A Hybrid BSO-ACS Algorithm for Vehicle Routing Problem with Time Windows on Road Networks

Mingde Liu*, Yang Shen†, Qi Zhao‡ and Yuhui Shi§

*†‡§Department of Computer Science and Engineering
Southern University of Science and Technology, Shenzhen, China 518055
Email: *l11849245@mail.sustech.edu.cn, †sheny3@mail.sustech.edu.cn, ‡zhaq@sustech.edu.cn, §shiyh@sustech.edu.cn

Abstract—The Vehicle Routing Problem with Time Windows (VRPTW) is NP-hard which has many real-world applications in logistics and transportation. The traditional VRPTW is defined on a complete graph with customers as nodes, but in the real-world, VRPTWs are more based on road networks. To better simulate the real-world scenarios, this paper studies the VRPTW on road networks. Most researchers solve the VRPTW on road networks by utilizing exact algorithms which can not deal with large size problem. In this paper, a hybrid BSO-ACS algorithm, which combines Brain Storm Optimization (BSO), Ant Colony System (ACS) and Local Search (LS), is proposed to solve the VRPTW on road networks. A set of instances based on the road network of southwest Shenzhen, China are generated as benchmark problems. The computational experiments demonstrate the effectiveness of the proposed algorithm.

Index Terms—vehicle routing problem, road network, brain storm optimization, ant colony system, local search

1. Introduction

The Vehicle Routing Problem (VRP) is an important combinatorial optimization problem that has attracted wide publicity for the last decades. It has many applications in transportation, logistics, and distribution. The VRP is a generic name given to a whole class of problems whose objective is to find optimal routes to service a given set of customers by a fleet of vehicles [1]. There are several variations of traditional VRP, such as Capacitated Vehicle Routing Problem (CVRP) [2], Vehicle Routing Problem with Pickup and Delivery (VRPPD) [3], Vehicle Routing Problem with Time Windows (VRPTW) [4], and Asymmetric cost matrix Vehicle Routing Problem (AVRP) [5], etc. Traditional VRPs are defined on a complete graph that only contains customers and a depot. The customers are usually connected with a straight line, and the distance between two customers is measured by the Euclidean distance. However, in real-world scenarios, VRPs often relies on road networks, which contain not only customers and a depot, but also nodes of the real road. Unlike traditional VRPs, customers are connected by road paths instead of a straight line. Recently, there are many researchers attempt to solve the VRP on road networks [6]–[10]. Most of them use exact algorithms, which can not solve large-scale problems in an acceptable time.

Solving the VRP on road networks includes two phases: path selection and vehicle routing [9]. In the path selection, the Dijkstra’s algorithm is usually used to find the path among customers. While in the vehicle routing, like traditional VRP algorithms, there are two main categories of approaches: exact algorithms and heuristic ones. Exact algorithms include: Column Generation (CG) [7], Mixed Integer Linear Programming (MILP) [9], Branch-and-Price (BP) [11], etc. These methods can get an optimal solution only on small-sized problems. With the increase in the size of the problem, the computational time of exact algorithms grows exponentially. To get a satisfying solution within an acceptable computational time for large-scale problems, heuristic algorithms are usually employed. VRP heuristic algorithms include two main categories, i.e., constructive heuristics and improvement heuristics. Constructive heuristics construct a feasible solution as much as possible. The most popular ones include saving algorithm [12], cluster-first route-second algorithm [13], nearest neighbor algorithm [14], etc. The improvement heuristics enhance the current solution by modifying solutions. It includes 2-opt [15], 3-opt [16], chain-exchange [17], cross and relocate [18], 1-exchange [19], etc. There are also many metaheuristics that are used to solve VRPs, like population based metaheuristics, Genetic Algorithms (GA) [20] and Ant Colony Systems (ACS) [21]. Other techniques are based on local search approaches, such as Tabu Search (TS) [6], Simulated Annealing (SA) [22], Adaptive Large Neighborhood Search (ALNS) [10], etc.

There are many comparison studies [23]–[25] that have analyzed the performance of different heuristic and metaheuristic algorithms for VRP. Their results showed that no single heuristic or metaheuristic could
be better than others at any time, but the hybridization of different algorithms can overcome the shortcoming of a single method. In this paper, a hybrid BSO-ACS algorithm is proposed to solve the VRPTW on road networks. At first, Dijkstra’s algorithm is employed in path selection to obtain the shortest path between two customers. Then we use a saving heuristic algorithm to initialize solutions. The core of the proposed algorithm is BSO, which contains three phases: 1) Clustering; 2) Generating new solutions; 3) Disrupting solutions. In phase 2), the ACS is performed to generate a new solution from parents like the combination operation in the memetic algorithm. In phase 3), a local search is used to improve individuals like local improvement in the memetic algorithm. There are some advantages in this hybrid algorithm: 1) The clustering operator of BSO can help ACS learn more similarity properties in one cluster, which accelerates the exploitation of ACS. 2) Since the ACS is easy to fall into local optimum by using deposited pheromones, a set of local search heuristics is used to disrupt the solution to avoid local optimum.

The main contributions of this paper are:

- A hybrid BSO-ACS algorithm is proposed, in which ACS is used to help BSO to combine advantages of solutions in one cluster, and local search is used to disrupt solutions to better explore the search space and avoid local optimum;
- 30 instances of VRPTW on real road networks from Shenzhen, China are adopted for experimental study, with the results demonstrate the better performance of the proposed algorithm than the compared approaches.

The remainder of this paper is organized as follows. Section 2 presents the definition of the problem. Section 3 firstly describes BSO, ACS, and Local Search which includes 2-opt, exchange and relocate neighborhoods, then proposes a hybrid BSO-ACS algorithm. Section 4 evaluates the performance of the proposed algorithm. Conclusions are provided in Section 5.

2. Problem description

2.1. Standard VRPTW

VRPTW is an important generalization of the VRP. It is a combinatorial optimization problem in logistics. The VRPTW can be formally defined as follows. Firstly, there is a directed complete graph \( G(V, E) \), where \( V = \{v_0, v_1, \ldots, v_n\} \) is the vertex set to represent the depot \( v_0 \) and customers \( \{v_1, v_2, \ldots, v_n\} \), and each vertices \( v_i \) located at coordinates \( (x_i, y_i) \); \( E = \{(v_i, v_j) | v_i, v_j \in V, i \neq j\} \) is the edge set, the edge is normally just a straight line between two vertices, and the cost \( cost_{ij} \) in edge \( (v_i, v_j) \) equals to the Euclidean distance between \( v_i \) and \( v_j \). The speed of the vehicle is often assumed to be one unit without loss of generality, so the time cost equals to the Euclidean distance between \( v_i \) and \( v_j \). Each customer has a demand \( q_i \), a time window \([e_i, l_i]\) and each vehicle has a capacity \( Q \). The vehicle should arrive at customer \( i \) within the time window \([e_i, l_i]\), and the vehicles’ capacity should be satisfied. The objective of VRPTW is to minimize the total distance cost of all vehicles.

2.2. VRPTW on road networks

The standard VRPTW assumes customers on a complete graph and sets the cost to euclidean distance. However, in the real world, most VRPTWs take place on road networks, and customers are connected by road paths instead of a straight line. Some researchers [7], [9], [11] focus on the VRPTW on road networks to be more suitable for practical application. The VRPTW on road networks can be formally defined as follows. First, there is a road network graph \( G(V, C, E) \), where \( V = \{v_0, v_1, \ldots, v_m\} \) is the vertex set to represent the depot \( v_0 \) and road nodes \( \{v_1, v_2, \ldots, v_m\} \), the customer set \( C = \{c_1, c_2, \ldots, c_n\} \) is the subset of the vertex set \( V \). \( E = \{(i, j)^p | i, j \in V, i \neq j, p = 1, \ldots, |E_{(i,j)}|\} \) is the path set representing the alternative paths between vertex \( i \) and \( j \). The distance between the two vertices is the path length, and the cost time equals the distance divided by vehicle’s speed. The mathematical model of VRPTW on road networks is defined as follows [11].

Parameters description:

\[
K \quad \text{the set of all vehicles} \\
V \quad \text{the set of all vertices} \\
C \quad \text{the set of all customers} \\
E \quad \text{the set of all paths} \\
N \quad \text{total number of customers} \\
Q \quad \text{maximum vehicle capacity} \\
cost_{ij} \quad \text{cost from vertex } i \text{ to vertex } j \\
t_{ij} \quad \text{travel time from vertex } i \text{ to vertex } j \\
q_i \quad \text{demand of customer } c_i \\
e_i \quad \text{earliest arrival of the time window of customer } c_i \\
l_i \quad \text{latest arrival of the time window of customer } c_i \\
s_i \quad \text{service time of customer } c_i \\
t_i \quad \text{arrival time of vertex } v_i \\
t_i' \quad \text{leave time of vertex } v_i \\
w_i \quad \text{wait time of customer } c_i
\]

Objective function:

\[
\min \sum_{k \in K} \sum_{(i,j)^p \in E} cost_{ij} x_{(i,j)^p,k} \\
\text{subject to:} \\
x_{(i,j)^p,k} = \begin{cases} 
1 & \text{if vehicle } k \text{ uses path } (i,j)^p \\
0 & \text{otherwise} 
\end{cases} \\
\sum_{i \in V} \sum_{p \in E_{(i,0)}} x_{(i,0)^p,k} = \sum_{j \in V} \sum_{p \in E_{(0,j)}} x_{(0,j)^p,k} = 1 \quad (\forall k \in K)
\]
3. Hybrid BSO-ACS Algorithm

This section describes the proposed algorithm for the VRPTW on road networks. Firstly we briefly introduce the BSO algorithm, local search heuristics used in the proposed algorithm, and the ACS algorithm. Then, a hybrid BSO-ACS algorithm is proposed to solve the VRPTW on road networks.

3.1. Brain Storm Optimization

Brain Storm Optimization (BSO) [26] is firstly proposed by Shi, which belongs to swarm intelligence algorithms [27]. It is based on the brainstorming process and has been used to solve traditional VRPs [28], [29]. The main idea of this algorithm is to divide solutions into clusters that the same cluster has similar properties. In the new solutions generation procedure, the cluster combines with each other to enhance the solution quality, while the disruption of one cluster’s solutions help the algorithm search more space to avoid stuck into local optimum. The procedure of BSO is shown in Fig. 1.

3.2. Local Search

Local search is a widely used method to solve VRPs. It starts from the current solution \( s \), and moves to another solution in its neighborhoods \( N(s) \). This section will introduce some classical neighborhood strategies.

\[
\sum_{k \in K} \sum_{i \in C} x_{(i,j)k} = 1 \quad (\forall j \in C) \quad (4)
\]

\[
\sum_{i \in C} q_i \sum_{j \in C, j \neq i} x_{(i,j)k} \leq Q \quad (\forall k \in K) \quad (5)
\]

\[
t_i' = t_i + w_i + s_i \quad (6)
\]

\[
w_j = \begin{cases} \max\{e_j - t_i' - t_{ij}, 0\} & \text{if } j \in C \\ 0 & \text{if } j \notin C \end{cases} \quad (\forall i, j \in V) \quad (7)
\]

\[
t_i + s_i + t_{ij} + w_i \leq t_j \quad (\forall i \in V, j \in C, i \neq j) \quad (8)
\]

\[
e_i \leq t_i + w_i \leq l_i \quad (\forall i \in C) \quad (9)
\]

The objective function (1) minimizes the total cost of all routes. Equation (2) shows a binary variable to represent whether a vehicle uses the path \((i, j)^p\). Equation (3) means the vehicle should start from and end in the depot. Equation (4) represents that each customers should only be serviced once by vehicles. Constraint (5) implies that the vehicle capacity should be satisfied. Constraint (8) is the travel time relationship between vertex \( v_i \) and its successor \( v_j \), and Inequality (9) means the vehicle should arrive at customer \( c_i \) within time window \([e_i, l_i]\).

3.2.1. 2-opt. 2-opt [15] is a widely used neighborhood approach first proposed by Croes for solving TSP (Traveling Salesman Problem). The main idea is to remove the intersecting edges in the routes and reverse the nodes between these edges. In VRPs, the 2-opt has two different operations for the single route or two routes, these two operations are shown in Fig. 2.

3.2.2. Exchange and Relocate. The exchange and relocated neighborhoods are firstly proposed by Savelsobergh [18]. In these neighborhoods, the sub-routes between routes may exchange and relocate, which are shown in Fig. 3.

3.3. Ant Colony System

Ant Colony Optimization (ACO) [30] is a swarm intelligence algorithm. It is inspired by the behavior of ants searching the shortest path to get food by pheromones. When searching food, firstly ants wander randomly and leave pheromones when a route is found. Shorter routes have more pheromones and are more attractive for ants to select since ants travel more frequently between food and nest.

When applying ACO to solve VRRs, ants select the next customer one by one from the depot until it can...
not select a feasible customer. Then, the ACO generates
a route and restarts from the depot to generate a new
route for another vehicle [31]. For an ant located in the
node of customer \(i\), its probability of selecting customer \(j\) is Eq. (10):

\[
p_{ij}^k = \begin{cases} \frac{(\tau_{ij}^\alpha)(\eta_{ij}^\beta)}{\sum_{k \in J_k(i)}(\tau_{ik}^\alpha)(\eta_{ik}^\beta)} & \text{if } j \in J_k(i) \\ 0 & \text{otherwise} \end{cases}
\]

where \(\tau_{ij}\) is the amount of pheromone deposited on
the path from customer \(i\) to \(j\), \(0 \leq \alpha\) is a parameter
which controls the relative influence of \(\tau_{ij}\), \(\eta_{ij}\) is the
desirability of the state transition (typically is \(1/d_{ij}\), i.e.,
the reciprocal of the distance between \(i\) and \(j\)), \(\beta \geq 1\)
is a parameter to control the influence of \(\eta_{ij}\), and \(J_k(i)\)
is a customer set which can be traveled by ant \(k\) from
customer \(i\).

When all ants have found their solutions, pheromones along the edges are updated by the
performance of solutions:

\[
\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \sum_k \Delta \tau_{ij}^k
\]

where \(0 \leq \rho \leq 1\) is the pheromone evaporation coeffi-
cient, and \(\Delta \tau_{ij}^k\) is the amount of pheromone deposited by ant \(k\), which is defined by

\[
\Delta \tau_{ij}^k = \begin{cases} 1/L_k & \text{if ant } k \text{ uses edge } (i, j) \text{ in its solution} \\ 0 & \text{otherwise} \end{cases}
\]

where \(L_k\) is the cost (usually distance cost) of the solution
found by ant \(k\).

Since ACO always uses the probability of \(p_{ij}\), it cannot
balance the exploration and exploitation. To han-
dles this issue, the Ant Colony System (ACS) [32] was proposed by Dorigo. Its state transition rule is defined

\[
s = \begin{cases} \arg \max_{j \in J_k(i)} \tau_{ij} \cdot \eta_{ij}^\beta & \text{if } q \leq q_0 \\ \frac{\tau_{ij}(\eta_{ij}^\beta)}{\sum_{k \in J_k(i)}(\tau_{ik}^\alpha)(\eta_{ik}^\beta)} & \text{otherwise} \end{cases}
\]

where \(0 \leq q \leq 1\) is a random number, \(0 \leq q_0\) is a
self-defined parameter which balance exploration and
exploitation.

ACS includes local and global pheromones. When a
solution is found by an ant, it is updated by the local
update Eq. (14) to refine the edges’ pheromone; when
a global best solution is found by ants, the \(\tau_{ij}\) of edge
\((i, j)\) which is used by the best solution will be updated
by Eq. (15).

\[
\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta \tau_{ij}
\]

\[
\tau_{ij} \leftarrow (1 - \alpha) \cdot \tau_{ij} + \alpha \cdot L^*_ij
\]

where \(\alpha\) is the pheromone evaporation rate, and \(L^*_ij\) is
the length of the best solution.

3.4. Proposed Hybrid BSO-ACS Algorithm

To solve VRPTWs on road networks, we propose
a hybrid BSO-ACS algorithm. In this algorithm, the
BSO clustering procedure can help the ACS combine
similarity properties which improves exploitation, and a
local search expands more search space to avoid stuck
into local optimal. In the proposed hybrid BSO-ACS
algorithm, we first use the Savings method [12] to gen-
erate initial solutions. In the clustering stage of BSO,
to make the process more efficient, we conduct the
clustering operation in the object space [33] with the
number of clusters set to two, i.e., better individuals,
“elitists” and worse individuals “normal”. In the new
solution generation stage, the ACS is used to generate
a new individual from two original clusters, and the local
search is used to generate a new individual from one
original cluster. More details are given in Alg. 1.
Algorithm 1 Hybrid BSO-ACS Algorithm

1: **Input:** $G(V,C,E)$
2: **Output:** $s_{best}$, best solution
3: use the saving method to initialize $n$ solutions
4: **while** not termination **do**
5: evaluate the solutions $S$ and sort solutions by fitness
6: divide solutions into two clusters by fitness, set the top $perc_e$ percentage as elitists and remaining as normals
7: **if** $rand(0,1) < p_e$ **then**
8: **if** $rand(0,1) < p_{one}$ **then**
9: randomly pick a solution $s_i$ from elitists
10: $s' \leftarrow$ local_search($s_i$)
11: **else**
12: randomly pick two solutions $s_i$ and $s_j$ from elitists
13: $s' \leftarrow$ ACS($s_i, s_j$)
14: **end if**
15: **else**
16: **if** $rand(0,1) < p_{one}$ **then**
17: randomly pick a solution $s_i$ from normals
18: $s' \leftarrow$ local_search($s_i$)
19: **else**
20: randomly pick two solutions $s_i$ and $s_j$ from normals
21: $s' \leftarrow$ ACS($s_i, s_j$)
22: **end if**
23: **if** $s'$ is better than the selected solution(s) $s$ **then**
24: $s \leftarrow s'$
25: **end if**
26: **end while**
27: return $s_{best}$ in solutions $S$

4. Experiments and Discussions

We use the road network in Nanshan district, southwest of Shenzhen, China from OpenStreetMap. The road network graph has 8214 nodes and 18558 road arcs. We generate 30 instances like the Solomon benchmark, which contains C1, C2, R1, R2, RC1, RC2, and all instances have 100 customers, respectively. Customers in “C” problems are clustered, and customers in “R” problems are random. In “RC” problems customers are partially clustered and partially randomly distributed. In problem sets 1 (i.e., R1, C1, and RC1), the scheduling horizon is short because of the low capacity of vehicles. In contrast, problem sets R2, C2 and RC2 have longer scheduling horizons. More instances can be obtained at https://github.com/lmingde/VRPTW_on_Road_Network_Benchmark.

To evaluate the performance of the proposed algorithm, we compare the proposed algorithm with ACS and ALNS in the total distances. The ALNS is a popular algorithm, and have shown its promise on various problems [10], [34]–[37].

4.1. Experiment Setup

The proposed algorithm is implemented in the Python programming language, and all experiments are conducted on an Intel Xeon E5-2650 CPU@2.30GHz PC with 16GB RAM.

There are some parameters in the hybrid algorithm. The parameters of BSO are $perc_e$, $p_e$, $p_{one}$. A small $perc_e$ value with a large $p_e$ value, conduct the algorithm to search in the neighborhood of elitists. This operation can facilitate exploitation, but may result in falling into local optimum. The $p_{one}$ is used to control the strength of disruption, which enhances the exploration. The parameters of ACS are $M$, $\beta$, $p$, $q_0$. Large $\beta$ and $q_0$ indicate greed strategy. When $p$ is small, the exploration of the algorithm will be reduced. When $p$ is a large value, the exploitation of the algorithm will be enhanced. The parameter setting is given in Table 1. All parameters are selected by tuning to balance the trade-off between computational cost and solution quality.

<table>
<thead>
<tr>
<th>parameter</th>
<th>value</th>
<th>parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>20</td>
<td>$perc_e$</td>
<td>0.1</td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$p_e$</td>
<td>0.2</td>
</tr>
<tr>
<td>$q_0$</td>
<td>0.1</td>
<td>$p_{one}$</td>
<td>0.6</td>
</tr>
<tr>
<td>$\rho$</td>
<td>0.1</td>
<td>max_iter</td>
<td>150</td>
</tr>
</tbody>
</table>

4.2. Result Analysis

The comparison results for all instances are given in Table 2, where “NV” represents the number of the used vehicles, “TD” represents the total distances. We use the gap metric to represent the cost reduction between our method and others, the gap is computed according to Eq. (16).

$$Gap = \frac{TD_{ours} - TD_{others}}{TD_{ours}}$$

As it can be observed from Table 2, the average number of vehicles obtained by the proposed algorithm is 10.33, which is better than ACS (10.83) and ALNS (10.37). The average total distance obtained by the proposed algorithm is 223648.04, which is superior to the ACS (249310.02) and ALNS (228705.44). For all instances, the total distance obtained by the hybrid BSO-ACS algorithm is smaller than ACS, and the gap ranges from -18.36% to -2.23%. The total distances obtained by the hybrid BSO-ACS algorithm are reduced by -9.13% to 1.13% comparing to ALNS. Counting the number of the best solutions in the three methods, the hybrid-BSO-ACS gets best in 25 out of 30 cases, while the ALNS obtains the best in the remaining five instances. There are some results which attained by hybrid algorithm are shown in Fig. 4, where the black point is the depot and the grey points are customers.
Table 3 summarizes the average cost and gap in each problem set. The experiments were run 10 times in each instance. In Table 3, compared with ACS, the hybrid BSO-ACS obtains solution results smaller -3.68%, -12.49%, -12.88%, -13.94%, -9.57% and -15.17% for C1, C2, R1, R2, RC1 and RC2. Gaps between hybrid BSO-ACS and ALNS are -0.91%, -3.77%, -2.24%, -4.06%, -0.52% and -3.39% for C1, C2, R1, R2, RC1, and RC2. It can be observed that gaps in C2, R2, and RC2 are bigger than that in C1, R1, and RC1. The reason is that the scheduling horizon is longer in C2, R2, RC2, i.e., the average length of routes are longer since the capacity of vehicles is larger. Local search in the proposed hybrid BSO-ACS algorithm generates a new solution from one single solution that is effective to deal with such long route cases.

5. Conclusions

This paper proposed a hybrid BSO-ACS algorithm to solve the VRPTW on road networks. In the proposed algorithm, the BSO uses Local Search and ACS to generate new solutions, and the solutions are improved iteratively. 30 instances based on the road network on the southwest of Shenzhen, China with 100 customers are generated. Among 30 instances, the proposed algorithm achieved 25 out of 30 better solutions than classic ACS and ALNS algorithm. The experimental results demonstrated the effectiveness of the proposed algorithm.

Acknowledgment

This work is partially supported by National Science Foundation of China under grant number
References


