Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 3516 – 3531
© TÜBİTAK
doi:10.3906/elk-1803-96

Solving vehicle routing problem for multistorey buildings using iterated local search

Osman GÖKALP, Aybars UĞUR[*]
Department of Computer Engineering, Faculty of Engineering, Ege University, İzmir, Turkey

Received: 14.03.2018 • **Accepted/Published Online:** 01.07.2019 • **Final Version:** 18.09.2019

Abstract: Vehicle routing problem (VRP) which is a well-known combinatorial optimisation problem that has many applications used in industry is also a generalised form of the travelling salesman problem. In this study, we defined and formulated the VRP in multistorey buildings (Multistorey VRP) for the first time and proposed a solving method employing iterated local search metaheuristic algorithm. This variant of VRP has a great potential for turning the direction of optimisation research and applications to the vertical cities area as well as the horizontal ones. Routes of part picking or placing vehicles/humans in multistorey plants can be minimised by this way. VRP can also be applied to the optimisation of delivering the packages (goods, meals, folders, mails, etc.) to rooms or locations of the structures such as buildings, and skyscrapers for travelling robots/humans using elevators and stairs. The first detailed multistorey building optimisation experiments were conducted by designing a series of scenarios with different parameter values (number of storeys, connections between storeys and customers). The results were presented and the effects of the various building structures over the performance were discussed.

Key words: Vehicle routing problem, multistorey buildings, combinatorial optimisation, iterated local search

1. Introduction

Most of the buildings in everyday life are multistorey structures such as business centres, schools, warehouses, hospitals, apartments, and hotels. In addition, the number of high-rise buildings increases in parallel with the demand for extra place in crowded cities. As multistorey buildings become bigger and more complex, it is getting more difficult to organize services and make plans. Therefore, classical optimisation problems need to be adapted and solved in such places. The vehicle routing problem (VRP) is one of the well-known combinatorial optimisation problems and is widely used in many fields of logistics and transportation. The classical form of this problem, also known as CVRP, aims to serve a number of customers by a fleet of vehicles under capacity constraints.

Considering multistorey buildings, many CVRP-like problems possibly arise especially when a high number of items such as mails, documents or goods are to be delivered to or picked up from various locations. In this case, a good route plan for a team of delivery persons with limited carrying capacities will help the service process improve. In other words, customers will be served in less time and delivery persons will spend less energy to do their work. Thus, on the one hand, customer satisfaction is improved. On the other hand, resources for service operations are used more efficiently, and overall costs are reduced. Similarly, costs for putting away and picking processes in warehouses can be improved in this way. Another important application

[*]Correspondence: aybars.ugur@ege.edu.tr

of CVRP in multistorey buildings might be for automated robots, which are employed for delivery or pick up operations.

After it was formulated by Dantzig and Ramser [1], many variants of the VRP have been proposed so far. One of the most famous variants of the VRP is VRP with time windows where distribution of items must be carried out within a time window, which is predefined for every single customer. Another problem type is VRP with pickup and delivery where customers can both send and deliver items. In stochastic VRP, some of the problem variables such as customer demands and travelling times are not known before the algorithm starts. Rather, they are determined randomly. Yet another variant is dynamic VRP in which some problem components may change dynamically while it is being solved. These are just some of the main types of the VRP that the number of variants will continue to grow as specific real world constraints are included in the problem.

Since VRP is an NP-hard problem, the time required to find an optimum result increases exponentially as the size of the problem increases. Therefore, only small instances can optimally be solved by the exact solvers in practice. When it is not possible to produce an optimal solution for larger problem instances, heuristic methods can be applied to find near-optimal solutions in a reasonable time. In general, heuristics do not exhaust the whole search space but approximate the local optimum by following some problem-specific rules. Here, there is a trade-off between time and quality of a solution that the longer the heuristic algorithm is run the better solution is produced. Despite its advantages, heuristics are not powerful enough to explore the whole search space. To overcome this limitation, metaheuristic algorithms are typically used for guiding heuristics and provide a balance between exploration and exploitation behaviours of the optimisation process.

The main contribution of this study is adapting and solving the classical VRP for multistorey buildings for the first time. After formulating the multistorey VRP, we proposed a method that enables existing CVRP optimisation methods for solving it. A comprehensive experimental study was conducted to investigate the effect of multistorey VRP structure on the performance of optimisation algorithm.

The remainder of this paper is organised as follows. In Section 2, related work is given by considering heuristic and metaheuristic algorithms for VRP and studies on navigation in multistorey buildings. Section 3 formulates the multistorey VRP by making a comparison with the classical one. The problem-solving method and its steps are explained in detail in Section 4. In Section 5, performance tests and results of the optimisation algorithm over different multistorey VRP cases are presented. Finally, the conclusion of this study and discussion along with the future work suggestions are given in Section 6.

2. Related work

Various heuristic and metaheuristic algorithms for VRP and its variants have been proposed so far. Heuristic algorithms are categorised into two groups, namely construction methods and improvement methods. While the former aims to build a solution from scratch by adding a single component at a time, the latter aims to improve an existing solution by modifying or rearranging its components. One of the famous construction heuristics is saving algorithm of Clarke and Wright [2] which builds a solution by merging tours according to a saving criterion. Sweep algorithm [3] is another widely used construction heuristic that it first creates clusters according to the polar coordinates of customers and then solves the travelling salesman problem per cluster. As for the improvement heuristics, λ-opt [4], Or-opt [5], and λ-interchange [6] are among the most common local search algorithms.

Although heuristics quickly find useful solutions, they tend to get stuck in local optima and needs a higher strategy to make further improvements on a solution. Thanks to their exploration abilities, metaheuristics are able to escape from local optima by exploring the search space more extensively. Genetic algorithms [7], tabu search [8], greedy randomized adaptive search procedure (GRASP) [9], evolution strategies [10], and large neighbourhood search [11] are examples of the main metaheuristics that have been applied to the VRP successfully so far in [12–16], respectively.

Navigation in multistorey buildings has been studied from different points of views. Zhang et al. [17] proposed a hierarchical path planner for autonomous robots using a multilayered topological map. In his doctoral dissertation [18], Coltin puts forward the pick-up and delivery problem with transfers and solved it for service robots called Cobots. Another study [19] is the least path algorithm, which considers the risk factors such as getting lost in an indoor space was applied to a multistorey building. Lin et al. [20] used a special file named Industry Foundation Classes (IFC), which includes semantic information as well as geometric information about building components, and find the shortest path by Fast Marching Method [21] after discretising the building space.

3. The multistorey VRP

In this paper, the term multistorey building has been used for any structure that has more than one storey and requires connections (e.g., stairs or lifts) between storeys to make it possible for travelling between any given two points. This study models storey surfaces as 2D Euclidean surfaces without any obstacle; hence, the shortest distance between two points on the same storey can be calculated using 2D Euclidean distance.

VRP in multistorey buildings (multistorey VRP) is different from the classical VRP because the search space is physically divided into layers. Thus, two new problem components, namely connection point and connection, are introduced in multistorey VRP. A connection point is kind of a vertex. However, unlike other vertices such as depot and customers, these are able to access other connection points on different storeys. On the other hand, connection is a logical term that is defined by linking two connection points and assigning a weight (cost) between them. In order to distinguish which connection points are connected with each other, we assign a unique number for each connection, and define the function $conn(x)$ as returning the corresponding number for the connection point x. Similarly, to distinguish which vertices are on which storeys, we assign a unique number for each storey, and define the function $stry(x)$ as returning the storey number of the vertex x.

Figure 1 demonstrates a building which has three storeys with 8 connections in total. The depot is on the ground storey and there are 2 customers on each storey. One possible solution consisting of two routes is illustrated by dashed lines. The route 1 starts from the depot, visits the customer on the left side, and goes to the second storey using the connection on the left side. After visiting two customers on the second storey, it returns back to the depot by using the connection on the back side. Similarly, the route 2 also starts from the depot, visits the customer on the right side, and goes directly to the third storey using the two connections on the right side. After visiting two customers on the third storey, it returns back to the depot by using the two consecutive connections on the front side. Note that the direction of the tours does not matter since the distances between points are considered as symmetric.

3.1. Mathematical formulation of the classical VRP

VRP can be described as follows. Let $G = (V, E)$ be a complete graph where $V = N \cup \{0\}$ is the vertex set and E is the edge set. Here, vertex 0 is the depot whereas vertices $N = \{1, 2, \cdots, n\}$ correspond to the customers. A tour $T = (0, v_1, v_2, \cdots, v_l, 0)$ is a cycle that starts from the depot, visits some of the customers $v_1, \cdots, v_l$, and returns to the depot again. Each customer $v_i \in N$ is associated with a demand d_i and each vehicle has

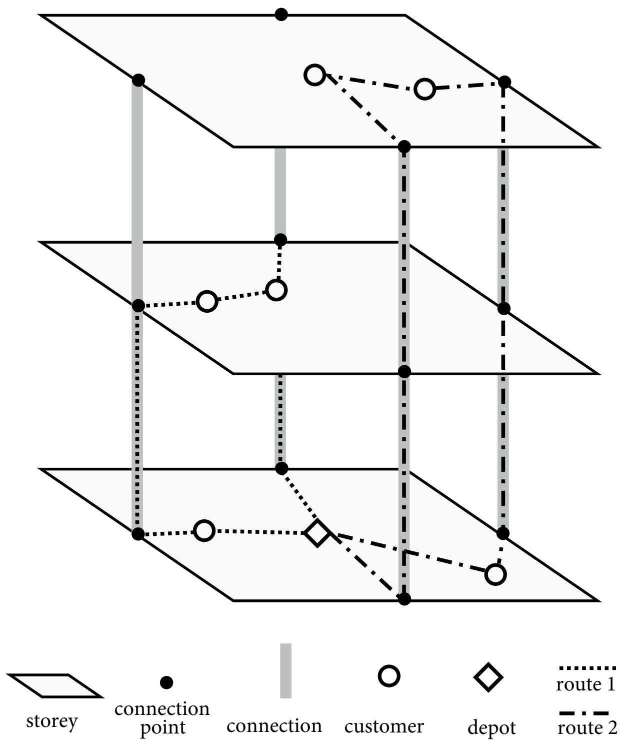


Figure 1. A 3-storey VRP instance with an example solution.

a capacity Q. A tour T is capacity feasible if inequality $\sum_{i=1}^{l} d_i \leq Q$ is satisfied for its customers. Also, a Euclidean distance $c_{(i,j)}$ for each edge $\{i,j\} \in E$ is calculated as its weight; hence, cost of a tour T can be calculated as $c_{0,v_1} + \sum_{i=1}^{l-1} c_{v_i,v_{i+1}} + c_{v_l,0}$. The main objective of the VRP is to find a set of capacity feasible tours that has the minimum cost in total while every customer is visited exactly once by one of the tours.

3.2. Mathematical formulation of the multistorey VRP

The multistorey VRP can be described as follows. Let $G_{MS} = (V_{MS}, E_{MS})$ be an incomplete graph where $V_{MS} = N \cup M \cup \{0\}$ is the vertex set and E_{MS} is the edge set. Here, vertex 0 is the depot, $N = \{1, 2, \cdots, n\}$ is the set of customers and $M = \{n+1, n+2, \cdots, n+m\}$ is the set of connection points. The edge set $E_{MS} = E_S \cup E_C$ consists of two subsets which are defined in (1) and (2) as follows:

$$E_S = \{(i,j) \in V_{MS} x V_{MS} \mid (i \neq j) \wedge stry(i) = stry(j)\}, \tag{1}$$

$$E_C = \{(i,j) \in M x M \mid (i \neq j) \wedge conn(i) = conn(j)\}, \tag{2}$$

where E_S is the edges between vertices on the same storey and E_C is the edges between connection points that form a connection. Weights of the edges in E_S are calculated by a Euclidean distance formula, whereas weights of the edges in E_C is determined by a problem designer.

Similar to the VRP, a tour $T = (0, v_1, v_2, \cdots, v_l, 0)$ is a cycle that starts from the depot, visits some of the customers and connection points and returns to the depot again. Each customer $v_i \in N$ is associated with a demand d_i and each vehicle has a capacity Q. For connection points $v_i \in M$, the demand d_i is set to 0. A tour T is capacity feasible if inequality $\sum_{i=1}^{l} d_i \leq Q$ is satisfied for its vertices.

A multistorey VRP solution consists of a set of capacity feasible tours that begins and ends at the depot by visiting customers and connection points. Like in classical VRP, all the customers must be visited exactly once by one of the resulting tours. However, this is not the case for connection points as they may be visited several times or never be used. Here, the set $V_R = N \cup \{0\}$ is defined as required vertices, while the set M is considered as optional. The main objective of the multistorey VRP is to find a set of capacity feasible tours that has the minimum cost in total while every customer in V_R is visited exactly once by one of the tours.

4. The problem-solving method

Algorithms to solve classical VRP are generally designed to operate on a complete graph that all vertices are connected with each other. However, multistorey VRP can only be represented by an incomplete graph. Figure 2 shows this difference with two examples. Figure 2a defines a 2-storey VRP for 5 customers. Here, connectivity between storeys is established by using the connection points 6 and 7. Thus, the graph representation of this example is incomplete. On the other hand, Figure 2b shows the classical VRP for 5 customers. Here, all customers and the depot is connected with each other. Hence, the graph representation of this example is complete.

To solve multistorey VRP using the existing algorithms, in the first step, the problem must be converted into the classical one by obtaining the complete graph from the incomplete one at hand. For this purpose, a data structure called shortest distance matrix, which is supposed to hold the shortest path lengths between each pair of required points (V_R), is calculated first. Thus, any algorithm that solves the VRP becomes capable of solving the multistorey VRP. In the second step, an optimisation algorithm solves the problem and outputs the final solution. Since the solution was produced from the shortest distance matrix and does not include connection points explicitly, in the third step, it is converted into the original multistorey version of the problem by inserting intermediate connection points in a correct order. The general process of this approach is explained in the following subsections.

4.1. Conversion of multistorey VRP to the classical VRP

The first part of the problem-solving process is to make the multistorey VRP solvable by the algorithms that have been used to tackle with the VRP. In this work, a graph of a multistorey VRP is represented by the distance matrix that is quite common in path planning problems. A formal description of a distance matrix for a multistorey VRP is given in Eq. (3) as follows:

$$DM(i,j) = \begin{cases} w(i,j), & \text{if } i \neq j, (i,j) \in E_{MS} \\ +\infty, & \text{if } i \neq j, (i,j) \notin E_{MS} \\ 0, & \text{if } i = j \end{cases} \tag{3}$$

where DM is the distance matrix, the vertices i and j are the elements of V_{MS}, (i,j) is the edge between vertex i and vertex j, $w(i,j)$ gives the weight of an edge (i,j) and E_{MS} is the edge set of the problem.

The goal of the conversion operation is to obtain the shortest distance matrix that includes shortest lengths between any pair of vertices of the set V_R. A formal description of a shortest distance matrix is given in Eq. (4) as follows:

$$SDM(i,j) = \begin{cases} sd(i,j), & \text{if } i \neq j, i \in V_R, j \in V_R \\ +\infty, & \text{if } i = j, i \in V_R, j \in V_R \end{cases} \tag{4}$$

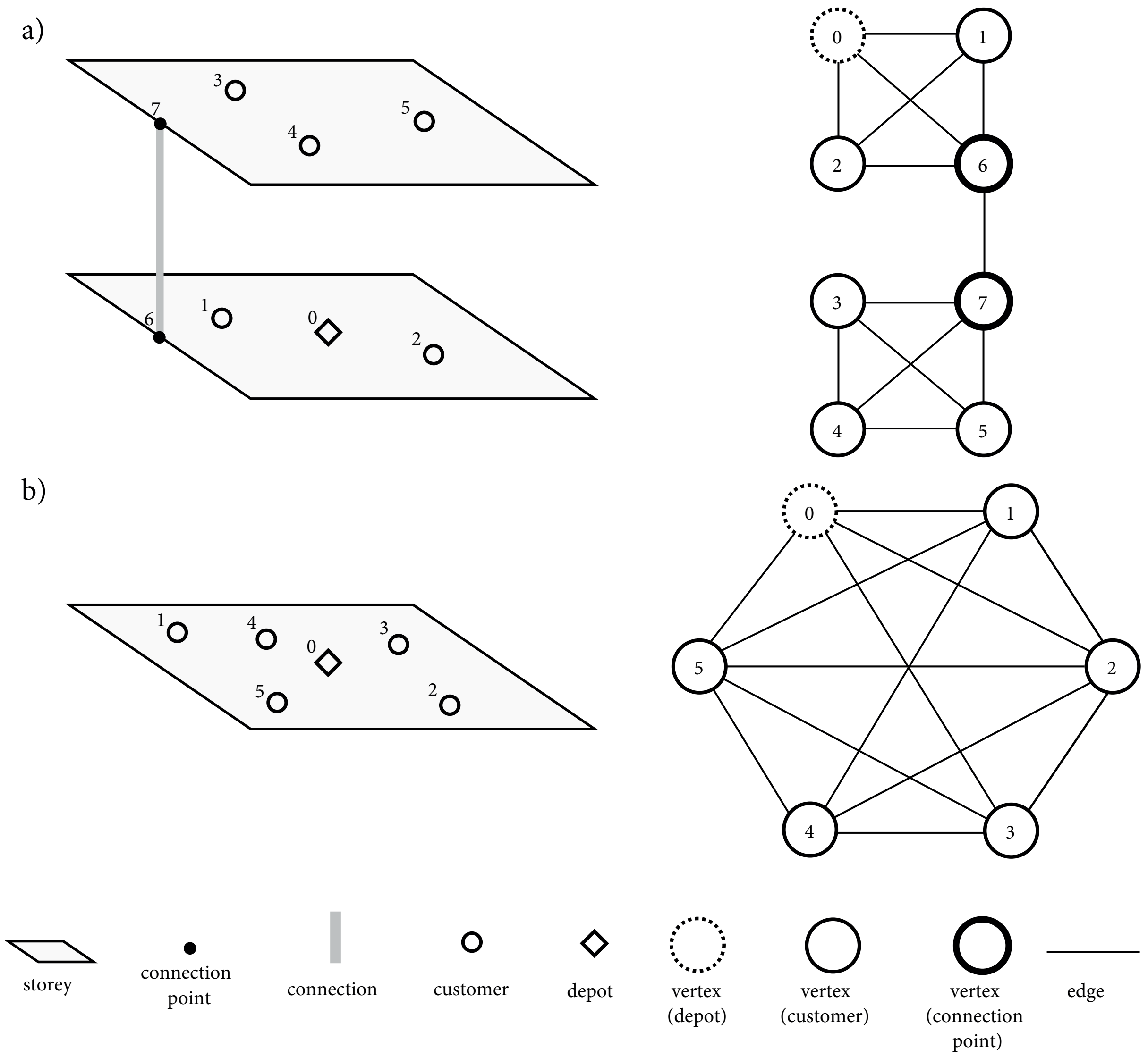
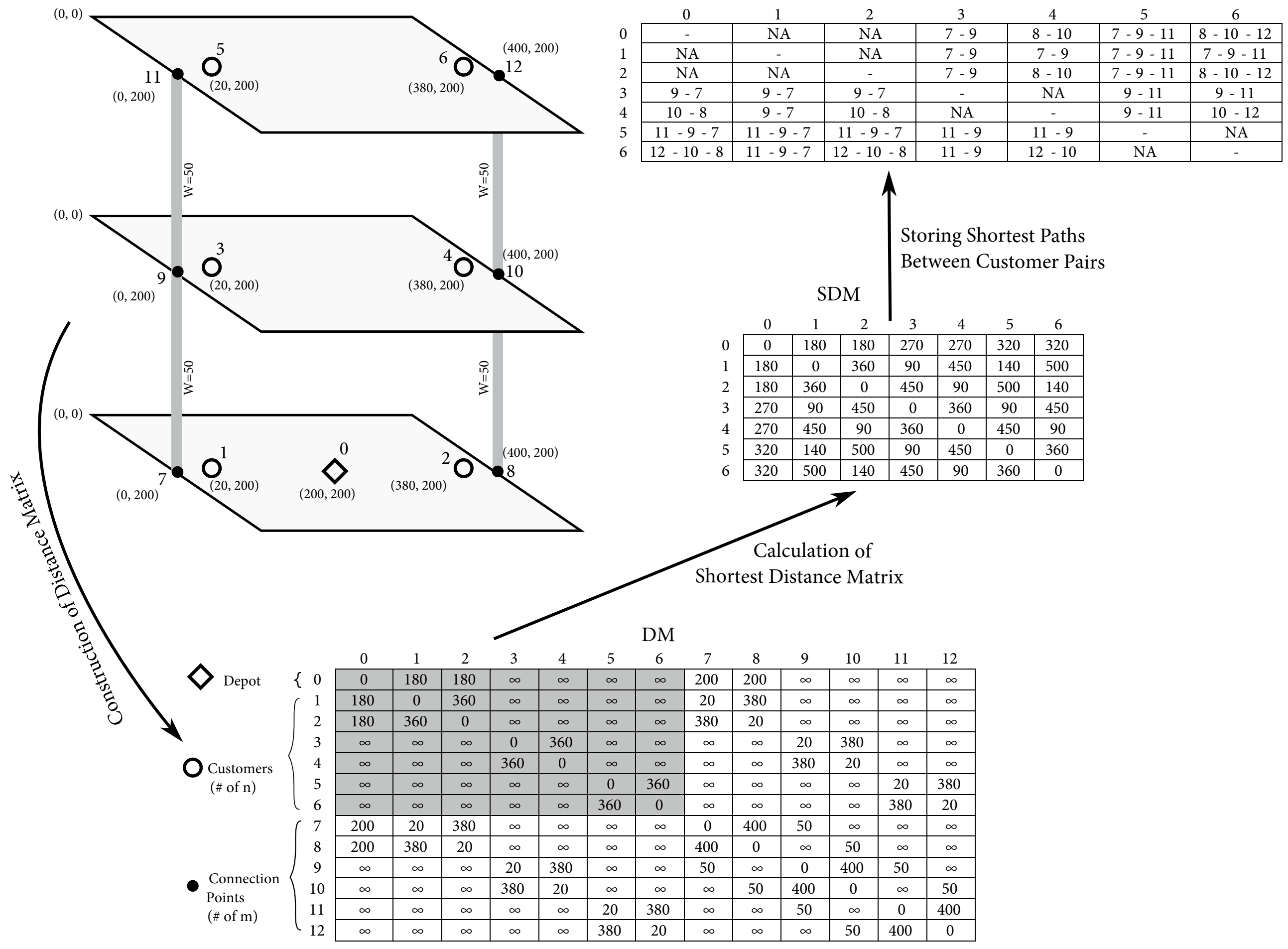


Figure 2. a) 2-storey problem example (left) and its incomplete graph representation (right), b) classical VRP example (left) and its complete graph representation (right).

where SDM is the shortest distance matrix, indices i and j are elements of the V_R and $sd(i,j)$ gives the shortest distance between vertex i and vertex j.

Shortest distance between any pair of vertices can be obtained from the SDM if both elements are in the V_R. Note that, after obtaining the SDM, existing edges between vertex i and vertex j will be updated if newly calculated shortest distance is smaller than the previous weight. Additionally, new edges are established between the vertices that are on different storeys.

A simple example is presented in Figure 3 to demonstrate the process of converting multistorey VRP to the classical VRP. In the first step, DM is constructed by using Eq. (3) by calculating the weights between nodes on the same storey using Euclidean distance. Note that the weights between connection points are assigned by a problem designer. As it is shown in the figure, size of the DM is $(n+m+1)\times(n+m+1)$, and it is organised as putting the depot at row 0, placing number of n customers between rows $[1,n]$ and adding m connections between rows $[n+1, n+m]$. Since shortest distances will only be calculated for the required nodes, or V_R, a shortest path algorithm is applied for every pair of distinct elements that are covered by the shortest distance matrix region that is shown with a grey rectangular area in DM (Figure 3). After calculation of shortest paths is done, SDM is obtained by extracting the region which takes part in the first $(n+1)$ rows and $(n+1)$ columns of the DM. It should also be mentioned here that the shortest paths themselves are stored in a different data structure since they will be referenced when adding intermediate connection points at the later stage.



	0	1	2	3	4	5	6
0	-	NA	NA	7 - 9	8 - 10	7 - 9 - 11	8 - 10 - 12
1	NA	-	NA	7 - 9	7 - 9	7 - 9 - 11	7 - 9 - 11
2	NA	NA	-	7 - 9	8 - 10	7 - 9 - 11	8 - 10 - 12
3	9 - 7	9 - 7	9 - 7	-	NA	9 - 11	9 - 11
4	10 - 8	9 - 7	10 - 8	NA	-	9 - 11	10 - 12
5	11 - 9 - 7	11 - 9 - 7	11 - 9 - 7	11 - 9	11 - 9	-	NA
6	12 - 10 - 8	11 - 9 - 7	12 - 10 - 8	11 - 9	12 - 10	NA	-

SDM

	0	1	2	3	4	5	6
0	0	180	180	270	270	320	320
1	180	0	360	90	450	140	500
2	180	360	0	450	90	500	140
3	270	90	450	0	360	90	450
4	270	450	90	360	0	450	90
5	320	140	500	90	450	0	360
6	320	500	140	450	90	360	0

DM

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	180	180	∞	∞	∞	∞	200	200	∞	∞	∞	∞
1	180	0	360	∞	∞	∞	∞	20	380	∞	∞	∞	∞
2	180	360	0	∞	∞	∞	∞	380	20	∞	∞	∞	∞
3	∞	∞	∞	0	360	∞	∞	∞	∞	20	380	∞	∞
4	∞	∞	∞	360	0	∞	∞	∞	∞	380	20	∞	∞
5	∞	∞	∞	∞	∞	0	360	∞	∞	∞	∞	20	380
6	∞	∞	∞	∞	∞	360	0	∞	∞	∞	∞	380	20
7	200	20	380	∞	∞	∞	∞	0	400	50	∞	∞	∞
8	200	380	20	∞	∞	∞	∞	400	0	∞	50	∞	∞
9	∞	∞	∞	20	380	∞	∞	50	∞	0	400	50	∞
10	∞	∞	∞	380	20	∞	∞	∞	50	400	0	∞	50
11	∞	∞	∞	∞	∞	20	380	∞	∞	50	∞	0	400
12	∞	∞	∞	∞	∞	380	20	∞	∞	∞	50	400	0

Figure 3. Illustration of a simple example of constructing distance matrix, calculating shortest distance matrix and storing shortest paths between customer pairs. Each storey is represented by a 400 × 400 square in 2D Euclidean space and all connection weights (W) are determined as 50

4.2. Optimisation by the iterated local search

Iterated local search (ILS) [22] metaheuristic is a simple yet effective algorithm that optimises a single solution by a series of perturbation and local search operations (See Algorithm 1). Perturbation procedure generates a new solution in the neighbourhood of the current solution. The level of the perturbation is critical; if it is very low then the algorithm may get stuck in a local optima; if it is very high then the algorithm may lost its search information and turns into a random search. Local search is another critical procedure of the algorithm; it should be strong enough to find near optimal solutions around the current one.

The implementation details of ILS algorithm in this study are presented as follows:

- Initial solution construction: We used parallel version of Clarke and Wright (CW) [2] construction heuristic for the initial starting solution generation. CW heuristic generates a solution by combining partial routes with each other according to the savings values.

- Perturbation: We used CROSS Exchange [23] neighbourhood structure where two random route segments from two random routes is exchanged. The length of the routes are also random and they may be at most L, which corresponds to the perturbation size.

- Local search: We used 6 different local search algorithms that are 2-opt, 2-opt*, swap, Or-Opt, string-exchange and string-exchange with segment inversion and applied them in random order while an improvement is observed (See Algorithm 2). These algorithms were implemented using a special technique called sequential search [24]. As sequential search allows pruning in the local search space, it runs faster than classical lexicographic search methods.

Algorithm 1: Iterated Local Search Metaheuristic

```
1  $S \leftarrow$ construct initial solution;
2  $S^* \leftarrow$ apply local search to $S$;
3  while termination condition is not satisfied do
4      $S' \leftarrow$ apply perturbation to $S^*$;
5      $S'' \leftarrow$ apply local search to $S'$;
6      if $S''$ is better than $S^*$ then
7          $S^* \leftarrow S''$;
8      end
9  end
10 return $S^*$;
```

To give a concrete example of how classical VRP solutions can be produced by the optimisation algorithm, we consider the following cases for the problem given in Figure 3.

(i) Demand of each customer = 10 and capacity of vehicles ≥ 60: In this case, capacity of one vehicle is sufficient for travelling all the customers. Therefore, the algorithm could produce one route {0-2-4-6-5-3-1-0} with the total cost of 1080.0.

(ii) Demand of each customer = 10 and capacity of vehicles = 30: In this case, capacity of one vehicle is not sufficient for travelling all the customers. Therefore, the algorithm could produce 2 routes {0-1-3-5-0} and {0-2-4-6-0} with the total cost of 1360.0.

(iii) Demand of each customer = 10 and capacity of vehicles = 20: In this case, capacity of one vehicle is not sufficient for travelling all the customers. Therefore, the algorithm could produce 3 routes {0-3-5-0}, {0-4-6-0} and {0-1-2-0} with the total cost of 2080.0.

Algorithm 2: Local search for ILS

```
   input: A solution $S$ to be improved
1  $localSearches$ ← {2-opt, 2-Opt*, swap, Or-Opt, string-exchange, string-exchange with segment
     inversion};
2  repeat
3      $improved$ ← false;
4      repeat
5          $selectedLS$ ← select a local search randomly from $localSearches$ that has not been applied
             so far in this loop ;
6          apply $selectedLS$ to $S$;
7          if $selectedLS$ made an improvement over $S$ then
8              $improved$ ← true;
9          end
10     until every local search is applied once;
11 until $improved = false$;
12 return $S$;
```

The case (i) above corresponds to the well-known travelling salesman problem since one route can visit all of the customers. As capacity of vehicles decrease, generally, the number of routes and the total cost of a problem increase. On the other hand, assuming that the multiple vehicles work in parallel, the completion time of visiting operation is bounded with the longest tour. Thus, even if the total cost is high, it is possible to visit all customers in less time.

4.3. Conversion of classical VRP solution to the multistorey solution

When the algorithm produces its output as VRP solution, it only includes the order of customers to be visited. However, the solution of the original multistorey problem is expected to include the connection points and in which order they are used. Thus, the obtained VRP solution should be extended to include the intermediate connection points between any consecutive customers if it is required. In order to achieve this operation, the shortest paths tracks must also be stored when calculating the shortest distance matrix and used for specifying paths between customers.

In order to give an example of how classical VRP solution is converted to the multistorey solution, we use again the problem shown in Figure 3. Using the table for the shortest paths between customer pairs table, we can convert the classical solution consisting of 3 routes, which was given at the subsection 4.2, as follows.

- Classical VRP route 1: {0-3-5-0}

  Multistorey VRP route after adding intermediate connection points: {0-**7-9**-3-**9-11**-5-**11-9-7**-0}

- Classical VRP route 2: {0-4-6-0}

  Multistorey VRP route after adding intermediate connection points: {0-**8-10**-4-**10-12**-6-**12-10-8**-0}

- Classical VRP route 3: {0-1-2-0}

  Multistorey VRP route (in this case intermediate connections nodes are not available): {0-1-2-0}

5. Experimental work

In this study, the experimental work is divided into two parts. The first part includes the preliminary computational study that tries to evaluate the performance of the ILS optimisation algorithm. For this purpose, we employ a classical VRP benchmark of Christofides et al. [25], which is one of the most used datasets in the literature. Because multistorey buildings can vary in storey count, storey size and connection count, in the second part of the experiments, we investigate the effect of the multistorey building structure on the performance of the optimisation algorithm. In addition, two other important VRP parameters, customer count and the depot location, are included in these experiments. Considering all of these building structures and VRP parameters, we generated and solved several multistorey VRP test instances, and discussed the results obtained.

5.1. The performance evaluation of the ILS optimisation algorithm

The optimisation algorithm was implemented in JAVA and all experiments were performed on a desktop PC with Intel(R) Core(TM) i7-6700 CPU and 64-bit operating system.

The implemented ILS algorithm has 3 parameters that are perturbation size (ps), best improvement (bi) and maximum string length (λ). The former is the main parameter of the ILS and controls the mutation strength in the perturbation step. The second one determines whether the best improvement search strategy is applied or not in local search algorithms. Finally, the last parameter sets the maximum length of the strings in Or-Opt and string-exchange algorithms.

In this work, we set $bi = false$ and $\lambda = 3$ as it is suggested in [14] for the small- and medium-sized problem instances. After the preliminary experiments we decided to set $ps = 8$. To show the effectiveness of the provided ILS algorithm, we conducted an experiment on 14 classical VRP benchmark problem instances of Christofides et al. [25] using the parameter values mentioned above. In this experiment, the algorithm was ran 25 times per problem instance. The running budget was set as 20,000 local search calls per run. The results obtained are shown in Table 1. The first column includes the problem instances along with their sizes. The second, third, fourth, and fifth columns give the average, best, worst, and standard deviation of the results, respectively. The column 6 lists the average CPU running times per problem instance. Finally, the column 7 lists the best-known solutions so far for the benchmark functions.

Table 1 shows that the optimisation algorithm found optimal solutions for 4 problems in all runs. The algorithm also found optimal solutions for 10 problems at least once in one of the 25 runs. The general performance of the implemented ILS algorithm can be evaluated by looking at the average deviation values in percentage to the best-known solutions for 14 problems. Considering the results of 25 runs, the average deviation value is calculated as 0.39%, which seems good enough to use the implemented algorithm for solving the VRP.

5.2. Generation of test problems for multistorey VRP

In this section, definition of the test cases and test problems generation are given. To generate test cases, some characteristic problem parameters including the number of storeys (nos), the number of connections ($nocn$), the number of customers (noc), storey width (sw) and the storey number of the depot (dsn) are determined. Definition of these main problem parameters are listed below:

- nos is the storey count in the building.

Table 1. Performance of the ILS algorithm on Christofides et al. benchmark problems.

Problem(n)	Avg.	Best	Worst	Std.	Avg. CPU(s)	BKS
1(50)	524.61	524.61	524.61	0.00	16.17	524.61
2(75)	838.19	835.26	842.48	2.19	30.70	835.26
3(100)	829.63	826.14	829.63	1.10	62.04	826.14
4(150)	1029.26	1028.42	1039.19	2.10	82.25	1028.42
5(199)	1306.29	1295.17	1313.37	5.75	136.07	1291.29
6(50)	556.51	555.43	565.56	2.42	12.65	555.43
7(75)	915.73	912.47	927.55	5.30	23.23	909.68
8(100)	871.05	865.94	874.9	4.15	37.14	865.94
9(150)	1172.15	1162.55	1195.55	9.34	57.13	1162.55
10(199)	1410.90	1400.74	1427.67	7.11	85.51	1395.85
11(120)	1042.12	1042.12	1042.12	0.00	148.25	1042.12
12(100)	819.56	819.56	819.56	0.00	74.38	819.56
13(120)	1544.76	1542.86	1547.31	1.45	87.21	1541.14
14(100)	866.37	866.37	866.37	0.00	71.69	866.37
Avg. Dev. %	0.39	0.08	1.01	–	–	–

- *nocn* is the number of connections between each adjacent storey, e.g., if the *nocn* is 2 for 4-storey building, there will be 6 connections in total.
- *noc* is the total number of customers in the building and it is divided equally between storeys, e.g., if the *noc* is 64 in a 4-storey building, there will be 16 customers on each storey.
- *sw* is the width of the storeys that are in a square form, e.g., if the *sw* is 400 in a 4-storey building, the surface area of each storey is 400 $\times$ 400 = 160,000 unit^2 and the surface area of the all storeys is 160,000 $\times$ 4 = 640,000 unit^2.
- *dsn* is the storey number of the depot on which it is placed.

This study uses 3 test cases to evaluate multistorey VRP by trying different combinations of main problem parameters. Test case 1 focuses on parameters *nocn*, *noc* and *nos* while the depot is always put on the centre of the ground storey. Here, the storey width is also fixed. In test case 2, total area of the building is fixed in addition to the parameters *nocn* and *dsn*. It tries different values of parameters *nos* and *noc* and let *sw* to be vary as the number of storeys changes. Finally, the test case 3 aims to analyse the effect of depot location. To this end, the depot is put on every single storey one by one (different *dsn* values) while the other problem parameters *sw*, *nocn*, *noc* and *nos* are fixed. Table 2 lists the parameter values used for each test case. If a parameter takes more than one value, it is separated by commas. Other important configurations for test problems generation are listed below:

- Vehicle capacities are set to 100.
- Customer demands are assigned to be a random value between [1, 20].
- Weights of the connections are set to 20.

- Depot is located at the centre of the storey.
- Connection points are placed at certain locations depending on their numbers so that they are at equal distances from each other (Figure 4).
- The locations of customers are determined by Halton sequence generator that produces quasi-random numbers to cover the storey surface more evenly

Table 2. Problem parameters of the test cases. *sw*: storey width, *nocn*: number of connections, *noc*: number of customers, *nos*: number of storeys, *dsn*: storey number of the depot.

Test cases	sw	nocn	noc	nos	dsn
Case 1	400	1,2,4,8	8,16,32,64	1,2,4,8	1
Case 2	$\sqrt{640,000/nos}$	4	8,16,32,64	1,2,4,8	1
Case 3	400	4	120	6	1,2,3,4,5,6

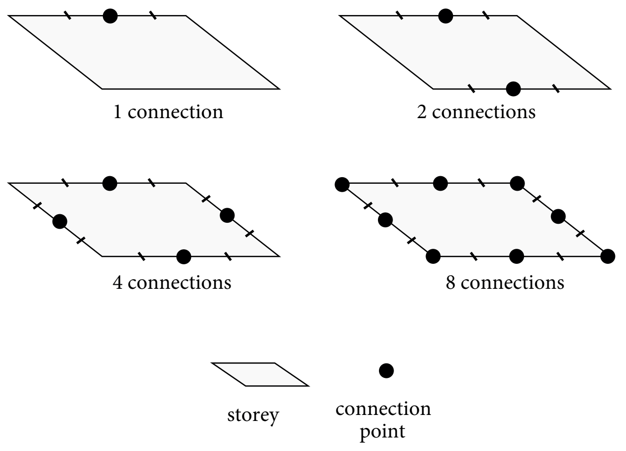


Figure 4. Placement of connection points in multistorey problem generation.

5.3. Computational results for multistorey VRP test cases

This part of the experimental study presents the computational results for the generated test problems. Problem instances were constructed according to the 3 test cases which are explained in Section 5.2. To get more accurate results, each problem instance was generated and solved for 25 times with different random seeds. Then, the average solution values were calculated per problem instance.

Table 3 gives the average cost values per problem instance of test case 1. It can be seen that the average cost value increases when the number of storeys increases. In fact, it is at the minimum level for the 1-storey case, which is also equivalent to the classical VRP. The reason is that the usage of connections adds additional cost as proportional to their connection weights and their distances from source and target points. On the other hand, increasing number of connections helps the optimisation algorithm to find low cost solutions. Because it increases the chances of finding a closer connection between pairs of points. Figure 5 shows this behaviour in detail. It can be seen that adding extra connections reduces the overall costs more when the number of customers is small than it is big. Also, this reduction is higher when the number of storeys is large that it is small. The reasons are that the rate of connection costs in total cost will be high if the number of connections is small and number of storeys is large.

Table 3. Average cost values per problem instance for Case 1.

		nocn=1	*nocn*=2	*nocn*=3	*nocn*=4
noc=8	*nos*=1	1296.13	1279.08	1272.15	1293.56
	nos=2	2054.85	1777.09	1734.36	1703.30
	nos=4	3137.17	2472.68	2156.10	2162.29
	nos=8	4299.37	3034.56	2684.24	2651.85
noc=16	*nos*=1	1771.24	1773.78	1789.31	1768.04
	nos=2	2934.22	2836.73	2625.68	2537.81
	nos=4	4481.52	3942.53	3501.28	3397.19
	nos=8	6819.50	5359.89	4514.76	4418.81
noc=32	*nos*=1	2603.24	2607.39	2615.97	2573.50
	nos=2	4566.00	4195.56	4053.79	3871.94
	nos=4	6389.44	6082.31	5735.77	5327.10
	nos=8	9565.50	8514.96	7498.37	7174.36
noc=64	*nos*=1	4036.13	4078.38	4035.78	4059.44
	nos=2	7333.67	6510.76	6327.97	6153.63
	nos=4	10283.91	9334.03	8892.24	8456.82
	nos=8	13828.45	13173.08	12532.07	11709.10

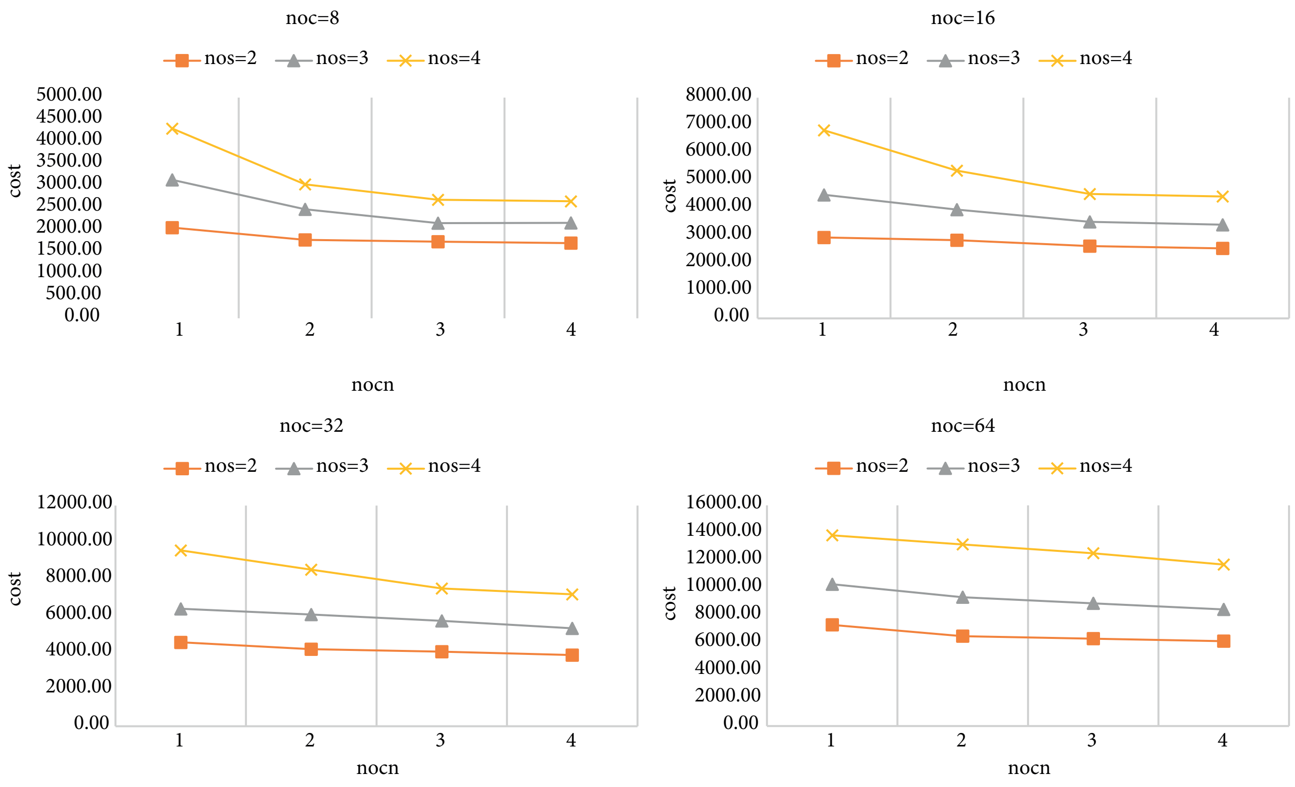


Figure 5. Effect of the number of connections per customer count.

Figure 6 demonstrates the results for the test case 2 which fixes the total surface area regardless of the number of storeys. Contrary to the test case 1, costs do not always grow as the number of storeys increases. Instead, the trend of the cost differs in the number of customers. For fewer customers, costs tend to decrease whereas for many customers it tends to increase. As the number of storeys increases, firstly, the storey widths get smaller and the average distance between each pair of customers reduces, and secondly, the usage of connections increases. The former situation helps to decrease the travelling costs whereas the latter one has the opposite effect. The average route count will be relatively small for fewer customers than the more customers and so the former condition dominates the latter. For the case of many customers, this will be vice versa.

Finally, Figure 7 shows the cost values for the changing depot locations. Unlike the previous test cases, the depot is not always placed on the ground storey, instead, it is removed and placed on another untried storey at each iteration. It is seen that the cost of the solution tends to decrease as the depot is shifted to the middle storeys. In fact, it takes the minimum value in the median storey (or one of the median storeys if the number of storeys is even). The reason is that the average use of connections decreases as the depot gets close to the middle storeys.

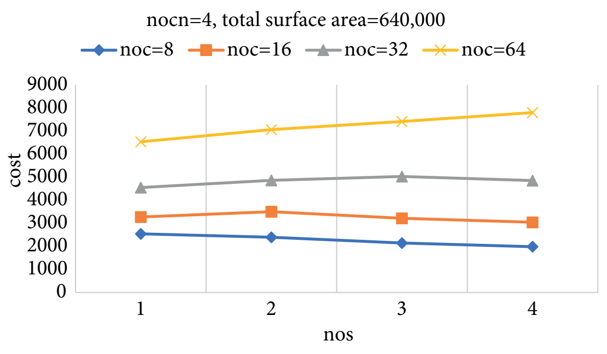


Figure 6. Effect of the number of storeys when the total surface area is fixed.

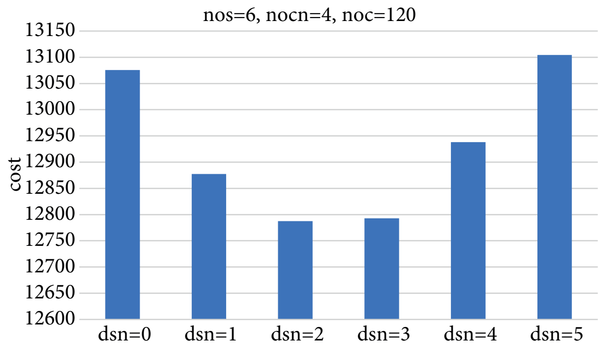


Figure 7. Effect of placing the depot on different storeys.

5.4. Comparison of the proposed method with dedicated vehicle for each storey

In this subsection, we show the advantage of multistorey problem solving strategy in comparison with the dedicated vehicle approach that assigns one vehicle per storey. Dedicated vehicle approach is a straightforward technique that does not exploit the alternative transition points between storeys. Instead, it tries to solve each storey as a separate problem and then combine individual solutions into a final single solution. Although this problem solving method seems simpler and does not require the conversion procedures explained before, it is likely to omit better routes that uses connections between adjacent storeys.

In order to see the difference between these two approaches, we designed an experiment on a 16-storey building with 10 customers per storey. Other configurations are listed as follows: each customer has 5 units of demand, storeys are connected with each other by 4 connections with 20 units of weight, and depot is located at the center of the ground storey. Table 4 gives the results of 25 runs for each algorithm. In the dedicated vehicle approach, each vehicle serves 10 customers with the total demand of 50. Thus, vehicle capacities must be minimum $Q = 50$ to fulfill the capacity constraint. When the same vehicle capacity is applied to the multistorey approach, the final cost was improved by 1.72% even though approximately the same number of routes were produced.

In addition, dedicated vehicle approach has another drawback when the vehicle capacity exceeds the total customer demand. Because the vehicle cannot travel between storeys, its remaining capacity is wasted. To see this phenomenon, the vehicle capacities were increased to 100 and the problem was re-solved for multistorey case. The last column of the Table 4 shows that the total cost was improved by 19.60% in this way.

Table 4. Comparison of our multistorey VRP solving method with dedicated vehicle approach.

Type of comparison	Dedicated vehicle approach	Multistorey approach (Q=50)	Multistorey approach (Q=100)
avg. cost	31,105.81	30,569.53	25,010.94
avg. route count	16	16.84	8.48
improvement % (avg. cost)	-	1.72	19.60

6. Conclusion

This paper has introduced and formulated the VRP in multistorey buildings that is called multistorey VRP. After a solution method has been presented to employ existing VRP optimisation algorithms for the proposed problem, it has been solved by the ILS optimisation algorithm. The second important contribution of this paper is that it conducted several computational experiments that address different aspects of the multistorey VRPs. For this purpose, three test cases have been defined and the results were obtained. It has been shown that how the number of customers, number of storeys, number of connections between the storeys, the location of the depot, and the total area of all storeys of the building influence the cost of a final solution.

As the population of people grows in the world, the number of vertical cities become more common and the number of high-rise buildings increases. This phenomenon increases the significance of solving the optimisation problems in the multistorey buildings. As one of the most important optimisation problems, the VRP has been seen valuable to be considered in this context because many people live or work in high-rise buildings and they have several demands to be fulfilled (postal delivery, transfer of goods, etc.). To that end, a group of employees or automated robots with specified carrying capacities can be used as in VRP. Although there have been many studies on the classical VRP, which can be classified as horizontal, almost no work has been done for the vertical case. This work will help to fill the gap in this area and increase the applicability of the problem to the buildings in the real world. Although this study has targeted the basic version of the problem, there exist many variants of the VRP that have arisen from real world needs. Thus, this study can be extended to cover time window constraints, stochastic demands, pick-up operations, multiple depots, etc. for further studies.

Acknowledgment

Author Osman Gökalp has been supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) with 2211 National Graduate Scholarship Program.

References

[1] Dantzig GB, Ramser JH. The truck dispatching problem. MANAGE SCI 1959; 6: 80-91.

[2] Clarke G, Wright JW. Scheduling of vehicles from a central depot to a number of delivery points. OPER RES 1964; 12: 568-581.

[3] Gillett BE, Miller LR. A heuristic algorithm for the vehicle-dispatch problem. OPER RES 1974; 22: 340-349.

[4] Lin S. Computer solutions of the traveling salesman problem. The Bell System Technical Journal 1965; 44: 2245-2269.

[5] Or I. Traveling Salesman-type combinatorial problems and their relation to the logistics of blood banking. PhD, Northwestern University, Evanston, USA, 1976.

[6] Osman IH. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. ANN OPER RES 1993; 41: 421-451.

[7] Holland JH. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Cambridge, MA, USA: MIT Press, 1992.

[8] Glover F. Tabu search part i. ORSA Journal on computing 1989; 1: 190-206.

[9] Feo TA, Resende MGC. Greedy randomized adaptive search procedures. Journal of global optimization 1995; 6: 109-133.

[10] Rechenberg I. Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuttgart, Germany: Frommann-Holzboog, 1973.

[11] Shaw P. Using constraint programming and local search methods to solve vehicle routing problems. In: International conference on principles and practice of constraint programming. Pisa, Italy. Berlin, Heidelberg: Springer Berlin Heidelberg; 1998. pp. 417-431.

[12] Potvin JY, Duhamel C, Guertin F. A genetic algorithm for vehicle routing with backhauling. APPL INTELL 1996; 6: 345-355.

[13] Toth P, Vigo D. The granular tabu search and its application to the vehicle-routing problem. INFORMS J COMPUT 2003; 15: 333-346.

[14] Prins C. A GRASP x Evolutionary Local Search Hybrid for the Vehicle Routing Problem. In: Pereira FB,Tavares J, editors. Bio-inspired Algorithms for the Vehicle Routing Problem. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. pp. 35-53.

[15] Mester D, Bräysy O. Active-guided evolution strategies for large-scale capacitated vehicle routing problems. COMPUT OPER RES 2007; 34: 2964-2975.

[16] Pisinger D, Ropke S. A general heuristic for vehicle routing problems. COMPUT OPER RES 2007; 34: 2403-2435.

[17] Zhang Q, Wu X, Liu B, Adiwahono AH, Dung TA, Chang TW. A hierarchical topological planner for multi-storey building navigation. In: 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM). Busan, South Korea. New York, NY, USA: IEEE; 2015. pp. 731-736.

[18] Coltin B. Multi-agent pickup and delivery planning with transfers. PhD, Carnegie Mellon University, Pittsburgh, USA, 2014.

[19] Vanclooster A, Ooms K, Viaene P, Fack V, Van de Weghe N, De Maeyer P. Evaluating suitability of the least risk path algorithm to support cognitive wayfinding in indoor spaces: an empirical study. APPL GEOGR 2014; 53: 128-140.

[20] Lin YH, Liu YS, Gao G, Han XG, Lai CY, GU M. The ifc-based path planning for 3d indoor spaces. Adv Eng Inform 2013; 27: 189-205.

[21] Sethian JA. A fast marching level set method for monotonically advancing fronts. P Natl a Sci India A 1996; 93: 1591-1595.

[22] Lourenço HR, Martin OC, Stutzle T. Iterated local search. In: Glover F,Kochenberger G (editors). Handbook of Metaheuristics. Boston, MA, USA: Springer US, 2003. pp. 320-353.

[23] Taillard E, Badeau P, Gendreau M, Guertin F, Potvin JY. A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science 1997; 31: 170-186.

[24] Irnich S, Funke B, Grünert T. Sequential search and its application to vehicle-routing problems. Comput Oper Res 2006; 33: 2405-2429.

[25] Christofides N, Mingozzi A, Toth P. The vehicle routing problem. In: Christofides N, Mingozzi A, Toth P, Sandi L, editors. Combinatorial Optimization. Chichester, UK: Wiley, 1979, pp. 315-338.