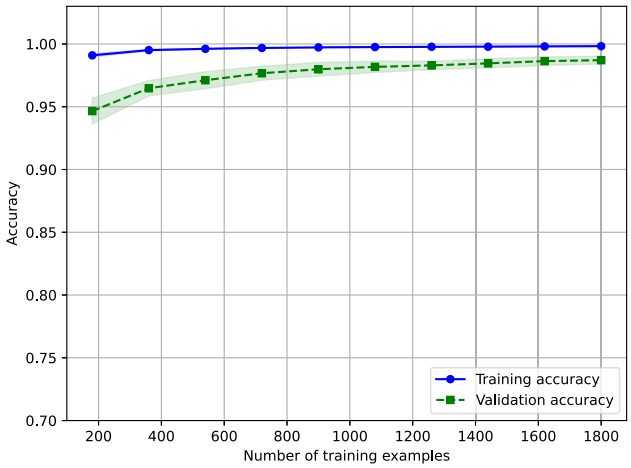


Fig. 6 Accuracy of the random forest algorithm for predicting the total cost

Experimental setting

The proposed data-driven supply chain optimization framework is applied to the three-echelon supply chain digital twin, which is developed by anyLogistix supply chain software. The data-driven evolutionary computation methods are coded in Python 3.7. All the simulation and optimization are executed on a computer with Intel Core i7-8565U CPU (1.80 GHz), 8 GB RAM, and Windows 11 operating system. In this supply chain, the initial stock is set to 1000, and the capacity b_i is set to 3000 products for all the facilities. To comprehensively evaluate the performance of the data-driven evolutionary algorithms, the computational experiments are conducted under different service level constraints including 0.95, 0.94, and 0.93.

We generated 2000 sampling points from the supply chain model and applied the random forest algorithm to construct the surrogate models. We conduct tenfold cross-validation to examine the accuracy of the random forest algorithm. The coefficient of determination (R^2) is used to reflect the accuracy of the surrogate models. The learning curves of the random forest algorithm are shown in Figs. 6 and 7. The random forest algorithm performs well for both surrogate models of total cost and service level. The accuracies are achieved more than 0.95 when the number of samples in the training data is increased to 400.

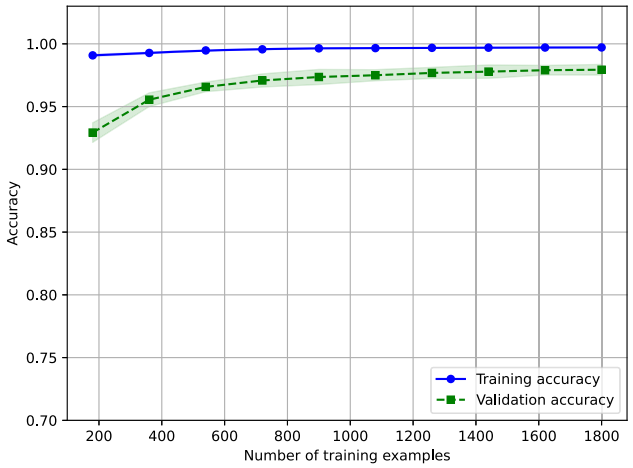


Fig. 7 Accuracy of the random forest algorithm for predicting the service level

Parameter tuning

Given that the proposed method is a versatile framework, it can be easily adapted to incorporate other evolutionary algorithms. In this paper, PSO, DE/rand/1, DE/best/2, DE/current-to-pbest/1, and the proposed EDE are implemented as the optimizer. The constraints handling methods in Section "Constraint-handling techniques" are applied to all these algorithms. We chose PSO and DE as our optimizers based on their proven effectiveness in previous research. The surrogate models have been widely applied to PSO in previous studies [44, 45, 49, 64–66]. Experimental results show that a surrogate model can accelerate the PSO search on various problems and proved to be efficient compared with other algorithms [18]. DE has been attracting considerable interest since it was proposed by Storn and Price [67]. In a recent study [58], the performance of fifteen metaheuristic algorithms and a random blind search was investigated on both benchmark test suite and real-world optimization problems. Their results demonstrated the superiority of DE and its variants on these problems. Several recent studies indicated that surrogate-assisted DE was efficient for expensive problems [25, 42, 43, 68].

The PSO algorithm is developed by Kennedy and Eberhart [69] and has become one of the most commonly used population-based algorithms. Since then, although many PSO variants have been proposed, the linearly decreasing inertia weight PSO algorithm [70] is known as the canonical PSO algorithm [71, 72]. The canonical PSO algorithm is implemented in this study.

The original PSO algorithm has three important parameters including inertia weight ω and two acceleration constants c_1 and c_2. By decreasing the value of inertia weight from $\omega_{\max}$ to $\omega_{\min}$ during the computation, the exploration and exploitation ability of PSO can be balanced and it is expected to have better performance. The inertia weight ω can take values between 0 and 1, and acceleration constants c_1 and c_2 commonly take values around 2 [56].

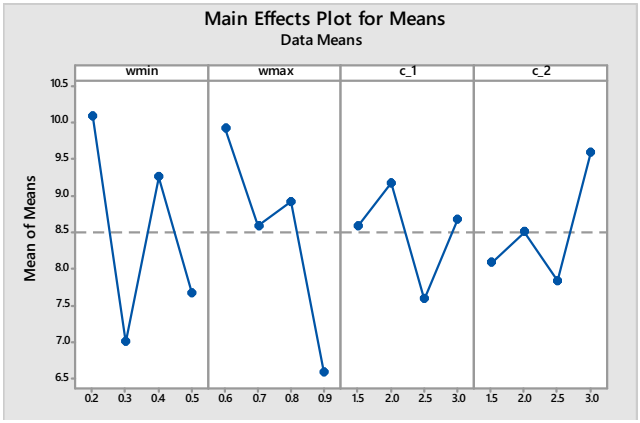
Table 3 The level values of parameters and the designs for five algorithms

Algorithms	Designs	Parameters	Level values
PSO	L16 (4^4)	ω_{min}	0.2, 0.3, 0.4, 0.5
		$\omega_{\max}$	0.6, 0.7, 0.8, 0.9
		c_1	1.5, 2.0, 2.5, 3.0
		c_2	1.5, 2.0, 2.5, 3.0
DE/rand/1	L16 (4^2)	F	0.6, 0.7, 0.8, 0.9
		C_r	0.1, 0.3, 0.5, 0.7
DE/best/2	L16 (4^2)	F	0.6, 0.7, 0.8, 0.9
		C_r	0.1, 0.3, 0.5, 0.7
DE/current-to-pbest/1	L16 (4^3)	F	0.6, 0.7, 0.8, 0.9
		C_r	0.1, 0.3, 0.5, 0.7
		p	0.05, 0.1, 0.2, 0.3
EDE	L16 (4^5)	F	0.6, 0.7, 0.8, 0.9
		C_r	0.1, 0.3, 0.5, 0.7
		p	0.05, 0.1, 0.2, 0.3
		G	1, 5, 10, 20
		γ	0.3, 0.5, 0.7, 0.9

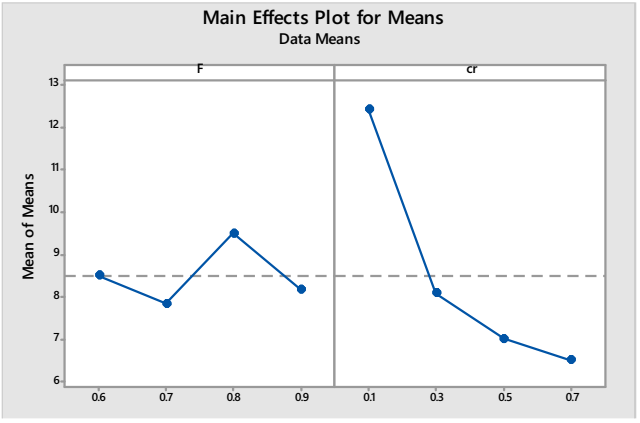
In the DE algorithms, the scaling factor F can be taken values in the interval of $[0,\ 2]$. In practice, $F \in [0.7,\ 0.9]$ is a good first choice [56]. The parameter C_r is the crossover rate in the interval of $[0,\ 1]$. In DE/current-to-pbest/1, The parameter p can take values in the range $(0,\ 1]$. The smaller the p, the greedier the algorithm behaves. In the original paper [60], the parameter p is set to 0.05. In EDE, the learning period G controls the learning speed. With a large G, the adoption rates update rapidly and vice versa. The decay parameter γ within the range $[0,\ 1]$.

The orthogonal array, also known as Taguchi design, is used to estimate the effects of parameters on the performance of these algorithms and tune their parameters. Table 3 shows the level values of parameters and the designs for five algorithms. The levels of parameters are set based on the previous studies [56, 60].

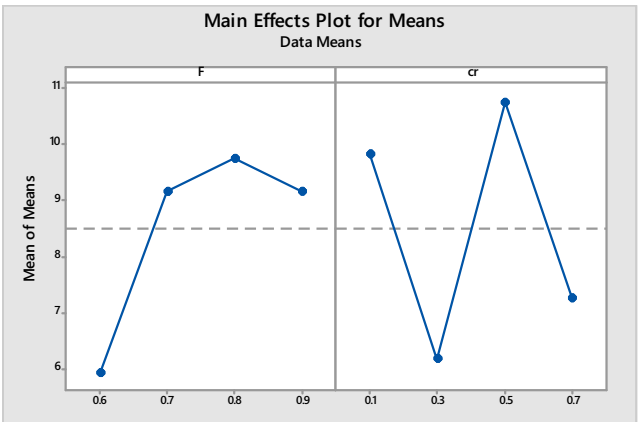
Table 3 shows the designs for different algorithms. For example, L16(4^2) means 16 experiments, and 4 factors with 2 levels each. To conduct a fair competition, L16 is set for all the algorithms for parameter tunning. We examine the performance of a parameter setting 10 times independently on a service level. The rank of each parameter design is calculated based on the total costs. When the service level constraint is violated, we multiply the constraint by a penalty value and add the result to the total costs. In this study, the penalty value is set to 1E + 08. The number of generations is set to 500 and the population size is 60 for all the algorithms. The main effects for means are depicted in Fig. 8.



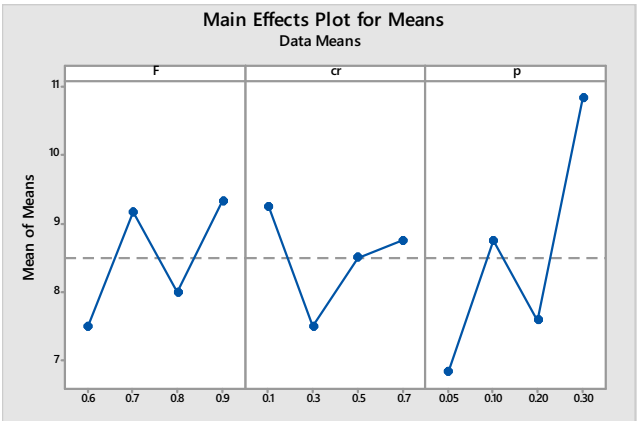
(a) PSO



(b) DE/rand/1



(c) DE/best/2



(d) DE/current-to-pbest/1

(e) EDE

Fig. 8 Main effects plot for means of five algorithms

According to the results in Fig. 8, the appropriate parameter settings are obtained as shown in Table 4.

Experimental results

To compare the performance of PSO, DE/rand/1, DE/best/2, DE/current-to-pbest/1, and EDE on the surrogate models, these evolutionary algorithms are used to search the optimal (s, S) inventory policy based on collected data samples. The parameter settings for these algorithms are shown in Table 4. Each algorithm runs 30 times independently on a service level constraint.

Table 5 shows the mean costs and the mean constraint violations obtained by these five algorithms on the surrogate

Table 4 Parameter settings for PSO, DE/rand/1, DE/best/2, DE/current-to-pbest/1, and EDE

Algorithms	Parameter settings
PSO	$\omega_{\min} = 0.3, \omega_{\max} = 0.9,$ $c_1 = 2.5, c_2 = 2.5$
DE/rand/1	$F = 0.7, C_r = 0.7,$
DE/best/2	$F = 0.9, C_r = 0.1$
DE/current-to-pbest/1	$F = 0.6, C_r = 0.3, p = 0.05$
EDE	$F = 0.9, C_r = 0.7, G = 20,$ $p = 0.05, \gamma = 0.5$

models. Note the obtained costs are the predicted values of the surrogate models. All the evolutionary algorithms can provide feasible solutions, except that PSO fails to obtain a feasible solution 5 times out of 30 experiments when the service level is set to 0.95. Furthermore, EDE obtained the minimum mean costs when the service level is set to 0.94 and 0.93, and EDE is the second-best when the service level is 0.95.

The t-test at 0.05 significance level is conducted to examine the significant difference between the total cost obtained by EDE and the other algorithms. The obtained p-value is shown in Table 6. Since PSO obtained infeasible solutions when the service level is 0.95, the p-value is not shown in this table. In the last row, the symbols " + ", and "−" represent that EDE is significantly better, worse than the other algorithms, and " = " represents that there is no significant difference. The results suggest that the EDE algorithm outperforms PSO in all experimental scenarios with statistical significance, and demonstrates superior performance compared to the other three DE algorithms in two experiments, while achieving similar performance in one experiment.

As shown in Table 7, the average rank of EDE is 1.33, while the other algorithms are larger than 2. Overall, EDE ranks first on the surrogate models.

Further experiments are conducted on the anyLogistix software to examine the efficiency of the data-driven methods. The results obtained by evolutionary algorithms are verified on the supply chain digital twin. Table 8 shows the experimental results of the supply chain simulation model. Table 8 reveals that there have been differences from Table 5 which is caused by the prediction errors of the surrogate models. In the last two rows of Table 8, the mean squared error (MSE) is calculated to show the prediction errors of the surrogate models.

Among the 30 experiments, the best results obtained by the five algorithms on the supply chain digital twin, and the

Table 5 Experiment results on the surrogate models

α	Criteria	PSO	DE/rand/1	DE/best/2	DE/current-to-pbest/1	EDE
0.95	Cost	1.2120E+08	1.2241E+08	1.2203E+08	1.2185E+08	1.2201E+08
	Violation	4.9587E−03	0	0	0	0
0.94	Cost	1.1945E+08	1.1912E+08	1.1911E+08	1.1911E+08	1.1900E+08
	Violation	0	0	0	0	0
0.93	Cost	1.1601E+08	1.1534E+08	1.1546E+08	1.1540E+08	1.1519E+08
	Violation	0	0	0	0	0

Table 6 p-value of the t-test between the total cost obtained by EDE and the other algorithms.

α	PSO	DE/rand/1	DE/best/2	DE/current-to-pbest/1
0.95	–	1.433E−02	4.482E−01	1.437E−01
0.94	9.589E−12	2.523E−02	2.932E−02	2.007E−02
0.93	2.542E−09	1.804E−01	1.904E−02	3.500E−02
+ / = /−	3/0/0	2/1/0	2/1/0	2/1/0

Table 7 The ranks of the five algorithms on the surrogate models

α	PSO	DE/rand/1	DE/best/2	DE/current-to-pbest/1	EDE
0.95	5	4	3	1	2
0.94	5	4	2	3	1
0.93	5	2	4	3	1
Ave. rank	5.00	3.33	3.00	2.33	1.33
Final rank	5	4	3	2	1

Table 8 Experiment results on supply chain simulation model

α	Criteria	PSO	DE/rand/1	DE/best/2	DE/current-to-pbest/1	EDE
0.95	Cost	1.2892E+08	1.2285E+08	1.2762E+08	1.2251E+08	1.2423E+08
	Violation	4.6866E−03	3.1637E−03	8.1722E−04	3.6269E−03	2.3426E−03
0.94	Cost	1.1936E+08	1.1900E+08	1.1905E+08	1.1931E+08	1.1903E+08
	Violation	2.0050E−03	2.9440E−03	2.9505E−03	2.8113E−03	2.7357E−03
0.93	Cost	1.1658E+08	1.1618E+08	1.1620E+08	1.1621E+08	1.1612E+08
	Violation	1.6779E−02	1.6148E−02	1.5662E−02	1.5724E−02	1.7850E−02
MSE (cost)		4.2366E+13	2.6516E+12	6.9923E+13	8.5796E+11	6.3502E+12
MSE (service level)		3.3163E−04	1.0952E−04	7.5467E−04	1.5703E−04	1.0344E−04

best results of the collected data samples are shown in Table 9. The best result is defined as the feasible solution that has the lowest cost. What can be clearly seen in this table is that the DE/rand/1, DE/best/2, and EDE can obtain lower costs than the supply chain date samples while satisfying the required service levels for all the experiments. PSO and DE/current-to-pbest/1 also obtained a better solution than the dataset when $\alpha = 0.94$ and $\alpha = 0.93$.

Overall, this section has provided the experimental results obtained by PSO, DE/rand/1, DE/best/2, DE/current-to-pbest/1, and EDE on surrogate models and on the supply chain simulation model. The results indicate that the data-driven evolutionary algorithms can provide even better results than the historical data in most situations. Also, our proposed EDE algorithm performs better than the other algorithms on the surrogate models. The following section will discuss the online method to improve the quality of solutions obtained by the proposed data-driven evolutionary algorithm.

Experiments results using additional data

While the solutions obtained in the previous section are satisfactory when compared to the training data, it is important to note that reducing prediction errors may result in improved solution quality. As shown in Table 8, the presence of such errors may cause data-driven evolutionary algorithms to generate infeasible solutions for the simulation model.

The experiments in the previous section are conducted without any new data. It is interesting to discuss the possibility of further improving the current solutions by adding new data to the surrogate models. This is especially useful in real-world inventory management as the algorithm can generate new inventory policy as new training data arrives. Even if the train data contains noises, the algorithm can adapt to changes of the environment using this online learning method. To efficiently improve the accuracy of the surrogate models, optimized solutions are added to the training data. The pseudocode of the online data-driven EDE algorithm is shown in Algorithm 3.

Algorithm 3: Online Data-Driven Evolutionary Algorithm

Input: Number of solutions n, training dataset D, service level α
Output: inventory policy $\mathbf{x}^*$, surrogate model M_c and M_r

while the stopping criterion is not met **do**
  use D to generate surrogate model M_c and M_r by Algorithm 1
  for $i = 1$ to n **do**
    execute the evolutionary algorithm using M_c and M_r and obtain an inventory policy $\mathbf{x}_i^*$
    evaluate the total cost $c(\mathbf{x}_i^*)$ and service level $r(\mathbf{x}_i^*)$ on the simulation model
  end for
  $D \leftarrow D \cup \left(\left(\mathbf{x}_1^*, c(\mathbf{x}_1^*), r(\mathbf{x}_1^*)\right), \ldots, \left(\mathbf{x}_n^*, c(\mathbf{x}_n^*), r(\mathbf{x}_n^*)\right)\right)$
end while
output the best feasible solution $\mathbf{x}^*$ and the improved surrogate model M_c and M_r

Table 9 Best results obtained by the algorithms, and the best results of the collected data samples on the supply chain simulation model under different service level constraints

α	Algorithms	Optimal inventory policy						Total costs	Service level (%)
		x_1	x_2	x_3	x_4	x_5	x_6		
0.95	PSO	1458	1472	1992	2492	1746	2132	1.2367E+08	95.00
	DE/rand/1	1452	800	672	2482	1151	672	1.2334E+08	95.06
	DE/best/2	1443	1854	1093	2493	2039	2273	1.2346E+08	95.00
	DE/current-to-pbest/1	1446	1604	466	2494	2738	2264	1.2391E+08	95.00
	EDE	1495	883	998	2311	1160	1575	1.2356E+08	95.30
	Data	1443	855	1433	2464	1667	2155	1.2360E+08	95.00
0.94	PSO	1447	764	660	1640	879	1096	1.1986E+08	94.05
	DE/rand/1	1456	676	699	1649	1710	755	1.1981E+08	94.05
	DE/best/2	1446	697	632	1650	815	1537	1.1998E+08	94.05
	DE/current-to-pbest/1	1450	692	598	1643	1975	668	1.1975E+08	94.05
	EDE	1446	790	676	1649	868	678	1.1976E+08	94.05
	Data	1321	1048	972	2213	1380	1518	1.2072E+08	94.14
0.93	PSO	1213	698	495	1558	812	2586	1.1687E+08	93.13
	DE/rand/1	1222	747	573	1552	1054	835	1.1659E+08	93.13
	DE/best/2	1225	1033	517	1558	1500	1688	1.1681E+08	93.13
	DE/current-to-pbest/1	1209	741	515	1538	798	1810	1.1649E+08	93.13
	EDE	1200	694	1437	1542	799	1646	1.1676E+08	93.13
	Data	1309	1458	722	1635	1964	1333	1.1695E+08	93.13

Table 10 Best results obtained by EDE on the supply chain simulation model

α	Optimal inventory policy						Total costs	Service level (%)
	x_1	x_2	x_3	x_4	x_5	x_6		
0.95	1436	1007	950	2287	1108	963	1.2346E+08	95.30
0.94	1448	680	585	1637	828	636	1.1970E+08	94.05
0.93	1205	761	818	1542	891	2015	1.1656E+08	93.13

The results in the previous section indicate that EDE has better performance than the other algorithms on the surrogate models. Thus, EDE's performance is analyzed by this method. In this experiment, Algorithm 3 is applied to three service levels, 0.95, 0.94, and 0.93. For each required service level, the termination condition is set to 5 iterations, and number of solutions n is set to 20. Therefore, 100 optimized data are added to the surrogate model for each required service level. The other parameters are set the same as in the previous section. The obtained best results are shown in Table 10.

As can be seen from the table above, the best results obtained by EDE after adding new training data are all improved compared with the results in Table 9. Furthermore, we execute EDE 30 times on service level 0.95, 0.94, and 0.93 using the improved surrogate model M_c and M_r that are generated by Algorithm 3. The obtained MSE for cost is 9.55E + 11, and the MSE for service level is 3.41E-05, which are lower than the MSE results obtained by the offline EDE in Table 8.

Discussion

Recent studies [10, 13, 73] have shown that important insights can be obtained from the implementation of the supply chain digital twins. Although the (*s, S*) inventory policy is widely used in the studies of supply chain digital twins, few studies discuss the optimization of the supply chain digital twins. To fill this research gap, this study proposed an efficient data-driven evolutionary algorithm for the service level constrained inventory optimization problem in supply chains. This section discusses the experimental results and provides some managerial insights.

First, this study found that the data-driven evolutionary algorithms can provide better solutions than the training data while satisfying the service level constraints. Furthermore, the obtained inventory policies are very different from the best historical data. As Table 9 shows, there are significant differences between the inventory policies of best historical data and the optimized results. These findings suggest that our proposed method does not only search the neighbor solutions around the best historical data but learns from the historical data and generates new inventory policies. Recent developments in the field of IoT technologies have enabled supply chain systems to trace and record the information of the whole supply chain. Our method can be used as a decision support system for supply chain management with historical data that is collected by IoT technologies.

Second, a general formulation for the service constrained inventory optimization problem is proposed, which involves three different types of constraints. A regeneration method is used to generate new inventory policies when the possible range of inventory levels is violated. Also, a repairing method is proposed to repair the infeasible (*s, S*) policies where the re-order level is larger than the order-up-to level. Finally, the superiority of feasible solutions is used to deal with the service level constraints. The experimental results in Table 5 show that all the DE algorithms can generate feasible solutions for the surrogate models, which means these constraint-handling techniques are effective. However, since the prediction error is inevitable, some feasible solutions in surrogate models may not satisfy the constraints in the simulation model. As shown in Table 8, the mean constraint violations for all the algorithms are larger than 0. In practice, reducing the number of surrogate models is important to improve the accuracy of estimation.

Third, the surrogate models are used to approximate the objective function and the constraints by using the training data in this study. The landscapes of the surrogate models are varying stem from the changes in the training data. An ensemble approach-based DE algorithm (EDE) is proposed to deal with the varied surrogate models. The experimental results in Table 5 suggest that the EDE algorithm obtained better mean results than PSO on the surrogate models in all the situations, and demonstrates superior performance compared to the other three DE algorithms in two scenarios, while achieving similar performance in one scenario. These findings suggest that the EDE algorithm performs well for this problem. Also, the ensemble approach-based algorithms are promising for optimizing the surrogate models. We have also examined the performance of online data-driven evolutionary optimization on this problem. The experimental results show that the performance of the proposed data-driven evolutionary algorithm can be improved by iteratively adding optimized solutions to the learning surrogate models. The online EDE algorithm achieved a significant improvement in accuracy of surrogate models compared to the offline EDE algorithm, with MSE of cost being reduced by 84.96% and the MSE of service level by 67.06%. On average, the online EDE algorithm has led to a 0.1% reduction in cost compared to the offline EDE algorithm.

Conclusion

This study has proposed a data-driven evolutionary algorithm to optimize the (*s, S*) inventory policy of supply chain digital twins. A general formulation for the service constrained inventory optimization problem has been proposed. The objective is to minimize the total costs and satisfy the required service level. The supply chain simulation model has been developed by anyLogistix software, which can be used to evaluate the total costs and service level. The random forest algorithm is used to construct surrogate models for approximating the objective function and service level constraints using the collected supply chain data. An ensemble approach-based DE algorithm (EDE) is proposed as the optimizer for the data-driven evolutionary algorithm. The proposed approach is examined by a three-echelon supply chain digital twin with a supplier, three distribution centers, and twenty customers. The experimental results suggest that the proposed method can reduce the total cost by 0.33% on average while maintaining the required service levels compared with the historical data, which is better than PSO and DE. Also, the solutions can be further improved by iteratively adding optimized solutions to the surrogate models.

This research has several practical applications. This research has offered a framework for efficiently optimizing supply chains using a data-driven approach. It should prove to be particularly valuable to solving the optimization problems of supply chain digital twins in practice. Furthermore, the recent developments in IoT technologies have heightened the need for data-driven optimization in supply chain management. This study lays the groundwork for future research into data-driven optimization for supply chain management based on big data collected by IoT technologies.

Several limitations and future research directions to the present study need to be noted. First, this study assumed that the inventory policy for a facility stays the same during a simulation period. It would be interesting to consider that a facility can use different inventory policies for different situations. In the future, it will be important to explore the potential use of reinforcement learning for such problems. Second, a supply chain digital twin is used to generate the training data in this study. In the future, it will be important to explore the potential use of data-driven evolutionary optimization based on real data collected by IoT technologies for actual supply chains. Third, A deterministic demand model is considered in this study. In the future, it will be important

to explore inventory optimization problems with stochastic demand. Fourth, a limitation of this study is that we only focused on the inventory optimization problem for a supply chain. Further work is needed to explore the usefulness of data-driven evolutionary optimization for other supply chain problems such as supplier selection, and routing problems.

Funding This work was supported by JSPS KAKENHI Grant Numbers JP22H01714, JP23K13514.

Data availability The data that support the findings of this study are openly available on github at https://github.com/zi-ang-liu/DDEA-EDE

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Chopra S, Meindl P (2016) Supply chain management: strategy, planning, and operation, 6th edn. Pearson, Boston
2. Singh D, Verma A (2018) Inventory management in supply chain. 7th Int Conf Mater Process Charact March 17–19 2017 5:3867–3872. https://doi.org/10.1016/j.matpr.2017.11.641
3. You F, Grossmann IE (2008) Mixed-integer nonlinear programming models and algorithms for large-scale supply chain design with stochastic inventory management. Ind Eng Chem Res 47:7802–7817. https://doi.org/10.1021/ie800257x
4. Dillon M, Oliveira F, Abbasi B (2017) A two-stage stochastic programming model for inventory management in the blood supply chain. Int J Prod Econ 187:27–41. https://doi.org/10.1016/j.ijpe.2017.02.006
5. Duan L, Ventura JA (2019) A dynamic supplier selection and inventory management model for a serial supply chain with a novel supplier price break scheme and flexible time periods. Eur J Oper Res 272:979–998. https://doi.org/10.1016/j.ejor.2018.07.031
6. Qiu Y, Qiao J, Pardalos PM (2019) Optimal production, replenishment, delivery, routing and inventory management policies for products with perishable inventory. Omega 82:193–204. https://doi.org/10.1016/j.omega.2018.01.006
7. Chu Y, You F, Wassick JM, Agarwal A (2015) Simulation-based optimization framework for multi-echelon inventory systems under uncertainty. Comput Chem Eng 73:1–16. https://doi.org/10.1016/j.compchemeng.2014.10.008
8. Wan X, Pekny JF, Reklaitis GV (2005) Simulation-based optimization with surrogate models—application to supply chain management. Comput Chem Eng 29:1317–1328. https://doi.org/10.1016/j.compchemeng.2005.02.018
9. Noordhoek M, Dullaert W, Lai DSW, de Leeuw S (2018) A simulation–optimization approach for a service-constrained multi-echelon distribution network. Transp Res Part E Logist Transp Rev 114:292–311. https://doi.org/10.1016/j.tre.2018.02.006
10. Aldrighetti R, Zennaro I, Finco S, Battini D (2019) Healthcare supply chain simulation with disruption considerations: a case study from northern Italy. Glob J Flex Syst Manag 20:81–102. https://doi.org/10.1007/s40171-019-00223-8
11. Ivanov D (2020) Predicting the impacts of epidemic outbreaks on global supply chains: a simulation-based analysis on the coronavirus outbreak (COVID-19/SARS-CoV-2) case. Transp Res Part E Logist Transp Rev 136:101922. https://doi.org/10.1016/j.tre.2020.101922
12. Marmolejo-Saucedo JA (2020) Design and development of digital twins: a case study in supply chains. Mob Netw Appl 25:2141–2160. https://doi.org/10.1007/s11036-020-01557-9
13. Burgos D, Ivanov D (2021) Food retail supply chain resilience and the COVID-19 pandemic: a digital twin-based impact analysis and improvement directions. Transp Res Part E Logist Transp Rev 152:102412. https://doi.org/10.1016/j.tre.2021.102412
14. Gerlach B, Zarnitz S, Nitsche B, Straube F (2021) Digital supply chain twins—conceptual clarification, use cases and benefits. Logistics. https://doi.org/10.3390/logistics5040086
15. Ivanov D, Dolgui A (2021) A digital supply chain twin for managing the disruption risks and resilience in the era of Industry 4.0. Prod Plan Control 32:775–788. https://doi.org/10.1080/09537287.2020.1768450
16. Singh S, Kumar R, Panchal R, Tiwari MK (2021) Impact of COVID-19 on logistics systems and disruptions in food supply chain. Int J Prod Res 59:1993–2008. https://doi.org/10.1080/00207543.2020.1792000
17. Defraeye T, Shrivastava C, Berry T et al (2021) Digital twins are coming: will we need them in supply chains of fresh horticultural produce? Trends Food Sci Technol 109:245–258. https://doi.org/10.1016/j.tifs.2021.01.025
18. Jin Y, Wang H, Sun C (2021) Data-driven evolutionary optimization: integrating evolutionary computation, machine learning and data science. Springer, Cham
19. Chugh T, Chakraborti N, Sindhya K, Jin Y (2017) A data-driven surrogate-assisted evolutionary algorithm applied to a many-objective blast furnace optimization problem. Mater Manuf Process 32:1172–1178. https://doi.org/10.1080/10426914.2016.1269923
20. Wang H, Jin Y, Jansen JO (2016) Data-driven surrogate-assisted multiobjective evolutionary optimization of a trauma system. IEEE Trans Evol Comput 20:939–952. https://doi.org/10.1109/TEVC.2016.2555315
21. Wang H, Jin Y (2020) A random forest-assisted evolutionary algorithm for data-driven constrained multiobjective combinatorial optimization of trauma systems. IEEE Trans Cybern 50:536–549. https://doi.org/10.1109/TCYB.2018.2869674
22. Wang H, Jin Y, Sun C, Doherty J (2019) Offline data-driven evolutionary optimization using selective surrogate ensembles. IEEE Trans Evol Comput 23:203–216. https://doi.org/10.1109/TEVC.2018.2834881
23. Li J-Y, Zhan Z-H, Wang H, Zhang J (2021) Data-driven evolutionary algorithm with perturbation-based ensemble surrogates. IEEE Trans Cybern 51:3925–3937. https://doi.org/10.1109/TCYB.2020.3008280
24. Tang Y, Chen J, Wei J (2013) A surrogate-based particle swarm optimization algorithm for solving optimization problems with expensive black box functions. Eng Optim 45:557–576. https://doi.org/10.1080/0305215X.2012.690759
25. Wang W, Liu H-L, Tan KC (2023) A surrogate-assisted differential evolution algorithm for high-dimensional expensive optimization

problems. IEEE Trans Cybern 53:2685–2697. https://doi.org/10.1109/TCYB.2022.3175533
26. Tako AA, Robinson S (2012) The application of discrete event simulation and system dynamics in the logistics and supply chain context. Decis Support Syst 52:802–815. https://doi.org/10.1016/j.dss.2011.11.015
27. Prinz R, Väätäinen K, Laitila J et al (2019) Analysis of energy efficiency of forest chip supply systems using discrete-event simulation. Appl Energy 235:1369–1380. https://doi.org/10.1016/j.apenergy.2018.11.053
28. Lee J-H, Kim C-O (2008) Multi-agent systems applications in manufacturing systems and supply chain management: a review paper. Int J Prod Res 46:233–265. https://doi.org/10.1080/00207540701441921
29. Dai H, Lin J, Long Q (2014) A fractal perspective-based methodological framework for supply chain modelling and distributed simulation with multi-agent system. Int J Prod Res 52:6819–6840. https://doi.org/10.1080/00207543.2014.919414
30. Nishi T, Matsuda M, Hasegawa M et al (2020) Automatic construction of virtual supply chain as multi-agent system using enterprise E-catalogues. Int J Autom Technol 14:713–722. https://doi.org/10.20965/ijat.2020.p0713
31. Matsuda M, Nishi T, Kamiebisu R et al (2021) Use of virtual supply chain constructed by cyber-physical systems concept. Procedia CIRP 104:351–356. https://doi.org/10.1016/j.procir.2021.11.059
32. Kamiebisu R, Saso T, Nakao J et al (2022) Use cases of the platform for structuring a smart supply chain in discrete manufacturing. Procedia CIRP 107:687–692. https://doi.org/10.1016/j.procir.2022.05.046
33. Busse A, Gerlach B, Lengeling JC et al (2021) Towards digital twins of multimodal supply chains. Logistics. https://doi.org/10.3390/logistics5020025
34. Timperio G, Tiwari S, Gaspar Sánchez JM et al (2020) Integrated decision support framework for distribution network design. Int J Prod Res 58:2490–2509. https://doi.org/10.1080/00207543.2019.1680894
35. Jin Y, Wang H, Chugh T et al (2019) Data-driven evolutionary optimization: an overview and case studies. IEEE Trans Evol Comput 23:442–458. https://doi.org/10.1109/TEVC.2018.2869001
36. Mazumdar A, Chugh T, Hakanen J, Miettinen K (2022) Probabilistic selection approaches in decomposition-based evolutionary algorithms for offline data-driven multiobjective optimization. IEEE Trans Evol Comput 26:1182–1191. https://doi.org/10.1109/TEVC.2022.3154231
37. Liu Z, Wang H, Jin Y (2022) Performance indicator-based adaptive model selection for offline data-driven multiobjective evolutionary optimization. IEEE Trans Cybern. https://doi.org/10.1109/TCYB.2022.3170344
38. Huang H-G, Gong Y-J (2023) Contrastive learning: an alternative surrogate for offline data-driven evolutionary computation. IEEE Trans Evol Comput 27:370–384. https://doi.org/10.1109/TEVC.2022.3170638
39. Yang C, Ding J, Jin Y, Chai T (2020) Off-line data-driven multiobjective optimization: knowledge transfer between surrogates and generation of final solutions. IEEE Trans Evol Comput. https://doi.org/10.1109/TEVC.2019.2925959
40. Guo D, Chai T, Jinliang Ding, Jin Y (2016) Small data driven evolutionary multi-objective optimization of fused magnesium furnaces. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, Athens, Greece, pp 1–8
41. Zhen H, Gong W, Wang L et al (2023) Two-stage data-driven evolutionary optimization for high-dimensional expensive problems. IEEE Trans Cybern 53:2368–2379. https://doi.org/10.1109/TCYB.2021.3118783
42. Gu H, Wang H, Jin Y (2022) Surrogate-assisted differential evolution with adaptive multi-subspace search for large-scale expensive optimization. IEEE Trans Evol Comput. https://doi.org/10.1109/TEVC.2022.3226837
43. Chen G, Li Y, Zhang K et al (2021) Efficient hierarchical surrogate-assisted differential evolution for high-dimensional expensive optimization. Inf Sci 542:228–246. https://doi.org/10.1016/j.ins.2020.06.045
44. Ji X, Zhang Y, Gong D et al (2023) Multisurrogate-assisted multitasking particle swarm optimization for expensive multimodal problems. IEEE Trans Cybern 53:2516–2530. https://doi.org/10.1109/TCYB.2021.3123625
45. Wang H, Jin Y, Doherty J (2017) Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems. IEEE Trans Cybern 47:2664–2677. https://doi.org/10.1109/TCYB.2017.2710978
46. Ji X, Zhang Y, Gong D, Sun X (2021) Dual-surrogate-assisted cooperative particle swarm optimization for expensive multimodal problems. IEEE Trans Evol Comput 25:794–808. https://doi.org/10.1109/TEVC.2021.3064835
47. Long H, Li P, Gu W (2020) A data-driven evolutionary algorithm for wind farm layout optimization. Energy 208:118310. https://doi.org/10.1016/j.energy.2020.118310
48. Fu C, Dong H, Wang P, Li Y (2022) Data-driven Harris Hawks constrained optimization for computationally expensive constrained problems. Complex Intell Syst. https://doi.org/10.1007/s40747-022-00923-2
49. Song X, Zhang Y, Gong D et al (2023) Surrogate sample-assisted particle swarm optimization for feature selection on high-dimensional data. IEEE Trans Evol Comput 27:595–609. https://doi.org/10.1109/TEVC.2022.3175226
50. Bartz-Beielstein T, Zaefferer M (2017) Model-based methods for continuous and discrete global optimization. Appl Soft Comput 55:154–167. https://doi.org/10.1016/j.asoc.2017.01.039
51. Han L, Wang H (2021) A random forest assisted evolutionary algorithm using competitive neighborhood search for expensive constrained combinatorial optimization. Memetic Comput 13:19–30. https://doi.org/10.1007/s12293-021-00326-9
52. Gu Q, Wang Q, Xiong NN et al (2022) Surrogate-assisted evolutionary algorithm for expensive constrained multi-objective discrete optimization problems. Complex Intell Syst 8:2699–2718. https://doi.org/10.1007/s40747-020-00249-x
53. Raschka S, Mirjalili V (2019) Python machine learning: machine learning and deep learning with Python, scikit-learn, and TensorFlow 2, 3rd edn. Packt, Birmingham
54. Wu G, Mallipeddi R, Suganthan PN et al (2016) Differential evolution with multi-population based ensemble of mutation strategies. Inf Sci 329:329–345. https://doi.org/10.1016/j.ins.2015.09.009
55. Liu Z, Nishi T (2022) Strategy dynamics particle swarm optimizer. Inf Sci 582:665–703. https://doi.org/10.1016/j.ins.2021.10.028
56. Yang X-S (2014) Nature-inspired optimization algorithms, 1st edn. Elsevier, Oxford. https://doi.org/10.1016/C2013-0-01368-0
57. Opara KR, Arabas J (2019) Differential evolution: a survey of theoretical analyses. Swarm Evol Comput 44:546–558. https://doi.org/10.1016/j.swevo.2018.06.010
58. Bujok P, Tvrdík J, Poláková R (2019) Comparison of nature-inspired population-based algorithms on continuous optimisation problems. Swarm Evol Comput 50:100490. https://doi.org/10.1016/j.swevo.2019.01.006
59. Das S, Mullick SS, Suganthan PN (2016) Recent advances in differential evolution—an updated survey. Swarm Evol Comput 27:1–30. https://doi.org/10.1016/j.swevo.2016.01.004
60. Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13:945–958. https://doi.org/10.1109/TEVC.2009.2014613
61. Talbi E-G (2009) Metaheuristics. John Wiley & Sons Inc, Hoboken

62. Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Methods Appl Mech Eng 186:311–338. https://doi.org/10.1016/S0045-7825(99)00389-8
63. Simulation | anyLogistix Help. https://anylogistix.help/tutorial/tutorial-simulation-main.html. Accessed 19 Jan 2023
64. Sun C, Jin Y, Zeng J, Yu Y (2015) A two-layer surrogate-assisted particle swarm optimization algorithm. Soft Comput 19:1461–1475. https://doi.org/10.1007/s00500-014-1283-z
65. Liu Y, Liu J, Jin Y (2022) Surrogate-assisted multipopulation particle swarm optimizer for high-dimensional expensive optimization. IEEE Trans Syst Man Cybern Syst 52:4671–4684. https://doi.org/10.1109/TSMC.2021.3102298
66. Ji X, Zhang Y, He C et al (2023) Surrogate and autoencoder-assisted multitask particle swarm optimization for high-dimensional expensive multimodal problems. IEEE Trans Evol Comput. https://doi.org/10.1109/TEVC.2023.3287213
67. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359. https://doi.org/10.1023/A:1008202821328
68. Cai X, Gao L, Li X, Qiu H (2019) Surrogate-guided differential evolution algorithm for high dimensional expensive problems. Swarm Evol Comput 48:288–311. https://doi.org/10.1016/j.swevo.2019.04.009
69. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of ICNN'95—International Conference on Neural Networks. 4:1942–1948
70. Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360). pp 69–73
71. Wang D, Tan D, Liu L (2018) Particle swarm optimization algorithm: an overview. Soft Comput 22:387–408. https://doi.org/10.1007/s00500-016-2474-6
72. Cheng S, Lu H, Lei X, Shi Y (2018) A quarter century of particle swarm optimization. Complex Intell Syst 4:227–239. https://doi.org/10.1007/s40747-018-0071-2
73. Ivanov D, Dolgui A, Sokolov B (2019) The impact of digital technology and Industry 4.0 on the ripple effect and supply chain risk analytics. Int J Prod Res 57:829–846. https://doi.org/10.1080/00207543.2018.1488086