STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **x**, Month 202x, pp 0–19.
Published online in International Academic Press (www.IAPress.org)

A two-phase constructive approach for capacitated vehicle routing problem in constructing population of solutions

Noor Nabeel [1,2,*], Syariza Abdul-Rahman [2], Juliana Wahid [3]

[1] *Department of Information Technology Management, Technical College of Management, Northern Technical University, Iraq*
[2] *School of Quantitative Sciences, Universiti Utara Malaysia, Malaysia*
[3] *School of Quantitative Sciences (AHSGS), Universiti Utara Malaysia, Malaysia*

Abstract The Capacitated Vehicle Routing Problem (CVRP) is a variant of VRP that has a fleet of same capacity vehicles in order to serve customers with the shortest distances. The tendency of researchers to employ population-based metaheuristics in solving CVRP requires constructing the population of initial solutions to be improved later. However, the initial solutions constructed should be feasible to ensure that the solutions can be converged in a reasonable computation time. This study proposed a two-phase constructive approach for CVRP that shows the ability to generate feasible solutions for all instances of datasets used in this study. The approach starts with the descending order of the customers according to their demands followed by assigning customers to vehicles in sequence considering the capacity of vehicles. If there are customers still unassigned, neighborhood structures are applied on the already assigned customers to get more space to accommodate the unassigned customers aiming to produce the base feasible solution. In the second phase, the population is generated by applying neighborhood structures to get a variety of feasible solutions. The proposed approach was applied on CVRP benchmark datasets A, B, P and E. The results show the superiority of the proposed approach in comparison with random construction method in terms of cost and run time. In addition, the two-phase construction approach shows feasibility-oriented results when compared with other constructive approaches proposed in the literature.

Keywords: Two-phase construction, Vehicle Routing Problem (VRP), Metaheuristic algorithms, Construction algorithms, Order by demands.

AMS 2010 subject classifications: 90C35, 90C59

DOI: 10.19139/soic-2310-5070-3744

1. INTRODUCTION

The Vehicle Routing Problem (VRP) generally involves finding the shortest path for a group of vehicles to serve a collection of customers while passing through each customer only once and returning to the same depot from which the vehicles start [1]. There are several constraints of VRP. Some VRP constraints are related to vehicle capacity, time window, and single or multiple depots. In Capacitated VRP (CVRP), the capacity of all vehicles is the same, and they share the same depot to start from and end at aiming to serve a set of customers, and each customer should be served by a vehicle only once. The solution of CVRP should obtain the shortest total distance, as its objective [2]. CVRP is categorized as NP-hard problems in which it is difficult to find a solution [3]. In solving CVRP, metaheuristic algorithms are commonly utilized, especially when hybridizing both types of metaheuristics local search and population-based algorithms. Population-based algorithms require constructing the population of feasible initial solutions prior to the improvement phase as in [4], [5], and [6]. After preparing the initial solutions using

*Correspondence to: Noor Nabeel (Email: noor.nabeel@ntu.edu.iq). Department of Information Technology Management, Northern Technical University. Aliminasa Street, Mosul, Ninava Province, Iraq (41001).

ISSN 2310-5070 (online) ISSN 2311-004X (print)

constructive algorithms, the role of improvement algorithms begins to improve the solution with the shortest distance. Several improvement algorithms have been utilized in the improvement phase, such as Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Genetic Algorithm (GA), and Whale Optimization Algorithm (WOA). The tendency of researchers to employ population-based metaheuristics in solving CVRP requires constructing the population of initial solutions to be improved later. Each solution generated in the construction phase is aimed at a feasible one, i.e., has no constraint violation to make sure the solution improvement will not take longer time to converge or struggle to find a feasible solution. However, research on the constructive algorithm for VRP is very scarce.

Several construction approaches for VRP have been applied in the literature, such as the parallel insertion algorithm [7], sequential construction [8], the Nearest Neighbor heuristic [9], Clarke and Wright Savings heuristic [10], and Sweep algorithm [11]. Most construction approaches consider distinct criteria such as distance, angle of customer location with the depot, savings (combining two points on a single trip), and/or other criteria. Nevertheless, customer's demands are one crucial criterion to consider in constructing initial solutions, since they directly impact route planning and vehicle utilization. Thus, assigning demands to vehicles must be implemented in a way that accommodates demands to obtain an efficient route while optimizing resources.

This study focuses on producing a construction method based on customer demands that utilized the most popular datasets in CVRP, which are A, B, P from [12] and E from [13]. This study improves routing decisions in CVRP by aligning the construction algorithm with customer demands and requires a very minimum number of parameters, since it only needs a population size. By integrating customer-centric insights, this approach improves efficiency and effectiveness, since the proposed method provides reasonable quality initial solutions in reasonable time for population-based improvement algorithms, while the utilization of popular datasets ensures robustness and practical applicability, making it a valuable contribution to the field. The rest of this paper is organized as follows; Section 2 provides the related works, while the proposed approach is described in Section 3. The results are presented in Section 4. Finally, the conclusions are summarized in Section 5.

2. RELATED WORKS

The Construction of an initial solution for the VRP is to ensure that the solution is feasible and provides a good starting point for optimization algorithms. Greedy algorithms have been employed to construct initial solutions for VRP by selecting the best available options at each step. For example, choosing the closest customer to the current location or selecting customers based on certain criteria like demand or time windows. The construction of a population of initial solutions can be implemented either sequentially, by constructing the routes of each solution one by one, or in parallel, by constructing the routes concurrently for each solution. Nevertheless, the quality and the processing time of the solution in sequential construction is found to be better than parallel construction [14]. This section reviews related works on previous approaches for solution construction in VRP aiming to show the construction criteria employed in previous studies.

Parallel insertion algorithm proposed by [7] constructs routes in parallel. In the beginning, the value of total demands of all nodes divided by the capacity constraint is assigned to *Kmin* which represents the value of minimum feasible routes. Routes are constructed concurrently with different nodes that have different distances from the depot. Thereafter, each node that was added to the route will be checked if it is the nearest node to the previous node in the route and does not violate the capacity constraint. When no more nodes can be added to the route due to no available capacity on the vehicle, unrouted node will be inserted to another route and the procedure iterated until all nodes are assigned. The approach was also applied in [2]. In solving CVRP using sequential insertion approach proposed by [8], a route is started with a random node. The next node is selected based on the shortest distance and concerned with the capacity constraint. The steps are repeated until no more demands can be accommodated by the vehicle. The sequential approach was also implemented by [15].

The Nearest Neighbor (NN) heuristic method was introduced by [9] that constructed the routes based on sequence. The first node in each route should be the nearest available node to the depot. The following nodes in the second step are selected according to their nearness to the previous node in the route while keeping in mind the vehicle

capacity. The second step is repeated if vehicle capacity constraint is not violated otherwise, a new route is started from the first step. The method was also applied by [16]. Savings heuristic introduced by [10] was applied in CVRP by [17] in such a way that each costumer node is being a part of a route starts and ends at depot (n0, ni, n0). Thus, a total of N – 1 routes are created where N is the total number of nodes including depot. In the second step, the saving for each pair of nodes is calculated by saving $s(n_i,n_j) = c(v_i,v_0) + c(v_0,v_j) - c(v_i,v_j)$, where c is the cost, $i \neq 0, j \neq 0, i \neq j$. List S is created to contain all savings sorted in non-increasing order. In the third step, the first saving in S is considered. Then, the existence of $route_x$ and $route_y$, where $x \neq y$, having the condition of edge $(i,0)$ in $route_x$ and edge $(0,j)$ in $route_y$. Moreover, the total demands of $route_x$ and $route_y$ should be less than or equal to vehicle capacity. When these two conditions are met then $route_x$ and $route_y$ are combined to produce $route_{new}$ after removing $(i,0)$ and $(0,j)$. Thereafter, whether these routes existed or not, the current saving is discarded and the next one in the list is checked. The third step is iterated until the number of routes equals to the number of vehicles. Despite the good performance of Clarke and Wright Savings algorithm in the beginning of its work, it tends to generate low quality routes in the end due to adding nodes located on the borders. Therefore, [13] and [18] proposed to improve the savings calculations. The saving is calculated as $s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij}$ where λ represents the distance between nodes to be joined.

Ewbank et al. [19] proposed a constructive approach based on Fuzzy C-Means (FCM) clustering for the CVRP. The proposed method first groups customers into clusters using the FCM algorithm and then constructs vehicle routes for each cluster. The experimental results showed that the proposed approach produced near-optimal solutions with an average percentage error below 5% for most benchmark instances while requiring relatively short execution times, making it suitable for solving large-scale CVRP instances.

Another constructive approach is sweep algorithm that selects nodes according to their angle with depot and distance to the last assigned node in the solution as in [11]. The algorithm starts at the depot and sweeps the customer locations in a counterclockwise direction. It is simple and fast, but it is not guaranteed that one will always obtain the shortest path solution. For example, a study by [20] applied a sweep algorithm with other heuristics to solve VRP waste collection and the results of sweep algorithm were not competitive. Subgroup of Cluster-First-Route-Second method composed of clustering and routing steps. During the first phase, a number of clusters equal to the number of routes are created to distribute the customers on them. During the second phase (routing phase), customers are ordered in each cluster. A variant algorithm introduced by [21] of the Subgroup of Cluster-First-Route-Second can be applied to all instances of CVRP as in [22]. In clustering phase, a node (vertex) is selected for each route to represent a seed node $v_{seed(k)}$ where k is vehicle (route) number. So, there will be K clusters equal to the number of vehicles and K nodes selected as seeds. The selection of seed node in each cluster can be based on either the maximum distance from the depot or according to the maximum demand. In the second step, the cost of adding each node to each cluster is calculated as $cost^k_{v_i} = (v_0,v_i) + c(v_i,v_{seed(k)}) - c(v_{seed(k)},v_0)$. Next, the algorithm depends on $cost^k_{v_i}$, d_i and the capacity of the vehicle to solve the Generalized Assignment Problem, which is the problem of inability to solve all instances of the problem using a specific method. Thus, during this step, the node with the minimum cost for a specific route will be added to this route. In routing phase, the nodes are re-ordered for each route by solving Traveling Salesman Problem (TSP) for each route. This approach opens the way to previous approaches to generate better quality solutions.

The method of Subgroup of Route-First-Cluster-Second begins firstly as TSP and then the nodes are divided into K routes. Applying this method requires an unlimited number of vehicles. In general, the results of employing this method are inferior compared to the results of other constructive methods [7]. Applying this method requires unlimited number of vehicles. In general, the results of employing this method are inferior compared to results of other constructive methods. Due to this reported inferiority, together with the requirement to solve a TSP over the entire customer set prior to route partitioning, a cost that grows sharply with instance size, Route-First-Cluster-Second (giant-tour) methods were judged impractical to include as an empirical baseline at the scale of the 87 instances (up to n = 101) evaluated here, and are discussed only as a point of reference from the literature. The Greedy Randomized Adaptive Search Procedure (GRASP) by [23] to arrange customers by their distances from the depot and assign customers to routes is performed simultaneously. The algorithm iterates until it confirms the allocation of all customers to routes. Applying this algorithm sometimes comes with insufficient number of feasible solutions to fulfill the required population size. The procedure was used by [24] to solve VRP. While the

majority of construction methods reviewed in this study show the ability to generate feasible solutions for the solution population, GRASP may produce infeasible solutions. The infeasible solution in VRP is usually undesirable since the algorithm requires much time exploring the search space from scratch, for the solution to be converged. The VRP with relaxed rules presented by [25] that classified customers to groups according to their importance while serving the customers performed with satisfying d-relaxed priority in which the commitment of priority is respected completely, partially, or even none depending on the value of parameter d that balances between human needs and operational needs. In addition, another rule is to be satisfied which is order of demands fulfillment to ensure satisfying all top priority customers at the end of the trip. Insertion heuristic introduced by [26] to solve VRP through selecting a number of seed customers to initialize the available routes. The selection of seed customer for each route is performed based on a certain criterion like nearest, farthest, demand, or random. The unrouted customers are inserted to routes under the condition of concerning feasibility according to VRP variant. Cheapest insertion heuristic that uses a combination of two seed types (farthest or random, nearest or random, nearest or farthest) is proposed by [27] to solve CVRP. Filling routes with unrouted customers is accomplished by selecting the nearest unrouted customer to the previously routed customer. The best combination in term of quality was farthest and random (FR) which evaluated by experiencing a number of CVRP instances with diverse complexity. Table 1 summarizes the reviewed construction approaches in terms of their selection criteria for solving VRP. The steps of the construction approaches are typically involved the ordering of the customers by distance, customers' angle from the depot, distance saving, group priority, and distance or random to construct the solution. It is important to distinguish the proposed demand-sorted construction approach from traditional Cluster-First-Route-Second (CFRS) heuristics (e.g., Fisher Jaikumar [21]). In CFRS approaches, high-demand or distant customers are selected as seed nodes to partition the customer set into K distinct geometric or assignment-based clusters prior to routing. In contrast, TPC does not perform any spatial or assignment clustering stage. Instead, TPC sorts all customer demands globally in descending order and sequentially packs customers directly into vehicle routes under capacity constraints. Remaining unassigned customers are accommodated via local neighborhood operators (Move and Swap). Thus, TPC operates as a direct global route construction mechanism rather than a two-stage clustering framework. Thus, generating feasible CVRP population with reasonable quality by an efficient construction algorithm that orders customers by demands in reasonable time may compete with other methods.

Table 1. Criteria in construction methods for VRP

Selection Criteria	**Author(s)**
Distance	[2], [7], [8], [15], [9], [16], [20], [22], [23]
Distance savings	[10], [17], [13]
Angle and distance	[11], [20]
Group priority	[19], [25]
Distance or random	[26], [27]

3. METHODOLOGY

In the random approach, the method starts with producing a random matrix with $population\_size$ rows and $dim-1$ columns where dim represents the number of dimensions in the instance. The number of random numbers is $dim-1$, because the depot is excluded and the content of random matrix are numbers generated randomly ranging from one to $dim-1$ for each row. Thereafter, an initial solution is generated starting with number zero (the depot) followed by numbers sequencing according to the sequence of numbers in corresponding row of random matrix in such a way that one customer is inserted into the route each time while summing the total demands of the customers in the route to ensure it is not exceeding the $vehicle\_capacity$. When the vehicle of the route cannot accommodate more demands, the route ends with zero which represents back to depot. The size of each solution will be equal to the $(number\ of\ customers + number\ of\ vehicles) + 1$. The process of assigning customers from random matrix

to construct routes continue in sequence until no more customers remained in random matrix row. The process is repeated until *population_size* is met.

The proposed two-phase constructive approach is designed to start with finding space on vehicles for high demand customers which affect simplifying and speeding up assigning customers to vehicles in general. The reason behind this approach is to cater the difficulty in obtaining feasible solution and long running time especially for large instances with high customers' demands, large number of customers, with limited capacity and limited number of vehicles to accommodate all demands to the specified vehicles. As shown in the pseudocode in Figure 1, the first phase involves setting how many solutions to be generated (*population_size*), number of available vehicles (*max_vehicles*), number of dimensions (customers and depot) (*dim*), maximum vehicle capacity (*vehicle_capacity*). Only the first parameter needs to be set since other parameters can be read from the instance file. Two vectors are created during this phase that are *cust_dem* vector which contains customers' demands including the depot ordered according to customer number in ascending order. Thereafter, *nodes_by_demand* vector is created in which customers are ordered according to their demands in descending order as shown in Figure 2. Let N be the set of customers and $dem[i]$ is customer i demand. The customers in *nodes_by_demand* vector are ordered as presented in Equation (1):

$$dem[nodes\_by\_demand[i]] \geq dem[nodes\_by\_demand[i+1]] \quad \text{for } i = 1,2,\ldots,N \tag{1}$$

The generation of base solution vector is started by sequentially allocating nodes from *nodes_by_demand* vector while considering capacity of vehicles by checking the value of *route_load* to not exceed *vehicle_capacity* value and to start and end each route of base solution at depot. Moreover, *nodes_count* variable to count each node that has been served by a vehicle and its value at the end of construction should equal the number of customers to satisfy serving all customers constraint. For each route r in base solution, $r = \{0, v_i, v_{i+1}, \ldots, v_n, 0\}$, the summation of demands in r shouldn't exceed vehicle capacity Q in such that:

$$\sum_{i=1}^{n} dem_{vi} \leq Q \tag{2}$$

Figure 3 shows an example of base solution for instance A-n32-k5 where customer 20 is represented as 19 (bold) using vehicle 1 and route 1. If it is not possible to assign all nodes into base solution with commitment of number of vehicles constraint due to a large number of customers and huge demands, then these nodes are allocated beyond the last vehicle.

Move neighborhood structure is implemented to reallocate unassigned nodes into vehicles that are within the range of available vehicles and with considering capacity constraint as shown in Figure 4 where customer 10 (in bold font) is reallocated from beyond last vehicle to vehicle 1 since adding the load of customer 10 will not exceed the capacity of vehicle 1.

If there are still unassigned customers that cannot be assigned to vehicles within the available range, a swap neighborhood structure is performed between customers within the range of available vehicles in order to change the load on vehicles hoping to gain more capacity that can accommodate customers in extra vehicles. In the second phase in Figure 1, the population of initial solutions will be generated according to the base solution. Once a feasible base solution is constructed with all nodes assigned to the available vehicles, a number of neighborhood structures selected by probability that are: 1) unconditional route move, 2) unconditional move, 3) move and 4) swap are performed to generate various feasible solutions and to enhance initial solution quality.

For unconditional route move, a route is selected randomly to change its sequence in solution vector. To get more variety and get better solution cost, an iteration of move and swap neighborhood structures are applied. As shown in Table 2, where *tD* is the total distance and t is the execution time, the maximum iteration was set to 12 based on a parameter sensitivity analysis conducted on representative benchmark instances covering the beginning, the middle, and the end of datasets A, B, P, and E. Different maximum iteration values were evaluated, and the results showed that setting the maximum iteration to 12 consistently produced the lowest average solution cost across the tested instances. Increasing the maximum iteration beyond 12 did not provide further improvement and, in most cases, resulted in higher solution costs while requiring additional computational effort. Therefore, the maximum iteration was fixed at 12 in all subsequent experiments. The deterioration in solution quality beyond 12 iterations indicates

```
Start
Set population_size, max_vehicles, dim, vehicle_capacity
Create cust_dem vector, Create nodes_by_dem vector
Phase 1: Construct a base solution by sequentially allocating
    nodes from
nodes_by_dem
nodes_count=0, route_load=0, zero_counter=0
base_sol = Const_base_sol(nodes_by_dem)
for i=0 to ((dim-1)){
if(i ==0){
  base_sol [i] = 0;
  zero_counter++;
  base_sol[i+1] = nodes_by_dem [i];
  route_load
  = cust_dem[nodes_by_dem[nodes_count]];
  nodes_count++;}
elseif(i>0){
  if (route_load
    + cust_dem[nodes_by_demand[nodes_counter]]
    > vehicle_capacity){
    base_sol[i]= 0;
    route_load=0;
    zero_counter++;}
  elseif (route_load
    + cust_dem[nodes_by_demand[nodes_counter]]
    <= vehicle_capacity){
    base_sol[i]=nodes_by_demand[nodes_counter];
    nodes_counter++;
    route_load= route_load +
      cust_dem [nodes_by_dem[nodes_counter]]
}}}
while (base_sol has nodes beyond last vehicle){
Move (node_beyond_last_vehicle, available_vehicle)
if unable to move nodes beyond last vehicle to any vehicle:
PerformSwapNeighborhood(base_sol)
Move (node_beyond_last_vehicle, available_vehicle)
initial_population = [base_sol] }
Phase 2: Initialize the initial population with the base solution
While (initial population size < population_size){
new_sol = PerformRandomNeighborhood(base_sol):
  (0<prob<=0.25) uncond_route_move
  (0.25<prob<=0.5) uncond_move
  (0.5<prob<=0.75) move
  (0.75<prob<1) swap
Sequential_move
For i=0 to 12
{
Move
Swap
}
if no duplication in the initial_population and new_sol is
    feasible
  initial_population.append(new_sol)
else discard new_sol}
End
```

Figure 1. Pseudocode of a two-phase constructive approach

Customer	**Demand**		**Customer**	**Demand**
2	19		20	24
3	21		25	24
-	-	**After order by demands**	-	-
-	-		-	-
-	-		-	-
30	2	$\longrightarrow$	27	2
31	14		30	2
32	9		19	1

Figure 2. Customer ordering according to demands

0	**19**	24	..	0	2	..	0	...	0	10	..	0
Vehicle 1				Vehicle 2				...	Vehicle n (last)			

Figure 3. An example of a base solution

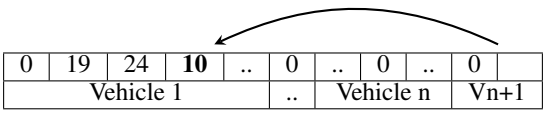

Figure 4. Reallocating beyond the last vehicle customers

that additional neighborhood moves may disrupt promising solution structures generated during the constructive phase. Consequently, increasing the maximum iteration does not necessarily lead to better initial solutions.

The maximum iteration here is 12 which is enough to show improvement for all instances since initial experiments showed that some instances with maximum iteration below 12 don't show satisfied improvement and if the iteration is set more than 12 the improvement is not guaranteed while consuming more time. Moreover, a sequential move is applied to increase solution quality in such a way that each non-zero node (customer) in solution vector is checked if it is moved to another location, from the beginning to the end, the distance will be reduced and not violating any constraint. The difference between move and sequential move is that, in move each non-zero node is checked if it is moved to a random location will improve the quality, whereas in sequential move all locations from location 0 to location *solution_size−1* in solution vector is checked to be the destination for move if the total distance can be reduced and the constraints will not be violated.

The mathematical definitions of the neighborhood operators employed in Phase 2 are as follows:

$$\textbf{Move:} \quad R'_b = R_b \cup \{i\}, \qquad R'_a = R_a \setminus \{i\}, \tag{3}$$

$$\textbf{Sequential Move:} \quad S = (i, i+1, \ldots, i+m),$$
$$R'_b = R_b \cup S, \qquad R'_a = R_a \setminus S, \tag{4}$$

$$\textbf{Unconditional Move:} \quad R'_b = R_b \cup X, \qquad R'_a = R_a \setminus X, \tag{5}$$

$$\textbf{Swap:} \quad R'_a = (R_a \setminus \{i\}) \cup \{j\}, \qquad R'_b = (R_b \setminus \{j\}) \cup \{i\}, \tag{6}$$

where X denotes either a single customer or a consecutive sequence of customers, and all resulting routes must satisfy the vehicle capacity constraint.

Assuming there are five routes for instance, a random route is chosen, let say the first route then it is moved to the third route. In unconditional move, a node is selected randomly to change its location in solution vector to a random location regardless of moving in the same route or to another route and even if this move will increase solution cost, but the new solution should be feasible. Move neighborhood structure is applied by randomly choosing a node to be moved to another location whether in the same route or to another route under the condition of improving solution cost or at least equal cost and keeping solution feasibility. A swap between a randomly selected pair of nodes is performed while considering getting better or equal solution cost without violating solution capacity and

Table 2. Parameter sensitivity analysis for selecting the maximum iteration value

Instance		Iter = 8	Iter = 10	Iter = 12	Iter = 14	Iter = 16
A-n32-k5	t_D	1071.60	1030.00	**913.50**	955.50	933.20
	t	16.51	16.69	**16.60**	16.70	17.10
A-n54-k7	t_D	1787.20	1811.30	**1324.90**	1658.00	1611.30
	t	19.10	20.70	**20.50**	21.10	23.10
A-n80-k10	t_D	2975.30	2744.50	**2080.70**	2662.00	2510.00
	t	19.10	19.50	**19.60**	20.90	21.10
B-n34-k5	t_D	998.80	948.20	915.20	924.30	**872.80**
	t	16.50	16.20	16.40	17.80	18.66
B-n57-k7	t_D	2268.40	2175.50	**1335.80**	2031.10	1950.80
	t	23.70	23.60	**24.20**	24.60	25.30
B-n78-k10	t_D	2336.90	2184.90	**1339.70**	1962.30	1939.60
	t	22.62	22.60	**22.80**	23.60	23.90
P-n22-k8	t_D	677.90	675.00	**630.70**	670.70	666.70
	t	16.90	17.20	**17.30**	17.90	17.30
P-n60-k10	t_D	1336.70	1269.40	**845.60**	1160.10	1130.20
	t	25.30	23.50	**25.00**	24.40	28.70
P-n101-k4	t_D	1582.30	1481.30	**845.20**	1343.40	1271.00
	t	27.80	28.40	**28.50**	29.80	30.70
E-n22-k4	t_D	478.30	466.60	**419.00**	477.50	452.70
	t	20.20	20.30	**23.60**	18.90	18.90
E-n51-k5	t_D	1022.90	972.20	**679.80**	946.20	869.40
	t	18.10	18.20	**18.60**	18.90	18.60
E-n101-k14	t_D	2222.10	2100.60	**1422.40**	1898.90	1880.90
	t	25.30	25.20	**25.70**	26.60	27.50

regardless of nodes locations. These structures will be repeated until satisfying the size of initial population. In case of solutions duplication, the same neighborhood structures mentioned previously in generating the population will be performed to get a duplication free population. If the generated solution is infeasible, then it will be discarded and the procedure of phase 2 is repeated to generate a new feasible solution. Duplicate detection was performed by comparing each newly generated solution vector against every solution already present in the population, position by position; the comparison terminates as soon as a single differing node is found, at which point the new solution is confirmed unique and continues to complete comparison with the remaining solutions. If every position matches an existing solution, the new solution is discarded as a duplicate and Phase 2 is repeated to generate a replacement. Because non-duplicate solutions typically differ within the first few positions (a consequence of the randomized neighborhood operators applied in Phase 2), the early-exit comparison terminates well before reaching the full solution length in practice, keeping the duplicate-check overhead a small fraction of total construction time; this is reflected in the construction times already reported in Tables 4–7, which include this check.

The datasets employed in this study are some most popular ones obtained from VRP web site[†] which provides datasets and information regarding best known solutions for those instances. The data instances used in this study are dataset A, B, P and E of CVRP. Three parts consisting of instance name which begins with a letter to represent dataset, middle part starts with letter *n* to represent number of dimensions including number of customers and the depot.

Third part starts with letter *k* to represent number of vehicles. For instance, instance A-n33-k5 belongs to dataset A with 33 dimensions (32 customers) and 5 vehicles. Extension of instance file is (*.vrp) and its contents are instance name, comment (contains dataset author, number of vehicles, optimal value), instance type which represents the

[†]http://vrp.atd-lab.inf.puc-rio.br/index.php/en

VRP variant, number of dimensions (customers), *edge_weight_type* represents the type of calculating the distance between nodes, capacity of vehicles, *node_coord_section* gives the coordinations of customers locations, demand part gives the demand for each customer, in *depot_section* a list of depots is given such that the number with no sign is the node to launch vehicles from, whilst the negative number is the node where the vehicles return to. In dataset A, the demands and coordinates of customers are selected randomly while the locations of customers in dataset B are clustered. The capacity per route is varies in dataset P. In set E, the customers are distributed randomly from a uniform spread of customers over their area.Table 3 summarizes the characteristics of the benchmark datasets used in this study.

Table 3. Characteristics of the CVRP benchmark datasets

Dataset	Number of instances	Customers range	Vehicles range	Capacity range
A	27	31–79	5–10	100
B	23	30–77	5–10	100
P	24	15–100	2–15	35–280
E	13	12–100	3–4	100–8000

4. RESULTS AND DISCUSSION

The approach was programmed with C++ and run on a computer with Core i5-6300U 2.4 GHz processor. The experiments were conducted with 10 runs for each CVRP instance. Both random approach and two-phase constructive approach (TPC) based on customer demand were tested on the CVRP instances. The random approach was tested only on dataset A to test on solution feasibility, while the TPC was tested on dataset A, B, P and E of CVRP. In this study, the maximum running time is set as 600 seconds for both approaches, else it will be considered as unable to obtain feasible solution. The 600-second limit was set based on preliminary timing tests showing that even the largest instances (e.g., E-n101-k14, P-n101-k4) completed construction in under 30 seconds on average; a 20-fold margin was adopted to accommodate variance across runs while still enforcing a practical bound to detect instances where feasible-space search stops.

Tables 4, 5, 6, 7 and 8 show the average (Avg), standard deviation (SD), minimum (Min), and optimal cost (OPT) for solution cost in addition to average run time (T_{Avg}), standard deviation run time (T_{SD}), and minimum run time (T_{Min}) measured in seconds. The result presented in Table 4 shows that the random construction approach could not generate feasible solutions within the specified running time for 4 instances of dataset A (A-n45-k6, A-n61-k9, A-n63-k9 and A-n65-k9), marked by a dash ($-$).

This infeasibility is mostly due to the high demands and/or less available vehicles since it is difficult to find a space on vehicles for the unassigned customers with high demands. Oppositely, from Tables 4, 5, 6, 7 and 8, the two-phase construction approach able to produce feasible solution population for all instances in reasonable time. When compared with random approach tested on dataset A presented in Table 3, the two-phase construction approach able to produce feasible solution in shorter time. Regarding solution quality, the two-stage construction approach performed better than the random construction approach for 22 out of 27 instances for dataset A. The experimental baseline in this study serves two distinct objectives within our evaluation framework. First, Random Construction is included as a baseline to represent standard unguided initial population generation in population-based metaheuristics (such as Genetic Algorithms and nature-inspired algorithms). Demonstrating a statistically significant improvement over random construction verifies that TPC successfully adds structured, high-quality spatial and capacity awareness into the initial search space. Second, to evaluate TPC against established constructive routing methods. This two-tiered setup enabled us to benchmark TPC both as an efficient population-initialization mechanism and as an effective standalone constructive heuristic.

A gap between the cost of each instance solution obtained by two-phase construction and the optimal solution is also obtained and calculated as follows:

$$\text{Gap}(\%) = \frac{cost_i - opt_i}{opt_i} \cdot 100\% \tag{7}$$

Table 4. Solution cost and time obtained by the random approach on dataset A

Instance	*Avg*	*SD*	*Min*	*Opt*	T_{Avg}	T_{SD}	T_{Min}
A-n32-k5	2075.3	105.76	1937	784	26.3	14.21	10
A-n33-k5	1865.4	117.04	1621	661	27.4	10.53	17
A-n33-k6	1714.1	80.01	1562	742	42.8	16.98	25
A-n34-k5	1862.3	104.08	1722	778	47.8	17.83	20
A-n36-k5	2006.6	134.18	1792	799	60.8	1.47	58
A-n37-k5	1900.6	91.25	1732	669	33.6	9.17	29
A-n37-k6	2225.4	92.19	2108	949	186.2	26.75	138
A-n38-k5	2013.1	78.60	1918	730	122.7	17.68	93
A-n39-k5	2044.4	76.29	1841	822	108.4	35.47	63
A-n39-k6	2349.3	136.74	2122	831	56.9	14.19	37
A-n44-k6	2374.3	80.19	2298	937	133.2	18.94	82
A-n45-k6	-	-	0	944	-	-	-
A-n45-k7	2633.5	157.13	2410	1146	51	6.24	42
A-n46-k7	2550	139.65	2369	914	53.2	3.89	44
A-n48-k7	2959.9	138.27	2766	1073	52	5.23	43
A-n53-k7	3047.6	142.99	2847	1010	171.6	39.96	126
A-n54-k7	3125.7	109.29	2992	1167	244.2	44.18	197
A-n55-k9	3069.4	130.91	2879	1073	132.1	29.36	81
A-n60-k9	3639.3	204.58	3393	1354	87.3	31.73	53
A-n61-k9	-	-	0	1034	-	-	-
A-n62-k8	3815.8	223.63	3406	1288	49.2	4.17	40
A-n63-k9	-	-	0	1616	-	-	-
A-n63-k10	3500.3	139.17	3287	1314	147.4	18.61	111
A-n64-k9	3661.6	115.66	3462	1401	156	18.42	121
A-n65-k9	-	-	0	1174	-	-	-
A-n69-k9	3803.8	155.54	3534	1159	120.3	15.11	103
A-n80-k10	4915.2	226.82	4484	1763	163.2	17.70	136

Where $cost_i$ is the solution cost of instance i and opt_i is the optimal cost of instance i. Then the average gap for all instances' solutions generated by two-phase construction of each data set under concern in this study is calculated to be compared with average gap of solutions obtained by other constructive approaches for the same data set. A smaller percentage gap indicates the superiority of the proposed algorithm.

Overall, construction time for the two-phase approach presented in Tables 4, 5, 6, 7 and 8 is reasonable for all instances while the solution cost is better than the random construction. However, the two-phase approach is still far from the optimal cost for 86 out of 87 data instances with gaps averaging 11.2% on dataset A, 15.6% on dataset B, 12.16% on dataset P, and 13.6% on dataset E and thus, solution cost needs to be improved later in improvement phase.

Moreover, the proposed approach successfully generated a population of 100 initial solutions for almost all datasets instances under concern in this study except of four instances that are: E-n13-k4 which in 3 out of 10 runs TPC generated 80 feasible solutions within time limit, P-n19-k2 30 feasible solutions generated through time limit, P-n20-k2 the proposed method generated 60 solutions, and for P-n22-k2 in 2 out of 10 runs the population size is 80 within time limit. If the execution time is allowed for longer than time limit, i.e. 600 seconds, the proposed method can generate feasible solutions with size of 100 solutions. The reason behind being not able to generate 100 solutions for the mentioned instances is due to the small size vector which reduces number of combinations and the toughness of committing constraints. Table 9 shows the percentage of successful constructed feasible solution for

dataset A, B, P and E of CVRP using TPC compared with other construction methods that applied in [28] which are sequential insertion (SI), parallel insertion (PI), nearest neighbour construction (NN), Clark and Wright construction (CWC), variant of CWC (VCWC), and sweep construction (SC). It can be seen that the two-phase approach has successfully generates feasible solutions for all data instances and competitive with other construction methods that applied in [28]. As shown in Table 9, only method SC is unable to generate feasible solutions for all datasets since it requires more vehicles to accommodate all customers. On the other hand, all other methods can generate feasible solutions for all datasets. Nevertheless, TPC provides a better trade-off between speed, feasibility, and solution quality compared to existing constructive heuristics. As shown in Table 10, TPC achieved lower average gaps than the re-implemented CWC (CWC2026) on all four datasets (11.2% vs. 20.9% on A, 15.6% vs. 24.47% on B, 12.16% vs. 18.95% on P, and 13.6% vs. 36.75% on E). Unlike the random construction approach, TPC also produced a feasible solution population for all 87 instances within a short and consistent construction time. It should also be noted that constructive heuristics that apply a local search or improvement phase, such as the fuzzy c-means clustering approach of [19], aim to produce a single refined solution per instance, whereas TPC is designed to generate a diverse population of feasible initial solutions without any improvement phase, for the purpose of initializing population-based optimization algorithms. Therefore, TPC does not only provide fast and feasible initial solutions, but also achieves a trade-off between construction speed, solution feasibility, diverse population, and solution quality among constructive heuristics evaluated under identical conditions, which makes it suitable to be used either as a standalone constructive heuristic or as a population initialization method for population-based approaches.

Although classic deterministic heuristics such as CWC achieve lower solution gaps on certain benchmark sets (e.g., sets P and E), TPC offers distinct operational advantages within the context of population-based optimization. First, CWC is a deterministic, single-solution heuristic that constructs only one trajectory. In contrast, TPC is designed to construct a diverse population of feasible initial solutions through its two-phase design, making it directly suitable for population-based metaheuristics (e.g., Genetic Algorithms, Evolutionary Algorithms). Second, TPC demonstrates significantly lower computational overhead compared to CWC (validated by the Wilcoxon signed-rank test on runtime, $p = 0.0005$). Therefore, a researcher would select TPC over CWC when initializing population-based metaheuristics where rapid construction of diverse, guaranteed-feasible initial populations is required.

Table 10 and Figure 5 provide the average gap (%) and the ranking of the proposed TPC compared with well-established constructive heuristics reported in the comparative study of Avdoshin and Beresneva [28], where all constructive heuristics were evaluated under a common experimental framework using the same benchmark datasets. For datasets A and B, the proposed TPC achieved lower average gaps than the Sequential Insertion (SI), Parallel Insertion (PI), Nearest Neighbor (NN), and Sweep Construction (SC) methods. For datasets P and E, the Clarke and Wright Savings (CWC) algorithm achieved lower average gaps than the proposed TPC. It should be noted that the CWC algorithm was primarily designed to construct a single high-quality solution, whereas the proposed TPC is designed to generate a population of feasible initial solutions for population-based optimization algorithms. Therefore, the proposed TPC and CWC have different objectives and should be evaluated according to their intended applications. Although the proposed TPC did not achieve the lowest average gaps for all datasets, it provides an effective constructive approach for generating feasible initial solutions that can be used in the initialization stage of population-based optimization algorithms. Moreover, Table 10 and Figure 5 show the average gap (%) of CWC2026 and FCM-based constructive heuristic. The min costs of instances considered by [19] are shown in Table 13.

Table 11 compares the minimum costs of dataset A obtained by two-phase construction method (TPC) proposed in this paper with the minimum costs of Savings Series (SS), and Savings Parallel (SP) from [14]. The results show that, the TPC is the best quality for 7 out of 27 instances highlighted with bold and italic and outperformed SS in 23 instances while generating better results compared to SP in 8 instances and equal in 1 (with bold). Table 12 compares the cost minimum values (Min) and execution time in seconds (T) achieved by TPC against CWC and CIH (FR) from [27]. The instances covered in Table 11 are the common ones that have been experimented by TPC and [27] since [27] tested two other instances from a data set not considered by TPC. The quality of solutions achieved by TPC is the best for instance A-n33-k5 while TPC is better than FR for dataset A, dataset P, and dataset E

Table 5. Solution cost and time obtained by the two-phase approach on dataset A

Instance	**Avg**	**SD**	**Min**	**Opt**	**Gap (%)**	T_{Avg}	T_{SD}	T_{Min}
A-n32-k5	913.50	49.00	840	784	7.14	16.6	3.10	11
A-n33-k5	730.00	29.98	683	661	3.33	15.7	4.88	11
A-n33-k6	820.60	44.20	758	742	2.16	23.3	1.34	21
A-n34-k5	887.80	39.81	810	778	4.11	21.0	2.21	18
A-n36-k5	900.50	36.56	863	799	8.01	15.0	1.63	13
A-n37-k5	770.80	28.53	705	669	5.38	22.8	2.62	20
A-n37-k6	1060.00	41.34	974	949	2.63	18.0	2.26	15
A-n38-k5	864.70	28.91	819	730	12.19	26.6	5.95	18
A-n39-k5	965.70	44.82	868	822	5.60	20.8	4.54	16
A-n39-k6	959.00	42.49	842	831	1.32	19.3	2.91	15
A-n44-k6	1100.80	39.02	1024	937	9.28	21.6	1.84	19
A-n45-k6	1084.50	45.42	1183	944	25.32	24.3	5.14	18
A-n45-k7	1319.40	47.96	1186	1146	3.49	23.9	1.73	21
A-n46-k7	1045.20	42.48	990	914	8.32	25.1	3.03	21
A-n48-k7	1243.60	42.75	1217	1073	13.42	19.7	1.25	18
A-n53-k7	1170.80	50.15	1252	1010	23.96	23.3	2.63	19
A-n54-k7	1324.90	44.97	1354	1167	16.02	20.5	2.64	17
A-n55-k9	1222.10	34.62	1305	1073	21.62	19.2	1.93	17
A-n60-k9	1572.70	75.92	1475	1354	8.94	19.0	2.40	16
A-n61-k9	1175.70	40.16	1281	1034	23.89	19.0	2.21	16
A-n62-k8	1461.70	58.75	1469	1288	14.05	16.4	2.12	14
A-n63-k9	1845.80	67.57	1832	1616	13.37	17.5	1.90	15
A-n63-k10	1528.90	51.56	1446	1314	10.05	18.0	1.76	15
A-n64-k9	1629.40	46.42	1540	1401	9.92	21.0	2.45	18
A-n65-k9	1335.80	48.06	1455	1174	23.94	24.2	1.40	22
A-n69-k9	1339.70	39.52	1302	1159	12.34	22.8	3.19	19
A-n80-k10	2080.70	85.65	1991	1763	12.93	19.6	4.63	12

Table 6. Solution cost and time obtained by the two-phase approach on dataset B

Instance	**Avg**	**SD**	**Min**	**Opt**	**Gap (%)**	T_{Avg}	T_{SD}	T_{Min}
B-n31-k5	773.00	40.23	682	672	1.49	15.6	2.50	13
B-n34-k5	915.20	43.16	814	788	3.30	16.4	3.10	13
B-n35-k5	1060.00	41.34	1002	955	4.92	18.0	2.26	15
B-n38-k6	920.40	36.56	835	802	3.73	15.2	1.63	13
B-n39-k5	670.80	28.53	596	549	8.56	22.8	2.62	20
B-n41-k6	955.70	44.82	996	829	20.14	20.8	4.54	16
B-n43-k6	864.70	28.91	793	742	6.87	26.6	5.95	18
B-n44-k7	1084.50	45.42	995	909	9.46	24.3	5.14	18
B-n45-k5	887.80	39.81	861	751	14.65	21.0	2.21	18
B-n45-k6	820.60	44.20	843	678	24.34	23.3	1.34	21
B-n50-k7	861.50	42.10	806	741	8.77	22.0	2.15	19
B-n50-k8	1420.00	58.21	1388	1312	5.79	18.2	3.01	16
B-n51-k7	1175.70	40.16	1359	1032	31.69	19.0	2.21	16
B-n52-k7	885.50	32.14	897	747	20.08	20.2	2.45	17
B-n56-k7	840.80	31.25	838	707	18.53	18.6	1.95	16
B-n57-k7	1335.80	48.06	1623	1153	40.76	24.2	1.40	22
B-n57-k9	1845.80	67.57	1749	1598	9.45	17.5	1.90	15
B-n63-k10	1629.40	46.42	1699	1496	13.57	21.0	2.45	18
B-n64-k9	1045.20	42.48	1200	861	39.37	25.1	3.03	21
B-n66-k9	1528.90	51.56	1651	1316	25.46	18.0	1.76	15
B-n67-k10	1170.80	50.15	1279	1031	23.93	23.3	2.63	19
B-n68-k9	1572.70	75.92	1358	1272	6.76	19.0	2.40	16
B-n78-k10	1339.70	39.52	1428	1221	16.95	22.8	3.19	19

Table 7. Solution cost and time obtained by the two-phase approach on dataset P

Instance	Avg	SD	Min	Opt	Gap (%)	T_{Avg}	T_{SD}	T_{Min}
P-n16-k8	485.40	18.25	451	450	0.22	14.8	1.10	13
P-n19-k2	225.80	12.14	237	212	11.79	15.2	1.35	14
P-n20-k2	232.50	10.56	249	216	1.39	14.9	1.20	13
P-n21-k2	231.40	14.80	212	211	0.47	15.1	1.42	13
P-n22-k2	235.60	12.33	218	216	0.93	15.5	1.63	14
P-n22-k8	630.70	32.45	603	590	2.20	17.3	2.15	15
P-n23-k8	565.40	28.16	533	529	0.76	18.0	2.40	16
P-n40-k5	515.20	22.10	469	458	2.40	21.4	2.85	18
P-n45-k5	555.80	25.42	545	510	6.86	22.3	3.10	19
P-n50-k7	610.60	29.34	598	554	7.94	21.0	2.95	17
P-n50-k8	692.50	31.18	801	631	26.94	22.6	3.05	18
P-n50-k10	740.30	34.12	765	696	9.91	23.0	3.24	19
P-n51-k10	820.40	38.56	865	741	16.73	24.2	3.80	20
P-n55-k7	635.80	27.91	608	568	7.04	21.5	2.70	18
P-n55-k8	662.70	29.15	655	588	11.39	22.8	3.12	19
P-n55-k10	765.40	35.62	766	694	10.37	24.5	3.95	20
P-n55-k15	1020.20	48.34	1168	945	23.60	26.3	4.15	22
P-n60-k10	845.60	40.25	888	744	19.35	25.0	3.84	21
P-n60-k15	1065.30	50.12	1085	968	12.09	27.2	4.60	22
P-n65-k10	885.70	42.18	881	792	11.24	25.6	4.10	21
P-n70-k10	920.40	44.50	1055	827	27.57	26.8	4.35	22
P-n76-k4	742.80	36.14	745	593	25.63	24.4	3.90	20
P-n76-k5	785.60	38.25	779	627	24.24	25.1	4.05	21
P-n101-k4	845.20	40.16	796	681	16.89	28.5	5.10	23

Table 8. Solution cost and time obtained by the two-phase approach on dataset E

Instance	Avg	SD	Min	Opt	Gap (%)	T_{Avg}	T_{SD}	T_{Min}
E-n13-k4	265.0	18.28	247	247	0.00	86.25	66.18	35
E-n22-k4	419.0	36.01	379	375	1.07	23.60	4.45	18
E-n23-k3	640.8	38.34	577	569	1.41	40.10	24.48	13
E-n30-k3	634.8	52.25	550	534	3.00	13.22	2.86	9
E-n31-k7	532.6	57.36	477	379	25.86	45.77	29.12	21
E-n33-k4	896.8	26.77	850	835	1.80	14.10	7.42	6
E-n51-k5	679.8	39.11	602	521	15.55	18.60	3.32	15
E-n76-k7	875.3	57.23	770	682	12.90	17.00	5.25	7
E-n76-k8	936.7	38.42	869	735	18.23	22.50	4.15	18
E-n76-k10	1209.9	79.65	1051	830	26.63	28.16	4.88	21
E-n76-k14	1340.7	43.38	1273	1021	24.68	58.20	4.66	53
E-n101-k8	1108.8	108.82	994	817	21.66	23.90	2.91	19
E-n101-k14	1422.4	59.44	1338	1071	24.93	25.70	6.57	15

in addition to instance B-n35-k5 from dataset B but with consuming more time than FR. Solutions quality generated by CWC outperformed TPC and FR. On the other hand, the shortest execution time for all datasets achieved by FR followed by TPC and CWC respectively. Thus, TPC represents a trade-off between the quality of CWC and speed of execution by FR especially for dataset A, P, and E.

Table 13 presents a comparison between the proposed TPC and the recent FCM-based constructive heuristic by [19], restricted to the benchmark instances common to both studies. Since Ewbank et al. evaluated a different subset of the standard A, B, P, and E datasets, the comparison was limited to these common instances to ensure a fair evaluation. As shown in Table 10, achieved lower average gaps than TPC across all four datasets (2.09% vs. 9.6% on A, 0.60% vs. 9.17% on B, 2.58% vs. 10.87% on P, and 2.08% vs. 5.95% on E). This outcome is expected, since applies a clustering procedure followed by a 2-opt local search to construct a single, refined solution per instance, whereas TPC is a constructive heuristic, without improvement phase, designed to generate a diverse population

of feasible initial solutions for initializing population-based optimization algorithms. Although both approaches fall under the category of constructive heuristics for the CVRP, they pursue different objectives, and their relative performance should therefore be interpreted based on their intended use rather than compared on solution quality alone.

Table 9. Percentage of Successful Solution Construction by Methods

Data set	% of successful constructed solution						
	TPC	SI	PI	NN	CWC	VCWC	SC
A	100	100	100	100	100	100	31
B	100	100	100	100	100	100	50
P	100	100	100	100	100	100	50
E	100	100	100	100	100	100	56.5

Table 10. Average gaps (%) obtained by different constructive methods

Average Gap (%)		Datasets			
		A	B	P	E
Methods	TPC	11.21 (2)	15.6 (3)	12.16 (2)	13.6 (2)
	SI	68.7	82.3	66.0	70.4
	PI	33.2	34.0	25.6	30.0
	NN	39.7	41.2	32.2	41.5
	CWC	5.0 (1)	4.3 (1)	6.9 (1)	6.4 (1)
	CWC2026	20.9	24.47	18.95	36.75
	FCM	2.09	0.60	2.58	2.08
	VCWC	12.7 (3)	9.8 (2)	11.3 (3)	17.5 (3)
	SC	40.2	31.7	31.4	36.4

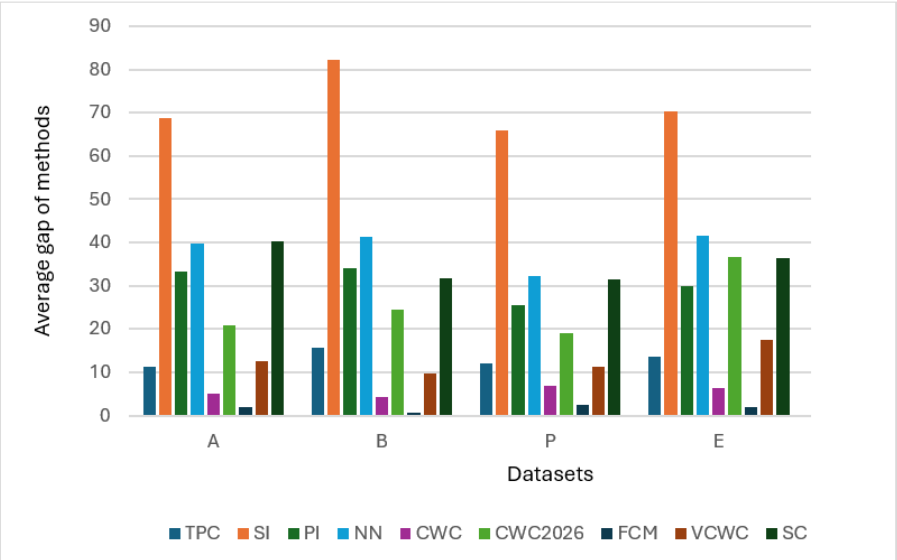

Figure 5. Average gap of methods

Table 11. Result comparison between the two-phase approach (TPC) and Saving Series (SS) and Savings Parallel (SP) by [14]

Instance	TPC	SS [14]	SP [14]
A-n32-k5	840	957	842
A-n33-k5	683	769	716
A-n33-k6	758	926	774
A-n34-k5	810	886	809
A-n36-k5	863	994	835
A-n37-k5	705	898	705
A-n37-k6	974	1061	977
A-n38-k5	819	847	770
A-n39-k5	868	1031	907
A-n39-k6	842	1024	857
A-n44-k6	1024	1155	1006
A-n45-k6	1183	1120	997
A-n45-k7	1186	1334	1198
A-n46-k7	990	1133	939
A-n48-k7	1217	1234	1110
A-n53-k7	1252	1167	1198
A-n54-k7	1354	1379	1209
A-n55-k9	1305	1277	1109
A-n60-k9	1475	1638	1408
A-n61-k9	1281	1302	1050
A-n62-k8	1469	1622	1368
A-n63-k9	1832	1904	1682
A-n63-k10	1446	1556	1352
A-n64-k9	1540	1763	1409
A-n65-k9	1455	1519	1266
A-n69-k9	1302	1312	1192
A-n80-k10	1991	1967	1840

Table 12. Result comparison between the two-phase approach (TPC) and CWC and FR by [21]

Instance	TPC		CWC		FR	
	Min	T	Min	T	Min	T
A-n33-k5	**683**	3	691	7.3	744	0.14
A-n46-k7	990	7	**939**	24.6	1083	0.16
A-n60-k9	1475	5	**1428**	70.6	1514	0.25
B-n35-k5	1002	2	**978**	10.3	1005	0.09
B-n45-k5	861	4	**758**	21.4	824	3.98
B-n68-k9	1358	4	**1322**	92.1	1349	0.30
B-n78-k10	1428	3	**1268**	165.6	1375	2.09
P-n76-k4	745	8	**655**	119	746	0.34
P-n101-k4	796	8	**766**	295.5	882	0.44
E-n30-k3	550	3	**538**	5.7	588	1.47
E-n51-k5	602	5	**588**	29.9	663	0.31
E-n76-k7	770	6	**738**	116.3	895	0.25

The Wilcoxon Signed-Rank Test was employed to compare the paired results obtained by the constructive methods because it is a non-parametric statistical test that does not require the assumption of normally distributed data. Since the distribution of solution quality produced by constructive heuristics cannot be assumed to follow a normal distribution, the Wilcoxon Signed-Rank Test provides an appropriate statistical method for evaluating the significance of the observed differences between the compared methods. Table 14 presents the statistical comparison between the proposed TPC and the Sequential Search (SS) method across $N = 27$ benchmark instances. The test yielded a positive rank sum $W+ = 325.5$ and a negative rank sum $W- = 52.5$. The Wilcoxon signed-rank test confirms a statistically significant difference in solution quality ($W = 52.5$, $Z = 3.27$, $p = 0.0011$), demonstrating that TPC achieves significantly better solution costs than SS. Similarly, Table 15 evaluates TPC against the Fisher and Jaikumar (FR) method across N = 12 instances. With $W+ = 67.0$ and $W- = 11.0$, TPC achieved statistically significantly lower solution costs than the FR method ($W = 11.0$, $p = 0.0325$). Finally, Table 16 compares the execution runtime of

Table 13. Result comparison between the proposed two-phase construction (TPC) and the FCM-based constructive heuristic proposed by Ewbank et al. [19]

Instance	TPC	FCM
A-n32-k5	840	787
A-n33-k5	683	662
A-n33-k6	758	745
A-n34-k5	810	786
A-n36-k5	863	802
A-n37-k5	705	672
A-n37-k6	**974**	1000
A-n38-k5	819	750
A-n39-k5	868	829
A-n39-k6	842	835
A-n44-k6	1024	1000
A-n46-k7	**990**	1000
A-n48-k7	1217	1083
A-n53-k7	1252	1021
A-n54-k7	1354	1188
A-n63-k10	1446	1329
A-n64-k9	1540	1438
A-n65-k9	1455	1202
A-n69-k9	1302	1179
A-n80-k10	1991	1794
B-n31-k5	682	679
B-n34-k5	814	793
B-n35-k5	1002	956
B-n38-k6	835	808
B-n39-k5	596	553
B-n41-k6	996	834
B-n43-k6	793	748
B-n44-k7	995	914
B-n50-k7	806	744
B-n52-k7	897	752
B-n63-k10	1699	1506
P-n19-k2	237	219
P-n20-k2	249	217
P-n21-k2	212	217
P-n22-k2	218	217
P-n40-k5	469	461
P-n45-k5	545	513
P-n50-k7	598	560
P-n55-k7	608	573
P-n55-k8	655	580
P-n55-k10	766	700
P-n76-k4	745	610
P-n76-k5	779	644
P-n101-k4	796	702
E-n22-k4	379	402
E-n23-k3	577	569*
E-n30-k3	550	539
E-n33-k4	850	837
E-n51-k5	602	535
E-n76-k7	770	692

TPC against the Clarke and Wright Savings (CWC) algorithm ($N = 12$). The test yielded $W+ = 78.0$ and $W- = 0.0$, indicating that the execution time of the proposed TPC is statistically significantly lower than that of CWC ($W = 0.0$, $p = 0.0005$). To assess whether performance differences across benchmark instances were statistically significant, we applied the non-parametric Wilcoxon signed-rank test. Although parametric alternatives such as the paired t-test are viable when data follows a normal distribution, performance metrics across heterogeneous Capacitated Vehicle Routing Problem (CVRP) benchmark sets, specifically percentage solution gaps and execution times, frequently exhibit right-skewness, heteroscedasticity, and scale variations across different instance sizes ($n =$

33 to $n = 101$). Following standard statistical guidelines in metaheuristic and evolutionary computation literature [29], non-parametric testing is preferred for pairwise instance comparisons because it relaxes the strict assumptions of normality and equal variance, remaining robust against potential performance outliers across diverse problem topologies. To address the practical applicability of TPC beyond the construction phase, TPC has been employed as the population initialization method for a population-based metaheuristic in a subsequent study [30]. In that study, an enhanced Red Deer Algorithm with an adaptive memory strategy (RDAM) was initialized using TPC-generated populations and evaluated on the same 87 CVRPLIB benchmark instances (datasets A, B, P, and E) used in this paper. RDAM outperformed the TPC-constructed solutions by 96.5% and the original RDA by 87.3%, achieving four optimal and one better-than-optimal solution, and produced competitive results, with and in several cases superior to, eight established metaheuristics from the literature, including ant colony optimization, a modified ant system, genetic algorithm variants, a cooperative firefly algorithm, and a multiple-colony artificial bee colony algorithm. These findings confirm that TPC-generated populations can be effectively refined by population-based metaheuristics to reach high-quality, near-optimal solutions, demonstrating that TPC's practical value as a population initializer is not speculative.

Table 14. Wilcoxon Signed Rank Test between SS and TPC

Instance	SS	TPC	SS - TPC	R	Signed_R
A-n32-k5	957	**840**	117	18	18
A-n33-k5	769	**683**	86	13	13
A-n33-k6	926	**758**	168	21	21
A-n34-k5	886	**810**	76	11	11
A-n36-k5	994	**863**	131	19	19
A-n37-k5	898	**705**	193	24	24
A-n37-k6	1061	**974**	87	14	14
A-n38-k5	847	**819**	28	6	6
A-n39-k5	1031	**868**	163	20	20
A-n39-k6	1024	**842**	182	22	22
A-n44-k6	1155	**1024**	131	19	19
A-n45-k6	**1120**	1183	-63	10	-10
A-n45-k7	1334	**1186**	148	20	20
A-n46-k7	1133	**990**	143	19	19
A-n48-k7	1234	**1217**	17	3	3
A-n53-k7	**1167**	1252	-85	12	-12
A-n54-k7	1379	**1354**	25	5	5
A-n55-k9	**1277**	1305	-28	6	-6
A-n60-k9	1638	**1475**	163	21	21
A-n61-k9	1302	**1281**	21	4	4
A-n62-k8	1622	**1469**	153	21	21
A-n63-k9	1904	**1832**	72	9	9
A-n63-k10	1556	**1446**	110	17	17
A-n64-k9	1763	**1540**	223	27	27
A-n65-k9	1519	**1455**	64	10	10
A-n69-k9	1312	**1302**	10	1	1
A-n80-k10	**1967**	1991	-24	4	-4

Table 15. Wilcoxon Signed Rank Test between Costs of FR and TPC

Instance	FR	TPC	FR - TPC	R	Signed_R
A-n33-k5	744	683	61	1	1
A-n46-k7	1083	990	93	2	2
A-n60-k9	1514	1475	39	3	3
B-n35-k5	1005	1002	3	4	4
B-n45-k5	824	861	-37	5	-5
B-n68-k9	1349	1358	-9	6	-6
B-n78-k10	1375	1428	-53	7	-7
P-n76-k4	746	745	1	8.5	8.5
P-n101-k4	882	796	86	8.5	8.5
E-n30-k3	588	550	38	10	10
E-n51-k5	663	602	61	11	11
E-n76-k7	895	770	125	12	12

Table 16. Wilcoxon Signed Rank Test between Run Time of CWC and TPC

Instance	CWC	TPC	CWC - TPC	R	Signed_R
A-n33-k5	7.3	3	4.3	1	1
A-n46-k7	24.6	7	17.6	2	2
A-n60-k9	70.6	5	65.6	3	3
B-n35-k5	10.3	2	8.3	4	4
B-n45-k5	21.4	4	17.4	5	5
B-n68-k9	92.1	4	88.1	6	6
B-n78-k10	165.6	3	162.6	7	7
P-n76-k4	119.0	8	111.0	8	8
P-n101-k4	295.5	8	287.5	9	9
E-n30-k3	5.7	3	2.7	10	10
E-n51-k5	29.9	5	24.9	11	11
E-n76-k7	116.3	6	110.3	12	12

5. CONCLUSION

This study proposed a two-phase constructive approach (TPC) that orders customers according to their demands into a base initial solution. The population of solutions is then generated according to the base initial solution by applying a set of neighborhood structures to produce various solutions. The datasets under concern in this study are the CVRP benchmark datasets of A, B, P, and E. The results revealed that the random construction approach was unable to generate feasible solutions for all instances of dataset A within the time limit. Meanwhile, the proposed TPC demonstrates the ability to efficiently generate feasible initial solutions while providing acceptable solution quality for initializing population-based optimization algorithms for all dataset instances of A, B, P, and E. Comparison between the two approaches also shows that the quality of 22 out of 27 solutions by the two-phase approach outperformed the quality of random construction for dataset A. The two-phase approach is also comparable with other constructive approaches proposed in the literature in terms of successful generation of feasible solutions and reasonable quality of solutions. The quality of results generated by the TPC shows reasonable performance compared to other approaches in the literature that produce feasible solutions to be improved later by population-based methods, except for CWC which generates better quality feasible solutions but with longer execution time. Cost quality for TPC is validated against inferior methods by employing the Wilcoxon rank test to ensure the significance of TPC, in addition to validating run time for TPC with CWC. For future work, adaptive or learning-based strategies have the potential to be explored to adjust the construction process dynamically based on the characteristics of the problem instance. Moreover, an efficient population-based metaheuristic method should also be considered for solution improvement.

ACKNOWLEDGEMENTS

The authors would like to express their gratitude to the Ministry of Higher Education (MoHE) Malaysia for supporting this study under the Fundamental Research Grant Scheme (FRGS/1/2021/ICT02/UUM/02/5), as well as RIMC, Universiti Utara Malaysia, for the study's administration.

REFERENCES

1. Ismail, S. N., K. R. Ku-Mahamud, and S. Abdul-Rahman, A review on delivery routing problem and its approaches. *Journal of Theoretical and Applied Information Technology*, vol. 95, no. 2, pp. 367–380, 2017.
2. Akpinar, S., Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem. *Expert Systems with Applications*, vol. 61, pp. 28–38, 2016. doi: 10.1016/j.eswa.2016.05.023.
3. Elshaer, R. and H. Awad, A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Computers and Industrial Engineering*, vol. 140, 2020. doi: 10.1016/j.cie.2019.106242.
4. Hassan, M. H., et al., A Hybrid Bio-Inspired Optimization Scheme for RSU Distribution in Vehicular Ad-Hoc Network. *Int. J. Emerg. Technol. Adv. Eng.*, vol. 13, no. 3, pp. 152–158, 2023.
5. Yahia, E. H., et al., Optimized Selective Harmonic Elimination in CHB-MLI Using Red-Tailed Hawk Algorithm for Unequal DC Sources. *Int. J. Robot. Control Syst.*, vol. 5, no. 2, pp. 830–845, 2025.
6. Alsarraj, M. F., Y. E. Ahmed, S. B. Ezzat, and J. Pasupuleti, Modeling and Optimization of Solar Panel Cooling using Machine Learning and Deep Learning Techniques. *NTU J. Renew. Energy*, vol. 10, no. 1, pp. 1–10, 2026.
7. G. Laporte, Nobert Yves, and M. Desrochers, "OPTIMAL ROUTING UNDER CAPACITY AND DISTANCE RESTRICTIONS.," *Oper. Res.*, vol. 33, no. 5, pp. 1050–1073, Oct. 1985, doi: 10.1287/opre.33.5.1050.
8. G. Laporte and F. Semet, "5. Classical Heuristics for the Capacitated VRP," *in The Vehicle Routing Problem, Society for Industrial and Applied Mathematics*, 2002, pp. 109–128.
9. T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," IEEE Transactions on Information Theory, vol. 13, no. 1. pp. 21–27, 1967, doi: 10.1109/TIT.1967.1053964.
10. G. Clarke and J. W. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Oper. Res.*, vol. 12, no. 4, pp. 568–581, Aug. 1964, doi: 10.1287/opre.12.4.568.
11. B. E. Gillett and L. R. Miller, "A Heuristic Algorithm for the Vehicle-Dispatch Problem," *Oper. Res.*, vol. 22, no. 2, pp. 340–349, Apr. 1974, doi: 10.1287/opre.22.2.340.
12. P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi, "Computational results with a branch-and-cut code for the capacitated vehicle routing problem," *Inst. Syst. Comput. Anal.*, vol. 495, no. 1, pp. 1–22, 1998.
13. N. Christofides and S. Eilon, "An Algorithm for the Vehicle-dispatching Problem," *Journal of the Operational Research Society*, vol. 20, no. 3. pp. 309–318, 1969, doi: 10.1057/jors.1969.75.
14. M. A. H. Akhand, T. Sultatana, M. I. R. Shuvo, and A.-M. Al-Mahmud, "Constructive and Clustering Methods to Solve Capacitated Vehicle Routing Problem," *Oriental journal of computer science and technology*, vol. 10, no. 3. pp. 549–562, 2017, doi: 10.13005/ojcst/10.03.02.
15. M. Gmira, M. Gendreau, A. Lodi, and J.-Y. Potvin, "Tabu search for the time-dependent vehicle routing problem with time windows on a road network," *Eur. J. Oper. Res.*, vol. 288, no. 1, pp. 129–140, Jan. 2021, doi: 10.1016/j.ejor.2020.05.041.
16. G. Niranjani and K. Umamaheswari, "Sustainable Waste Collection Vehicle Routing Problem for COVID-19," *Intelligent Automation and Soft Computing*, vol. 33, no. 1. pp. 457–472, 2022, doi: 10.32604/iasc.2022.024264.
17. "Capacitated Vehicle Routing Problem for Thailand's Steel Industry via Saving Algorithms," *J. Syst. Manag. Sci.*, Jun. 2021, doi: 10.33168/JSMS.2021.0211.
18. J. Joubert and S. Claasen, "A sequential insertion heuristic for the initial solution to a constrained vehicle routing problem," *ORiON*, vol. 22, no. 1, Jun. 2006, doi: 10.5784/22-1-36.
19. H. Ewbank, R. Wanke, R. H. C. Takano, and A. C. P. L. F. de Carvalho, "The capacitated vehicle routing problem revisited: Using fuzzy c-means clustering," *Expert Systems with Applications*, vol. 152, Art. no. 113351, 2020, doi: 10.1016/j.eswa.2020.113351.
20. N. A. Mat, A. M. Benjamin, and S. Abdul-Rahman, "Efficiency of Heuristic algorithms in solving waste collection vehicle routing problem: A case study," *Journal of Social Sciences Research*, vol. 2018, no. Special Issue 6. pp. 695–700, 2018, doi: 10.32861/jssr.spi6.695.700.
21. M. L. Fisher and R. Jaikumar, "A generalized assignment heuristic for vehicle routing," *Networks*, vol. 11, no. 2. pp. 109–124, 1981, doi: 10.1002/net.3230110205.
22. W. Chowmali and S. Sukto, "A hybrid FJA-ALNS algorithm for solving the multi-compartment vehicle routing problem with a heterogeneous fleet of vehicles for the fuel delivery problem," *Decis. Sci. Lett.*, vol. 10, no. 4, pp. 497–510, 2021, doi: 10.5267/j.dsl.2021.6.001.
23. S. R. Ait Haddadene, N. Labadie, and C. Prodhon, "A GRASP × ILS for the vehicle routing problem with time windows, synchronization and precedence constraints," *Expert Syst. Appl.*, vol. 66, pp. 274–294, Dec. 2016, doi: 10.1016/j.eswa.2016.09.002.
24. H. Yahyaoui, I. Kaabachi, S. Krichen, and A. Dekdouk, "Two metaheuristic approaches for solving the multi-compartment vehicle routing problem," *Operational Research*, vol. 20, no. 4. pp. 2085–2108, 2020, doi: 10.1007/s12351-018-0403-4.
25. T. T. Doan, N. Bostel, and M. H. Hà, "The vehicle routing problem with relaxed priority rules," *EURO J. Transp. Logist.*, vol. 10, p. 100039, 2021, doi: 10.1016/j.ejtl.2021.100039.
26. MARIUS M. SOLOMON, "Solomon-AlgorithmsVehicleRouting-1987.pdf."
27. M. Alssager, Z. A. Othman, and M. Ayob, "Cheapest Insertion Constructive Heuristic based on Two Combination Seed Customer Criterion for the Capacitated Vehicle Routing Problem," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 7, no. 1, p. 207, Feb. 2017, doi:

10.18517/ijaseit.7.1.1792.
28. S. M. Avdoshin and E. N. Beresneva, "Constructive heuristics for Capacitated Vehicle Routing Problem: a comparative study," *Proc. Inst. Syst. Program. RAS*, vol. 31, no. 3, pp. 145–156, 2019, doi: 10.15514/ISPRAS-2019-31(3)-12.
29. J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011, doi: 10.1016/j.swevo.2011.02.002.
30. N. Nabeel, S. Abdul-Rahman, and J. Wahid, "An Enhanced Red Deer Algorithm with Adaptive Memory Strategy for the Capacitated Vehicle Routing Problem," *J. Inf. Commun. Technol.*, vol. 25, no. 2, pp. 70–95, 2026.