Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 19(1): 271–286.
DOI: 10.3934/mbe.2022014
Received: 15 September 2021
Accepted: 02 November 2021
Published: 12 November 2021

Research article

A coevolutionary algorithm based on the auxiliary population for constrained large-scale multi-objective supply chain network

Xin Zhang[1,2], Zhaobin Ma[1], Bowen Ding[1], Wei Fang[1] and Pengjiang Qian[1,*]

[1] School of Artificial Intelligence and Computer Science, and Jiangsu Key Laboratory of Media Design and Software Technology, Jiangnan University, Wuxi 214122, China
[2] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China

* **Correspondence:** Email: qianpjiang@126.com.

Abstract: Supply chain network is important for the enterprise to improve the operation and management, but has become more complicated to optimize in reality. With the consideration of multiple objectives and constraints, this paper proposes a constrained large-scale multi-objective supply chain network (CLMSCN) optimization model. This model is to minimize the total operation cost (including the costs of production, transportation, and inventory) and to maximize the customer satisfaction under the capacity constraints. Besides, a coevolutionary algorithm based on the auxiliary population (CAAP) is proposed, which uses two populations to solve the CLMSCN problem. One population is to solve the original complex problem, and the other population is to solve the problem without any constraints. If the infeasible solutions are generated in the first population, a linear repair operator will be used to improve the feasibility of these solutions. To validate the effectivity of the CAAP algorithm, the experiment is conducted on the randomly generated instances with three different problem scales. The results show that the CAAP algorithm can outperform other compared algorithms, especially on the large-scale instances.

Keywords: supply chain network; large-scale optimization; multi-objective optimization; constrained optimization; coevolutionary algorithm

1. Introduction

In the traditional manufacturing industry, enterprises always focus on their own single business, such as supplying raw materials, producing parts, assembling parts, or transporting. The isolating management pattern may be uncompetitive in the face of the volatile market, because enterprises cannot obtain complete information in time [1]. Afterwards, the pattern of supply chain management is proposed. Supply chain network (SCN) organizes enterprises from different stages together, and these enterprises work cooperatively and create a win-win situation [2,3]. Therefore, supply chain network (SCN) plays a significant role in the whole operation of the enterprise [4,5].

In the practical application of supply chain network, researchers can solve the SCN optimization problem through the mathematical modeling, the simulation, the data-driven, the game model, and other methods [6–8]. Evolutionary algorithms adopt a set of solutions (i.e., the population composed of multiple solution individuals) to solve problems, which have good global optimization ability and have become important methods to solve complex optimization problems in recent years [9,10]. For example, to solve the constrained optimization problems, evolutionary algorithms were combined with the random direction repair which was an effective constraint-handling method [11]. A constraint-objective cooperative coevolution framework was proposed to solve the large-scale constrained optimization problems, which allocated different computing resources to the decomposed sub-problems according to their contributions [12]. Therefore, evolutionary algorithms are often used to solve the complicated SCN optimization problems [13–15].

For example, Wang et al. [16] used genetic algorithm to choose suppliers and distribution centers and optimize the production quantity and the transportation volume in the iron and steel supply chain network, so as to minimize the operation cost and carbon emission of the network system. The problem is a multi-objective optimization problem of mixed integer programming with constraints. Sun et al. [17] used ant colony optimization algorithm to solve the scheduling optimization problem of mass customization supply chain and to minimize the production scheduling time by determining the optimal scheduling of cooperative suppliers. Zhang et al. [18] used the cooperative particle swarm optimization algorithm to solve the large-scale supply chain network design problem in the uncertain environment and to minimize the operation cost of the whole system by determining the optimal supplier and warehouse location scheme and the transportation logistics between network nodes. Akkad et al. [19] used a multi-objective heuristic approach to solve the collection and distribution problems of city logistics and to minimize both the fuel consumption and the emission of greenhouse gases. Saragih et al. [20] used a heuristic method to solve a location-inventory-routing problem in a three-echelon supply chain system with large scales. These problems have the characteristics of large scale, multiple objectives, or constraints which are the common difficulties of the SCN problems in the reality. However, the researchers did not solve the problem which has the three characteristics simultaneously.

In this paper, a constrained large-scale multi-objective supply chain network (CLMSCN) model is proposed, which is much closer to the reality and considers large scale, multiple objectives, and constraints in the same time. This model is to optimize the total operation cost and the customer satisfaction under the capacity constraints simultaneously. The total operation cost includes the production cost of the products, the inventory cost of the products, and the transportation cost among different SCN members. The proposed CLMSCN is a demand-driven model, which starts from the customers raising the demand to the SCN platform. According to the demand quantity of customers, the SCN members on the platform, including distributors and suppliers, begins to ordering, supplying, or distributing. This model is more competitive in the industry 4.0 era, since it starts from the market demand directly and helps to improve the operation efficiency with lower cost.

Besides, this paper proposes a coevolutionary algorithm based on the auxiliary population (CAAP), which uses two populations to solve the CLMSCN problem with complicated constraints. The first population is to solve the original problem with constraints, and a linear repair operator is designed to improve the feasibility of the infeasible solutions. The second population is to solve the problem without any constraints, which plays an auxiliary role to explore more solution space. With the cooperative work of the two populations, the CAAP algorithm can solve the CLMSCN problem in the experiment.

The rest of this paper is organized as follows. Section 2 describes the proposed CLMSCN problem in detail. Section 3 introduces all the components of the proposed CAAP algorithm. Afterwards, Section 4 discusses and analyzes the experimental results from different angles. Finally, Section 5 gives the conclusion of this paper.

2. Problem description of CLMSCN

To optimize the total operation cost and the customer satisfaction, the CLMSCN model is proposed in this paper. The illustration of the CLMSCN problem is shown in Figure 1, which involves three kinds of members (including suppliers, distributors, and customers) and four stages (including customers sending demands to distributors, distributors sending orders to suppliers, suppliers supplying materials to distributors, and distributors transporting materials to customers). The whole process will last for several time periods. The numbers of suppliers, distributors, customers, and time periods are denoted as *S*, *D*, *C* and *T*, correspondingly. Before the description of the four stages of the problem, all variables involved in the CLMSCN problem model are described as follows:

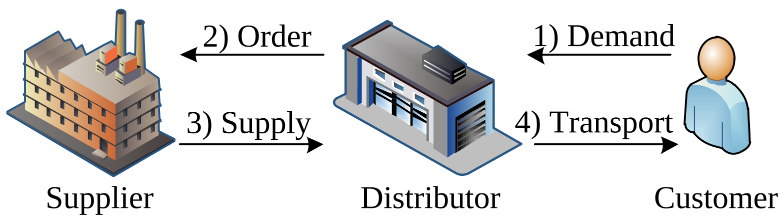


Figure 1. Illustration of the CLMSCN problem.

Indices:

i: index of suppliers, $i \in \{1, \ldots, S\}$
j: index of distributors, $j \in \{1, \ldots, D\}$
k: index of customers, $k \in \{1, \ldots, C\}$
t: index of time periods, $t \in \{1, \ldots, T\}$

Parameters:

S: number of suppliers
S_cap_i: capacity of the supplier i
S_pdCost_i: production cost per unit capacity and time of the supplier i
D: number of distributors
D_cap_j: capacity of the distributor j
$D\_inv_{j,t}$: inventory quantity of the distributor j in the period t
$D\_inv_{j,0}$: initial inventory quantity of the distributor j
$D\_invCost_j$: inventory cost per unit capacity and time of the distributor j
C: number of customers
$C\_dem_{k,t}$: demand quantity of the customer k in the period t
$SD\_tpCost_{i,j}$: transport cost per unit capacity and time from the supplier i to the distributor j
$DC\_tpCost_{j,k}$: transport cost per unit capacity and time from the distributor j to the customer k
T: number of time periods

Decision variables:

$DS\_ord_{i,j,t}$: order quantity of the supplier i from the distributor j in the period t
$DC\_tp_{j,k,t}$: transport quantity of the distributor j to the customer k in the period t

The whole process of the CLMSCN problem in Figure 1 can be described as follows:

Stage 1) **Demand**: Each customer sends demands to the supply chain platform in each time period ($C\_dem_{k,t}$). It should be noted that the total demand quantity of the customer k is known, but the demand quantity of the customer to each distributor is unknown.

Stage 2) **Order**: If the previous inventory cannot satisfy the customer demands, distributors will send orders to the suppliers ($DS\_ord_{i,j,t}$). In this stage, two capacity constraints should be satisfied. Firstly, in the period t, the total order quantity and the previous inventory of the distributor should not be larger than its capacity ($\Sigma_i(DS\_ord_{i,j,t}) + D\_inv_{j,t-1} \leq D\_cap_j$). Secondly, the total order quantity of a supplier should not be larger than its capacity ($\Sigma_j(DS\_ord_{i,j,t}) \leq S\_cap_i$). Besides, the production cost of suppliers should be expended ($Prod\_cost = \Sigma_i\Sigma_j\Sigma_t(S\_pdCost_i \times DS\_ord_{i,j,t})$).

Stage 3) **Supply**: Suppliers transport materials to distributors. The transport cost from suppliers to distributors ($Tp\_cost1 = \Sigma_i\Sigma_j\Sigma_t(SD\_tpCost_{i,j} \times DS\_ord_{i,j,t})$) and the inventory cost of distributors ($Inv\_cost = \Sigma_j\Sigma_t(D\_invCost_j \times D\_inv_{j,t})$) should be expended. Besides, the inventory quantity of distributors should be updated ($D\_inv_{j,t} = D\_inv_{j,t-1} + \Sigma_i(DS\_ord_{i,j,t}) - \Sigma_k(DC\_tp_{j,k,t}) = \Sigma_l(\Sigma_i(DS\_ord_{i,j,t}) - \Sigma_k(DC\_tp_{j,k,t})) + D\_inv_{j,0}$, $l \in \{1, \ldots, t\}$).

Stage 4) **Transport**: Distributors transport materials to customers ($DC\_tp_{j,k,t}$). The transport cost from distributors to customers ($Tp\_cost2 = \Sigma_j\Sigma_k\Sigma_t(DC\_tpCost_{j,k} \times DC\_tp_{j,k,t})$) should be expended. When the demands of all customers are completely satisfied, the customer satisfaction is highest.

There are two assumptions in the problem model:

1) The production quantity of a supplier is equal to the total order quantity.

2) The production time of suppliers and the transport time from suppliers to distributors and from distributors to customers are ignored.

The two objectives of this problem is to minimize the operation cost ($f1 = Prod\_cost + Tp\_cost1 + Inv\_cost + Tp\_cost2$) and to maximize the customer satisfaction ($f2 = \Sigma_k\Sigma_t(\Sigma_j(DC\_tp_{j,k,t})/C\_dem_{k,t})$), which are mutually exclusive. The second objective can be equivalent to minimize $f2 = \Sigma_k\Sigma_t(C\_dem_{k,t})/$

($\Sigma_j\Sigma_k\Sigma_t(DC\_tp_{j,k,t})$ + 1.0).

Based on the above descriptions, the final mathematical model of this CLMSCN problem can be described as follows:
Minimize

$$\begin{aligned} f1 = &\sum_{i=1}^{S} S\_pdCost_i \sum_{j=1}^{D}\sum_{t=1}^{T} DS\_ord_{i,j,t} + \sum_{i=1}^{S}\sum_{j=1}^{D} SD\_tpCost_{i,j} \times \sum_{t=1}^{T} DS\_ord_{i,j,t} \\ &+\sum_{j=1}^{D} D\_invCost_j \times \sum_{t=1}^{T}(\sum_{l=1}^{t}(\sum_{i=1}^{S} DS\_ord_{i,j,l} - \sum_{k=1}^{C} DC\_tp_{j,k,l}) + D\_inv_{j,0}) \\ &+\sum_{j=1}^{D}\sum_{k=1}^{C} DC\_tpCost_{j,k} \times \sum_{t=1}^{T} DC\_tp_{j,k,t} \end{aligned} \tag{1}$$

$$f2 = \frac{\sum_{k=1}^{C}\sum_{t=1}^{T} C\_dem_{k,t}}{\sum_{k=1}^{C}\sum_{t=1}^{T}\sum_{j=1}^{D} DC\_tp_{j,k,t} + 1.0} \tag{2}$$

Subject to

$$DS\_ord_{i,j,t} \geq 0, i \in \{1,...,S\}, j \in \{1,...,D\}, t \in \{1,...,T\} \tag{3}$$

$$DC\_tp_{j,k,t} \geq 0, j \in \{1,...,D\}, k \in \{1,...,C\}, t \in \{1,...,T\} \tag{4}$$

$$\sum_{j=1}^{D} DS\_ord_{i,j,t} \leq S\_cap_i, i \in \{1,...,S\}, t \in \{1,...,T\} \tag{5}$$

$$\begin{aligned} D\_inv_{j,t} &= D\_inv_{j,t-1} + \sum_{i=1}^{S} DS\_ord_{i,j,t} - \sum_{k=1}^{C} DC\_tp_{j,k,t}, j \in \{1,...,D\}, t \in \{1,...,T\} \\ &= \sum_{l=1}^{t}(\sum_{i=1}^{S} DS\_ord_{i,j,l} - \sum_{k=1}^{C} DC\_tp_{j,k,l}) + D\_inv_{j,0} \geq 0 \end{aligned} \tag{6}$$

$$\begin{aligned} &\sum_{i=1}^{S} DS\_ord_{i,j,t} + D\_inv_{j,t-1} = \\ &\sum_{i=1}^{S} DS\_ord_{i,j,t} + \sum_{l=1}^{t-1}(\sum_{i=1}^{S} DS\_ord_{i,j,l} - \sum_{k=1}^{C} DC\_tp_{j,k,l}) + D\_inv_{j,0} \leq D\_cap_j, \\ &j \in \{1,...,D\}, t \in \{1,...,T\} \end{aligned} \tag{7}$$

The operation cost ($f1$) and the customer satisfaction (equivalent to $f2$) are calculated as Eqs (1) and (2), correspondingly. As for the domain of variables, the decision variables $DS\_ord_{i,j,t}$ and $DC\_tp_{j,k,t}$ and the inventory quantity of distributors ($D\_inv_{j,t}$, which is updated in the third stage **Supply**) are all not smaller than zero, as shown in Eqs (3), (4), and (6), correspondingly. As for the capacity constraints in the second stage (**Order**), for the supplier i, its order quantity from all distributors ($\Sigma_j(DS\_ord_{i,j,t})$) should not exceed its capacity (S_cap_i), as shown in Eq (5). For the distributor j, its order quantity to all suppliers ($\Sigma_i(DS\_ord_{i,j,t})$) and previous inventory ($D\_inv_{j,t-1}$) should not exceed its capacity (D_cap_j), as shown in Eq (7).

3. CAAP algorithm

3.1. Solution encoding

The solution in the proposed CAAP approach is encoded into a single dimensional array, as shown in Figure 2. Each solution has ($dim = S{\times}D{\times}T{+}D{\times}C{\times}T$) dimensions. The first ($S{\times}D{\times}T$) dimensions are consisted of ***DS_ord***, and the last ($D{\times}C{\times}T$) dimensions are consisted of ***DC_tp***.

Dimension:	$1 \ldots S{\times}D{\times}T$	$S{\times}D{\times}T+1 \ldots S{\times}D{\times}T + D{\times}C{\times}T$
	DS_ord	***DC_tp***

Figure 2. Illustration of the solution encoding.

3.2. Initialization

Before the iteration, the two populations in the CAAP approach are randomly initialized. To generate more feasible solutions, reasonable lower and upper bounds should be determined. From the observation of Eqs (3) and (4), the lower bounds of ***DS_ord*** and ***DC_tp*** of the solution can be set as 0. For ***DS_ord*** of a solution, the decision variable $DS\_ord_{i,j,t}$ represents the order quantity of the distributor j to the supplier i in the period t, so it should not be larger than the capacity of the supplier i and the distributor j. Therefore, the upper bound of $DS\_ord_{i,j,t}$ is the smaller one between S_cap_i and D_cap_j ($\min(S\_cap_i, D\_cap_j)$). For ***DC_tp*** of a solution, the decision variable $DC\_tp_{j,k,t}$ represents the transport quantity of the distributor j to the customer k in the period t, so it should not be larger than the capacity of the supplier i and the demand quantity of the customer k. Therefore, the upper bound of $DC\_tp_{j,k,t}$ is the smaller one between D_cap_j and $C\_dem_{k,t}$ ($\min(D\_cap_j, C\_dem_{k,t})$). Finally, the decision variables are initialized randomly within the reasonable range.

3.3. Linear repair operator

If an infeasible solution is generated in the population, the linear repair operator will be used to improve the feasibility of the solution. It can be seen that the constraints (Eqs (5)–(7)) of this problem are all the linear functions of decision variables (***DS_ord*** and ***DC_tp***). Therefore, to improve the feasibility of solutions, the decision variables can be linearly changed according to the constraints Eqs (5)–(7).

From the observation of the constraints Eqs (5)–(7), three conclusions can be obtained as follows:

1) To improve the solutions which violate the constraint Eq (5), the decision variables in ***DS_ord*** should decrease.

2) To improve the solutions which violate the constraint Eq (6), the decision variables in ***DS_ord*** should increase, or the decision variables in ***DC_tp*** should decrease.

3) To improve the solutions which violate the constraint Eq (7), the decision variables in ***DS_ord*** should decrease, or the decision variables in ***DC_tp*** should increase.

Therefore, to maintain consistency as much as possible, three rules are formulated to repair infeasible solutions as follows:

1) To repair the infeasible solutions violating the constraint Eq (5), the decision variables in ***DS_ord*** will decrease proportionally.

2) To repair the infeasible solutions violating the constraint Eq (6), the decision variables in ***DC_tp*** will decrease proportionally.

3) To repair the infeasible solutions violating the constraint Eq (7), the decision variables in ***DS_ord*** will decrease proportionally.

For example, for the infeasible solutions violating the constraint Eq (5), the new $DS\_ord_{i,j,t}$ ($j \in \{1, \ldots, D\}$) need to approximately decrease by $(\Sigma_j(DS\_ord_{i,j,t}) - S\_cap_i)/D$ proportionally, and the details are shown in Algorithm 1. For the infeasible solutions violating the constraint Eq (6), $DC\_tp_{j,k,t}$ ($k \in \{1, \ldots, C\}$) need to approximately decrease by $-D\_inv_{j,t}/C$ proportionally, and the details are shown in Algorithm 2. For the infeasible solutions violating the constraint Eq (7), $DS\_ord_{i,j,t}$ ($i \in \{1, \ldots, S\}$) need to approximately decrease by $(\Sigma_i(DS\_ord_{i,j,t}) + D\_inv_{j,t\text{-}1} - D\_cap_j)/D$ proportionally, and the details are shown in Algorithm 3.

It should be noted that the repair operator is time-consuming. Therefore, a self-adaptive repair probability of each solution is set. The initial repair probability of each solution is set as $10^{-4} \times dim$. After an evaluation, if the solution is repaired, its repair probability will decrease by $10^{-5} \times dim$; otherwise, the probability will increase by $10^{-5} \times dim$.

```
Algorithm 1: Linear Repair Operator 1
Input: the solution including **DS_ord**, $i$, $t$, $D$, $T$
Output: the repaired solution
1:  Calculate $eValue = \Sigma_j(DS\_ord_{i,j,t}) - S\_cap_i$;
2:  $count = 0$;
3:  **While** $eValue > 0$ and $count < T$ **do**: // the loop will continue until $\Sigma_j(DS\_ord_{i,j,t}) > S\_cap_i$
    or the number of the loop iterations $count$ is smaller than $T$
4:      $decrease = eValue / D$;
5:      $sum = 0$;
6:      **For** $j \in \{1, \ldots, D\}$ **do**:
7:          **If** $DS\_ord_{i,j,t} > decrease$ **do**://after decreasing, $DS\_ord_{i,j,t}$ should not be smaller than 0
8:              $DS\_ord_{i,j,t} = DS\_ord_{i,j,t} - decrease$;
9:          $sum = sum + DS\_ord_{i,j,t}$;
10:     $eValue = sum - S\_cap_i$;
11:     $count = count + 1$;
```

Algorithm 2: Linear Repair Operator 2

Input: the solution including ***DC_tp***, *j*, *t*, *C*, *T*

Output: the repaired solution

```
1:  Calculate $eValue = - D\_inv_{j,t}$;
2:  Calculate $tmp = - eValue + \Sigma_k(DC\_tp_{j,k,t})$;
3:  $count = 0$;
4:  While $eValue > 0$ and $count < T$ do: // the loop will continue until the constraint (6) is
    satisfied or the number of the loop iterations count is smaller than T
5:      $decrease = eValue / C$;
6:      $sum = 0$;
7:      For $k \in \{1, \ldots, C\}$ do:
8:          If $DC\_tp_{j,k,t} > decrease$ do:
9:              $DC\_tp_{j,k,t} = DC\_tp_{j,k,t} - decrease$;
10:         $sum = sum + DC\_tp_{j,k,t}$;
11:     $eValue = sum - tmp$;
12:     $count = count + 1$;
```

Algorithm 3: Linear Repair Operator 3

Input: the solution including ***DS_ord***, *j*, *t*, *S*, *T*

Output: the repaired solution

```
1:  Calculate $eValue = \Sigma_i(DS\_ord_{i,j,t}) + D\_inv_{j,t-1} - D\_cap_j$;
2:  Calculate $tmp = eValue - \Sigma_i(DS\_ord_{i,j,t})$;
3:  $count = 0$;
4:  While $eValue > 0$ and $count < T$ do: // the loop will continue until the constraint (7) is
    satisfied or the number of the loop iterations count is smaller than T
5:      $decrease = eValue / S$;
6:      $sum = 0$;
7:      For $i \in \{1, \ldots, S\}$ do:
8:          If $DS\_ord_{i,j,t} > decrease$ do:
9:              $DS\_ord_{i,j,t} = DS\_ord_{i,j,t} - decrease$;
10:         $sum = sum + DS\_ord_{i,j,t}$;
11:     $eValue = sum + tmp$;
12:     $count = count + 1$;
```

3.4. Complete CAAP algorithm

The CAAP algorithm applies the coevolutionary constrained multi-objective optimization framework [21], and uses two populations (***pop1*** and ***pop2***) to solve the CLMSCN problem. ***pop1*** is to solve the original problem with complex constraints, and ***pop2*** is to solve the CLMSCN problem without constraints. Besides, the linear repair operator is designed to repair the infeasible solutions in ***pop1*** as much as possible.

Algorithm 4 shows the pseudo code of the CAAP algorithm in details. The input parameter, *Popsize*, is the population size of ***pop1*** and ***pop2***, and the output parameters, ***PS*** and ***PF***, are the Pareto solutions (the best solutions of a multi-objective problem) and their fitness values correspondingly. Firstly, ***pop1*** and ***pop2*** are initialized randomly, and the infeasible solutions in ***pop1*** are repaired by the linear repair operator. Then, the iterative process is conducted, and the termination condition of this algorithm in line 4 is set as the maximal running time. In the iterative process, ***Parent1*** and ***Parent2*** are selected from ***Parent2*** correspondingly by the roulette wheel selection. Then, in lines 7 and 8, offsprings are generated by NSGA_II with the simulated binary crossover and polynomial mutation [22–24]. Then, ***pop1*** and ***pop2*** are updated and evaluated in lines 9 to 13. In lines 12 and 13, the fast-non-dominated-sort in [22] and the farthest-candidate approach in [25] are used to select the non-dominated solutions. Finally, ***PS*** and ***PF*** are updated.

```
Algorithm 4: CAAP
Input: Popsize
Output: PS, PF
1:  Randomly initialize and evaluate pop1, and repair the solutions in pop1 according to the
    repair probabilities;
2:  Randomly initialize and evaluate pop2;
3:  Update PS and PF;
4:  While the termination condition is not satisfied do:
5:     Parent1 ← Select popsize/2 parents from pop1 by the roulette wheel selection;
6:     Parent2 ← Select popsize/2 parents from pop2 by the roulette wheel selection;
7:     Off1 ← Generate popsize/2 offsprings based on Parent1 by NSGA_II;
8:     Off2 ← Generate popsize/2 offsprings based on Parent2 by NSGA_II;
9:     pop1 ← pop1 ∪ Off1 ∪ Off2;
10:    pop2 ← pop2 ∪ Off1 ∪ Off2;
11:    Evaluate pop1 and pop2, and repair solutions in pop1;
12:    pop1 ← Select popsize solutions in pop1 by the fast-non-dominated-sort and the
    farthest-candidate approach;
13:    pop2 ← Select popsize solutions in pop2 by the fast-non-dominated-sort and the
    farthest-candidate approach;
14:    Update PS and PF;
```

4. Experimental verification and comparisons

4.1. Experimental settings

To validate the performance of the CAAP algorithm, three different scales of instance sets are randomly generated. Each scale has five instances, such as I_1 to I_5 of the small scale, II_1 to II_5 of the middle scale, and III_1 to III_5 of the large scale, and the configurations and the maximum execution time of the instance sets are shown in Table 1. Besides, MPCMO_BBPSO (multiple populations for constrained multi-objective optimization with bare-bones particle swarm optimization), PBBPSO (bare-bones particle swarm optimization with the Pareto optimality), NSGA_II (nondominated sorting genetic algorithm II, [22]), MOEA/D (multi-objective evolutionary algorithm based on decomposition, [26]), and NSLS (nondominated sorting and local search, [25]) are used to compare with the CAAP algorithm for the CLMSCN problem. MPCMO_BBPSO, which applies the multiple populations for multiple objectives framework [27], takes the constraints as a new objective and uses the bare-bones particle swarm optimization to solve the CLMSCN problem. PBBPSO uses bare-bones particle swarm optimization which is competitive in solving the supply chain optimization problem. NSGA_II, MOEA/D, and NSLS are the competitive algorithms for the multi-objective optimization problems. For the fair comparison, the population size of all algorithms is set 50, and the linear repair operator proposed in this paper is also used in the all compared algorithms to repair the infeasible solutions.

Table 1. Configurations and execution time of the instance sets.

No.	*S*	*W*	*C*	*T*	Dimension (*D*)	Execution time (Seconds)
I	5	10	15	10	2015	10
II	10	20	50	20	24030	100
III	30	50	100	30	195080	500

4.2. Compared results

Table 2. The HV values of all algorithms on different instances.

Instance	CAAP	MPCMO_BBPSO	PBBPSO	NSGA_II	MOEA/D	NSLS
I_1	**1.65E + 06**	1.38E + 06	7.64E + 05	1.07E + 06	1.20E + 06	9.98E + 05
I_2	3.89E + 06	3.04E + 06	2.05E + 06	2.38E + 06	**3.96E + 06**	2.50E + 06
I_3	**4.28E + 06**	3.41E + 06	2.19E + 06	2.49E + 06	3.81E + 06	2.58E + 06
I_4	4.44E + 06	3.65E + 06	2.17E + 06	3.23E + 06	**4.83E + 06**	2.93E + 06
I_5	**2.95E + 06**	2.37E + 06	1.06E + 06	1.64E + 06	2.57E + 06	1.84E + 06
Avg.	**3.44E + 06**	2.77E + 06	1.65E + 06	2.16E + 06	3.27E + 06	2.17E + 06
II_1	**6.02E + 05**	5.48E + 05	4.88E + 05	5.32E + 05	3.42E + 05	3.56E + 05
II_2	**9.01E + 05**	8.10E + 05	7.41E + 05	8.13E + 05	5.66E + 05	5.35E + 05
II_3	**1.53E + 06**	1.38E + 06	1.19E + 06	1.45E + 06	9.70E + 05	9.20E + 05
II_4	**9.53E + 05**	8.64E + 05	7.93E + 05	8.89E + 05	6.66E + 05	5.60E + 05
II_5	**7.94E + 05**	6.94E + 05	6.26E + 05	7.18E + 05	5.20E + 05	4.28E + 05
Avg.	**9.57E + 05**	8.59E + 05	7.67E + 05	8.81E + 05	6.13E + 05	5.60E + 05
III_1	**1.57E + 06**	1.39E + 06	1.29E + 06	1.52E + 06	9.80E + 05	6.59E + 05
III_2	**1.42E + 06**	1.23E + 06	1.14E + 06	1.37E + 06	8.17E + 05	5.91E + 05
III_3	**1.34E + 06**	1.17E + 06	1.11E + 06	1.30E + 06	8.57E + 05	5.45E + 05
III_4	**1.23E + 06**	1.09E + 06	1.00E + 06	1.20E + 06	7.26E + 05	5.17E + 05
III_5	**1.16E + 06**	1.03E + 06	9.25E + 05	1.13E + 06	7.07E + 05	5.16E + 05
Avg.	**1.34E + 06**	1.18E + 06	1.09E + 06	1.30E + 06	8.17E + 05	5.66E + 05

To evaluate the performance of the CAAP algorithm and other contestant algorithms from diverse angles, two metrics are used, HV (hypervolume) and C(A, B) (A and B are two algorithms) [28]. The HV value represents the hypervolume surrounded by the reference point and the Pareto front. For the minimization problem solved in this paper, if the HV value of an algorithm is higher, it means the algorithm is better than other algorithms. The C(A, B) value means the ratio of the solutions which are obtained by the algorithm B and are dominated by the solutions obtained by the algorithm A. Therefore, if the C(A, B) value is larger than the C(B, A) value, it means that the algorithm A can obtain better solutions than the algorithm B.

Table 2 shows the HV values of all algorithms on different instances. The data in bold represent the best result among all algorithms. It can be observed from Table 2 that for the instances in small scale, CAAP can obtain the best results on 3 instances, followed by MPCMO_BBPSO and MOEA/D. For the instances in middle and large scale, CAAP can obtain the best results on all instances, followed by MPCMO_BBPSO and NSGA_II. With the problem scale increasing, the advantage of the performance of CAAP does not deteriorate.

Table 3 and Table 4 show the C(A, B) values of all compared algorithms with CAAP on different instances. From the observation of Table 3, CAAP can obtain 2, 5 and 4 better results than MPCMO_BBPSO, PBBPSO, and NSGA_II on small instances, correspondingly. For the middle and large instances, CAAP obtains the best results on all instances compared with MPCMO_BBPSO, PBBSO, and NSGA_II. From the observation of Table 4, CAAP outperforms MOEA/D and NSLS on all different instances.

Table 3. The C(A, B) values of MPCMO_BBPSO, PBBPSO, and NSGA_II with CAAP on different instances (%).

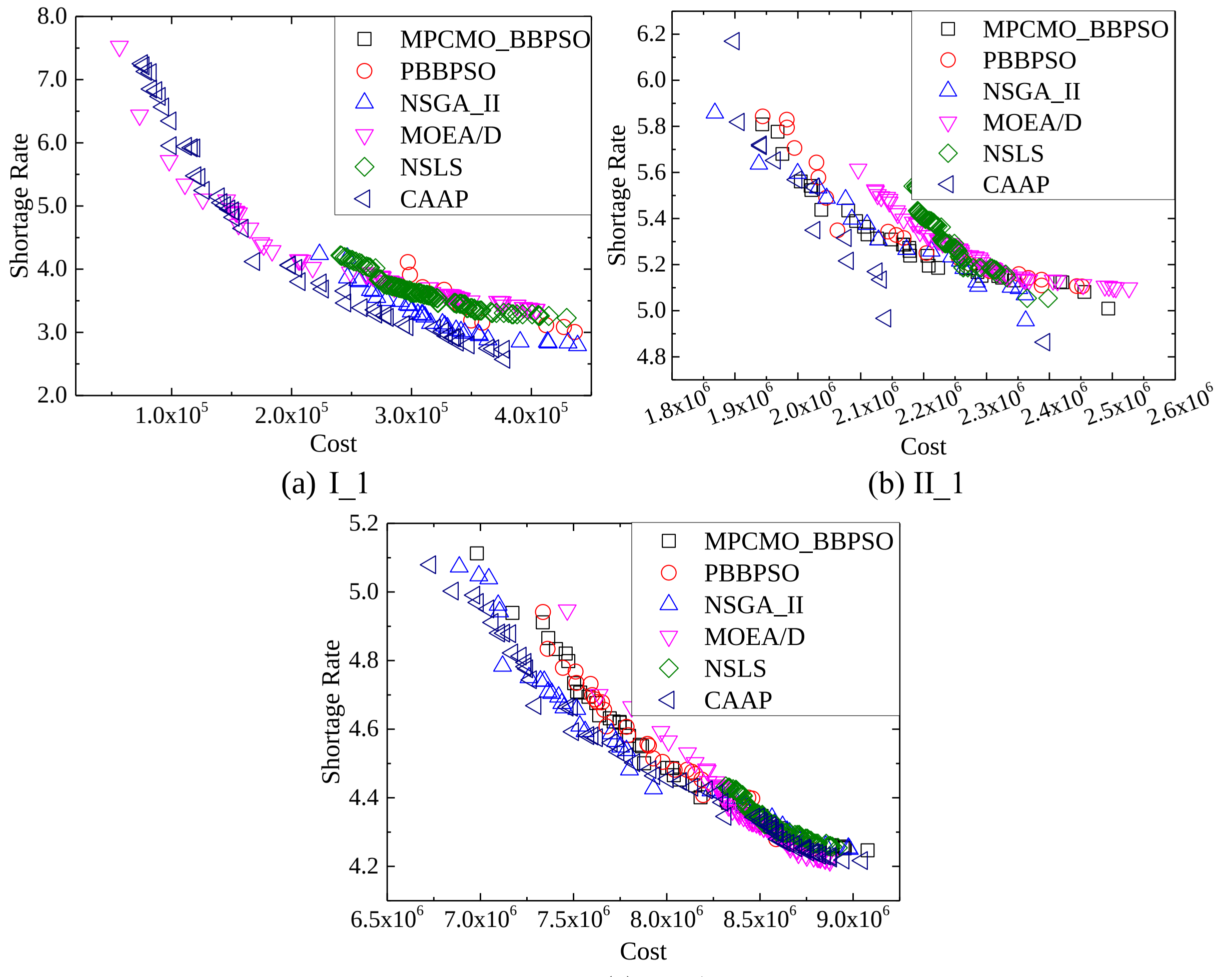
	MPCMO_BBPSO		PBBPSO		NSGA_II	
Instance	C(CAAP, –)	C(–, CAAP)	C(CAAP, –)	C(–, CAAP)	C(CAAP, –)	C(–, CAAP)
I_1	0.18	**0.30**	**1.00**	0.00	**1.00**	0.00
I_2	**0.40**	0.16	**1.00**	0.00	**0.51**	0.09
I_3	0.26	**0.44**	**1.00**	0.00	**1.00**	0.00
I_4	0.27	**0.38**	**1.00**	0.00	0.00	**0.58**
I_5	**0.23**	0.22	**1.00**	0.00	**1.00**	0.00
Avg.	0.27	**0.30**	**1.00**	0.00	**0.70**	0.13
II_1	**0.91**	0.07	**0.95**	0.00	**0.85**	0.29
II_2	**0.95**	0.04	**1.00**	0.00	**0.88**	0.12
II_3	**1.00**	0.00	**1.00**	0.00	**0.91**	0.03
II_4	**0.93**	0.07	**1.00**	0.00	**0.54**	0.20
II_5	**1.00**	0.00	**1.00**	0.00	**0.88**	0.04
Avg.	**0.96**	0.04	**0.99**	0.00	**0.81**	0.14
III_1	**0.92**	0.04	**0.88**	0.12	**0.82**	0.21
III_2	**0.78**	0.12	**0.84**	0.22	**0.70**	0.12
III_3	**0.89**	0.07	**0.78**	0.17	**0.81**	0.12
III_4	**0.79**	0.09	**0.60**	0.32	**0.66**	0.26
III_5	**0.92**	0.00	**0.97**	0.00	**0.58**	0.26
Avg.	**0.86**	0.06	**0.82**	0.17	**0.71**	0.19

Table 4. The C(A, B) values of MOEA/D and NSLS with CAAP on different instances (%)

	MOEA/D		NSLS	
Instance	C(CAAP, –)	C(–, CAAP)	C(CAAP, –)	C(–, CAAP)
I_1	**0.93**	0.34	**1.00**	0.00
I_2	**0.68**	0.66	**1.00**	0.00
I_3	**1.00**	0.00	**1.00**	0.00
I_4	**0.34**	0.33	**1.00**	0.00
I_5	**0.64**	0.33	**1.00**	0.00
Avg.	**0.72**	0.33	**1.00**	0.00
II_1	**1.00**	0.00	**1.00**	0.00
II_2	**1.00**	0.00	**1.00**	0.00
II_3	**1.00**	0.00	**1.00**	0.00
II_4	**1.00**	0.00	**1.00**	0.00
II_5	**1.00**	0.00	**1.00**	0.00
Avg.	**1.00**	0.00	**1.00**	0.00
III_1	**0.46**	0.40	**0.96**	0.01
III_2	**0.59**	0.22	**0.39**	0.20
III_3	**1.00**	0.00	**1.00**	0.00
III_4	**0.52**	0.17	**0.69**	0.15
III_5	**0.90**	0.05	**1.00**	0.00
Avg.	**0.69**	0.17	**0.81**	0.07

For further illustration, Figure 3 depicts the Pareto fronts of all algorithms on three instances from different problem scales. For the minimization problem solved in this paper, if the Pareto front of an algorithm is the closest to the coordinate axes, it means that the algorithm can get the best results among these algorithms. Besides, if the Pareto front of an algorithm is longest, it represents that the algorithm can obtain most non-dominated solutions that spread widely in the solution space. It can be observed from Figure 3 that the Pareto fronts of CAAP on the three instances are almost all the closest and longest among all algorithms, which demonstrates that CAAP can obtain the best results on different instances for the CLMSCN problem.

From the above observations, CAAP can outperform the compared algorithms on different instances. The reason may be that CAAP uses two populations to cooperatively solve the complicated constrained multi-objective problems. One population is to solve the original problem with complex constraints, and the linear repair operator with the self-adaptive repair probability is used to improve the feasibility of solutions. The other population is to solve the simplified problem without constraints, which helps to explore more solution space quickly. The cooperative mechanism of the two populations is effective to solve the complex constrained problem.



(a) I_1 (b) II_1 (c) III_1

Figure 3. Pareto Fronts of all algorithms on instances.

4.3. Effectiveness of the linear repair operator

Table 5. The repair probability of all algorithms on different instances (%).

Instance	CAAP	MPCMO_BBPSO	PBBPSO	NSGA_II	MOEA/D	NSLS
I	21.58	10.80	24.44	17.17	0.79	93.40
II	60.52	83.71	99.79	13.53	100.00	100.00
III	78.47	95.59	99.91	49.59	100.00	100.00
Avg.	53.52	63.37	74.71	26.77	66.93	97.80

Table 6. The C (A, B) of CAAP and CAAP_noRep on different instances (%).

Instance	C(CAAP, CAAP_noRep)	C(CAAP_noRep, CAAP)
I	**0.49**	0.42
II	**1.00**	0.00
III	**1.00**	0.00
Avg.	**0.83**	0.14

To validate the effectiveness of the linear repair operator, the repair probability of all algorithms on different instances are recorded in Table 5. It can be observed that the repair probability of all algorithms is all much larger than zero, and with the problem scale increasing, the repair probability increases. The results illustrate that the linear repair operator can help algorithms improve the feasibility of solutions. Besides, the C(A, B) of CAAP and CAAP_noRep (CAAP without the linear repair operator) on different instances is shown in Table 6. It should be noted that CAAP_noRep uses the superiority of feasible solutions [29] as the constraint handling method. CAAP can get much better results than CAAP_noRep, especially on middle and large instances. It can also be concluded that the linear repair operator is effective to improve the solution quality of algorithms for solving the CLMSCN problem.

5. Conclusions

This paper designs the CLMSCN model with complex constraints and multiple objectives which both considers the minimization of the total operation cost and the customer satisfaction as the optimization objectives. Besides, to solve the complex problem effectively, the CAAP algorithm is proposed in this paper, which applies two populations to solve the problem. The first population is to solve the original CLMSCN problem with complex constraints, and the linear repair operator will be used to improve the feasibility of solutions. The second population is to solve the simplified problem without constraints, which helps to quickly expend the search space of solutions. The experimental results show that the cooperative mechanism of CAAP helps to obtain much better solutions than the compared algorithms on the CLMSCN problem with complex constraints and multiple objectives, especially on the large-scale instances. In the future work, we will design more relatively general constraint-handling methods for the complex problems, and apply the cooperative mechanism of CAAP to solve other supply chain network optimization with complex constraints and multiple objectives, such as the minimization of the fuel consumption and the emission of greenhouse gases.

Acknowledgments

The work described in this paper was substantially supported in part by the National Key R&D Program of China under Grant 2017YFC1601000 and 2017YFC1601800, the National Natural Science Foundation of China under Grant No. 62106088, No. 62172192, and No. 62073155, in part by the High level personnel project of Jiangsu Province (JSSCBS20210852), and in part by the Fundamental Research Funds for the Central Universities, Jiangnan University and Jilin University.

Conflict of interest

The authors declare there is no conflicts of interest.

References

1. M. Aranguren, K. K. Castillo-Villar, M. Aboytes-Ojeda, A two-stage stochastic model for co-firing biomass supply chain networks, *J. Clean. Prod.*, **319** (2021), 128582. doi: 10.1016/j.jclepro.2021.128582.
2. C. Dong, C. Chen, X. Shi, C. T. Ng, Operations strategy for supply chain finance with asset-backed securitization: Centralization and blockchain adoption, *Int. J. Prod. Econ.*, **241** (2021), 108261. doi: 10.1016/j.ijpe.2021.108261.
3. D. Ramón-Lumbierres, F. J. H. Cervera, J. Minguella-Canela, A. Muguruza-Blanco, Optimal postponement in supply chain network design under uncertainty: an application for additive manufacturing, *Int. J. Prod. Res.*, **59** (2020), 5198–5215. doi: 10.1080/00207543.2020.1775908.
4. X. Zhang, Z. H. Zhan, J. Zhang, Multi-objective direction driven local search for constrained supply chain configuration problem, in *Proceedings of ACM Genetic and Evolutionary Computation Conference (GECCO)*, Cancun, (2020), 299–300. doi: 10.1145/3377929.3389929.
5. X. Zhang, Z. H. Zhan, J. Zhang, A fast efficient local search-based algorithm for multi-objective supply chain configuration problem, *IEEE Access*, **8** (2020), 62924–62931. doi: 10.1109/ACCESS.2020.2983473
6. H. Shirazi, R. Kia, P. Ghasemi, A stochastic bi-objective simulation-optimization model for plasma supply chain in case of COVID-19 outbreak, *Appl. Soft. comput.*, **112** (2021), 107725. doi: 10.1016/j.asoc.2021.107725.
7. X. Xu, M. D. Rodgers, W. Guo, Hybrid simulation models for spare parts supply chain considering 3D printing capabilities, *J. Manuf. Syst.*, **59** (2021), 272–282. doi: 10.1016/j.jmsy.2021.02.018.
8. Y. Zhang, S. A. R. Khan, Green supply chain network optimization under random and fuzzy environment, *Int. J. Fuzzy Syst.*, **2021** (2021). doi: 10.1007/s40815-020-00979-7.
9. X. Ma, X. Li, Q. Zhang, K. Tang, Z. Liang, W. Xie, et al., A survey on cooperative co-evolutionary algorithms, *IEEE Trans. Evolut. Comput.*, **23** (2019), 421–441. doi: 10.1109/TEVC.2018.2868770.
10. R. Tanabe, H. Ishibuchi, A review of evolutionary multimodal multi-objective optimization, *IEEE Trans. Evolut. Comput.*, **24** (2020), 193–200. doi: 10.1109/TEVC.2019.2909744.
11. P. Xu, W. Luo, X. Lin, J. Zhang, Y. Qiao, Evolutionary continuous constrained optimization using random direction repair, *Inf. Sci.*, **566** (2021), 80–102. doi: 10.1016/j.ins.2021.02.055.
12. P. Xu, W. Luo, X. Lin, J. Zhang, Y. Qiao, X. Wang, Constraint-objective cooperative coevolution for large-scale constrained optimization, *ACM Trans. Evol. Learn. Optim.*, **1** (2021), 1–26. doi: 10.1145/3469036.
13. Q. Gao, H. Xu, A. Li, The analysis of commodity demand predication in supply chain network based on particle swarm optimization algorithm, *J. Comput. Appl. Math.*, **400** (2022), 113760. doi: 10.1016/j.cam.2021.113760.
14. F. Goodarzian, S. F. Wamba, K. Mathiyazhagan, A. Taghipour, A new bi-objective green medicine supply chain network design under fuzzy environment: Hybrid metaheuristic algorithms, *Comput. Indust. Eng.*, **160** (2021), 107535. doi: 10.1016/j.cie.2021.107535.
15. F. Goodarzian, V. Kumar, A. Abraham, Hybrid meta-heuristic algorithms for a supply chain network considering different carbon emission regulations using big data characteristics, *Soft Comput.*, **25** (2021), 7527–7557. doi: 10.1007/s00500-021-05711-7.

16. Z. Dai, Multi-material and multi-cycle cost optimization of supply chain network and hybrid genetic algorithm, *Appl. Res. Comput.*, **31** (2014), 2620–2624.
17. J. Sun, J. Lin, Study of supply chain optimization scheduling in mass customization based on ant colony algorithm, *J Comput. Appl.*, **11** (2006), 2631–2638.
18. X. Zhang, K. J. Du, Z. H. Zhan, S. Kwong, T. L. Gu, J. Zhang, Cooperative coevolutionary bare-bones particle swarm optimization with function independent decomposition for large-scale supply chain network design with uncertainties, *IEEE Trans. Cybern.*, **50** (2020), 4454–4468. doi: 10.1109/TCYB.2019.2937565.
19. M. Z. Akkad, T. Bányai, Multi-objective approach for optimization of city logistics considering energy efficiency, *Sustainability*, **12** (2020), 7366. doi: 10.3390/su12187366.
20. N. I. Saragih, S. N. Bahagia, Suprayogi, I. Syabri, A heuristic method for location-inventory-routing problem in a three-echelon supply chain system, *Comput. Ind. Eng.*, **127** (2019), 875–886. doi: 10.1016/j.cie.2018.11.026.
21. Y. Tian, T. Zhang, J. Xiao, X. Zhang, Y. Jin, A coevolutionary framework for constrained multi-objective optimization problems, *IEEE Trans. Evolut. Comput.*, **25** (2021), 102–116. doi: 10.1109/TEVC.2020.3004012.
22. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multi-objective genetic algorithm: NSGA-II, *IEEE Trans. Evolut. Comput.*, **6** (2002), 182–197. doi: 10.1109/4235.996017.
23. K. Deb, K. Sindhya, T. Okabe, Self-adaptive simulated binary crossover for real-parameter optimization, in *Proceedings of ACM Genetic and Evolutionary Computation Conference (GECCO)*, (2007), 1187–1194. doi: 10.1145/1276958.1277190.
24. K. Liagkouras, K. Metaxiotis, An elitist polynomial mutation operator for improved performance of MOEAs in computer networks, in *International Conference on Computer Communication and Networks (ICCCN)*, (2013), 1–5. doi: 10.1109/ICCCN.2013.6614105.
25. B. Chen, W. Zeng, Y. Lin, D. Zhang, A new local search-based multi-objective optimization algorithm, *IEEE Trans. Evolut. Comput.*, **19** (2015), 50–73. doi: 10.1109/TEVC.2014.2301794.
26. Q. Zhang, H. Li, MOEA/D: A multi-objective evolutionary algorithm based on decomposition, *IEEE Trans. Evolut. Comput.*, **11** (2007), 712–731. doi: 10.1109/TEVC.2007.892759.
27. Z. Zhan, J. Li, J. Cao, J. Zhang, H. S. Chung, Y. Shi, Multiple populations for multiple objectives: a coevolutionary technique for solving multi-objective optimization problems, *IEEE Trans. Evolut. Comput.*, **43** (2013), 445–463. doi: 10.1109/TSMCB.2012.2209115.
28. E. Zitzler, L. Thiele, Multi-objective evolutionary algorithms: a comparative case study and the strength Pareto approach, *IEEE Trans. Evolut. Comput.*, **3** (1999), 257–271. doi: 10.1109/4235.797969.
29. K. Deb, An efficient constraint handling method for genetic algorithms, *Comput. Methods Appl. Mech. Eng.*, **186** (2000), 311–338. doi: 10.1016/S0045-7825(99)00389-8.