Cooperative coevolutionary bare-bones particle swarm optimization with function independent decomposition for large-scale supply chain network design with uncertainties

Cooperative Coevolutionary Bare-Bones Particle Swarm Optimization With Function Independent Decomposition for Large-Scale Supply Chain Network Design With Uncertainties

Xin Zhang, *Student Member, IEEE*, Ke-Jing Du, Zhi-Hui Zhan, *Senior Member, IEEE*, Sam Kwong, *Fellow, IEEE*, Tian-Long Gu, and Jun Zhang, *Fellow, IEEE*

***Abstract*—Supply chain network design (SCND) is a complicated constrained optimization problem that plays a significant role in the business management. This article extends the SCND model to a large-scale SCND with uncertainties (LUSCND), which is more practical but also more challenging. However, it is difficult for traditional approaches to obtain the feasible solutions in the large-scale search space within the limited time. This article proposes a cooperative coevolutionary bare-bones particle swarm optimization (CCBBPSO) with function independent decomposition (FID), called CCBBPSO-FID, for a multiperiod three-echelon LUSCND problem. For the large-scale issue, binary encoding of the original model is converted to integer encoding for dimensionality reduction, and a novel FID is designed to efficiently decompose the problem. For obtaining the feasible solutions, two repair methods are designed to repair the infeasible solutions that appear frequently in the LUSCND problem. A step translation method is proposed to deal with the variables out of bounds, and a labeled reposition operator with adaptive probabilities is designed to repair the infeasible solutions that violate the constraints. Experiments are conducted on 405 instances with three different scales. The results show that CCBBPSO-FID has an evident superiority over contestant algorithms.**

Manuscript received June 18, 2019; accepted August 12, 2019. Date of publication September 20, 2019; date of current version September 23, 2020. This work was supported in part by the Outstanding Youth Science Foundation under Grant 61822602, in part by the National Natural Science Foundation of China under Grant 61772207 and Grant 61873097, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, in part by the Guangdong–Hong Kong Joint Innovation Platform under Grant 2018B050502006, and in part by the Hong Kong GRF-RGC General Research Fund 9042489 under Grant CityU 11206317. This article was recommended by Associate Editor Y. Tan. *(Corresponding authors: Zhi-Hui Zhan; Jun Zhang.)*

X. Zhang, K.-J. Du, and Z.-H. Zhan are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, also with the State Key Laboratory of Subtropical Building Science, South China University of Technology, Guangzhou 510006, China, and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, Guangzhou 510006, China (e-mail: zhanapollo@163.com).

S. Kwong is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

T.-L. Gu is with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China.

J. Zhang is with Victoria University, Melbourne, VIC 8001, Australia (e-mail: junzhang@ieee.org).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2019.2937565

***Index Terms*—Bare-bones particle swarm optimization (BBPSO), cooperative coevolution (CC), large-scale supply chain network design under uncertainties (LUSCND).**

I. Introduction

THE SUPPLY CHAIN network (SCN) has attracted increasing attention in different fields, such as production manufacturing [1], [2] and business management [3], [4]. The members of SCN include suppliers, manufacturers, warehouses, distributors, retailers, and customers. These members and the flows of finances and materials among them make up the entire network. The aim of the design of SCN (SCND) is to minimize the total cost of the entire system and to meet demands of different members [5]–[7]. An excellent SCND can optimize enterprise management, improve efficiency, and finally encourage consumptions.

Different SCND models have different features. For example, SCND under uncertainties (USCND) has uncertain factors, such as uncertain demand of customers and uncertain supply of suppliers [8]. These uncertain factors make it difficult to evaluate the solutions and generate feasible solutions. There are two main simulation methods of uncertainties to help the fitness evaluation (FE): 1) the scenario analysis and 2) the Monte Carlo method (MCM). The scenario analysis only analyzes limited scenarios of uncertainties [9]. Differently, MCM simulates the uncertainties in each evaluation of solutions and is more complicated [10]. This article also adopts MCM to the USCND.

The SCND/USCND problem can be regarded as a constrained optimization problem (COP). Due to the great potential of evolutionary algorithms (EAs) in solving COPs, some researchers have already used EAs for SCND problems [11]. Salem and Haouari [10] applied the particle swarm optimization (PSO) to solve a more complex USCND problem with multiple periods and uncertainties in both supply and demand. Sahebjamnia *et al.* [12] used the red deer algorithm (RDA) for the sustainable tire closed-loop SCND problem.

However, it is still difficult for the existing algorithms to effectively solve the large-scale USCND (LUSCND) problem, which involves both the large-scale challenge and the uncertain challenge. On the large-scale issue, a large number of suppliers and customers are included in the LUSCND to meet the growing market demand [13]. Unfortunately, these increasing SCN members lead to a larger solution space and more constraints, making it more difficult to find feasible and optimal solutions. An efficient method for the large-scale issue is in great need to obtain the satisfactory solutions within the limited time. Besides, as the scale of the LUSCND becomes large, the number of constraints will increase sharply, and infeasible solutions may appear frequently in this complex problem. Most of the current works for SCND only apply penalty functions on infeasible solutions, being useless to find a feasible solution for the LUSCND problem. Moreover, in the uncertain environment, the constraints in the LUSCND are also uncertain [10]. These uncertain constraints will also increase the number of constraints. Therefore, it is hard for traditional approaches to obtain feasible solutions of the LUSCND problem. New methods are needed to replace the penalty functions and to help finding the feasible solutions.

Therefore, this article focuses on the practical and challenging LUSCND problem and proposes an efficient approach to solve this problem. On the aspect of the problem model, the USCND in [10] is extended with large-scale variables. On the aspect of the algorithm design, an efficient method for the large-scale issue and two new methods for constraint handling are proposed for the LUSCND problem with complex constraints.

First, to relieve the large-scale challenges, the model is simplified by replacing the binary coding of the location with the integer coding (under the assumption that a member, like a supplier, can choose one location at most), which helps to reduce the solution space of the problem. If there are n binary values (n locations to be chosen), the solution space of the location will reduce from 2^n to n. Moreover, a cooperative coevolutionary bare-bones PSO (CCBBPSO) with function independent decomposition (FID) is proposed to solve the LUSCND problem. The cooperative coevolutionary (CC) strategy is a promising solver for large-scale optimization problems [14]–[16]. It decomposes the problems into several parts and solves these parts independently via different populations, which can search solutions in different directions and help to increase the solution diversity. Although there have been many decomposition methods, determining how to decompose problems effectively is still a critical problem for the CC strategy. Differential grouping (DG) via interdependencies between variables (IaV) has a high grouping accuracy [15], [16]. However, the IaV is always difficult to be obtained because it needs complex calculations and consumes many FEs. On the contrary, the problem structure of LSCND is relatively specific because the variables have different functions. That is, the variables of LSCND belong to different kinds of SCN members which perform different tasks in the problem. Therefore, this article proposes an FID-based CC strategy to decompose the LSCND problem according to different functions of variables.

Second, to deal with the infeasible solutions of the LUSCND problem, two repair methods are designed. In the traditional works for COPs, the gradient-based repair method is effective to repair the infeasible solutions by the gradient information from the infeasible solutions to the feasible region [17], [18]. However, it is extremely difficult to obtain the gradient information of the LUSCND problem because there are a large number of variables. In this article, a step translation method (STM) is designed to map the out-of-bound values within the boundaries, and a labeled reposition operator with adaptive probabilities (LRO_{ap}) is designed to repair infeasible solutions that violate the constraints.

The main contributions of this article are shown as follows.

1) A new encoding scheme for the LUSCND model is proposed to reduce the solution space.
2) The CCBBPSO with FID is proposed to solve the LUSCND problem. During the evolution, the problem is decomposed into several subproblems by FID, and these subproblems are solved independently.
3) STM and LRO_{ap} are designed to repair infeasible solutions and improve the solving efficiency of the CCBBPSO-FID algorithm.

These innovative strategies enable CCBBPSO-FID to deal with larger data scales from thousands to hundreds of thousands dimensions. To evaluate the performance of CCBBPSO-FID, three scales of the instance sets are generated. The proposed algorithm is compared with several existing algorithms. First, as the LUSCND model is extended from the USCND model in [10], the global version PSO (GPSO) used in [10] is compared. Also, BBPSO [19] is compared because it is the basic algorithm of CCBBPSO-FID. Second, the competitive swarm optimizer (CSO) [20] is compared because it is an efficient algorithm for large-scale optimization. Moreover, two novel algorithms, including the social engineering optimizer (SEO) [21] and RDA [12], are compared because the SEO is a recent algorithm for complex problems and the RDA is a recent algorithm used for the LUSCND problems. Third, CCBBPSO-FID is compared with non-PSO (e.g., differential evolution (DE)-based [22]) algorithms and CCBBPSO with other decomposition schemes to verify the advantages by combining CCBBPSO with FID. For the fair comparison, these algorithms are all combined with the proposed repair methods, including STM and LRO_{ap}. Fourth, all algorithms without the repair methods are compared to prove the effectiveness of STM and LRO_{ap}. The experimental results show that CCBBPSO-FID obtains a better performance on the LUSCND problem.

The remainder of this article is organized as follows. Section II introduces the definitions of LUSCND and the original BBPSO algorithm. Section III describes all components of CCBBPSO-FID in detail. Section IV presents the exhaustive experiments and analyzes the results in different aspects. Finally, Section V gives a conclusion of this article.

II. Backgrounds

A. LUSCND Problem Definition

SCND is to design a network which is composed of a logistic system, a supply system, and a distribution system [23].

The logistic system records the order information of customers and warehouses; the supply system reflects the production patterns of suppliers; and the distribution system describes how products are distributed among different members. This article focuses on the LUSCND with a large-scale logistic system under supply and demand uncertainties [10]. There have been some relevant studies on the LUSCND problem, and the main similarities and differences between them and this article are shown in Table S.I in the supplementary material, from both the model formulation aspect and the algorithm design aspect. The main similarity is that the algorithms used in these studies are all EAs, and the main differences include different problem models and algorithm design details. As shown in Table S.I in the supplementary material, this article mainly focuses on and contributes to the algorithm design like the large-scale tackling and the constraint handling to propose an efficient algorithm but not the new problem model. In this sense, we adopt the USCND model in [10] because it considers multiple periods. Moreover, this model considers only the economic dimension so that it fits our objective to minimize the total cost of the entire system. Nevertheless, we extend the USCND to the large-scale model to make it more practical. To make the LUSCND model more reasonable, the following suppositions and constraints are considered in this article.

1) Different from most of the existing works that consider only one period, our model considers multiple periods and three echelons, including suppliers, warehouses, and customers.
2) This model has the capacity limitation. For example, the order quantity of a supplier cannot exceed its capacity; the inventory and the order quantity of a warehouse cannot exceed its capacity; and the supply of a customer cannot exceed its demand.
3) A supplier or warehouse can choose one location at most.
4) Each supplier can supply for several warehouses, and each warehouse can transport to several customers, and vice-versa.

This model is applicable in the three-echelon LUSCND that includes suppliers, warehouses, and customers. Fig. 1 illustrates the application scenario of this LUSCND with S suppliers, W warehouses, and C customers. In this model, the workflow mainly includes four stages.

1) Customers send the uncertain demands to warehouses (***C_demand***) in a continuous period of time.
2) Some warehouses with the optimal locations (***W_open***) are chosen to serve the customers and will send order to the suppliers (***WS_order***).
3) Some suppliers with the optimal locations (***S_open***) are chosen to serve warehouses and provide products to warehouses (***WS_order*** $\times$ ***S_yield***, where ***S_yield*** represents the uncertain output rates of suppliers).
4) Warehouses chosen in the second stage will transport the products to the customers (***WC_ship***).

It should be noted that as shown in Fig. 1, the uncertainties include the demand of customers and the supply of suppliers to make the model closer to the practical applications. To satisfy the uncertain demand of customers, it is better to supply materials as much as possible, but too much supply will cost more, including the fixed cost, the inventory cost, and the shipping cost. Therefore, to satisfy the demand and cut the cost simultaneously, the competent locations of suppliers and warehouses (***S_open*** and ***W_open***) should be optimally chosen. Moreover, the order quantity of warehouses to suppliers (***WS_order***) and the transport quantity of warehouses to customers (***WC_ship***) should be reasonably optimized to satisfy the demand of customers with the minimum cost.

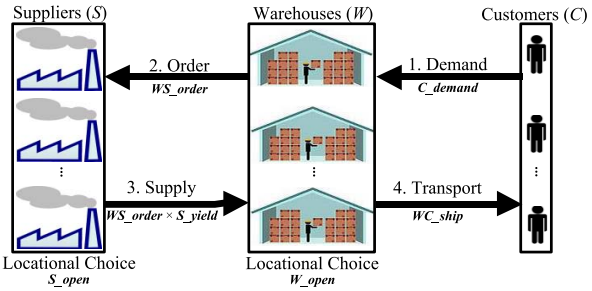

Fig. 1. Illustration of LUSCND.

In order to clearly describe the relationship between the decision variables and the objective function of the model, the details of the model formulation are described as follows. For customer $k(k = 1, 2, \ldots, C)$, in period $t(t = 1, 2, \ldots, T)$, it has a random demand quantity $C\_demand_{k,t}$ and the penalty $C\_penalty_k$ for a unit of the demand unmet. For warehouse $j(j = 1, 2, \ldots, W)$, it has W_n locations that can be chosen, and the lth $(l = 1, 2, \ldots, W\_n)$ location is with the capacity of $W\_capacity_{j,l}$ and the fixed cost $W\_fixedCost_{j,l}$ (e.g., the rent cost, the maintenance cost, etc.). The inventory cost is $W\_inventoryCost_j$ for storing a unit of goods in a period, and the initial inventory is $W\_inventory_{j,0}$. For supplier i $(i = 1, 2, \ldots, S)$, it has S_n locations to be chosen. The capacity of the rth $(r = 1, 2, \ldots, S\_n)$ location is $S\_capacity_{i,r}$ and the fixed cost is $S\_fixedCost_{i,r}$ (e.g., the production cost, the management cost, etc.). Besides, if a unit of capacity of supplier i is unused in a period, the penalty is $S\_penalty_i$. The random output rate of supplier i at period t is denoted by $S\_yield_{i,t}$, and the real supply quantity is the total order quantity of warehouses multiplied by $S\_yield_{i,t}$. The transportation costs from supplier i to warehouse j and from warehouse j to customer k with a unit of goods are denoted by $T\_sup2war_{i,j}$ and $T\_war2cus_{j,k}$, respectively.

The decision variables include ***S_open*** (the chosen locations of suppliers), ***W_open*** (the chosen locations of warehouses), ***WS_order*** (the order quantity of warehouses to suppliers), and ***WC_ship*** (the shipping quantity of warehouses to customers). $S\_open_{i,r}$ represents whether the rth location of supplier i is selected $(S\_open_{i,r} = 1)$ or not $(S\_open_{i,r} = 0)$, and $W\_open_{j,l}$ represents whether the lth location of warehouse j is selected $(W\_open_{j,l} = 1)$ or not $(W\_open_{j,l} = 0)$. $WS\_order_{i,j,t}$ represents the order quantity of supplier i from warehouse j at period t, and $WC\_ship_{j,k,t}$ represents the shipping quantity of warehouse j to customer k at period t. The objective is to obtain the optimal ***S_open***, ***W_open***, ***WS_order***, and ***WC_ship*** to minimize the total cost. The structure of the solution is shown

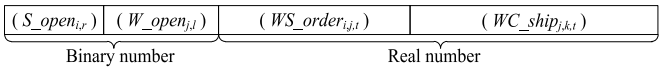

Fig. 2. Structure of the solution.

in Fig. 2, where ***S_open*** and ***W_open*** are binary numbers, and ***WS_order*** and ***WC_ship*** are real numbers.

The total cost *totalCost* of LUSCND can be divided into four parts, including the fixed cost, the inventory cost, the shipping cost, and the penalty. Specifically, the fixed cost *fixedCost* can be formulated as

$$\text{fixedCost} = \sum_{i=1}^{S}\sum_{r=1}^{S\_n} S\_fixedCost_{i,r} \times S\_open_{i,r} + \sum_{j=1}^{W}\sum_{l=1}^{W\_n} W\_fixedCost_{j,l} \times W\_open_{j,l}. \quad (1)$$

The inventory cost *inventoryCost* can be formulated as

$$\text{inventoryCost} = \sum_{j=1}^{W}\sum_{t=1}^{T} W\_inventoryCost_{j} \times W\_inventory_{j,t} \quad (2)$$

where $W\_inventory_{j,t}$ is the inventory quantity of warehouse j at period t, and it is calculated as follows to keep the production–inventory balance of warehouse j:

$$\begin{aligned} W\_inventory_{j,t} &= \left(\sum_{i=1}^{S} S\_output_{i,j,t} + W\_inventory_{j,t-1}\right) - \sum_{k=1}^{C} WC\_ship_{j,k,t} \\ &= \sum_{l=1}^{t}\left(\sum_{i=1}^{S} S\_output_{i,j,l} - \sum_{k=1}^{C} WC\_ship_{j,k,l}\right) + W\_inventory_{j,0} \end{aligned} \quad (3)$$

where $S\_output_{i,j,t}$ is the real output quantity of supplier i to warehouse j at period t and is calculated as follows:

$$S\_output_{i,j,t} = S\_yield_{i,t} \times WS\_order_{i,j,t}. \quad (4)$$

The shipping cost *shippingCost* can be formulated as

$$\text{shippingCost} = \sum_{i=1}^{S}\sum_{j=1}^{W}\sum_{t=1}^{T} T\_sup2war_{i,j} \times S\_output_{i,j,t} + \sum_{j=1}^{W}\sum_{k=1}^{C}\sum_{t=1}^{T} T\_war2cus_{j,k} \times WC\_ship_{j,k,t}. \quad (5)$$

At last, the penalty *Penalty* can be formulated as

$$\text{Penalty} = \sum_{i=1}^{S}\sum_{t=1}^{T} S\_penaly_{i} \times S\_rest_{i,t} + \sum_{k=1}^{C}\sum_{t=1}^{T} C\_penalty_{k} \times C\_unmet_{k,t} \quad (6)$$

where $S\_rest_{i,t}$ is the unused capacity of supplier i at period t, and $C\_unmet_{k,t}$ is the unmet quantity of customer k at period t, and they can be calculated as follows:

$$S\_rest_{i,t} = \sum_{r=1}^{S\_n} S\_capacity_{i,r} \times S\_open_{i,r} - S\_yield_{i,t} \times \sum_{j=1}^{W} WS\_order_{i,j,t} \quad (7)$$

$$C\_unmet_{k,t} = C\_demand_{k,t} - \sum_{j=1}^{W} WC\_ship_{j,k,t}. \quad (8)$$

Therefore, the final objective function of LUSCND can be formulated as

$$\text{Minimize} \quad \text{totalCost} = \text{fixedCost} + \text{inventoryCost} + \text{shippingCost} + \text{Penalty} \quad (9)$$

$$\text{subject to:} \quad \sum_{r=1}^{S\_n} S\_open_{i,r} \in \{0,1\},\ i = 1,2,\ldots,S \quad (10)$$

$$\sum_{l=1}^{W\_n} W\_open_{j,l} \in \{0,1\},\ j = 1,2,\ldots,W \quad (11)$$

$$WS\_order_{i,j,t} \geq 0,\ i = 1,2,\ldots,S,\ j = 1,2,\ldots,W,\ t = 1,2,\ldots,T \quad (12)$$

$$WC\_ship_{j,k,t} \geq 0,\ j = 1,2,\ldots,W,\ k = 1,2,\ldots,C,\ t = 1,2,\ldots,T \quad (13)$$

$$\sum_{j=1}^{W} WS\_order_{i,j,t} \leq \sum_{r=1}^{S\_n} S\_capacity_{i,r} \times S\_open_{i,r},\ i = 1,2,\ldots,S,\ t = 1,2,\ldots,T \quad (14)$$

$$W\_inventory_{j,t-1} + \sum_{i=1}^{S} WS\_order_{i,j,t} \leq \sum_{l=1}^{W\_n} W\_capacity_{j,l} \times W\_open_{j,l},\ j = 1,2,\ldots,W,\ t = 1,2,\ldots,T \quad (15)$$

$$W\_inventory_{j,t} \geq 0,\ j = 1,2,\ldots,W,\ t = 1,2,\ldots,T \quad (16)$$

$$C\_unmet_{k,t} \geq 0,\ k = 1,2,\ldots,C,\ t = 1,2,\ldots,T. \quad (17)$$

The model formulation is clarified according to the workflow of the LUSCND problem, as shown in Fig. 1. In the first stage, customers send demands to warehouses (***C_demand***) in a continuous period of time (T). In the second stage, some warehouses with the optimal locations are chosen to serve the customers (***W_open***) and will send order to suppliers (***WS_order***). In constraint (11), if $\sum_{l=1}^{W\_n} W\_open_{j,l} = 0$, warehouse j is not open; otherwise, the lth location with $W\_open_{j,l} = 1$ represents the location that warehouse j selects. The order quantity of warehouses to suppliers (***WS_order***) should not be smaller than 0, as shown in (12), and should be not larger than the capacity of suppliers (***S_capacity***), as shown in (14). In addition, the current inventory of warehouses [***W_inventory*** calculated by (3)] and the order quantity

TABLE I
NOTATIONS IN THE LUSCND PROBLEM

Type	Name	Meaning
Indices	i	index of suppliers
	j	index of warehouses
	k	index of customers
	t	index of periods
	r	index of the location of suppliers
	l	index of the location of warehouses
Parameters	T	number of periods
	S	number of suppliers
	W	number of warehouses
	C	number of customers
	S_n	location number of each supplier
	$S\_capacity_{i,r}$	capacity of i in r
	$S\_fixedCost_{i,r}$	fixed cost of i in r
	$S\_penalty_i$	penalty cost of i
	$S\_yield_{i,t}$	output rate of i in t
	W_n	location number of each warehouse
	$W\_capacity_{j,l}$	capacity of j in l
	$W\_fixedCost_{j,l}$	fixed cost of j in l
	$W\_inventoryCost_j$	inventory cost of j
	$W\_inventory_{j,0}$	initial inventory quantity of j
	$C\_demand_{k,t}$	demand quantity of k in t
	$C\_penalty_k$	penalty cost of k
	$T\_sup2war_{i,j}$	transportation cost of i to j
	$T\_war2cus_{j,k}$	transportation cost of j to k
	$totalCost$	total cost of the system
	$fixedCost$	fixed cost of the system
	$inventoryCost$	inventory cost of the system
	$W\_inventory_{j,t}$	inventory quantity of j at t
	$S\_output_{i,j,t}$	output quantity of i to j at t
	$shippingCost$	shipping cost of the system
	$Penalty$	penalty cost of the system
	$C\_unmet_{k,t}$	unmet quantity of k at t
	$S\_rest_{i,t}$	unused capacity of i at t
Decision Variables	$S\_open_{i,r}$	whether r of i is selected
	$W\_open_{j,l}$	whether l of j is selected
	$WS\_order_{i,j,t}$	order quantity from j to i at t
	$WC\_ship_{j,k,t}$	ship quantity from j to k at t

of warehouses to suppliers (***WS_order***) should not be larger than the capacity of warehouses (***W_capacity***), as shown in (15). The current inventory of warehouses (***W_inventory***) should not be smaller than 0, as shown in (16). In the third stage, some suppliers with the optimal locations are chosen to serve warehouses (***S_open***) and provide products to warehouses (***S_yield*** $\times$ ***WS_order***). In constraint (10), if $\sum_{r=1}^{S\_n} S\_open_{i,r} = 0$, supplier i is not open; otherwise, the rth location with $S\_open_{i,r} = 1$ represents the location that supplier i selects. In the fourth stage, the warehouses chosen in the second stage will transport the products to the customers (***WC_ship***). The transport quantity (***WC_ship***) should not be smaller than 0, as shown in (13). In addition, the unmet quantity of customers (***C_demand*** – ***WC_ship***) should not be smaller than 0, as shown in (17).

Table I summarizes the notations of all the indices, parameters, and decision variables used in the LUSCND. To make the real-world applications of our proposed approach clearer for the managers of SCND, Fig. 3 shows an example of how to use the solutions obtained by an optimization algorithm to the LUSCND problem at a period. In this solution, there are one open supplier $S_{1,3}$ (the third location is chosen for S_1), two open warehouses $W_{1,3}$ and $W_{2,2}$ (the third location is chosen for W_1, and the second location is chosen for W_2), and four customers (C_1, C_2, C_3, and C_4). The capacities of $S_{1,3}$, $W_{1,3}$, and $W_{2,2}$ are shown at the top left in the figure, and the initial inventory quantities of $W_{1,3}$ and $W_{2,2}$ are shown in parentheses. In this period, it is assumed that the yield rate of $S_{1,3}$ is 0.8, and ***WS_order*** and ***WC_ship*** (the decision variables) are shown as solid lines with a number on the left.

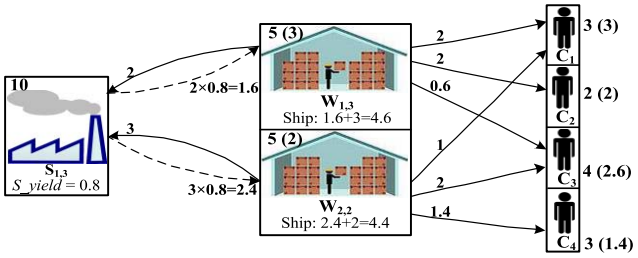

Fig. 3. Example of the LUSCND problem.

The example can be stated in four steps. First, customers send demands to warehouses. It is assumed that the total demand quantities of the four customers are 3, 2, 4, and 3, respectively. It is noted that there is no need to calculate the demand quantity of each customer to each warehouse in the FE, since the related cost (the penalty from customers) can be obtained by ***WC_ship*** in (6) and (8). Second, warehouses with the optimal locations are chosen to serve the customers (***W_open***, $W_{1,3}$ and $W_{2,2}$ in this example), and send orders to the supplier (***WS_order***). For example, $W_{1,3}$ sends 2 units of order to $S_{1,3}$. Third, the supplier with the optimal location is chosen (***S_open***, $S_{1,3}$ in the example) and transports goods to warehouses (***S_yield*** $\times$ ***WS_order***), shown as the dotted lines with a number on the right. The yield rate of $S_{1,3}$ is 0.8, and it can deliver $2 \times 0.8 = 1.6$ units of goods to $W_{1,3}$. Similarly, $S_{1,3}$ sends $3 \times 0.8 = 2.4$ units of goods to $W_{2,2}$. After getting the supply, the quantities of $W_{1,3}$ and $W_{2,2}$ including supplies and inventory are $1.6 + 3 = 4.6$ and $2.4 + 2 = 4.4$, respectively. Finally, the warehouses that have been chosen in the second step deliver goods to customers (***WC_ship***). As shown in the parentheses, C_1, C_2, C_3, and C_4 get 3, 2, 2.6, 1.4 units of goods, respectively. The demands of C_3 and C_4 are not satisfied.

B. BBPSO

BBPSO, developed by Kennedy [19], is a simplified version of PSO. Instead of using the velocity vector to update the position vector, BBPSO applies the Gaussian distribution to set the position information directly, as shown in (18)

$$x_{i,d}(g+1) = N\left(\mu_{i,d}(g), \sigma_{i,d}^2(g)\right) \tag{18}$$

$$\mu_{i,d}(g) = 0.5 \times \left(p\text{best}_{i,d}(g) + g\text{best}_d(g)\right) \tag{19}$$

$$\sigma_{i,d}(g) = \left|p\text{best}_{i,d}(g) - g\text{best}_d(g)\right| \tag{20}$$

where $x_{i,d}(g+1)$ is the dth dimension of the ith individual at iteration $g+1$; $N(\mu_{i,d}(g), \sigma_{i,d}(g))$ is a random number obeying

$$\begin{aligned}\text{Minimize}\quad & \text{totalCost} = \text{fixedCost} + \text{inventoryCost} + \text{shippingCost} + \text{Penalty}\\ &= \sum_{i=1}^{S} S\_fixedCost_{i,S\_open_i} + \sum_{j=1}^{W} W\_fixedCost_{j,W\_open_j}\\ &\quad + \sum_{j=1}^{W}\sum_{t=1}^{T} W\_inventoryCost_j \times \left(\sum_{l=1}^{t}\left(\sum_{i=1}^{S} S\_yield_{i,l} \times WS\_order_{i,j,l} - \sum_{k=1}^{C} WC\_ship_{j,k,l}\right) + W\_inventory_{j,0}\right)\\ &\quad + \sum_{i=1}^{S}\sum_{j=1}^{W}\sum_{t=1}^{T} T\_sup2war_{i,j} \times S\_yield_{i,t} \times WS\_order_{i,j,t} + \sum_{j=1}^{W}\sum_{k=1}^{C}\sum_{t=1}^{T} T\_war2cus_{j,k} \times WC\_ship_{j,k,t}\\ &\quad + \sum_{i=1}^{S}\sum_{t=1}^{T} S\_penaly_i \times \left(S\_capacity_{i,S\_open_i} - S\_yield_{i,t} \times \sum_{j=1}^{W} WS\_order_{i,j,t}\right) + \sum_{k=1}^{C}\sum_{t=1}^{T} C\_penalty_k\\ &\quad \times \left(C\_demand_{k,t} - \sum_{j=1}^{W} WC\_ship_{j,k,t}\right) \end{aligned} \tag{21}$$

$$S\_open_i \in \{0, 1, 2, \ldots, S\_n\},\ \ i = 1, 2, \ldots, S \tag{22}$$

$$W\_open_j \in \{0, 1, 2, \ldots, W\_n\},\ \ j = 1, 2, \ldots, W \tag{23}$$

$$\sum_{j=1}^{W} WS\_order_{i,j,t} \le S\_capacity_{i,S\_open_i},\ \ i = 1, 2, \ldots, S, t = 1, 2, \ldots, T \tag{24}$$

$$\sum_{l=1}^{t-1}\left(\sum_{i=1}^{S} S\_yield_{i,l} \times WS\_order_{i,j,l} - \sum_{k=1}^{C} WC\_ship_{j,k,l}\right) + W\_inventory_{j,0} + \sum_{i=1}^{S} WS\_order_{i,j,t} \le W\_capacity_{j,W\_open_j}$$
$$j = 1, 2, \ldots, W, t = 1, 2, \ldots, T \tag{25}$$

$$\sum_{l=1}^{t}\left(\sum_{i=1}^{S} S\_yield_{i,l} \times WS\_order_{i,j,l} - \sum_{k=1}^{C} WC\_ship_{j,k,l}\right) + W\_inventory_{j,0} \ge 0,\ \ j = 1, 2, \ldots, W, t = 1, 2, \ldots, T \tag{26}$$

$$C\_demand_{k,t} - \sum_{j=1}^{W} WC\_ship_{j,k,t} \ge 0,\ \ k = 1, 2, \ldots, C, t = 1, 2, \ldots, T \tag{27}$$

the Gaussian distribution with the mean value $\mu_{i,d}(g)$ and the deviation $\sigma_{i,d}(g)$; $p\text{best}_{i,d}(g)$ is the personal best of the ith individual; and $g\text{best}_d(g)$ is the dth dimension of the global best individual. Another advantage of BBPSO is that there are no parameters needed to be tuned. Thus, BBPSO has attracted great attention in diverse applications [24]–[27].

III. CCBBPSO-FID Approach

A. Solution Encoding for Dimensionality Reduction

According to the LUSCND problem definition, the dimension of a solution (including ***S_open***, ***W_open***, ***WS_order***, and ***WC_ship***) is equal to $S \times S\_n + W \times W\_n + S \times W \times T + W \times C \times T$. When the LUSCND problem consists of tens of suppliers, warehouses, and periods and one hundred customers, the dimension increases sharply to 10 000. It needs a large storage for a solution, which increases the problem-solving difficulty. An effective approach is to reduce the dimensionality of the model. From the view of data types of decision variables, ***WS_order*** and ***WC_ship*** with real numbers cannot be compressed. For ***S_open*** ($\sum_{r=1}^{S\_n} S\_open_{i,r} \in \{0, 1\}$) with binary numbers, the array $[S\_open_{i,1}, S\_open_{i,2}, \ldots, S\_open_{i,S\_n}] \in \{0, 1\}^{S\_n}$ can be replaced by an integer $S\_open_i \in \{0, 1, 2, \ldots, S\_n\}$. If S_open_i is 0, supplier i is not open; otherwise, S_open_i represents the location that supplier i selects. Similar to ***S_open***, $[W\_open_{j,1}, W\_open_{j,2}, \ldots, W\_open_{j,W\_n}] \in \{0, 1\}^{W\_n}$ can be replaced by $W\_open_j \in \{0, 1, 2, \ldots, W\_n\}$. Then, the dimension will be reduced from $S \times S\_n + W \times W\_n$ to $S + W$. Meanwhile, the runtime will be reduced during the entire evolutionary process. Fig. 4 illustrates an example of the encoding scheme of ***S_open*** and ***WS_order*** in this article. Because the real number coding is used in the programming, values in ***S_open*** are needed to be rounded down to be feasible. For example, in Fig. 4, after being rounded down, $S\_open_1 = 3$ represents that the first supplier (S_1) is open in the third location, and $S\_open_2 = 0$ represents that the second supplier (S_2) is not open. Values in ***WS_order*** represent the order quantity of warehouses to suppliers. For example, $WS\_order_{1,2,0} = 3.6$ (in the first row and the second column) represents that the first warehouse (W_1) sends an order of 3.6 units to the second supplier (S_2) at the period $t = 0$. The encoding schemes of ***W_open*** and ***WC_ship*** are similar to these of ***S_open*** and ***WS_order***, respectively, and they are not shown in the figure.

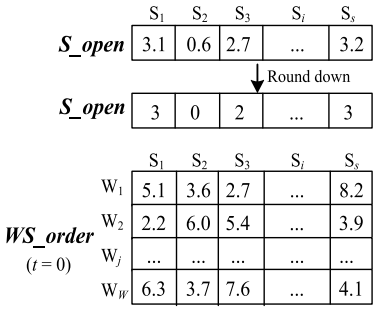

Fig. 4. Illustration of the encoding schemes of ***S_open*** and ***WS_order***.

After these modifications and removing the intermediate variables, the final model formulation of the objective function is more clearly shown in (21), as shown at the top of the previous page, whose relationship with the decision variables, including ***S_open***, ***W_open***, ***WS_order***, and ***WC_ship***, is also clearer. Moreover, the objective function is subject to (12), (13), and (22)–(27), as shown at the top of the previous page. Constraints (22)–(27) are equivalent to (10), (11), and (14)–(17), respectively. In order to minimize the *totalCost* of (21), the locations of suppliers (***S_open***) have to be optimally chosen so that they not only satisfy (22) and (24), but also have low *S_fixedCost* and small *S_capacity*. Also, the locations of warehouses (***W_open***) have to be optimally chosen so that they satisfy (23) and (25) and have low *W_fixedCost*. Moreover, the ***WS_order*** and ***WC_ship*** have to be optimized to be subject to (12), (13), and (24)–(27).

B. Fitness Evaluation With Uncertainties

Apart from the large-scale challenge in the LUSCND, how to deal with the uncertainties is an another difficulty. The FEs of solutions in the uncertain problems and the dynamic optimization problems (DOPs) are both not fixed during the entire evolution of algorithms. It is noted that the fitness function of DOPs in each certain environment is fixed although it changes in different environments [28]–[30]. However, in the LUSCND problem, the fitness function is always uncertain, with the uncertain demand of customers (***C_demand***) and the uncertain supply of suppliers (***S_yield***). Therefore, the strategies for DOPs are not suitable for the LUSCND problem. In this article, the MCM is used to evaluate the LUSCND problem by multiple samplings of the uncertain factors.

In the literature, the MCM, also called statistic testing method or random sampling, is used to solve mathematical and physical problems with uncertainties [31]–[33]. It simulates uncertain factors through random sampling and obtains approximate fitness values. Due to its simplicity, this method is widely applied in the USCND problems [10], [34]. Therefore, the MCM is also used to deal with the supply and demand uncertainties in this article. For each solution, ten simulations with different ***S_yield*** and ***C_demand*** are randomly sampled. The fitness value of each simulation is calculated as (21) by using certain values of uncertain factors. Then, the mean result of the ten simulations is regarded as the final fitness value.

It is worth mentioning that *fixedCost* and the shipping cost of warehouses to customers (*shippingCost*2) are not related to uncertain factors, so there is no need to recalculate them in the simulation. Other costs, including *inventoryCost*, the shipping cost of suppliers to warehouses (*shippingCost*1), and *Penalty*, are calculated ten times with ten different samples of ***S_yield*** and ***C_demand***. The sum of *fixedCost* and *shippingCost*2, and the mean result of ten simulations is recorded as the final fitness value. During the calculation, the feasibility of the solution is judged, and an error code (*ecode*) is recorded which will be used in the repair operator LRO_{ap} in Section III-F. If *ecode* is equal to 0, it means that the individual is feasible, and the calculation continues until the MCM is executed ten times; otherwise, it means that the individual is infeasible, and this calculation will end in advance. The fitness value of infeasible solutions is set to 10 000 000. Fig. 5 shows two cases in the calculation, including a feasible solution and an infeasible solution. It is noted that Fig. 5(b) illustrates an infeasible solution case, where the number 7 is only an example and it can also be any other values smaller than 10.

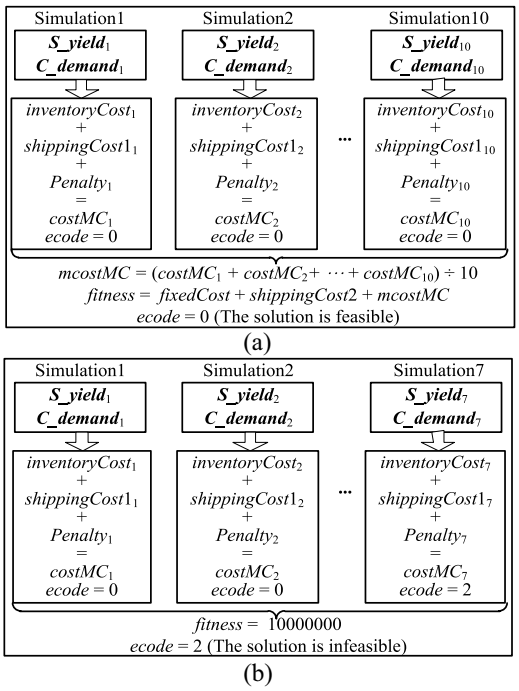

Fig. 5. Calculation of the fitness values. (a) Calculation of a feasible solution. (b) Calculation of an infeasible solution.

C. Initialization

Random initialization is used to increase the population diversity and distribute individuals among the solution space widely. For ***S_open*** and ***W_open***, they are generated randomly within $\{0, 1, 2, \ldots, S\_n\}$ and $\{0, 1, 2, \ldots, W\_n\}$, respectively. For ***WS_order*** and ***WC_ship***, two generation methods are designed, and the common point of them is to ensure that ***WS_order*** and ***WC_ship*** are smaller than ***S_capacity*** and ***W_capacity***, respectively. The first method is that they are set randomly within [0, *capacity*], where *capacity* is a known parameter to set ***S_capacity*** and ***W_capacity***. The second method is to deduce their values from formulas, being more complicated. From (24), it is deduced

that ***WS_order*** can be set randomly within [0, *capacity/W*]. In (3) and (4), if $S\_yield_{i,t}$ is 1 and the inventory of warehouse j (including $W\_inventory_{j,t}$ and $W\_inventory_{j,t-1}$) is 0, $\sum_{i=1}^{S} WS\_order_{i,j,t} \approx \sum_{k=1}^{C} WC\_ship_{j,k,t}$ and ***WC_ship*** can be set randomly within [0, (capacity$/W \times S)/C$].

The initialization includes two steps. First, the global optimal solution ***xbest*** is generated randomly by the first method. Second, four subpopulations (swarm_1, swarm_2, swarm_3, and swarm_4) are set separately via the second method according to the construction method introduced in Section III-D. For example, a solution $\boldsymbol{x}$ in swarm_1 consists of ***x.S_open*** (initialized in swarm_1) and ***xbest.W_open***, ***xbest.WS_order***, ***xbest.WC_ship***.

D. Cooperative Coevolution

Inspired by the idea of "divide-and-conquer," Potter and De Jong [35] designed the CC strategy. It is an effective method for large-scale optimization [14]–[16], [36]–[38]. The first step of this strategy is to divide the variables into several parts with a specific decomposition strategy, which can be regarded as the dimensionality reduction. Then, these parts are solved with different subpopulations independently, and the local best solutions of subpopulations are obtained to update ***xbest***.

1) FID-Based Decomposition: Since the CC strategy was proposed, there have been several decomposition methods [14]–[16]. The aim is to assign interacting variables to one group as more as possible. If interdependent variables are placed in different groups, they will be blind to the synchronized information of other interdependent variables, since different groups are evolved separately, and it will also be difficult to find the best values of these interdependent variables. The popular approach is to group variables via IaV [14]–[16], but it has high computational complexity, especially for super-large-scale problems. However, with the clear solution structure of LUSCND, there is no need to calculate IaV.

In this article, FID is devised to divide the problem according to different functions of different parts of the solution structure. The solution structure consists of four parts (***S_open***, ***W_open***, ***WS_order***, and ***WC_ship***). Different parts have different functions, since they belong to different kinds of SCN members which perform different tasks. Specifically, the decision variables of ***S_open*** belong to suppliers which decide the locations of suppliers; the decision variables of ***W_open*** belong to warehouses which decide the locations of warehouses; the decision variables of ***WS_order*** belong to warehouses which decide the order quantities of warehouses to suppliers; and the decision variables of ***WC_ship*** belong to warehouses which decide the transportation quantities of warehouses to customers. Therefore, this article decomposes the LUSCND problem into four parts to deal with different subproblems, including locations of suppliers, locations of warehouses, order quantities of warehouses to suppliers, and transportation quantities of warehouses to customers. The four decomposed parts include ***S_open***, ***W_open***, ***WS_order***, and ***WC_ship***. In the proposed CCBBPSO algorithm, four different subpopulations, called swarm_1, swarm_2, swarm_3, and swarm_4, are used to evolve the four different parts, respectively.

Different from other decomposition methods that have to learn the IaV or to regroup variables during the evolution, FID based on different functions divides the problem naturally. It is executed only at the beginning of the evolution and does not consume FEs. In addition, FID has more general applicability in large-scale realistic problems, since it decomposes the solution by functions of different roles in problems.

2) Construction of Solutions in Subpopulations: Although decomposed parts of the solution space are evolved separately, it is necessary to constitute the entire solution to obtain the fitness value. In the proposed algorithm, a partial solution of an individual is integrated with other parts of ***xbest*** to obtain a complete solution, same as [14]. For example, a solution $\boldsymbol{x}$ in swarm_1 is composed of ***x.S_open*** (variables solved in swarm_1) and ***xbest.W_open***, ***xbest.WS_order***, ***xbest.WC_ship***. That is, those variables not solved in this swarm are borrowed from the global best solution ***xbest***.

E. STM-Based Boundary Processing

After updating the individual $\boldsymbol{x}$ by (18), if some variables exceed the search ranges, they will be restricted within reasonable ranges to satisfy the constraints. For variables belonging to ***S_open*** and ***W_open***, they are limited within $\{0, 1, \ldots, S\_n\}$ and $\{0, 1, \ldots, W\_n\}$, respectively.

For variables belonging to ***WS_order***, such as $WS\_order_{i,j,t}$, it is limited within $[0, \min(S\_capacity_{i,r}, W\_capacity_{j,l})]$, where i and j are the corresponding supplier and warehouse, and r and l are the corresponding locations. It indicates that the order quantity of supplier i from warehouse j at period t cannot exceed the capacity of warehouse j and supplier i.

For variables belonging to ***WC_ship***, such as $WC\_ship_{j,k,t}$, it is limited within $[0, \min(W\_capacity_{j,l}, C\_demand_{k,t})]$, where k and t are the corresponding customer and period. It shows that the shipping quantity of warehouse j to customer k at period t cannot exceed the capacity of warehouse j and the demand of customer k.

For variables out of bounds, if they are simply set to the bound values, many infeasible variables will have the same value (i.e., the bound value). This method also decreases the diversity of solutions. Therefore, STM is proposed in this article to map illegal values to legal values. The illustration of STM is shown in Fig. 6, where X is the illegal value and X' is the corresponding legal value after the STM operation. Herein, L and U are the lower bound and the upper bound, respectively, and *len* is the length between L and U. Then, X is translated to X' by the step of *len*, where $\overrightarrow{XX'}$ is the motion vector. That is, if $X > U$, X will be moved to the left until it becomes smaller than $U(X = X - \text{ceil}((X - U)/(U - L)) \times (U - L))$; otherwise if $X < L$, X will be moved to the right until it becomes larger than $L(X = X + \text{ceil}((L - X)/(U - L)) \times (U - L))$.

F. LRO-Based Infeasible Solution Repairing

The penalty function is widely used in the complicated problems to calculate the fitness values of infeasible solutions [10], [39]. However, with the scale growing, the

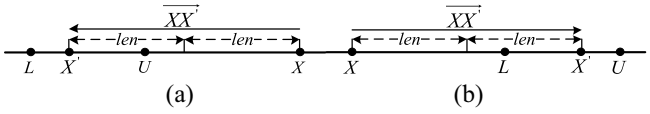

Fig. 6. Illustration of STM. (a) $X > U$. (b) $X < L$.

number of constraints will rise sharply, making it more difficult to obtain a feasible solution. Therefore, the penalization is not an effective method, and the efficient way is to repair infeasible solutions. In this article, LRO is proposed to repair illegal values repeatedly by resetting them to reasonable values. LRO includes four repair operators Repair_1, Repair_2, Repair_3, and Repair_4, which are designed for constraints (24)–(27), respectively. Only one kind of the operators is executed in each repair, and the label is set to record which repair operator will be executed. Recording the label helps to save running time, since the algorithm does not need to rejudge which repair operator to be used every time. For a complex problem with multiple constraints, LRO is an effective repair method to deal with these constraints separately, and the label helps to distinguish different repair operators for different constraints in the multiple repair process.

For simplicity, each repair operator only changes a specified part of solutions. Specifically, Repair_1 only repairs ***S_open***; Repair_2 only repairs ***W_open***; and Repair_3 and Repair_4 only repair ***WC_ship***. Since ***WS_order*** is involved in (24)–(26) and hard to update, it will not be changed in LRO.

In Repair_1 for (24), if $\sum_{j=1}^{W} WS\_order_{i,j,t} > \max_{r\in\{1,\ldots,S\_n\}}(S\_capacity_{i,r})$, it represents the total order quantity of supplier i at period t exceeds the maximum capacity of supplier i, and this repair operator will be useless after changing ***S_open***; otherwise, supplier i will choose one location where it can hold the capacity of $\sum_{j=1}^{W} WS\_order_{i,j,t}$. Considering the corresponding costs, the legal location with the smallest capacity and fixed cost is chosen.

In Repair_2 for (25), operations are similar to those in Repair_1. If $W\_inventory_{j,t-1} + \sum_{i=1}^{S} WS\_order_{i,j,t} > \max_{l\in\{1,\ldots,W\_n\}}(W\_capacity_{j,l})$, where $W\_inventory_{j,t-1}$ is calculated by (3), it represents the sum of the previous inventory and new supplies of warehouse j at period t exceed the warehouse's maximum capacity, and this operator will not work after changing ***W_open***; otherwise, warehouse j will choose the legal location with the smallest capacity and fixed cost.

If $W\_inventory_{j,t}$ is smaller than 0, it represents constraint (16) [equivalent to (26)] is not satisfied, and Repair_3 will be used. Let *difference* denote $-W\_inventory_{j,t}$ of the infeasible solution; then, $W\_inventory_{j,t}$ should increase to 0 at least to make the solution feasible. Let *WC_sum* denote $\sum_{k=1}^{C} WC\_ship_{j,k,t}$. Since only ***WC_ship*** is changed in this operator, it can be deduced from (3) that *WC_sum* should decrease by *difference*. If *WC_sum* $<$ *difference*, the operator is useless after updating ***WC_ship***; otherwise, $WC\_ship_{j,k,t}$ can decrease proportionally by *difference* $\times$ $(WC\_ship_{j,k,t}/WC\_sum)$ to satisfy the condition. It is remarkable that the precision problem also makes the theoretical feasible solutions infeasible. To solve the problem, $WC\_ship_{j,k,t}$ should be subtracted a small real number *precision* (set as 0.02 in the algorithm), if $WC\_ship_{j,k,t}$ is bigger than *precision*.

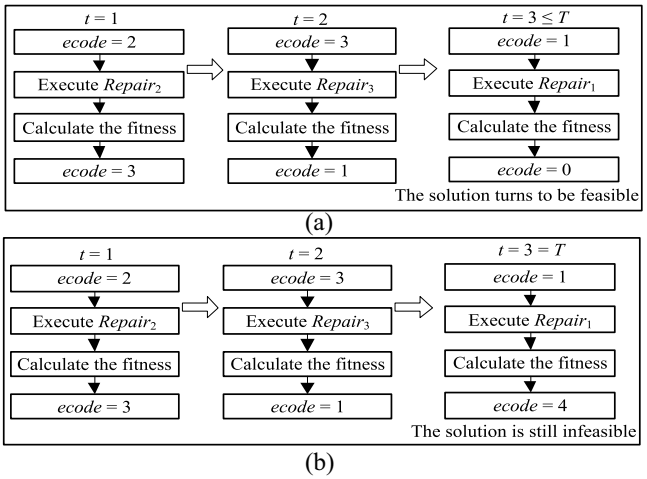

Fig. 7. Two examples of LRO. (a) Solution is feasible after repair. (b) Solution is still infeasible after repair.

Operations in Repair_4 for (17) [equivalent to (27)] are similar to those in Repair_3. If $C\_unmet_{k,t}$ is smaller than 0, it represents constraint (17) is not satisfied. Let *difference* denote $-C\_unmet_{k,t}$ and *WC_sum* denote $\sum_{j=1}^{W} WC\_ship_{j,k,t}$ in (8). Then, if *WC_sum* = *difference*, $WC\_ship_{j,k,t}$ can decrease proportionally by *difference* $\times$ $(WC\_ship_{j,k,t}/WC\_sum)$.

After the FE, if *ecode* (the label) is $e(e = 1, 2, 3, 4)$, Repair_e will be executed. At the end of a repair operator, the fitness value is calculated again, and *ecode* is also returned. If *ecode* is 0, it represents that the solution turns to be feasible, and this repair operator ends with success; otherwise, if the solution is still infeasible after being repaired T (the number of periods in the LUSCND problem) times, LRO ends with failure. Fig. 7 shows two examples of LRO, including successful and unsuccessful repair.

LRO is useful but time consuming. In the proposed algorithm, P_repair_i is set adaptively to control the repair probability of individual i, and the initial value is set to $0.01 \times S$ (the number of suppliers in the LUSCND problem). If individual i has been repaired in the last generation, P_repair_i will decrease by $0.001 \times S$; otherwise, this value will increase by $0.001 \times S$. LRO with the adaptive probability is called as LRO_{ap}, and it helps to find feasible solutions of the large-scale problem within the acceptable time, being proved in the experiments.

G. Complete CCBBPSO-FID Algorithm

The flowchart of CCBBPSO-FID is shown in Fig. 8. As mentioned above, the important parts of CCBBPSO-FID include the CC strategy based on FID, the boundary processing method STM, and the repair operator LRO_{ap}.

IV. Experimental Results and Analysis

A. Experimental Setting

It is difficult to obtain the instance set of the LUSCND problem in real life, especially for large-scale data, so the instance set is generated randomly. First, all capacities (***S_capacity*** and ***W_capacity***) are generated randomly

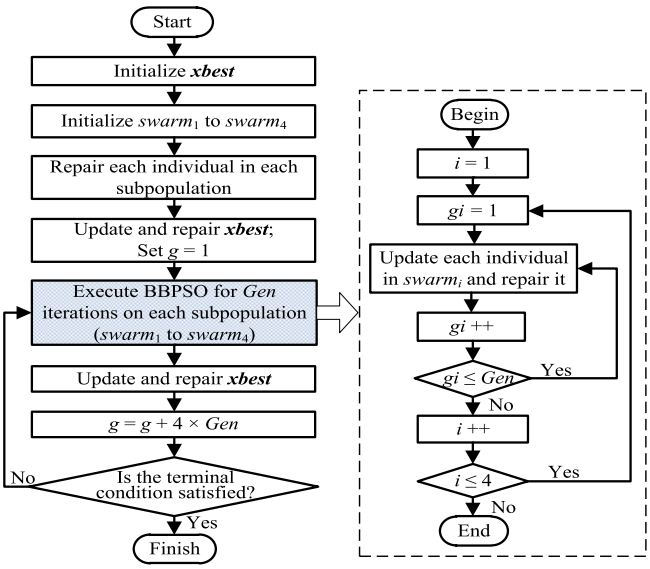

Fig. 8. Flowchart of the CCBBPSO-FID algorithm.

Step 1) Calculate the mean capacity $Mcap\_w_j$ of warehouse j:

$$Mcap\_w_j = (\sum_{l=1}^{W\_n} W\_capacity_{j,l}) / W\_n$$

Step 2) Calculate the mean capacity $Mcap\_s_i$ of supplier i:

$$Mcap\_s_i = (\sum_{r=1}^{S\_n} S\_capacity_{i,r}) / S\_n$$

Step 3) Calculate $\sum_{i=1}^{S}\sum_{j=1}^{W} WS\_order_{i,j,t}$ from (24), and assume $\sum_{j=1}^{W} WS\_order_{i,j,t} \approx S\_capacity_{i,S\_open_i} \approx Mcap\_s_i$:

$$\sum_{i=1}^{S}\sum_{j=1}^{W} WS\_order_{i,j,t} \approx \sum_{i=1}^{S} Mcap\_s_i$$

Step 4) Calculate $\sum_{i=1}^{S} WS\_order_{i,j,t} \approx \sum_{i=1}^{S}\sum_{j=1}^{W} WS\_order_{i,j,t} / W \approx \sum_{i=1}^{S} Mcap\_s_i / W$.

Step 5) Assume $\sum_{l=1}^{t-1}(\sum_{i=1}^{S} S\_yield_{i,l} \times WS\_order_{i,j,l} - \sum_{k=1}^{C} WC\_ship_{j,k,l}) = 0$ in (25), and calculate $W\_inventory_{j,0}$:

$$W\_inventory_{j,0} \approx \left| W\_capacity_{j,W\_open_j} - \sum_{i=1}^{S} WS\_order_{i,j,t} \right| \approx \left| Mcap\_w_j - \sum_{i=1}^{S} Mcap\_s_i / W \right|$$

Fig. 9. Generation of $W\_inventory_{j,0}$.

within [capacity, 2 × capacity], where *capacity* is set to 5.0; all costs (***S_penalty***, ***S_fixedCost***, ***W_inventoryCost***, ***W_fixedCost***, ***C_penalty***, ***T_sup2war***, and ***T_ware2cus***) are set randomly within [0, cost], where *cost* is set to 10.0. Second, $W\_inventory_{j,0}$ ($j = 1, 2, \ldots, W$), ***S_yield***, and ***C_demand*** are set according to the following methods.

1) Generation of $W\_inventory_{j,0}$: By analyzing (25) and (26), it can be observed that the feasibility of solutions is closely related to $W\_inventory_{j,0}$. Equations (24) and (25) are considered to obtain reasonable values of $W\_inventory_{j,0}$, as shown in Fig. 9.

2) Generation of Uncertain Factors: To verify the performance of algorithms, different levels of uncertain factors are generated, similar to [10]. ***S_yield*** is generated randomly within three domains (YL: [0.8, 0.9], YM: [0.7, 0.9], and YH: [0.6, 0.9]), where YL, YM, and YH mean low, medium, and high level of uncertainties for the yield rate of suppliers, respectively. Moreover, these values obey the normal distribution. ***C_demand*** is generated in three different probability distribution function (PDF) like normal (Norm), log-normal (Log), and triangular (Tri), and each kind of distribution includes three levels according to the coefficient of variances (CV) (DL: CV = 0.05; DM: CV = 0.1; and DH: CV = 0.2), where DL, DM, and DH mean low, medium, and high level of uncertainties for the demand of customers, respectively. For the Norm and Log distributions, their mean values are set randomly within [capacity, 2 × capacity]. For the Tri distribution, its lower limit a, upper limit b, and mode c are set to 0, 1/capacity, and 2/(3 × capacity), respectively, to ensure the height of the triangular $2/(b-a)$ is 2 × capacity. Totally, there are 3 × 3 × 3 = 27 uncertain factors for each instance.

TABLE II
SCALE CONFIGURATIONS AND EXECUTION TIME OF THE INSTANCE SETS

No.	S	W	C	T	Dimension (D)	Execution time (Seconds)
I	5	10	15	10	2015	1
II	10	20	50	20	24030	10
III	30	50	100	30	195080	100

Data at three scales are generated with the same values of S_n and W_n ($S\_n = W\_n = 5$), and other parameters are shown in Table II. For the fair comparison, the execution time of each algorithm is equal for the same test set. Each scale includes five instances, such as I_0, I_1, I_2, I_3, and I_4 for Test I. There are 3 (scales) × 5 (instances) × 27 (uncertainty factors) = 405 uncertain instances in total.

B. Experimental Setup

All algorithms were implemented in C++, and experiments were run on the PC with Intel Core i7-7700 and 8.0-GB memory. Compared algorithms are classified into three groups: the first one includes PSOs, such as GPSO [10] and BBPSO [19]; the second one includes a large-scale optimization algorithm—CSO [20] and two recent algorithms—SEO [21] and RDA [12]. Furthermore, to verify the effectiveness of combining CCBBPSO with FID, the third group includes CCDE-FID (using DE/rand/1 in [22] as the optimizer) and two other CCBBPSO variants called average decomposition of CCBBPSO (CCBBPSO-AD) and random decomposition of CCBBPSO (CCBBPSO-RD). It is noted that four versions of DE in [22] are tested, including DE/rand/1 with the crossover rate CR = 0.1 and 0.9, and DE/best/1 with CR = 0.1 and 0.9 (the amplification factor $F = 0.5 + 0.5 \times \text{rand}(0, 1)$ for all versions). Herein, only the results of the first version (DE/rand/1 with CR = 0.1) are present because it has generally the best performance among the four DE variants. Besides, DG is not compared with FID since it consumes a large number of FEs, as shown in Table S.II in the supplementary material. STM and LRO_{ap} are used in all algorithms for the fair comparison.

The control variable method [40] is used to investigate the parameters of all the algorithms to find out the optimal values,

TABLE III
OPTIMAL PARAMETER SETTINGS OF ALGORITHMS

Algorithm	Tuned parameters
CCBBPSO-FID	$N\_pop = 20$, $Gen = 10$
GPSO	$N\_pop = 100$, $c1 = c2 = 2.0$
BBPSO	$N\_pop = 100$
CSO	$N\_pop = 500$
SEO	$nAttacker = 30$, $\alpha = 0.1$, $\beta = 0.05$
RDA	$N\_pop = 200$, $Nmale = 30$, $\alpha = 0.9$, $\beta = 0.5$, $\gamma = 0.8$
CCDE-FID	$N\_pop = 80$

TABLE IV
EXPERIMENTAL RESULTS IN TEST I

Inst.	GPSO	BBPSO	CSO	SEO	RDA	CCDE -FID	CCBBPSO -AD	CCBBPSO -FID
YL	8746	8692	8538	8246	8699	6761	*6002904*	**6701**
YM	8598	8537	8384	8148	8541	*14215*	*5951115*	**6747**
YH	8470	8403	8241	8047	8410	*14273*	*6165789*	**6806**
DL	8611	8555	8395	8160	8559	6822	*5869688*	**6760**
DM	8597	8539	8386	8139	8544	*21615*	*6084346*	**6746**
DH	8607	8538	8383	8142	8547	6811	*6165775*	**6748**
Norm	8560	8501	8345	8095	8504	6766	*6010300*	**6704**
Log	8633	8573	8417	8184	8579	*14250*	*6210198*	**6783**
Tri	8622	8558	8403	8162	8567	*14232*	*5899310*	**6767**
Avg.	8605	8544	8388	8147	8550	*11750*	*6039936*	**6752**

and the results are shown as Tables S.III–S.V in the supplementary material. The optimal parameter settings of algorithms are shown in Table III. The settings of CCBBPSO-AD and CCBBPSO-RD are the same as that of CCBBPSO-FID. Each instance is tested 30 times independently, and the mean results are recorded.

C. Experimental Comparisons With Other Algorithms

Exhaustive experiments are conducted on 405 instances, including Test I, Test II, and Test III which are shown in Table II. Each instance set contains 135 (5×27) instances. For saving space, these instances (Inst.) are classified in three different ways: 1) based on the level of ***S_yield***, each test set is divided into YL, YM, and YH; 2) based on the CV level of ***C_demand***, each test set is divided into DL, DM, and DH; and 3) based on the PDF of ***C_demand***, each test set is divided into Norm, Log, and Tri. Each group includes 45 (135/3) instances (each instance has an average result of 30 times), and the average result [*totalCost* in (21)] of each group is shown in Tables IV–VI. For example, to calculate the average result of the group "YL," we first calculate the average result of 30 times for each instance in YL, and then calculate the average result of 45 instances. The last row (Avg.) in tables shows the average result of each set. Results in *italic* type mean that the global best solution is infeasible in some tests. Results in bold type are the best results among all algorithms. Besides, all results of CCBBPSO-RD and the results of SEO and RDA on Test III are equal to 10 000 000 and, therefore, are not shown in tables, indicating that CCBBPSO-RD, SEO, and RDA cannot obtain feasible solutions on the corresponding test sets.

TABLE V
EXPERIMENTAL RESULTS IN TEST II

Inst.	GPSO	BBPSO	CSO	SEO	RDA	CCDE -FID	CCBBPSO -AD	CCBBPSO -FID
YL	50089	49930	49431	49913	49761	45264	46683	**44762**
YM	49900	49877	48781	*6719802*	*1965556*	52589	46723	**44712**
YH	49365	49369	48462	*10000000*	*9985254*	52582	54093	**44724**
DL	49840	49804	48956	*5570287*	*4014989*	45285	46753	**44795**
DM	49747	49686	48850	*5562861*	*4007521*	45196	46676	**44695**
DH	49766	49686	48868	*5636566*	*3978060*	59953	54069	**44709**
Norm	49450	49409	48565	*5592200*	*3896793*	44901	46391	**44411**
Log	50011	49952	49110	*5548231*	*4110893*	60196	54305	**44945**
Tri	49892	49815	48999	*5629284*	*3992885*	45337	46803	**44842**
Avg.	49784	49725	48891	*5589905*	*4000190*	50145	49166	**44733**

TABLE VI
EXPERIMENTAL RESULTS IN TEST III

Inst.	GPSO	BBPSO	CSO	CCDE -FID	CCBBPSO -AD	CCBBPSO -FID
YL	163107	163189	160581	154888	158192	**151036**
YM	162311	162313	160768	154553	157917	**150713**
YH	162169	162168	161815	154103	157490	**150336**
DL	162786	162800	161239	154696	158023	**150899**
DM	162377	162370	160926	154408	157742	**150582**
DH	162424	162500	160998	154440	157834	**150604**
Norm	161500	161568	160054	153543	156896	**149725**
Log	163141	163183	161698	155159	158496	**151346**
Tri	162946	162918	161411	154843	158207	**151015**
Avg.	162529	162557	161055	154515	157866	**150695**

1) Comparisons Regarding the Scale of the Problem: From the observation of Table IV, it can be seen that CCBBPSO-FID yields the minimum cost in each group with an obvious advantage, followed by CSO and SEO. On the contrary, CCBBPSO-AD and CCDE-FID obtain infeasible solutions in most of the test groups. As S in this dataset is small, the frequency of LRO_{ap} (related to repair probability) drops, so some algorithms cannot obtain feasible solutions. GPSO, BBPSO, and RDA have similar results, and they perform worse than CCBBPSO-FID.

In Table V, the results show that CCBBPSO-FID always obtains the minimum cost results on all the tested instances, and only its mean result is lower than 45 000. Among the three well-performed CC algorithms, both CCDE-FID and CCBBPSO-AD are worse than CCBBPSO-FID. CSO, the algorithm for large-scale optimization, performs worse than CCBBPSO-FID and obtains better results than GPSO, BBPSO, CCDE-FID, and CCBBPSO-AD. Besides, SEO and RDA cannot obtain the feasible solutions in most test groups.

As shown in Table VI, although the data scale increases, the performance of CCBBPSO-FID does not deteriorate, and all results of it on Test III are still the best. Other versions of the CC algorithms, CCDE-FID and CCBBPSO-AD, perform better than other contestant algorithms. The results of CSO are better than those of classical PSOs (GPSO and BBPSO) and are beaten by the CC algorithms.

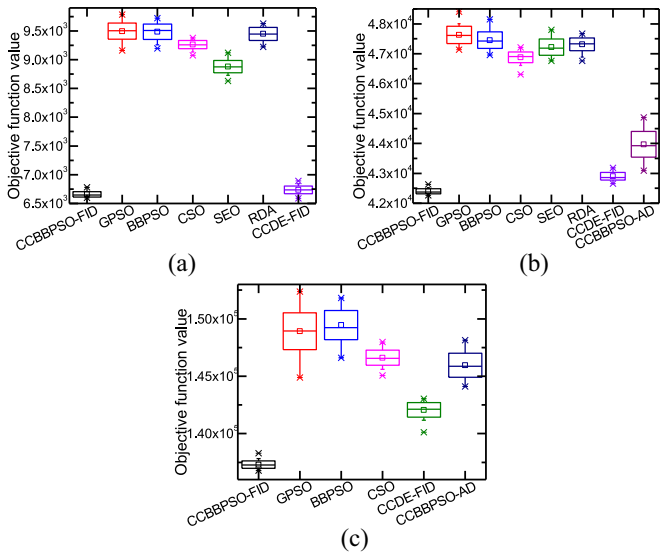

Fig. 10. Box Plots of Results. (a) I_0_YL_DL_Norm. (b) II_0_YL_DL_Norm. (c) III_0_YL_DL_Norm.

In conclusion, CCBBPSO-FID shows an absolute advantage over the compared algorithms in different data scales. In addition, with the data scale increasing, CCBBPSO-FID still remains the best algorithm. Generally speaking, CSO, the popular algorithm for the large-scale optimization problems, also has promising performance on the LUSCND problem, and often does much better than GPSO, BBPSO, SEO, and RDA. However, the CC algorithms, including CCDE-FID, CCBBPSO-AD, and CCBBPSO-FID, show a better performance on Test II and Test III. It indicates that the CC strategy is more effective to deal with the LUSCND problem and more suitable for large-scale optimization. The excellent performance of both CCDE-FID and CCBBPSO-FID further shows that the FID is a promising strategy for large-scale optimization. Besides, CCDE-FID is outperformed by CCBBPSO-FID. The reason may be that some promising solutions are eliminated in the selection operator of DE, which degrades the population diversity especially for the complicated COP. However, BBPSO reserves these solutions even if they are not improved.

For further illustration, box charts of three instances are shown in Fig. 10, and situations of other instances are similar. I_0_YL_DL_Norm, II_0_YL_DL_Norm, and III_0_YL_DL_Norm are with the uncertain setting of YL, DL, and Norm, and are chosen from Test I, Test II, and Test III, respectively. As revealed by figures, CCBBPSO-FID not only performs much better than contestant algorithms but also is much more robust (with the minimum volume). It indicates that CCBBPSO-FID is excellent to solve the LUSCND problem of either small-scale or large-scale data. Moreover, the Wilcoxon rank-sum test at a 5% significance level is used for the statistical comparisons. The results of nine typical instances from three different data scales are shown in Table S.VI in the supplementary material. It can be seen that CCBBPSO-FID has a significant advantage over all the other compared algorithms on the nine instances.

2) Comparisons Regarding the Sensibility to Uncertainties: From the aspect of ***S_yield***, based on the observation of Tables V and VI, it can be revealed that results mostly become smaller as the yield level gets higher (from YL to YH). YH includes a wider range [0.6, 0.9] with smaller values of the yield rate. It can be seen that if the yield rate decreases, the corresponding transport and inventory fees will decrease. Therefore, results of YH are smaller than those of YM and YL.

From the aspect of ***C_demand***, it can be observed that results are smaller when the demand level is higher (from DL to DH). High demand level means that there is more orders of suppliers. As a result, the unused capacity of suppliers will be less, and the corresponding penalty will decrease. Results are also different with different PDFs of demands (Norm, Log, and Tri). From the observation of Tables V and VI, it can be concluded that the results of Norm are smaller than those of Log and Tri, and the results of Log are the biggest.

To further analyze the influence of uncertainties on these algorithms, the mean maximum difference between results of different uncertainties is recorded, as shown in Table VII. For example, for the results of CCBBPSO-FID on Test I, the maximum difference among YL/YM/YH ($6806 - 6701 = 105$), DL/DM/DH ($6760 - 6746 = 14$), Norm/Log/Tri ($6792 - 6716 = 76$) and is 105. It can be seen that the difference of CCBBPSO-FID is smaller than that of other algorithms, which reveals that CCBBPSO-FID is less sensible to uncertain factors. The slight sensitivity is mainly due to that FID divides solutions into relatively independent parts which are solved by different subpopulations, helping to reduce the influence of uncertain factors on the whole. The slight sensibility also helps to enhance the robustness of CCBBPSO-FID, as shown in Fig. 10.

TABLE VII
MAXIMUM DIFFERENCE BETWEEN RESULTS OF DIFFERENT UNCERTAINTIES

No.	GPSO	BBPSO	CSO	SEO	RDA	CCDE -FID	CCBBPSO -AD	CCBBPSO -FID
I	276	289	297	199	289	*14804*	*310887*	**105**
II	724	561	969	*9935493*	*9950087*	15296	7915	**534**
III	1642	1615	1644	--	--	1616	**1601**	1621
Avg.	881	822	970	--	--	10572	106801	**753**

D. Analysis of Convergence Speed

To analyze the convergence speed of algorithms, the convergence curves are drawn for three instances, as shown in Fig. 11, and situations of other instances are similar. Since the maximum FEs of algorithms are different within the same execution time, the length of the line for each algorithm is different in the figures. From the observation of figures, it can be concluded that the convergence speeds of the CC algorithms (CCBBPSO-FID, CCBBPSO-AD, and CCDE-FID) are slower than those of other algorithms in the beginning. However, after some iterations, the evolution in non-CC algorithms (GPSO, BBPSO, CSO, SEO, and RDA) stagnates quickly. On the other hand, the CC algorithms surpass other algorithms afterward, except for CCBBPSO-AD on I_0_YL_DL_Norm, and CCBBPSO-FID obtains the best results in the end. The reason is that the CC algorithms solve the problem with four subpopulations and take four times FEs more than other algorithms to

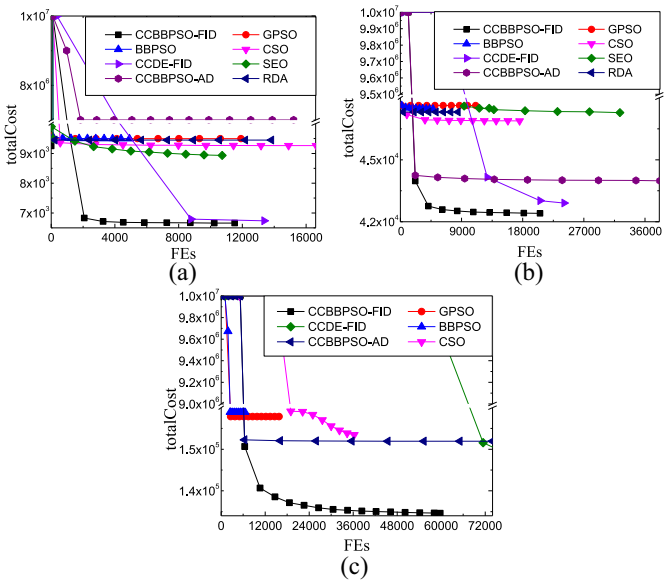

Fig. 11. Convergence curves of different algorithms. (a) I_0_YL_DL_Norm. (b) II_0_YL_DL_Norm. (c) III_0_YL_DL_Norm.

optimize the entire solutions. Therefore, they converge slowly at first but then converge quickly with the cooperative cooperation and the accumulated evolutionary information of all subpopulations. Moreover, FID decomposes the problem effectively based on different functions of different SCN members in the LUSCND problem, and it also helps CCBBPSO-FID and CCDE-FID to obtain the promising solutions.

E. Effectiveness of the CC Strategy

By comparing BBPSO with CCBBPSO-FID and CCBBPSO-AD in Tables V and VI, it can be seen that with the CC strategy, two CCBBPSO algorithms obtain better results, and the difference is obvious. CCBBPSO decomposes the problem into different parts, and these parts are solved separately by different subpopulations. These operations enhance the local search ability effectively. However, BBPSO only uses one population to deal with the problem, which limits the search ability of the algorithm. Further, the CC algorithms also perform superior to other competitor algorithms.

F. Effectiveness of FID

Random decomposition and average decomposition are designed to validate the effectiveness of FID. As mentioned above, CCBBPSO-RD that decomposes the problem aimlessly cannot obtain feasible solutions in all tests. By analyzing Tables IV–VI, it can be revealed that CCBBPSO-AD shows poorer performance than CCBBPSO-FID. The reason is that the average decomposition only divides the problem in the same size without considering the characteristics of the problem structure. Differently, FID decomposes the problem based on different functions of SCN members, and these functions are used to deal with different parts of the solution space. As a result, FID helps the algorithm to solve the problem more effectively. In addition, it improves the overall convergence speed of CCBBPSO, as shown in Section IV-D.

TABLE VIII
RESULTS OF ALGORITHMS WITH AND WITHOUT LROAP ON TEST I

Repair	GPSO	BBPSO	CSO	SEO	RDA	CCDE -FID	CCBBPSO -AD	CCBBPSO -FID
Yes	8605	8544	8388	8147	8550	11750	6039936	**6752**
No	*1104074*	*1067004*	***50427***	*35458*	***484774***	*10000000*	*10000000*	*10000000*

TABLE IX
SUCCESS RATE OF LROAP (%)

No.	GPSO	BBPSO	CSO	SEO	RDA	CCDE -FID	CCBBPSO -AD	CCBBPSO -FID
I	0.03	0.08	0.39	4.37	0.06	3.79	0.05	**6.42**
II	0.03	0.09	0.46	3.62	0.06	2.72	0.26	**7.29**
III	0.01	0.02	0.12	--	--	1.30	0.04	**8.67**
Avg.	0.02	0.06	0.32	--	--	2.60	0.12	**7.46**

G. Effectiveness of LRO_{ap}

Table VIII shows the results of algorithms with and without LRO_{ap} on Test I. It is obvious that algorithms without LRO_{ap} cannot obtain feasible results (the data are in *italic* type). Moreover, results of all algorithms without LRO_{ap} on Test II and Test III are approximately equal to 10 000 000 and not shown. It means that all algorithms without LRO_{ap} cannot obtain feasible solutions in large-scale data.

The execution number and the success number of LRO_{ap} are recorded during the evolution. The mean success rate of LRO_{ap} in all algorithms is shown in Table IX. Remarkably, CCBBPSO-FID has the highest success rate among all the algorithms. It indicates that solutions obtained by CCBBPSO-FID are much more promising and closer to feasible ones. Furthermore, with the data scale bigger, it is more difficult for compared algorithms to repair solutions successfully, but the situation is contrary to CCBBPSO-FID. It can also be concluded that compared with the average decomposition of CCBBPSO-AD, FID of CCBBPSO-FID is more capable of finding feasible solutions. The highest mean success rate of all instances is up to 7.46%, and it proves that LRO_{ap} is effective for the LUSCND problems.

V. CONCLUSION

SCND has important commercial values because it can help enterprises manage and plan the production line well and, then, cut cost and increase revenue. LUSCND is a more popular variant that extends the SCND model with large-scale and uncertain factors, but it is more difficult for existing approaches to solve. In this article, the efficient CCBBPSO-FID algorithm is proposed to handle this complex optimization problem. For the uncertain factors, MCM is used to calculate the fitness values in the uncertain environment. For the large-scale issue, the solution structure is first modified with the dimensionality reduction. Then, the CC strategy is combined with BBPSO to decompose the problem and to solve the decomposed parts with different subpopulations. Moreover, FID based on different functions of SCN members is devised

in the CC strategy for the effective decomposition of the problem. At last, the boundary processing method STM and the repair operator LRO_{ap} are proposed to repair the infeasible solutions. LRO_{ap} can also be used on other problems with multiple constraints, since it handles constraints separately and finds feasible solutions efficiently.

Three scales of the instance set are generated to validate the effectiveness and efficiency of CCBBPSO-FID, and the dimension of the largest scale is up to 195 080. The experimental results show that CCBBPSO-FID outperforms all the compared algorithms on all the data scales, and the increasing of the data scale does not hamper the performance of CCBBPSO-FID. Moreover, the CC strategy with FID helps the proposed algorithm to have a slight sensibility to uncertainties and to perform robustly. The LRO_{ap} is efficient to repair infeasible solutions of the LUSCND problem with complicated constraints.

Therefore, it is expected that CCBBPSO-FID will make significant contributions to the LUSCND application. To this aim, Fig. 3 and explanations in Section II-A can give SCN managers clear guidance to utilize the CCBBPSO-FID algorithm and its solutions. Nevertheless, SCND is still a developing research topic and many new features may appear in the new application scenarios. Therefore, future work will include solving the LUSCND problems with more factors, such as environmental dimensions [2], social dimensions [12], and routing decisions [41], which may be solved by the multiobjective [42]–[44], many-objective [45], and multimodal [46]–[48] algorithms.

REFERENCES

[1] M.-C. Chen, Y.-H. Hsiao, and H.-Y. Huang, "Semiconductor supply chain planning with decisions of decoupling point and VMI scenario," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 5, pp. 856–868, May 2017.

[2] A. M. Fathollahi-Fard, M. Hajiaghaei-Keshteli, and S. Mirjalili, "Hybrid optimizers to solve a tri-level programming model for a tire closed-loop supply chain network design problem," *Appl. Soft Comput.*, vol. 70, pp. 701–722, Sep. 2018.

[3] S. Qi, Y. Zheng, M. Li, L. Lu, and Y. Liu, "Secure and private RFID-enabled third-party supply chain systems," *IEEE Trans. Comput.*, vol. 65, no. 11, pp. 3413–3426, Nov. 2016.

[4] H. Reefke and D. Sundaram, "Key themes and research opportunities in sustainable supply chain management—Identification and evaluation," *Omega B*, vol. 66, pp. 195–211, Jan. 2017.

[5] M. Hajiaghaei-Keshteli, K. S. Abdallah, and A. M. Fathollahi-Fard, "A collaborative stochastic closed-loop supply chain network design for tire industry," *Int. J. Eng.*, vol. 31, no. 10, pp. 1715–1722, Oct. 2018.

[6] Y. Zhou, D.-C. Gong, B. Huang, and B. A. Peters, "The impacts of carbon tariff on green supply chain design," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 3, pp. 1542–1555, Jul. 2017.

[7] S. Haykin and P. Setoodeh, "Cognitive radio networks: The spectrum supply chain paradigm," *IEEE Trans. Cogn. Commun. Netw.*, vol. 1, no. 1, pp. 3–28, Mar. 2015.

[8] A. M. Fathollahi-Fard and M. Hajiaghaei-Keshteli, "A stochastic multi-objective model for a closed-loop supply chain with environmental considerations," *Appl. Soft Comput.*, vol. 69, pp. 232–249, Aug. 2018.

[9] A. M. Fathollahi-Fard, M. Hajiaghaei-Keshteli, and S. Mirjalili, "Multi-objective stochastic closed-loop supply chain network design with social considerations," *Appl. Soft Comput.*, vol. 71, pp. 505–525, Oct. 2018.

[10] R. W. Salem and M. Haouari, "A simulation-optimisation approach for supply chain network design under supply and demand uncertainties," *Int. J. Prod. Res.*, vol. 55, no. 7, pp. 1845–1861, 2017.

[11] M. Hajiaghaei-Keshteli and A. M. Fathollahi-Fard, "A set of efficient heuristics and metaheuristics to solve a two-stage stochastic bi-level decision-making model for the distribution network problem," *Comput. Ind. Eng.*, vol. 123, pp. 378–395, Sep. 2018.

[12] N. Sahebjamnia, A. M. Fathollahi-Fard, and M. Hajiaghaei-Keshteli, "Sustainable tire closed-loop supply chain network design: Hybrid metaheuristic algorithms for large-scale networks," *J. Clean. Prod.*, vol. 196, pp. 273–296, Sep. 2018.

[13] A. Samadi, N. Mehranfar, A. M. F. Fard, and M. Hajiaghaei-Keshteli, "Heuristic-based metaheuristics to address a sustainable supply chain network design problem," *J. Ind. Prod. Eng.*, vol. 35, no. 2, pp. 102–117, 2018.

[14] R. Shang, K. Dai, L. Jiao, and R. Stolkin, "Improved memetic algorithm based on route distance grouping for multiobjective large scale capacitated arc routing problems," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 1000–1013, Apr. 2016.

[15] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Trans. Math. Softw.*, vol. 42, no. 2, p. 13, Feb. 2016.

[16] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "DG2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 6, pp. 929–942, Dec. 2017.

[17] P. Chootinan and A. Chen, "Constraint handling in genetic algorithms using a gradient-based repair method," *Comput. Oper. Res.*, vol. 33, no. 8, pp. 2263–2281, Aug. 2006.

[18] Y. Huang, L. Wang, W. Guo, Q. Kang, and Q. Wu, "Chance constrained optimization in a home energy management system," *IEEE Trans. Smart Grid*, vol. 9, no. 1, pp. 252–260, Jan. 2018.

[19] J. Kennedy, "Bare bones particle swarms," in *Proc. IEEE Swarm Intell. Symp.*, 2013, pp. 80–87.

[20] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.

[21] A. M. Fathollahi-Fard, M. Hajiaghaei-Keshteli, and R. Tavakkoli-Moghaddam, "The social engineering optimizer (SEO)," *Eng. Appl. Artif. Intell.*, vol. 72, pp. 267–293, Jun. 2018.

[22] Z. H. Zhan *et al.*, "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 704–716, Mar. 2017.

[23] C. A. Silva, J. M. C. Sousa, T. A. Runkler, and J. M. G. S. da Costa, "Distributed supply chain management using ant colony optimization," *Eur. J. Oper. Res.*, vol. 199, no. 2, pp. 349–358, Dec. 2009.

[24] M. Campos, R. A. Krohling, and I. Enriquez, "Bare bones particle swarm optimization with scale matrix adaptation," *IEEE Trans. Cybern.*, vol. 44, no. 9, pp. 1567–1578, Sep. 2014.

[25] H. Wang, S. Rahnamayan, H. Sun, and M. G. H. Omran, "Gaussian bare-bones differential evolution," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 634–647, Apr. 2013.

[26] J.-H. Zhang, Y. Zhang, and Y. Zhou, "Path planning of mobile robot based on hybrid multi-objective bare bones particle swarm optimization with differential evolution," *IEEE Access*, vol. 6, pp. 44542–44555, 2018.

[27] J. Li and Y. Tan, "The bare bones fireworks algorithm: A minimalist global optimizer," *Appl. Soft Comput.*, vol. 62, pp. 454–462, Jan. 2018.

[28] X.-F. Liu, Z.-H. Zhan, and J. Zhang, "Neural network for change direction prediction in dynamic optimization," *IEEE Access*, vol. 6, pp. 72649–72662, 2018.

[29] M. Rong, D. Gong, Y. Zhang, Y. Jin, and W. Pedrycz, "Multi-directional prediction approach for dynamic multi-objective optimization problems," *IEEE Trans. Cybern.*, vol. 49, no. 9, pp. 3362–3374, Sep. 2019.

[30] X.-F. Liu *et al.*, "Neural network-based information transfer for dynamic optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published. doi: 10.1109/TNNLS.2019.2920887.

[31] W.-A. Zhang, M. Z. Q. Chen, A. Liu, and S. Liu, "Aperiodic optimal linear estimation for networked systems with communication uncertainties," *IEEE Trans. Cybern.*, vol. 47, no. 8, pp. 2256–2265, Aug. 2017.

[32] S. Salomon, G. Avigad, P. J. Fleming, and R. C. Purshouse, "Active robust optimization: Enhancing robustness to uncertain environments," *IEEE Trans. Cybern.*, vol. 44, no. 11, pp. 2221–2231, Nov. 2014.

[33] J.-S. Wang and G.-H. Yang, "Output-feedback control of unknown linear discrete-time systems with stochastic measurement and process noise via approximate dynamic programming," *IEEE Trans. Cybern.*, vol. 48, no. 7, pp. 1977–1988, Jul. 2018.

[34] J. A. Littlefield, J. Marriott, G. A. Schivley, and T. J. Skone, "Synthesis of recent ground-level methane emission measurements from the U.S. natural gas supply chain," *J. Clean. Prod.*, vol. 148, pp. 118–126, Apr. 2017.

[35] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 1994, pp. 249–257.

[36] Z.-J. Wang *et al.*, "Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2019.2933499.

[37] B. Xu, Y. Zhang, D. Gong, Y. Guo, and M. Rong, "Environment sensitivity-based cooperative co-evolutionary algorithms for dynamic multi-objective optimization," *IEEE/ACM Trans. Comput. Biol. Bioinformat.*, vol. 15, no. 6, pp. 1877–1890, Nov. 2017.

[38] S. Zheng, J. Li, A. Janecek, and Y. Tan, "A cooperative framework for fireworks algorithm," *IEEE/ACM Trans. Comput. Biol. Bioinformat.*, vol. 14, no. 1, pp. 27–41, Jan./Feb. 2017.

[39] C. Saha, S. Das, K. Pal, and S. Mukherjee, "A fuzzy rule-based penalty function approach for constrained evolutionary optimization," *IEEE Trans. Cybern.*, vol. 46, no. 12, pp. 2953–2965, Dec. 2016.

[40] K. D. Carlson and J. Wu, "The illusion of statistical control: Control variable practice in management research," *Org. Res. Methods*, vol. 15, no. 3, pp. 413–435, Jun. 2012.
[41] G. Yu, F. Li, and Y. Yang, "Robust supply chain networks design and ambiguous risk preferences," *Int. J. Prod. Res.*, vol. 55, no. 4, pp. 1168–1182, 2017.
[42] Z.-G. Chen *et al.*, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2912–2926, Aug. 2019.
[43] Y. Zhang, D.-W. Gong, and J. Cheng, "Multi-objective particle swarm optimization approach for cost-based feature selection in classification," *IEEE/ACM Trans. Comput. Biol. Bioinformat.*, vol. 14, no. 1, pp. 64–75, Jan./Feb. 2017.
[44] X. Li and K.-C. Wong, "Evolutionary multiobjective clustering and its applications to patient stratification," *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1680–1693, May 2019.
[45] X.-F. Liu, Z.-H. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, "Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 587–602, Aug. 2019.
[46] Z.-J. Wang *et al.*, "Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 6, pp. 894–908, Dec. 2018.
[47] H. Zhao *et al.*, "Local binary pattern-based adaptive differential evolution for multimodal optimization problems," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2019.2927780.
[48] Z.-J. Wang *et al.*, "Automatic niching differential evolution with contour prediction approach for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, to be published. doi: 10.1109/TEVC.2019.2910721.

Xin Zhang (S'17) received the B.S. and M.S. degrees from Northeast Normal University, Changchun, China, in 2014 and 2017, respectively. She is currently pursuing the Ph.D. degree with the South China University of Technology, Guangzhou, China.

Her current research interests include evolutionary computation, swarm intelligence, and their applications in real-world problems, such as supply chain network design.

Ke-Jing Du received the B.S. degree from the School of Business, Sun Yat-sen University, Guangzhou, China, in 2012, and the M.S. degree in operation and supply chain management from the Department of Management Science, City University of Hong Kong, Hong Kong, in 2014.

She is currently a Research Assistant with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. Her current research interests include evolutionary computation and supply chain management, especially intelligent application in supply chain network, scheduling, and control.

Zhi-Hui Zhan (M'13–SM'18) received the bachelor's and Ph.D. degrees from the Department of Computer Science, Sun Yat-sen University, Guangzhou, China, in 2007 and 2013, respectively.

From 2013 to 2015, he was a Lecturer and an Associate Professor with the Department of Computer Science, Sun Yat-sen University. Since 2016, he has been a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, where he is also the Changjiang Scholar Young Professor and the Pearl River Scholar Young Professor. His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, and their applications in real-world problems, and in environments of cloud computing and big data.

Dr. Zhan was a recipient of the China Computer Federation Outstanding Ph.D. Dissertation for his Doctoral thesis, the IEEE Computational Intelligence Society Outstanding Ph.D. Dissertation, the Outstanding Youth Science Foundation from National Natural Science Foundations of China in 2018, and the Wu Wen Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence in 2017. He is listed as one of the Most Cited Chinese Researchers in Computer Science. He is currently an Associate Editor of *Neurocomputing*.

Sam Kwong (M'93–SM'04–F'14) received the B.S. degree in electrical engineering from the State University of New York at Buffalo, Buffalo, NY, USA, in 1983, the M.S. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1985, and the Ph.D. degree from the University of Hagen, Hagen, Germany, in 1996.

From 1985 to 1987, he was a Diagnostic Engineer with Control Data Canada, Ottawa, ON, Canada, where he designed the diagnostic software to detect the manufacture faults of the very large-scale integration chips in the Cyber 430 machine. Then, he joined Bell Northern Research, Ottawa, ON, Canada, as a Member of Scientific Staff. In 1990, he joined the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, as a Lecturer, where he is currently a Professor with the Department of Computer Science. His current research interests include pattern recognition, evolutionary computations, and video analytics.

Prof. Kwong has been the Vice President for IEEE Systems, Man and Cybernetics for conferences and meetings from since 2014. He was elevated to an IEEE Fellow for his contributions on optimization techniques for cybernetics and video coding in 2014. He is also appointed as an IEEE Distinguished Lecturer for IEEE SMC society in March 2017.

Tian-Long Gu received the M.Eng. degree from Xidian University, Xi'an, China, in 1987, and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 1996.

From 1998 to 2002, he was a Research Fellow with the School of Electrical and Computer Engineering, Curtin University of Technology, Perth, WA, Australia, and a Post-Doctoral Fellow with the School of Engineering, Murdoch University, Perth. He is currently a Professor with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin, China. His current research interests include formal methods, data and knowledge engineering, software engineering, and information security protocol.

Jun Zhang (F'17) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Visiting Scholar with Victoria University, Melbourne, VIC, Australia. His current research interests include computational intelligence, cloud computing, high performance computing, operations research, and power electronic circuits.

Dr. Zhang was a recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, and the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.