Heliyon 7 (2021) e08029

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

Hybrid modified ant system with sweep algorithm and path relinking for the capacitated vehicle routing problem

Arit Thammano *, Petcharat Rungwachira

Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand

ARTICLE INFO

Keywords:
Vehicle Routing Problem (VRP)
Capacitated Vehicle Routing Problem (CVRP)
Sweep Algorithm (SA)
Modified Ant System (MAS)
Path Relinking (PR)

ABSTRACT

Vehicle routing problem is a widely researched combinatorial optimization problem. We developed a hybrid of three strategies—a modified ant system, a sweep algorithm, and a path relinking—for solving a capacitated vehicle routing optimization problem, a vehicle routing problem with a capacity constraint. A sweep algorithm was used to generate initial solutions and assign customers to vehicles, followed by a modified ant system to generate new generations of good solutions. Path relinking was used for building a better solution (candidate) from a pair of guiding and initial solutions. Finally, a local search method—swap, insert, reverse and greedy search operations—was used to prevent solutions from getting trapped in a local minimum. Performance of the proposed algorithm was evaluated on three datasets from CVRPLIB. Our proposed method was at least competitive to state-of-the-art algorithms in terms of the total route lengths. It even surpassed the best-known solution in the P-n55-k8 instance. Our findings can lower the transportation cost by reducing the travelling distance and efficiently utilizing the vehicle capacity.

1. Introduction

Currently, online purchasing is a popular choice for many people around the world because it offers many kinds of products without having to spend time traveling to real stores. During the Covid-19 pandemic, online purchasing has been even more popular. Optimal vehicle routing is an important logistic factor that helps make online purchasing successful and efficient, leading to acceptance of online purchasing by former and new customers. In addition, optimal vehicle routing helps reduce oil (a limited energy resource) consumption and air pollution from vehicle exhausts.

The Vehicle Routing Problem (VRP) is a combinatorial optimization problem; it aims to find an optimal route with the lowest cost for delivering goods to specified customers in various locations. Since a vehicle has a limited load capacity, a VRP solution needs to consider the load capacity as a constraint. A Capacitated Vehicle Routing Problem (CVRP), a similar problem, finds sets of routes to be traversed by a fleet of vehicles. The vehicles start from a depot, deliver goods to customers, and then return to the depot. A customer is served by only one vehicle. Each vehicle can serve several customers when their demands do not exceed the vehicle capacity [1].

Optimization algorithms can be classified into algorithms that give an exact or an approximate solution. An exact algorithm is guaranteed to find the best solution. However, generally it requires a large amount of computational time to get the solution. Since CVRP is an NP-hard problem, in which finding the optimal solution is very hard and time-consuming, algorithms that provide an approximate solution are more practical and commonly used to solve large CVRP problems [2]. Algorithms that provide an approximate solution are of 2 groups: approximation and meta-heuristic algorithms. Approximation algorithms have theoretical performance guarantees for the solution and the computational time [3]. Haimovich and Rinnooy Kan [4] introduced the first Polynomial-Time Approximation Scheme (PTAS) for the two-dimensional Euclidean CVRP with time complexity of $O(\log \log n)$. Then Asano et al. [5] improved the result to $q = O(\log n/\log \log n)$. More recently, Das and Mathieu [6] introduced the first Quasi-Polynomial-Time Approximation Scheme (QPTAS) for the two-dimensional Euclidean CVRP. Their scheme gave a $(1+\varepsilon)$-approximation for the two-dimensional Euclidean CVRP in time $n^{(\log n)^{O(1/\varepsilon)}}$. However, their approximation scheme is no longer a QPTAS in general metric space of a fixed doubling dimension. Khachay et al. [7] extended the QPTAS proposed by Das and Mathieu [6] to the case of metric spaces of a fixed doubling dimension. They replaced the exhaustive search in the original QPTAS with the novel internal dynamic

* Corresponding author.
E-mail address: arit@it.kmitl.ac.th (A. Thammano).

https://doi.org/10.1016/j.heliyon.2021.e08029
Received 16 April 2021; Received in revised form 30 July 2021; Accepted 15 September 2021

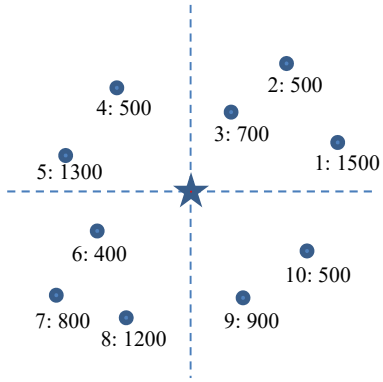


Figure 1. Example of locations and demands of customers. For each location, the customer index and demand are specified as index: demand.

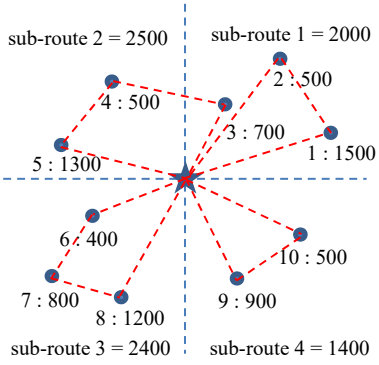


Figure 2. Example of four sub-routes from SA.

program to ensure that the resulting approximation scheme became QPTAS for an arbitrary fixed doubling dimension $d > 1$.

Meta-heuristic algorithms commonly used to solve CVRP can be further divided into an individual search algorithm and a population-based algorithm. The individual search algorithms generate one solution at a time and improves it until it becomes the best solution; the population-based algorithms generate many solutions (a swarm of solutions) at a time, then improve them, and select the best solution among the improved solutions. Examples of algorithms of the former type are Simulated Annealing, Tabu Search (TS) and Greedy Randomized Adaptive Search Procedure (GRASP), while algorithms of the latter type are Ant Colony Optimization (ACO), Firefly Algorithm (FA), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO).

Ezugwu and Adewumi [8] presented a Discrete Symbiotic Organism Search (DSOS) algorithm for solving the Travelling Salesman Problem (TSP). The original Symbiotic Organism Search (SOS) algorithm was inspired by symbiotic relationships among organisms in an ecosystem: mutualism, commensalism, and parasitism. It has been very effective in solving numerical optimization problems. DSOS is an extended version of SOS for solving combinatorial optimization problems. Three mutations were used in DSOS: a swap mutation operator, that randomly picked two placeholders and swapped the two cities at those placeholders; an

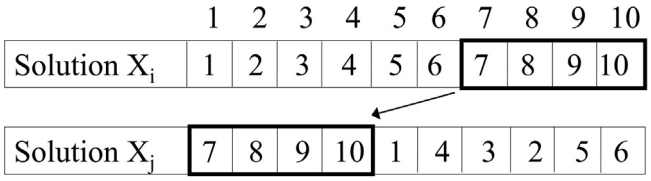


Figure 4. Example of offspring generation.

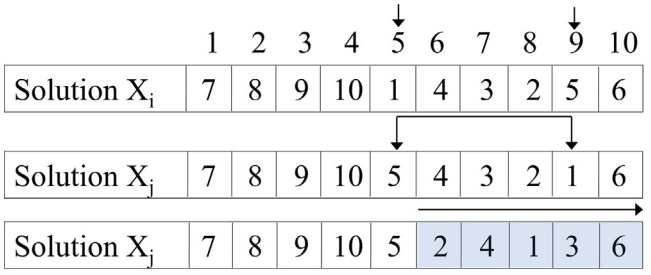


Figure 5. Example of offspring mutation.

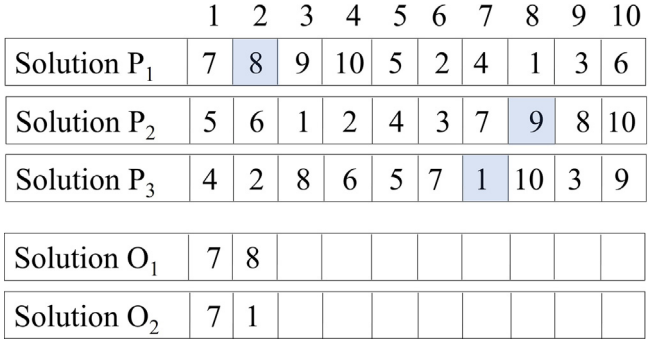
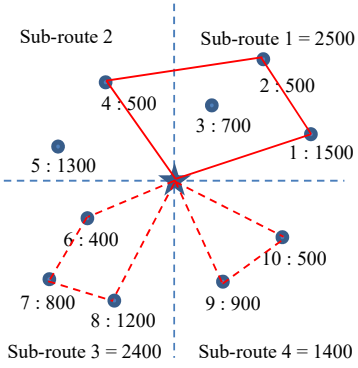
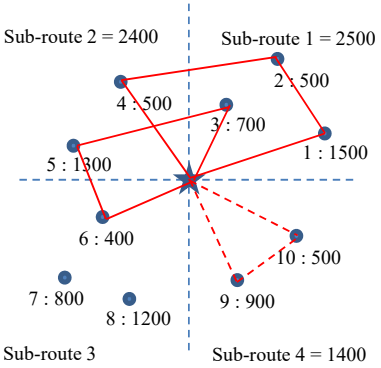


Figure 6. Example of crossover process.



a) Adjusted SR1 sub-route



b) Adjusted SR2 sub-route

Figure 3. Example of adjusted routes (dashed lines represent originally constructed routes; solid lines represent adjusted routes).

Customer	1	2	3	4	5	6	7	8	9	10
Demand	1500	500	700	500	300	600	800	1200	500	300

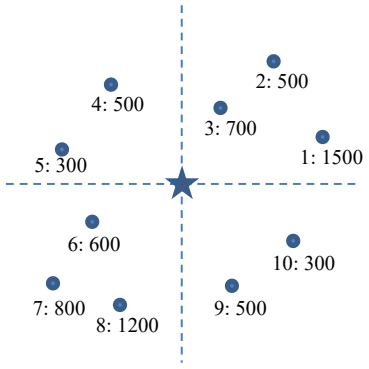


Figure 7. Example of customer locations and demands.

R1	1	2	3	4	5	6	7	8	9	10
R2	2	3	4	5	6	7	8	9	10	1
R3	3	4	5	6	7	8	9	10	1	2
:	:	:	:	:	:	:	:	:	:	:

Figure 8. Example of routes.

Route	City	S1	S2	S3	Constraint Violation
R1	0 1 2 0 3 **4** 5 6 0 7 8 9 10 0	2000	2100	**2800**	True
R2	0 2 3 4 5 0 6 7 0 8 **9** **10** 1 0	2000	1400	**3500**	True
R3	0 3 4 5 6 0 7 8 9 0 10 1 2 0	2100	2500	2300	False
:	:	:	:	:	:

Figure 9. Example of clustered routes with demands.

inversion mutation operator, that randomly picked two placeholders and reversed the order of the cities to be visited in those two places; and an insertion mutation operator, that randomly picked two placeholders, ***i*** and ***j***, and inserted the city in placeholder ***i*** into placeholder ***j***.

Zhong *et al.* [9] described a hybrid method for solving the TSP based on an artificial bee colony (ABC) algorithm and a threshold acceptance criterion. This method converted a continuous ABC to a discrete ABC, in part, by redesigning the solution updating equation, that depended on the information from other bees and the problem at hand. The threshold acceptance criterion was used in place of a greedy acceptance strategy to decide whether to accept a new solution.

Liao and Liu [10] described a hierarchical algorithm that decomposed a Traveling Salesman Problem (TSP) into subproblems by Density Peaks Clustering algorithm, which was used for clustering a node and defining the center of each cluster. Ant Colony Optimization Algorithm was used to find the optimal route for each subproblem, and then connect all center nodes of all subproblems together. Finally, the global route was optimized by k-opt algorithm.

Osaba *et al.* [11] presented a Discrete Water Cycle Algorithm (DWCA), inspired by the water cycle process, *i.e.* how streams flow into rivers and the sea. DWCA is a discrete version of the Water Cycle Algorithm (WCA). Authors used DWCA to solve the Symmetric Traveling Salesman Problem (STSP) and Asymmetric Traveling Salesman Problem (ATSP). In DWCA, Hamming distance was used to determine the distance between the streams and their corresponding rivers or sea. The distance influenced the movement of streams. Streams had two movement methods: insertion for slow moving and 2-opt for fast moving. The evaporation condition and raining process were used to prevent overly fast convergence and becoming stuck in a local optimum.

Marinaki and Marinakis [12] showed a hybrid algorithm for solving vehicle routing problems with stochastic demands; it merged Combinatorial Neighborhood Topology Glowworm Swarm Optimization (CNTGSO) with Variable Neighborhood Search (VNS) and Path Relinking (PR). Two types of vectors were used in this algorithm: $\boldsymbol{x_i}$, with continuous values for calculation of movement of glowworms by GSO, and $\boldsymbol{y_i}$, with discrete values for representation of routes. A local search technique based on VNS was applied to each glowworm to enhance the exploitation capabilities.

Goel and Maini [13] presented a hybrid of Ant Colony System (ACS) and Firefly Algorithm (FA) for solving Capacitated Vehicle Routing Problem (CVRP). In this hybrid algorithm, a discrete version of the ACS served as the basic framework, while the FA was used to explore the search space for new solutions. Since ACS has a drawback of premature convergence, pheromone shaking process was used to prevent the algorithm from getting trapped at local optima.

Pop *et al.* [14] presented a novel two-level optimization approach for solving a clustered vehicle routing problem (CluVRP), in which customers were grouped into a given number of clusters. The two-level optimization decomposed CluVRP into two sub-problems: upper-level and lower-level ones. The upper-level sub-problem created a global route, using a genetic algorithm to connect all clusters. The lower-level sub-problem used the Concorde TSP solver, after transforming the global route into a TSP, to determine the visiting order within a cluster.

Yousefikhoshbakht and Sedighpour [15] proposed a hybrid of the sweep algorithm and the elite ant colony optimization for solving the multiple traveling salesman problem (MTSP). First, initial routes were created using the sweep algorithm. Then the elite ant colony optimization and 3-opt local search were used to further improve the routes.

Chen *et al.* [16] used a hybrid of two-stage sweep algorithm and greedy search, for solving CVRP. Sweep algorithm was used to cluster all customers into groups. Then, it re-clustered the customers, adding customers from adjacent clusters, based on vehicle capacity constraints. These steps helped avoiding local optima. Lastly, a greedy search was used to determine the shortest route for each vehicle.

Osaba *et al.* [17] designed a Discrete Firefly Algorithm (DFA) for solving a newspaper distribution system problem, with a recycling policy, *i.e.,* a Rich Vehicle Routing Problem (R-VRP). Their R-VRP took many

Route	City	S1	S2	S3	Constraint Violation
R1	0 1 2 **4** 0 3 5 6 7 0 8 9 10 0	2500	2400	2000	False
R2	0 2 3 4 5 **9** 0 6 7 **10** 0 8 1 0	2500	1700	**2700**	True
R3	0 3 4 5 6 0 7 8 9 0 10 1 2 0	2100	2500	2300	False
:	:	:	:	:	:

Figure 10. Example of adjusted sub-routes.

	1	2	3	4	5	6	7	8	9	10
Solution R_1	1	2	4	3	5	6	7	8	9	10
Solution R_2	2	1	4	3	6	7	5	9	8	10
Solution R_3	1	3	4	2	5	6	7	8	9	10
Solution R_4	1	2	4	5	6	3	7	8	9	10

Figure 11. Example of solutions in the ranked *R-Set*.

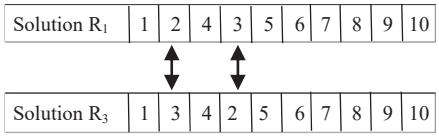


Figure 12. Example of Hamming distance between solutions R1 and R3.

constraints into account, for instance, asymmetry, simultaneous pickup and delivery, variable traveling times, and forbidden paths. Since the original FA was for continuous data, their DFA was designed to deal with discrete data. The Hamming distance was used to determine the distance between two different fireflies, and an insertion function was used to move fireflies from one place to another.

Hannan *et al.* [18] presented a modified particle swarm optimization (PSO) algorithm for finding an optimized route for waste collection. The idea was that finding an optimized route could be assisted by real-time data, from smart bin sensors, to determine which bin was full, and the waste in it had to be collected. Sub-routes were constructed from neighboring locations of smart bins according to vehicle capacity and threshold waste level. Four local search algorithms—2-Opt*, Or-Opt-1, 2-Opt, and Or-Opt—were used to improve routes.

Altabeeb *et al.* [19] presented a hybrid firefly algorithm for a capacitated vehicle routing problem (CVRP-FA) that integrated 2 local search

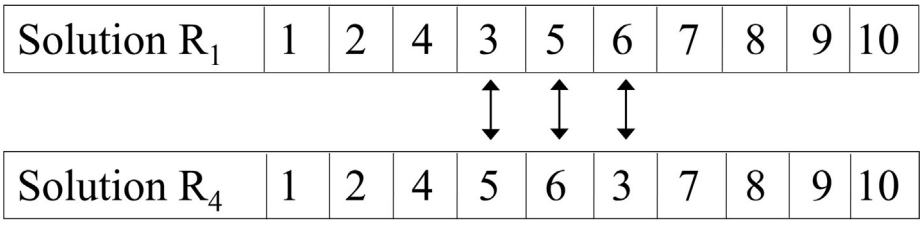


Figure 13. Example of Hamming Distance between solutions R1 and R4.

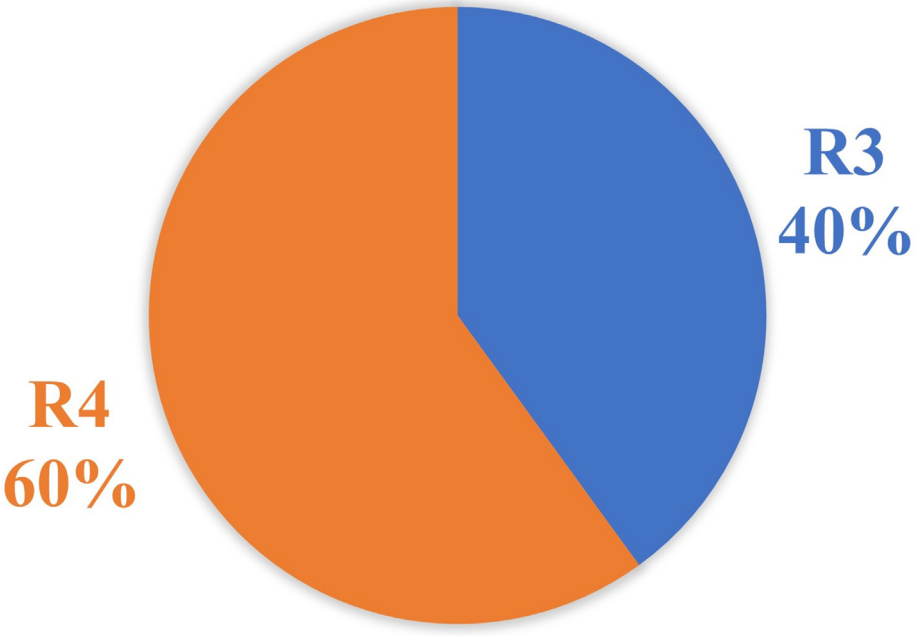


Figure 14. Example of the roulette wheel.

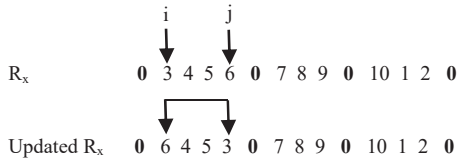


Figure 15. Example of the swap operation.

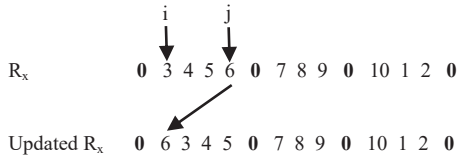


Figure 16. Example of the insert operation.

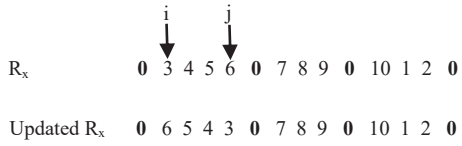


Figure 17. Example of the reverse operation.

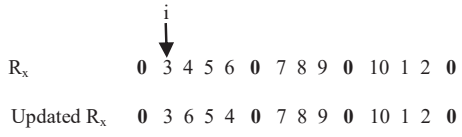


Figure 18. Example of the greedy search operation.

Table 1. Initial parameter values.

Parameter	Value
ρ	0.01–1
τ	3
α	1
β	2
maxT	10000

algorithms and genetic operators to FA. In CVRP-FA, initial fireflies were randomly generated under the capacity constraint. The Hamming distance was used to calculate the distance between two fireflies. Partially matched crossover was used to generate two new offspring. The better one was chosen as a new firefly. CVRP-FA applied two local search algorithms—improved 2-opt and 2-h-opt—to enhance the solution. Two types of mutation—swapping two customers within the same sub-route and swapping two customers in two different sub-routes—were used to prevent the premature convergence.

Although many researchers have studied VRP, there remain many questions in this field that need to be addressed. Since VRP is an NP-hard problem, researchers are still attempting to find more efficient methods to solve it.

This paper describes a hybrid algorithm that combines sweep algorithm for generating good initial solutions, Modified Ant System (MAS) for subsequent building of new generations of good solutions, and path relinking for finding a better solution from a pair of guiding and initial solutions. Finally, four local search methods were used to further improve the solutions.

The paper is organized as follows: Section 2 describes several background concepts; Sections 3 and 4 discuss our methods and experimental results. Section 5 concludes the paper.

2. Background concepts

This section briefly describes the background of VRP, ant Colony Optimization, Sweep Algorithm and Path Relinking.

2.1. VRP

VRP is a problem of determining optimal delivery or pickup routes from a depot to a number of geographically scattered customers. The objective of VRP is to minimize the total distance in serving all customers, as shown in (1):

$$Z = \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1,\ i\neq j}^{n} d_{ij} X_{ij}^{k} \tag{1}$$

$$X_{ij}^{k} = \begin{cases} 1; & \textit{if vehicle } k \textit{ travels from } i \textit{ to } j \\ 0; & \textit{otherwise} \end{cases} \tag{2}$$

where n is the number of customer sites, m is the number of vehicles and d_{ij} is the distance from customer i to customer j.

CVRP is a VRP with the vehicle capacity constraint [16]. The total customer demands on each vehicle must be less than its load capacity:

$$\sum_{i=1}^{n} q_i Y_i^k < Q \tag{3}$$

$$Y_i^k = \begin{cases} 1; & \textit{if customer } i \textit{ is served by vehicle } k \\ 0; & \textit{otherwise} \end{cases} \tag{4}$$

where $k = 1, 2, \ldots, m$; m is the number of vehicles; q_i is the demand of customer i, and Q is the vehicle capacity.

2.2. Ant colony optimization

Ant Colony Optimization (ACO) is commonly used to solve combinatorial optimization problems. Ant System (AS) [20] is the first ACO algorithm. It is inspired by the behavior of ants finding food. AS searches for the shortest route from the ant colony to the food source with the help of pheromone, a chemical substance that ants use to communicate with each other.

In the beginning, each ant is assigned a different starting city to increase route diversity. The next city for an ant to visit is determined by the transition probability:

$$P_{ij}^{k} = \frac{(\tau_{ij})^{\alpha}(d_{ij})^{\beta}}{\sum_{l\in p}(\tau_{il})^{\alpha}(d_{il})^{\beta}} \tag{5}$$

where $P_{ij}^{k}$is the transition probability of ant k choosing a city j to travel to from city i; τ_{ij} is the pheromone level along the path from city i to j; d_{ij} is the distance from city i to city j; α and β are parameters that control the relative weight of the pheromone and the distance, respectively.

In the pheromone update process, the pheromone level on each path is updated after all ants have finished their tours. A path on which many ants have traversed will have a higher pheromone level than a less frequently traversed path, and is updated by these equations:

$$\tau_{ij}' = \rho\tau_{ij} + \sum_{k=1}^{n} \Delta\tau_{ij}^{k} \tag{6}$$

Table 2. Comparative results of MAS-SA-PR and MAS on Set A.

	Instance	BKS	MAS			MAS-SA-PR		
			Best	%Dev	Time (mm:ss)	Best	%Dev	Time (mm:ss)
1	A-n32-k5	784	813	3.70	00:21	**796**	1.53	01:17
2	A-n33-k5	661	668	1.06	00:22	**661**	0.00	00:51
3	A-n33-k6	742	743	0.13	00:23	**742**	0.00	01:09
4	A-n34-k5	778	779	0.13	00:24	**778**	0.00	04:12
5	A-n36-k5	799	818	2.38	00:29	**799**	0.00	01:27
6	A-n37-k5	669	693	3.59	00:31	**669**	0.00	00:48
7	A-n37-k6	949	953	0.42	00:33	**949**	0.00	02:12
8	A-n38-k5	730	738	1.10	00:36	**730**	0.00	01:41
9	A-n39-k5	822	832	1.22	00:38	**822**	0.00	02:32
10	A-n39-k6	831	834	0.36	00:36	**831**	0.00	01:54
11	A-n44-k6	937	**937**	0.00	00:53	**937**	0.00	01:39
12	A-n45-k6	944	948	0.42	01:22	**944**	0.00	02:25
13	A-n45-k7	1146	1164	1.57	00:58	**1146**	0.00	03:08
14	A-n46-k7	914	934	2.19	00:57	**914**	0.00	01:48
15	A-n48-k7	1073	1123	4.66	01:05	**1073**	0.00	02:03
16	A-n53-k7	1010	1047	3.66	01:40	**1014**	0.40	04:21
17	A-n54-k7	1167	**1169**	0.17	01:38	**1169**	0.17	05:30
18	A-n55-k9	1073	1076	0.28	01:41	**1074**	0.09	07:59
19	A-n60-k9	1354	1399	3.32	02:02	**1358**	0.30	09:23
20	A-n61-k9	1034	1051	1.64	03:00	**1034**	0.00	08:48
21	A-n62-k8	1288	1338	3.88	02:14	**1310**	1.71	08:42
22	A-n63-k9	1616	1623	0.43	03:12	**1622**	0.37	09:07
23	A-n63-k10	1314	1327	0.99	02:23	**1319**	0.38	10:49
24	A-n64-k9	1401	1453	3.71	02:43	**1416**	1.07	05:22
25	A-n65-k9	1174	1178	0.34	03:24	**1177**	0.26	08:20
26	A-n69-k9	1159	1175	1.38	03:09	**1159**	0.00	20:58
27	A-n80-k10	1763	1801	2.16	04:46	**1792**	1.64	09:30

The best result among the compared algorithms is shown in bold.

$$\Delta\tau_{ij}^{k} = \frac{1}{Z^{k}} \tag{7}$$

where ρ is the pheromone persistence coefficient. If the ant k walks along the path *(i, j)*, the pheromone level of the path will increase by $\Delta\tau_{ij}^{k}$, defined in Eq. (7). Z^{k} is the total distance that the ant k has traversed.

The MAX-MIN Ant System (MMAS) [21] improves on the AS algorithm. In MMAS, the pheromone level has maximum and minimum limits. After each iteration, the pheromone level is updated according to Eqs. (8) and (9).

$$\tau_{ij}^{'} = \rho\tau_{ij} + \Delta\tau_{ij}^{best} \tag{8}$$

$$\Delta\tau_{ij}^{best} = \frac{1}{f(s^{best})} \tag{9}$$

where $f(s^{best})$ may be either $f(s^{ib})$, the total distance of the best solution in that iteration, or $f(s^{gb})$, the total distance of the global best solution in all iterations. If the updated τ_{ij} is higher than the maximum value, it is set to the maximum value, and if the updated τ_{ij} is lower than the minimum, it is set to the minimum value.

Zao et al. [22] proposed a modified MMAS. The modified MMAS calculates $\tau_{\max}$ and $\tau_{\min}$, the maximum and minimum pheromone levels using these equations:

$$\tau_{\max} = \frac{1}{\rho f(s^{best})} \tag{10}$$

$$\tau_{\min} = \frac{1}{f(s^{worst})} \tag{11}$$

The pheromone update process uses the global best solution of all iterations as well as the best and second-best solutions of each iteration to update pheromone levels in each iteration. To avoid getting trapped in local minima when selecting the next city to visit, a random number is generated and compared with q_{bias}, a bias parameter, to decide which city is the next destination. More precisely, the algorithm randomly generates $p \in (0,1)$ and compares it to q_{bias}. If $p > q_{bias}$, the city with the highest transition probability is selected as the next city, otherwise the city with the second highest transition probability is selected as the next city. This increases route diversity. In addition, the pheromone evaporation rate is modified so that it changes dynamically when the solution becomes stuck in a local minimum.

2.3. Sweep algorithm

Gillett and Miller [23] developed a Sweep Algorithm (SA) for solving VRP. In their SA, a customer location is indicated by a polar coodinate. Then routes are constructed according to ranked polar coordinates of the customers and the vehicle capacity constraint. In route construction, polar angles of all customer locations are ranked from the smallest to the largest, and every route that starts from a randomly selected location is constructed by connecting that location to the location with the next larger angle from the start location. This continues until all customer locations are connected into a route.

A Two-stage Sweep Algorithm [16] is an improved version of SA. In the first stage, customers are clustered by using the original SA. The second stage of this two-stage sweep algorithm adjusts each cluster by adding a customer (or customers) from an adjacent cluster that does not violate the constraint. This enables construction of more diverse routes. The steps of this algorithm are described below.

Table 3. Comparative results of MAS-SA-PR and MAS on Set B.

	Instance	BKS	MAS			MAS-SA-PR		
			Best	%Dev	Time (mm:ss)	Best	%Dev	Time (mm:ss)
1	B-n31-k5	672	**672**	0.00	00:21	**672**	0.00	03:28
2	B-n34-k5	788	789	0.13	00:28	**788**	0.00	00:35
3	B-n35-k5	955	963	0.84	00:28	**955**	0.00	01:17
4	B-n38-k6	805	806	0.12	00:36	**805**	0.00	02:04
5	B-n39-k5	549	560	2.00	00:38	**549**	0.00	01:15
6	B-n41-k6	829	**829**	0.00	00:43	**829**	0.00	01:26
7	B-n43-k6	742	749	0.94	00:47	**742**	0.00	03:14
8	B-n44-k7	909	922	1.43	00:52	**909**	0.00	01:22
9	B-n45-k5	751	755	0.53	01:03	**751**	0.00	01:22
10	B-n45-k6	678	691	1.92	01:26	**678**	0.00	01:56
11	B-n50-k7	741	746	0.67	01:19	**741**	0.00	02:21
12	B-n50-k8	1312	1331	1.45	01:17	**1322**	0.76	06:27
13	B-n51-k7	1032	**1032**	0.00	01:44	**1032**	0.00	02:27
14	B-n52-k7	747	771	3.21	01:39	**747**	0.00	03:35
15	B-n56-k7	707	727	2.83	02:02	**709**	0.28	03:05
16	B-n57-k7	1153	1164	0.95	02:56	**1157**	0.35	04:17
17	B-n57-k9	1598	1651	3.32	01:54	**1611**	0.81	06:04
18	B-n63-k10	1496	1557	4.08	02:23	**1521**	1.67	06:11
19	B-n64-k9	861	868	0.81	03:31	**864**	0.35	08:07
20	B-n66-k9	1316	1337	1.60	02:57	**1321**	0.38	23:28
21	B-n67-k10	1032	1075	4.17	02:59	**1039**	0.68	06:08
22	B-n68-k9	1272	1298	2.04	03:25	**1290**	1.42	05:24
23	B-n78-k10	1221	1245	1.97	04:44	**1239**	1.47	12:26

The best result among the compared algorithms is shown in bold.

Step 1. Convert all customer locations to polar coordinates, specified by r and θ (defined in Eq. (12)). Then sort all customers in ascending order according to their angles.

$$\theta_i = \begin{cases} \tan^{-1}\frac{y_i}{x_i}, & x_i > 0, y_i > 0 \\ \pi + \tan^{-1}\frac{y_i}{x_i}, & x_i < 0 \\ 2\pi + \tan^{-1}\frac{y_i}{x_i}, & x_i > 0, y_i < 0 \end{cases} \tag{12}$$

where x_i and y_i are the cartesian coordinates of the location of customer i.

Step 2. Select one customer as a starting point of the first sub-route. Then, the next customer on the list is added to the sub-route as long as the vehicle capacity constraint is satisfied. However, if the next customer demand exceeds the remaining capacity of the vehicle, create a new sub-route with that customer as a starting point. This process continues until all customers have been assigned to one of the sub-routes.

Step 3. To obtain diverse initial solutions, Step 2 is repeated n-1 times with any customer that has not been chosen as the starting point of the first sub-route of the previous routes.

Step 4. Combine the adjacent sub-routes.

For example, as illustrated in Figure 1, the customer indices and demands are shown. There are 10 customers to be served by 4 capacitated vehicles, each with a capacity of 2500 kg. First, each customer location is converted to polar coordinates and sorted in ascending order according to the angles. The sorted list is as follows: 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10. Next, customer 1 is randomly selected as a starting point of the first sub-route. Then the process describes in "Step 2" is carried out. As illustrated in Figure 2, the customers are grouped into four clusters (four sub-routes). The total demand for each cluster is less than or equal to 2500 kg. Sub-route 1, SR1, covers customers 1 and 2 with a total customer demand of 2000; SR2 covers customers 3, 4 and 5, with a total demand of 2500; SR3 covers customers 6, 7 and 8, with a total demand of 2400; and SR4 covers customers 9 and 10 with a total demand of 1400. Therefore, the final route is 0-1-2-0-3-4-5-0-6-7-8-0-9-10-0, where 0 signifies a transition from a vehicle (or sub-route) to another vehicle (or the next sub-route).

After the route construction ends, the next step is to combine adjacent clusters. In Figure 2, the demand of customers in SR1 is only 2000 kg, less than the vehicle capacity. Hence, the 500 kg demand of customer 4 on SR2 can be transferred to SR1, and the new adjusted SR1 covers customers 1, 2, and 4, with a total demand of 2500 kg, see Figure 3a. The above adjustment leaves SR2 with only customers 3 and 5, with a total demand of 2000. Hence, the 400 kg demand of customer 6, previously assigned to SR3, can be transferred to SR2. Then, the new adjusted SR2 now covers customers 3, 5 and 6, with a total customer demand of 2400 kg, see Figure 3b. Next, this process is further applied to the next sub-route until all sub-routes are adjusted.

2.4. Path relinking

Path Relinking (PR) [24, 25] is a method for exploring trajectories connecting elite solutions. PR selects an elite guiding solution and an initial solution from a reference set, *R-Set*. Then it generates routes that start with the initial solution and end with the guiding solution, with the hope of finding better solutions along these newly generated routes. If no new solutions are better than the guiding solution, the final solution from path relinking will be the guiding solution itself. The three steps in path relinking are:

Step 1: Building a Reference Set

A Reference Set (*R-Set*) is a set of good solutions. It is initially constructed using Tabu Search (TS) method. As the path relinking iterates, more solutions are added to the *R-Set*.

Step 2: Pairing guiding and initial solutions

Guiding and initial solutions are elite solutions in *R-Set*. The way of pairing them strongly affects the performance of the path relinking. Five common pairing methods are:

Table 4. Comparative results of MAS-SA-PR and MAS on Set P.

	Instance	BKS	MAS			MAS-SA-PR		
			Best	% Dev	Time (mm:ss)	Best	% Dev	Time (mm:ss)
1	P-n16-k8	450	**450**	0.00	00:05	**450**	0.00	02:17
2	P-n19-k2	212	**212**	0.00	00:05	**212**	0.00	00:12
3	P-n20-k2	216	**216**	0.00	00:06	**216**	0.00	00:25
4	P-n21-k2	211	**211**	0.00	00:06	**211**	0.00	00:26
5	P-n22-k2	216	**216**	0.00	00:07	**216**	0.00	00:23
6	P-n22-k8	603	**603**	0.00	00:16	**603**	0.00	00:43
7	P-n23-k8	529	**529**	0.00	00:21	**529**	0.00	00:39
8	P-n40-k5	458	467	1.97	00:38	**458**	0.00	02:36
9	P-n45-k5	510	529	3.73	00:52	**510**	0.00	02:18
10	P-n50-k7	554	587	5.96	01:12	**556**	0.36	02:27
11	P-n50-k8	631	658	4.28	01:43	**634**	0.48	06:03
12	P-n50-k10	696	705	1.29	01:29	**697**	0.14	05:12
13	P-n51-k10	741	746	0.67	01:53	**741**	0.00	05:43
14	P-n55-k7	568	601	5.81	01:33	**573**	0.88	02:56
15	P-n55-k8	588	588	0.00	01:34	**576**	-2.04	10:37
16	P-n55-k10	694	705	1.59	01:34	**698**	0.58	07:19
17	P-n60-k10	744	772	3.76	02:15	**745**	0.13	09:53
18	P-n60-k15	968	990	2.27	02:46	**968**	0.00	09:37
19	P-n65-k10	792	827	4.42	02:43	**792**	0.00	19:00
20	P-n70-k10	827	844	2.06	04:04	**830**	0.36	16:23
21	P-n76-k4	593	625	5.40	03:57	**594**	0.17	17:36
22	P-n76-k5	627	643	2.55	04:01	**627**	0.00	23:46

The best result among the compared algorithms is shown in bold.

1) Set the best solution in *R-Set* to be the guiding solution and the worst solution to be the initial solution.
2) Set the best solution in *R-Set* to be the guiding solution and the second-best solution to be the initial solution.
3) Set the best solution in *R-Set* to be the guiding solution and the solution with the maximum Hamming distance from it to be the initial solution.
4) Randomly select the guiding and initial solutions.
5) Select the guiding and initial solutions that are separated by the maximum Hamming distance between any two solutions in the *R-Set*.

Step 3: Generating a candidate solution.

A candidate solution is generated from a pair of guiding and initial solutions by comparing and finding the difference between the guiding and initial solutions. Next, the initial solution is moved towards the guiding solution if there is a difference. The number of movements is equal to the number of solution placeholders in which the cities are different. The candidate solution can be either the guiding solution itself or a solution resulted from the movements if it is better than the guiding solution.

3. Our proposed method

This section describes our hybrid algorithm, a combination of MAS, SA, and PR, for solving CVRP. The main algorithm and its components are explained in the following subsections.

3.1. Hybrid modified ant system with sweep algorithm and path relinking (MAS-SA-PR)

This hybrid algorithm uses MAS as the main algorithm. Our version of MAS uses the pheromone updating strategy which is inspired from MMAS. However, instead of using either the global best or the iteration best solutions, our MAS updates the pheromone level based on the combination of these 3 solutions: the global best, the iteration best, and the second iteration best solutions. The pseudocode of the MAS-SA-PR algorithm is shown in Algorithm 1 and its steps are described below:

Step 1: Initializing parameters

This first step initializes the number of ants and customers, the maximum number of iterations (*maxT*), the initial pheromone level (τ), the pheromone evaporation rate (ρ), and the size of *R-Set* and *G-Set* (s).

Step 2: Initializing routes

This step uses a sweep algorithm (see Section 3.2) to construct initial routes from the input data. The sweep algorithm clusters customers based on the polar angles of their locations. The number of initial routes is equal to the number of ants and generally equal to the number of customers to be visited.

Step 3: Generating improved solutions and finding the best solution

1) Generating offspring

To generate an offspring from a parent route, we randomly select a customer in the parent route and take a sub-route, starting from the selected customer towards the end of the route, to be the starting sub-route for an offspring. Next, to complete the offspring, customers that have not been included in the sub-route are selected one after another based on its transition probability, Eq. (5). To be exact, the customer with the highest transition probability is selected immediately, only if its transition probability $\leq S_{bias}$, where S_{bias} is a randomly generated value, acting as a threshold to reduce the possibility of getting trapped in a local minimum. Otherwise, the customer with the second highest transition probability is selected. This strategy increases the diversity of constructed routes. For example, as illustrated in Figure 4, if X_i is a parent route and 7 is a randomly selected customer, the starting customer for the offspring, X_j, will be 7 and the sub-route 7-8-9-10 will be the starting sub-route for this offspring. The remaining customers, to be added to the empty slots of the offspring, are further selected according to the procedure above, *i.e.* 1-4-3-2-5-6 in this example.

For each offspring, customers are clustered into sub-routes according to the number of vehicles and the vehicle capacity constraints. If the clustering cannot cluster all customers, with the specified number of vehicles, the following adjustment is performed. First, the customers, which have not been assigned any vehicle, are selected, one by one. Then, the best cluster, among all clusters, is determined (the one that can accommodate that specific customer and has the shortest total distance after the accommodation), and that customer is assigned to it. However, if no vehicle can accommodate the leftover customers, the customers in the route are randomly swapped, and the adjustment starts again. This is repeated until the maximum number of iterations, t_{max}, is reached. Finally, for each offspring, the total distance is calculated and the constraint violation (a boolean which identifies whether any customers remain unassigned to the sub-route) is recorded.

2) Mutating the offspring

a. Mutate each offspring by randomly selecting two customers in the offspring, then swap those customers. Next, the remaining customers, starting from the one next to the first swapped customer, are reordered. This reordering uses the transition probability and S_{bias} mentioned above. For example, in Figure 5, the solution X_i is 7-8-9-10-1-4-3-2-5-6. If the randomly selected customer indices are 5 and 9, customers 1 and 5 are swapped. Then, the remaining customers are reordered, starting from the customer to the right of the customer at index 5.
b. Cluster customers according to the number of vehicles and the vehicle capacity constraints. Then the total distance is calculated, and the constraint violation is recorded.

Table 5. Comparative results of MAS-SA-PR, LNS-ACO, RBMC-ABC, and CVRP-FA on Set A.

	Instance	BKS	LNS-ACO		RBMC-ABC		CVRP-FA		MAS-SA-PR	
			Best	%Dev	Best	%Dev	Best	%Dev	Best	%Dev
1	A-n32-k5	784	**784**	0.00	**784**	0.00	796	1.53	796	1.53
2	A-n33-k5	661	**661**	0.00	**661**	0.00	**661**	0.00	**661**	0.00
3	A-n33-k6	742	**742**	0.00	**742**	0.00	**742**	0.00	**742**	0.00
4	A-n34-k5	778	**778**	0.00	**778**	0.00	**778**	0.00	**778**	0.00
5	A-n36-k5	799	**799**	0.00	**799**	0.00	**799**	0.00	**799**	0.00
6	A-n37-k5	669	**669**	0.00	**669**	0.00	**669**	0.00	**669**	0.00
7	A-n37-k6	949	**949**	0.00	**949**	0.00	**949**	0.00	**949**	0.00
8	A-n38-k5	730	**730**	0.00	**730**	0.00	**730**	0.00	**730**	0.00
9	A-n39-k5	822	**822**	0.00	**822**	0.00	**822**	0.00	**822**	0.00
10	A-n39-k6	831	**831**	0.00	**831**	0.00	**831**	0.00	**831**	0.00
11	A-n44-k6	937	**937**	0.00	**937**	0.00	**937**	0.00	**937**	0.00
12	A-n45-k6	944	958	1.48	**944**	0.00	953	0.95	**944**	0.00
13	A-n45-k7	1146	**1146**	0.00	**1146**	0.00	1147	0.09	**1146**	0.00
14	A-n46-k7	914	**914**	0.00	**914**	0.00	**914**	0.00	**914**	0.00
15	A-n48-k7	1073	1084	1.03	**1073**	0.00	**1073**	0.00	**1073**	0.00
16	A-n53-k7	1010	**1010**	0.00	**1010**	0.00	1011	0.10	1014	0.40
17	A-n54-k7	1167	**1167**	0.00	**1167**	0.00	1172	0.43	1169	0.17
18	A-n55-k9	1073	**1073**	0.00	**1073**	0.00	1074	0.09	1074	0.09
19	A-n60-k9	1354	**1354**	0.00	**1354**	0.00	1355	0.07	1358	0.30
20	A-n61-k9	1034	1067	3.19	1035	0.10	1039	0.48	**1034**	0.00
21	A-n62-k8	1288	1308	1.55	**1292**	0.31	1298	0.78	1310	1.71
22	A-n63-k9	1616	1649	2.04	1627	0.68	1630	0.87	**1622**	0.37
23	A-n63-k10	1314	1329	1.14	1320	0.46	**1314**	0.00	1319	0.38
24	A-n64-k9	1401	**1415**	1.00	1416	1.07	1420	1.36	1416	1.07
25	A-n65-k9	1174	1185	0.94	**1174**	0.00	1178	0.34	1177	0.26
26	A-n69-k9	1159	1170	0.95	1170	0.95	1162	0.26	**1159**	0.00
27	A-n80-k10	1763	1815	2.95	1786	1.30	**1773**	0.57	1792	1.64

The best result among the compared algorithms is shown in bold.

3) Select the best solution from the sets of offspring and mutated offspring. If the best solution is better than the worst in *R-Set*, the worst solution is replaced by this best solution.
4) Perform Path relinking (see Section 3.3).
5) Selecting n individuals for the next generation

   a. The parent, offspring, and mutated offspring population are combined with Path relinking solutions into a single population.
   b. The top 0.7n routes are selected from the combined population and included into the next generation population. The remaining 0.3n routes of the next generation population are selected from the rest of the combined population. These remaining routes must have the constraint violation = "false."
   c. If any of the next generation population is better than the worst solution in *R-Set*, *R-Set* will be updated with that solution.
   d. If any of the next generation population with the constraint violation = "false" is better than the worst solution in *G-Set*, *G-Set* will be updated with that solution.

6) Updating the pheromone level

The pheromone level of each path is updated based on 3 solutions — *gBest* (the best solution in *G-Set*), the best and second-best solutions in the next generation population. In updating, the pheromone level of each path is updated by using Eqs. (8) and (9).

7) Perform local searches (see Section 3.4) on the solutions in *R-Set* and *G-Set*.
8) When more than 80% of the next generation population is identical, a new set of population is generated by using the following procedure.

   a. Randomly select a solution from *G-Set* or *R-Set* (if there is no solution in the *G-Set*) to be Parent 1, P_1. Then randomly select another two different solutions from the next generation population to be Parent 2, P_2, and Parent 3, P_3.
   b. Select the first customer of P_1 to be the first customer of two offsprings.
   c. For each offspring, the next customer is selected from a choice of three customers, each of which is next to the current customer in the parent P_1, P_2 and P_3. Among these three choices, the one led to the minimum cost is selected for offspring 1, O_1, and the second minimum cost for offspring 2, O_2. Then repeat the above process to choose the next customers until completing the routes for O_1 and O_2.

      For example, in Figure 6, customer 7 is the first customer of P_1. Therefore, it is set to be the first customer of O_1 and O_2. The next customers to be put next to customer 7 in O_1 and O_2 are selected from the customers that are next to customer 7 in parents P_1, P_2, and P_3. These are 8, 9, and 1. The next customer for O_1 is 8 if it incurs the minimum cost from connecting to customer 7, and the next customer for O_2 is 1 if it incurs the second minimum cost from connecting to customer 7.
   d. Cluster customers according to the number of vehicles and the vehicle capacity constraints. Then the total distance is calculated, and the constraint violation is recorded.

Step 4: The termination criterion is checked. If it is met, *gbest* is returned as the best solution found. Otherwise, Step 3 is repeated.

Algorithm 1. Hybrid modified ant system with sweep algorithm and path relinking

Table 6. Comparative results of MAS-SA-PR, LNS-ACO, RBMC-ABC, and CVRP-FA on Set B.

	Instance	BKS	LNS-ACO		RBMC-ABC		CVRP-FA		MAS-SA-PR	
			Best	%Dev	Best	%Dev	Best	%Dev	Best	%Dev
1	B-n31-k5	672	**672**	0.00	**672**	0.00	**672**	0.00	**672**	0.00
2	B-n34-k5	788	**788**	0.00	**788**	0.00	**788**	0.00	**788**	0.00
3	B-n35-k5	955	**955**	0.00	**955**	0.00	**955**	0.00	**955**	0.00
4	B-n38-k6	805	**805**	0.00	**805**	0.00	806	0.12	**805**	0.00
5	B-n39-k5	549	**549**	0.00	**549**	0.00	550	0.18	**549**	0.00
6	B-n41-k6	829	**829**	0.00	**829**	0.00	**829**	0.00	**829**	0.00
7	B-n43-k6	742	**742**	0.00	**742**	0.00	**742**	0.00	**742**	0.00
8	B-n44-k7	909	**909**	0.00	**909**	0.00	**909**	0.00	**909**	0.00
9	B-n45-k5	751	**751**	0.00	**751**	0.00	**751**	0.00	**751**	0.00
10	B-n45-k6	678	**678**	0.00	**678**	0.00	686	1.18	**678**	0.00
11	B-n50-k7	741	**741**	0.00	**741**	0.00	**741**	0.00	**741**	0.00
12	B-n50-k8	1312	1319	0.53	**1316**	0.30	1318	0.46	1322	0.76
13	B-n51-k7	1032	**1032**	0.00	**1032**	0.00	**1032**	0.00	**1032**	0.00
14	B-n52-k7	747	**747**	0.00	**747**	0.00	**747**	0.00	**747**	0.00
15	B-n56-k7	707	**707**	0.00	**707**	0.00	709	0.28	709	0.28
16	B-n57-k7	1153	**1153**	0.00	**1153**	0.00	**1153**	0.00	1157	0.35
17	B-n57-k9	1598	**1598**	0.00	1604	0.38	1610	0.75	1611	0.81
18	B-n63-k10	1496	1514	1.20	**1497**	0.07	1503	0.47	1521	1.67
19	B-n64-k9	861	874	1.51	**861**	0.00	862	0.12	864	0.35
20	B-n66-k9	1316	1330	1.06	1322	0.46	**1319**	0.23	1321	0.38
21	B-n67-k10	1032	1050	1.74	**1037**	0.48	1042	0.97	1039	0.68
22	B-n68-k9	1272	1290	1.42	**1277**	0.39	1278	0.47	1290	1.42
23	B-n78-k10	1221	1228	0.57	1239	1.47	**1224**	0.25	1239	1.47

The best result among the compared algorithms is shown in bold.

Table 7. Comparative results of MAS-SA-PR, LNS-ACO, RBMC-ABC, and CVRP-FA on Set P.

	Instance	BKS	LNS-ACO		RBMC-ABC		CVRP-FA		MAS-SA-PR	
			Best	%Dev	Best	%Dev	Best	%Dev	Best	%Dev
1	P-n16-k8	450	**450**	0.00	**450**	0.00	**450**	0.00	**450**	0.00
2	P-n19-k2	212	**212**	0.00	**212**	0.00	**212**	0.00	**212**	0.00
3	P-n20-k2	216	**216**	0.00	**216**	0.00	**216**	0.00	**216**	0.00
4	P-n21-k2	211	**211**	0.00	**211**	0.00	**211**	0.00	**211**	0.00
5	P-n22-k2	216	**216**	0.00	**216**	0.00	**216**	0.00	**216**	0.00
6	P-n22-k8	603	**603**	0.00	**603**	0.00	**603**	0.00	**603**	0.00
7	P-n23-k8	529	**529**	0.00	**529**	0.00	**529**	0.00	**529**	0.00
8	P-n40-k5	458	**458**	0.00	**458**	0.00	**458**	0.00	**458**	0.00
9	P-n45-k5	510	**510**	0.00	**510**	0.00	**510**	0.00	**510**	0.00
10	P-n50-k7	554	**554**	0.00	**554**	0.00	**554**	0.00	556	0.36
11	P-n50-k8	631	643	1.90	**631**	0.00	**631**	0.00	634	0.48
12	P-n50-k10	696	**696**	0.00	698	0.29	697	0.14	697	0.14
13	P-n51-k10	741	747	0.81	742	0.13	742	0.13	**741**	0.00
14	P-n55-k7	568	**568**	0.00	**568**	0.00	**568**	0.00	573	0.88
15	P-n55-k8	588	588	0.00	577	-1.87	**576**	-2.04	**576**	-2.04
16	P-n55-k10	694	**694**	0.00	699	0.72	698	0.58	698	0.58
17	P-n60-k10	744	755	1.48	**745**	0.13	749	0.67	**745**	0.13
18	P-n60-k15	968	977	0.93	**968**	0.00	**968**	0.00	**968**	0.00
19	P-n65-k10	792	800	1.01	**792**	0.00	**792**	0.00	**792**	0.00
20	P-n70-k10	827	837	1.21	840	1.57	**827**	0.00	830	0.36
21	P-n76-k4	593	598	0.84	**593**	0.00	**593**	0.00	594	0.17
22	P-n76-k5	627	645	2.87	628	0.16	628	0.16	**627**	0.00

The best result among the compared algorithms is shown in bold.

```
Algorithm 1: HYBRID MODIFIED ANT SYSTEM WITH SWEEP ALGORITHM AND PATH RELINKING
BEGIN
    Initialize parameters;
    Initialize routes with SWEEP ALGORITHM;
    REPEAT
        FOR each parent route                                   /* Generating offspring */
            Randomize the selected customer;
            Take a subroute, starting from the selected customer towards the end of the route, to be the
            starting sub-route for an offspring;
            FOR each remaining position in the offspring
                Calculate the transition probabilities of the unassigned customers by using (5);
                P_max ← the highest transition probability;
                IF P_max ≤ S_bias THEN
                    Select the customer with the highest transition probability;
                ELSE
                    Select the customer with the second highest transition probability;
                END IF
            END FOR
            Cluster customers in the offspring according to the vehicle capacity constraints;
        END FOR
        FOR each offspring                                      /* Mutating offspring */
            Randomly select two customers and swap them;
            Remove all customers located after the first swapped customer;
            FOR each empty position after the first swapped customer
                Calculate the transition probabilities of the unassigned customers by using (5);
                Determine the highest transition probability (P_max);
                IF P_max ≤ S_bias THEN
                    Select the customer with the highest transition probability;
                ELSE
                    Select the customer with the second highest transition probability;
                END IF
            END FOR
            Cluster customers in the offspring according to the vehicle capacity constraints;
        END FOR
        Best ← the best solution from the combined sets of offspring and mutated offspring;
        IF Best is better than the worst solution in R-Set THEN
            Replace the worst solution in R-Set with Best;
        END IF
        Perform PATH RELINKING on solutions in R-Set;
        Select the next generation polpulation from the combined sets of parent, offspring, mutated offspring
        and path relinking solutions;
        FOR each of the next generation population
            IF this solution is better than the worst solution in R-Set THEN
                Replace the worst solution in R-Set with this solution;
            END IF
            IF this solution is better than the worst solution in G-Set AND
               the constraint violation = "false" THEN
                Replace the worst solution in G-Set with this solution;
            END IF
        END FOR
        Update the pheromone level of each path;
        Perform local searches on solutions in G-Set and R-Set;
        gBest ← the best solution in G-Set;
        IF more than 80% of the next generation population is identical THEN
            Generate a new population;
        END IF
    UNTIL the maximum number of iterations is reached;
    RETURN gBest;
END
```

3.2. Sweep algorithm and the sub-route adjustment method

We use a sweep algorithm to generate initial routes and assign customers to vehicles. The sweep algorithm clusters the customers based on the polar angles of their locations. The detailed steps for the sweep algorithm and the sub-route adjustment method are set out in Algorithm 2 and described below:

Step 1: Calculate the polar angle, θ, of the location of every customer, defined in Eq. (12), and sort them in ascending order.

Step 2: Generating routes.

Generate routes (or possible solutions) according to the number of vehicles and the vehicle capacity constraints.

1) Assign a different starting customer location for each route. Then sequentially assign the remaining customers in the route in ascending polar angle.
2) Cluster customers according to the number of vehicles and the vehicle capacity constraints. For each route, select the first customer as a starting point of the first sub-route. Then, the next customer is added to the sub-route as long as the vehicle capacity constraint is satisfied. However, if the next customer demand exceeds the remaining capacity of the vehicle, create a new sub-route with that customer as a

starting point. This process continues until all customers have been assigned to one of the sub-routes.

3) If the clustering cannot successfully cluster all customers, under the constraint on the number of vehicles, start a sub-route adjustment. Sub-route adjustment starts by searching for customers who do not violate capacity and distance constraints in the next-to-the-current cluster and the next-to-next one, then move them to the current cluster. After a sub-route is adjusted, *i.e.* a cluster is modified, the current customer locations are re-clustered.
4) Calculate the total distance of all sub-routes and set the constraint violation of each route as true or false to identify the routes that violate the vehicle capacity constraint.

The locations and demands of customers, illustrated in Figure 7, are used to demonstrate how the sweep algorithm and sub-route adjustment method work. In this example, the number of vehicles is limited to 3, each with a 2500 kg capacity. Figure 8 shows routes that have not yet been clustered. Figure 9 shows clustered routes and customer demands for each vehicle. Route R1 has three sub-routes: R1S1 covers customers 1 and 2 with a total customer demand of 2000 kg; R1S2 covers the customers 3, 4, 5 and 6 with a total customer demand of 2100 kg; and R1S3 covers customers 7, 8, 9 and 10 with a total customer demand of 2800 kg. Therefore, the constraint violation of R1 is set to be true because the total customer demand of R1S3 is 2800 kg, violating the capacity constraint.

Since the constraint violation of R1 is true, a sub-route adjustment is started. R1S1, which has a capacity of up to 2500 kg, only covers a demand of 2000 kg. Hence, 500 kg of demand can be transferred from another cluster to it. A customer whose demand does not exceed 500 kg in R1S2 is sought. Customer 4 has a demand of 500 kg. Therefore, customer 4 is transferred to R1S1. The adjusted R1S1 covers the locations of customers 1, 2 and 4 with a total demand of 2500 kg. The remaining customers are then re-clustered. The adjusted R1S2 covers customers 3, 5, 6 and 7 with a total demand of 2400 kg; the adjusted R1S3 covers customers 8, 9 and 10 with a total demand of 2000 kg. Now, as no sub-path in R1 violates the capacity constraints, the constraint violation of R1 becomes false. The final route of R1 is 0-1-2-4-0-3-5-6-7-0-8-9-10-0 (where 0 acts as a marker denoting the beginning or end of a sub-route), as illustrated in Figure 10.

Route R2 has three sub-routes: R2S1 covering customers 2, 3, 4 and 5 with a total demand of 2000 kg, R2S2 covering the customers 6 and 7 with a total demand of 1400 kg, R2S3 covering customers 8, 9, 10 and 1 with a demand of 3500 kg. The constraint violation of R2 is true because the demand of R2S3 is 3500 kg, violating the capacity constraint.

Therefore, a sub-route adjustment is performed on the route R2. R2S1 covers a demand of 2000 kg and can cover an additional 500 kg. Since R2S2 has no customer with demand $\leq$500 kg, the search continues in R2S3. In R2S3, customer 9 has a 500 kg demand that can be transferred to R2S1. Hence, after the adjustment, the adjusted R2S1 covers customers 2, 3, 4, 5 and 9 with a total demand of 2500 kg, and the remaining customers are re-clustered. After re-clustering, R2S2 still covers customers 6 and 7 with a demand of 1400 kg, and the adjusted R2S3 covers the customers 8, 10 and 1 with a demand of 3000 kg. The constraint violation of R2 is true because the demand of the adjusted R2S3 still violates the constraint. Therefore, another sub-route adjustment for R2 is started, but this time the goal is to search for customers to fill in R2S2 (which currently handles 1400 kg but can cover up to 2500 kg). In R2S3, customer 10 has a 300 kg demand, that can be transferred to R2S2. Hence, the adjusted R2S2 cover customers 6, 7 and 10 with a demand of 1700 kg, and the remaining customers are re-clustered. After re-clustering, the adjusted R2S3 covers the customers 8 and 1 with a demand of 2700 kg. That is, this re-clustering is not successful and the constraint violation of R2 is still true.

Route R3 has no sub-route violating the vehicle capacity constraint, so the constraint violation is false, and no sub-route needs to be adjusted. Route R3 is thus 0-3-4-5-6-0-7-8-9-0-10-1-2-0.

Step 3: Building *R-Set*

The top *s* routes from Step 2 are sorted in ascending order of their objective values, and keeped in a reference set, *R-Set*.

Step 4: Building *G-Set*

The top *s* routes that have the constraint violation = "false" are sorted in ascending order of their objective values, and keeped in *G-Set*. The best route is set to be *gBest*.

3.3. Path relinking for CVRP

We use path relinking to create a candidate solution from a pair of guiding and initial solutions. The detailed steps for the path relinking algorithm are shown in Algorithm 3 and described below:

Step 1: Generating pairs of guiding and initial solutions.

1) Building a roulette wheel

A roulette wheel is a common selection method in combinatorial optimization [26]. In this pairing step, a roulette wheel is built based on the Hamming distances between the best solution and the solutions in the bottom half of the ranked *R-Set*.

Algorithm 2. Sweep algorithm and the sub-route adjustment method.

```
Algorithm 2: SWEEP ALGORITHM AND THE SUB-ROUTE ADJUSTMENT METHOD
BEGIN
    Calculate the polar angle of every customer location;
    i = 1;
    FOR each route                                          /* Generating routes */
        Assign customer i to be the starting customer;
        Assign the remaining customers according to the ascending polar angle;
        Cluster customers according to the number of vehicles and the vehicle capacity constraints;
        IF the clustering is successful THEN
            Set the constraint violation = "false";
        ELSE
            Perform a sub-route adjustment;
            IF the sub-route adjustment is successful THEN
                Set the constraint violation = "false";
            ELSE
                Set the constraint violation = "true";
            END IF
        END IF
        Calculate the total distance of the route;
        i = i+1;
    END FOR
    Build R-Set with s best routes;
    Build G-Set with s best routes that have the constraint violation = "false";
END
```

Figure 11 shows an example of solutions in ranked *R-Set*, where R1 is the best solution and R4 is the worst solution. Figure 12 shows an example of Hamming distance determination between solutions R_1 and R_3—between R_1 and R_3, there are two index positions in which the elements (customers) are different, hence the Hamming distance between R_1 and R_3 is 2. Similarly, Figure 13 shows an example of Hamming distance determination between solutions R_1 and R_4—between R_1 and R_4, there are three index positions in which the customers are different, hence the Hamming distance between R_1 and R_4 is 3.

Figure 14 shows an example of a roulette wheel according to the Hamming distances between solutions R_1 and R_3 as well as between solutions R_1 and R_4. The chance of selecting R_3 is 40% (2/5), and the chance of selecting R4 is 60% (3/5). Given that a randomly selected real number r is between 0 and 1, R_3 will be selected if $0.0 < r \leq 0.40$, but R_4 will be selected if $0.40 < r \leq 1$.

2) Pairing guiding and initial solutions

Each solution in the top half of the ranked *R-Set* is assigned as a guiding solution. Then each guiding solution is paired with an initial solution selecting from the roulette wheel in 1).

Algorithm 3. Path relinking.

```
Algorithm 3: PATH RELINKING
BEGIN
    Build a roulette wheel based on Hamming distances between the best solution and the solutions in the worst half of the
    ranked R-Set;
    FOR each solution in the top half of the ranked R-Set          /* Pairing guiding and initial solutions */
        Assign the solution to be a guiding solution;
        Select an initial solution by using the roulette wheel;
        Pair the guiding and the selected initial solutions;
    END FOR
    FOR each pair of guiding and initial solutions
        vBest ← initial solution
        REPEAT
            Identify positions in which the customers are different in the guiding and initial solutions;
            FOR each identified position
                Generate a new solution by swapping the customers of the guiding and initial solutions in
                that position;
                Cluster customers of a new solution according to the number of vehicles and the vehicle
                capacity constraints;
                Calculate the total distance of a new solution;
            END FOR
            Determine the solution with the shortest distance;
            Initial solution ← the solution with the shortest distance;
            IF initial solution is better than vBest THEN
                vBest ← initial solution;
            END IF
            IF initial solution is better than the worst solution in G-Set AND
               the constraint violation = "false" THEN
                Replace the worst solution in G-Set with this solution;
            END IF
        UNTIL initial solution = guiding solution
    END FOR
    IF vBest is better than the worst solution in R-Set THEN
        Replace the worst solution in R-Set with vBest;
    END IF
END
```

Step 2: Generating a new solution from each pair of solutions from Step 1.

1) Set an initial solution to be *vBest*.
2) Identify positions in which the customers are different in the guiding and initial solutions.
3) Generate a new solution from the guiding and initial solutions based on the positions identified in 2).

a. Swap the customers of the two solutions in the identified positions, one by one. Then cluster customers of all newly generated solutions and calculate the objective values of the solutions. The one with the lowest objective value is selected.
b. Set the selected solution as the initial solution for the next iteration.
c. If the new initial solution is better than vBest, update vBest.
d. If the new initial solution has the constraint violation = "false," and is better than the worst solution in G-Set, G-Set will be updated with that solution
e. Perform steps a to d until the initial solution becomes a guiding solution.

4) Update the *R-Set* with *vBest* if it is better than the worst solution in *R-Set*.

3.4. Local searches for CVRP

In our hybrid algorithm, swap, insert, reverse and greedy search operations are used to locally search sub-routes of every vehicle in *G-Set*. Each time the local search operation produces a solution that is better than the worst in *R-Set*, the worst solution is replaced by the new solution. Similarly, if the new solution is better than the worst in *G-Set*, the worst solution is replaced by the new solution. The details of the four local searches are described below.

1) Swap operation

A swap operation randomly selects two positions, $\boldsymbol{i}$ and $\boldsymbol{j}$, in each sub-route and then swaps them. This swapping is repeated until the specified number of iterations is reached.

For example, as illustrated in Figure 15, route Rx has three sub-routes: RxS1, RxS2 and RxS3. RxS1 is 0-3-4-5-6-0. For the first round of swapping, positions 1 and 4 are randomly selected, which causes customers 3 and 6 to swap. After swapping, the updated RxS1 is 0-6-4-5-3-0.

2) Insert operation

An insert operation randomly selects two positions, ***i*** and ***j***, in each sub-route. Then it inserts the customer at position ***j*** into position ***i*** and moves all customers in positions between ***i*** and ***j-1*** to the right. This insertion is repeated until the specified number of iterations is reached.

For example, in Figure 16, RxS1 is 0-3-4-5-6-0. For the first round of insertion, the randomly selected positions are 1 and 4, then customer 6 at position 4 is inserted into position 1, and all customers between position 1 and 3 are moved to the right. After the insertion, the updated RxS1 is 0-6-3-4-5-0.

3) Reverse operation

A reverse operation randomly selects two positions, ***i*** and ***j***, in each sub-route and reverses the sequence of customers between ***i*** and ***j***. This is repeated until the specified number of iterations is reached.

For example, in Figure 17, RxS1 is 0-3-4-5-6-0 and the selected positions are 1 and 4, so all customers between positions 1 and 4 are reversed. After the reversal, the new RxS1 is 0-6-5-4-3-0.

4) Greedy search operation

A greedy search operation starts with the first customer of each sub-route. Then the next customer of the sub-route is selected based on the distances from the first. This is repeated until all customers in the sub-route have been placed.

For example, in Figure 18, RxS1 is 0-3-4-5-6-0. The greedy search operation starts with the first customer, which is customer 3. Then the customer in the same sub-route who is the nearest to customer 3 is selected. If the distances between customers 3 and 4, 3 and 5, and 3 and 6 are 9, 13, and 5 respectively, the shortest distance is 5. Then customer 6 is selected as the next customer in RxS1. This is repeated until all customers in RxS1 have been placed. After adjustment, the updated RxS1 is 0-3-6-5-4-0.

4. Experimental results

The datasets used in evaluating the performance of our MAS-SA-PR algorithm were the symmetric CVRP taken from Capacitated Vehicle Routing Problem Library (CVRPLIB) [27]. Our MAS-SA-PR was tested with three datasets: set A, set B, and set P. Sets A, B, and P consist of 27, 23, and 22 instances respectively. Its performance was compared to two sets of algorithms:

i) MAS – This comparison determined whether the hybrid MAS-SA-PR performed better than the individual MAS.
ii) Three state-of-the-art algorithms: LNS-ACO [28], RBMC-ABC [29] and CVRP-FA [19].

In the experiments, the parameters were initialized to the values shown in Table 1, where ρ is evaporation rate; τ is pheromone level; α and β are parameters that control the relative weight of the pheromone and the distance, respectively; lastly, *maxT* is the maximum number of iterations. The algorithm was run ten times on each instance. Each run stopped when the maximum number of iterations (*maxT*) was reached.

Instance names have the form *x-nxx-kx*, where *x* is a set name, *nxx* is the number of customers, and *kx* is the number of vehicles. The deviation from the best-known solution, *%Dev*, was calculated from the difference between the global best solution, *gBest*, and the best-known solution, BKS.

$$\%Dev = \frac{gBest - BKS}{BKS} \times 100 \tag{13}$$

4.1. Result comparison with MAS

Tables 2, 3, and 4 compare results of MAS-SA-PR and MAS on set A, set B, and set P respectively. Performance was measured in terms of the ability to find the shortest route and the computational time. These three tables report the best solution found, the deviation from the best-known solution, and the computational time. The better result between the two algorithms is shown in bold.

Tables 2, 3, and 4 shows that the hybrid MAS-SA-PR performed better than MAS individually. When the individual MAS obtained the best-known solution, MAS-SA-PR also obtained it. When the individual MAS did not obtain the best-known solution, MAS-SA-PR always obtained a better solution. However, MAS-SA-PR took, 1.25 to 26.22 times, more computational time than the individual MAS.

4.2. Result comparison with three state-of-the-art algorithms

In this comparison, performance was measured in terms of the ability to find the shortest route. Tables 5, 6, and 7 report the best solution found by each algorithm and the deviation from the best-known solution. The best result among the compared algorithms is shown in bold.

Results from 27 standard instances from set A are shown in Table 5. For 16 instances, the best solution from MAS-SA-PR was the same as the best-known solution. For 20 instances, the best solution from MAS-SA-PR was either better than or the same as the best solution from LNS-ACO. For 19 instances, the best solution from MAS-SA-PR was either better than or the same as the best solution from RBMC-ABC. For 22 instances, the best solution from MAS-SA-PR was either better than or the same as the best solution from CVRP-FA. *%Dev* for our MAS-SA-PR ranged from 0.00 to 1.71.

Similarly, results from 23 standard instances from set B are shown in Table 6. For 13 instances, the best solution from MAS-SA-PR was the same as the best-known solution. For 17 instances, the best solution from MAS-SA-PR was either better than or the same as the best solution from LNS-ACO. For 15 instances, the best solution from MAS-SA-PR was either better than or the same as the best solution from RBMC-ABC. For 15 instances, the best solution from MAS-SA-PR was either better than or the same as the best solution from CVRP-FA. *%Dev* for our MAS-SA-PR ranged from 0.00 to 1.67.

The computational results from 22 standard instances from set P was shown in Table 7. For one dataset, P-n55-k8, the best solution from the proposed algorithm was better than the previous BKS. For 14 instances, the best solution from MAS-SA-PR was the same as BKS. For 18 instances, the best solution from MAS-SA-PR was either better than or the same as the best solution from LNS-ACO. For 18 instances, the best solution from MAS-SA-PR was either better than or the same as the best solution from RBMC-ABC. For 17 instances, the best solution from MAS-SA-PR was either better than or the same as the best solution from CVRP-FA. The *%Dev* of the proposed MAS-SA-PR, ranging from -2.04 to 0.88.

5. Conclusions

We developed and evaluated a hybrid MAS-SA-PR algorithm for solving capacitated vehicle routing problems. A sweep algorithm (SA) was used to generate initial solutions and assign customers to vehicles. A modified ant system (MAS) was used to construct offspring from parents of the current generation. Path relinking (PR) was used to further improve a solution by a guiding from the elite solution. The performance of our algorithm was compared with that of the individual MAS and three state-of-the-art algorithms. Our algorithm was very effective with small-sized and medium-sized datasets—the deviations from the best-known solutions were all less than 1.71%. Even though it did not perform any better or worse with large-sized datasets than the other state-of-the-art algorithms, for one benchmark instance, it generated a better solution than the best-known solution. However, the complexity of our new method caused its computation time to be quite long. We will attempt to reduce this complexity in our future work, so that it will be more efficient.

Declarations

Author contribution statement

A. Thammano, P. Rungwachira: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Funding statement

This work was supported by King Mongkut's Institute of Technology Ladkrabang.

Data availability statement

Data associated with this study has been deposited at the Capacitated Vehicle Routing Problem Library (CVRPLIB).

Declaration of interests statement

The authors declare no conflict of interest.

Additional information

No additional information is available for this paper.

References

[1] Elshaer R., Awad H. "A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants" Comput. Ind. Eng., vol. 140, 2020.
[2] A.E.-S. Ezugwu, A.O. Adewumi, M.E. Frîncu, Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem, Expert Syst. Appl. 77 (2017) 189–210.
[3] P. Toth, D. Vigo, The Vehicle Routing Problem, Society for Industrial and Applied Mathematics, 2002.
[4] M. Haimovich, A.H.G. Rinnooy Kan, Bounds and heuristics for capacitated routing problem, Math. Oper. Res. 10 (4) (1985) 527–542.
[5] T. Asano, N. Katoh, H. Tamaki, T. Tokuyama, Covering points in the plane by k-tours: towards a polynomial time approximation scheme for general k, in: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, 1997, pp. 275–283.
[6] A. Das, C. Mathieu, A quasi-polynomial time approximation scheme for Euclidean capacitated vehicle routing, in: Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, 2010, pp. 390–403.
[7] Khachay M., Ogorodnikov Y., Khachay D., "Efficient approximation of the metric CVRP in spaces of fixed doubling dimension," J. Global Optim., vol. 80, 2021, pp. 679-710.
[8] A.E.-S. Ezugwu, A.O. Adewumi, Discrete symbiotic organisms search algorithm for travelling salesman problem, Expert Syst. Appl. 87 (2017) 70–78.
[9] Y. Zhong, J. Lin, L. Wang, H. Zhang, Hybrid discrete artificial bee colony algorithm with threshold acceptance criterion for traveling salesman problem, Inf. Sci. 421 (2017) 70–84.
[10] E. Liao, C. Liu, A hierarchical algorithm based on density peaks clustering and ant colony optimization for traveling salesman problem", IEEE Access 6 (2018) 38921–38933.
[11] E. Osaba, J. Del Ser, A. Sadollah, M.N. Bilbao, D. Camacho, A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem, Appl. Soft Comput. 71 (2018) 277–290.
[12] M. Marinaki, Y. Marinakis, A glowworm swarm optimization algorithm for the vehicle routing problem with stochastic demands", Expert Syst. Appl. 46 (2016) 145–163.
[13] R. Goel, R. Maini, A hybrid of ant colony and firefly algorithm (HAFA) for solving vehicle routing problems, J. Comp. Sci. 25 (2018) 28–37.
[14] P.C. Pop, L. Fuksz, A.H. Marc, C. Sabo, A novel two-level optimization approach for clustered vehicle routing problem, Comput. Ind. Eng. 115 (2018) 304–318.
[15] M. Yousefikhoshbakht, M. Sedighpour, A combination of sweep algorithm and elite ant colony optimization for solving the multiple traveling salesman problem, Proc. Roman. Acad. Series A 13 (4) (2012) 295–301.
[16] M.H. Chen, P.C. Chang, C.Y. Chiu, S.P. Annadurai, A hybrid two-stage sweep algorithm for capacitated vehicle routing problem" Proceeding of the 2015 International Conference on Control, Autom. Robot. (2015).
[17] E. Osaba, X.S. Yang, F. Diaz, E. Onieva, A.D. Masegosa, A. Perallos, A discrete firefly algorithm to solve a rich vehicle routing problem modeling a newspaper distribution system with recycling policy, Soft Comp. 21 (2017) 5295–5308.
[18] M.A. Hannan, M. Akhtar, R.A. Begum, H. Basri, A. Hussain, E. Scavino, Capacitated vehicle-routing problem model for scheduled solid waste collection and route optimization using PSO algorithm, Waste Manag. 71 (2018) 31–41.
[19] A.M. Altabeeb, A.M. Mohsen, A. Ghallab, An improved hybrid firefly algorithm for capacitated vehicle routing problem", Appl. Soft Comp. J. 84 (2019).
[20] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern.-Part B 26 (1) (1996) 29–41.
[21] T. Sützle, H.H. Hoos, Max-min ant system, Future Generat. Comput. Syst. 16 (8) (2000) 889–914.
[22] G. Zhao, W. Luo, R. Sun, C. Yin, A modified max-min ant system for vehicle routing problems, in: Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing, 2008.
[23] B.E. Gillett, L.R. Miller, A heuristic algorithm for the vehicle-dispatch problem, Oper. Res. 22 (1974) 205–451.
[24] S.C. Ho, M. Gendreau, Path relinking for the vehicle routing problem, J. Heuristics 12 (2006) 55–72.
[25] M.G.C. Resende, C.C. Ribeiro, GRASP with path-relinking: recent advances and applications, Metaheuristics: Progr. Real Prob. Solv. 32 (2005) 29–63.
[26] F. Yu, X. Fu, H. Li, G. Dong, Improved roulette wheel selection-based genetic algorithm for TSP, in: Proceedings of the 2016 International Conference on Network and Information Systems for Computers, 2016.
[27] Capacitated vehicle routing problem library (CVRPLIB) [Online] Available: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/.
[28] S. Akpinar, Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem, Expert Syst. Appl. 61 (2016) 28–38.
[29] K.K.H. Ng, C.K.M. Lee, S.Z. Zhang, K. Wu, W. Ho, A multiple colonies artificial bee colony algorithm for a capacitated vehicle routing problem and re-routing strategies under time-dependent traffic congestion, Comput. Ind. Eng. 109 (2017) 151–168.