Research Article

A Two-Stage Simulated Annealing Algorithm for the Many-to-Many Milk-Run Routing Problem with Pipeline Inventory Cost

Yu Lin,¹ Zheyong Bian,¹ Shujing Sun,² and Tianyi Xu¹

¹College of Management & Economics, Tianjin University, 92 Weijin Road, Nankai District, Tianjin 300072, China

²Simon Business School, University of Rochester, 304 University Park, Rochester, NY 14620, USA

Correspondence should be addressed to Zheyong Bian; tjubianzheyong@163.com

Received 20 March 2015; Revised 21 July 2015; Accepted 22 July 2015

Academic Editor: Antonios Tsourdos

Copyright © 2015 Yu Lin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In recent years, logistics systems with multiple suppliers and plants in neighboring regions have been flourishing worldwide. However, high logistics costs remain a problem for such systems due to lack of information sharing and cooperation. This paper proposes an extended mathematical model that minimizes transportation and pipeline inventory costs via the many-to-many Milk-run routing mode. Because the problem is NP hard, a two-stage heuristic algorithm is developed by comprehensively considering its characteristics. More specifically, an initial satisfactory solution is generated in the first stage through a greedy heuristic algorithm to minimize the total number of vehicle service nodes and the best insertion heuristic algorithm to determine each vehicle's route. Then, a simulated annealing algorithm (SA) with limited search scope is used to improve the initial satisfactory solution. Thirty numerical examples are employed to test the proposed algorithms. The experiment results demonstrate the effectiveness of this algorithm. Further, the superiority of the many-to-many transportation mode over other modes is demonstrated via two case studies.

1. Introduction

Concomitant with the accelerated pace of globalization, economic competition has intensified. For better development, companies around the world have committed themselves to reducing costs, of which one of the most important constituents is logistics cost. A sign of this is the flourishing economic and technological development zones in China, where numerous enterprises are collocated in the same area. Other examples include the four major motor cities of the world (Toyota Motor City, Detroit Motor City, Turin Motor City, and Stuttgart Motor City). Accordingly, a multisupplier and multiplant logistics network system based on a factory's spatial location or functional interaction is formed. Nevertheless, many systems of this kind have not fully developed into a real "cohesive cluster," especially China's automotive industry parks. Most automotive parks in China are simply "clustered" based on spatial location instead of the functional interaction among them. Therefore, inbound logistics transportation for

this many-to-many (n - n) system remains a problem. Without doubt, inbound logistics transportation problems of this kind are more complex. However, proper planning can generate enormous benefits: it can not only strengthen cooperation among enterprises and contribute to their further development, but also reduce the cost of the entire logistics system and mitigate the pressure on logistics and circulation costs for society.

Inbound logistics transportation methods can generally be classified into three types: direct shipping, Milk-run, and cross-docking. Berman and Wang [1] describe these three modes and their respective characteristics. Cross-docking is characterized by long transportation time and large pipeline inventory and is therefore suitable for long-distance transportation. In contrast, direct shipping and Milk-run have the advantage of short transportation time and small pipeline inventory, which make them suitable for short-range transportation. Considering the large number of suppliers and plants and the short distances between

them, direct shipping or Milk-run is more applicable for an industrial park. This paper focuses on Milk-run route planning with multiple suppliers and plants within a certain adjacent area.

The name “Milk-run” originated from the traditional system by which milk was sold in the West, in which the milkman simultaneously delivered full bottles of milk and collected the empty ones according to a predefined route. After completing a roundtrip, he returned with the empty bottles to the starting point. This method has subsequently been extensively utilized in miscellaneous industries and the auto manufacturing companies of the world and has developed into a popular way of picking up and delivering goods for multiple suppliers and plants using the same freight car [2]. Nemoto et al. [3] presented two Milk-run methods used by Toyota’s inbound logistics. The first is a many-to-one ($n-1$) mode, in which the car assembly manufacturer dispatches one truck at a specified time to visit various suppliers (i.e., parts suppliers) following a predefined route to collect parts or products and deliver them to the factories. The second is a one-to-many ($1-n$) mode, in which the parts are consolidated at the place of departure in such a way that they are fully loaded on the truck, transported, and unloaded at the various destinations. For both of these logistics modes, managers need to design and plan for specific routes, so as to minimize the overall logistics costs. Taking this rationale a step further, when all suppliers and plants share information with each other, then the many-to-many ($n-n$) Milk-run transportation described in [1] is achieved; that is, trucks pick up products from one or several suppliers and deliver them to one or several plants. Compared with Toyota’s $n-1$ and $1-n$ Milk-run system, this integrated $n-n$ method can significantly reduce the transportation costs to a larger extent for the entire logistics system.

The problem this paper focuses on is based on the classical vehicle routing problem (VRP). The generic VRP and many of its practical occurrences were studied intensively in the literature (Raff [4], Laporte and Nobert [5]). Then some extensive versions of VRP, such as vehicle routing with time windows (VRPTW) [6, 7] and multidepot vehicle routing problems (MDVRP) [8, 9], have been widely studied. These achievements of VRP have been already applied in the Milk-run system. Chuah and Yingling [10] and Jiang et al. [11] studied the common frequency routing problem in the Milk-run system and minimized transportation and inventory costs using column generation combined with the tabu search algorithm. Using genetic algorithm and a robust optimization approach, Sadjadi et al. [2] and Jafari-Eskandari et al. [12, 13] solved general and specific Milk-run issues with time windows and inventory uncertainty. Zhang and Li [14] analyzed the impact of the Milk-run method on supply chains, the conditions for using this method, and the changes in cost before and after adopting the Milk-run plan for a company. They also optimized the Milk-run mode by classifying containers into various carrying capacities based on the integral slice algorithm model. Yun et al. [15] analyzed the relationship between inventory and transportation, established an integrated transportation and inventory optimization model based on Milk-run, proposed

genetic algorithm to solve the vehicle scheduling problem, applied a stepwise iterated algorithm to balance inventory and transportation costs, and eventually minimized the total costs for the entire Milk-run logistics system. Du et al. [16] investigated the parameter settings of a real-time vehicle-dispatching system for consolidating Milk-runs. Ma and Sun [17] employed a mutation ant colony algorithm to solve the Milk-run vehicle routing problem with the fastest completion time based on dynamic optimization. However, all of these studies targeted the multiple suppliers to one plant ($n-1$) problem. Effectually, none of them can be directly applied to Milk-run systems with multiple suppliers and plants. Given the status quo, this paper proposes a solution to the Milk-run routing problem involving multiple suppliers and plants ($n-n$).

A number of researchers have already studied many-to-many transportation problems. The Pickup and Delivery Problem (PDP) is one of the research areas in $n-n$ transportation issues. In general PDP, there is no restriction on the choice of pickup point to provide products for each specified delivery point, as long as the demand of the delivery point is satisfied. However, in practical terms, for inbound logistics systems, a plant’s demand for each supplier is always clearly determined because the plants will place specific orders with each supplier. Thus, the general PDP is not entirely consistent with the actual $n-n$ inbound logistics systems. Furthermore, the set of variables employed to represent the commodities in PDP [18, 19] is not suitable for problems in practical terms. Further, because, in many manufacturing industries, such as auto manufacturing, the number of parts used in an individual assembly plant is more than 3,000 [20], the problem will become more complex if the set is defined based on the types of products. In light of the discussions above, PDP is not applicable for the type of logistics systems focused on in this paper. Therefore, an effective mathematical model with determined demand (hereafter, indicating the total demand of plant p for supplier i regarding one or more products, which will be described in more detail in Section 2) should be constructed.

The cross-docking system is another transportation mode in inbound logistics involving multiple suppliers and plants. Several scholars have already studied vehicle routing problems in this system. For example, Berman and Wang [1] considered the problem of selecting the appropriate distribution strategy (direct shipping and cross-docking) for delivering a family of products from a set of suppliers to a set of plants so that the total transportation, pipeline inventory, and plant inventory costs were minimized. Lee et al. [21] optimized vehicle routing schedules by planning and scheduling vehicle routes during the pickup and delivery stages of cross-docking. Musa et al. [22] employed the ant colony algorithm to solve the $n-n$ transportation problem of the cross-docking network. Miao et al. [23] employed a hybrid genetic algorithm to solve the multiple crossdocks problem with supplier and customer time windows. Mousavi and Tavakkoli-Moghaddam [24] minimized the total cost by simultaneously taking cross-dock’s location and route planning into consideration. The cross-docking transportation mode has been used extensively in long-distance transportation. However, it is not applicable

for short-range transportation [2, 25]. In the cross-docking system, a vehicle must first pick up goods from suppliers and then transport them to the cross-dock, where inbound materials are sorted, consolidated, and stored until the outbound shipment is complete and ready to ship and finally deliver the goods to each plant. This increases the distance and the time needed for each transportation task and, thus, unnecessary pipeline inventory costs. In contrast, the Milk-run system has no transfer system similar to the cross-dock because all goods are directly delivered to the corresponding plant after they are picked up from each supplier. Consequently, all processes, including loading and unloading, are completed in the vehicle within a shorter time, which makes the Milk-run mode more suitable for short-distance transportation. Therefore, it is necessary to develop an effective method to solve the Milk-run routing problem involving multiple suppliers and plants in neighboring regions.

From the discussion above, this paper contributes to solving the Milk-run routing problem involving multiple suppliers and plants in neighboring regions, that is, the many-to-many Milk-run routing problem (MM-MRP). In addition, we also consider the pipeline inventory cost caused by damage to products during the transportation process. It is necessary to consider the pipeline inventory costs because in some specific situations, some products, materials, and commodities, such as parts of some precision instruments, fragile parts, refrigerated foods, and some primary products, are more likely to be damaged with longer transportation time and more frequent loading and unloading times. On the basis of the above discussion, we firstly formulate the many-to-many Milk-run routing problem with pipeline inventory cost (MM-MRP-PIC) and extend the vehicle routing problem (VRP) model to MM-MRP-PIC model.

Because VRP is NP hard, MM-MRP-PIC, an extension of VRP, is surely NP hard. Exact algorithms are difficult to find the optimal solutions within a reasonable time. Many scholars have developed heuristic algorithms to solve the VRP. Alfa et al. [26], Osman [27], and Van Breedam [28] applied simulated annealing to solve VRP and verified its effectiveness. Tabu search was developed by Osman [27] and Gendreau et al. [29] to solve VRP. Furthermore, genetic algorithms [30, 31], ant colony optimization [32–35], and particle swarm optimization [36, 37] are also developed to solve VRP successfully. However, some evolutionary algorithms, such as genetic algorithm, are inefficient to solve MM-MRP-PIC because the problem is so complex that they will spend much time to tackle illegal solutions generated during iterative process. In addition, ant colony can solve the problems whose objective function is only involved with transportation cost while it is difficult to tackle the pipeline inventory cost. Particle swarm optimization is also difficult to be designed for this problem because so various mutations should be considered that the PSO formulas cannot define these mutations together. However, simulated annealing can easily tackle illegal solutions and various mutations simultaneously considering the pipeline inventory cost. Therefore, we employ simulated annealing and make some improvements to solve MM-MRP-PIC. The algorithm is two-stage simulated annealing with limited search scope

(TSSALSS). The effectiveness of this algorithm is verified via thirty numerical examples. Compared with traditional SA, TSSALSS not only significantly reduces the computing time, but also improves the quality of the solutions.

The remainder of this paper is organized as follows. In Section 2, we describe the problem and propose the MM-MRP-PIC model. Section 3 is devoted to the solution method (TSSALSS) for this model. Thirty numerical examples are presented to demonstrate the efficacy of our algorithm in Section 4. In Section 5, we present two case studies to demonstrate the validity of the many-to-many transportation mode. Finally, we conclude this paper in Section 6.

2. Formulation

2.1. Problem Description. This paper focuses on logistics systems comprising multiple suppliers and multiple plants in certain neighboring regions. In such a system, every supplier produces one or more parts, and each of the plants places corresponding orders to suppliers. The total demand of plant p for the products of supplier i (D_{pi}) can be indicated by the ratio of the products' total volume to the truck capacity. The transportation process is organized as follows. A certain number of trucks, each of which has its own flow, are first scheduled for the shipping task. In each flow, the truck sets off from a plant, picks up products at a number of suppliers, delivers those products to corresponding plants, and finally returns to the departure point upon finishing the transportation task. This paper targets route planning for this kind of logistics systems, with the objective of minimizing transportation and pipeline inventory costs for the entire system.

The following three assumptions are made to this problem.

Assumption 1. Product quantities cannot be split and each vehicle's transportation frequency is one.

Assumption 2. There are enough vehicles at each plant to complete the transportation task.

Assumption 3. All vehicles employed for transportation are of the same type and have cargo capacity $Q \geq D_{pi}$.

2.2. Notations

(1) *Constants.* Consider the following.

$S = \{1, 2, \dots, n\}$: set of suppliers.

$P = \{n + 1, n + 2, \dots, n + m\}$: set of plants.

$N = \{1, 2, \dots, n + m\}$ ($N = S \cup P$): set of nodes, including all suppliers and plants.

$V = \{1, 2, \dots, s\}$: set of vehicles employed to complete the transportation task, with each vehicle corresponding to a path; that is, V is also the set of flows.

C_{ij} : transportation cost for shipping one truckload of products from node i to node j , $i, j \in N$.

D_{pi} : the demand of plant p for the products of supplier i .

v_{pi} : pipeline inventory cost per unit time per unit of product D_{pi} .

t_{ij} : time spent in transportation from node i to node j .

Q : capacity of each vehicle.

β : loading or unloading time per unit of product.

(2) *Decision Variables*. Consider the following:

$$x_{ij}^k = \begin{cases} 1, & \text{if vehicle } k \text{ passes from node } i \text{ to node } j \\ 0, & \text{otherwise,} \end{cases} \quad i, j \in N, k \in V, \quad (1)$$

$$y_{pi}^k = \begin{cases} 1, & \text{if vehicle } k \text{ transports } p\text{'s demand for } i\text{'s product} \\ 0, & \text{otherwise} \end{cases} \quad p \in P, i \in S, k \in V.$$

(3) *Variables That Are Dependent on the Decision Variables*. Consider the following:

L_{ij}^k : pipeline inventory cost for products on vehicle k per unit time from node i to node j , $i, j \in N, k \in V$.

T_i^k : time spent when vehicle k loads products at supplier i , $i \in S$.

T_p^k : time spent when vehicle k unloads products at plant p , $p \in P$.

2.3. *The Proposed Mathematical Model for MM-MRP-PIC*. Consider the following:

$$\begin{aligned} \text{Min } Z_1 = & \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} C_{ij} \cdot x_{ij}^k + \sum_{k \in V} \sum_{j \in S} \sum_{i \in S} L_{ij}^k (T_j^k + t_{ij}) \\ & + \sum_{k \in V} \sum_{i \in S} \sum_{p \in P} L_{ip}^k t_{ip} \\ & + \sum_{k \in V} \sum_{p \in P} \sum_{q \in P} L_{pq}^k (T_p^k + t_{pq}), \end{aligned} \quad (2)$$

where

$$\begin{aligned} T_i^k &= \beta \sum_{p \in P} y_{pi}^k D_{pi}, \quad i \in S, k \in V \\ T_p^k &= \beta \sum_{i \in S} y_{pi}^k D_{pi}, \quad p \in P, k \in V \end{aligned} \quad (3)$$

subject to

$$\sum_{p \in P} \sum_{i \in S} y_{pi}^k D_{pi} \leq Q, \quad k \in V \quad (4)$$

$$\sum_{k \in V} y_{pi}^k = \text{sgn}(D_{pi}), \quad p \in P, i \in S \quad (5)$$

$$\sum_{j \in N} x_{ji}^k = \text{sgn} \left(\sum_{p \in P} y_{pi}^k \right), \quad i \in S, k \in V \quad (6)$$

$$\sum_{j \in N} x_{ij}^k = \text{sgn} \left(\sum_{p \in P} y_{pi}^k \right), \quad i \in S, k \in V \quad (7)$$

$$\sum_{j \in N} x_{pj}^k = \text{sgn} \left(\sum_{i \in S} y_{pi}^k \right), \quad p \in P, k \in V \quad (8)$$

$$\sum_{j \in N} x_{jp}^k = \text{sgn} \left(\sum_{i \in S} y_{pi}^k \right), \quad p \in P, k \in V \quad (9)$$

$$\sum_{p \in P} \sum_{i \in S} x_{pi}^k = 1, \quad k \in V \quad (10)$$

$$\sum_{i \in S} \sum_{p \in P} x_{ip}^k = 1, \quad k \in V \quad (11)$$

$$\sum_{i, j \in U} x_{ij}^k \leq |U| - 1, \quad (12)$$

$$2 \leq |U| \leq |N| - 2, \quad U \subset N, k \in V$$

$$L_{pi}^k = 0, \quad p \in P, i \in S, k \in V \quad (13)$$

$$L_{ij}^k = x_{ij}^k \left(\sum_{l \in N} L_{li}^k + \sum_{p \in P} D_{pi} y_{pi}^k v_{pi} \right), \quad (14)$$

$$i \in S, j \in N, k \in V$$

$$L_{pj}^k = x_{pi}^k \left(\sum_{l \in N} L_{lp}^k - \sum_{i \in S} D_{pi} y_{pi}^k v_{pi} \right), \quad (15)$$

$$p \in P, j \in N, k \in V$$

$$x_{ij}^k \in \{0, 1\}, \quad i, j \in N, k \in V \quad (16)$$

$$y_{pi}^k \in \{0, 1\}, \quad p \in P, i \in S, k \in V. \quad (17)$$

The first part of the objective function is direct transportation cost; the second, third, and fourth parts are pipeline inventory costs. The pipeline inventory costs depend on the road transportation time and residence time for freight loading or unloading at each point. An immediate idea for calculating the pipeline inventory costs for products D_{pi} is multiplying the pipeline inventory costs for products per unit time and the corresponding pipeline transportation time. The pipeline transportation time of D_{pi} is the period starting from the vehicle's departure time at supplier i and ends at the arrival time at plant p . It includes transportation time on the way,

loading time at various other suppliers, and unloading time at other plants in the same flow (we ignore the loading time of D_{pi} at supplier i and the unloading time at plant p because the pipeline inventory cost of D_{pi} caused by the loading time at supplier i and the unloading time at plant p are constant and independent of the decision variables). Although this idea is easy to understand, it is difficult to describe using mathematical models. Therefore, this paper divides the above process into three stages and calculates the respective cost for each part.

- (1) Pickup stage: the pipeline inventory cost when vehicle k picks up products at supplier i and goes to the next supplier j can be represented by the product of the costs per unit time for the freight (L_{ij}^k) and the sum of the transportation time on road (t_{ij}) and the pickup time at supplier j (T_j^k); that is, $L_{ij}^k \times (t_{ij} + T_j^k)$. T_j^k is proportional to the loading volume of vehicle k at supplier j , and the scale factor is β .
- (2) Stage between the final pickup point and the first delivery point: the pipeline inventory cost is determined by the product of the costs per unit time for the freight (L_{ip}^k) and the time spent on transportation (t_{ip}); that is, $L_{ip}^k \times t_{ip}$.
- (3) Delivery stage: the pipeline inventory cost when vehicle k unloads products at plant p and goes to the next plant q can be represented by the product of the costs per unit time for the freight (L_{pq}^k) and the sum of the transportation time on road (t_{pq}) and the unloading time at p (T_p^k); that is, $L_{pq}^k \times (t_{pq} + T_p^k)$. T_p^k is proportional to the unloading volume of vehicle k at plant p , and the scale factor is also assumed to be β .

The following is a detailed description of constraints (4)–(17). Equation (4) signifies that the transportation task of vehicle k cannot exceed its capacity. Equation (5) means that as long as plant p has a demand for products from supplier i , a vehicle should be arranged to facilitate the transportation task. Equations (6)–(9) ensure that the path of each vehicle forms a loop. In classical VRP, there can only be one vehicle flow into and out of each node. In contrast, in this paper, more than one truck can go into and out of each node because multiple vehicles can be employed to execute different tasks at the same node. However, every vehicle that flows into a node must also leave that node; that is, the inflow and outflow for each vehicle at each node must be the same. In each flow, a vehicle must first pick up products from suppliers, then deliver goods to corresponding plants, and finally return to the departure point. To ensure that this process is conducted in an orderly manner, (10)–(11) restrict each vehicle to one single route from the plants to the suppliers, and vice versa. Equation (12) ensures that the solution does not contain any illegal isolated subtour. Equation (13) signifies that the pipeline inventory cost from the first plant to the starting supplier is zero. This is obvious because the vehicle sets off from a plant; therefore, it is empty during this period.

Equation (14) represents the pipeline inventory cost per unit time for vehicle k from node i ($i \in S$) to node j ($j \in N$) in the pickup stage, which is the sum of the pipeline inventory cost per unit time for vehicle k before node i and that of the newly loading freight. Equation (15) is similar to (14); it represents the pipeline inventory cost per unit time for vehicle k from node p ($p \in P$) to node j ($j \in N$) in the delivery stage. Equations (16)–(17) signify the range of decision variables.

3. Approach

As an extension of VRP, MM-MRP-PIC is NP hard. As a result, a two-stage simulated annealing algorithm with limited search scope (TSSALSS) is developed in this paper. Geng et al. [38] employed an adaptive simulated annealing algorithm with greedy search to solve the traveling salesman problem (TSP) and verified its effectiveness. In this paper, the process of limiting the search scope in TSSALSS is similar to the greedy search in [38], but we use a much simpler method to limit the search scope instead of greedy search which requires calculating the objective function frequently. A detailed solution process for TSSALSS is given below.

Varanelli and Cohoon [39] stated that conventional SA can be initiated at a low temperature in an attempt to improve the heuristic solution. Thus, a faster heuristic (the first stage) is used to replace the SA actions occurring at the highest temperatures in the cooling schedule. In the first stage, an initial satisfactory solution is constructed in the following three steps: (1) Determine an initial feasible task allocation plan. (2) Minimize the total number of vehicle service nodes Z_2 (calculated using formula (19) or (20)) by adjusting the task allocation for each vehicle. (3) Employ the best insertion heuristic algorithm to determine the route for each vehicle, which is achieved when the transportation cost is relatively optimal. In the second stage, the solution obtained in the first stage is improved by the *Simulated Annealing (SA) algorithm*. In order to improve the search efficiency, in this paper, the scope of the search in the algorithm is limited to a specific range. That is, in the search process, all feasible solutions are subject to the constraint that the total number of vehicle service nodes be not greater than $\gamma \cdot Z_2^{\min}$, in which $Z_2^{\min}$ is obtained in the first stage and γ is a number slightly larger than one; we find $1.1 < \gamma < 1.25$ to be the most appropriate. Because it is much simpler to calculate ΔZ_2 (the two neighbor solutions' difference in the total number of vehicle service nodes, Z_2) than to calculate ΔZ_1 (the two neighbor solutions' difference in the objective function value, Z_1), the search process will be much more efficient with the limited search scope. This method can not only reduce the unwanted time caused by searching the overall feasible solution domain, but also prevent the global optimal solution being overlooked by appropriate expansion of the search scope adjusted by coefficient γ . The structure of the algorithm is shown in Figure 1. The following is a detailed description of the algorithm.

3.1. The First Stage of Algorithm. This stage consists chiefly of three steps: determination of an initial feasible task allocation

FIGURE 1: Flow chart for the TSSALSS.

FIGURE 2: Instances of the four neighbors.

plan, minimization of the total number of service nodes (Z_2) accessed by all vehicles via the greedy heuristic algorithm, and determination of each vehicle's route via the best insertion heuristic algorithm. The algorithms used in each step are as follows.

Algorithm 1 (formation of an initial feasible task allocation plan). We first determine the number of vehicles (s) needed:

$$s \leq \left\lceil \frac{\sum_{p \in P} \sum_{i \in S} D_{pi}}{\alpha \cdot Q} \right\rceil, \quad (18)$$

TABLE 1: Results obtained for the first stage of algorithm.

Instance	$\bar{Z}_2$	$\bar{Z}_2^{\min}$	$\bar{Z}_1(X^0)$	$\bar{Z}_1(X)$	$\bar{T}(s)$	M
S1	37.1	21.3	650.756	415.469	0.0038	7000
S2	42.1	28.4	997.775	589.488	0.0053	7000
S3	75.2	41.3	1717.88	827.196	0.011	25000
S4	87.9	59.1	2160.15	1271.33	0.015	25000
S5	70.8	36.8	1555.01	721.974	0.0089	20000
S6	78.5	46.0	1402.56	800.643	0.012	20000
S7	130.6	70.2	2938.82	1310.18	0.023	60000
S8	162.0	90.7	3845.58	1855.90	0.035	60000
M1	304.2	142.0	6571.25	2610.43	0.085	200000
M2	347.4	184.4	9322.98	4429.06	0.11	200000
M3	459.7	213.1	14318.9	5404.58	0.21	500000
M4	531.6	273.0	14118.2	6618.70	0.27	500000
M5	1026.1	425.5	31096.5	11244.5	0.60	1500000
M6	1101.0	546.2	34617.8	15927.7	0.81	1500000
M7	867.0	374.0	27949.6	9759.69	0.39	1000000
M8	928.9	467.9	30983.4	13920.5	0.54	1000000
M9	1710.2	710.3	52170.3	18493.5	1.85	3000000
M10	1856.0	916.0	57091.2	25443.4	1.66	3000000
L1	899.1	472.1	36851.0	17388.2	0.60	1000000
L2	1795.3	913.2	73359.4	33653.0	1.73	3000000
L3	1799.8	946.8	78274.9	37801.9	1.86	3000000
L4	1606.1	1857.2	174523.0	77891.9	5.41	9000000
L5	1877.7	947.7	68970.8	31358.7	1.78	3000000
L6	3770.7	1831.3	148326.3	65545.2	5.17	9000000
L7	3768.3	1857.3	144844.0	64654.8	5.54	9000000
L8	7564.6	3719.6	287945.5	129019.1	17.13	30000000
L9	1926.6	963.2	75724.3	33979.2	1.70	3000000
L10	3823.3	1860.0	149613.3	65616.0	5.07	9000000
L11	3830.4	1927.5	208620.9	91394.0	5.31	9000000
L12	7680.7	3746.5	416486.8	181603.0	16.84	30000000

where α is a decimal slightly less than one. Note that if α is too close to one, we may miss the optimal solution, but if α is too small, it will cause unnecessary search times. We find that $\alpha \in [0.85, 0.9]$ is most appropriate.

To ensure that (4) is satisfied, that is, no vehicle's transportation task exceeds its loading capacity, the following method is used for task allocation:

Randomly assign all transportation tasks to every vehicle.

Do while at least one vehicle's loading tasks are larger than the capacity Q .

Select the vehicles with the maximum and minimum loading tasks, respectively.

Do while the transportation task of the maximum-loading-task vehicle is no longer greater than its loading capacity.

Assign one task of the maximum-loading-task vehicle to the minimum-loading-task vehicle.

End do

End do

A feasible task allocation plan is thus obtained; this is indicated by the set $Y = \{y_1, y_2, \dots, y_s\}$, where $y_k = \{y_{pi}^k \mid \forall p, i\}$. Determine the service nodes, represented by nodes_k , for each vehicle. For vehicle k , if $\sum_{p \in P} y_{pi}^k \geq 1$ ($\forall i \in S, k \in V$), then $i \in \text{nodes}_k$; otherwise $i \notin \text{nodes}_k$. Similarly, if $\sum_{i \in S} y_{pi}^k \geq 1$ ($\forall p \in P, k \in V$), then $p \in \text{nodes}_k$; otherwise $p \notin \text{nodes}_k$. The collection of nodes for all the vehicles is then represented by the set $\text{flow} = \{\text{nodes}_1, \text{nodes}_2, \dots, \text{nodes}_s\}$.

Algorithm 2 (greedy heuristic algorithm). This algorithm minimizes the total number of service nodes (Z_2) accessed by all vehicles, represented by

$$\min Z_2 = \sum_{k \in V} \sum_{i \in S} \text{sgn} \left(\sum_{p \in P} y_{pi}^k \right) + \sum_{k \in V} \sum_{p \in P} \text{sgn} \left(\sum_{i \in S} y_{pi}^k \right); \quad (19)$$

that is,

$$\min \sum_{k \in V} |\text{nodes}_k|. \quad (20)$$

The greedy heuristic algorithm is designed as follows:

Set the current vehicle task allocation plan as the one obtained in Algorithm 1 and calculate its total number of service nodes (Z_2). Initialize $Z_2^{\min} = Z_2$.

For $f = 1 : M$

Randomly choose a vehicle k_1 ($k_1 \in V$); one of its tasks is $D_{p_0 i_0}$ ($y_{p_0 i_0}^{k_1} = 1$), and another vehicle is k_2 ($k_2 \in V, k_2 \neq k_1$).

Do while the sum of $D_{p_0 i_0}$ and k_2 's transportation task exceeds the loading capacity of k_2

Repeat the random selection of $k_1, k_2, D_{p_0 i_0}$

End do

If $|\text{nodes}_{k_1}| + |\text{nodes}_{k_2}|$ decreases or remains unchanged ($\Delta Z_2 \leq 0$) when k_2 is employed to finish transportation task $D_{p_0 i_0}$

Accept the change and

update y_{k_1}, y_{k_2} ($y_{p_0 i_0}^{k_1} = 1 \rightarrow y_{p_0 i_0}^{k_1} = 0, y_{p_0 i_0}^{k_2} = 0 \rightarrow y_{p_0 i_0}^{k_2} = 1$) $Z_2^{\min} = Z_2^{\min} + \Delta Z_2$, and $\text{nodes}_{k_1}, \text{nodes}_{k_2}$ accordingly

End if

End for

Output the task allocation plan and the total number of service nodes ($Z_2^{\min}$) accessed by all vehicles.

Algorithm 3 (best insertion heuristic algorithm). For the logistics system in this paper, each vehicle passes by suppliers to pick up products first and then delivers goods to

TABLE 2: Results obtained for the three algorithms (TSSALSS, TSSA, and SA).

Instances	TSSALSS						TSSA						SA						
	Average	Minimum	Std	T	NI	Average	Minimum	Std	T	$ S(X^{*}) $	Gap 1	NI	Average	Minimum	Std	T	$ S(X^{*}) $	Gap 1	NI
S1	304.901	29715	4.41	1.73	30988	302.645	295.69	3.65	1.96	5	-0.74%	37414	303.769	298.92	2.46	3.12	6	-0.37%	56618
S2	456.536	449.86	3.98	1.88	30988	452.370	444.21	4.83	2.04	6	-0.91%	37414	455.791	448.72	4.04	3.16	7	-0.16%	56618
S3	602.408	585.28	8.14	5.67	72306	602.332	587.65	9.84	6.83	6	-0.01%	99770	598.319	584.15	7.82	8.89	4	-0.68%	152817
S4	879.412	848.56	18.20	5.70	72306	876.568	837.68	27.13	7.29	5	-0.32%	99770	875.906	846.73	19.89	8.61	6	-0.40%	152817
S5	501.301	493.02	5.18	3.59	51647	503.956	490.13	8.55	4.11	3	0.53%	74828	502.876	493.48	5.24	6.02	3	0.31%	114613
S6	629.063	612.58	10.24	3.43	51647	630.897	620.04	7.24	4.34	4	0.29%	74828	629.803	614.61	9.64	6.14	3	0.12%	114613
S7	934.877	902.45	19.29	4.05	92965	929.226	893.19	24.96	7.62	5	-0.60%	149655	936.248	906.73	20.39	13.16	2	0.15%	235956
S8	1239.98	1192.3	27.04	4.47	92965	1244.04	1217.4	19.98	7.31	3	0.33%	149655	1247.59	1216.6	20.09	13.82	3	0.61%	235956
M1	1788.32	1748.3	33.93	9.44	206589	1790.89	1768.9	12.69	19.35	1	0.14%	374138	1789.84	1760.8	15.76	34.67	2	0.08%	602942
M2	3056.33	2961.1	46.96	10.38	206589	3099.02	3067.3	20.22	19.58	0	1.40%	374138	3100.79	3047.8	36.31	36.12	1	1.45%	619767
M3	3362.66	3316.6	34.38	23.01	413178	3423.01	3371.1	32.18	47.55	0	1.79%	872990	3403.08	3351.0	37.11	83.74	1	1.20%	1446124
M4	4304.45	4244.3	42.61	24.32	413178	4400.65	4343.8	43.99	46.28	0	2.23%	872990	4393.82	4311.3	51.87	84.20	0	2.08%	1459131
M5	6668.53	6596.3	48.92	73.86	1239535	6757.88	6669.6	55.29	134.65	0	1.34%	2244830	6803.68	6673.4	80.34	233.27	0	2.03%	3810673
M6	10201.1	10107	60.24	76.35	1239535	10415.9	10338	86.07	135.77	0	2.11%	2244830	10426.8	10302	93.99	239.09	0	2.21%	3796910
M7	5933.78	5880.7	59.37	74.02	1239535	6025.07	5928.7	60.39	127.23	1	1.54%	2244830	6034.00	5941.9	62.11	226.57	0	1.69%	3796910
M8	9039.82	8905.8	84.22	79.19	1239535	9192.46	9040.1	122.74	137.51	0	1.69%	2244830	9212.56	9086.4	100.60	256.14	0	1.91%	3860891
M9	10582.9	10396	132.05	160.04	2582364	10790.7	10561	173.01	320.22	0	1.96%	4988511	10805.6	10536	222.69	570.52	0	2.10%	8804100
M10	13712.1	13523	137.71	184.77	2788953	14000.7	13810	151.68	395.58	0	2.10%	5237937	14005.5	13834	133.30	672.36	0	2.14%	9244305
L1	9829.9	9766.5	58.41	83.20	1342829	9972.3	9813.0	87.24	151.05	0	1.45%	2494256	10033.1	9917.0	88.39	275.78	0	2.07%	4327041
L2	18127.8	17930	119.90	203.29	2995542	18621.3	18342	186.38	411.13	0	2.72%	5487363	18809.6	18477	273.08	728.70	0	3.76%	9785545
L3	25307.3	25023	170.72	358.27	3098836	26132.2	25663	283.87	722.54	0	3.26%	5612075	26134.0	25733	209.27	1103.96	0	3.27%	10100378
L4	44704.2	44276	319.65	1532.21	8712249	46402.4	45235	730.92	3212.35	0	3.80%	16941776	46353.9	45637	503.28	5932.82	0	3.69%	30370067
L5	20877.3	20651	143.83	224.84	3098836	21468.2	21039	308.38	423.63	0	2.83%	5612075	21518.3	21356	273.78	743.19	0	3.07%	10100378
L6	39325.9	38912	251.02	824.93	8712249	40496.4	40076	296.43	1564.54	0	2.98%	16941776	40566.8	40233	219.51	2688.31	0	3.16%	30370067
L7	38951.6	38327	389.55	1675.24	10759079	40706.9	39124	1266.30	3973.80	0	4.51%	19527397	40560.1	39646	495.65	7432.02	0	4.13%	34207264
L8	77932.8	77134	472.58	5708.22	29953307				>10000			56492589				>10000			96608294
L9	21374.5	21195	144.58	259.16	3615309	21933.1	21573	271.82	488.21	0	2.61%	6235639	21954.4	21373	365.09	882.13	1	2.71%	11222642
L10	39113.6	38850	210.16	867.88	9801280	40401.4	39762	430.52	1631.88	0	3.29%	18244990	40463.4	39831	354.59	2982.57	0	3.45%	32706226
L11	54512.3	54081	373.65	1787.21	11680795	56645.6	54883	1122.00	4209.09	0	3.91%	20733929	57025.3	56219	617.77	8397.48	0	4.61%	35694937
L12	110389.7	109834	765.05	6045.78	34276626				>10000			71917382				>10000			121540613

TABLE 3: Comparison of SA-GS, TS, GA, and TSSALSS.

Instances	TSSALSS			SA-GS			TS			GA		
	Average	Minimum	T	Average	Minimum	T	Average	Minimum	T	Average	Minimum	T
S1	304.901	297.15	1.73	302.733	296.80	9.85	306.544	300.18	5.92	369.470	359.03	23.43
S2	456.536	449.86	1.88	453.692	447.84	10.79	457.090	446.83	6.43	550.552	539.90	32.75
S3	602.408	585.28	5.67	600.790	586.33	39.02	599.563	588.24	19.95	728.918	712.55	109.65
S4	879.412	848.56	5.70	880.433	850.10	36.73	883.270	853.16	18.28	1066.26	1032.8	123.67
S5	501.301	493.02	3.59	500.094	494.37	21.69	500.032	492.16	11.60	615.338	606.99	92.34
S6	629.063	612.58	3.43	624.186	610.55	20.38	629.761	614.59	12.61	773.413	753.07	91.45
S7	934.877	902.45	4.05	930.638	901.27	30.76	936.710	909.08	16.67	1146.19	1113.0	124.53
S8	1239.98	1192.3	4.47	1236.74	1207.8	37.09	1248.65	1200.4	15.26	1518.47	1480.3	138.05
M1	1788.32	1748.3	9.44	1789.61	1747.5	89.83	1803.16	1763.0	49.87	2232.40	2212.0	384.95
M2	3056.33	2961.1	10.38	3045.58	2964.2	83.74	3082.92	2986.6	45.02	3811.97	3737.0	429.05
M3	3362.66	3316.6	23.01	3379.41	3317.3	203.34	3388.22	3348.8	82.99	4232.26	4152.9	948.38
M4	4304.45	4244.3	24.32	4304.95	4237.4	219.50	4347.06	4280.8	95.04	5446.28	5371.0	1010.25
M5	6668.53	6596.3	73.86	6670.79	6600.3	718.94	6675.87	6662.9	282.29	8420.00	8349.1	3415.91
M6	10201.1	10107	76.35	10196.3	10116	711.86	10384.7	10226	439.45	12848.1	12751	3794.89
M7	5933.78	5880.7	74.02	5947.83	5890.1	704.68	5986.00	5940.1	297.43	7576.72	7479.4	3596.00
M8	9039.82	8905.8	79.19	9038.74	8905.3	731.35	9198.02	9065.2	359.17	11483.2	11435	4117.48
M9	10582.9	10396	160.04	10593.4	10403	1319.08	10826.3	10651	735.17	14028.2	13371	9126.51
M10	13712.1	13523	184.77	13717.8	13531	1503.93	14096.0	13926	888.05			>10000
L1	9829.93	9766.5	83.20	9837.62	9774.0	788.55	10019.6	9951.1	378.80	12693.8	12626	4488.83
L2	18127.8	17930	203.29	18139.5	17934	1737.07	18780.4	18558	897.94			>10000
L3	25307.3	25023	358.27	25316.2	25091	3198.06	26261.4	26026	1829.06			>10000
L4	44704.2	44276	1532.21			>10000	46890.2	46547	8276.54			>10000
L5	20877.3	20651	224.84	20868.0	20635	1968.49	21526.6	21310	1159.61			>10000
L6	39325.9	38912	824.93	39335.6	38934	7034.29	40950.1	40387	3759.47			>10000
L7	38951.6	38327	1675.24			>10000	40809.6	40309	8570.97			>10000
L8	77932.8	77134	5708.22			>10000			>10000			>10000
L9	21374.5	21195	259.16	21387.4	21293	2217.22	22163.2	21998	1231.80			>10000
L10	39113.6	38850	867.88	39141.1	38904	7419.42	40431.7	40435	4159.30			>10000
L11	54512.3	54081	1787.21			>10000			>10000			>10000
L12	110389.7	109834	6045.78			>10000			>10000			>10000

plants. Considering this point, we modify the best insertion heuristic algorithm in [40] to ensure that (7)-(8) must be satisfied when inserting nodes. The pickup and delivery routes for all vehicles are represented by the sets $\text{pickup_route} = \{\text{PR}_1, \text{PR}_2, \dots, \text{PR}_s\}$ and $\text{delivery_route} = \{\text{DR}_1, \text{DR}_2, \dots, \text{DR}_s\}$, in which PR_k and DR_k indicate vehicle k 's routes. The best insertion heuristic algorithm is designed as follows:

Initialize $\text{PR}_k \in \text{pickup_route}$ and $\text{DR}_k \in \text{delivery_route}$ as empty.

For $k = 1 : s$ //construct the k th vehicle's path

Select $C_{p_0 i_0} = \max(C_{p_i}), p, i \in \text{nodes}_k, p \in P, i \in S$. Insert i_0 into PR_k and p_0 into DR_k and put i_0, p_0 into tabu list tabu_k .

Do while $|\text{nodes}_k| > |\text{tabu}_k|$

Randomly select node $d \in \text{nodes}_k \setminus \text{tabu}_k$
($d \in \text{nodes}_k, d \notin \text{tabu}_k$)

If $d \leq n$ (n is the total number of suppliers)

Insert d into PR_k , The insertion position of d is the place where the least increase in transportation cost is achieved

Else

Insert d into DR_k .

End if

Put d into tabu_k

End do

End for

Output satisfactory solution X_0 .

3.2. The Second Stage of Algorithm. For the SA in this paper, four neighbors are first designed, and the search scope is limited to a specific range by requiring that the total number

TABLE 4: Vehicle routes of Mode 1.

Vehicles	Routes of each vehicle
1	O-G-B-F-N-Q-P-O
2	Q-J-Q
3	M-D-C-P-M
4	O-J-E-M-O
5	Q-H-Q
6	N-K-A-F-O-M-P-Q-N
7	P-H-E-P
8	O-H-O
9	M-J-M
10	N-A-F-Q-N
11	N-H-N
12	M-I-L-P-Q-N-O-M
13	M-H-M
14	M-G-A-M
15	P-J-P
16	Q-J-L-E-N-Q
17	P-D-C-N-Q-P
18	M-B-F-O-M

TABLE 5: Task allocation plan of Mode 1.

Plants	Suppliers											
	A	B	C	D	E	F	G	H	I	J	K	L
M	14*	18	3	0	4	18	14	13	12	9	6	12
N	10	0	0	17	16	1	1	11	12	16	6	16
O	6	18	3	3	0	6	1	8	12	4	6	0
P	6	1	17	17	7	6	1	7	12	15	6	0
Q	10	1	17	17	16	10	1	5	12	2	6	12

*14 indicates that task D_{MA} is performed by vehicle 14. If D_{MA} is zero, * will be zero.

of vehicle service nodes ($\sum_{k \in V} |\text{nodes}_k|$) be not greater than $\gamma \cdot Z_2^{\min}$. The four neighbors are performed at probabilities P_1 , P_2 , P_3 , and P_4 , respectively. A detailed description of the four neighbors with limited search scope designed in this paper is given below, and instances of the four neighbors are shown in Figure 2.

Algorithm 4 (the four neighbor structures with limited search scope). Note that only Neighbors 3 and 4 have the possibility to generate new solutions whose total number of vehicle service nodes (Z_2) will be greater than $\gamma \cdot Z_2^{\min}$. Neighbors 1 and 2 will not change the total number of vehicle service nodes because Neighbors 1 and 2 only change the vehicle routes and do not change the task allocation plan.

If rand < P_1 //Neighbor 1: block reverse mutation between suppliers (i.e., 2-opt)

Randomly choose $PR_k \in \text{pickup_route}$

If nodes in PR_k are more than two

Randomly select two positions in PR_k and reverse all the nodes between the two positions

TABLE 6: Vehicle routes of Mode 2.

Plants	Routes for each plant
M	0-B-F-0-G-A-0-J-0-E-L-I-0-C-K-0-H-0
N	0-G-I-L-J-E-D-0-H-F-A-K-0
O	0-F-B-A-K-C-D-0-J-I-G-0-H-0
P	0-F-A-B-G-I-0-E-H-0-J-0-K-C-D-0
Q	0-F-A-B-G-0-E-J-0-H-I-L-0-D-C-K-0

TABLE 7: Vehicle routes of Mode 3.

Suppliers	Routes for each supplier
A	0-M-P-0-Q-N-0-O-0
B	0-Q-P-M-0-O-0
C	0-Q-P-0-M-O-0
D	0-N-O-Q-P-0
E	0-P-N-Q-M-0
F	0-O-M-P-Q-N-0
G	0-M-P-Q-N-O-0
H	0-M-0-N-0-O-0-P-0-Q-0
I	0-M-O-N-Q-P-0
J	0-M-0-O-0-P-0-N-0-Q-0
K	0-N-Q-P-M-O-0
L	0-Q-N-M-0

Else

Give up the neighborhood operation

End if

End if

If rand < P_2 //Neighbor 2: block reverse mutation between plants

Randomly choose $DR_k \in \text{delivery_route}$

If nodes in DR_k are more than two

Randomly select two positions in DR_k and reverse all the nodes between the two positions

Else

Give up the neighborhood operation

End if

End if

If rand < P_3 //Neighbor 3: a certain transportation task for a vehicle is handed over to another one

Randomly choose a vehicle k_1 ($k_1 \in V$), one of its tasks $D_{p_0 i_0}$ ($D_{p_0 i_0} \neq 0$), and another vehicle k_2 ($k_2 \in V, k_2 \neq k_1$).

Do while the sum of $D_{p_0 i_0}$ and k_2 's transportation task exceeds the loading capacity of k_2 , or $\sum_{k \in V} |\text{nodes}_k|$ exceeds $\gamma \cdot Z_2^{\min}$ after the handing over operation //limit the search scope

TABLE 8: Results for the three transportation plans.

Transportation modes	Objective function	Total distance	Pipeline inventory cost	Number of vehicles
Mode 1	894.6	841.5	53.1	18
Mode 2	1190.1	1126.5	63.6	19
Mode 3	1156.9	1113.5	43.4	24

TABLE 9: Vehicle routes of Mode 1.

Vehicles	Routes of each vehicle
1	C-29-14-9-28-24-30-H-J-C
2	A-19-18-25-8-I-E-A
3	F-17-23-16-15-26-G-F
4	H-21-6-11-2-B-J-H
5	B-28-9-14-29-13-3-4-H-B
6	H-11-1-23-A-E-I-H
7	I-29-14-15-E-A-I
8	F-28-9-29-14-G-F
9	J-8-25-18-19-7-H-J
10	B-2-1-11-30-9-G-F-B
11	E-28-9-4-I-A-E
12	D-19-5-7-30-4-C-D
13	I-3-13-26-22-A-E-I
14	D-3-9-14-13-C-D
15	D-27-26-22-C-D
16	C-29-12-10-D-C
17	A-6-21-20-I-E-A
18	D-8-24-18-25-28-C-D
19	E-10-12-27-26-I-A-E
20	I-26-27-13-3-4-H-J-I
21	B-11-1-H-B
22	G-22-20-21-6-8-F-G
23	D-15-17-23-16-I-C-D
24	J-26-27-12-10-20-H-B-J
25	C-7-5-2-11-I-A-E-D-C
26	F-19-5-7-8-G-F
27	J-1-6-21-20-C-D-J
28	J-17-23-16-15-22-H-B-J
29	B-7-30-24-8-25-18-19-5-B
30	F-25-18-24-4-3-G-F
31	H-7-5-30-20-22-I-J-H
32	G-12-10-27-13-F-G
33	J-17-23-16-E-A-J
34	A-30-24-I-E-A

Repeat the random selection of $k_1, k_2, D_{p_0 i_0}$

End do

Hand $D_{p_0 i_0}$ over to k_2 , update $y_{pi}^{k_1} = 1 \rightarrow y_{pi}^{k_1} = 0, y_{pi}^{k_2} = 0 \rightarrow y_{pi}^{k_2} = 1$,

If $\sum_{p \in P} y_{pi}^{k_1} = 0$ //modify vehicle k_1 's pickup route

Delete node i from PR_{k_1}

End if

If $\sum_{i \in S} y_{pi}^{k_1} = 0$ //modify vehicle k_1 's delivery route

Delete node p from DR_{k_1}

End if

If $i \notin \text{nodes}_{k_2}$ before the neighborhood operation //modify vehicle k_2 's pickup route

Insert node i into any position of PR_{k_2}

End if

if $p \notin \text{nodes}_{k_2}$ before the neighborhood operation //modify vehicle k_2 's delivery route

Insert node p into any position of DR_{k_2}

End if

End if

If $P_4 < \text{rand}$ //Neighbor 4: exchange of transportation tasks among vehicles

Do while after exchanging the two tasks, either vehicle's transportation task exceeds its loading capacity, or $\sum_{k \in V} |\text{nodes}_k| \geq \gamma \cdot Z_2^{\min}$ //limit the search scope

Repeat the random selection of $k_1, k_2, D_{p_1 i_1}, D_{p_2 i_2}$

End do

Modify the two vehicles' pickup and delivery routes as in Neighbor 3

End if

Algorithm 5 (simulated annealing). The temperature decreases at an exponential rate, that is, $t_{\text{current}} = \alpha \cdot t_{\text{current}}$, where α is the cooling rate and t_{current} is the current temperature. t_{start} is the initial temperature. The algorithm terminates when it reaches the final temperature (t_{end}). The epoch length (Len) is constant.

Initialize $t_{\text{current}} = t_{\text{start}}, X_{\text{current}} = X_0$, which is obtained by the first stage of algorithm, $X_{\text{best}} = X_0, E_{\text{best}} = Z_1(X_0)$

Do while $t_{\text{current}} \geq t_{\text{end}}$

Do while $it \leq \text{Len}$

Use Algorithm 4 to generate a neighbor solution (X') of X_{current}

If $Z_1(X') \leq Z_1(X_{\text{current}})$ // $\exp((Z_1(X_{\text{current}}) - Z_1(X'))/t_{\text{current}}) < \text{rand}$

TABLE 10: Task allocation plan of Mode 1.

Plants	Suppliers																													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A	6	25	13	11	25	17	25	2	11	19	6	19	13	7	7	33	33	2	2	17	17	13	33	34	2	13	19	11	7	34
B	21	10	5	5	29	4	29	29	5	24	21	24	5	5	28	28	28	29	29	24	4	28	28	29	29	24	24	5	5	29
C	27	25	14	12	12	27	12	18	14	16	25	16	14	14	23	23	23	18	12	27	27	15	23	18	18	15	15	1	16	12
D	27	25	14	12	12	27	12	18	14	16	25	16	14	14	23	23	23	18	12	27	27	15	23	18	18	15	15	18	16	12
E	6	25	13	11	25	17	25	2	11	19	6	19	13	7	7	33	33	2	2	17	17	13	33	34	2	13	19	11	7	34
F	10	10	30	30	26	22	26	22	8	32	10	32	32	8	3	3	3	30	26	22	22	22	3	30	30	3	32	8	8	10
G	10	10	30	30	26	22	26	26	10	32	10	32	32	8	3	3	3	30	26	22	22	22	3	30	30	3	32	8	8	10
H	21	4	20	20	31	4	9	9	1	24	21	24	20	5	28	28	28	9	9	31	4	31	6	1	9	20	24	1	5	1
I	6	25	13	11	31	17	31	2	11	19	6	19	13	7	7	23	23	2	2	17	17	31	6	34	2	19	20	11	7	34
J	27	4	20	20	31	4	9	9	1	24	4	24	20	1	28	33	33	9	9	31	4	31	33	1	9	20	20	1	1	31

TABLE 11: Vehicle routes of Mode 2.

Plants	Routes for each plant
A	A-20-24-19-18-25-8-28-9-4-22-A-23-17-1-11-2-5-7-30-6-21-A-26-3-13-14-29-12-10-27-15-16-A
B	B-1-17-23-16-15-22-20-21-6-30-B-2-11-B-26-27-10-12-29-14-9-25-18-B-7-24-4-3-13-28-8-19-5-B
C	C-13-14-29-12-10-27-C-4-30-7-5-2-11-1-6-21-20-C-26-15-17-23-16-22-C-24-19-18-25-8-28-9-3-C
D	D-15-16-23-17-6-21-20-22-26-D-10-D-13-3-25-18-19-5-7-2-1-11-30-D-27-4-24-8-28-9-14-29-12-D
E	E-22-3-9-28-8-25-18-19-24-4-20-E-23-E-15-27-10-12-29-14-13-26-E-16-21-6-30-7-5-2-11-1-17-E
F	F-28-9-13-27-10-12-29-14-F-25-18-19-8-F-24-30-7-5-2-11-1-6-21-20-F-3-26-15-17-23-16-22-4-F
G	G-29-14-G-25-18-5-7-19-8-28-9-G-24-30-2-11-1-6-21-20-4-3-G-12-10-27-15-17-23-16-22-26-13-G
H	H-1-11-2-7-5-19-18-25-8-24-30-H-27-10-12-29-14-13-3-9-28-4-H-17-23-16-15-26-22-20-21-6-H
I	I-16-23-17-1-11-2-5-7-30-6-I-26-27-10-12-29-14-13-3-9-28-I-15-I-22-4-8-25-18-19-24-21-20-I
J	J-22-3-13-29-14-9-28-4-21-6-J-20-26-27-12-10-15-16-23-17-J-11-30-24-8-25-18-19-5-7-2-1-J

Accept X' as the current solution
 $X_{\text{current}} = X'$
 If $Z_1(X') \leq Z_1(X_{\text{best}})$
 $X_{\text{best}} = X', E_{\text{best}} = Z_1(X')$
 End if
 End if
 $it = it + 1$
 End do
 $t_{\text{current}} = \alpha \cdot t_{\text{current}}$
 End do

4. Computational Experiments

In this section, thirty numerical examples are designed to test the performance of the proposed algorithms. We compared our algorithms with the standard simulated annealing (SA) and two-stage simulated annealing (TSSA) to demonstrate the effectiveness of the improved algorithm. The same initial solutions are set for the three algorithms. We apply this benchmark method in order to compare the three algorithms fairly. We can not only obtain the gaps among the three algorithms, but also the number of cases for which TSSALSS performs worse than TSSA and SA with the same initial solutions. In order to verify the robustness and validity of

the proposed algorithms, we test our algorithms for three groups of instances: small-scale problems with 3 to 5 plants and 10 to 20 suppliers, medium-scale problems with 10 to 20 plants and 20 to 50 suppliers, and large-scale problems with 10 to 30 plants and 100 to 300 suppliers. Simultaneously, it is obvious to observe the trend of gaps among the three algorithms with the scales of instances increasing. Furthermore, in order to test whether the proposed algorithms can adapt to numerical examples with different characteristics, the data are generated by the following methods: the coordinates of suppliers and plants are drawn from uniform distributions and the ranges of these distributions increase with the scales of instances increasing; some of the transportation tasks D_{pi} are set to be zeros and the nonzero tasks D_{pi} are drawn from the uniform distributions from 5 to 15 or from 5 to 30; v_{pi} (pipeline inventory cost per unit time per unit of product D_{pi}) are all drawn from the same uniform distribution from 0 to 0.1; the loading or unloading time per unit of product (β) is also drawn from the uniform distribution from 0.015 to 0.025. The detailed setting of data and parameters in the algorithms are listed in Appendix A. The algorithms were programmed in MATLAB 2012b. All of the experiments were carried out on a Lenovo T4900v i5 3470 computer with 4 GB (3.47 available) RAM and a 3.20 GHz CPU. Computation times are in seconds.

We first generated ten initial feasible task allocation plans using Algorithm 1 and determined ten initial feasible

TABLE 12: Vehicle routes of Mode 3.

Suppliers	Routes for each supplier
1	1-J-A-E-I-D-C-G-F-H-B-1
2	2-H-J-A-E-I-D-C-G-F-2-B-2
3	3-G-F-B-H-J-A-E-I-D-C-3
4	4-H-B-J-A-E-I-D-C-G-F-4
5	5-B-5-H-J-A-E-I-D-C-G-F-5
6	6-J-B-H-6-F-G-C-D-E-A-I-6
7	7-F-G-C-D-I-E-A-J-7-H-B-7
8	8-F-G-8-B-H-J-A-E-I-D-C-8
9	9-C-D-I-E-A-J-H-B-F-9-G-9
10	10-D-10-C-I-E-A-J-H-B-F-G-10
11	11-H-J-A-E-I-D-C-G-F-B-11
12	12-D-C-I-E-A-J-H-B-F-G-12
13	13-C-D-I-E-A-J-H-B-F-G-13
14	14-G-C-14-D-I-E-A-J-H-B-F-14
15	15-E-A-I-J-H-B-F-G-C-D-15
16	16-E-A-J-H-B-F-G-C-D-I-16
17	17-A-E-I-D-C-G-F-B-H-J-17
18	18-F-G-C-D-I-E-A-J-H-B-18
19	19-F-G-C-D-I-E-A-J-H-B-19
20	20-I-A-E-D-C-G-F-B-J-H-20
21	21-H-B-J-21-F-G-C-D-E-A-I-21
22	22-D-C-G-F-B-H-J-A-E-I-22
23	23-E-I-D-C-G-F-B-H-J-A-23
24	24-I-A-E-D-C-G-F-24-B-J-H-24
25	25-B-H-J-A-E-I-D-C-G-F-25
26	26-I-E-A-J-H-B-F-G-C-26-D-26
27	27-C-27-D-G-F-B-H-J-A-E-I-27
28	28-H-B-J-A-E-I-D-C-G-28-F-28
29	29-D-29-C-I-E-A-J-H-B-F-G-29
30	30-H-B-J-A-E-I-D-C-G-F-30

solutions $X^0 = \{X_1^0, X_2^0, \dots, X_{10}^0\}$ by randomly sequencing the service nodes for each vehicle. Then, ten initial satisfactory solutions $X = \{X_1, X_2, \dots, X_{10}\}$ were obtained using Algorithms 2 and 3 based on the same ten initial feasible task allocation plans. The results are presented in Table 1, where $\bar{Z}_2$ indicates the average number of nodes that all the vehicles should serve in the ten initial feasible solutions $X^0 = \{X_1^0, X_2^0, \dots, X_{10}^0\}$, $\bar{Z}_2^{\min}$ indicates the average number of nodes that all the vehicles should serve in ten initial satisfactory solutions $X = \{X_1, X_2, \dots, X_{10}\}$, $\bar{Z}_1(X)$ is the average value of the objective function, $\bar{T}$ represents the average CPU time for the first stage of algorithm, and M indicates the total number of iterations. From Table 1, it can be seen that significant improvements in the value of the objective function were obtained via the first stage of algorithm within a short time.

Table 2 presents the results of the following three algorithms: (1) two-stage simulated annealing algorithm with limited search scope (TSSALSS), (2) two-stage simulated annealing algorithm (TSSA), and (3) simulated annealing (SA) algorithm. We utilized SA to improve the initial feasible

solutions $X^0 = \{X_1^0, X_2^0, \dots, X_{10}^0\}$ and obtained $X^{3*} = \{X_1^{3*}, X_2^{3*}, \dots, X_{10}^{3*}\}$. TSSALSS and TSSA were employed to generate $X^{1*} = \{X_1^{1*}, X_2^{1*}, \dots, X_{10}^{1*}\}$ and $X^{2*} = \{X_1^{2*}, X_2^{2*}, \dots, X_{10}^{2*}\}$ based on the initial satisfactory solutions $X = \{X_1, X_2, \dots, X_{10}\}$, which were obtained via the first stage of algorithm from the initial feasible task allocation plans. $S(X^{2*})$ is defined as follows: $S(X^{2*}) = \{X_i^{2*} \mid Z_1(X_i^{2*}) < Z_1(X_i^{1*}), X_i^{1*} \in X^{1*}\}$, where $Z_1(X)$ indicates the objective function value of solution X . $S(X^{3*})$ is similarly defined. Column $|S(X^{2*})|(|S(X^{3*})|)$ is the number of solutions in $S(X^{2*})(S(X^{3*}))$, which, respectively, represents the number of cases for which TSSALSS performs worse than TSSA and SA. Column gap 1 (gap 2) is the gap between average $Z_1(X_i^{1*})$ and average $Z_1(X_i^{2*})(Z_1(X_i^{3*}))$. The columns “average,” “minimum,” “std,” “T,” and “NI,” respectively, represent the objective function value of the best solution among the ten trials, the average objective function value of the ten trials, the standard deviation of the ten solutions, the average time spent by the two algorithms, and the number of iterations performed by the two algorithms.

From the computational results above, we can conclude the following:

- (1) TSSA is much faster than SA but has no obvious advantage in terms of the quality of the solution. This is because the initial temperature of TSSA is lower, and its total number of iterations is less.
- (2) Although TSSALSS has no significant advantage over TSSA in the small-scale numerical examples, it generates higher-quality solutions within shorter time frames in both the medium-scale and large-scale numerical examples. Moreover, in Figure 3, it can be seen that the solution obtained using TSSALSS tends to improve in quality as the scale increases.
- (3) The convergence rate of these three algorithms decreases in the order of TSSALSS, TSSA, and SA, which is shown in Figure 4. Figure 4 is the computational result of L9 using the three algorithms. Note that the abscissa axis is the cooling times rather than the total number of iterations. The epoch length of TSSALSS, TSSA, and SA at each temperature is 35000, 50000, and 50000, respectively (see Appendix A, row L9).

In addition, we compared our algorithm (TSSALSS) with three common metaheuristic algorithms from the literature: simulated annealing with greedy search (SA-GS) [38], genetic algorithm (GA) [30], and tabu search (TS) [29]. The neighborhood structures of SA-GS and TS are the same as those proposed in the Section 3.2. The crossover operators and mutation operators of GA are newly designed for the proposed problem. The mutation operators are the same as the neighborhood structures in Section 3.2. The crossover operators include the task allocation crossover and the routes crossover. The task allocation crossover will generate illegal solutions; thus we legalize these illegal solutions once they occur. The routes' crossover is similar to that of the traveling

TABLE 13: Results for the three transportation plans.

Transportation modes	Objective function	Total distance	Pipeline inventory cost	Number of vehicles
Mode 1	3036.2	2182.6	853.6	34
Mode 2	3285.6	2481.7	803.9	37
Mode 3	4082.7	5056.4	973.7	44

FIGURE 3: Trends in the gaps.

salesman problem (TSP) [41]. These three algorithms were all executed ten times in the same running condition as that of TSSALSS. The results obtained are listed in Table 3, where the columns “average,” “minimum,” and “T” indicate the objective function value of the best solution among the ten trials, average objective function value of the ten trials, and average time spent by these algorithms.

As the table shows, TSSALSS spends the least amount of time. Although the quality of the solutions obtained by TSSALSS is not particularly high compared with that of the other two algorithms (SA-GS and TS), especially SA-GS, when the scale of the problems is relatively small, the advantage of TSSALSS becomes evident as the scale of the instances increases. GA is less efficient and less precise compared with the other three algorithms because the crossover operators generate many illegal solutions and it spends much time to tackle these illegal solutions.

5. Case Study

5.1. Case One. The first case study in this paper focuses on the inbound logistics routing problem for a feed factory in Shandong, China. This feed factory has five plants and twelve suppliers nearby. The location of every plant and supplier, tasks, and pipeline inventory costs per unit time are given in

FIGURE 4: Computational result for L9 using the three algorithms.

Appendix B. We optimized the following three transportation modes.

- (1) Mode 1: Milk-run among all plants and suppliers, that is, n - n .
- (2) Mode 2: Milk-run among all suppliers for each plant, that is, n -1.
- (3) Mode 3: Milk-run among all plants for each supplier, that is, 1- n .

The routes and task allocation plans of the three models are given in Tables 4–7.

Table 8 presents the results for the three transportation plans. For the total objective function, Mode 1 is 24.8% less than Mode 2 and 22.7% less than Mode 3. For traveling distance, Mode 1 is 25.3% less than Mode 2 and 24.4% less than Mode 3. Although the pipeline inventory cost of Mode 3 is only 18.3% lower than that of Mode 1, the number of vehicles employed in Mode 3 is 33.3% more than that of Mode 1. From the discussions above, it is obvious that Mode 1 holds significant advantages for logistics systems comprising multiple suppliers and plants.

5.2. Case Two. In order to demonstrate the validity of the many-to-many Milk-run mode in the case of large scale, we use the computer to generate an n - n problem randomly, in which the case has ten plants and thirty suppliers. The location of every plant and supplier, volumes of tasks, and pipeline

TABLE 14

Instance	m	n	s	Coordinate	D_{pi}	v_{pi}	M	P_1	P_2	P_3	P_4	t_0	t_e	Len	γ	t_0	t_e	Len	SA	Len
S1	3	10	4	$U[0, 30]$	$U[5, 15] \setminus 30$	$U[0, 0.1]$	7000	0.37	0.13	0.5	0.2	30	0.15	300	1.2	30	0.05	300	0.05	300
S2	3	10	6	$U[0, 30]$	$U[5, 30] \setminus 30$		7000	0.37	0.13	0.5	0.2	30	0.15	300	1.2	30	0.05	300	0.05	300
S3	3	20	7	$U[0, 30]$	$U[5, 15] \setminus 60$		25000	0.43	0.07	0.5	0.2	30	0.15	600	1.2	30	0.05	800	0.05	800
S4	3	20	13	$U[0, 30]$	$U[5, 30] \setminus 60$		25000	0.43	0.07	0.5	0.2	30	0.15	600	1.2	30	0.05	800	0.05	800
S5	5	10	6	$U[0, 30]$	$U[5, 15] \setminus 50$		20000	0.33	0.17	0.5	0.2	30	0.15	400	1.2	30	0.05	600	0.05	600
S6	5	10	10	$U[0, 30]$	$U[5, 30] \setminus 50$		20000	0.33	0.17	0.5	0.2	30	0.15	400	1.2	30	0.05	600	0.05	600
S7	5	20	11	$U[0, 30]$	$U[5, 15] \setminus 100$		60000	0.4	0.1	0.5	0.2	30	0.15	800	1.19	30	0.05	1200	0.05	1200
S8	5	20	19	$U[0, 30]$	$U[5, 30] \setminus 100$		60000	0.4	0.1	0.5	0.2	30	0.15	800	1.19	30	0.05	1200	0.05	1200
M1	10	20	22	$U[0, 35]$	$U[5, 15] \setminus 200$	$U[0, 0.1]$	200000	0.33	0.17	0.5	0.2	30	0.15	2000	1.18	30	0.05	3000	0.05	3000
M2	10	20	40	$U[0, 35]$	$U[5, 30] \setminus 200$		200000	0.33	0.17	0.5	0.2	30	0.15	2000	1.18	30	0.05	3000	0.05	3000
M3	10	30	34	$U[0, 35]$	$U[5, 15] \setminus 300$		500000	0.37	0.13	0.5	0.2	30	0.15	4000	1.17	30	0.05	7000	0.05	7000
M4	10	30	59	$U[0, 35]$	$U[5, 30] \setminus 300$		500000	0.37	0.13	0.5	0.2	30	0.15	4000	1.17	30	0.05	7000	0.05	7000
M5	20	30	67	$U[0, 40]$	$U[5, 15] \setminus 600$		1500000	0.3	0.20	0.5	0.2	30	0.15	12000	1.15	30	0.05	18000	0.05	18000
M6	20	30	116	$U[0, 40]$	$U[5, 30] \setminus 600$		1500000	0.3	0.20	0.5	0.2	30	0.15	12000	1.15	30	0.05	18000	0.05	18000
M7	20	50	57	$U[0, 40]$	$U[5, 15] \setminus 500$		1000000	0.36	0.14	0.5	0.2	30	0.15	12000	1.15	30	0.05	18000	0.05	18000
M8	20	50	97	$U[0, 40]$	$U[5, 30] \setminus 500$		1000000	0.36	0.14	0.5	0.2	30	0.15	12000	1.15	30	0.05	18000	0.05	18000
M9	20	50	112	$U[0, 40]$	$U[5, 15] \setminus 1000$		3000000	0.36	0.14	0.5	0.2	30	0.15	25000	1.14	30	0.05	40000	0.05	40000
M10	20	50	197	$U[0, 40]$	$U[5, 30] \setminus 1000$		3000000	0.36	0.14	0.5	0.2	30	0.15	27000	1.14	30	0.05	42000	0.05	42000
L1	10	100	100	$U[0, 50]$	$U[5, 30] \setminus 500$	$U[0, 0.1]$	1000000	0.45	0.05	0.5	0.2	30	0.15	13000	1.14	30	0.05	20000	0.05	20000
L2	10	100	196	$U[0, 50]$	$U[5, 30] \setminus 1000$		3000000	0.45	0.05	0.5	0.2	30	0.15	29000	1.13	30	0.05	44000	0.05	44000
L3	10	200	193	$U[0, 60]$	$U[5, 30] \setminus 1000$		3000000	0.48	0.02	0.5	0.2	30	0.15	30000	1.13	30	0.05	45000	0.05	45000
L4	10	200	394	$U[0, 60]$	$U[5, 30] \setminus 2000$		9000000	0.48	0.02	0.5	0.2	40	0.15	80000	1.13	40	0.05	130000	0.05	130000
L5	20	100	199	$U[0, 50]$	$U[5, 30] \setminus 1000$		3000000	0.42	0.08	0.5	0.2	30	0.15	30000	1.13	30	0.05	45000	0.05	45000
L6	20	100	388	$U[0, 50]$	$U[5, 30] \setminus 2000$		9000000	0.42	0.08	0.5	0.2	40	0.15	80000	1.13	40	0.05	130000	0.05	130000
L7	20	200	397	$U[0, 50]$	$U[5, 30] \setminus 2000$		9000000	0.45	0.05	0.5	0.2	50	0.15	95000	1.13	50	0.05	145000	0.05	145000
L8	20	200	784	$U[0, 50]$	$U[5, 30] \setminus 4000$		30000000	0.45	0.05	0.5	0.2	70	0.15	250000	1.13	70	0.05	400000	0.05	400000
L9	30	100	196	$U[0, 50]$	$U[5, 30] \setminus 1000$		3000000	0.38	0.12	0.5	0.2	30	0.15	35000	1.13	30	0.05	50000	0.05	50000
L10	30	100	387	$U[0, 50]$	$U[5, 30] \setminus 2000$		9000000	0.38	0.12	0.5	0.2	40	0.15	90000	1.13	40	0.05	140000	0.05	140000
L11	30	200	387	$U[0, 70]$	$U[5, 30] \setminus 2000$		9000000	0.43	0.07	0.5	0.2	60	0.15	100000	1.13	60	0.05	150000	0.05	150000
L12	30	200	776	$U[0, 70]$	$U[5, 30] \setminus 4000$		30000000	0.43	0.07	0.5	0.2	80	0.15	280000	1.13	80	0.05	500000	0.05	500000

n : number of suppliers; m : number of plants; s : number of vehicles; coordinate $U(a, b)$: coordinates of all the suppliers and plants are drawn from a uniform distribution between a and b ; $U[a, b] \setminus c$: the volumes of all the nonzero tasks are drawn from $U[a, b]$, where c is the number of non-zero-volume tasks. M is the number of iterations in Algorithm 2. Besides, the cooling coefficients α are all equal to 0.95, and the coefficients β are drawn from $[0.015, 0.025]$.

TABLE 15: Data of the demands D_{pi} and v_{pi} (pipeline inventory cost per unit time per unit of product D_{pi}) for each supplier and plant.

Plants	Suppliers																							
	A		B		C		D		E		F		G		H		I		J		K		L	
	D_{pA}	v_{pA}	D_{pB}	v_{pB}	D_{pC}	v_{pC}	D_{pD}	v_{pD}	D_{pE}	v_{pE}	D_{pF}	v_{pF}	D_{pG}	v_{pG}	D_{pH}	v_{pH}	D_{pI}	v_{pI}	D_{pJ}	v_{pJ}	D_{pK}	v_{pK}	D_{pL}	v_{pL}
M	0.75	5	0.66	1	0.63	7	0	5	0.2	3	0.27	9	0.2	2	0.71	3	0.1	3	0.95	1	0.05	8	0.4	1
N	0.23	5	0	1	0	7	0.05	5	0.05	3	0.13	9	0.07	2	0.63	3	0.03	3	0.68	1	0.01	8	0.05	3
O	0.3	5	0.07	1	0.17	7	0.07	5	0	3	0.07	9	0.1	2	0.88	3	0.06	3	0.73	1	0.03	8	0	2
P	0.17	5	0.23	1	0.3	7	0.1	5	0.13	3	0.17	9	0.22	2	0.65	3	0.19	3	0.98	1	0.05	8	0	2
Q	0.66	5	0.1	1	0.33	7	0.08	5	0.06	3	0.11	9	0.13	2	0.76	3	0.09	3	0.94	1	0.01	8	0.1	5

Suppliers: A(38, 45) supplier of bean pulp; B(40, 37) cottonseed meal; C(12, 47) distiller's grains; D(7, 33) peanut meal; E(24, 18) corn gluten meal; F(34, 37) fermented soybean meal; G(48, 29) rice bran meal; H(28, 20) flour; I(33, 6) wheat; J(18, 13) corn; K(23, 53) whey powder; and L(24, 9) amino acids, calcium hydrogen phosphate, protein powder, and so forth.

Plants: M(25, 25); N(16, 33); O(28, 29); P(11, 22); and Q(17, 29).

TABLE 16: Coordinates of both suppliers and plants.

(a)

Coordinates	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Suppliers										28	29	30			
															15	16	17	18	19	20	21	22	23	24				25	26	27
X	13.09	2.93	17.09	15.66	0.39	16.90	4.94	7.62	12.36	29.88	10.16	26.03	17.91	16.84	32.96	31.53	34.71	2.46	4.18	19.05	17.98	24.16	32.45	8.87	0.74	25.51	26.52	11.52	19.25	9.65
Y	33.10	34.19	6.37	15.02	24.92	23.12	25.55	13.70	4.77	1.37	30.12	0.04	3.63	0.02	18.26	26.18	34.06	16.28	18.19	20.13	21.96	18.24	28.53	17.97	12.57	12.97	6.18	6.82	1.62	23.58

(b)

Coordinates	A	B	C	D	Plants						I	J
					E	F	G	H				
X	31.00	1.50	20.69	29.26	34.94	4.30	15.68	12.98	25.54	16.74		
Y	28.50	34.52	7.78	4.46	25.64	2.44	0.75	27.23	21.55	32.76		

TABLE 17: Data of the demands D_{pi} for each supplier and plant.

Plants	The demands D_{pi} for each supplier																													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A	7	8	9	10	10	14	6	15	9	15	14	12	8	7	7	8	11	7	6	12	5	8	7	13	6	9	5	12	15	10
B	9	15	11	6	14	12	12	15	10	8	14	12	11	12	14	10	13	9	6	5	5	8	12	12	13	11	10	9	15	6
C	13	7	8	6	8	8	13	15	8	8	7	7	15	10	10	10	13	12	9	13	14	6	7	14	10	14	12	6	8	8
D	5	8	7	9	12	15	15	8	10	14	6	9	12	10	6	5	12	9	11	13	9	13	15	14	8	8	9	10	15	5
E	5	13	7	5	5	10	11	12	14	11	9	10	11	15	9	15	9	7	8	5	13	9	5	13	6	14	15	13	14	8
F	5	14	9	11	13	7	14	10	15	15	7	11	7	15	15	14	11	14	5	6	14	14	6	10	7	13	15	14	5	6
G	7	14	6	11	12	14	14	13	10	5	10	10	15	14	6	6	7	6	13	7	12	15	9	7	15	10	15	14	12	12
H	15	12	12	13	7	14	12	7	7	10	10	11	5	11	7	11	6	5	6	11	15	6	12	11	8	5	7	11	12	5
I	10	11	15	15	15	8	11	5	14	14	13	6	7	8	15	6	9	10	9	14	14	9	5	14	8	8	13	8	7	8
J	8	9	5	12	14	7	12	11	10	14	5	12	6	15	9	15	9	5	7	9	12	10	8	7	10	14	10	10	10	6

The capacity of each truck is 100. β is equal to 0.02. The speed of each vehicle is equal to 40.

inventory costs per unit time are given in Appendix C. We similarly optimized the three transportation modes mentioned in Section 5.1. The routes and task allocation plans of the three modes are given in Tables 9–12. Suppliers are numbered from 1 to 30, and plants are numbered from A to J.

Table 13 presents the results for the three transportation plans. From the comparison between Mode 1 and Mode 2, although Mode 1 has a greater pipeline inventory cost (853.6) than Mode 2 does (803.9), the total cost and the total distance of Mode 1 (3036.2 and 2182.6, resp.) are both smaller than those of Mode 2 (3285.6 and 2481.7, resp.). Furthermore, Mode 1 needs three vehicles less than Mode 2 does. From the comparison between Mode 1 and Mode 3, all the four values of Mode 1 (objective function, total distance, pipeline inventory cost, and number of vehicle) are lower than those of Mode 3. From the analysis of Case Two, we can demonstrate the superiority of many-to-many transportation mode in the case of large-scale instances.

6. Conclusions

In this paper, we considered the problem of Milk-run route planning with multiple suppliers and plants and proposed and constructed a new mathematical model that minimizes transportation and pipeline inventory costs. To solve the proposed model, we developed a two-stage heuristic algorithm that comprehensively considers the characteristics of the problem. The first stage of the algorithm minimizes the total number of service nodes visited by all vehicles, while the second stage limits the search scope of SA to improve search efficiency. The results obtained from thirty small-, medium-, and large-scale numerical examples verified the efficacy of the algorithms proposed in this paper, and their performance proved to be more effective than the traditional SA. Finally, we studied the two cases and confirmed that the n - n Milk-run transportation mode is superior to other modes.

The work presented herein has the following limitations: (1) because of the complexity of the problem, the proposed

algorithm was not compared with exact algorithms; (2) in this paper, we only considered the case in which the transportation frequency is one, whereas, in practice, the frequency of transportation within a day may be performed more than once to reduce the inventory of each plant. As a consequence, we plan to extend the model to take this into account. We also plan to consider long-distance transportation and extend the model to include the cross-docking strategy. Further, we will consider the effect of using different types of vehicles to perform the transportation tasks.

Appendices

A. Data of Instances and Parameters of the Three Algorithms

See Table 14.

B. Data of Case One

See Table 15.

The capacity of each truck is one. v_{pi} is determined by the products. For convenience, we use transportation distance (using Euclidean distances), instead of transportation costs. The units of distances are all calculated in kilometers per hour. β is equal to 0.02. The speed of each vehicle is equal to 40 kilometers per hour.

C. Data of Case Two

See Tables 16, 17, and 18.

Conflict of Interests

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

TABLE 18: Data of v_{pi} : pipeline inventory cost per unit time per unit of product D_{pi} .

Plants	Suppliers																													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A	0.30	0.20	0.17	0.24	0.27	0.06	0.08	0.28	0.26	0.05	0.28	0.28	0.32	0.19	0.07	0.28	0.02	0.07	0.11	0.18	0.17	0.25	0.07	0.19	0.11	0.14	0.20	0.09	0.13	0.29
B	0.03	0.03	0.19	0.31	0.29	0.30	0.08	0.20	0.12	0.32	0.08	0.10	0.00	0.03	0.32	0.27	0.06	0.19	0.04	0.21	0.08	0.02	0.26	0.11	0.20	0.06	0.02	0.05	0.22	0.24
C	0.04	0.12	0.12	0.18	0.23	0.21	0.09	0.06	0.22	0.31	0.30	0.08	0.21	0.23	0.01	0.26	0.18	0.20	0.22	0.31	0.32	0.26	0.24	0.03	0.28	0.20	0.29	0.09	0.27	0.11
D	0.22	0.32	0.04	0.21	0.01	0.23	0.30	0.02	0.20	0.13	0.17	0.19	0.10	0.15	0.12	0.33	0.07	0.19	0.02	0.01	0.29	0.22	0.22	0.10	0.01	0.06	0.08	0.18	0.03	0.31
E	0.25	0.07	0.08	0.19	0.23	0.08	0.30	0.30	0.22	0.26	0.19	0.24	0.07	0.33	0.03	0.08	0.21	0.20	0.18	0.17	0.02	0.22	0.16	0.29	0.32	0.21	0.09	0.04	0.10	0.01
F	0.25	0.12	0.12	0.30	0.15	0.29	0.20	0.24	0.02	0.25	0.03	0.16	0.28	0.00	0.25	0.08	0.05	0.29	0.14	0.25	0.24	0.08	0.14	0.26	0.01	0.11	0.25	0.05	0.28	0.05
G	0.01	0.01	0.06	0.15	0.23	0.27	0.32	0.20	0.30	0.10	0.28	0.07	0.10	0.17	0.05	0.24	0.26	0.21	0.14	0.11	0.09	0.11	0.31	0.07	0.02	0.15	0.25	0.09	0.12	0.32
H	0.26	0.04	0.06	0.26	0.18	0.02	0.28	0.20	0.29	0.29	0.09	0.19	0.22	0.17	0.25	0.29	0.21	0.06	0.14	0.09	0.12	0.07	0.02	0.30	0.30	0.14	0.02	0.32	0.29	0.25
I	0.15	0.24	0.02	0.18	0.23	0.16	0.08	0.24	0.23	0.21	0.15	0.13	0.03	0.25	0.16	0.31	0.12	0.15	0.13	0.17	0.04	0.07	0.19	0.21	0.15	0.23	0.02	0.30	0.07	0.29
J	0.04	0.26	0.20	0.23	0.21	0.24	0.10	0.14	0.08	0.11	0.18	0.07	0.25	0.17	0.29	0.24	0.24	0.15	0.15	0.01	0.09	0.16	0.14	0.16	0.19	0.26	0.05	0.19	0.03	0.05

References

- [1] O. Berman and Q. Wang, "Inbound logistic planning: minimizing transportation and inventory cost," *Transportation Science*, vol. 40, no. 3, pp. 287–299, 2006.
- [2] S. J. Sadjadi, M. Jafari, and T. Amini, "A new mathematical modeling and a genetic algorithm search for milk run problem (an auto industry supply chain case study)," *The International Journal of Advanced Manufacturing Technology*, vol. 44, no. 1-2, pp. 194–200, 2009.
- [3] T. Nemoto, K. Hayashi, and M. Hashimoto, "Milk-run logistics by Japanese automobile manufacturers in Thailand," *Procedia—Social and Behavioral Sciences*, vol. 2, no. 3, pp. 5980–5989, 2010.
- [4] S. Raff, "Routing and scheduling of vehicles and crews. The state of the art," *Computers & Operations Research*, vol. 10, no. 2, pp. 63–211, 1983.
- [5] G. Laporte and Y. Nobert, "Exact algorithms for the vehicle routing problem," *Surveys in Combinatorial Optimization*, vol. 31, pp. 147–184, 1987.
- [6] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.
- [7] O. Bräysy and M. Gendreau, "Vehicle routing problem with time windows, Part II: metaheuristics," *Transportation Science*, vol. 39, no. 1, pp. 119–139, 2005.
- [8] J. Renaud, G. Laporte, and F. F. Boctor, "A tabu search heuristic for the multi-depot vehicle routing problem," *Computers & Operations Research*, vol. 23, no. 3, pp. 229–235, 1996.
- [9] A. Lim and F. Wang, "Multi-depot vehicle routing problem: a one-stage approach," *IEEE Transactions on Automation Science and Engineering*, vol. 2, no. 4, pp. 397–402, 2005.
- [10] K. H. Chuah and J. C. Yingling, "Routing for a just-in-time supply pickup and delivery system," *Transportation Science*, vol. 39, no. 3, pp. 328–339, 2005.
- [11] Z. Jiang, Y. Huang, and J. Wang, "Routing for the milk-run pickup system in automobile parts supply," in *Proceedings of the 6th CIRP-Sponsored International Conference on Digital Enterprise Technology*, vol. 66 of *Advances in Intelligent and Soft Computing*, pp. 1267–1275, Springer, Berlin, Germany, 2010.
- [12] M. Jafari-Eskandari, S. J. Sadjadi, M. S. Jabalameli et al., "A robust optimization approach for the milk run problem (An auto industry Supply Chain Case Study)," in *Proceedings of the International Conference on Computers & Industrial Engineering (CIE '09)*, pp. 1076–1081, IEEE, 2009.
- [13] M. Jafari-Eskandari, A. R. Aliahmadi, and G. H. H. Khaleghi, "A robust optimisation approach for the milk run problem with time windows with inventory uncertainty: an auto industry supply chain case study," *International Journal of Rapid Manufacturing*, vol. 1, no. 3, pp. 334–347, 2010.
- [14] J. Zhang and Y. Li, "Analysis and optimization of the milk-run model in automotive industry—an automobile manufacturing company case study," *Journal of Japan Industrial Management Association*, vol. 61, no. 3, pp. 214–221, 2010.
- [15] L. Yun, G. Xian-Long, and S. Chao-Chun, "Inventory-transportation integrated optimization based on milk-run model," in *Proceedings of the 1st International Conference on E-Business and E-Government (ICEE '10)*, pp. 3372–3376, IEEE, May 2010.
- [16] T. Du, F. K. Wang, and P.-Y. Lu, "A real-time vehicle-dispatching system for consolidating milk runs," *Transportation Research Part E: Logistics and Transportation Review*, vol. 43, no. 5, pp. 565–577, 2007.
- [17] J. Ma and G. Sun, "Mutation ant colony algorithm of milk-run vehicle routing problem with fastest completion time based on dynamic optimization," *Discrete Dynamics in Nature and Society*, vol. 2013, Article ID 418436, 6 pages, 2013.
- [18] G. Berbeglia, J. F. Cordeau, I. Gribkovskaia, and G. Laporte, "Static pickup and delivery problems: a classification scheme and survey," *TOP*, vol. 15, no. 1, pp. 1–31, 2007.
- [19] M. W. Savelsbergh and M. Sol, "The general pickup and delivery problem," *Transportation Science*, vol. 29, no. 1, pp. 17–29, 1995.
- [20] X. Zhu, S. J. Hu, Y. Koren, and N. Huang, "A complexity model for sequence planning in mixed-model assembly lines," *Journal of Manufacturing Systems*, vol. 31, no. 2, pp. 121–130, 2012.
- [21] Y. H. Lee, J. W. Jung, and K. M. Lee, "Vehicle routing scheduling for cross-docking in the supply chain," *Computers & Industrial Engineering*, vol. 51, no. 2, pp. 247–256, 2006.
- [22] R. Musa, J.-P. Arnaout, and H. Jung, "Ant colony optimization algorithm to solve for the transportation problem of cross-docking network," *Computers & Industrial Engineering*, vol. 59, no. 1, pp. 85–92, 2010.
- [23] Z. Miao, K. Fu, and F. Yang, "A hybrid genetic algorithm for the multiple crossdocks problem," *Mathematical Problems in Engineering*, vol. 2012, Article ID 316908, 18 pages, 2012.
- [24] S. M. Mousavi and R. Tavakkoli-Moghaddam, "A hybrid simulated annealing algorithm for location and routing scheduling problems with cross-docking in the supply chain," *Journal of Manufacturing Systems*, vol. 32, no. 2, pp. 335–347, 2013.
- [25] J. Kaneko and W. Nojiri, "The logistics of just-in-time between parts suppliers and car assemblers in Japan," *Journal of Transport Geography*, vol. 16, no. 3, pp. 155–173, 2008.
- [26] A. S. Alfa, S. S. Heragu, and M. Chen, "A 3-opt based simulated annealing algorithm for vehicle routing problems," *Computers & Industrial Engineering*, vol. 21, no. 1-4, pp. 635–639, 1991.
- [27] I. H. Osman, "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Annals of Operations Research*, vol. 41, no. 4, pp. 421–451, 1993.
- [28] A. Van Breedam, "Improvement heuristics for the vehicle routing problem based on simulated annealing," *European Journal of Operational Research*, vol. 86, no. 3, pp. 480–490, 1995.
- [29] M. Gendreau, A. Hertz, and G. Laporte, "A tabu search heuristic for the vehicle routing problem," *Management Science*, vol. 40, no. 10, pp. 1276–1290, 1994.
- [30] B. M. Baker and M. A. Ayechew, "A genetic algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 30, no. 5, pp. 787–800, 2003.
- [31] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 31, no. 12, pp. 1985–2002, 2004.
- [32] B. Bullnheimer, R. F. Hartl, and C. Strauss, *Applying the ANT System to the Vehicle Routing Problem*, Meta-Heuristics, Springer, New York, NY, USA, 1999.
- [33] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Advanced Engineering Informatics*, vol. 18, no. 1, pp. 41–48, 2004.
- [34] B. Yu, Z.-Z. Yang, and B. Yao, "An improved ant colony optimization for vehicle routing problem," *European Journal of Operational Research*, vol. 196, no. 1, pp. 171–176, 2009.
- [35] C.-H. Chen and C.-J. Ting, "An improved ant colony system algorithm for the vehicle routing problem," *Journal of the Chinese Institute of Industrial Engineers*, vol. 23, no. 2, pp. 115–126, 2006.

- [36] T. J. Ai and V. Kachitvichyanukul, "Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem," *Computers & Industrial Engineering*, vol. 56, no. 1, pp. 380–387, 2009.
- [37] Y. Marinakis, M. Marinaki, and G. Dounias, "A hybrid particle swarm optimization algorithm for the vehicle routing problem," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 4, pp. 463–472, 2010.
- [38] X. Geng, Z. Chen, W. Yang, D. Shi, and K. Zhao, "Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search," *Applied Soft Computing*, vol. 11, no. 4, pp. 3680–3689, 2011.
- [39] J. M. Varanelli and J. P. Cohoon, "A two-stage simulated annealing methodology," in *Proceedings of the 5th Great Lakes Symposium on VLSI*, pp. 50–53, Buffalo, NY, USA, March 1995.
- [40] *The Vehicle Routing Problem*, SIAM, 2001.
- [41] I. M. Oliver, D. J. Smith, and J. R. C. Holland, "Study of permutation crossover operators on the traveling salesman problem," in *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms: July 28–31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*, L. Erlbaum Associates, Hillsdale, NJ, USA, 1987.