Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 179102, 18 pages
http://dx.doi.org/10.1155/2015/179102

Research Article

A Hybrid Approach to Solve a Model of Closed-Loop Supply Chain

Nafiseh Tokhmehchi,[1] Ahmad Makui,[2] and Soheil Sadi-Nezhad[1]

[1]*Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran*
[2]*Department of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran*

Correspondence should be addressed to Nafiseh Tokhmehchi; nafiset@ymail.com

Received 18 October 2014; Revised 11 January 2015; Accepted 2 February 2015

Academic Editor: Lionel Amodeo

This paper investigates a closed-loop supply chain network, including plants, demand centers, as well as collection centers, and disposal centers. In forward flow, the products are directly sent to demand centers, after being produced by plants, but in the reverse flow, reused products are returned to collection centers and, after investigating, are partly sent to disposal centers and the other part is resent to plants for remanufacturing. The proposed mathematical model is based on mixed-integer programming and helps minimizing the total cost. Total costs include the expenditure of establishing new centers, producing new products, cargo transport in the network, and disposal. The model aims to answer these two questions. (1) What number and in which places the plants, collection centers, and disposal centers will be constructed. (2) What amount of products will be flowing in each segment of the chain, in order to minimize the total cost. Four types of tuned metaheuristic algorithms were used, which are hybrid forms of genetic and firefly algorithms. Finally an adequate number of instances are generated to analyse the behavior of proposed algorithms. Computational results reveal that iterative sequentialization hybrid provides better solution compared with the other approaches in large size.

1. Introduction

Nowadays all countries are focusing on sustainable development, which is only achieved by saving and optimal using of the limited and nonrenewable resources in that country. Governments implement a vast range of activities to achieve such purpose, including enacting rules and regulations to encourage using environment-friendly raw materials in industrial complexes, reduce the usage of fossil and oil energy, and recycle the products in both public and private sectors. Sustainable supply chain management includes superseding, reusing, recycling, and defusing materials, as well as improving public wellbeing, and reducing chain costs.

When companies use a forward-reverse system (closed-loop), many environmental draw backs will be prevented, for instance, energy waste, high consumption, and transportation costs. Supply chain networks of sustainable development are considered suitable means for designing closed-loop production networks. This method fits the procedures in an economic cycle, which helps economic and environmental development significantly.

Forward activities include improving the design, engineering, production and supply, marketing, sales, and after-sale services of a new product. Reverse activities consist of reverse logistics, usage and disposal places, testing, classifying, refreshing, recycling, marketing, and reselling. Economic approaches, governmental strategies, and customer pressure are 3 main aspects of reverse logistics [1].

There is a key difference between reverse chain and waste management. Waste management means an efficient collection procedure, which does not reuse the products, but in reverse-network, the products are resurrected.

A closed loop network contrives the forward and reverse activities in a unique system. In other words, the integration

of forward and reverse supply chains forms closed-loop supply chain. This integration can improve economic and environmental functions in most of the industries.

In recent decades, the pressure of world economy has forced companies to introduce new core competencies. Not only are the companies encouraged by environmental rules to recycle the used products, but also they have learned the economic benefits of this approach; hence, developing reverse logistics and closed-loop networks have become a key focal point for researchers and companies.

The idea of this research arises from a tire manufacturer that aspires to supply the demands of their clients and reduce their costs. Knowing that they felt the need to design a closed-loop supply chain, which includes locating and constructing new plants for manufacturing and remanufacturing with two different technologies, collection centers, and disposal centers, therefore, this research tries to develop a mathematical model for the industry in question. Suitable solutions are sought for the same purpose.

The remainder of the paper is organized as follows. In Section 2, we offer a relevant literature on closed-loop supply chain network. Section 3 is dedicated to define and mathematical formulation of the problem. Comprehensive explanations of proposed solutions are discussed in Section 4. Section 5 is dedicated to computational results including data gathering, tuning the parameters and comparisons between solution approaches. Finally in Section 6, conclusions are provided and some future suggestions are given.

2. Literature Review

There have been lots of studies about supply chain network design. In order to focus more precisely, we divided our review in three parts including forward networks, reverse networks, and closed-loop networks. Additionally at the end of this section, we will present 2 tables to provide some comparison between different studies.

Most of studies in supply chain design related to forward network such as [2–10] Sabri and Beamon [2] developed a mixed integer linear programming (MILP) model with the aim to minimize cost and maximize service level and flexibility. The mentioned model can be used for simultaneous strategic and operational planning. Syarif et al. [3] considered a multistage logistic problem and formulated it with MILP model. In order to be solved, they utilized spanning tree-based genetic algorithm. Jayaraman and Ross [4] focused on planning and execution stages of a PLOT (production, logistics, outbound, and transportation) system. They also presented an MILP model and solved it with a simulated annealing methodology. Miranda and Garrido [5] presented a simultaneous approach to incorporate inventory control decisions in a distribution network design system. In this way, they developed a mixed integer nonlinear programming (MINLP) model and solved it with the Lagrangian relaxation and the subgradient method. Altiparmak et al. [6] developed a MINLP model with some conflicting objectives consisting of cost, service level, and equity of the capacity utilization ratio. For a solution, they used a genetic algorithm approach. Amiri [7] developed a MILP model and provided a heuristic solution procedure based on Lagrangian relaxation for a distribution network. Tsiakis and Papageorgiou [8] determined the optimal configuration of a production and distribution network with respect to operational and financial limitations. In this way they proposed a MILP model with the aim of cost minimization and solved it with an exact method. Pishvaee et al. [9] considered a socially responsible supply chain network design. In addition to dealing with uncertain conditions in a network a robust possibilistic programming was developed. Pishvaee et al. [10] developed a credibility-based fuzzy mathematical programming model with the purpose to minimize total costs and the environmental impacts for a green logistics design under uncertainty.

In this part we focus on reverse logistic networks by review papers [11–17]. Govindan et al. [11] classified 382 papers by problem classes, solution, and modeling approaches. Mentioned paper is generally divided into 4 parts including reverse logistics, closed-loop, sustainable and green but more focused on the first two parts. Jayaraman et al. [12] considered a reverse distribution network and developed an MILP model for it. Apart from that, they introduced a heuristic solution methodology. Listeş and Dekker [13] developed a stochastic approach for product recovery network design and applied it to a real case study on recycling sand from demolition waste in The Netherlands. Aras et al. [14] developed a mixed-integer nonlinear facility location-allocation model under pick-up strategy with capacitated vehicles. The solution approach was based on a Tabu search method. Schweiger and Sahamie [15] considered a combined continuous and discrete facility location problem containing home-recycling techniques, as well as selling and disposing technologies. In addition a hybrid Tabu search algorithm is used to solve it. Eskandarpour et al. [16] proposed a biobjective MILP model for postsales reverse logistics network operated by a 3PL. They solved their model with a novel multistart variable neighborhood search with three new encoding–decoding mechanisms. Roghanian and Pazhoheshfar [17] considered multiproduct, multistage reverse logistics network under stochastic environment and proposed a probabilistic mix integer linear model for it. They also presented a priority based genetic algorithm to solve their mathematical model.

Papers [18–28] belong to closed loop supply chain. Fleischmann et al. [18] presented a MILP facility location model and used it to analyze the impact of product return flows on logistics networks. Ko and Evans [19] developed a MINLP model for a dynamic integrated forward/reverse logistics network and solved it by a genetic algorithm. Pishvaee et al. [20] tried to minimize the total costs and maximize responsibility. They applied mixed-integer programming with two objectives, to find a set of nondominated solutions. Pishvaee and Torabi [21] used stochastic approach in closed-loop supply chains with uncertainty conditions. They aimed to prevent suboptimizing the proposed model. This study combined the design decisions in both forward and reverse networks and also used fuzzy approach to solve the model. Amin and Zhang [22] considered a supply chain network with the aim to maximize profit, minimize the rate of defects, and maximize importance of suppliers. Their model included 2 phases. In

the first phase, the selection factor of suppliers is determined in a reverse flow. The second phase introduced a mixed-integer programming model. This was the first attempt to simultaneously select the suppliers and allocate the orders in a closed-loop network. Pishvaee and Razmi [23] applied a multiobjective fuzzy programming model to design an environment-friendly supply chain. Mentioned model had several parameters of uncertainty and intended to minimize the costs and environmental effects.

2013 and 2014 are remarkable years with regard to the researches on closed-loop systems. Numerous articles, such as [24–28], have been published in these years. Amin and Zhang [24] used a mixed-integer linear model for minimizing the total costs and at the same time considering environmental factors. They applied Weighted Sum Method and ε-constraint Method to develop the model. They also tried to probe the effect of demand uncertainty and return uncertainty in the network. The mentioned article used stochastic programming technique for this purpose. Ramezani et al. [25] devised a multiobjective stochastic model to design a forward-reverse logistic chain in uncertainty environment. This model incorporates 3 levels in forward flow and 2 levels in reverse flow and attempted to maximize the interest, responsiveness to clients, and quality in the logistic network. Soleimani et al. [26] developed a mix integer linear model for a multiechelon, multiproduct, and multiperiod closed loop network. In their model plants, warehouses and distributors can stratify customer demand. For a solution they utilized CPLEX and genetic algorithm. Faccio et al. [27] introduced a linear programming model for a closed-loop problem, with the aim to minimize the total costs of this chain. Parametric study and robust analysis of this model have been contrasted with the classic forward-flow approach and social responsibility. Devika et al. [28] developed a mixed integer programming model in closed-loop supply chain based on triple bottom line approach. They also presented three novel hybrid metaheuristic methods based on adapted imperialist competitive algorithms and variable neighborhood search to solve it.

These mentioned researches indicate that metaheuristic algorithms are becoming the most efficient approaches to solve complex problems in supply chain network design.

Table 1 is dedicated to coding system and Table 2 shows the mentioned models in different studies.

3. Problem Definition

The problem in this paper has been taken from a closed-loop supply chain, including plants, collection centers, demand centers, and disposal centers. Plants are able to manufacture new products and remanufacture the returned products. In forward flow, the products are dispatched to demand centers, after being produced by factories. In reverse flow, the returned products are returned to collection centers. Collection centers are in charge of the following:

(i) collecting the returned products from demand centers,

(ii) determining the quality of returned products, by observing and categorizing into two groups: renewable nonrenewable,

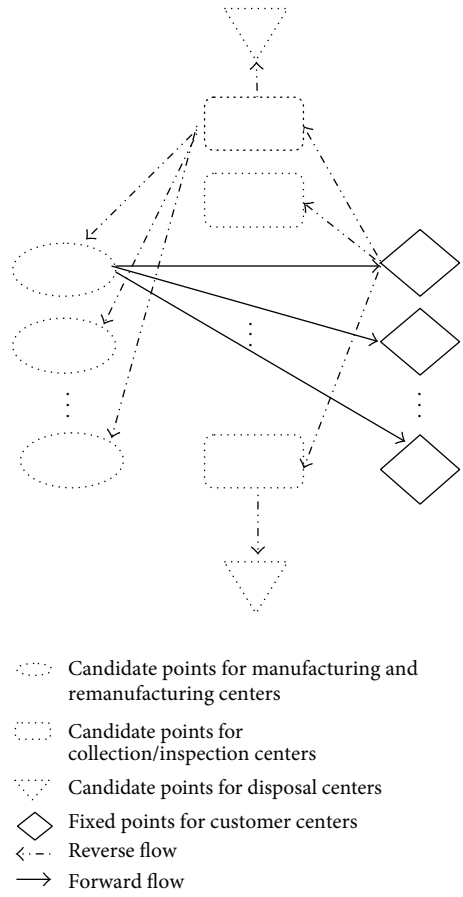


Figure 1: Proposed closed-loop supply chain.

(iii) sending renewable returned products to plants (for remanufacturing),

(iv) sending nonrenewable returned products to disposal centers.

The model tries to answer this question. In which places and what number of plants, collection centers, and disposal centers will be established, and what amount of products must go there to minimize the total cost. Figure 1 provides a pictorial representation of the closed-loop supply chain. In addition to defining the problem precisely we consider the following assumption and limitations.

(i) All the returned products from demand centers are kept in collection centers. The demand centers have fixed locations.

(ii) The potential locations, capacity of plants, collection centers, and disposal centers are already determined.

(iii) The needs of every demand center can be met with the help of several plants. When founding a new plant, diverse manufacturing and remanufacturing technologies are available. The needs of demand centers will be met completely.

(iv) There is no cargo-flow between the same level facilities.

Table 1: Coding of literature review.

Type of network		Modeling	
Forward logistic	F	Continuous	
Reverse logistic	R	Continuous approximation	CA
Closed loop	C	Discrete	
Layers of network		Mixed integer linear programming	MILP
Forward logistic stage		Mixed integer nonlinear programming	MINLP
Supplier centers	SC	Stochastic mixed integer programming	SMIP
Manufacturing centers	MC	Fuzzy mixed integer programming	FMIP
Distribution centers	DC	Probabilistic mixed integer linear programming	PMILP
Reverse logistic stage		Combined continuous and discrete	
Redistribution centers	RDC	Combined continuous and mixed integer programming	CMIP
Recycling centers	RyC	Objective of model	
Collection/inspection centers	CIC	Cost	Co
Disposal centers	DsC	Profit	Pr
Recovery centers	RoC	Quality	Qu
Disassembly center	DaC	Environmental impact	EI
Processing center	PrC	Social impact	SI
Returning center	RC	Flexibility	Fle
Remanufacturing centers	RMC	Service level	SL
Feature of model		Tardiness	Td
Product		Defect rate	DR
Single product	SP	Importance of external suppliers	IES
Multiproduct	MP	Capacity utilization	CU
Period		Solution methodology	
Multiperiod	MPr	Exact (small sizes)	Ex
Single period	SPr	Lagrangian relaxation-based	LR
Demand		Metaheuristic	MH
Deterministic	D	Interactive fuzzy solution approach	IF
Stochastic	S	ε-constraint method	EC
Fuzzy	Fu	Other heuristics	H
Random	Ra		
Facility capacity			
Capacitated	Ca		
Uncapacitated	UCa		

(v) A certain percent of products taken from demand centers will be returned by collection centers later on.

(vi) Every collection center is allowed to send nonrenewable products to diverse disposal centers.

3.1. Mathematical Model. According to what is mentioned above, the mathematical model for this problem is mixed-integer programming. The following indices, variables, and parameters are used to model the problem in hand.

3.1.1. Indices and Sets

i: index for potential location of plants $i = 1, \ldots, I$,

l: index for potential location of collection-inspection centers $l = 1, \ldots, L$,

n: index for technology types in plants $n = 1, \ldots, N$,

s: index for potential location of disposal centers $s = 1, \ldots, S$,

j: index for products $j = 1, \ldots, J$,

k: index for fixed location of customers (demand centers) $k = 1, \ldots, K$.

3.1.2. Variables

u_{ikjn}: quantity of product j produced with technology n at plant i for demand center k,

v_{klj}: quantity of returned product j shipped from demand center k to collection-inspection center l,

w_{lij}: quantity of returned product j shipped from collection-inspection center l to plant i,

Table 2: Review of previously published literature.

Reference	Type of network	Layers of network	Feature of model	Modeling	Solution methodology	Objective of model
[2]	F	SC, MC[d], DC[d]	MP, SPr, D, Ca	MILP	Ex	Co, Fle, SL
[3]	F	SC, MC[d], DC[d]	SP, SPr, D, Ca	MILP	MH	Co
[4]	F	MC, DC[d]	MP, SPr, D, Ca	MILP	MH	Co
[5]	F	MC, DC[d]	SP, MPr, S, Ca	MINLP	LR	Co
[6]	F	SC, MC[d], DC[d]	SP, SPr, D, Ca	MINLP	MH	Co, SL, CU
[7]	F	MC[d], DC[d]	SP, SPr, D, Ca	MILP	LR	Co
[8]	F	MC[d], DC[d]	MP, SPr, D, Ca	MILP	Ex	Co
[9]	F	MC[d], DC[d]	SP, SPr, Fu, Ca	FMIP	Ex	Co, SI
[10]	F	MC[d], DC[d]	SP, SPr, Fu, Ca	FMIP	IF	Co, EI
[12]	R	CIC[d], RoC[d]	SP, SPr, D, Ca	MILP	H	Co
[13]	R	CIC[d], RyC[d]	MP, SPr, S, Ca	SMIP	Ex	Co
[14]	R	CIC[d]	MP, SPr, D, UCa	MINLP	MH	Co
[15]	R	SC, MC, CIC[d]	MP, MPr, D, Ca	CMIP	MH	Co
[16]	R	MC, DsC, CIC, RoC[d]	MP, SPr, D, Ca	MILP	MH	Co, Td
[17]	R	MC, DaC[d], PrC[d], RC, RyC	MP, SPr, Ra, Ca	PMILP	MH	Co
[18]	C	CIC[d], RoC[d], MC[d], DC[d]	SP, SPr, D, UCa	MILP	Ex	Co
[19]	C	RoC, MC, DC[d]	MP, MPr, D, Ca	MINLP	MH	Co
[20]	C	MC[d], RoC[d], CIC[d], DsC[d], DC[d]	SP, SPr, D, Ca	MILP	MH	Co, SL
[21]	C	MC, DC[d], CIC[d], RoC[d], RyC[d]	SP, MPr, Fu, Ca	FMLP	IF	Co, Td
[22]	C	SC[d], RMC[d], DaC[d]	MP, SPr, D, Ca	MILP	Ex	Pr, DR, IES
[23]	C	MC[d], CIC[d], RoC, DsC	SP, SPr, Fu, Ca	FMIP	IF	Co, EI
[24]	C	MC[d]/RMC[d], CIC[d], DsC	MP, SPr, S, Ca	MILP	EC	Co, EI
[25]	C	SC, MC[d], DC[d], CIC[d], RMC, DsC[d]	MP, SPr, S, Ca	MILP	EC	Pr, SL, Qu
[26]	C	SC[d], MC[d], DsC[d], DC[d], RDC[d], CIC[d]	MP, MPr, D, Ca	MILP	MH	Co
[27]	C	MC[d], DsC, DC[d]	MP, SPr, D, Ca	MILP	Ex	Co
[28]	C	SC, MC[d], DC[d], CIC[d], RoC[d], RMC[d], RyC[d], DsC[d]	SP, SPr, D, Ca	MILP	MH	Co, EI, SI
This paper	C	MC[d]/RMC[d], CIC[d], DsC[d]	MP, SPr, D, Ca	MILP	MH	Co

[d]Facilities to be located in the network.

x_{ljs}: quantity of returned product j shipped from collection-inspection center l to disposal center s,

y_{in}: 1 if the plant center i is to be established with technology n, 0 otherwise,

z_l: 1 if the collection-inspection center l is to be established, 0 otherwise,

g_s: 1 if the disposal center s is to be established, 0 otherwise.

3.1.3. Parameters

m_{ijn}: unit cost of manufacturing product j in plant i with technology n,

tk_j: unit cost of transportation product j from plants to customer centers (Per km),

ta_j: unit cost of transportation product j from customer centers to collection-inspection centers (Per km),

tg_j: unit cost of transportation product j from collection-inspection centers to plants (Per km),

td_j: unit cost of transportation product j from collection-inspection centers to disposal centers (Per km),

f_{in}: fixed cost of establishing plant i with technology n,

f_l': fixed cost of establishing collection-inspection center l,

f_s'': fixed cost of establishing disposal center s,

a_j: unit saving cost of product j,

b_{js}: unit disposal cost of product j at center s,

e_{ik}: distance between plant i and customer center k,

h_{il}: distance between collection-inspection center l to plant i,

o_{ls}: distance between collection-inspection center l to disposal center s,

d_{lk}: distance between collection-inspection center l to customer center k,

p_{kj}: demand of customer center k for product j,

q_{kj}: amount of returned product j from customer center k,

r_j: minimum disposal rate for product j,

c_{ijn}: maximum manufacturing capacity of plant i for product j with technology n,

c'_{lj}: maximum capacity of collection-inspection center l for product j,

c''_{sj}: maximum capacity of disposal center s for product j.

3.1.4. Proposed Mathematical Formulation. The mathematical model of this problem becomes

$$\begin{aligned}\text{Min:}\quad TC &= \sum_i \sum_n f_{in} y_{in} + \sum_l f'_l z_l + \sum_s f''_s g_s \\ &+ \sum_i \sum_k \sum_j \sum_n \left(m_{ijn} + tk_j e_{ik}\right) u_{ikjn} \\ &+ \sum_k \sum_l \sum_j ta_j d_{lk} v_{klj} \\ &+ \sum_l \sum_i \sum_j \left(-a_j + tg_j h_{il}\right) w_{lij} \\ &+ \sum_s \sum_l \sum_j \left(b_{js} + td_j o_{ls}\right) x_{ljs}\end{aligned} \tag{1}$$

$$\text{s.t.}\quad \sum_i \sum_n u_{ikjn} \geq p_{kj}, \quad \forall k, j, \tag{2}$$

$$\sum_l \sum_j w_{lij} + \sum_k \sum_j \sum_n u_{ikjn} \leq \sum_n \sum_j c_{ijn} y_{in}, \quad \forall i, \tag{3}$$

$$\sum_l v_{klj} \leq \sum_i \sum_n u_{ikjn}, \quad \forall k, j, \tag{4}$$

$$r_j \sum_k v_{klj} \leq \sum_s x_{ljs}, \quad \forall l, j, \tag{5}$$

$$\sum_k \sum_j v_{klj} \leq z_l \sum_j c'_{lj}, \quad \forall l, \tag{6}$$

$$\sum_k v_{klj} = \sum_i w_{lij} + \sum_s x_{ljs}, \quad \forall l, j, \tag{7}$$

$$\sum_l \sum_j x_{ljs} \leq g_s \sum_j c''_{js}, \quad \forall s, \tag{8}$$

$$\sum_l v_{klj} = q_{kj}, \quad \forall k, j, \tag{9}$$

$$\sum_n y_{in} \leq 1, \quad \forall i, \tag{10}$$

$$y_{in}, z_l, g_s \in \{0, 1\}, \quad \forall i, l, n, s \tag{11}$$

$$u_{ikjn}, v_{klj}, w_{lij}, x_{ljs} \geq 0, \quad \forall i, k, l, j, n, s. \tag{12}$$

The objective is minimizing the total costs. There are 7 types of cost. The first 3 are the constant expenses of establishing new plants, new collection centers, and disposal centers. The 4th part includes the transportation and manufacturing costs of producing new products. The 5th expenditure will be recycling and transporting the returned products. The 6th segment is associated with the total costs of recycling and transporting returned products from collection centers to factories. Finally, the 7th part of target function represents the transportation and disposal costs.

Constraint (2) guarantees that the total number of manufactured products for each demand center is equal to or greater than their demand rate. Constraint (3) shows the capacity of plants. Constraint (4) shows that the flow rate in forward method is higher than reverse flow. Constraint (5) represents the disposal amount. Constraint (6) presents the limited capacity of collection centers. Equation (7) indicates that the amount of returned products from customer centers is equal to the amount of returned products to plants and the amount of products sent to disposal centers from every collection center for every product. Constraint (8) illustrates the limited capacity of disposal centers. Equation (9) refers to the returned products. Constraint (10) demonstrates that every plant, if established, enjoys only one type of technology. In other words no combination of technologies will be accepted. The two final constraints in the model focus on the decision variables. Three variables are associated with establishing and not establishing the plants, collection centers, and disposal centers, respectively, in their potential locations. These variables are binary. Other decision variables in the last constraint are defined as nonnegative.

4. Solutions

For the optimization problems with large scale and high complexity, classic solutions will not be reliable. The solutions must search the feasible space in a more sophisticated way. As mentioned in literature review nowadays many researchers

devise metaheuristic algorithms for complicated problems of supply chain.

Before applying metaheuristic we must show that the closed-loop problem is NP hard. In this way we mentioned the fact that the closed-loop problem is a capacitated location-allocation problem and also can be considered as a multiple-choice Knapsack problem, for it is known to be NP-hard [3, 29, 30]. So metaheuristic algorithm can be one of the best candidate solvers for such problems. The problem in question becomes more complicated by increasing the number of facilities and product technologies. This might make the accurate and heuristic solutions inapplicable; hence, metaheuristic algorithms become the focal point. Among the metaheuristic solving techniques, hybridization methods play an important role in optimizing the case.

4.1. Genetic Algorithms. The main idea of genetic algorithms arises from nature. These are effective random searching techniques in mega-scale target areas. They search in different directions to find the optimum answer.

Genetic algorithms usually generate several random solutions to the problem initially. This set of solutions is called "The initial population," and each member of the set is titled "Chromosome." In order to find better chromosomes, the operators of genetic algorithm cross the chromosomes over each other. This cross-over leads to mutation, so that new chromosomes are generated. The best members are separated. This is how the second generation is formed. In other words, the operators of selecting, crossing over, and mutating every generation produce a new generation of chromosomes. The new generation is different from the previous one. The whole process is repeated over and over for the next generations, until the ultimate solution is obtained. The steps of this algorithm are explained below in detail, respectively:

(1) creating and apprising the initial population that is generated randomly,

(2) selecting the parents with one of the selection methods (roulette wheel, tournament, and random),

(3) applying crossover operator on the selected chromosomes as parents,

(4) selecting the chromosomes for mutating,

(5) applying the mutation operator,

(6) creating a new population from the best members of all the 3 main populations: crossed-over, mutated, and initial population,

(7) appraising the new population,

(8) restarting from step (2), if the ending conditions are not met.

4.1.1. The Solution Structure. The first step to solve a problem with metaheuristic algorithms is creating a structure for different solutions. This is called "coding phase." In low-complexity problems with a small number of variables, the simplest method for presenting the solution structure is using a string in problems. Matrix view is sometimes applied as well.

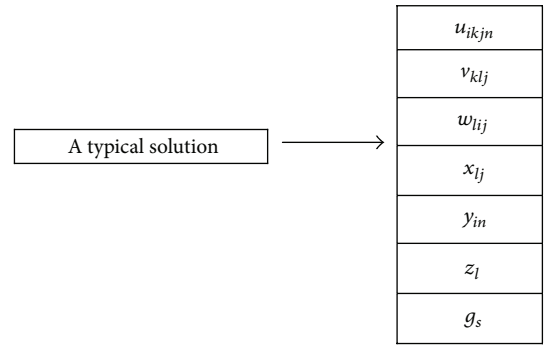


Figure 2: An illustration of chromosomal structure.

With modern calculation tools, such as Matlab Software, stronger and more efficient structures have become available for presenting the solutions. Data structure is one of them. In data structure, every answer is associated with a structure. The structure will also have substructures based on the number of variables in the problem. For the current problem, every answer is initially presented with one structure. There are 7 key variables, so every answer has 7 substructures, which are also divided into substructures according to the problem scale (number of indexes for manufacturers, collection centers, demand centers, disposal centers, etc.). The scheme of one answer is shown in Figure 2.

4.1.2. Selection. For every member, a pair of parents is selected. The aim is to choose the best elements. Even the weakest element has the chance to be selected. This prevents stopping on local solutions. Numerous selection operators have been introduced so far, for example, operators roulette wheel, tournament, and random. This paper has coded all the 3 methods, but the vast range of tests proved that method roulette wheel is the best selection technique for our model.

4.1.3. Crossover. Genetic algorithms show the probability for crossover operator, which is between 0 and 1. This presents the probability of creating offspring. This is the likelihood of chromosomes being crossed over again. The crossover of every pair of chromosomes will create an offspring, which is a new member of the next generation. This operator tries to find suitable potential answers in the next generation.

There are various cross-over methods, such as methods partial-mapped crossover (PMX), order crossover (OX), multipoint crossover, and cycle crossover (CX) [29]. The current study uses multipoint crossover technique, which is a developed version of single-point crossover. Two structures are primarily considered. Then several locations are selected in the structure randomly. The elements of these locations are substituted with each other. A scheme of this crossover technique comes in Figure 3.

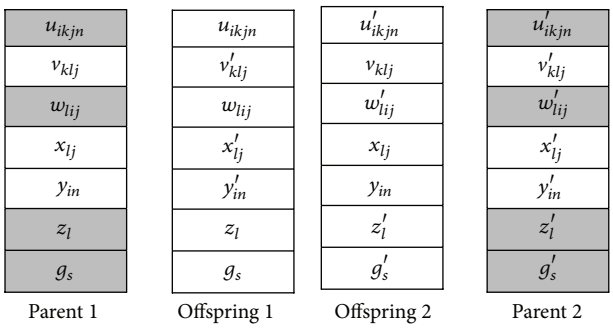


Figure 3: An illustration of crossover operations.

4.1.4. Fitness Function. Fitness function illustrates the qualification of each answer in every step of the algorithm. This function is the objective function in most of optimization problems. It must find the minimum or maximum amount, except for penalty functions. Penalty function was one of the essentials according to the conditions and nature of a problem. After applying genetic algorithms for chromosomes, the new solutions might not always conform to the conditions of the problem. This is why penalty functions are used in the target function. The penalties stick to the objective function, like weights. They do not let improper chromosomes repeat in the next generation or appear in the final solution.

This paper creates the initial solutions with the technique that prevents generating any type of unfeasible answer. During cross-over and mutation steps, if the changes in elements lead to such answers, again a correction method heals the answers towards the feasible space; hence, penalty function will not be necessary; therefore, the fitness function in this study will be exactly the same as target function in (1). The suggested correction method is divided into 3 techniques according to the constraints of this model. The first one is "less than or equal to constraints." Assume constraint $\sum_j X_{ji} \leq R_i$ is applied and, for certain reasons, variable X_{ji} that is generated in chromosome structure has violated feasibility conditions ($\sum_j X_{ji} \succ R_i$). In such case, we find the maximum number in the structure of X_{ji} and suppose it is equal to the minimum number in the structure. This is repeated over and over until we get back to feasibility.

The second case is "greater than or equal to constraints." Assume constraint $\sum_j X_{ji} \geq R_i$ exists and, for certain reasons, the generated X_{ji} created by the constraint is not feasible ($\sum_j X_{ji} \prec R_i$). In such case, we find the minimum number in the structure of X_{ji} and make it equal to the maximum number in the structure. This is repeated until we get back to feasibility.

The 3rd case is "equality constraint." Assume the existence of constraint $\sum_j X_{ji} = R_i$. The generated variable X_{ji} does not fit equality conditions ($\sum_j X_{ji} \neq R_i$). In such case again, we create a matrix of random numbers between 0 and 1 and called it X'_{ji}. The new X_{ji} is obtained from the following formula:

$$X_{ji} = \frac{R_i X'_{ji}}{\sum_j X'_{ji}}. \tag{13}$$

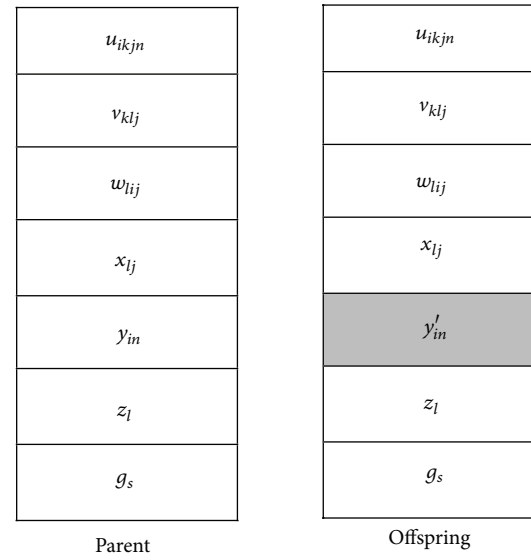


Figure 4: An illustration of mutation operations.

It is noteworthy that all cases 1, 2, and 3 can be generalized to several summations.

4.1.5. Mutation. Genetic algorithms have small and constant change probability, which is between 0 and 1. According to this probability, child chromosomes change randomly and come across mutation. There are various methods of mutating, such as methods inversion, insertion, and uniform.

In the suggested chromosomes' structure, every variable sits in a fixed location; therefore, using mutation operators, that substitute the location of one variable with another, leads to losing the answer structure; hence, methods such as uniform can be proper choices for mutating.

The proposed mutation operator in the current study selects one of the 7 variables randomly and causes some change in it. Figure 4 shows a scheme of mutating method.

4.1.6. Finishing Conditions. It is obvious that the designed algorithm, which searches through the answer space, has to stop somewhere and somehow; otherwise, the search procedure will grow too long and sometimes does not come to any result at all.

Several methods have been introduced and devised so far about the finishing criteria of metaheuristic algorithms, for instance, gaining an acceptable level in the answer and passing through a certain number of generations in the algorithm and a particular number of loops without improvement in answer.

Some researchers use a time limit for ending the algorithm. They restrict the run time or the maximum time without gaining an improved answer. Among all introduced criteria, time-based techniques seem to be unreliable because various computers have different processing speeds; besides, every year new models with higher speed and vaster features come to market.

This paper regards the number of generations as finishing criterion for the algorithm. Every algorithm stops after cycling through a certain number of generations.

4.2. Firefly Algorithm. Firefly algorithm was introduced firstly in 2007 by someone named Yang [31]. This is one of the newest optimization algorithms, whose idea arises from nature, and is based upon swarm intelligence. In FF algorithm, a few artificial fireflies are initially generated randomly, with a structure similar to the chromosomes in genetic algorithm.

After generating and distributing the fireflies, every firefly radiates some light, whose intensity represents the optimality level of its location. In other words, the firefly with higher fitness is more attractive and grabs more attention. The attractiveness of fireflies is based on their brightness and light strength; hence, the closer they are to the light source, the more attractive they are. Additionally, if a firefly is weak but close to the light source, again it can enjoy suitable attractiveness. The light intensity of every firefly is continuously compared with the light strength of other fireflies, and the difference of light level stimulates the firefly to move. This means if the firefly has higher fitness than the contrasted firefly, it moves to the new location, which is an average between the location of higher-fitness firefly and the contrasted one.

If the problem objective is minimization, lighter firefly is attracted towards the dimmer firefly. In this algorithm, even the weakest firefly moves randomly inside the problem domain with the aim of increasing the chance of finding global optimum.

The decrease in light level is caused and determined by light absorption factor and controlled within a certain scale. The information exchange among fireflies is done via light radiation towards each other. Finally, this whole group procedure causes the fireflies to tend towards the more optimal point.

4.2.1. The Procedure of FF. The steps of this algorithm are shown below.

(1) Generate the initial population of fireflies.

(2) Calculate light intensity I for every firefly (we can always consider objective function of a firefly as light intensity of it [31]):

$$I_x \propto \mathrm{TC}_x. \tag{14}$$

(3) For every firefly (as "i") and for every other firefly (as "j"), if the received light intensity of firefly "i" is less than the received light of firefly "j" ($I_i \prec I_j$), we produce new solution according to following formula:

$$x_i' = x_i + \beta_0 e^{-\delta r^2} \left(x_j - x_i\right) + \alpha\varepsilon. \tag{15}$$

X_i is the location of firefly i, X_j is the location of firefly j, α is the mutation coefficient (it is between 0 and 1 [31]). β_0 is attraction coefficient base value (usually considered 1 [31]), ε is a random vector (usually considered sign [rand−1/2]), and δ is a light absorption coefficient (it typically varies from 0.01 to 100 [31]).

In addition r is the Cartesian distance to light source and calculated by the following equation [31]:

$$r = \left\| x_i - x_j \right\|. \tag{16}$$

(4) Evaluate the new fireflies.

(5) Determine the best detected answer.

(6) Loop to step (3), if the ending conditions are not true.

Again, the structure of each firefly is similar to the chromosome in Figure 2. Cost function is similarly calculated as well.

4.3. Hybridization of Algorithms. With the help of hybridizing process, two or more algorithms can be combined to create new methods, which inherit part of the characteristics of each creator algorithm and form a new algorithm with new different characteristics [32]. The goal of algorithms' hybridization is improving their characteristics or creating new characters that are not available in each algorithm alone. That is why in the recent years more and more hybrid algorithms are being born and applied.

Among different hybrids, combining a search method within a genetic algorithm can upgrade the search performance and also it is a chance to hybrid optimization to attain their target [33].

The range of hybridization methods is enormous, for example, sequentialization and parallel methods. In current study, four hybridization methods have been devised, and with the help of Matlab Software, all of them were coded; besides, a summary about the functionality of each of the 4 hybridization methods is explained with the related chart.

4.3.1. Algorithm PHGF. The first suggested algorithm is a simple parallelization technique. In this technique, the calculations of genetic and firefly algorithms are performed in a parallel way with similar cycles, and the final output is chosen from the best answers of both algorithms. Figure 5 shows an illustration of simple parallelization hybridization.

The steps of PHGF are shown below.

(1) Randomly generate an initial population (p_g).

(2) While (not stop condition)

  (a) For $h = 1$: H_g (H_g is the number of individuals in the population)

    (a.1) Recombine p_c percent of population to produce p_g' offspring and then evaluate them.

    (a.2) Mutate p_m percent of population to produce p_g'' offspring and then evaluate them.

    (a.3) Merge population, offspring generated via recombination and offspring generated from mutation in a unique set ($p_g \cup p_g' \cup p_g''$) and then sort them.

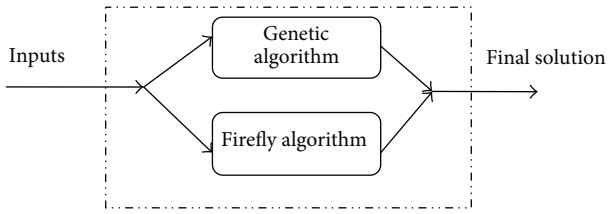


Figure 5: An illustration of simple parallelization hybridization.

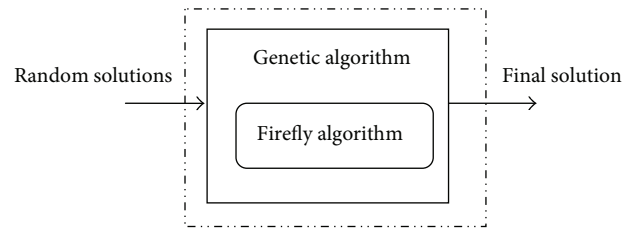


Figure 6: An illustration of embedded hybridization.

(a.4) Truncate best members of merged set and generate new population of size p_g $(p_g \cup p_g' \cup p_g'') \rightarrow p_g$

End for h

End while

(3) Save H_g members of best solutions obtain by GA in final_{ga}

(4) Randomly generate an initial population as artificial fireflies (p_f)

(5) While (not stop condition)

(b) For $n = 1: H_f$ (H_f is the total number of fireflies)

(b.1) For $m = 1: H_f$

If $I_n < I_m$

Generate new solution and update it (p_f) according to formulation in (15).

End if

End for m

End for n

End while

(6) Save H_f members of best solutions obtain by FF in final_{ff}

(7) Merge best of GA and best of FF $(\text{final}_{\text{ga}} \cup \text{final}_{\text{ff}}) \rightarrow \text{final}_h$

(8) Rank the member of final_h and select best of them as PHGF solution.

4.3.2. Algorithm EHGF. The second suggested algorithm is based on embedding technique. In this technique, one solution is embedded in another. According to the algorithms used in this paper, Genetic algorithm is considered the base, and firefly algorithm is only used for diversifying the answers. In other words, firefly plays the part of mutating operator in this suggested hybrid. Figure 6 shows an illustration of embedded hybridization.

The steps of EHGF are shown below.

(1) Randomly generate an initial population (p)

(2) While (not stop condition)

(a) for $h = 1: H$ (H is the number of individuals)

(a.1) Recombine p_c percent of population to produce p' offspring and then evaluate them.

(a.2) Merge population and offspring generated via recombination in a unique set $(p \cup p')$ and then sort them.

(a.3) Truncate best members of merged set and generate new population of size p. $(p \cup p') \rightarrow p_g$

End for h

(b) For $n = 1: H$

(b.1) For $m = 1: H$

If $I_n < I_m$

Generate new solution and update it according to formulation in (15) to form p_f.

End if

End for m

End for n

(c) Mutate p_m percent of population (p_f) to produce p'' offspring and then evaluate them.

(d) Merge population generated via mutation and population from firefly algorithm in a unique set $(p_f \cup p'')$ and then sort them.

(e) Truncate best members of merged set and generate new population of size p. $(p_f \cup p'') \rightarrow p$

End while

(3) Select best of final generation (p) as EHGF solution.

4.3.3. Algorithm ISHGF. In this technique, the output of genetic algorithm is used as the input of firefly algorithm, and then the output of firefly algorithm will be used as the input of genetic algorithm. This will be done over and over until finishing conditions come true. Such new algorithm is obtained from hybridization with iterative sequentialization method and is called ISHGF.

In ISHGF, according to the sizes of GA and FF population members, 3 scenarios might occur. The first scenario is $H_g = H_f$ (H_g is the number of individuals in the population and H_f is the total number of fireflies). In such case, the answers are sent continually from GA to FF algorithm and from FF to

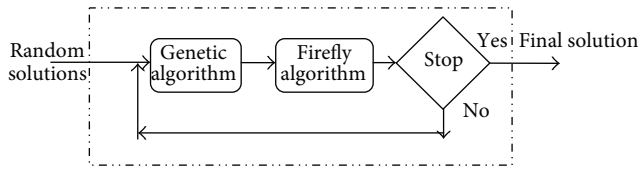


Figure 7: An illustration of iterative sequentialization hybridization.

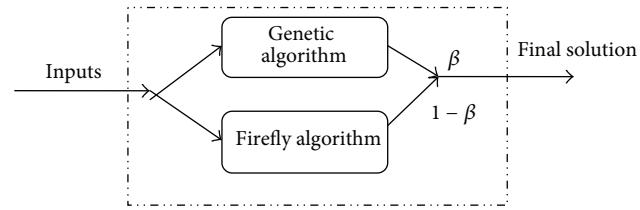


Figure 8: An illustration of simple workload division hybridization.

GA, until gaining the desired answer. The second scenario is $H_g > H_f$. In such case, the best members of GA population sit in the place of FF members, and the remaining group of members whose number is calculated by subtracting H_g from H_f are thrown away. Of course when the algorithm has not reached the desired answer, GA receives all the members of FF and compensates its lacking members by generating random answers. The 3rd scenario is $H_g < H_f$. Hereon, all GA members go into FF algorithm, and new random answers are generated as many as $H_f - H_g$. Of course in this scenario again, if the ending conditions do not arise, the algorithm returns to its beginning, so GA selects only as many as H_g members among the best answers of H_f and throws the rest away. Figure 7 shows an illustration of iterative sequentialization hybridization.

The steps of ISHGF are shown below ($H_g > H_f$).

(1) Randomly generate an initial population (p_g)

(2) While (not stop condition)

  (a) For $h = 1: H_g$ (H_g is the number of individuals in the population)

    (a.1) Recombine p_c percent of population to produce p'_g offspring and then evaluate them.

    (a.2) Mutate p_m percent of population to produce p''_g offspring and then evaluate them.

    (a.3) Merge population, offspring generated via recombination and offspring generated from mutation in a unique set ($p_g \cup p'_g \cup p''_g$) and then sort them.

    (a.4) Truncate best members of merged set and generate new population of size p_g. ($p_g \cup p'_g \cup p''_g$) $\rightarrow$ p_g

  End for h

  (b) Set best members of final population in size H_f as artificial fireflies (H_f is the total number of fireflies).

  (c) For $n = 1: H_f$

    (c.1) For $m = 1: H_f$

      If $I_n < I_m$

        Generate new solution and update it according to formulation in (15) to form p_f

      End if

    End for m

  End for n

  (d) Rank the solution, update the best and also generate ($H_g - H_f$) members randomly and add them to p_f (Added members + $p_f \rightarrow p_g$).

End while

(3) Select best of final generation as ISHGF solution.

4.3.4. Algorithm WHGF. The fourth suggested algorithm is simple workload division. This technique is very similar to simple parallelization, meaning that every algorithm works in parallel with and independent of other algorithms, but in the end, the collected answers appear in the final answer according to the determined share for every algorithm. For example, firefly algorithm has 60% and genetic algorithm has 40% share in the ultimate answer. Obviously, the final solution is the best answer among all the answers collected from both participant algorithms in the hybrid process. Figure 8 shows an illustration of simple workload division hybridization.

The steps of WHGF are shown below.

(1) Randomly generate an initial population (p_g)

(2) While (not stop condition)

  (a) For $h = 1: H_g$ (H_g is the number of individuals in the population)

    (a.1) Recombine p_c percent of population to produce p'_g offspring and then evaluate them.

    (a.2) Mutate p_m percent of population to produce p''_g offspring and then evaluate them.

    (a.3) Merge population, offspring generated via recombination and offspring generated from mutation in a unique set ($p_g \cup p'_g \cup p''_g$) and then sort them.

    (a.4) Truncate best members of merged set and generate new population of size p_g ($p_g \cup p'_g \cup p''_g$) $\rightarrow$ p_g

  End for h

End while

(3) Save H_g members of best solutions obtain by GA in final_{ga}

(4) Randomly generate an initial population as artificial fireflies (p_f)

(5) While (not stop condition)

(b) For $n = 1: H_f$ (H_f is the total number of firefly)

(b.1) For $m = 1: H_f$

If $I_n < I_m$

Generate new solution and update it (p_f) according to formulation in (15)

End if

End for m

End for n

End while.

(6) Save H_f members of best solutions obtain by FF in final_{ff}

(7) Merge β% best of GA and $1 - \beta$% best of FF (β% (final_{ga}) $\cup$ $1 - \beta$% (final_{ff})) $\rightarrow$ final_h

(8) Rank the member of final_h and select best of them as WHGF solution.

5. Computational Results

5.1. Data Gathering. In order to investigate the performance of suggested algorithms, numerous tests are required. First of all, certain numbers of the parameters required in the problem must be determined. The performance of suggested solutions is to be appraised in various scales: small, medium, and large. Knowing that, direct input of numbers into the parameters, and changing the numbers in every test will make the solution too complicated; hence, uniform statistical distribution was applied here. Table 3 shows the values of required parameters for solving this problem.

5.2. Tuning the Parameters. For improving the performance of suggested hybrid algorithms, a particular process called parameter tuning is required. Parameter tuning keeps the required parameters of the problem in the best possible range; so that every parameter performs in the best possible way. Numerous techniques have been used so far for setting the parameters of metaheuristic algorithms, including Trial And Error, Full-Factorial, Taguchi, Response Surface, and Neural Networks.

This paper aims to set the parameters of all 4 suggested algorithms with Taguchi Method, which was innovated by Taguchi in 1986 [34]. The first step is finding and introducing the influential parameters in algorithm. Since the suggested solutions are all based on genetic and firefly algorithms, the influential factors in performance of their hybrid also include the parameters of those two.

The parameters of genetic algorithm include mutation rate, crossover rate, number of iterations, and number of chromosomes in the population. The selected parameters in firefly algorithm also include the number of artificial fireflies, light absorption coefficient, mutation coefficient, and the maximum number of iterations. The parameter values are defined in 3 levels: small, medium and large. These values are shown with indexes 1, 2, and 3, respectively.

Table 3: Values of model parameters.

Parameter	Levels
m_{ijn}	~Uniform (100, 150)
$tk_j ta_j tg_j td_j$	~Uniform (0.4, 0.7)
f_{in}	~Uniform (50000, 80000)
f_l'	~Uniform (15000, 25000)
f_s''	~Uniform (1000, 5000)
a_j	~Uniform (20, 50)
b_{js}	~Uniform (1, 3)
$e_{ik} h_{il} o_{ls} d_{lk}$	~Uniform (50, 1000)
p_{kj}	~Uniform (100, 300)
q_{kj}	~Uniform (60, 170)
r_j	~Uniform (0.1, 0.5)
c_{ijn}	~Uniform (300, 600)
c_{lj}'	~Uniform (50, 150)
c_{sj}''	~Uniform (50, 100)

Since the mathematical model of this problem is not similar to other studies, the value of parameters cannot be compared with other researches; therefore, these levels have been determined with trial-and-error, within their permitted range. Taguchi solution approach looks at the number of factors and the levels and determines the required number of tests accordingly. Since this paper has selected 8 factors and 3 levels, 27 problems are to be solved. Taguchi technique in this problem has been performed by Minitab Software, on a PC with 3 GB RAM. Table 4 shows the considered parameters and their levels.

For tuning the parameters, the obtained values should become normalized, so as to make them independent of range. Numerous methods are available for making the parameters dimensionless, including methods fuzzy normalization, and linear normalization. This paper uses related percentage deviation (RPD) technique, with the following formula:

$$\text{RPD} = \frac{\left|\text{Algorihm Solution} - \text{Best Solution}\right|}{\text{Best Solution}} \times 100. \quad (17)$$

Table 5 shows the results of 27 times problem solving, with different levels of factors, and in RPD form.

The ultimate aim of parameter tuning is adjusting each of the algorithms in their best possible position. This is why a sample problem is initially selected and solved with various levels of parameters, and then the results are entered in a column. The best answer is the minimum value (because this is a minimizing problem). According to RPD formula, the best answer turns into zero, and the other answers are evaluated according to their deviation from the best; therefore, since every algorithm is evaluated separately, every column has 1 zero. The obtained numbers in every algorithm were entered into Minitab separately. The following graphs show the results of suggested levels for parameters, in Taguchi Method. Minitab Software introduces two graphs for every algorithm: one associated with mean and the other one with SNR. Every Mean diagram consists of 8 graphs, indicating

Table 4: Levels of algorithm parameters.

	Parameter	1	2	3
Genetic algorithm	C_P: crossover percentage	0.4	0.6	0.8
	M_P: mutation percentage	0.1	0.2	0.3
	I_{GA}: number of GA iterations	50	100	150
	P_S: population size	50	70	100
Firefly algorithm	N_F: number of fireflies	100	150	200
	L: light absorption coefficient	0.01	50	100
	M_F: mutation coefficient	0.2	0.4	0.8
	I_F: number of FF iterations	50	100	150

Table 5: Values of proposed algorithms in parameter tuning.

Problem number	C_P	M_P	I_{GA}	P_S	N_F	L	M_F	I_F	PHGF	EHGF	ISHGF	WHGF
1	1	1	1	1	1	1	1	1	52.3200	1.1200	14.0600	5.8000
2	2	2	2	2	1	1	1	1	71.2600	41.8600	16.8700	1.7700
3	3	3	3	3	1	1	1	1	14.0300	2.1800	14.4000	18.7800
4	2	1	1	1	2	2	2	1	4.1300	11.8000	18.0400	41.0000
5	3	2	2	2	2	2	2	1	12.9900	23.9200	38.1800	4.5800
6	1	3	3	3	2	2	2	1	3.1900	6.9100	12.7900	0.0000
7	3	1	1	1	3	3	3	1	2.4700	31.3700	7.2200	53.9300
8	1	2	2	2	3	3	3	1	18.0600	0.0000	57.5700	11.8900
9	2	3	3	3	3	3	3	1	31.8800	18.8100	41.1100	48.2300
10	1	3	2	1	3	2	1	2	10.5400	3.7000	17.6400	1.4100
11	2	1	3	2	3	2	1	2	93.4500	65.1600	62.4800	39.3300
12	3	2	1	3	3	2	1	2	92.3300	12.7800	56.5400	17.2400
13	2	3	2	1	1	3	2	2	10.9800	21.4500	43.1000	11.4100
14	3	1	3	2	1	3	2	2	13.9500	14.7800	68.5200	12.2000
15	1	2	1	3	1	3	2	2	17.7200	72.2900	39.4100	55.9300
16	3	3	2	1	2	1	3	2	33.4700	23.8600	33.4600	16.7600
17	1	1	3	2	2	1	3	2	19.3600	79.2100	41.6500	32.1600
18	2	2	1	3	2	1	3	2	91.3600	66.3900	43.7200	23.8700
19	1	2	3	1	2	3	1	3	28.7100	44.5700	53.7800	3.0500
20	2	3	1	2	2	3	1	3	4.6800	0.6900	6.6500	6.4200
21	3	1	2	3	2	3	1	3	23.8800	21.4600	82.0400	11.8900
22	2	2	3	1	3	1	2	3	29.0600	72.3600	55.5700	74.6000
23	3	3	1	2	3	1	2	3	82.1800	4.8300	51.6000	82.5400
24	1	1	2	3	3	1	2	3	47.0000	81.5000	44.8400	46.3100
25	3	2	3	1	1	2	3	3	19.8900	35.3300	0.0000	98.0700
26	1	3	1	2	1	2	3	3	0.0000	7.0600	1.7000	18.0400
27	2	1	2	3	1	2	3	3	78.8300	47.7000	37.6400	80.2100

parameter levels. The best value in every level is the minimum; for example, in the diagram of crossover rate in simple parallelization algorithm, level 1 with the lowest value is considered the optimum value for that parameter.

The other diagram is SNR, which illustrates the optimum level of every parameter in another way. The higher SNR level is better. Here is a caveat to keep in mind. If the parameters are tuned properly, the results of both Means and SNR diagrams will be equal; otherwise, if the interpretation of every parameter in average diagram is different from its optimum level in SNR diagram, the tests must be repeated.

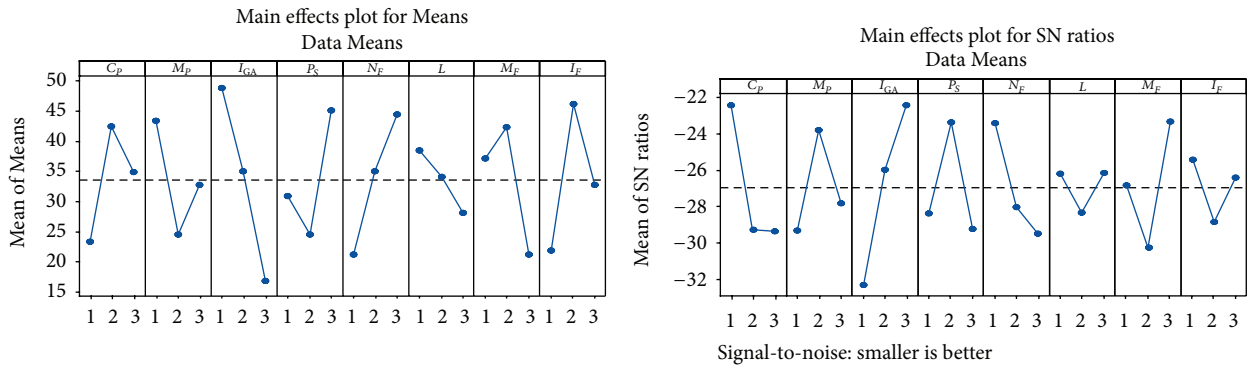
Note that we have one type of population in the whole EHGF algorithm; therefore, the number of fireflies does not define.

Table 6 shows the tuned levels of parameters in every algorithm, according to Figures 9, 10, 11, and 12.

After entering the tuned value of every parameter, the algorithm will be used for solving 40 problems in small,

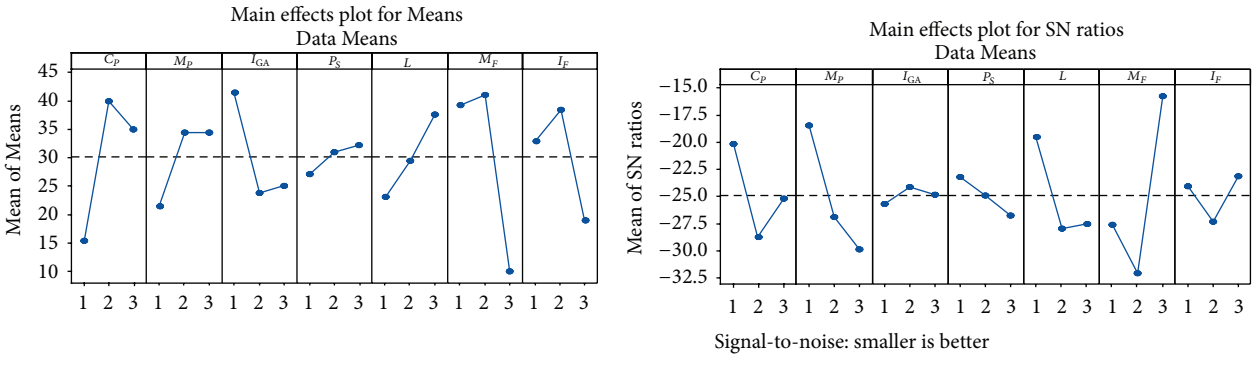
Table 6: Tuned parameters.

Algorithms	C_P	M_P	I_{GA}	P_S	N_F	L	M_F	I_F
PHGF	1	2	3	2	1	3	3	1
EHGF	1	1	2	1	—	1	3	3
ISHGF	1	3	2	1	1	1	3	1
WHGF	1	1	3	2	2	2	3	1



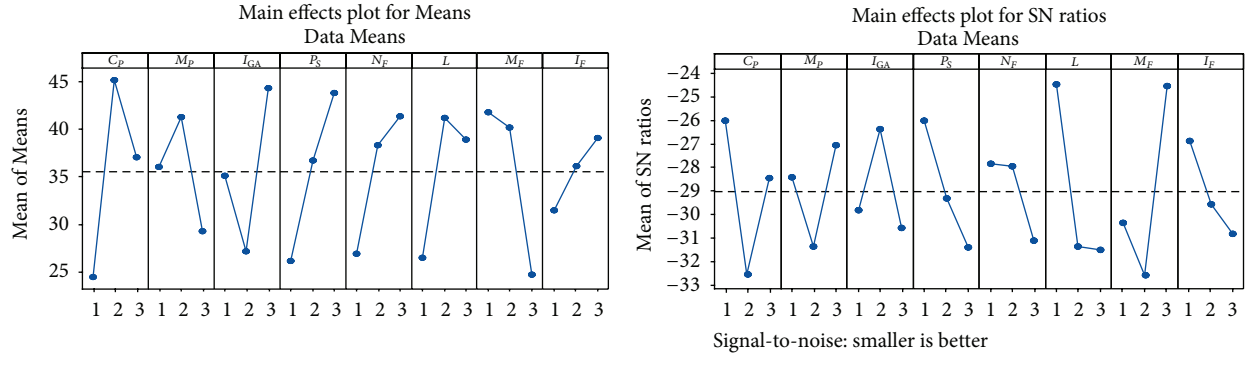
(a) (b)

Figure 9: Main effects plots of Means (a) and SNR (b) for PHGF.



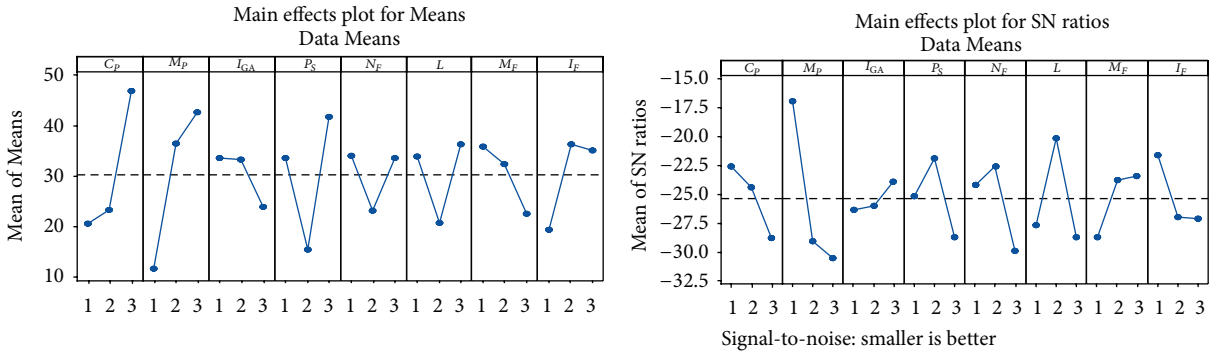
(a) (b)

Figure 10: Main effects plots of Means (a) and SNR (b) for EHGF.



(a) (b)

Figure 11: Main effects plots of Means (a) and SNR (b) for ISHGF.



(a) (b)

Figure 12: Main effect plots for Mean (a) and SNR (b) of WHGF.

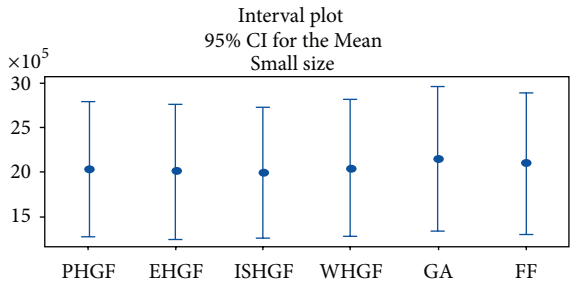


Figure 13: Comparison between algorithms in small size.

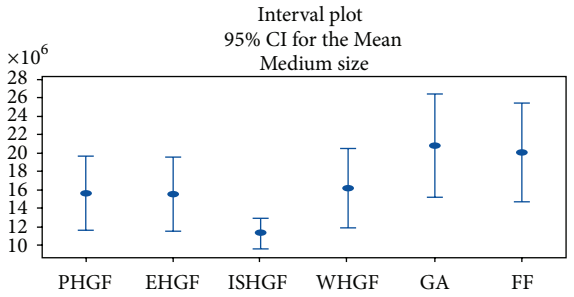


Figure 14: Comparison between algorithms in medium size.

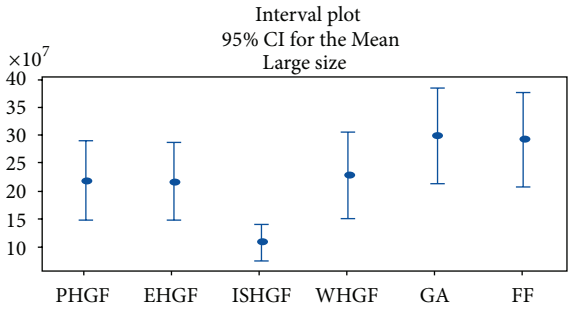


Figure 15: Comparison between algorithms in large size.

medium, and large scale. Table 7 demonstrates the resulted values. In addition we consider our base algorithms GA and FF to investigate effect of hybridization on the test problems.

5.3. Comparisons. The values obtained from all 6 solutions are contrasted according to statistical comparison techniques, with the aim to find the best solution. For each problem, the proposed algorithms are run 3 times, and then the average value is considered. In order to determine the significant difference between different values obtained by algorithm, ANOVA Statistical Test was devised with SPSS Software. Note that in order to compare more precisely each size considered separately. Tables 8, 9, and 10 and Figures 13, 14, and 15 show the comparisons between algorithms.

As shown in Figure 13 and also from ANOVA Table 8 (sig $>$ 0.05) there is no significant difference between algorithms in solving small size problems.

Although it seems that there is a difference between algorithms in Table 9 (Sig $<$ 0.05) because of overlapping between plots in Figure 14, we cannot certainly report which one is better in solving medium size problems.

Value obtained for sig lower than 0.05 is indicating that there is a significant difference between algorithms. In addition as shown in Figure 15, it is obvious that ISHGF achieves lower cost in large size than the other algorithms.

6. Conclusion

This paper introduces a new model for reducing the costs of closed-loop supply chain. The model includes several plants that are able to manufacture and remanufacture various types of products, using diverse technologies. In forward flow, the demanded products of customer centers were sent to them directly, without any go-between. In reverse flow, the collection centers are in charge of collecting and evaluating the products, so as to decide about sending the products to plants or disposal centers.

The proposed mathematical model in this problem grew more and more complicated by increasing the number of facilities, products, and technologies, so that classic methods became unreliable for solving the problem; hence, this paper

Table 7: Result of objective function in different sizes.

Test number	i	n	l	k	s	j	PHGF	EHGF	ISHGF	WHGF	GA	FF
Small size												
1	3	2	4	3	3	3	393576	393576	393576	393576	393576	393576
2	4	3	5	3	4	5	698845	698845	698845	698845	698845	698845
3	5	5	3	4	3	6	1067536	1064270	1064270	1065080	1068635	1067054
4	6	6	6	7	5	7	2158435	2148399	2148399	2201551	2399887	2202546
5	7	7	5	8	6	8	2858856	2814122	2748879	2857432	2987687	2976654
6	8	7	5	9	8	9	3636078	3607543	3606396	3767543	3896554	3865543
7	9	9	8	7	8	10	2158435	2158356	2148399	2201551	2376787	2259767
8	10	7	5	8	6	8	2858856	2814122	2748879	2857432	2976543	2905532
9	11	8	8	6	5	6	1742796	1684336	1678985	1701216	1865443	1801216
10	12	7	6	8	5	8	2737472	2719836	2700264	2735864	2865534	2798755
Medium size												
11	15	18	14	18	17	15	10996158	10979629	9774905	11995749	12924442	11952525
12	16	14	15	16	15	12	8712242	8739928	8503092	8770480	10765551	10655449
13	16	18	14	16	18	17	8156921	8006446	7509688	8175722	10678656	10654548
14	17	13	17	12	17	13	7806508	7906765	7455443	7894554	11677166	10766156
15	17	17	16	14	18	12	9793332	9845677	9325596	9776563	10976546	10617665
16	18	19	15	15	12	14	9607417	9635377	9066544	9795875	11987761	11276765
17	18	13	15	13	18	15	8977340	8764466	8053433	8657933	11876764	12487765
18	20	25	24	23	22	25	20929984	20719064	13421243	21928467	29988761	28866554
19	21	22	25	22	21	20	19332874	19664777	13766436	19147367	27745443	26876547
20	22	20	24	23	22	25	20968413	21013803	13613456	22022560	30989866	29977543
21	23	24	25	22	20	22	21257583	22043677	14075544	22290761	30445545	29998977
22	24	22	24	24	24	23	24156236	24477472	14154453	25276767	31766544	30243311
23	25	25	24	23	25	25	25132359	25632467	14879885	27698700	30775450	29131425
24	27	28	25	22	26	21	23524389	20556578	14079875	23035363	28765433	27875432
Large size												
25	40	30	20	30	20	30	39350882	38687623	16568206	39137465	45243073	44897765
26	50	35	25	35	30	35	53080191	54756375	35076364	58090324	61645455	59876453
27	60	40	24	40	30	40	69001276	69013803	55613456	70022560	78865564	77653422
28	70	45	25	45	35	45	86980117	85043677	60755440	89290761	95432121	91665534
29	80	50	24	50	45	50	127136406	134477472	65554453	125276767	158654430	148544768
30	90	55	24	55	50	55	129311201	125632467	71879885	127698767	175437681	170766443
31	100	60	25	60	55	60	153461145	150556578	75479875	150353632	179866554	178686554
32	110	65	36	65	60	65	200032401	212626371	91696575	200898322	308787651	298976633
33	120	70	37	70	65	70	205869473	214363366	100689990	229642535	348776542	336765543
34	130	75	38	75	60	75	235587893	233453673	115756433	245644424	358776523	350875641
35	140	80	40	80	70	80	267022500	259080977	138777655	265543523	368776421	357976592
36	160	90	41	90	80	90	336749110	326902525	147565442	344252522	408756455	407644327
37	180	95	45	95	85	95	373113655	367699878	169676567	387668145	459896754	449086765
38	190	95	45	100	80	95	392707538	382889301	181878668	437897798	480764533	471866543
39	200	100	50	100	90	100	410369288	402245224	199232433	405255598	489796875	478754532
40	220	110	90	110	80	110	426265014	421533566	209631466	478326559	589886552	580875641

Table 8: ANOVA of small size.

	Sum of squares	df	Mean square	F	Sig.
Between groups	$1.773E11$	5	$3.545E10$	0.031	1.000
Within groups	$6.218E13$	54	$1.151E12$		
Total	**6.236E13**	**59**			

Table 9: ANOVA for medium size.

	Sum of squares	df	Mean square	F	Sig.
Between groups	8.480E14	5	1.696E14		
Within groups	4.413E15	78	5.657E13	2.998	0.016
Total	**5.261E15**	**83**			

Table 10: ANOVA for large size.

	Sum of squares	df	Mean square	F	Sig.
Between groups	3.813E17	5	7.626E16		
Within groups	1.659E18	90	1.843E16	4.137	0.002
Total	**2.040E18**	**95**			

tried to use 4 different types of hybrid algorithms based on genetic and firefly algorithms.

After tuning the optimum parameters of those two algorithms, using Taguchi Approach in Minitab Software, the parameters were used for solving 40 problems with small, medium, and large scale. Finally, ANOVA Statistical Test was devised with SPSS Software, and this shows that iterative sequentialization technique gave the best answers in large size.

Closed-loop supply chain has considerable economic, environmental, and social effects; therefore, it has become a focal point in recent years. Such model can consider social and environmental factors, which are influential in the network. It can be formulated in multiobjective form.

Increasing the number of levels and participants of the supply chain design can help improve its compatibility with the needs of that industry and society.

Aside from that, considering uncertainty in parameters, and investigating the flexibility of model with regard to gradual changes can also help make the model significantly more robust.

Using other metaheuristic methods and artificial neural networks can also be considered among the future solution ideas.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] M. T. Melo, S. Nickel, and F. Saldanha-da-Gama, "Facility location and supply chain management—a review," *European Journal of Operational Research*, vol. 196, no. 2, pp. 401–412, 2009.

[2] E. H. Sabri and B. M. Beamon, "A multi-objective approach to simultaneous strategic and operational planning in supply chain design," *Omega*, vol. 28, no. 5, pp. 581–598, 2000.

[3] A. Syarif, Y. S. Yun, and M. Gen, "Study on multi-stage logistic chain network: a spanning tree-based genetic algorithm approach," *Computers and Industrial Engineering*, vol. 43, no. 1-2, pp. 299–314, 2002.

[4] V. Jayaraman and A. Ross, "A simulated annealing methodology to distribution network design and management," *European Journal of Operational Research*, vol. 144, no. 3, pp. 629–645, 2003.

[5] P. A. Miranda and R. A. Garrido, "Incorporating inventory control decisions into a strategic distribution network design model with stochastic demand," *Transportation Research Part E: Logistics and Transportation Review*, vol. 40, no. 3, pp. 183–207, 2004.

[6] F. Altiparmak, M. Gen, L. Lin, and T. Paksoy, "A genetic algorithm approach for multi-objective optimization of supply chain networks," *Computers and Industrial Engineering*, vol. 51, no. 1, pp. 196–215, 2006.

[7] A. Amiri, "Designing a distribution network in a supply chain system: formulation and efficient solution procedure," *European Journal of Operational Research*, vol. 171, no. 2, pp. 567–576, 2006.

[8] P. Tsiakis and L. G. Papageorgiou, "Optimal production allocation and distribution supply chain networks," *International Journal of Production Economics*, vol. 111, no. 2, pp. 468–483, 2008.

[9] M. S. Pishvaee, J. Razmi, and S. A. Torabi, "Robust possibilistic programming for socially responsible supply chain network design: a new approach," *Fuzzy Sets and Systems*, vol. 206, pp. 1–20, 2012.

[10] M. S. Pishvaee, S. A. Torabi, and J. Razmi, "Credibility-based fuzzy mathematical programming model for green logistics design under uncertainty," *Computers and Industrial Engineering*, vol. 62, no. 2, pp. 624–632, 2012.

[11] K. Govindan, H. Soleimani, and D. Kannan, "Reverse logistics and closed-loop supply chain: a comprehensive review to explore the future," *European Journal of Operational Research*, vol. 240, no. 3, pp. 603–626, 2015.

[12] V. Jayaraman, R. A. Patterson, and E. Rolland, "The design of reverse distribution networks: models and solution procedures," *European Journal of Operational Research*, vol. 150, no. 1, pp. 128–149, 2003.

[13] O. Listeş and R. Dekker, "A stochastic approach to a case study for product recovery network design," *European Journal of Operational Research*, vol. 160, no. 1, pp. 268–287, 2005.

[14] N. Aras, D. Aksen, and A. G. Tanugur, "Locating collection centers for incentive- dependent returns under a pick-up policy with capacitated vehicles," *European Journal of Operational Research*, vol. 191, no. 3, pp. 1223–1240, 2008.

[15] K. Schweiger and R. Sahamie, "A hybrid Tabu Search approach for the design of a paper recycling network," *Transportation Research Part E: Logistics and Transportation Review*, vol. 50, no. 1, p. 98, 2013.

[16] M. Eskandarpour, E. Nikbakhsh, and S. H. Zegordi, "Variable neighborhood search for the bi-objective post-sales network design problem: a fitness landscape analysis approach," *Computers & Operations Research*, vol. 52, part B, pp. 300–314, 2014.

[17] E. Roghanian and P. Pazhoheshfar, "An optimization model for reverse logistics network under stochastic environment by using genetic algorithm," *Journal of Manufacturing Systems*, vol. 33, no. 3, pp. 348–356, 2014.

[18] M. Fleischmann, P. Beullens, J. M. Bloemhof-Ruwaard, and L. N. Van Wassenhove, "The impact of product recovery on logistics network design," *Production and Operations Management*, vol. 10, no. 2, pp. 156–173, 2001.

[19] H. J. Ko and G. W. Evans, "A genetic algorithm-based heuristic for the dynamic integrated forward/reverse logistics network for 3PLs," *Computers & Operations Research*, vol. 34, no. 2, pp. 346–366, 2007.

[20] M. S. Pishvaee, R. Z. Farahani, and W. Dullaert, "A memetic algorithm for bi-objective integrated forward/reverse logistics network design," *Computers and Operations Research*, vol. 37, no. 6, pp. 1100–1112, 2010.

[21] M. S. Pishvaee and S. A. Torabi, "A possibilistic programming approach for closed-loop supply chain network design under uncertainty," *Fuzzy Sets and Systems*, vol. 161, no. 20, pp. 2668–2683, 2010.

[22] S. H. Amin and G. Zhang, "An integrated model for closed-loop supply chain configuration and supplier selection: multi-objective approach," *Expert Systems with Applications*, vol. 39, no. 8, pp. 6782–6791, 2012.

[23] M. S. Pishvaee and J. Razmi, "Environmental supply chain network design using multi-objective fuzzy mathematical programming," *Applied Mathematical Modelling: Simulation and Computation for Engineering and Environmental Systems*, vol. 36, no. 8, pp. 3433–3446, 2012.

[24] S. H. Amin and G. Zhang, "A multi-objective facility location model for closed-loop supply chain network under uncertain demand and return," *Applied Mathematical Modelling*, vol. 37, no. 6, pp. 4165–4176, 2013.

[25] M. Ramezani, M. Bashiri, and R. Tavakkoli-Moghaddam, "A new multi-objective stochastic model for a forward/reverse logistic network design with responsiveness and quality level," *Applied Mathematical Modelling*, vol. 37, no. 1-2, pp. 328–344, 2013.

[26] H. Soleimani, M. Seyyed-Esfahani, and M. A. Shirazi, "Designing and planning a multi-echelon multi-period multi-product closed-loop supply chain utilizing genetic algorithm," *International Journal of Advanced Manufacturing Technology*, vol. 68, no. 1–4, pp. 917–931, 2013.

[27] M. Faccio, A. Persona, F. Sgarbossa, and G. Zanin, "Sustainable SC through the complete reprocessing of end-of-life products by manufacturers: a traditional versus social responsibility company perspective," *European Journal of Operational Research*, vol. 233, no. 2, pp. 359–373, 2014.

[28] K. Devika, A. Jafarian, and V. Nourbakhsh, "Designing a sustainable closed-loop supply chain network based on triple bottom line approach: a comparison of metaheuristics hybridization techniques," *European Journal of Operational Research*, vol. 235, no. 3, pp. 594–615, 2014.

[29] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, New York, NY, USA, 1997.

[30] H.-F. Wang and H.-W. Hsu, "A closed-loop logistic model with a spanning-tree based genetic algorithm," *Computers and Operations Research*, vol. 37, no. 2, pp. 376–389, 2010.

[31] X. S. Yang, "Firefly algorithm, lévy flights and global optimization," in *Research and Development in Intelligent Systems XXVI*, pp. 209–218, Springer, Berlin, Germany, 2010.

[32] K. Eshghi and M. Kariminasab, *Combinatorial Optimization & Metaheuristics*, Mehr Azin Press, Tehran, Iran, 2012.

[33] F. G. Lobo and D. E. Goldberg, "Decision making in a hybrid genetic algorithm," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 121–125, IEEE Press, Indianapolis, Ind, USA, April 1997.

[34] G. Taguchi, S. Chowdhury, and Y. Wu, *Taguchi's Quality Engineering Handbook*, Wiley, New York, NY, USA, 2005.