Optimization Letters (2022) 16:333–353
https://doi.org/10.1007/s11590-020-01693-x

ORIGINAL PAPER

A variable neighborhood search for the last-mile delivery problem during major infectious disease outbreak

Li Jiang[1,2] · Xiaoning Zang[1,2] · Junfeng Dong[1,2] · Changyong Liang[1,2] · Nenad Mladenovic[3]

Received: 3 June 2020 / Accepted: 15 December 2020 / Published online: 4 January 2021

Abstract

During major infectious disease outbreak, such as COVID-19, the goods and parcels supply and distribution for the isolated personnel has become a key issue worthy of attention. In this study, we propose a delivery problem that arises in the last-mile delivery during major infectious disease outbreak. The problem is to construct a Hamiltonian tour over a subset of candidate parking nodes, and each customer is assigned to the nearest parking node on the tour to pick up goods or parcels. The aim is to minimize the total cost, including the routing, allocation, and parking costs. We propose three models to formulate the problem, which are node-based, flow-based and bilevel programing formulations. Moreover, we develop a variable neighborhood search algorithm based on the ideas from the bilevel programing formulations to solve the problem. Finally, the proposed algorithm is tested on a set of randomly generated instances, and the results indicate the effectiveness of the proposed approach.

Keywords Variable neighborhood search · Last-mile delivery · COVID-19 · Bilevel programing

1 Introduction

In 2019–2020, an infectious disease commonly known as COVID-19 has outbroken in 49 countries and regions. As of May 10, 2020, the total number of confirmed COVID-19 infections in the world has exceeded 4.3 million [1]. In the absence of

✉ Xiaoning Zang
xnzang123@126.com

[1] School of Management, Hefei University of Technology, Hefei 230009, Anhui, PR China

[2] Key Laboratory of Process Optimization and Intelligent Decision-Making, Ministry of Education, Hefei 230009, Anhui, PR China

[3] Department of Industrial and Systems Engineering, Research Center on Digital Supply Chain and Operations Management, Khalifa University, 999041 Abu Dhabi, UAE

an effective vaccine, community isolation has become a workable measure to avoid virus transmission. In China, measures such as community isolation have effectively controlled the spread of COVID-19, bringing the total number of confirmed COVID-19 infections to less than 84,500.

During community isolation, most people choose to isolate and work at home, and the goods supply and distribution for the isolated personnel has become a key issue worthy of attention. In China, most communities were isolated during the COVID-19 outbreak, and the personnel going in and out of the communities were restricted. The courier could not deliver goods or parcels to customers' homes. An effective distribution method that can ensure safe delivery and reduce delivery cost is urgently needed.

In this study, we propose a last-mile delivery problem during major infectious disease outbreak. In the problem, a vehicle departs from the depot and visits a subset of candidate parking nodes. When the vehicle arrives at a candidate parking node, the nearby customers go to the visited parking node to pick up their goods or parcels. This study focuses on delivery network optimization, and the purpose is to find a tour with minimum total cost (routing, self-pickup, and parking costs) over a subset of candidate parking nodes, in which each customer is assigned to the nearest parking node on the tour to pick up goods or parcels. We develop the node-based, flow-based, and bi-level programing formulations for the problem. We also propose a variable neighborhood search (VNS) based on the ideas from the bi-level programing formulations to solve the problem.

The rest of the paper is organized as follows. The related literature is reviewed in Sect. 2. Three models are proposed in Sect. 3 to formulate the proposed problem. VNS based on the ideas from the bi-level programing formulations is detailed in Sect. 4. Extensive computational experiments are carried out in Sect. 5 to investigate the efficiency of the proposed algorithm. Finally, conclusions are drawn in Sect. 6.

2 Literature review

Last-mile delivery during major infectious disease outbreak is a complex supply chain optimization problem [2, 3], and this problem integrates routing and allocation decisions. In an early study, Laporte et al. [4] proposed a single vehicle routing allocation problem (SVRAP) to locate a set of post boxes, in which customers are assigned to nearby post boxes. Akinc and Srikanth [5] proposed a similar SVRAP, which has an application in mobile service facility, and they developed a branch-and-bound algorithm that was tested on randomly generated problem instances with up to 100 customers to solve the problem. Beasley and Nascimento [6] surveyed SVRAP and several similar problems, such as the covering salesman problem (CSP), covering tour problem (CTP), and so on. Vogt et al. [7] proposed a unifying framework for VRAP and developed a tabu search to solve the problem. Baniasadi et al. [8] proposed a transformation technique for solving the clustered generalized traveling salesman problem (CGTSP); the objective of this problem is to find the minimal route that visits exactly one node from each subcluster to ensure that all subclusters of each cluster are visited consecutively. Moreover, Ghoniem et al. [9]

proposed a vehicle routing with demand allocation problem (VRDAP) that arises in non-profit food distribution systems. In their problem, each customer is assigned to a selected delivery site to collect his/her demand. Their aim was to construct multiple tours with minimum routing and allocation costs over a set of delivery sites. They developed a column generation approach to solve the problem. Solak et al. [10] presented a Benders decomposition approach for the VRDAP, which was compared with CPLEX. Reihaneh and Ghoniem [11] proposed a branch-and-price algorithm for solving the VRDAP, and they tested the approach on 60 randomly generated instances with up to 25 delivery sites and 50 customers.

The ring star problem (RSP) pertains to minimizing routing and allocation costs of a cycle, and it has applications in telecommunication network design problem. The first version of the RSP is called median cycle problem. The problem is to construct a Hamiltonian cycle, in which the unvisited nodes are assigned to the nearest node on the cycle. Perez et al. [12], Renaud et al. [13], and Calvete et al. [14] proposed the variable neighborhood search and tabu search, random key evolutionary algorithm, and bi-level programing-based evolutionary algorithm for solving the RSP, respectively. Furthermore, Baldacci et al. [15] proposed the capacity ring star problem (CmRSP), which was to find m Hamiltonian cycles. Each customer that was not visited by the cycles was assigned to the nearest customer or Steiner node on the cycles. Naji-Azimi et al. [16, 17] introduced two heuristic procedures based on VNS to solve the CmRSP, and their method solved the instances within 200 vertices.

Current and Schilling [18] proposed a CSP in which each node that was not on the cycle had to be covered by at least one node on the cycle. Laporte and Semet [19] proposed a CTP in which the nodes were classified into three groups: nodes that must be visited, nodes that must be covered, and nodes that can be covered or visited. Furthermore, Golden et al. [20], Salari et al. [21] and Salari et al. [22] proposed local search algorithm, integer programing-based local search, and hybrid ant colony optimization to solve the CSP, respectively.

3 Problem statement

This paper focuses on the single vehicle routing and allocation optimization, and the problem can be described as: given a network $G = (V, E \cup A)$, where V is the node set, E is the edge set and A is arc set. Node set V is defined as $V = \{0\} \cup W \cup T$, where 0 is the depot, W is customer set, and T is parking spot set. Edges in $E = \{(i, j)\}$ refer to undirected delivery links from node $i \in T \cup \{0\}$ to node $j \in T \cup \{0\}$. Arcs in $A = \{< i, j >\}$ refer to undirected self-pickup links from customer $i \in W$ to parking spot $j \in T$. Edge $(i, j) \in E$ and arc $< i, j > \in A$ are associated with a non-negative delivery cost c_{ij} and a self-pickup cost d_{ij}, respectively. p_i is the parking cost of parking node $i \in T$.

The problem is to construct a Hamiltonian cycle over a subset of node in $T \cup \{0\}$, in which each customer in W is assigned to the nearest parking spot in T on the cycle. The goal is to minimize the total cost, including the delivery and self-pickup costs. Figure 1 illustrates an example of the proposed problem.

3.1 Node-based formulation

In this subsection, we develop a node-based formulation for the problem. The subtour elimination constraints are the well-known constraints for the traveling salesman problem (TSP) [23–25]. Other variables to formulate the problem are defined as follows:

$$w_i = \begin{cases} 1, & \textit{parking node } i \textit{ is visited by tour} \\ 0, & \textit{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \textit{edge } (i,j) \textit{ is visited by tour} \\ 0, & \textit{otherwise} \end{cases}$$

$$y_{ij} = \begin{cases} 1, & \textit{customer } i \textit{ is assigned to parking node } j \\ 0, & \textit{otherwise} \end{cases}$$

The node-based problem can be formulated as:

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij} + \sum_{<i,j>\in A} d_{ij}y_{ij} + \sum_{i\in T} p_i w_i \tag{1a}$$

$$w_0 = 1 \tag{1b}$$

$$\sum_{j\in T\cup\{0\}} x_{ij} = \sum_{j\in T\cup\{0\}} x_{ji} = w_i, \quad \forall i \in T \cup \{0\} \tag{1c}$$

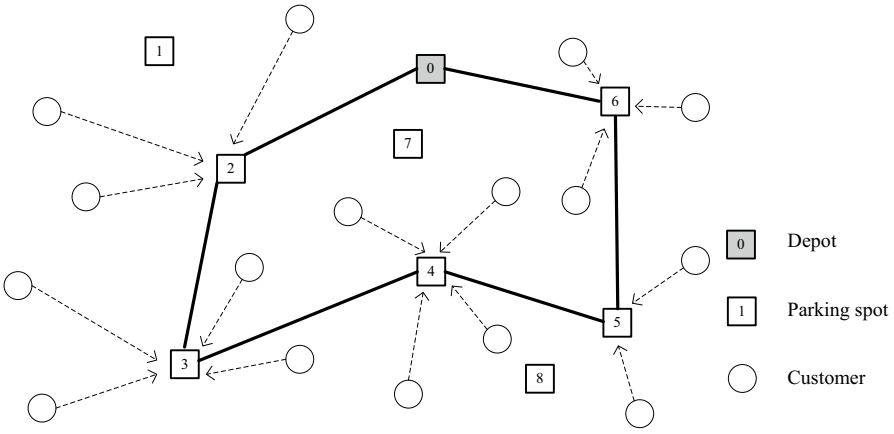

Fig. 1 An example for the proposed problem

$$\sum_{(i,j)\in E(S)} x_{ij} \le |S| - 1, \quad S \subseteq T, S \neq \phi \tag{1d}$$

$$\sum_{j \in T} y_{ij} = 1, \quad \forall i \in W \tag{1e}$$

$$y_{ij} \leq w_j, \quad \forall i \in W, \forall j \in T \tag{1f}$$

$$w_i = \{0, 1\}, \quad \forall i \in T, \tag{1g}$$

$$x_{ij} = \{0, 1\}, \quad \forall i, j \in T \cup \{0\} \tag{1h}$$

$$y_{ij} = \{0, 1\}, \quad \forall i \in W, \forall j \in T \tag{1i}$$

Objective function (1a) is to minimize the routing, self-pickup, and parking costs. Constraint (1b) indicates that the depot must be visited. Constraint (1c) represents the in-degree and out-degree of the parking node. If the parking node is visited, then the in-degree and out-degree of the parking node are both 1; otherwise, they are both 0. Constraint (1d) denote the subtour elimination constraints. Constraint (1e) represents that each customer can only be assigned to one parking node. Constraint (1f) indicates that if a parking node is assigned to a customer, then the parking node must be visited by the tour. Constraints (1g–1i) define the variables.

3.2 Flow-based formulation

In this subsection, we develop a flow-based formulation for the problem. We define a variable u_{ij} to eliminate the subtour of the tour. u_{ij} is an integer variable, representing the tour load after leaving vertex i and just before visiting vertex j. Other variables are the same as those in the node-based formulation, and the flow-based problem is formulated as:

$$\min \sum_{(i,j) \in E} c_{ij} x_{ij} + \sum_{<i,j> \in A} d_{ij} y_{ij} + \sum_{i \in T} p_i w_i \tag{2a}$$

$$w_0 = 1 \tag{2b}$$

$$\sum_{j \in T \cup \{0\}} x_{ij} = \sum_{j \in T \cup \{0\}} x_{ji} = w_i, \quad \forall i \in T \cup \{0\} \tag{2c}$$

$$\sum_{j \in \{0\} \cup T} u_{ij} - \sum_{j \in \{0\} \cup T} u_{ji} = \sum_{j \in \{0\} \cup T} x_{ji}, \quad \forall i \in T \tag{2d}$$

$$u_{ij} \leq |T| x_{ij}, \quad \forall i, j \in T \tag{2e}$$

$$\sum_{j \in T} y_{ij} = 1, \quad \forall i \in W \tag{2f}$$

$$y_{ij} \leq w_j, \quad \forall i \in W, \forall j \in T \tag{2g}$$

$$w_i = \{0, 1\}, \quad \forall i \in T \tag{2h}$$

$$x_{ij} = \{0, 1\}, \quad \forall i, j \in T \cup \{0\} \tag{2i}$$

$$y_{ij} = \{0, 1\}, \quad \forall i \in W, \forall j \in T \tag{2j}$$

$$u_{ij} \geq 0, \quad \forall i, j \in T \cup \{0\} \tag{2k}$$

Objective function (2a) is to minimize the routing, self-pickup, and parking costs. Constraints (2b–2c) are the same as Constraints (1b–1c). Constraints (2d–2e) are the subtour elimination constraints. Constraints (2f–2g) are the customer assignment constraints. Constraints (2h–2k) define the variables.

3.3 Bilevel programming formulation

The bilevel programing problem consists of a upper-level optimization problem (UOP) and a lower-level optimization problem (LOP), and the UOP is determined by the LOP [26–28]. For the proposed problem, the UOP is a leader problem, which is to decide on the parking location. The LOP includes two follower problems, which are TSP and customer assignment problem.

$$\min \ G(w) + H(w) + \sum_{i \in T} p_i w_i \tag{3a}$$

$$w_0 = 1 \tag{3b}$$

$$w_i = \{0, 1\}, \quad \forall i \in T \tag{3c}$$

The firs follower problem is a TSP, which is described as:

$$\min \sum_{(i,j) \in E} c_{ij} x_{ij} \tag{3d}$$

$$\sum_{j \in T \cup \{0\}} x_{ij} = \sum_{j \in T \cup \{0\}} x_{ji} = w_i, \quad \forall i \in T \cup \{0\} \tag{3e}$$

$$\sum_{j\in\{0\}\cup T} u_{ij} - \sum_{j\in\{0\}\cup T} u_{ji} = \sum_{j\in\{0\}\cup T} x_{ji}, \quad \forall i \in T \tag{3f}$$

$$u_{ij} \le |T| x_{ij}, \quad \forall i,j \in T \tag{3g}$$

$$x_{ij} = \{0,1\}, \quad \forall i,j \in T \cup \{0\} \tag{3h}$$

$$u_{ij} \ge 0, \quad \forall i,j \in T \cup \{0\} \tag{3i}$$

The second follower problem is a customer assignment problem, which is describes as:

$$\min \sum_{<i,j>\in A} d_{ij} y_{ij} \tag{3j}$$

$$\sum_{j\in T} y_{ij} = 1, \quad \forall i \in W \tag{3k}$$

$$y_{ij} \le w_j, \quad \forall i \in W, \forall j \in T \tag{3l}$$

$$y_{ij} = \{0,1\}, \quad \forall i \in W, \forall j \in T \tag{3m}$$

Objective function (3a) is to minimize the total cost, including the routing, self-pickup, and parking costs. Objective functions (3d) and (3j) are to minimize the routing cost and self-pickup cost, respectively. Constraints (3b) indicate that the depot must be visited. Constraints (3e–3i) are the TSP constraints. Constraints (3j–3m) are the customer assignment problem constraints.

4 The variable neighborhood search

The variable neighborhood search (VNS), introduced by Mladenovic and Hansen [29], has effective results in solving combination problems [30–33]. In this section, we propose a VNS heuristic to solve the proposed problem, which uses the ideas from the bi-level programing formulations proposed in Sect. 3. In the VNS, the main procedure is used to explore the solution space of the leader problem, a 3-opt procedure is developed to solve the first follower problem, and a nearest neighborhood (NN) is applied to solve the second follower problem. The VNS heuristic begins from an initial solution, which is generated by the NN algorithm over the depot and all parking nodes. The initial solution is set as the current solution, and a new initial solution is obtained by shaking the current solution. Subsequently, a local search is applied to explore the space of the new initial solution to obtain a local optimal solution. If the local optimal solution is better than the incumbent, then the current

solution is moved to the local optimal solution. The heuristic proposed by Lin and Kernighan (LKH) [34] is applied to improve the tour of current solution, and the search with the initial neighborhood ($k = 1$) is continued; otherwise, set $k = k + 1$. Algorithm 1 details the framework of the VNS for the proposed problem.

Algorithm 1 The variable neighborhood search

Comment: $f(l\_sol)$ is the objective function value for l_sol; l_sol and b_sol are the local and best optimal solutions, repectively;

```
1: b_sol ← initial_solution;
2: while (termination criterion not met) do
3:     k ← 1;
4:     while (k < k_max) do
5:         l_sol ← shaking(b_sol, k);
6:         l_sol ← local_search(l_sol);
7:         if (f(l_sol) < f(b_sol)) then
8:             k ← 1;
9:             b_sol ← l_sol;
10:            b_sol.tour ← LKH(b_sol.tour);
11:        else
12:            k ← k + 1;
13:        end if
14:    end while
15: end while
```

4.1 Solution representation

A solution in the VNS consists of three sequences. The first sequence is a binary sequence (location), which represents the solution of the leader problem. The second and third sequences are both an integer sequence (tour and assignment), and they indicate the solution of the first and second follower problems, respectively. In the "location" sequence, 0 and 1 indicate that the parking node is visited or is not visited by the tour, respectively. Each integer in "tour" represents the position of each parking node on the tour. Each value in "allocation" explains which customer is assigned to which parking node. Figure 2 illustrates an example of the solution. The "location" sequence in Fig. 2a represents that parking nodes 2, 3, 4, 5, and 6 are visited by the tour, and parking nodes 1 and 7 are not visited. The "tour" sequence in Fig. 2b shows that the tour over the visited parking nodes is 0-2-3-4-5-6-0. The "allocation" sequence in Fig. 2c indicates that customers 9, 10, and 11 are allocated to parking node 2; customers 12, 13, 14, and 15 are allocated to parking node 3; and customers 16 is assigned to parking node 4, and so on.

Node	0	1	2	3	4	5	6	7	...
Location	1	0	1	1	1	1	1	0	...

(a) The "location" sequence

Node	0	1	2	3	4	5	6	7	...
Tour	1	0	2	3	4	5	6	0	...

(b) The "tour" sequence

Node	9	10	11	12	13	14	15	16	...
Allocation	2	2	2	3	3	3	3	4	...

(c) The "allocation" sequence

Fig. 2 Illustrative example for the solution

4.2 Local search

The neighborhood structures in the VNS, including inversion and exchange procedures, are used to solve the leader problem of the proposed problem. The 3-opt and NN procedures are used to solve the first and second follower problems, respectively, on the basis of the current solution obtained from the leader problem. The local search iteratively applies the inversion and exchange procedures to improve the current solution until it cannot be improved. Algorithm 2 details the framework of the local search procedure, and the following subsections provides the details of the proposed procedures.

Algorithm 2 The local search

Input: l_sol;
Output: l_sol;
1: **while** (l_sol can be improved) **do**
2: $Inversion(l\_sol)$;
3: $Exchange(l\_sol)$;
4: **end while**

4.2.1 Inversion procedure

The first neighborhood structure of the VNS is the inversion procedure, which is to extract a parking node $i \in T$ and invert its "location" value. Essentially, a new

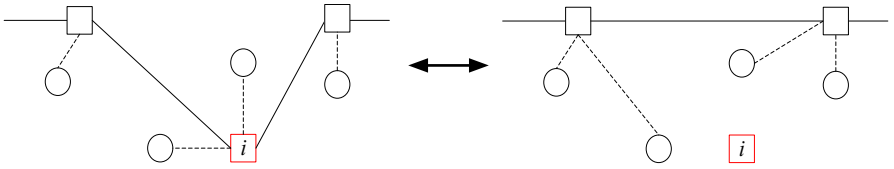

Fig. 3 Illustrative example for the inversion procedure

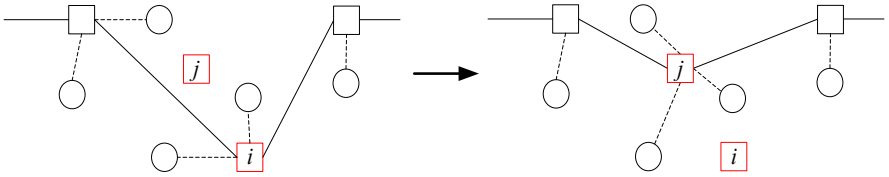

Fig. 4 Illustrative example for the exchange procedure

solution generation process of the inversion procedure can be described as follows. First, the parking node $i \in T$ is selected randomly, and its "location" value is inverted. If the original "location" value of the parking node $i \in T$ is 1, it is dropped from the tour; otherwise, it is inserted to its best position on the tour. That is, the selected parking node is inserted from the first position of the tour to the last one, and the insertion position with the minimal insertion cost is selected as the best position. Second, the NN procedure is used to re-assign the customers to the parking nodes that are on the tour of the current solution. Finally, a 3-opt procedure is applied to minimize the length of the tour of the current solution. Figure 3 illustrates an example of the inversion procedure.

4.2.2 Exchange procedure

The second neighborhood structure of the VNS is the exchange procedure, which is to extract two parking nodes $i(location(i) = 1)$ and $j(location(j) = 0)$ and exchange their "location" values. In practice, the exchange procedure generates a new solution by the following steps. First, parking nodes $i(location(i) = 1)$ and $j \in S(location(j) = 0)$ are selected, where S is a set of R parking nodes nearest to the node i, and the "location" values of the two nodes are exchanged. The parking node i on the tour is replaced by parking node j to obtain a new tour. Second, the customers are re-assigned to the parking nodes that are on the new tour by the NN procedure. Finally, the 3-opt procedure is applied to minimize the length of the new tour to obtain a new solution. Figure 4 illustrates an example of the exchange procedure.

4.3 The procedure for follower problems

4.3.1 3-opt procedure

The 3-opt procedure attempts to remove three edges of the tour and then replaces three other edges to obtain a better solution, which is then used to minimize the tour of the current solution. Figure 5 illustrates a 3-opt move. Helsgaun [35] proposed a 3-opt (forward and backward insertion) move. In the 3-opt, a node i is randomly selected from the tour, the node j is selected from the T nearest nodes to node i, and node k is selected from the T nearest nodes to node j. Then, the edges x_1, x_2, and x_3 are selected from the edges before and after nodes i, j, and k, respectively. As shown in Fig. 5, the 3-opt replaces edges x_1, x_2 and x_3 with edges y_1, y_2, and y_3 to generate a new tour. The process iterates until the stop condition is met.

4.3.2 NN procedure

The (NN) algorithm is to assign each customer to the nearest parking node on the tour. Essentially, when the "location" value of each parking node is fixed, the optimal solution of the second follower problem is the solution in which each customer is assigned to the nearest parking node on the tour.

4.4 Shaking procedure

To escape from the local optimum, the VNS applies a shaking procedure to perturb the current solution. In particular, the shaking procedure is to randomly select k (an input parameter) parking nodes, and invert their "location" values. Then, each customer is re-assigned to the parking node whose "location" value is 1 by the NN procedure. Finally, the 3-opt procedure is applied to minimize the length of the new tour to obtain a new solution.

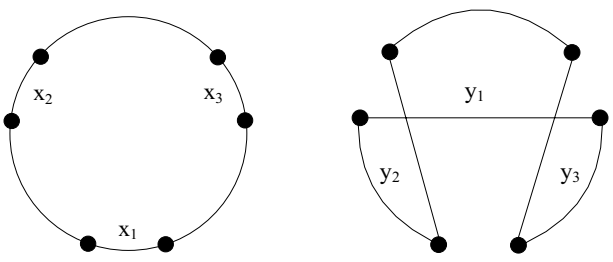

Fig. 5 A 3-opt move

5 Computational results

To test the performance of the proposed VNS heuristic, a set of 108 instances are proposed, which are generated by using the TSP library (TSPLIB) instances proposed by [36]. The problems range from 51 to 783 nodes, which are divided into small- ($51 \leq |V| \leq 100$), medium- ($150 \leq |V| \leq 200$), and large-size ($225 \leq |V| \leq 783$) instances. For each problem, n is the number of the nodes, the first node is depot, the next $\lfloor \alpha(n-1) \rfloor$ nodes with $\alpha \in \{0.25, 0.5, 0.75\}$ are parking spots, and the last ones are customers. For a pair of nodes i and j, l_{ij} is the Euclidean distance, which is computed according to the TSPLIB standard. The routing cost c_{ij} is equal to the Euclidean distance l_{ij} ($c_{ij} = l_{ij}$).the self-pickup cost is $d_{ij} = \lceil \lambda l_{ij} \rceil$, where λ is set as {0.25,0.50,0.75}.

The VNS is coded in Microsoft Visual C++, and the node-based and flow-based formulations are solved using the CPLEX 12.7.1 framework. The experiments are performed on a personal computer with 2.2 GHz Intel Core I5-5200U CPU and 4 GB RAM.. In addition, the termination criterion is set as 5n, and parameters R and T of the exchange and 3-opt procedures are both set as $\lfloor \alpha(n-1) \rfloor$.

5.1 Results for small- and medium-size instances

Tables 1 and 2 show the results obtained by CPLEX and the proposed VNS for the small- and medium-size instances. The running time of CPLEX for solving each instance is limited to 3600 s. In these tables, the first columns provide the names of instances, and the second and third columns contain the values of α and λ, respectively. The fourth to sixth columns are the results obtained by solving the node-based model over each instance. "Obj." columns refer to the objective function of the optimal solution. "Gap" columns present the gaps from the lower bound. "TT" columns are the total running times. The next three columns report the same results obtained by running the flow-based model. Columns 10 to 13 are the results obtained by VNS algorithm. "Best" and "Avg." columns refer to the best and average objective function of the solutions over five independent runs, respectively. "Dev." columns include the deviation of the "Avg." from the optimal objective function, which are calculated as $(\text{Avg.} - \text{Opt})/\text{Opt} \times 100$. "Opt" is the best objective function of the solution obtained by the VNS and CPLEX. "Time" columns contain the average running time of the VNS over five runs.

For the 36 small size instances in Table 1, the node-based and flow-based formulations have found 34 and 35 best solutions with the average running time of 1285.95 and 849.71 s, respectively. The results in Table 1 show that the both exact methods perform well on small size instances in which most of the optimal solutions have been obtained. As shown in Table 2, for the 36 medium size instances, the node-based and flow-based formulations have only obtained 15 and 14 best solutions, respectively. Several instances have not obtained the optimal solutions, for which the time threshold of CPLEX has been reached. The results in the two tables show that all of the optimal solutions have been obtained over the small and medium

Table 1 Comparison for the small size instances

Instance	α	λ	Node-based formulation			Flow-based formulation			VNS			
			Obj	Gap	TT	Obj	Gap	TT	Best	Avg	Dev	TT
eil51	0.25	0.25	**230**	0.00	1.11	**230**	0.00	1.44	**230**	**230**	0.00	0.03
eil51	0.25	0.50	**360**	0.00	1.08	**360**	0.00	1.55	**360**	**360**	0.00	0.03
eil51	0.25	0.75	**467**	0.00	0.45	**467**	0.00	2.23	**467**	**467**	0.00	0.03
eil51	0.50	0.25	**174**	0.00	3.18	**174**	0.00	10.86	**174**	**174**	0.00	0.06
eil51	0.50	0.50	**266**	0.00	15.49	**266**	0.00	18.40	**266**	**266**	0.00	0.09
eil51	0.50	0.75	**333**	0.00	9.87	**333**	0.00	17.50	**333**	**333**	0.00	0.09
eil51	0.75	0.25	**105**	0.00	2.34	**105**	0.00	7.25	**105**	**105**	0.00	0.06
eil51	0.75	0.50	**169**	0.00	21.45	**169**	0.00	36.40	**169**	**169**	0.00	0.09
eil51	0.75	0.75	**216**	0.00	112.01	**216**	0.00	44.20	**216**	**216**	0.00	0.09
eil76	0.25	0.25	**304**	0.00	3.90	**304**	0.00	3.40	**304**	**304**	0.00	0.06
eil76	0.25	0.50	**483**	0.00	3.72	**483**	0.00	4.15	**483**	**483**	0.00	0.08
eil76	0.25	0.75	**618**	0.00	1.06	**618**	0.00	2.77	**618**	**618**	0.00	0.05
eil76	0.50	0.25	**234**	0.00	1841.36	**234**	0.00	790.72	**234**	**234**	0.00	0.11
eil76	0.50	0.50	**343**	0.00	1518.44	**343**	0.00	238.40	**343**	**343**	0.00	0.10
eil76	0.50	0.75	**417**	0.00	73.14	**417**	0.00	100.20	**417**	**417**	0.00	0.10
eil76	0.75	0.25	**149**	5.31	3600.00	**149**	0.00	1390.10	**149**	**149**	0.00	0.08
eil76	0.75	0.50	225	16.02	3600.00	**224**	5.70	3600.00	**224**	**224**	0.00	0.09
eil76	0.75	0.75	**271**	13.17	3600.00	**271**	11.30	3600.00	**271**	**271**	0.00	0.09
rd100	0.25	0.25	**5261**	0.00	23.76	**5261**	0.00	15.33	**5261**	**5261**	0.00	0.09
rd100	0.25	0.50	**7950**	0.00	8.45	**7950**	0.00	6.48	**7950**	**7950**	0.00	0.08
rd100	0.25	0.75	**10,502**	0.00	8.08	**10,502**	0.00	4.69	**10,502**	**10,502**	0.00	0.07
rd100	0.50	0.25	**3975**	18.60	3600.00	**3975**	4.40	3600.00	**3975**	**3975**	0.00	0.30
rd100	0.50	0.50	**5550**	8.52	3600.00	**5550**	0.00	2436.41	**5550**	**5550**	0.00	0.20
rd100	0.50	0.75	**6769**	0.00	2182.02	**6769**	0.00	104.96	**6769**	**6769**	0.00	0.23

Table 1 (continued)

Instance	α	λ	Node-based formulation			Flow-based formulation			VNS			
			Obj	Gap	TT	Obj	Gap	TT	Best	Avg	Dev	TT
rd100	0.75	0.25	2730	32.91	3600.00	2704	26.31	3600.00	**2694**	**2694**	0.00	0.30
rd100	0.75	0.50	**3400**	18.40	3600.00	**3400**	10.36	3600.00	**3400**	**3400**	0.00	0.33
rd100	0.75	0.75	**3953**	11.21	3600.00	**3953**	2.13	3600.00	**3953**	**3953**	0.00	0.20
eil101	0.25	0.25	**344**	0.00	15.52	**344**	0.00	13.65	**344**	**344**	0.00	0.13
eil101	0.25	0.50	**521**	0.00	3.69	**521**	0.00	8.87	**521**	**521**	0.00	0.13
eil101	0.25	0.75	**670**	0.00	2.87	**670**	0.00	6.65	**670**	**670**	0.00	0.10
eil101	0.50	0.25	**235**	0.00	981.76	**235**	0.00	315.82	**235**	**235**	0.00	0.20
eil101	0.50	0.50	**354**	2.41	3600.00	**354**	0.00	186.73	**354**	**354**	0.00	0.26
eil101	0.50	0.75	**436**	0.00	428.33	**436**	0.00	69.51	**436**	**436**	0.00	0.22
eil101	0.75	0.25	**129**	0.00	2100.00	**129**	0.00	1809.20	**129**	**129**	0.00	0.16
eil101	0.75	0.50	**180**	12.83	3600.00	**180**	0.00	958.45	**180**	**180**	0.00	0.21
eil101	0.75	0.75	**212**	0.00	931.00	**212**	0.00	383.24	**212**	**212**	0.00	0.24
Avg			1626.0		1285.95	1625.2		849.71	**1624.9**	**1624.9**	**0.00**	**0.13**

Table 2 Comparison for the medium size instances

Instance	α	λ	Node-based formulation			Flow-based formulation			VNS			
			Obj	Gap	TT	Obj	Gap	TT	Best	Avg	Dev	TT
kroA150	0.25	0.25	**18,493**	7.4	3600.0	**18,493**	0.0	1858.1	**18,493**	18,493.0	0.0	0.7
kroA150		0.50	**26,090**	0.0	633.0	**26,090**	0.0	29.1	**26,090**	26,090.0	0.0	0.7
kroA150		0.75	**33,198**	0.0	52.6	**33,198**	0.0	11.4	**33,198**	33,198.0	0.0	0.4
kroA150	0.50	0.25	14,391	24.6	3600.0	14,532	21.5	3600.0	**14,369**	14,369.0	0.0	1.0
kroA150		0.50	19,211	12.6	3600.0	19,211	7.2	3600.0	**19,196**	19,196.0	0.0	1.7
kroA150		0.75	23,202	7.9	3600.0	**23,197**	0.2	3600.0	**23,197**	23,197.0	0.0	1.8
kroA150	0.75	0.25	10,155	39.5	3600.0	10,452	37.3	3600.0	**10,033**	10,033.0	0.0	1.8
kroA150		0.50	**13,507**	31.5	3600.0	13,559	24.6	3600.0	**13,507**	13,507.0	0.0	1.9
kroA150		0.75	**15,892**	28.2	3600.0	16,353	20.5	3600.0	**15,892**	15,892.0	0.0	1.0
kroB150	0.25	0.25	**17,502**	0.6	3600.0	**17,502**	0.0	99.8	**17,502**	17,502.0	0.0	1.1
kroB150		0.50	**25,485**	0.0	20.6	**25,485**	0.0	17.0	**25,485**	25,485.0	0.0	0.6
kroB150		0.75	**32,713**	0.0	8.0	**32,713**	0.0	12.5	**32,713**	32,713.0	0.0	0.6
kroB150	0.50	0.25	13,602	26.7	3600.0	13,564	19.2	3600.0	**13,527**	13,527.0	0.0	1.5
kroB150		0.50	18,137	13.7	3600.0	18,207	8.0	3600.0	**18,106**	18,106.0	0.0	1.0
kroB150		0.75	**22,164**	8.2	3600.0	**22,164**	1.6	3600.0	**22,164**	22,164.0	0.0	0.9
kroB150	0.75	0.25	10,571	50.2	3600.0	12,150	52.3	3600.0	**10,203**	10,203.0	0.0	2.7
kroB150		0.50	**12,762**	36.5	3600.0	15,158	41.5	3600.0	**12,762**	12,762.0	0.0	1.8
kroB150		0.75	14,906	30.6	3600.0	15,822	26.5	3600.0	**14,829**	14,829.0	0.0	1.5
kroA200	0.25	0.25	20,071	9.3	3600.0	**20,058**	2.3	3600.0	**20,058**	20,058.0	0.0	1.7
kroA200		0.50	**29,864**	0.0	1727.7	**29,864**	0.0	70.4	**29,864**	29,864.0	0.0	1.4
kroA200		0.75	**38,278**	0.0	533.6	**38,278**	0.0	34.1	**38,278**	38,278.0	0.0	1.7
kroA200	0.50	0.25	16,409	33.3	3600.0	22,472	51.5	3600.0	**15,311**	15,311.0	0.0	3.6
kroA200		0.50	26,713	33.5	3600.0	26,844	32.4	3600.0	**21,208**	21,212.0	0.0	3.9
kroA200		0.75	25,939	9.1	3600.0	29,564	17.4	3600.0	**25,938**	25,938.0	0.0	3.4

Table 2 (continued)

Instance	α	λ	Node-based formulation			Flow-based formulation			VNS			
			Obj	Gap	TT	Obj	Gap	TT	Best	Avg	Dev	TT
kroA200	0.75	0.25	11,232	45.8	3600.0	13,942	66.4	3600.0	**11,169**	11,169.0	0.0	5.1
kroA200		0.50	14,872	32.6	3600.0	17,130	41.6	3600.0	**14,746**	14,746.0	0.0	3.1
kroA200		0.75	17,594	25.2	3600.0	23,808	45.5	3600.0	**17,238**	17,238.0	0.0	5.0
kroB200	0.25	0.25	20,778	8.5	3600.0	**20,777**	3.3	3600.0	**20,777**	20,777.0	0.0	1.3
kroB200		0.50	**30,454**	0.0	415.7	**30,454**	0.0	69.1	**30,454**	30,454.0	0.0	1.3
kroB200		0.75	**38,874**	0.0	1143.7	**38,874**	0.0	31.3	**38,874**	38,874.0	0.0	1.6
kroB200	0.50	0.25	17,002	33.4	3600.0	15,737	24.5	3600.0	**15,137**	15,137.0	0.0	2.6
kroB200		0.50	**20,662**	14.3	3600.0	22,262	17.2	3600.0	**20,662**	20,662.0	0.0	3.2
kroB200		0.75	26,448	13.3	3600.0	28,871	17.7	3600.0	**25,351**	25,351.0	0.0	3.1
kroB200	0.75	0.25	11,060	45.8	3600.0	13,429	54.1	3600.0	**10,591**	10,591.0	0.0	3.1
kroB200		0.50	14,153	34.8	3600.0	15,873	38.2	3600.0	**14,106**	14,106.0	0.0	2.5
kroB200		0.75	16,857	29.9	3600.0	20,467	41.0	3600.0	**16,620**	16,620.0	0.0	2.9
Avg			20,534.5	19.1	2926.0	21,570.9	19.8	2662.0	**20,212.4**	**20,212.6**	**0.0**	**2.0**

size instances by the proposed VNS algorithm. In these tables, the best solutions found by each approach are represented in bold. The deviations of the average solutions obtained by the VNS algorithm from the optimal solution indicates the superiority of the proposed method, where the overall deviations are both 0 for small and medium instances.

5.2 Results for large-size instances

The results of the VNS algorithm over 36 large-size instances are presented in Table 3. In this table, the first to third columns have the same meaning as those in Tables 1 and 2. The fourth to ninth columns are the results obtained by solving the node-based and flow-based models. "Obj." columns refer to the objective function of the optimal solution. "Gap" columns present the gaps from the lower bound. "TT." columns are the total running times. Columns 10 to 13 are the results obtained by VNS algorithm. "Best" and "Avg." columns refer to the best and average objective function of the solutions over five independent runs, respectively. "Dev." columns include the deviation of the "Avg." from the optimal objective function, and it is calculated as $(\text{Avg.} - \text{Opt})/\text{Opt} \times 100$. "Opt" is the best objective function of the solution obtained by the VNS and CPLEX. "TT." columns contain the average running time of the VNS over five runs.

The results of the algorithms for the large-size instances are given in Table 3. The proposed VNS obtains a large number of the best solutions as the size of the instances increases. At the same time, the performances of the exact algorithms reduce significantly. The results in Table 3 show that the proposed VNS obtains 36 best solutions out of 36 instances with a percentage of 100% for the large-size problems. By contrast, the node-based and flow-based models find only 10 and 3 best solutions within a limit running time of 3600 s, respectively. In addition, the proposed VNS solves all the large problems with an average running time of 27.16 s. The results illustrate that the proposed VNS has a good performance for solving large-size problems, and the node-based model outperforms the flow-based model.

6 Conclusion

In this work, we propose a delivery problem that arises in the last-mile delivery during major infectious disease outbreak, such as COVID-19. The problem is formulated by three mathematical models: the node-based, flow-based, and bi-level programing models. A VNS based on the ideas from the bi-level programing formulation is developed to solve the problem. This formulation divides the delivery problem into one leader and two follower subproblems. The proposed VNS is tested on 108 randomly instances, which are generated by using the TSPLIB benchmark instances. The experimental results indicate that the proposed VNS has the best performance for solving the delivery problem. Specifically, it obtains 108 best solutions out of 108 instances, and the node-based model outperforms the flow-based model as the size of the problem increases.

Table 3 The results for the large-size instances

Instance	α	λ	Node-based formulation			Flow-based formulation			VNS			
			Obj	Gap	TT	Obj	Gap	TT	Best	Avg	Dev	TT
tsp225	0.25	0.25	**5631**	5.20	3600.00	5633	2.05	3600.00	**5631**	**5631.0**	0.00	2.48
tsp225	0.25	0.50	**10,566**	2.37	3600.00	**10,566**	0.24	3600.00	**10,566**	**10,566.0**	0.00	1.18
tsp225	0.25	0.75	**15,430**	1.45	3600.00	**15,430**	0.00	2786.69	**15,430**	**15,430.0**	0.00	2.14
tsp225	0.50	0.25	**2887**	14.64	3600.00	2903	12.63	3600.00	**2887**	**2887.0**	0.00	5.64
tsp225	0.50	0.50	**4926**	8.09	3600.00	4930	6.07	3600.00	**4926**	**4926.0**	0.00	5.08
tsp225	0.50	0.75	**6912**	4.77	3600.00	6922	3.12	3600.00	**6912**	**6912.0**	0.00	4.28
tsp225	0.75	0.25	1484	38.97	3600.00	1667	48.03	3600.00	**1467**	**1467.0**	0.00	5.12
tsp225	0.75	0.50	1998	27.33	3600.00	2055	30.74	3600.00	**1955**	**1955.0**	0.00	6.33
tsp225	0.75	0.75	**2404**	19.22	3600.00	2528	24.79	3600.00	**2404**	**2404.0**	0.00	5.97
pr299	0.25	0.25	76,424	9.96	3600.00	76,435	8.08	3600.00	**76,411**	**76,411.0**	0.00	3.73
pr299	0.25	0.50	**139,946**	4.79	3600.00	139,958	3.90	3600.00	**139,946**	**139,946.0**	0.00	5.88
pr299	0.25	0.75	**203,183**	3.21	3600.00	203,218	2.55	3600.00	**203,183**	**203,183.0**	0.00	5.65
pr299	0.50	0.25	41,963	22.73	3600.00	41,871	20.63	3600.00	**41,864**	**41,864.0**	0.00	6.58
pr299	0.50	0.50	70,572	13.18	3600.00	**70,570**	11.47	3600.00	**70,570**	**70,570.0**	0.00	7.52
pr299	0.50	0.75	99,165	9.34	3600.00	98,958	7.60	3600.00	**98,799**	**98,799.0**	0.00	6.23
pr299	0.75	0.25	21,168	45.00	3600.00	21,732	47.53	3600.00	**21,139**	**21,139.0**	0.00	9.47
pr299	0.75	0.50	29,330	31.76	3600.00	29,128	30.58	3600.00	**29,038**	**29,038.0**	0.00	3.23
pr299	0.75	0.75	36,513	24.07	3600.00	36,507	22.09	3600.00	**36,500**	**36,500.0**	0.00	6.91
u574	0.25	0.25	99,704	1.92	3600.00	99,754	1.73	3600.00	**99,697**	**99,697.0**	0.00	34.36
u574	0.25	0.50	195,615	0.94	3600.00	195,754	0.84	3600.00	**195,538**	**195,538.0**	0.00	19.93
u574	0.25	0.75	**291,235**	0.58	3600.00	291,284	0.94	3600.00	**291,235**	**291,235.0**	0.00	30.45
u574	0.50	0.25	27,898	12.59	3600.00	38,295	38.11	3600.00	**27,856**	**27,856.0**	0.00	34.83
u574	0.50	0.50	49,859	6.75	3600.00	54,749	16.46	3600.00	**49,737**	**49,737.0**	0.00	50.30
u574	0.50	0.75	71,562	4.48	3600.00	74,579	9.38	3600.00	**71,482**	**71,482.0**	0.00	33.48

Table 3 (continued)

Instance	α	λ	Node-based formulation			Flow-based formulation			VNS			
			Obj	Gap	TT	Obj	Gap	TT	Best	Avg	Dev	TT
u574	0.75	0.25	14,880	26.36	3600.00	15,058	30.01	3600.00	**14,708**	**14,708.0**	0.00	18.69
u574	0.75	0.50	23,936	15.32	3600.00	24,258	18.57	3600.00	**23,687**	**23,687.0**	0.00	23.42
u574	0.75	0.75	32,645	10.22	3600.00	35,998	20.33	3600.00	**32,610**	**32,610.0**	0.00	25.97
rat783	0.25	0.25	33,098	1.20	3600.00	33,131	1.57	3600.00	**33,097**	**33,097.0**	0.00	57.29
rat783	0.25	0.50	65,565	0.59	3600.00	65,548	0.54	3600.00	**65,546**	**65,546.0**	0.00	65.31
rat783	0.25	0.75	97,856	0.36	3600.00	97,857	0.35	3600.00	**97,855**	**97,855.0**	0.00	64.68
rat783	0.50	0.25	15,484	4.20	3600.00	15,866	6.93	3600.00	**15,477**	**15,477.0**	0.00	57.00
rat783	0.50	0.50	30,082	2.10	3600.00	30,262	2.94	3600.00	**30,081**	**30,081.0**	0.00	81.95
rat783	0.50	0.75	44,596	1.40	3600.00	44,613	1.58	3600.00	**44,592**	**44,592.0**	0.00	71.11
rat783	0.75	0.25	4965	19.92	3600.00	5181	24.68	3600.00	**4902**	4902.8	0.00	55.01
rat783	0.75	0.50	8698	10.94	3600.00	8832	13.18	3600.00	**8663**	**8663.0**	0.00	79.31
rat783	0.75	0.75	12,436	7.86	3600.00	12,580	9.38	3600.00	**12,372**	**12,372.0**	0.00	81.25
Avg			52,517.1		3600.00	53,183.6		3577.41	**52,465.6**	**52,465.7**	0.0	27.16

Future works will mainly be conducted in the directions of vehicle routing with demand allocation problem, multi vehicle routing with demand allocation problem, and so on.

Acknowledgments This work was supported by the Ministry of Chinese Education, Humanities and Social Sciences under Grant 17YJA630037; the National Science and Technology Support Program of China (Grant Nos. 71331002); the Project of Graduate Teaching Quality in Hefei University of Technology (Grant No. 110-4116000050). In addition, the work of Prof Mladenovic is partly supported by Khalifa University of Science and Technology under Award No. RC2 DSO and the framework of the Grant Number BR05236839.

References

1. MedSci homepage, https://www.medsci.cn/article/show_article.do?id=3ced19388666in. Accessed 10 May 2020
2. Fathollahi-Fard, A.M., Govindan, K., Hajiaghaei-Keshteli, M., Ahmadi, A.: A green home health care supply chain: New modified simulated annealing algorithms. J. Clean. Prod. **240**, 118200 (2019)
3. Karakostas, P., Panoskaltsis, N., Mantalaris, A., Georgiadis, M.C.: Optimization of CAR T-cell therapies supply chains. Comput. Chem. Eng. **139**, 106913 (2020)
4. Laporte, G., Nobert, Y., Arpin, D.: An exact algorithm for solving a capacitated location-routing problem. Ann. Oper. Res. **6**, 291–310 (1986)
5. Akinc, U., Srikanth, K.: Optimal routing and process scheduling for a mobile service facility. Networks **22**, 163–183 (1992)
6. Beasley, J.E., Nascimento, E.M.: The vehicle routing-allocation problem: a unifying framework. TOP **4**, 65–86 (1996)
7. Vogt, L., Poojari, C.A., Beasley, J.E.: A tabu search algorithm for the single vehicle routing allocation problem. J. Oper. Res. Soc. **58**, 467–480 (2007)
8. Baniasadi, P., Foumani, M., Smith-Miles, K., Ejov, V.: A transformation technique for the clustered generalized traveling salesman problem with applications to logistics. Eur. J. Oper. Res. **285**, 444–457 (2020)
9. Ghoniem, A., Scherrer, C.R., Solak, S.: A specialized column generation approach for a vehicle routing problem with demand allocation. J. Oper. Res. Soc. **64**, 114–124 (2013)
10. Solak, S., Scherrer, C., Ghoniem, A.: The stop-and-drop problem in nonprofit food distribution networks. Ann. Oper. Res. **221**, 407–426 (2014)
11. Reihaneh, M., Ghoniem, A.: A branch-and-price algorithm for a vehicle routing with demand allocation problem. Eur. J. Oper. Res. **272**, 523–538 (2019)
12. Perez, J.A.M., Moreno-Vega, J.M., Martin, I.R.: Variable neighborhood tabu search and its application to the median cycle problem. Eur. J. Oper. Res. **151**, 365–378 (2003)
13. Renaud, J., Boctor, F.F., Laporte, G.: Efficient heuristics for Median Cycle Problems. J. Oper. Res. Soc. **55**, 179–186 (2004)
14. Calvete, H.I., Gale, C., Iranzo, J.A.: An efficient evolutionary algorithm for the ring star problem (vol 231, pg 22. Eur. J. Oper. Res. **246**(2015), 343–343 (2013)
15. Baldacci, R., Dell'Amico, M., Gonzalez, J.S.: The capacitated m-ring-star problem. Oper. Res. **55**, 1147–1162 (2007)
16. Naji-Azimi, Z., Salari, M., Toth, P.: A heuristic procedure for the Capacitated m-ring-star problem. Eur. J. Oper. Res. **207**, 1227–1234 (2010)
17. Naji-Azimi, Z., Salari, M., Toth, P.: An integer linear programming based heuristic for the capacitated m-ring-star problem. Eur. J. Oper. Res. **217**, 17–25 (2012)
18. Current, J.R., Schilling, D.A.: The covering salesman problem. Transp. Sci. **23**, 208–213 (1989)
19. Gendreau, M., Laporte, G., Semet, F.: The covering tour problem. Oper. Res. **45**, 568–576 (1997)
20. Golden, B., Naji-Azimi, Z., Raghavan, S., Salari, M., Toth, P.: The generalized covering salesman problem. Informs J. Comput. **24**, 534–553 (2012)

21. Salari, M., Naji-Azimi, Z.: An integer programming-based local search for the covering salesman problem. Comput. Oper. Res. **39**, 2594–2602 (2012)
22. Salari, M., Reihaneh, M., Sabbagh, M.S.: Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem. Comput. Ind. Eng. **83**, 244–251 (2015)
23. Desrochers, M., Laporte, G.: Improvements and extensions to the Miller–Tucker–Zemlin subtour elimination constraints. Oper. Res. Lett. **10**, 27–36 (1991)
24. Bektas, T.: A note on the lifted Miller–Tucker–Zemlin subtour elimination constraints for the capacitated vehicle routing problem. Eur. J. Oper. Res. **158**, 793–795 (2004)
25. Miller, C.: Integer programming formulations and traveling salesman problems. J. Assoc. Comput. Mach. **7**, 326–329 (1960)
26. Bard, J.F.: Practical Bilevel Optimization, Algorithms and Applications. Kluwer Academic Publishers, Dordrecht (1998)
27. Colson, B., Marcotte, P., Savard, G.: An overview of bilevel optimization. Ann. Oper. Res. **153**, 235–256 (2007)
28. Dempe, S.: Foundations of Bilevel Programming. Kluwer Academic Publishers, Dordrecht, Boston, London (2002)
29. Mladenovic, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**, 1097–1100 (1997)
30. Qiu, Y., Wang, L., Xu, X., Fang, X., Pardalos, P.M.: A variable neighborhood search heuristic algorithm for production routing problems. Appl. Soft Comput. **66**, 311–318 (2018)
31. Karakostas, P., Sifaleras, A., Georgiadis, M.C.: Adaptive variable neighborhood search solution methods for the fleet size and mix pollution location-inventory-routing problem. Expert Syst. Appl. **153**, 113444 (2020)
32. Gruler, A., Panadero, J., de Armas, J., Moreno Perez, J.A., Juan, A.A.: Combining variable neighborhood search with simulation for the inventory routing problem with stochastic demands and stock-outs. Comput. Ind. Eng. **123**, 278–288 (2018)
33. Karakostas, P., Sifaleras, A., Georgiadis, M.C.: A general variable neighborhood search-based solution approach for the location-inventory-routing problem with distribution outsourcing. Comput. Chem. Eng. **126**, 263–279 (2019)
34. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the travelling-salesman problem. Oper. Res. **21**, 498–516 (1973)
35. Helsgaun, K.: An effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur. J. Oper. Res. **126**, 106–130 (2000)
36. Reinelt, G.: TSPLIB—a traveling salesman problem library. ORSA J. Comput. **3**, 376–384 (1991)