International Journal of Engineering & Technology, 9 (1) (2020) 1-8

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

A clustering algorithm for solving the vehicle routing assignment problem in polynomial time

L. W. Rizkallah [1], M. F. Ahmed [1] *, N. M. Darwish [1]

[1] *Computer Engineering Dept., Faculty of Engineering, Cairo University*
**Corresponding author E-mail: mona_farouk@eng.cu.edu.eg*

Abstract

The Vehicle Routing Problem (VRP) consists of a group of customers that needs to be served. Each customer has a certain demand of goods. A central depot having a fleet of vehicles is responsible for supplying the customers with their demands. The problem is composed of two sub-problems: The first sub-problem is an assignment problem where both the vehicles that will be used as well as the customers assigned to each vehicle are determined. The second sub-problem is the routing problem in which for each vehicle having a number of customers assigned to it, the order of visits of the customers is determined. Optimal number of vehicles as well as optimal total distance should be achieved. In this paper, an approach for solving the first sub-problem, the assignment problem, is presented. In the approach, a clustering algorithm is proposed for finding the optimal number of vehicles by grouping the customers into clusters where each cluster is visited by one vehicle. This work presents a polynomial time clustering algorithm for finding the optimal number of clusters. Also, a solution to the assignment problem is provided. The proposed approach was evaluated using Solomon's C1 benchmarks where it reached optimal number of clusters for all the benchmarks in this category. The proposed approach succeeds in solving the assignment problem in VRP achieving a solving time that surpasses the state-of-the-art approaches provided in the literature. It also provides a means of working with varying num-ber of customers without major increase in solving time.

Keywords: *Agglomerative Hierarchical Clustering; Clarke and Wright Saving Method; Clustering Algorithms; Solomon's Benchmarks; Vehicle Routing Problems.*

1. Introduction

Vehicle Routing Problems are applied in a wide range of applications. Examples of those applications are: transportation, communication, military, bank deliveries, postal deliveries, industrial refuse collection, dial-a-ride service, scheduling and many others. As a result of such range of applications, additional attributes are added to the problem to let it be used in more realistic situations and to be applied to further applications. Such attributes add additional constraints to the problem introducing different variants of the vehicle routing problem. For example, an additional constraint is added to the capacity of each vehicle preventing it from serving customers with total demands exceeding such capacity. This problem is called Capacitated Vehicle Routing Problem (CVRP). Another constraint can be added to the problem which restricts the service time of each customer to be within a time window; such variant is called Vehicle Routing problem with Time Window constraint (VRPTW).
This paper targets the assignment problem of the Vehicle Routing Problem. It proposes an enhancement in the solving time of this problem by introducing a clustering technique based on Clarke and Wright saving method.

2. Problem statement

A formulation of the problem is given as follows: Let G= (V, A) be a complete undirected graph [1] where V is the set of vertices and A is the set of edges. Vertex v_o is a single depot from which a number of vehicles with capacity Q depart to serve customers. The remaining vertices $v_i \in V\backslash\{v_o\}$ represent customers to be served where i={1,…n} and n is the total number of customers to be served. Each customer has a non-negative demand q_i, a service duration τ_i and a time window $[e_i, l_i]$ which is the interval of allowable visit times. By definition $q_0 = 0$ and $\tau_0 = 0$. Arc (i, j) represents an arc from v_i to v_j, hence it indicates that a vehicle can move from v_i to v_j directly. Each arc is associated with a travel distance c_{ij} and a travel time δ_{ij}. A feasible route r is defined as a sub-graph of G that starts and ends at v_0 where each vertex is visited only once, the time constraints are met and the sum of demands of customers on the route doesn't exceed Q. In order to model the start and the end of a route, another vertex is added to the set of vertices which is v_{n+1} representing the depot as the last vertex in the route; hence, the depot is modeled by two vertices v_0 and v_{n+1}. The maximum number of routes that can be constructed is m routes where m is the number of vehicles available at the depot. The aim of VRPTW is to construct a minimum number of routes while preserving the above constraints and minimizing the total distance traveled by all vehicles.

3. Related work

Literature contains many techniques that are used to solve the vehicle routing problem. Such approaches include exact methods [2, 3]. Other approaches include local search improvement heuristics [4, 5] and constructive heuristics [6]. Meta-heuristics approaches are also introduced such as beam search [7], tabu search [8], genetic algorithms [1] and many more.
The problem consists of two optimization sub-problems which are: minimizing the total number of routes (vehicles) where customers are assigned to each route and minimizing the total travel distance.

3.1. Assignment problem solving techniques

Since the major interest in this paper is the assignment problem, the techniques used to solve this sub-problem are highlighted. Simulated Annealing [9] was used to solve the assignment problem. The generation of the neighbors in the SA is based on the following move operators: 2-exchange, Or-exchange, relocation, crossover and exchange. The proposed approach [10] first tries to minimize the number of vehicles by using Route Minimization heuristic (RM) [11]. The technique adopted [12] depends on Particle Swarm Optimization (PSO) algorithm. The assignment problem was solved [13] using Ant Colony Optimization (ACO) technique. They modified the ACO to include a semi-greedy heuristic. The modified ACO is used in the clustering phase. Two-phase tabu search was used [14] where the first phase depends on dense packing of customers within a route. The second phase is a post processing phase that tries to split routes and introduce further vehicles as long as this enhances the solution. A branch and price method to solve the assignment problem was used [15]. K-Means was applied to generate the clusters [7]; since the number of clusters is unknown, a dichotomous search is used to try several values of the number of clusters and determine the best one. Dynamic programming was adopted [16] in which the VRP is decomposed into sub-problems. They use state transition for customers carrying states: if a vehicle passes by customer c, then the c^{th} element in the state is filled with the customers ID. All different carrying states for a vehicle that are feasible can be enumerated and the optimal ones are chosen. Branch-Price-Cut approach was followed [17] where a forward labeling algorithm was used in which each path is represented by a label. The algorithm starts with a single empty path then it should be extended by considering all feasible successors. Since such approach is time consuming, authors apply a dominance rule to identify the labels that can be discarded. In [28], the solution is first clustered by choosing the farthest node from the depot since it is the most critical one, then the algorithm starts to choose the closest nodes to that one to put with the same cluster. If the demand is exceeded a new cluster is formed by finding the farthest node. In [29] dynamic programming is used, and it is reported that it is most successful on the clustered instances.
As shown most of the algorithms used to solve the assignment problem are NP-complete. That means they require large amount of time to solve the problem. This work proposes an enhancement in the solving time of the assignment problem by introducing a clustering technique based on Clarke and Wright saving method.

3.2. Clustering techniques

This subsection gives a brief description of the clustering algorithms that are found in literature. In the Hierarchical Methods [18], clusters are built gradually either in top-down or bottom-up manner. There are two types of these methods which are Agglomerative hierarchical clustering and Divisive hierarchical clustering. In the Agglomerative hierarchical clustering, each object represents one cluster, and then merging of these objects is performed until a certain structure is reached. In Divisive hierarchical clustering, initially all objects belong to the same cluster, and then division into smaller clusters is performed until the desired structure is obtained. Another technique is the Partitioning Methods [19] where an initial partitioning of objects is obtained and then relocation is performed by moving instances from one cluster to another. The number of clusters should be pre-set by the user. In order to obtain the initial partitioning, enumeration of all possible partitioning should be made, since this is not feasible, heuristics are used. Example of such methods is the k-means where the objects are partitioned into k clusters. The center of each cluster is the mean of all objects that belong to the cluster. In Density-based Methods the objects in each cluster are assumed to be drawn from a certain probability distribution. The clusters and their distribution parameters should be identified. The main idea of this method is to keep adding objects to the cluster as long as the density of the cluster (i.e. the number of objects in the cluster) hasn't exceeded a certain predefined threshold [20]. The Model-based Methods try to fit the data points corresponding to the objects to be clustered into mathematical models. In these methods, each cluster represents a concept or a class [21]. In the Grid-based Methods the space is partitioned into a certain number of cells that forms the grid structure on which clustering operations are performed [22]. There are also the Soft-computing Methods to perform clustering [23]; for example, evolutionary approaches such as Genetic Algorithms are used in clustering in which the candidate clustering is encoded as chromosomes. Another soft computing technique used for clustering is the Simulated Annealing.

4. Methods

In this paper, we propose a clustering algorithm to solve the assignment sub-problem. The clustering algorithm groups the customers into clusters where each cluster is visited by one vehicle and hence the optimal number of routes (vehicles) is the number of clusters. The clustering algorithm proposed in this work is based on Clarke and Wright saving method [24]. Before proceeding with the approach, we will first introduce the Clarke and Wright Saving Method.

4.1. Clarke and wright saving method

One of the best-known constructive heuristics methods is Clarke and Wright saving method. Many papers are based on this method. The algorithm starts with an initial solution in which every customer is served by a different vehicle making every customer in a separate route. In order to merge routes together, the heuristic works by computing an n*n saving matrix where n is the number of customers. An entry in the matrix in row i and column j is given by the following equation:

$$S_{ij} = d_{0i} + d_{0j} - d_{ij} \quad (1)$$

Where d_{0i} is the Euclidean distance between the depot and customer i, d_{0j} is the Euclidean distance between the depot and customer j, and d_{ij} is the Euclidean distance between customer i and customer j. After computing this saving matrix, the heuristic searches for the largest

saving value in this matrix for a certain customer i, say this value is the entry S_{ij}, then customer i is linked with customer j and the two routes are merged as long as the capacity constraints aren't violated. When merging happens, the saving matrix is recomputed and the method is repeated to merge further routes. The heuristic is called saving as it computes the saving that will be obtained if customer j comes in the route after customer i. This method has been enhanced several times [6, 25].

4.2. The proposed clustering method

This work performs clustering based on the idea of the saving method by Clarke and Wright. In the saving method algorithms, the distance to the depot is taken into account as it measures the saving that will result from placing route j after i. Since we don't consider the routing in this phase, the distance to the depot is insignificant. Actually, it might mislead the clustering process. We will demonstrate such observation by applying Clarke and Wright saving algorithm to the example in Fig. 1.

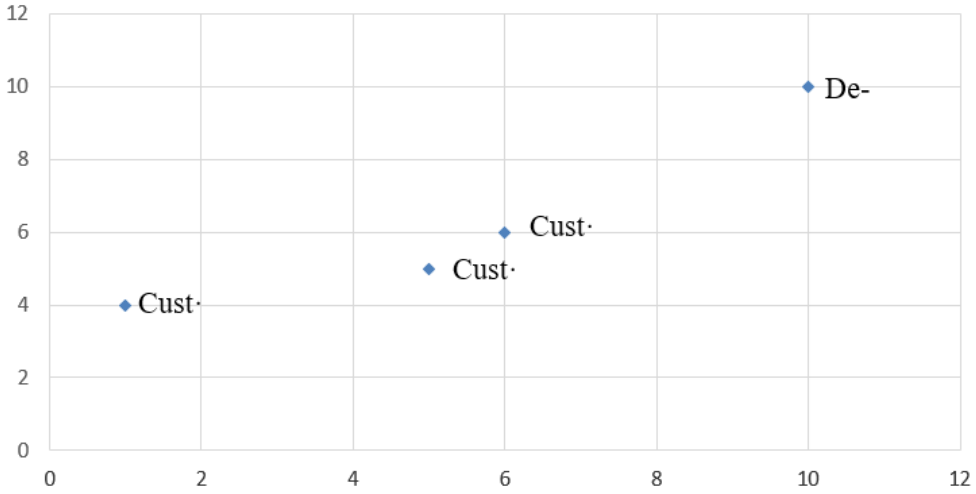

Fig. 1: Clarke and Wright Saving Method problem.

In Fig. 1 the depot is at (10, 10), customer i is at (5, 5), customer j is at (1, 4) and customer k is at (6, 6). We will apply the saving method equation, Eq. 1, for customer i to check if it is better to merge it with customer j or k. We have:

S_{ij} =7.1+10.8-4.1=13.8

S_{ik} =7.1+5.7-1.4=11.4

The saving method will choose merging customer i with j since S_{ij} is larger than S_{ik}, although it is obvious that merging i with k will result in better enhancement in the distance. This is because the distance to the depot affects the decision, so customers that are far from the depot may cause a large saving value although they might not be as near as other customers to customer i. This will result in a situation that may require more effort to reach the optimal solution in the subsequent sub-problems. Therefore, the clustering algorithm should consider the customers' mutual distance while performing the clustering. Consequently, this work modified the saving method algorithm. The clustering algorithm introduced in this work defines a saving matrix with dimensions n*n where each entry in the matrix is of the form:

$$S_{ij} = d_{ij} \quad (2)$$

Where d_{ij} is the Euclidean distance between customer (route) i and customer (route) j. The algorithm works by merging the customers that have the lowest saving value (not the highest as in the original method). This is due to the fact that the largest saving in distance will result from merging customers that are as close as possible to each other. Fig. 2 outlines the algorithm introduced in this work.

Algorithm 1: Clustering Algorithm

Input: A list of customers with their demands also their x and y coordinates
Output: c clusters of the customers

```
MoreGrouping=true;
While (MoreGrouping)
{
    MoreGrouping=false;
    ComputeSavingMatrix();
    foreach customer i
    {
        foreach customer j
            Determine if j will result in the largest saving

        K = customer that results in highest saving
        if (customer i is in a cluster and customer k isn't in a cluster )
        {
            if (the demands of the cluster of i + demand of k ≤ total capacity )
                Put customer k in the same cluster as i; MoreGrouping=true;
            Else
                Generate a new cluster and insert in it customer k
        }
        Else if (customer k is in a cluster and customer i isn't in a cluster)
        {
            if (the demands of the cluster of k + demand of i ≤ total capacity )
                Put customer i in the same cluster as k; MoreGrouping=true;
            Else
                Generate a new cluster and insert in it customer i
        }
        Else if (both customers i and k are in different clusters)
        {
            if (demands of cluster i + demands of cluster k ≤ total capacity )
                Merge the two clusters together; MoreGrouping=true;
            Else
                Don't merge clusters

        }
        Else if (both customers i and k aren't in a cluster)
        {
            Create a new cluster and insert customer i
            if (demand of customer i + demand of customer k ≤ total capacity)
                Put customer k in the same cluster as customer i; MoreGrouping=true
            Else
                Generate a new cluster and insert in it customer k

        }
    }
    foreach cluster c
        Consider cluster c as a new customer formed
        Demand of c=total demands of customers in cluster c
        X coordinate of c=average of x coordinates of customers in c
        Y coordinate of c=average of Y coordinates of customers in c
}
```

Fig. 2: The Proposed Clustering Algorithm

The algorithm performs the clustering based on the new saving method described. It first computes the saving matrix between all the customers. Then it determines the customers that can be merged together while preserving the capacity constraints. After passing by all customers, the clusters generated are considered as the new customers in the problem and the algorithm is applied on the new customers. The algorithm continues as long as there are clusters that can be merged together. Once no more clusters can be merged due to the capacity constraints violation, the algorithm terminates returning the optimal number of clusters. This is indicated by the Boolean variable named MoreGrouping. Since the proposed approach first starts by putting each customer in a cluster and then makes merging checks, therefore, this clustering algorithm is considered an Agglomerative hierarchical clustering as discussed in section 3.2.

4.3. Time complexity analysis

In this section, each part in the algorithm is analyzed in order to compute the time complexity. The number of customers in the problem is denoted by n. ComputeSavingMatrix() function computes n*n matrix, therefore, its time complexity is $O(n^2)$. Concerning the two nested foreach loops (i.e. foreach customer i and foreach customer j), their complexity is $O(n^2)$. The last foreach loop (i.e. foreach cluster c) has a time complexity of O(C) where C is the number of clusters and $C \leq n$. The outermost while loop is repeated as long as the capacity constraints are met to further merge more clusters. Once the capacity constraints are violated by further merging, no more clusters are merged and the algorithm terminates reporting the number of clusters found by the algorithm as well as the customers in each cluster. Therefore, the worst-case complexity of repeating the while loop (i.e. While (MoreGrouping)) in order to merge the clusters is O(log(n)) which will be reached when all clusters are merged together and the algorithm outputs only one cluster having all customers. This makes the worst-case complexity of the algorithm:

$$O(\log(n)*(n^2 + n^2 + c)) \quad (3)$$

$$=O(\log(n)*(2*n^2 + c))$$

$$\cong O(\log(n)*(2*n^2)) \text{ (Since } c << n)$$

$$\cong O(\log(n)*(n^2))$$

$$\cong O(n^3)$$

However, such complexity is unlikely to be reached due to the capacity constraint added to the problem which makes it unlikely for a vehicle capacity to fit all customers' demands meaning that such algorithm will require a time less than $O(\log(n)*n^2)$ to be solved.

5. Results

The proposed algorithm was tried on Solomon's benchmarks [26, 27]. Solomon generated benchmarks problems that point out several factors affecting the behavior of the routing and scheduling algorithms. These factors include: geographical location of customers, the number of customers serviced by a vehicle, percentage of time-constrained customers and tightness and positioning of the time windows. Solomon benchmarks include three sets where each set contains benchmarks having a certain number of customers. The numbers of customers in the sets are 25, 50 and 100 customers.

Each benchmark contains the following parameters:

- The maximum number of vehicles that can be used
- The maximum capacity of each vehicle (same for all vehicles)
- Customers Details. For each customer the following details are given:
- The X and Y Coordinates of the customer defining his geographical position.
- The customer's Demand.
- The Service Time of the customer.
- The Time Window of the customer.

In order to assess the effectiveness of this work, two main factors need to be considered:

1) The number of benchmarks that reached the optimal solutions.
2) The approach's solving time.

Each of the above factors will next be reported and analyzed.

This work targeted the C1 category of Solomon's benchmarks and applied the new algorithm on the three sets having n=25, 50 and 100 where n is the number of customers. Each set contains 9 benchmarks to make a total of 27 benchmarks. In each set the optimal number of vehicles is the same. Table 1 summarizes this work's output giving the number of vehicles reached by the algorithm for each of the three sets and the solving time the algorithm takes to reach such solution.

Table 1: Summarized Results for the Proposed Work

Number of customers	Number of vehicles	Solving time in seconds
25	3	0.04
50	5	0.04
100	10	0.05

The first column of Table 1 identifies the number of customers, the second column gives the number of vehicles resulted from the approach and the third column gives the approach's solving time. If this work reaches the optimal number of vehicles, then such value is written in bold in the second column indicating that this is the optimal solution. Optimal solutions can be found in [4, 7, 16 and 17]. The results found by this work indicate that the number of vehicles reached by the approach is the same as the number of vehicles of the optimal solutions which are 3, 5 and 10 vehicles for n=25, n=50 and n=100 respectively. This work reached optimal solutions for 27 out of 27 of the benchmarks.

Considering the second factor, the graph in Fig. 3 illustrates the effectiveness of this work's solving time.

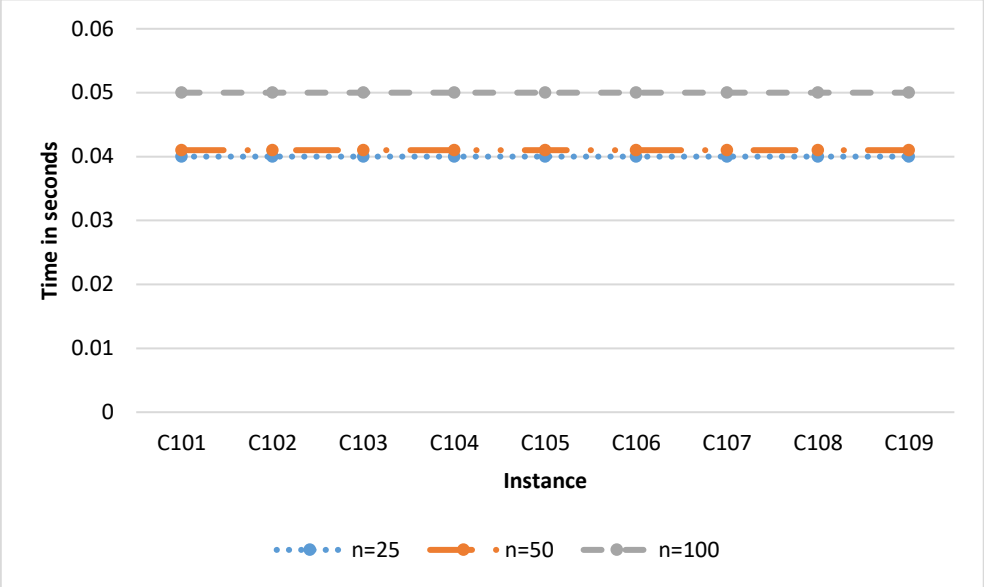

Fig. 3: Solving Time for C1 Category.

In the graph there are three curves, one for each value of n as shown. The x-axis represents the instance and the y-axis represents the time of the clustering algorithm in seconds. We used Intel® Core™2 Duo processor 2.1 GHz. For n=25, the clustering algorithm took 0.04 seconds on all the benchmarks in this subcategory. For n=50, the algorithm took 0.04 seconds. For n=100, the algorithm took 0.05 seconds. Fig. 4 clarifies the relation between the number of customers and the solving time.

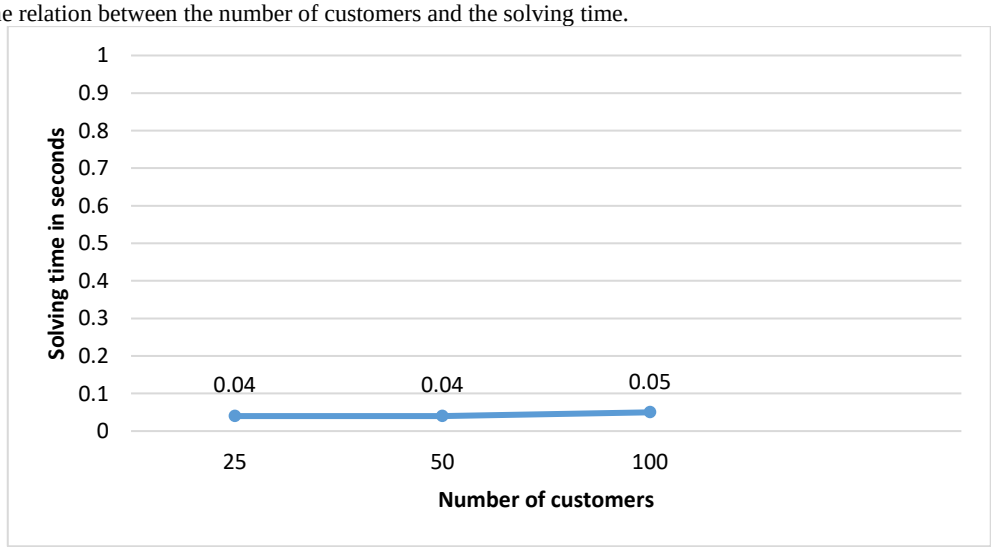

Fig. 4: Number of Customers vs. Solving Time.

The x-axis represents the number of customers and the y-axis represents the corresponding solving time the algorithm takes for each number. It illustrates how the increase in the number of customers affects the increase in the solving time. As shown, the graph seems linear approaching a constant function meaning that the increase in the number of the customers doesn't cause an observable increase in the solving time which is a major problem in the NP-hard problems proving that this work successfully presents a polynomial time clustering algorithm for solving an NP-hard problem.

In order to assess the competitiveness of this work, we compared this work's solving time with the solving time of other approaches found in the literature that reached optimal solutions as well. Table 2 clarifies such comparison.

Table 2: Comparison to Other Approaches' Solving Times

Approach used	Solving time class	Reference
Simulated Annealing	NP-complete	Bent and Van Hentenryck (2004) [9]
Branch and Price	NP-complete	Prescott et al. (2009) [15]
Route Minimization Heuristic	NP-complete	Nagata et al. (2010) [10]
Ant Colony Optimization	NP-complete	Chen et al. (2012) [13]
Particle Swarm Optimization	NP-complete	Hu Wenbin et al. (2013) [12]
Tabu Search	NP-complete	Jiang Jun et al. (2014) [14]
K-means clustering	NP-hard	Akeb et al. (2015) [7]
Modified Clarke and Wright saving method	Polynomial	This work (2018)

The first column of Table 2 gives the name of the approach used, the second column gives the time complexity class of such approach and the third one gives the reference of the approach. The results in Table 2 indicate that this work reached the best solving time class for the

assignment problem in the vehicle routing problem. To our knowledge, no other approaches in the literature achieved a better solving time for such problem.

6. Discussion

From the results, it is noted that this work achieved the best time complexity in solving the assignment problem in VRPTW. The algorithm reached the optimal number of clusters for the Solomon's C1 benchmarks. Table 1 clarifies this point and also reports the running time of the instances to support such finding. It is also noted that the running time does not increase with increasing the number of customers, and this can be seen in Table 1 from the running times for instances with 25, 50 and 100 customers. Fig. 4 reports the change in the running time with the increase in the number of customers and it is observed that the time is nearly constant. From Table 2, it is found that all other algorithms that were tried on Solomon's benchmarks and solved the assignment problem first have time complexities larger than this work. This work is proved to achieve the best time complexity for solving the assignment problem in Solomon's C1 category.

7. Conclusion and future work

This work successfully introduces a polynomial time clustering algorithm that solves the assignment sub-problem in the vehicle routing problem. The algorithm determines the optimal number of clusters as well as the customers in each cluster. This work tried the new algorithm on Solomon's C1 benchmarks and it reached optimal number of clusters for all the benchmarks in this category. The algorithm proved to be efficient when applied on different number of customers since it doesn't cause a tremendous increase in solving time by increasing the number of customers. The algorithm also simplifies solving the second sub-problem (which is minimizing the total travel distance), since each vehicle only traverses one cluster containing the customers that are as near as possible to each other and hence the overhead of moving from a cluster to another is eliminated meaning that moving between customers that are relatively far away from each other is excluded from the beginning.
Future work directions include further enhancement in the algorithm's time complexity. Also, other categories of benchmarks can be tried. Another direction can explore the subsequent sub-problem which is the routing problem in order to determine the order of visits of the customers in each cluster.

References

[1] Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N. and Rei, W., 2012. "A Hybrid Genetic Algorithm for Multi-Depot and Periodic Vehicle Routing Problems", Operations Research, Vol. 60(3), pp. 611-624. https://doi.org/10.1287/opre.1120.1048.

[2] Fukasawa, R., Longo, H., Lysgaard, J., Poggi de Aragão, M., Reis, M., Uchoa, E. and Werneck, R. F., 2006. "Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem", Mathematical Programming, Vol. 106(3), pp. 491-511. https://doi.org/10.1007/s10107-005-0644-x.

[3] Lysgaard, J., Letchford, A.N. and Eglese, R.W., 2004. "A New Branch-and-Cut Algorithm for Capacitated Vehicle Routing Problems", Mathematical Programming, Series A, Vol. 100, pp. 423-445. https://doi.org/10.1007/s10107-003-0481-8.

[4] Dhahri, A., Zidi, K. and Ghedira, K., 2014. "Variable Neighborhood Search Based Set Covering ILP Model for the Vehicle Routing Problem with Time Windows", Procedia Computer Science, Vol. 29, PP. 844–854. https://doi.org/10.1016/j.procs.2014.05.076.

[5] Hong L., 2012. "An Improved LNS Algorithm for Real-Time Vehicle Routing Problem with Time Windows", Comput Oper Res, Vol. 39(2), pp. 151–163. https://doi.org/10.1016/j.cor.2011.03.006.

[6] Pichpibul, Tantikorn and Kawtummachai, R., 2012. "An Improved Clarke and Wright Savings Algorithm for the Capacitated Vehicle Routing Problem", Science Asia, Vol. 38(3), pp. 307-318. https://doi.org/10.2306/scienceasia1513-1874.2012.38.307.

[7] Akeb, Hakim, Bouchakhchoukha, A. and Hifi, M., 2015. "A Three-Stage Heuristic for the Capacitated Vehicle Routing Problem with Time Windows", Recent Advances in Computational Optimization, Vol. 580, pp. 1-19. https://doi.org/10.1007/978-3-319-12631-9_1.

[8] Cordeau, J. F., Laporte, G. and Mercier, A., 2011. "A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows", Journal of the Operational Research Society, Vol. 52, pp. 928-936. https://doi.org/10.1057/palgrave.jors.2601163.

[9] Bent, R.,Van and Hentenryck, P., 2004. "A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows", Transp Sci, Vol. 38(4), pp. 515–530. https://doi.org/10.1287/trsc.1030.0049.

[10] Nagata Y., Bräysy O. and Dullaert W., 2010. "A Penalty-Based Edge Assembly Memetic Algorithm for the Vehicle Routing Problem with Time Windows", Comput Oper Res, Vol. 37(4), pp. 724–737. https://doi.org/10.1016/j.cor.2009.06.022.

[11] Nagata, Yuichi and Bräysy, O., 2009. "A powerful Route Minimization Heuristic for the Vehicle Routing Problem with Time Windows", Operations Research Letters, Vol. 37(5), pp. 333-338. https://doi.org/10.1016/j.orl.2009.04.006.

[12] Wenbin, H., 2013. "A Hybrid Chaos-Particle Swarm Optimization Algorithm for the Vehicle Routing Problem with Time Window", Entropy, Vol. 15(4), pp. 1247-1270. https://doi.org/10.3390/e15041247.

[13] Ruey-Maw, C., Hsieh, F., and Di-Shiun, W., 2012. "Heuristics Based Ant Colony Optimization for Vehicle Routing Problem", Industrial Electronics and Applications (ICIEA), 2012 7th IEEE Conference.

[14] Jun, J., 2014. "Vehicle Routing Problem with a Heterogeneous Fleet and Time Windows", Expert Systems with Applications, Vol. 41(8), pp. 3748-3760. https://doi.org/10.1016/j.eswa.2013.11.029.

[15] Prescott-Gagnon, E., Desaulniers, G., and Rousseau, L.-M., 2009. "A branch-and-Price Based Large Neighborhood Search Algorithm for the Vehicle Routing Problem with Time Windows", Networks, Vol. 54(4), pp. 190-204. https://doi.org/10.1002/net.20332.

[16] Mahmoudi, Monirehalsadat, and Zhou, X., 2016. "Finding Optimal Solutions for Vehicle Routing Problem with Pickup and Delivery Services with Time Windows: A Dynamic Programming Approach Based on State–Space–Time Network Representations", Transportation Research, Part B: Methodological, Vol. 89, pp. 19-42. https://doi.org/10.1016/j.trb.2016.03.009.

[17] Diego, P., 2017. "New Enhancements for the Exact Solution of the Vehicle Routing Problem with Time Windows", INFORMS Journal on Computing, Vol. 29(3), pp. 489-502. https://doi.org/10.1287/ijoc.2016.0744.

[18] Everitt, B. S., Landau, S., Leese, M., and Stahl, D., 2011. "Hierarchical Clustering", In: Cluster Analysis, 5th ed., John Wiley & Sons Ltd, Chichester, UK. https://doi.org/10.1002/9780470977811.

[19] Madhulatha TS., 2012. "An overview on Clustering Methods", Journal of Engineering arXiv: 1205.1117., Vol. 2(4), pp. 719-725. https://doi.org/10.9790/3021-0204719725.

[20] Kriegel HP, Kröger P., Sander J., and Zimek A., 2011. "Density-Based Clustering", Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, Vol. 1(3), pp. 231-240. https://doi.org/10.1002/widm.30.

[21] Bouveyron, Charles, and Brunet-Saumard, C., 2014. "Model-Based Clustering of High-Dimensional Data: A Review", Computational Statistics & Data Analysis, Vol. 71, pp. 52-78. https://doi.org/10.1016/j.csda.2012.12.008.

[22] Pradeep, R., Singh, S., 2010. "A Survey of Clustering Techniques", International Journal of Computer Applications, Vol. 7(12), pp. 1-5. https://doi.org/10.5120/1326-1808.

[23] Hruschka, E., Carlos, S., Campello, R., Freitas, A., and De Carvalho, A., 2009. "A Survey of Evolutionary Algorithms for Clustering. Systems", Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions, Vol. 39(2), pp. 133-155. https://doi.org/10.1109/TSMCC.2008.2007252.
[24] Barnhart, Cynthia, Laporte, G., 2006. "Handbooks in Operations Research & Management Science", Transportation, Vol. 14, pp. 367-417.
[25] Caccetta, Louis, Alameen, M., and Abdul-Niby, M., 2013. "An Improved Clarke and Wright Algorithm to Solve the Capacitated Vehicle Routing Problem", Engineering, Technology & Applied Science Research, Vol. 3(2), pp. 413.
[26] Vehicle Routing Problem. [Internet]. Spain: University of Malaga [updated 2013 Jan 7; cited 2016 May 18]. Available from: http://neo.lcc.uma.es/vrp/
[27] Solomon Benchmarks. [Internet]. Norway: Company of Scientific and Industrial Research at the Norwegian Institute of Technology [updated 2008 April 18; cited 2016 May 18]. Available from: https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark.
[28] Shin, K., & Han, S. (2012). A centroid-based heuristic algorithm for the capacitated vehicle routing problem. *Computing and Informatics*, *30*(4), 721-732.
[29] Desrochers, M., Desrosiers, J., & Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, *40*(2), 342-354. https://doi.org/10.1287/opre.40.2.342.