

**International Journal of Logistics Systems and Management**

ISSN online: 1742-7975 - ISSN print: 1742-7967  
<https://www.inderscience.com/ijlsm>

---

**An innovative method to solve classic and dynamic multi-depot vehicle routing problem**

Reza Rahimi Baghbadorani, Mohammad Haeri

**DOI:** [10.1504/IJLSM.2022.10049733](https://doi.org/10.1504/IJLSM.2022.10049733)

**Article History:**

|                   |                 |
|-------------------|-----------------|
| Received:         | 18 January 2022 |
| Accepted:         | 09 June 2022    |
| Published online: | 13 January 2025 |

---

## An innovative method to solve classic and dynamic multi-depot vehicle routing problem

---

Reza Rahimi Baghbadorani and  
Mohammad Haeri\*

Advanced Control Systems Lab,  
Sharif University of Technology,  
Tehran, Iran  
Email: reza.rahimi@alum.sharif.edu  
Email: haeri@sharif.ir  
\*Corresponding author

**Abstract:** A significant part of distribution companies' final interest and cost is related to delivering goods to customers. Due to constraints on service hours and service distances, logistics companies usually build multiple depots to serve a large number of dispersed customers. This is why multi-depot vehicle routing problems (MDVRP) win more and more companies' attention to thrift their cost. In this research, first, we address a classic form of MDVRP and ponder its solution. A tuning parameter is introduced to trade-off between the final and computation costs. Then, considering issues like traffic congestion variations, increasing or decreasing number of customers, and other uncertainties, dynamic MDVRP is analysed. A receding horizon approach is proposed, and its performance in three scenarios is investigated. Simulation results show that the proposed algorithm is feasible and efficient to solve the dynamic MDVRP.

**Keywords:** vehicle routing problem; VRP; classic MDVRP; receding horizon; dynamic MDVRP; capacitated clustering.

**Reference** to this paper should be made as follows: Rahimi Baghbadorani, R. and Haeri, M. (2025) 'An innovative method to solve classic and dynamic multi-depot vehicle routing problem', *Int. J. Logistics Systems and Management*, Vol. 50, No. 1, pp.28–50.

**Biographical notes:** Reza Rahimi Baghbadorani has earned his MSc in Control Engineering from the Sharif University of Technology, Iran. His area of research is control and optimisation. He has also several research papers published in national and international conferences in the fields of control, operation research, and transportation. He is currently serving as a PhD student at the Delft University of Technology, Netherlands.

Mohammad Haeri received his BSc and MSc both in Electrical Engineering from Amirkabir University of Technology in 1985 and 1988 respectively. He got his PhD in Electrical Engineering from University of Saskatchewan, Canada in 1994. Since 1994, he is with Electrical Engineering Department, Sharif University of Technology where he is now a Professor and Head of the Advanced Control System Lab. His research area includes theory and application of model predictive control, networked systems, applications of fractional calculus in engineering, and modelling of physiological systems.

This paper is a revised and expanded version of a paper entitled ‘A new approach to solve MDVRP in lower computation time’ presented at 2021 29th Iranian Conference on Electrical Engineering (ICEE); ‘A novel two-phase approach to solve multi-depot vehicle routing problem’ presented at 2021 25th International Conference on System Theory, Control and Computing (ICSTCC) and ‘Model predictive based approach to solve DVRP with traffic congestion’ presented at Control Conference Africa (CCA) 2021, Magaliesberg, South Africa, 7–8 December 2021.

---

## **1 Introduction**

Without any cast of doubt, vehicle routing problem (VRP) plays a crucial role in logistics, transportation, and other pertinent fields. Regarding ultimate cost and interest, besides concomitant efficacy has to do with environmental issues and increasing greenhouse gases related to VRP, this problem has a considerable share in the quandaries mentioned above (Reyes et al., 2017; Moghdani et al., 2020). Therefore, routing problem has drowned most researchers’ and engineers’ attention in different fields. The VRP is an optimisation problem involves with some customers distributed geographically and a fleet of vehicles to service them. Solving this problem aims to serve all customers at a minimum cost (Goel and Maini, 2017). This cost involves variety of issues based on the problem, such as driving distance, shipping time, and fuel consumption (Zhang et al., 2020; Gil et al., 2020). Considering the nature and application of the problem, VRP is divided into several types with relevant constraints (Kumar and Panneerselvam, 2014; Wang et al., 2020); among them capacitated vehicle routing problem (CVRP) that a fleet of vehicles in a depot serves customers with capacity constraint containing vehicles’ capacity and customers’ need, vehicle routing problem with time window (VRPTW) in which fleet of vehicles in a depot serve customers in a specific predetermined range of time, and multi-depot vehicle routing problem (MDVRP) which is similar to CVRP with the difference that due to the geographical extent of customers, several depots are used to serve customers, are the most significant in the vanguard. In the most cases of VRP, the cost between two nodes is considered fixed and permanent with time passing.

VRP and its variants can be considered as an extension to the travelling salesman problem (TSP) (Contardo and Martinelli, 2014); therefore, with this in mind and NP-hardness of TSP, it is concluded that VRP and its variants could be categorised as NP-hard problem as well. This means that finding an optimal solution for VRP in high-dimensional problems is not practical because of exponentially increasing computational complexity. In this scene, although applying the exact methods such as branch and bound, and branch and cut (Khoufi et al., 2019) is reasonable to solve small VRP but considering the problem in real world circumstances including number of customers, using these methods would not be thriftily and cost-effective. Hence, most approaches used to solve VRP and its variants are either heuristic or meta-heuristic (Dixit et al., 2019; Rabbani et al., 2017; Azad and Hasin, 2019). The most important heuristic and meta-heuristic algorithms, inspired by biological or natural phenomena, include genetic algorithm, particle swarm optimisation, ant colony, tabu search, and simulated annealing. Although these methods do not offer guarantee of optimality, but high application value as the good solutions could be generated within a reasonable time.

However, the consuming time could be increased rapidly as the dimensional of problem increases. Also, they have several tuning parameters that should be selected properly with respect to the problem in hand (Rahimi Baghbadorani et al., 2021).

In real world applications, most of the logistics and transportation problems are essentially dynamic. Only parts of information are available which could be changed by time as well. The reasons in forefront causing dynamism are changes in traffic congestion during the day, car accidents and breakdowns, and increasing or decreasing number of customers to name a few. These facts make the VRP challenging and introduce a new type of VRP called dynamic vehicle routing problem (DVRP). At least one of the factors changes in DVRP and the final cost is affected in accordance (Yu and Yang, 2019). Also, only initial information about the cost and conditions is available. As time passing, these conditions may change; therefore, the problem should be solved again to determine an apt solution. Similar to VRP and its variants, DVRP is an NP-hard problem as well, and most approaches used to solve it are either heuristic or meta-heuristic. It is worth mentioning that in a DVRP the main issue is the running time, thus one needs a fast algorithm to perform new computations whenever it is required. The most focus in this research is on the approach to solve the static problem with reasonable cost and in single iteration opposed to the heuristic methods needing to solve the MDVRP iteratively. This subject is revealed in a real-time problem such as a dynamic MDVRP in which one needs fast algorithms. It is worth mentioning that a few numbers of articles discussed this subject. These works define dynamic MDVRP as an MDVRP in which only the number of customers varies over the time. From this point of view, since our definition of dynamism includes all uncertainties and prospective variations, according to the best knowledge of authors, the present work is considered new where the subject is investigated in general.

In continuance, first, we analyse the classic MDVRP and present a two-stage approach to solve the problem in less computation time and reasonable cost. Also, we introduce a tuning parameter ( $\alpha$ ) to trade-off between the final cost and consuming time in an MDVRP. Then we combine the proposed approaches for solving classic MDVRP and receding horizon feature of the model predictive control to find solution for dynamic MDVRP. In our proposed approach, vehicles serve customers based on their initial data and known traffic model, then by updating the data in specific intervals a new clustering of vehicles is performed if there are notable changes in the condition for a given prediction horizon. Finally, the optimum solution in light of the prediction horizon is computed and implemented.

The rest of the paper is organised as follows. Section 2 describes the classic MDVRP, and Section 3 defines the methodology to solve this problem. The dynamic MDVRP and proposed algorithm for solving it are presented in Section 4. Simulation results, including the benchmark instances, are presented and discussed in Section 5, containing effectiveness evaluation and comparisons with results from the other approaches. The conclusions are given in Section 6.

## 2 Classic MDVRP

An MDVRP could be defined as graph  $G = (V, A)$ , where  $V = \{1, 2, \dots, n_c + n_d\}$  and  $A = \{(i, j): i, j \in V, i \neq j\}$  are the set of vertices and edges, respectively. The vertex set  $V$  is divided into two subsets  $V^c = \{1, 2, \dots, n_c\}$  and  $V^d = \{n_c + 1, n_c + 2, \dots, n_c + n_d\}$  representing the set of customers and depots. In each depot, there are maximum  $K$

vehicles with capacity  $Q$ . Also, each customer has a specific demand  $q_i (i \in V^c)$ .  $F = [f_{ij}]$ ,  $i, j \in V$ ,  $f_{ij} = f_{ji}$  depicts the cost matrix explaining the cost between every two nodes in  $A$  (Salhi et al., 2014). An MDVRP aims to minimise the vehicles' routes such that the primary and ultimate depot of each vehicle is the same, all customers are served, each customer met only once, and the customers' demands which are served by a vehicle do not exceed the vehicle capacity.

The following relations explain an MDVRP by gratifying the required constraints.

$$\min J = \sum_i \sum_j \sum_k b_{ij}^k f_{ij} \quad (1)$$

Subject to

$$\sum_i \sum_j b_{ij}^k q_j \leq Q, \forall k \in K, \quad (2)$$

$$\sum_k \sum_j b_{ij}^k = 1, \forall i \in V^c, \quad (3)$$

$$\sum_k \sum_i b_{ij}^k = 1, \forall j \in V^c, \quad (4)$$

$$\sum_k \sum_i b_{ii}^k = 0, \forall i \in V, \quad (5)$$

$$\sum_k \sum_i b_{ij}^k = \sum_k \sum_j b_{ji}^k, \forall j \in V^d, \quad (6)$$

$$\sum_k \sum_j b_{ij}^k \geq 1, \forall i \in V^d, \quad (7)$$

$$\sum_j b_{ij}^k \leq 1, \forall k \in K, \forall i \in V^d, \quad (8)$$

$$\sum_k \sum_{i,j} (b_{ij}^k + b_{ji}^k) \leq 1, i, j = V^c, \quad (9)$$

$$\sum_j b_{ij}^k \geq \sum_j b_{zj}^k, \forall k \in K, \forall z \in V^c, \forall i \in V^d. \quad (10)$$

Cost function  $J$  is defined in equation (1) and presents the total cost that should be minimised.  $b_{ij}^k$  is a binary variable describing vehicles' routes which is 1 when the  $k^{\text{th}}$  vehicle goes toward customer  $j$  after serving customer  $i$  and 0 otherwise. Constraint (2) hampers being more demand in a vehicle than its capacity. Constraints (3) and (4) prohibit servicing each customer more than once, which means that each customer should be met just once. Constraint (5) restrains loop shaping in each node. The number of incoming and outgoing vehicles should be the same for each depot. Also, at least one vehicle should be exited from each depot and these are represented by equations (6) and (7). Constraints (8) and (9) shape the proper loop entail depot. Vehicles must start from depots and this is guaranteed by equation (10).

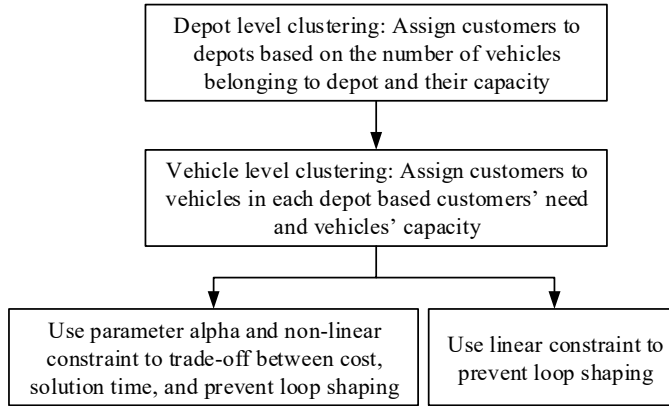
Solution to the above optimisation problem does not guarantee an appropriate result because it may cause in an improper loop, meaning that there could exist some loops without depots. Considering  $s_1, s_2, \dots, s_l$  as customers should be served by a vehicle, a loop such as  $s_1 \rightarrow s_2 \rightarrow \dots \rightarrow s_l \rightarrow s_1$  which does not include a depot could not be acceptable. To consider this issue, a new amendment is proposed in the next section.

Also, due to the NP-hardness of the MDVRP, solving the problem considering equations (1)–(10) could become impossible by increasing the number of customers, vehicles, and depots. In other word, modelling the problem in a way that the model prohibits the sub-tour with linear equation (due to the lack of solver to solve nonlinear integer programming) is impossible; therefore, it is obligatory to explain substitute methods prohibiting loop shaping (creating sub-tours). In the next section, we propose a new approach to handle the issue by decomposing the problem into sub-problems with less computational complexity.

### 3 Methodology

Due to the existing complexity, solving an MDVRP directly exploiting the exact methods is time consuming if not impossible. Instead, we propose a new approach by breaking the problem into sub-problems with reduced dimension. At first, customers are assigned to each depot based on the number of vehicles in a depot, capacity of vehicles, and demands of customers. Then, if the complexity still exists, reduction is further performed at the vehicle level, where customers of each depot are allocated to vehicles. Two approaches could be used to assign customers to vehicles. Once breaking the problem into some sub-problems and reducing their dimensions, a mixed-integer linear programming (MILP) could be employed to solve each sub-problem. The final solution would be union of solutions of sub-problems. Figure 1 illustrates the procedure.

**Figure 1** Main steps of solving a classic MDVRP



#### 3.1 Depot level clustering

Assume  $(x_j^d, y_j^d)$  and  $(x_i^c, y_i^c)$  represent coordinates of the  $j^{\text{th}}$  depot and the  $i^{\text{th}}$  customer, respectively, where  $j = 1, 2, \dots, n_d$  and  $i = 1, 2, \dots, n_c$ . Also, consider  $c_{ij}$  as a cost between the  $i^{\text{th}}$  customer and the  $j^{\text{th}}$  depot, which can be the Euclidean distance, deriving time, fuel consumption, or any other cost computed from the existing data. For more convenience, here  $c_{ij}$  respects the Euclidean distance between the  $i^{\text{th}}$  customer and the  $j^{\text{th}}$  depot. Thus,  $c_{ij}$  is computed as follows:

$$c_{ij} = \sqrt{(x_i^c - x_j^d)^2 + (y_i^c - y_j^d)^2}, i = 1, \dots, n_c, j = 1, \dots, n_d.$$

The following problem is solved to allocate customers to depots.

$$\min_{z_{ij}} \sum_{j=1}^{n_d} \sum_{i=1}^{n_c} c_{ij} z_{ij}, \quad (11)$$

Subject to

$$\sum_{j=1}^{n_d} z_{ij} = 1, i = 1, 2, \dots, n_c, \quad (12)$$

$$\sum_{i=1}^{n_c} q_i z_{ij} \leq KQ, j = 1, 2, \dots, n_d. \quad (13)$$

**Figure 2** Structure of P01

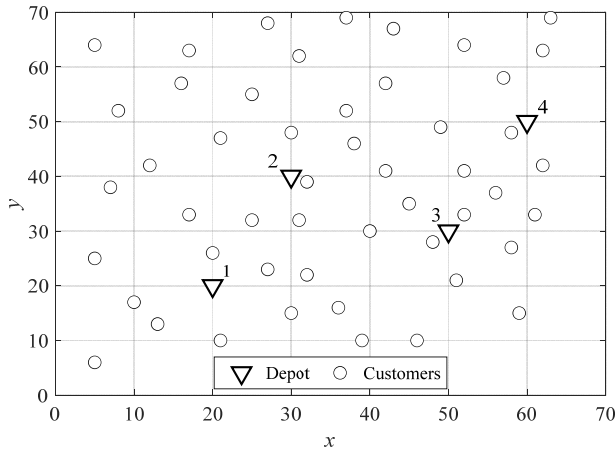
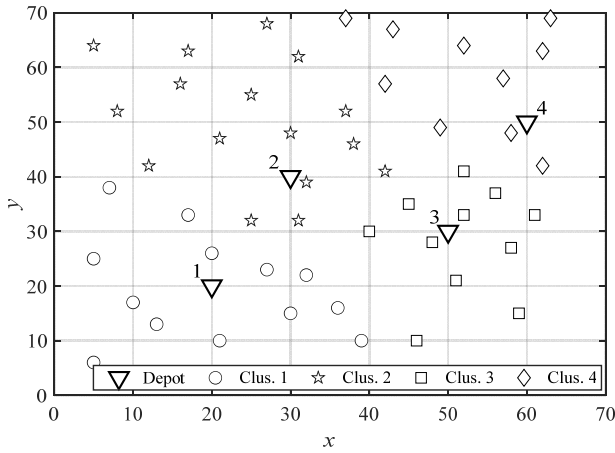


**Figure 3** Allocating customers to depots in P01



Each depot possesses maximum  $K$  vehicles with capacity  $Q$ .  $q_i$  represents the  $i^{\text{th}}$  customer's need and  $z_{ij}$  is a binary variable determines whether the  $i^{\text{th}}$  customer belongs to the  $j^{\text{th}}$  depot or not. This problem aims to assign customers to depots based on the defined cost in equation (11). Each customer should allocate to one depot, which is guaranteed by equations (12) and (13) satisfies the capacity constraint.

Figure 2 illustrates problem P01 from Cordeau et al. (1997) dataset, which has 50 customers and 4 depots that own maximum 4 vehicles with capacity 80. The proposed approach was implemented and broke this problem into four sub-problems (Figure 3).

### 3.2 Vehicle level clustering

After assigning customers to depots, if the computational complexity related to sub-problems is still high, customers will be allocated to vehicles in each depot. This part is more challenging in lieu of the former one because the clusters' centroids are unknown. Capacitated clustering (CC) is used to assign customers to vehicles. To do so, two methods could be employed. According to the first method, each vehicle in a depot seizes some customers breaking the problem into several TSPs. Then all newly defined problems are solved using MILP. In the second approach, customers are assigned to some vehicles, and the problem is turned to some CVRPs.

Referring to previous point, customers in each depot are clustered into relevant groups based on the number of vehicles being in a depot. Suppose  $r_1^j, r_2^j, \dots, r_w^j$  and  $q_1^j, q_2^j, \dots, q_w^j$  represent the customer assigned to the  $j^{\text{th}}$  depot and their requests, respectively. By using CC, customers in the  $j^{\text{th}}$  depot will be allocated to  $N^j$  clusters representing the adequate number of vehicles for serving customers. We exploit an improved  $k$ -means algorithm (Geetha et al., 2009) to execute CC. Following steps demonstrate an improved  $k$ -means algorithm.

- Compute adequate clusters ( $N^j$ ): Considering vehicle capacity and customers belonging to each depot,  $N^j$  is computed as follows:

$$N^j = \sum_{i=1}^w \frac{q_i^j}{Q}, i = 1, 2, \dots, w \quad (14)$$

- Select initial centroids: The  $N^j$  initial centroids are selected based on the customers' demands belonging to the  $j^{\text{th}}$  depot. By sorting customers decreasingly based on their needs (for example, suppose  $r_w^j \geq \dots \geq r_2^j \geq r_1^j$  are sorted based on customers' needs where  $r_w$  has the most demand and  $r_1$  has the least one).  $N^j$  customers with the most demands are adopted as initial centroids. For no loss of generality, in the case when customers have similar demands, the one with the lower index will be selected as initial centroid avoiding duplication and helping one has a similar result in each time when the algorithm is run.
- Compute priority: After determining centroids, the cost between the  $i^{\text{th}}$  customer and  $N^j$  centroid is computed. This cost is represented by  $\text{cost}_{iu}$ , where  $i$  and  $u$  indicate the  $i^{\text{th}}$  customer and  $u^{\text{th}}$  cluster, respectively. The  $i^{\text{th}}$  customer's priority is defined as  $P_i$  explaining the customer's tendency to assign to the  $u^{\text{th}}$  cluster.

$$P_i = \frac{\text{cost}_{iu}}{q_i^j}, i = 1, \dots, w, u = 1, \dots, N^j$$

- Allocate customers to clusters: The  $i^{\text{th}}$  customer is assigned to cluster  $u$  based on its priority, meaning that the  $i^{\text{th}}$  customer selects a cluster with the highest priority. When the following condition is violated

$$\sum_{i=1}^{N_u} q_i^j \leq Q, \quad (15)$$

The customer will be assigned to the nearest cluster in terms of the priority.  $N_u$  is the number of customers in the  $u^{\text{th}}$  cluster and  $Q$  points to the vehicle's capacity.

- Update centroids and check the criteria: When all customers of the  $j^{\text{th}}$  depot are assigned to  $N^j$  clusters, the new centroids' coordinates is computed as follows:

$$x_u = \sum_{i=1}^{N_u} \frac{x_i^u}{N_u}, y_u = \sum_{i=1}^{N_u} \frac{y_i^u}{N_u}, u = 1, 2, \dots, N^j.$$

The third to fifth steps run iteratively till  $x_u$  and  $y_u$  stay unchanged.

Algorithm 1 illustrates the steps needed for allocating customers to vehicles. Figure 4 displays results of performing Algorithm 1 for the first depot branched off from P01.

**Algorithm 1** Improved  $k$ -means algorithm for the  $j^{\text{th}}$  depot

---

**Input:** Coordinate  $(x_i^j, y_i^j)$ , demands  $q_i^j$ , customer  $r_i^j$

**Output:**  $N^j$  clusters

**Procedure:**

Calculate  $N^j$  based on customers' demands and vehicle capacity

Select first  $N^j$  centroids based on arranging customers' demands

**while**  $flag = 0$

**for** each customer  $r_i^j$

**while**  $r_i^j$  is not assigned

      Calculate cost between each  $N^j$  centroids and sort it decreasingly

      Assign nearest centroid of  $r_i^j$  as  $s$

      Group all unassigned customers as  $G$  with  $s$  as their nearest centroid

      Calculate the priority value for  $r_i^j \in G$

      Assign  $r_i^j \in G$  to their nearest centroid based on priority and capacity constraint

**if**  $r_i^j$  is not assigned

      choose the next nearest centroid

**end if**

**end while**

**end for**

  Updating new centroids

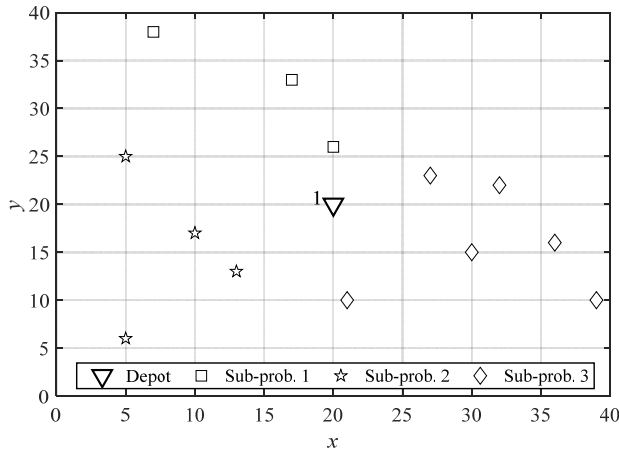
**if** centroids are not changed

```

    flag = 1
  end if
end while

```

---

**Figure 4** Allocating customers to vehicles in the first depot of P01

### 3.3 First method of prohibiting improper loop shaping

After assigning customers to vehicles, MDVRP can be assumed as several TSPs. Thus, by solving sub-problems, the original MDVRP is solved. As mentioned in Section 2, preventing improper loops in the solution of an MDVRP requires nonlinear constraints to be considered in the problem definition. However, breaking MDVRP to several TSPs can be helpful to apply linear constraints to stop shaping improper loops in each sub-problem. The following relations (Jiang et al., 2007) could be utilised to solve TSPs without shaping incorrect loops.

$$0 \leq y_{ij} \leq (N^j - 1)b_{ij}, \forall i, j \in V^j, i \neq j, \quad (16)$$

$$\sum_{j \neq i} y_{ij} = N^j - 1, \forall j \in V_D^j, \forall i \in V_C^j, \quad (17)$$

$$\sum_{i \neq j} y_{ij} - \sum_{z \neq j} y_{jz} = 1, \forall j \in V_C^j, \quad (18)$$

where  $N^j$  is the number of customers in the  $j^{\text{th}}$  sub-problem,  $V^j$  is the number of nodes in the  $j^{\text{th}}$  sub-problem which entail  $V_D^j$  and  $V_C^j$  representing depot and customer nodes respectively, and  $y_{ij}$  is a continuous variable which represents the commodity flow in  $ij$  edge. Constraints (16)–(18) are arc constraints and guarantee single loop in the solution to each sub-problem. To prove the claim let assume that the solution in one sub-problem contains more than one distinct loop. Now consider loop  $L_0 \rightarrow L_1 \rightarrow \dots \rightarrow L_b \rightarrow L_0$  which does not contain a depot. From equations (16) and (18) it follows that  $y_{L_j-1L_j} - y_{(L_jL_{j+1}) \bmod (b+1)} = 1$  for all  $j = 1, \dots, b$ , therefore  $y_{L_0L_1} = y_{L_bL_0} + b$ , which

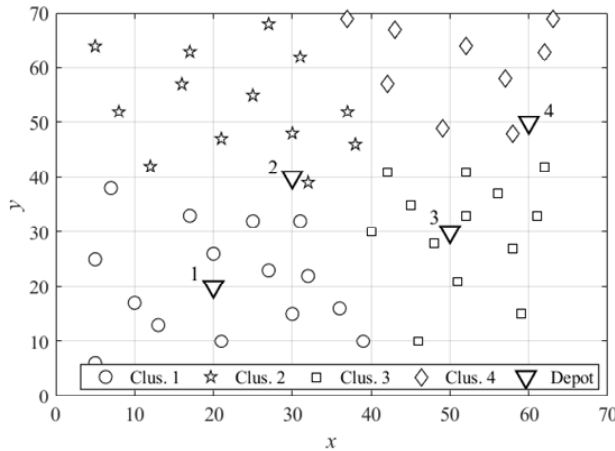
contradicts equation (18) where  $y_{L_b L_0} - y_{L_0 L_1} = 1$ . Thus, no loop can exist without depot. Since each depot involves exactly one loop, thus at most one loop is generated.

Using equations (3), (4), and (16)–(18), each sub-problem is solved, and the final solution is obtained.

### 3.4 Second method of prohibiting improper loop shaping

First, customers should be assigned to depots. Several approaches can be used to perform this clustering such as the approach described in Rahimi Baghbadorani et al. (2021) or the one presented in Section 3.1. Figures 3 and 5 show their results respectively. Then, like the previous subsection, customers are assigned to vehicles in each depot with the difference that the sub-problems will not be solved for a single-vehicle. The number of clusters in each depot is determined based on tuning parameter  $\alpha$  appointing the specific number of vehicles in each cluster to serve customers regarding the customers' demands and vehicle's capacity. To more enlighten, suppose  $r_1^j, r_2^j, \dots, r_w^j$  and  $q_1^j, q_2^j, \dots, q_w^j$  demonstrate customers and their demands in the  $j^{\text{th}}$  depot, respectively.  $N^j$  is computed according to equation (14). Unlike the previous subsection, where this parameter indicates the number of clusters in each depot, in this subsection,  $N^j$  represents the number of vehicles should be used by each depot. Thus, the number of clusters can be changed from 1 to  $N^j$ . This implies that more than one vehicle could serve customers in some clusters. For example, assume a depot exploits three vehicles to gratify customers' demands. Then, customers could be clustered in three groups where a single vehicle serves them in each group, or they could be flocked into two groups where one group is served by two vehicles and another one is served by one, or finally, they could be grouped in one bunch that are serviced by three vehicles. This clustering is crystallised by  $\alpha$ , which is tuned between 0 and 1 with interval  $(N^j - 1)^{-1}$ . For  $\alpha = 0$ , there will be only one cluster with  $N^j$  vehicles and for  $\alpha = 1$  there are  $N^j$  clusters with single vehicle for each of them.

**Figure 5** Allocating customers to depots in P01 using Rahimi Baghbadorani et al. (2021) method

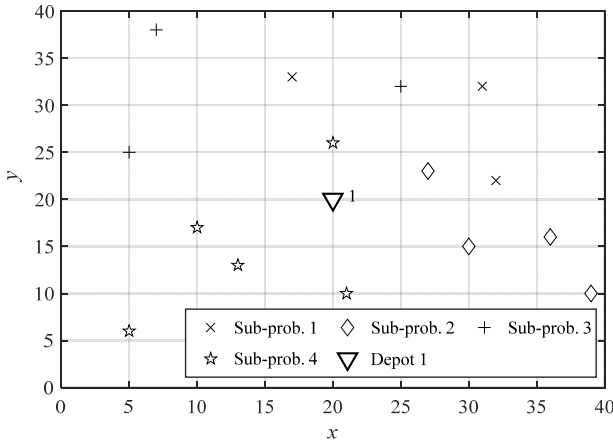


In the previous example, the depot possesses three vehicles then,  $\alpha$  could assume 0, 0.5, or 1, that implies one cluster with three vehicles, two clusters using one and two vehicles, and three clusters with one vehicle. The clustering method here is similar to the previous subsection, except that the capacity condition would be as

$$\sum_{i=1}^{N_u} q_i^j \leq k_u(\alpha)Q, \quad (19)$$

where  $k_u(\alpha)$  determines the number of vehicle in cluster  $u$ . Figure 6 shows the clustering of customers in the first depot of P01 considering  $\alpha = 1$ .

**Figure 6** Allocating customers to vehicles in the first depot of P01 with  $\alpha = 1$



After identifying the clusters, equations (1)–(10) are used to solve the problem for each cluster based on the number of vehicles. But, solving equations (1)–(10) does not guarantee to have no improper loops. However, imposing the following condition could reduce the probability of shaping the improper loops and speed up the problem solution.

$$b_{ij}^k + \sum_j \sum_{h=1, h \neq k}^{k_u} b_{ij}^h \leq 1, \forall k \in k_u, \forall i, j \in V^c \quad (20)$$

This constraint has been used to prevent vehicles from interfering in each other's tasks. It means that the good cannot be shipped from customer  $i$  to customer  $i + 1$  by vehicle  $v_i$ , but shipped from customer  $i + 1$  to the next customer by vehicle  $v_j$ .

Also, Lemma 1 helps in case of an unacceptable loop in the way that the problem is solved iteratively with an additional constraint leading to remove similar improper loops in each iteration.

**Lemma 1:** Let vertices of  $G$  be labelled as  $v_1, v_2, \dots, v_n$  and adjacency matrix  $G$  be  $A$  where row/column  $i$  of  $A$  corresponds to  $v_i$ . Let  $L = D - A$ , where  $D$  is a diagonal matrix with  $(D)_{ii} = \deg(v_i)$ . This matrix is called Laplacian matrix of  $G$ .  $G$  is acyclic if and only if  $\text{trace}(L) = 2\text{rank}(L)$ .

*Proof:*  $G$  is acyclic if and only if it is a forest, i.e.,  $G$  has  $c$  components and exactly  $n - c = \text{rank}(L)$  edges ( $|E(G)|$ ). On the other hand, according to the handshaking lemma,  $|E(G)| = 0.5\text{trace}(L)$ . Thus, the lemma is proved.

Using equations (1)–(10), (20) and Lemma 1, leads to a decent solution.

### 3.5 Solving MILP

By breaking the original problem into small ones, a set of MILPs is defined. The following steps are used to solve MILPs.

- **LP pre-processing:** This part helps us to reduce the problem size. In other words, this part prohibits repetitive variables from causing sparsity in the constraint matrix and eases finding a feasible solution.
- **LP running:** Problem is solved without considering binary or integer variables in this section. This gives us a relaxed solution.
- **Integer programming pre-processing and cut generation:** The main goal in this part is to eliminate some improper solution acquired in the previous step, fix some integer solution, tighten LP relaxation, and add some constraints to make the solution close to the integer ones (Atamtürk et al., 2000; Cornuéjols, 2008).
- **Branch and Bound:** A systematic approach is used in this part to find the final solution. The branch-and-bound method constructs a sequence of sub-problems that attempt to converge to a solution of the MILP (Theurich et al., 2021). The sub-problems give a sequence of upper and lower bounds on the main problem. The first upper bound is any feasible solution, and the first lower bound is the solution to the relaxed problem. The performance depends on the branching section (choosing which variable to split) and bounding section (choosing which node to explore). We use strong branching and pseudo-cost branching strategies (Achterberg et al., 2005) and the best projection for bounding (Morrison et al., 2016).

**Figure 7** Vehicles' routes in P01, (a) vehicle routes in P01 using Section 3.3 method (b) vehicle routes in P01 using Section 3.4 method

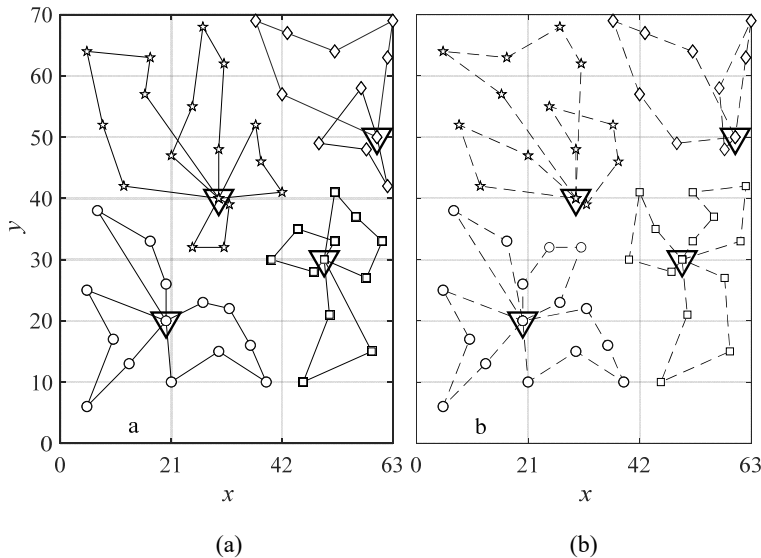


Figure 7 and Table 1 show the vehicles’ routes by solving MILP in P01 considering approaches introduced in Sections 3.3 and 3.4 ( $\alpha$  equals 1 in all depots). As it is seen, in both methods each vehicle has a proper loop beginning from depots and ending up in the same depot. The final price of each method is reported in Table 2.

**Table 1** Vehicles’ routes in P01

|         |           | Section 3.3 method |    |    |    |    |    |    |    | Section 3.4 method |    |    |    |    |    |    |    |    |
|---------|-----------|--------------------|----|----|----|----|----|----|----|--------------------|----|----|----|----|----|----|----|----|
| Depot 1 | Vehicle 1 | 51                 | 4  | 18 | 25 | 51 |    |    |    | 51                 | 19 | 40 | 41 | 13 | 51 |    |    |    |
|         | Vehicle 2 | 51                 | 13 | 41 | 40 | 19 | 51 |    |    |                    | 51 | 18 | 25 | 51 |    |    |    |    |
|         | Vehicle 3 | 51                 | 17 | 37 | 15 | 45 | 44 | 42 | 51 | 51                 | 42 | 44 | 45 | 15 | 37 | 51 |    |    |
|         | Vehicle 4 |                    |    |    |    |    |    |    |    | 51                 | 17 | 12 | 47 | 4  | 51 |    |    |    |
| Depot 2 | Vehicle 1 | 52                 | 46 | 12 | 47 | 52 |    |    |    | 52                 | 6  | 24 | 14 | 52 |    |    |    |    |
|         | Vehicle 2 | 52                 | 6  | 48 | 26 | 8  | 27 | 52 |    |                    |    | 52 | 8  | 26 | 7  | 43 | 23 | 52 |
|         | Vehicle 3 | 52                 | 1  | 32 | 11 | 52 |    |    |    | 52                 | 27 | 48 | 1  | 32 | 46 | 52 |    |    |
|         | Vehicle 4 | 52                 | 14 | 24 | 43 | 7  | 23 | 52 |    |                    |    |    |    |    |    |    |    |    |
| Depot 3 | Vehicle 1 | 53                 | 16 | 50 | 34 | 30 | 53 |    |    |                    | 53 | 9  | 50 | 16 | 21 | 34 | 53 |    |
|         | Vehicle 2 | 53                 | 9  | 38 | 5  | 49 | 53 |    |    |                    | 53 | 10 | 33 | 39 | 30 | 53 |    |    |
|         | Vehicle 3 | 53                 | 10 | 33 | 39 | 53 |    |    |    | 53                 | 49 | 5  | 11 | 38 | 53 |    |    |    |
| Depot 4 | Vehicle 1 | 54                 | 21 | 29 | 2  | 20 | 54 |    |    |                    | 54 | 29 | 20 | 36 | 35 | 54 |    |    |
|         | Vehicle 2 | 54                 | 22 | 31 | 28 | 3  | 36 | 35 | 54 | 54                 | 3  | 28 | 31 | 22 | 2  | 54 |    |    |

4 Dynamic MDVRP

Considering uncertainties such as traffic congestion, car accidents and breakdowns, and other unpredicted circumstances that could happen during the operation, using a classic VRP solution can irritate the final cost and gives an irrational solution due to perpetual variation in the cost between nodes (Kucharska, 2019). Therefore, regarding this issue, the dynamic MDVRP is studied, and a fair solution based on receding horizon is presented. Noteworthy, in dynamic model, it is assumed that a deterministic model of traffic varying by time is known. Thus, the innovation of the dynamic approach is using the receding horizon and optimising the vehicles’ routes based on the deterministic model of the traffic, control horizon, and prediction horizon.

Implementing a dynamic MDVRP initiates from the classical counterpart solution, i.e., based on the existing traffic data, the classical MDVRP is solved and applied to vehicles as an initial solution. Then, after an adequate time, situation is examined for any updates. If there are no notable changes, vehicles continue to serve the customers according to the existing solution; otherwise, the problem is solved again and new solution conveyed to the vehicles.

To make the problem more practical, a handful of routes with different distances are considered between the nodes where the number is determined by the designer. For more enlighten, Figure 8 depicts  $E$  routes between the  $i^{th}$  and  $j^{th}$  customers. Also, traffic profiles ( $T$ ) between the nodes are assumed to have normal distribution (Sabar et al., 2019) with means and variances ascertained by the designer (also, they can be selected based on

previous data). Figure 9 illustrates a sample  $\mathcal{T}$  between the  $i^{\text{th}}$  and  $j^{\text{th}}$  nodes along the  $m^{\text{th}}$  route.

**Figure 8** Available routes between the  $i^{\text{th}}$  and the  $j^{\text{th}}$  customers

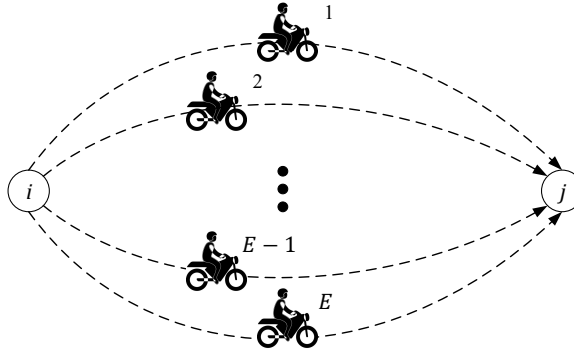
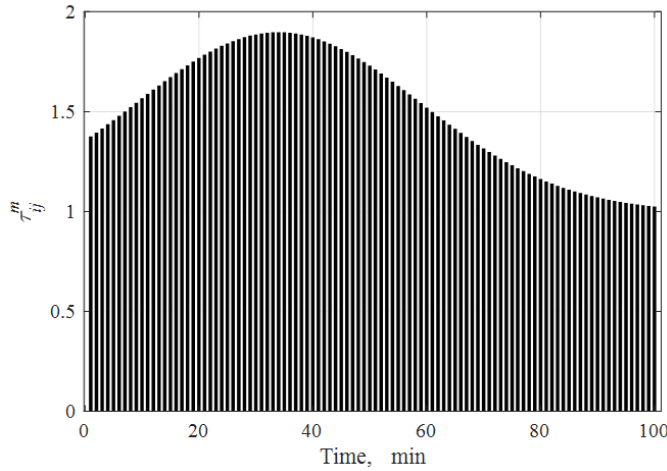


**Figure 9**  $\mathcal{T}_{ij}^m$  between nodes  $i$  and  $j$  in the  $m^{\text{th}}$  path



Having traffic profile and Euclidean distance between all nodes along different routes, the cost amidst every two nodes is calculated as follows:

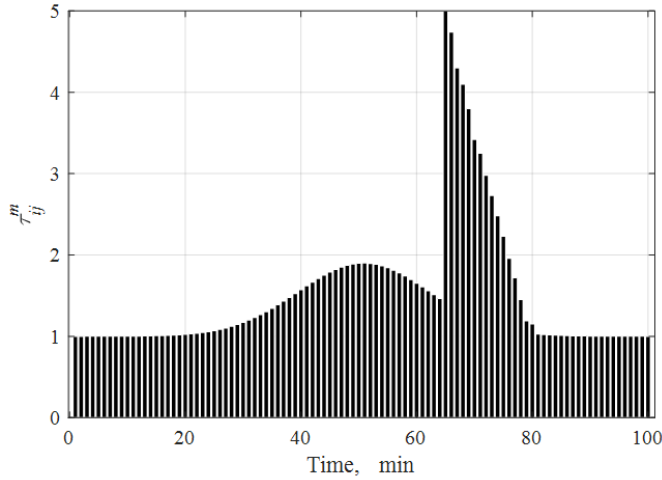
$$f_{ij}^m(t) = d_{ij}^m \mathcal{T}_{ij}^m(t) \text{ and } \mathcal{T}_{ij}^m(t) = a_{ij}^m + \frac{1}{\sigma_{ij}^m \sqrt{2\pi}} e^{-\frac{(t - \mu_{ij}^m)^2}{2\sigma_{ij}^{m2}}},$$

where  $f_{ij}^m$ ,  $d_{ij}^m$ , and  $\mathcal{T}_{ij}^m$  depict the cost, Euclidean distance, and traffic profile between the  $i^{\text{th}}$  and  $j^{\text{th}}$  nodes along the  $m^{\text{th}}$  route respectively.  $\mathcal{T}_{ij}^m$  entails  $a_{ij}^m$  and  $(\mu_{ij}^m, \sigma_{ij}^m)$  that represent the traffic profile adjustment and normal distribution variables. Thus,  $\mathcal{T}_{ij}^m$  could be changed when car crash occurs to face real situation. Figure 10 shows  $\mathcal{T}_{ij}^m$

changes by a car crash at  $t = 65$  and is distorted for a while and then came back to its normal form. The more the accident intensity is, the more distorted  $\mathcal{T}_{ij}^m$  will be.

Complexity of the problem decreases using CC in two stages along with assigning the customers to vehicles. Then MILP are solved to determine the optimal routes. By specifying the routes, vehicles begin servicing the customers. To implement a dynamic MDVRP, status are monitored each  $T$  sample and the problem is solved again as a notable changes occurs. For solving the problem again, new CCs are performed with vehicles coordinate as new centroids and it continues until vehicles recognise their customers.

**Figure 10**  $\mathcal{T}_{ij}^m$  changes by a car crash



Assume  $m_{ij}$ ,  $n$ , and  $k$  are the minimum cost between the  $i^{\text{th}}$  customer and the  $j^{\text{th}}$  vehicle, customers who are not serviced yet, and the total vehicles serving customers. Solution to the following problem determines customers of each vehicle.

$$\min_{z_{ij}} \sum_{j=1}^k \sum_{i=1}^n m_{ij} z_{ij} \quad (21)$$

Subject to

$$\sum_{j=1}^k z_{ij} = 1, i = 1, 2, \dots, n, \quad (22)$$

$$\sum_{i=1}^n q_i z_{ij} \leq Q, j = 1, 2, \dots, k, \quad (23)$$

where  $z_{ij}$  is a binary variable unrolls whether the  $i^{\text{th}}$  customers belong to the  $j^{\text{th}}$  vehicle. After identifying customers of each vehicle, it should be assessed how many customers in each cluster should be served. Control horizon ( $h_c$ ) is used to determine which customers should be served by a vehicle.  $h_c$  is chosen by the user and helps to decline computational complexity. Solution to the following problem elects customers which should be served by each vehicle.

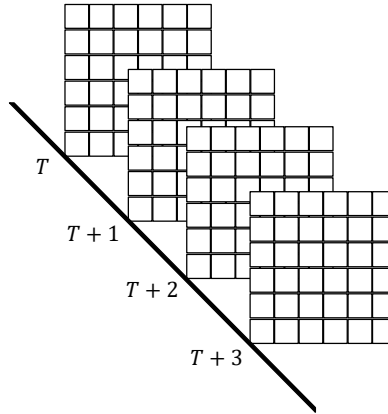
$$\min J = \sum_i b_i^j f_i^j, i = 1, 2, \dots, n_j, \quad (24)$$

$$h_c v^j T - tol \leq \sum_i b_i^j f_i^j \leq h_c v^j(t) T + tol, \quad (25)$$

where  $n_1, n_2, \dots, n_k$  depicts the number of customers in each cluster,  $f_i^j$  is a cost between the  $i^{\text{th}}$  customers belong to the  $j^{\text{th}}$  vehicle,  $b_i^j$  is a binary variable enlightens the  $i^{\text{th}}$  customers belong to the  $j^{\text{th}}$  vehicle should be served or not,  $v^j(t)$  is a mean speed of the  $j^{\text{th}}$  vehicle at moment  $t$ , and  $tol$  is a constant to relax constraint (25). After specifying customers who should be served, using equations (1)–(6), the minimum cost between vehicles and customers will be determined and vehicles serve the customers based on the new solution. This process continues either changes happen in the system or a vehicle serves its last customer.

The crucial point related to solving the problem is assessing the situation and deciding whether solving should be repeated or the existing solution would be adequate. The cost function shows the cost between all nodes considering traffic congestion momentarily; thus, a cost tensor can be assumed showing the cost matrix at different moments (Figure 11). Decision could be taken according to the tensor matrices by calculating the norm difference between two successive cost matrices at  $T + j - 1$  and  $T + j$  for  $j = 1, \dots, h_{p1}$  and calling their mean values as mean1. The calculations are iterated for another horizon  $h_{p2}$  ( $h_{p2} > h_{p1}$ ) and mean2 is computed. The condition for updating the routes is emerged as  $\text{mean1} > \text{mean2}$ ; otherwise, the vehicles will continue to their previous paths.

**Figure 11** Cost matrix in each sample time



*Remark 1:* Results presented in this work is based on mean values. However, the standard deviations of norms could be used as well. Also, we have used norm 2 of tensor matrices, which can be replaced by other norms such as norm 1 or Frobenius norm.

## 5 Simulation result

In this section, approaches described in Sections 3.3, 3.4, and 4 are evaluated. We ran our experiments on a PC with a Core i7-7700HQ, 2.80 GHz, 4GB RAM, and MATLAB 2019. Section 5.1 presents test results using methods described in Sections 3.3 and 3.4 and Section 5.2 provides results of dynamic MDVRP method introduced in Section 4.

### 5.1 Test results using methods in Sections 3.3 and 3.4

The performance of the methods explained in Sections 3.3 and 3.4 for solving a classical MDVRP is assessed in this part. In this regard, we use Cordeau et al. (1997) dataset to evaluate our methods, including a broad spectrum of problem variables. In this paper, we use 11 instances of Cordeau dataset entailing medium to large size scale ( $n = 50$  to 360 customers,  $d = 2$  to 9 depots, and  $K = 2$  to 6 vehicles). The efficacy of the proposed methods is compared with GVNS (Bezerra et al., 2018) and Sadati et al. (2020) methods where a meta-heuristic approach is used to solve MDVRP. Note that the optimum cost in GVNS and Sadati et al. (2020) are achieved after several runs; however, the final cost acquired by methods in Sections 3.3 and 3.4 is obtained after only one-time execution. For instance, the optimal cost in GVNS is accomplished after 30 runs.

The GVNS algorithm was coded in C++ and tested on a computer with Intel Core i3-2370M, 2.40 GHz, and 4GB RAM. Sadati's method has been implemented on Dell Precision T7810 model PC equipped with Intel Xeon E5-2690 v4 2.60 GHz processor and 32 GB RAM running 64-bit Windows 7 Professional, and it was coded in C# of Microsoft Visual Studio 2017 and employed IBM CPLEX version 12.7.1 as the commercial MILP solver called inside C#. A comparison between the systems used to solve the proposed approaches has been made in <http://www.cpubenchmark.net>, which shows the CPU speed of the system used in this article is approximately equal to the CPU speed of the system used in Bezerra et al. (2018) and is lower than the CPU speed of Sadati et al. (2020). Also, a comparison between the program's execution time by C++ and MATLAB shows that the speed of C++ is about ten times faster than MATLAB (Aruoba and Fernández Villaverde, 2015). Since C++ and C# have the same background, C#'s speed will be higher than MATLAB as well.

Table 2 presents solutions found by the proposed algorithms with their computational time and best-known solution (BKS), which has been found so far. Although, the proposed methods in Sections 3.3 and 3.4 ( $\alpha = 1$ ) result in more cost than GVNS and Sadati et al. (2020); they reduce computation time dramatically. Thus, these approaches could be implemented in real-time affairs such as dynamic MDVRP. Also, in Ramos et al. (2020), a comparison has been made between two-commodity flow formulation and three other methods, showing that these methods need much time to solve Courdeau's instances compared to approaches presented in Sections 3.3 and 3.4; even in some cases, they were not able to solve the problem. Also, Stodola (2018) used a meta-heuristic method in to solve MDVRP. This method needs longer solution time and provides a worse gap than GVNS and Sadati et al. (2020) methods indicated in Table 2. While the 'GVNS' method was able to solve the first ten examples, it was unable to solve other examples starting from example P23. Example P23 was selected in this study to show that the proposed approaches could solve other examples as well and in reasonable time.

**Table 2** Comparing performance of the proposed algorithms with GVNS and Sadati et al. (2020)

| <i>Inst</i> | <i>n</i> | <i>m</i> | <i>d</i> | <i>BKS</i> | <i>GVNS</i><br>average | <i>T (sec)</i><br>average | <i>Sadati</i><br>average | <i>T (sec)</i><br>average | <i>Section 3.3</i><br>method | <i>T (sec)</i> | <i>Section 3.4</i><br>method | <i>T (sec)</i> |
|-------------|----------|----------|----------|------------|------------------------|---------------------------|--------------------------|---------------------------|------------------------------|----------------|------------------------------|----------------|
| p01         | 50       | 4        | 4        | 576.84     | 591.09                 | 36                        | 591.09                   | 9.35                      | 614                          | 1.00           | 634.55                       | 17.7           |
| p02         | 50       | 2        | 4        | 473.53     | 476.66                 | 33.6                      | 476.66                   | 20.87                     | 511                          | 1.67           | 515.47                       | 17.37          |
| p03         | 75       | 3        | 5        | 641.19     | 641.49                 | 134.6                     | 641.49                   | 141.86                    | 693                          | 2.06           | 697.36                       | 21.75          |
| p04         | 100      | 8        | 2        | 1,001.4    | 1,025.44               | 702.6                     | 1,008.47                 | 258.75                    | 1,069                        | 5.76           | 1,072.2                      | 23.4           |
| p05         | 100      | 5        | 2        | 750.03     | 757.46                 | 564.6                     | 758.87                   | 159.10                    | 835                          | 7.75           | 866                          | 35.4           |
| p06         | 100      | 6        | 3        | 876.50     | 889.79                 | 502.8                     | 881.76                   | 194.81                    | 928                          | 6.40           | 981.76                       | 23.5           |
| p07         | 100      | 4        | 4        | 881.97     | 898.31                 | 405                       | 896.96                   | 94.75                     | 965                          | 6.18           | 977.5                        | 28             |
| p12         | 80       | 5        | 2        | 1,318.95   | 1,326.85               | 166.2                     | 1,318.95                 | 13.75                     | 1,366                        | 3.38           | 1,429.2                      | 17             |
| p15         | 160      | 5        | 4        | 2,505.42   | 2,567.62               | 1,421.4                   | 2,522.79                 | 255.44                    | 2,731                        | 8.15           | 2,841                        | 30.06          |
| p18         | 240      | 5        | 6        | 3,702.85   | 3,866.25               | 1,800.6                   | 3,802.9                  | 302.91                    | 4,155                        | 32.7           | 4,288                        | 56.55          |
| p23         | 360      | 4        | 9        | 6,095      | 6,130                  | -----                     | 6,145.58                 | 609.22                    | 6,273                        | 37.77          | 6,544                        | 79             |

As it is mentioned in Section 3.4,  $\alpha$  is used to tune the complexity of sub-problems. Lower  $\alpha$  means that customers of a depot are divided into fewer groups. Hence, the total cost would be lesser than the case that the customers are divided into more groups. Table 3 indicates the execution time and the final cost, which could be a guide for the user to select  $\alpha$  according to the problem in hand.

**Table 3** Efficacy of changing  $\alpha$  in P01

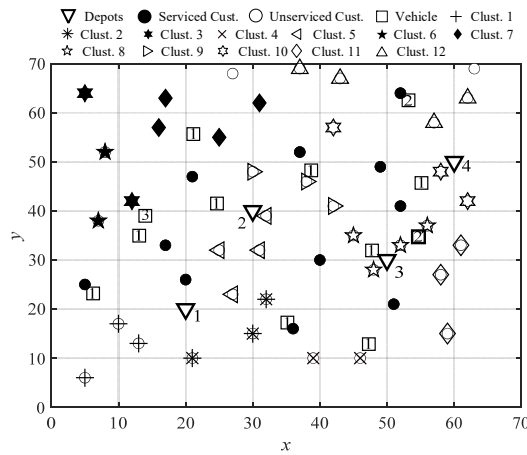
| $\alpha$ for depot 1 | $\alpha$ for depot 2 | $\alpha$ for depot 3 | $\alpha$ for depot 4 | Cost   | $T$ (sec) |
|----------------------|----------------------|----------------------|----------------------|--------|-----------|
| 1                    | 1                    | 1                    | 1                    | 634.55 | 17.7      |
| 1                    | 1                    | 1                    | 0                    | 623    | 59.57     |
| 0.66                 | 0.5                  | 0.5                  | 1                    | 632.83 | 35.65     |
| 0.33                 | 0.5                  | 0.5                  | 1                    | 631.03 | 40.87     |
| 0.33                 | 0.5                  | 0.5                  | 0                    | 620.56 | 66.5      |
| 0.33                 | 0.5                  | 0                    | 0                    | 603.74 | 146.91    |

## 5.2 Dynamic MDVRP test results

The performance of the proposed method in paralysing dynamic MDVRP is indicated by solving an example. We use P01 to perform our approach. Figure 2 illustrates the structure of P01. Regarding Section 3.3, problem P01 is decomposed into small sub-problems in two-stage, and by solving each sub-problem, servicing is commenced based on the results [Figure 7(a)]. Traffic situations are checked every  $T = 10$  sample times and whenever  $\text{mean1} (h_{p1} = 5) \geq \text{mean2} (h_{p2} = 10)$ , the problem is solved again with the new conditions. To solve the problem, we consider  $h_c = 2$ . The service's time is considered 100 sample times, and the range for parameters  $\sigma$ ,  $\mu$ , and  $m$  are,  $[80, 120]$ ,  $[80, 120]$ , and  $[1, 3]$ , respectively, and  $\alpha$  is considered 1. The receding horizon parameters are selected as follows:

- $\frac{T_{\text{service time}}}{20} \leq T \leq \frac{T_{\text{service time}}}{10}$ , so we select  $T = 10$ .
- $h_p T \geq T_{\text{service time}}$ , then  $h_p$  is considered 10.
- $0.1h_p \leq h_c \leq 0.2h_p$ , accordingly  $h_c$  is selected 2.

Figure 12 shows the situation where the solution has been updated during the service time and vehicles serve some specific customers which can be different from the previous ones. It is seen that a vehicle is able to serve customers who were not in its primary cluster [compare Figure 12 and Figure 7(a)] showing the cooperation among the vehicles. The numbers in squares, illustrating vehicles, represent the route's number between two customers. Also, to emulate actual conditions, we add a normal random noise with amplitude 0.03 to the cost matrix in each sample time. Moreover, we considered some car accidents and breakdowns between some nodes at different time intervals and different routes randomly. A car crash effect on the traffic profile can be seen in Figure 10.

**Figure 12** Updating routed during servicing in P01**Table 4** Final cost comparison among three scenarios

| <i>Final cost in the first scenario<br/>(using Euclidean distance to<br/>determine vehicle's routes)</i> | <i>Final cost in the second<br/>scenario (using Euclidean<br/>distance + existing data to<br/>determine vehicle's routes)</i> | <i>Final cost in the third<br/>scenario (using receding<br/>horizon approach to<br/>determine vehicle's routes)</i> |
|----------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| 5,184.3                                                                                                  | 4,657                                                                                                                         | 3,950                                                                                                               |

**Table 5** Comparing performance of the proposed algorithms with the first and second scenario in dynamic MDVRP

| <i>Inst</i> | <i>Final cost in the<br/>first scenario (<math>J_1</math>)</i> | <i>Final cost in the<br/>second scenario (<math>J_2</math>)</i> | <i>Final cost in the third<br/>scenario (<math>J_3</math>)</i> | <i>gap (<math>J_3/J_i</math> %)</i> |           |
|-------------|----------------------------------------------------------------|-----------------------------------------------------------------|----------------------------------------------------------------|-------------------------------------|-----------|
|             |                                                                |                                                                 |                                                                | $J_3/J_1$                           | $J_3/J_2$ |
| p02         | 4,478.1                                                        | 4,103.3                                                         | 3,265.7                                                        | 27.1                                | 20.4      |
| p03         | 5,861.6                                                        | 5,201.6                                                         | 4,431.9                                                        | 24.4                                | 14.8      |
| p04         | 8,115.2                                                        | 7,320.7                                                         | 5,910.1                                                        | 27.2                                | 19.3      |
| p05         | 7,115.2                                                        | 6,308.8                                                         | 5,239.9                                                        | 26.4                                | 16.9      |
| p06         | 7,761.4                                                        | 6,981.3                                                         | 5,701.6                                                        | 26.5                                | 18.3      |
| p07         | 8,002.2                                                        | 7,150.3                                                         | 5,798.2                                                        | 27.5                                | 18.9      |
| p12         | 9,138.9                                                        | 8,302.6                                                         | 6,380.7                                                        | 30.2                                | 23.1      |
| p15         | 20,154.3                                                       | 17,106.4                                                        | 11,390.8                                                       | 43.5                                | 33.4      |
| p18         | 34,782.6                                                       | 26,127.2                                                        | 16,781.5                                                       | 51.8                                | 35.8      |
| p23         | 52,691.2                                                       | 38,457.1                                                        | 24,810.4                                                       | 53.0                                | 35.5      |

To evaluate the proposed approach's performance, we compare the final cost in three scenarios (Kim et al., 2016). In the first scenario, the solution is achieved considering just a static situation (the cost between nodes is computed only based on the Euclidean distance); however, it is applied to the problem being in a dynamic situation. The second scenario is similar to the first one with the difference that in the second scenario, the cost between nodes is considered similar to previous days' (or months') cost achieved in a dynamic manner (in the first and second scenario vehicles start servicing based on the primary cost matrix, and cost matrix will not be updated during the service time), and in

the third scenario, the proposed approach is applied in a dynamic mode. The results of three scenarios for problem P01 are shown in Table 4. The problem has been solved five times for each scenario (due to randomness nature), and the average has been reported in Table 4. It is incontrovertibly axiomatic that the proposed approach reduces the final cost by about 23.8% compared to the classic scenario. These scenarios also are applied to several examples of the Cordeau dataset, and each of them is solved five times. The result is reported in Table 5 having gap represents improvement percentage of the proposed method compared to the first and second scenarios demonstrating the similar results in all of them.

## 6 Conclusions

Although dynamic MDVRP could represent wide variety of scenarios in real world applications, a few researchers have focused on studying this variant of problems in the recent years. In this paper we have addressed a dynamic MDVRP involved with traffic congestion. The proposed algorithm breaks the problem into small sub-problems and applies a receding horizon approach dropping computational complexity impressively. Besides dynamic MDVRP, classic MDVRP was tackled by two rational approaches decreasing execution time dramatically. To illustrate the efficacy of proposed approaches the comparison of static and dynamic problems are conducted. Simulation results show that the receding horizon approach is efficient in solving dynamic MDVRP. The future works of interest contain analyse a dynamic MDVRP considering the possibility of customers' mobility during time service, applying other methods for extracting traffic model (such as Markov chain, machine learning method, and regression), and computing the complexity order of proposed approach and its feasibility.

## References

- Achterberg, T., Koch, T. and Martin, A. (2005) 'Branching rules revisited', *Operations Research Letters*, Vol. 33, No. 1, pp.42–54.
- Aruoba, S.B. and Fernández Villaverde, J.F. (2015) 'A comparison of programming languages in economics', *Journal of Economic Dynamics and Control*, Vol. 58, pp.265–273.
- Atamtürk, A., Nemhauser, G.L. and Savelsbergh, M.W.P. (2000) 'Conflict graphs in solving integer programming problems', *European Journal of Operational Research*, Vol. 121, No. 1, pp.40–55.
- Azad, T. and Hasin, M.A.A. (2019) 'Capacitated vehicle routing problem using genetic algorithm: a case of cement distribution', *International Journal of Logistics Systems and Management*, Vol. 32, No. 1, pp.132–146.
- Bezerra, S.N., Souza, M.J.F., De Souza, S.R. and Coelho, V.N. (2018) 'A VNS based algorithm with adaptive local search for solving the multi-depot vehicle routing problem', *6th International Conference on Variable Neighborhood Search, ICVNS 2018*, Sithonia, Greece, 4–7 October.
- Contardo, C. and Martinelli, R. (2014) 'A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints', *Discrete Optimization*, Vol. 12, pp.129–146.
- Cordeau, J.F., Gendreau, M. and Laporte, G. (1997) 'A tabu search heuristic for periodic and multi-depot vehicle routing problems', *International Journal of Network and Management*, Vol. 30, No. 2, pp.105–119.

- Cornuéjols, G. (2008) 'Valid inequalities for mixed integer linear programs', *Mathematical Programming B*, Vol. 112, No. 1, pp.3–44.
- Dixit, A., Mishra, A. and Shukla, A. (2019) 'Vehicle routing problem with time windows using meta-heuristic algorithms: a survey', in Yadav, N., Yadav, A., Bansal, J., Deep, K. and Kim, J. (Eds.): *Harmony Search and Nature Inspired Optimization Algorithms, Advances in Intelligent Systems and Computing*, Springer, Singapore, p.741.
- Geetha, S., Poonthalir, G. and Vanathi, P.T. (2009) 'Improved k-means algorithm for capacitated clustering problem', *INFOCOMP Journal of Computer Science*, Vol. 8, No. 4, pp.52–59.
- Gil, A.F., Sánchez, M.G., Lalla-Ruiz, E. and Castro, C. (2020) 'Cumulative VRP with time windows: a trade-off analysis', *International Conference on Computational Logistics*, Springer, Cham.
- Goel, R. and Maini, R. (2017) 'Vehicle routing problem and its solution methodologies: a survey', *International Journal of Logistics Systems and Management*, Vol. 28, No. 4, pp.419–435.
- Jiang, H., Zhang, X., Li, M. and Che, H.Y. (2007) 'Using Gavish-Grave LP to formulate the directed black and white traveling salesman problem', *7th International Conference on Computational Science*, Springer, Berlin, Heidelberg.
- Khoufi, I., Laouiti, A. and Adjih, C. (2019) 'A survey of recent extended variants of the traveling salesman and vehicle routing problems for unmanned aerial vehicles', *Drones*, Vol. 3, No. 3, p.66.
- Kim, G., Ong, Y.S., Cheong, T. and Tan, P.S. (2016) 'Solving the dynamic vehicle routing problem under traffic congestion', *IEEE Transactions on Intelligent Transportation Systems*, Vol. 17, No. 8, pp.2367–2380.
- Kucharska, E. (2019) 'Dynamic vehicle routing problem-predictive and unexpected customer availability', *Symmetry*, Vol. 11, No. 4, p.546.
- Kumar, S. and Panneerselvam, R. (2014) 'A survey on the vehicle routing problem and its variants', *Intelligent Information Management*, Vol. 4, No. 3, pp.66–74.
- Moghdani, R., Salimifard, K., Demir, E. and Benyettou, A. (2020) 'The green vehicle routing problem: a systematic literature review', *Journal of Cleaner Production*, Vol. 279, p.123691.
- Morrison, D.R., Jacobson, S.H., Sauppe, J.J. and Sewell, E.C. (2016) 'Branch-and-bound algorithms: a survey of recent advances in searching, branching, and pruning', *Discrete Optimization*, Vol. 19, pp.79–102.
- Rabbani, M., Tahaei, Z., Farrokhi-Asl, H. and Saravi, N.A. (2017) 'Using meta-heuristic algorithms and hybrid of them to solve multi compartment vehicle routing problem', *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pp.1022–1026.
- Rahimi Baghbadorani, R., Zajkani, M.A. and Haeri, M. (2021) 'A new approach to solve MDVRP in lower computation time', *29th Iranian Conference on Electrical Engineering (ICEE)*, Tehran, Iran, 18–20 May.
- Ramos, T.R.P., Gomes, M.I. and Póvoa, A.P.B. (2020) 'Multi-depot vehicle routing problem: a comparative study of alternative formulations', *International Journal of Logistics Research and Applications*, Vol. 23, No. 2, pp.103–120.
- Reyes, D., Savelsbergh, M. and Toriello, A. (2017) 'Vehicle routing with roaming delivery locations', *Transportation Research Part C: Emerging Technologies*, Vol. 80, pp.71–91.
- Sabar, N.R., Bhaskar, A., Chung, E., Turky, A. and Song, A. (2019) 'A self-adaptive evolutionary algorithm for dynamic vehicle routing problems with traffic congestion', *Swarm and Evolutionary Computation*, Vol. 44, pp.1018–1027.
- Sadati, M.E.H., Aksen, D. and Aras, N. (2020) 'The r-interdiction selective multi-depot vehicle routing problem', *International Transactions in Operational Research*, Vol. 27, No. 2, pp.835–866.

- Salhi, S., Imran, A. and Wassen, N.A. (2014) 'The multi-depot vehicle routing problem with heterogeneous vehicle fleet: formulation and a variable neighborhood search implementation', *Computers & Operations Research*, Vol. 52, pp.315–325.
- Stodola, P. (2018) 'Using metaheuristics on the multi-depot vehicle routing problem with modified optimization criterion', *Algorithms*, Vol. 11, No. 5, p.74.
- Theurich, F., Fischer, A. and Scheithauer, G. (2021) 'A branch-and-bound approach for a vehicle routing problem with customer costs', *Euro Journal on Computational Optimization*, Vol. 9, p.100003.
- Wang, J., Sun, Y., Zhang, Z. and Gao, S. (2020) 'Solving multi-trip pickup and delivery problem with time windows and manpower planning using multiobjective algorithms', *IEEE/CAA Journal of Automatica Sinica*, Vol. 7, No. 4, pp.1134–1153.
- Yu, G. and Yang, Y. (2019) 'Dynamic routing with real-time traffic information', *Operational Research*, Vol. 19, No. 4, pp.1033–1058.
- Zhang, W., Gajpal, Y., Appadoo, S.S. and Wei, Q. (2020) 'Multi-depot green vehicle routing problem to minimize carbon emissions', *Sustainability*, Vol. 12, p.3500 [online] <https://doi.org/10.3390/su12083500>.