https://doi.org/10.26552/com.C.2009.4.38-42



Andrej Chu *

# ANT COLONY OPTIMIZATION METHOD AND SPLIT-DELIVERY VEHICLE ROUTING PROBLEM

*This paper deals with a split delivery vehicle routing problem, which is a modification of a vehicle routing problem. It consists in delivery routes optimization in communications network containing initial city of all routes and a given number of places, which is necessary to include in delivery routes, where a customer can be served by more than one vehicle. The objective is to find a set of vehicle routes that serve all the customers and the total distance traveled is minimized. The split delivery vehicle routing problem is NP hard, therefore we present a solution approach by three heuristics, and a metaheuristics called Ant colony optimization (ACO).*

*Keywords: split delivery vehicle routing problem, integer programming, ant colony optimization, metaheuristics*

## 1. Introduction

Vehicle routing problem (VRP) is a classical problem in operations research. It consists in delivery routes optimization in communications network containing depot of all routes and a given number of cities, which is necessary to include in delivery routes. In VRP a set of customers needs to be served and a fleet of capacitated vehicles is available to do so. The objective is the minimization of costs, which usually means minimizing the total distance traveled. In most VRPs it is assumed that the demand of a customer is given and is less than or equal to the capacity of a vehicle and that each customer has to be served by exactly one vehicle, i.e., there is a single-visit assumption. The condition is that the sum of demands of the cities on the route should be less than or equal to the capacity of vehicle. The vehicle capacity is the limitary factor in this problem.

It is obvious that when a customer's demand exceeds the vehicle capacity it is necessary to visit that customer more than once. Even when all customer demands are less than or equal to the vehicle capacity, it may be beneficial to use more than one vehicle to serve a customer. In the split delivery vehicle routing problem (SDVRP) the single-visit assumption is relaxed and each customer may be served by more than one vehicle.

The SDVRP is, similarly like the traveling salesman problem (which is reduced on this vehicle routing problem in the case of big enough capacity of a vehicle $V$), an NP hard problem.

Let us set the mathematical model SDVRP, which is based on Miller-Tucker-Zemlin formulation of the traveling salesman problem.

The binary variable $x_{ij}^k$ equals 1, if the edge $(i,j)$ is included in the solution, so it means that the vehicle goes from the city $i$ to the city $j$, otherwise this variable's value is equal to zero. Each customer has a demand $q_i$, which can be less than, equal, or greater than the vehicle capacity $V$. The variable $q_i^k$ represents the quantity delivered to the $i$-th customer on k-th the route.

The distance between the cities $i$ and $j$ is $c_{ij}$. Each vehicle has capacity $V$ and has to start and finish its tour at the depot. A customer may be visited more than once.

The objective function (1) represents the sum of all edges distances in the solution, hence the sum of all routes length of the solution. The equation (2) assures that the $k$-th route leaves the depot just once. The equation (3) assures that only one edge comes out from city $i$ on the $k$-th route. Similarly, the (4) sets that number of edges coming from the city $j$ on the $k$-th route is the same as number of edges coming into the city (0 or 1). The equations (3) and (4) are not applied on the city 1, because from and into the city 1 there are as many edges as many routes there are. The inequality (5) defines variable $u_i$, which represents the demand on the route $k$ from the city 1 to the city $i$. This condition also has the anti-cycling effect – it prevents creating the sub-cycles in the solution. The condition (7) means that demand on the route $k$ does not exceed the vehicle capacity $V$. Constraints (8) ensure that the demand $q_i$ of customer $i$ is completely satisfied. Constraints (9) impose that a delivery to a customer $i$ on route $k$ can only take place if route $k$ is selected and the total quantity delivered on a selected route cannot exceed the vehicle capacity $V$.

The SDVRP model is (1)–(11):

* **Andrej Chu**
Faculty of Informatics and Statistics, University of Economics, Czech Republic, E-mail: andrej.chu@vse.cz

$$\min \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}^{k} \quad (1)$$

$$\sum_{j=2}^{n} x_{1j}^{k} \le 1,\ k = 1, 2, ..., K \quad (2)$$

$$\sum_{j=2}^{n} x_{ij}^{k} \le 1,\ i = 2, ..., n;\ k = 1, 2, ..., K \quad (3)$$

$$\sum_{i=1}^{n} x_{ij}^{k} = \sum_{i=1}^{n} x_{ji}^{k},\ j = 2, 3, ..., n;\ k = 1, 2, ..., K \quad (4)$$

$$u_i^k + q_j^k - V\left(1 - x_{ij}^k\right) \le u_j^k,\ i = 1, 2, ..., n;$$

$$j = 2, 3, ..., n;\ i \ne j;\ k = 1, 2, ..., K \quad (5)$$

$$u_1^k = 0,\ k = 1, 2, ..., K \quad (6)$$

$$q_i^k \le u_i^k \le V,\ i = 2, 3, ..., n;\ k = 1, 2, ..., K \quad (7)$$

$$\sum_{k=1}^{K} q_i^k = q_i,\ i = 2, 3, ..., n \quad (8)$$

$$0 \le q_i^k \le q_i \sum_{j=1}^{n} x_{ij}^{k},\ i = 2, 3, ..., n;\ k = 1, 2, ..., K \quad (9)$$

$$x_{ii}^{k} = 0,\ i = 1, 2, ..., n;\ k = 1, 2, ..., K \quad (10)$$

$$x_{ij}^{k} \in \{0, 1\},\ i, j = 1, 2, ..., n;\ k = 1, 2, ..., K \quad (11)$$

## 2. Heuristic methods

As the split delivery vehicle routing problem is NP hard, for the large problem the solution of the model cannot be obtained for acceptable computer time consumption. It will be useful to propose the heuristic methods for the large scale problem solution. The modified heuristic methods for the traveling salesman problem respective the vehicle routing problem will be used. There are the following heuristics: nearest neighborhood method, insert method and savings method. All the methods are illustrated on a numerical experiment. The differences of the results obtained by those methods are shown on the case study.

It is assumed that the distance matrix $C$ is symmetric, not negative. Let us denote $M$ the set of cities that was not included in any route. In the beginning of the method the set $M$ equals {2, 3, ..., $n$}. The heuristic method ends when the set $M$ is empty. The route will be denoted as $tr = (tr(1), tr(2), ..., tr(m))$, where $tr(1) = tr(m) = 1$.

### 2.1 The nearest neighborhood method

By this method the following steps are executed until the set M is empty:

Step 1. Let us denote the city with the lowest distance $c_{1i}$ as k and let it form the route $tr(1) = 1$, $tr(2) = k$, $tr(3) = 1$, let it set $m = 3$. If $q_k \le V$, delete the city $k$ from the set M and set $\overline{V} = V - q_k$, and continue to step 2. Otherwise the city is not deleted from the set M, and city has the demand $p_k = q_k - V$, the route is closed and the method continues by the step 1 (starting the new route).

Step 2. If the set M is empty, the method stops. Let us find the city $k$ from M which minimizes the distance $c_{tr(m-1),k}$ and for which the route enlargement $tr$ by the city $k$ by inserting it after the city $tr(m-1)$ satisfies the condition $q_k \le \overline{V}$. If $q_k \le \overline{V}$, the route is closed and the method follows by the step 1 (starting the new route), otherwise step 3.

Step 3. Let us enlarge the route $tr$ by the city $k$ by inserting this city after the city $tr(m-1)$, then increment $m$ by 1. If $q_k \le \overline{V}$ delete the city $k$ from the set M, set free capacity $\overline{V} = \overline{V} - q_k$ and continue by step 2, otherwise set $q_k = q_k - \overline{V}$ and continue to step 1.

### 2.2 The insertion method

The following steps are done until the set $M$ is not empty.

Step 1. Let us denote the city k with the highest distance $c_{1i}$ and put the route $tr(1) = 1$, $tr(2) = k$, $tr(3) = 1$, and $m = 3$. The city $k$ is deleted from the set $M$. If the set $M$ is empty the method ends.

Step 2. Find city $k$ from the set $M$ by following these conditions:

- it minimizes number $d = c_{tr(i),k} + c_{k,tr(i+1)} - c_{tr(i),tr(i+1)}$ for all $i = 1, 2, ..., m-1$ and $k \in M$,
- the route is enlarged by the city $k$ by inserting this city between the city $tr(i)$ and $tr(i+1)$, where $i$ minimizes the value $d$. If $q_k \le V$, the city $k$ is deleted from the set M and $\overline{V} = \overline{V} - q_k$, go to the step 2. Otherwise the city is not deleted from the set M, and city has the demand $q_k = q_k - V$, the route is closed and the method continues by the step 1 (starting the new route).
- enlarge the route $tr$ by the city $k$ by inserting this city between the city $tr(i)$ and $tr(i+1)$, increase $m$ by 1. If the set M is empty the method ends, otherwise it follows the step 2.

### 2.3 The savings methods

The following steps are executed until the set M is empty:

Step 1. If the set M contains only the city $k$, the route $tr(1) = 1$, $tr(2) = k$, $tr(3) = 1$ is formed, let $m = 3$. If $q_k \le V$ then the $k$ will be deleted from the set M and the method ends. If not, $k$ is not deleted from $M$, $q_k = q_k - V$, repeat step 1.

Step 2. Let us find the pair of cities from M in the form $(k,l)$, that maximizes savings $s_{ij} = c_{1i} + c_{j1} - c_{ij}$. If this pair $(k,l)$ does not exist, form $(1,k,1)$, $(1,l,1)$, stop.

Step 3. Put the route $tr(1) = 1$, $tr(2) = k$, $tr(3) = i$, $tr(4) = 1$, $m = 4$. If $q_k + q_k - V$, then $k$ and $l$ will be deleted from M and $V' = V - (q_k + q_l)$, step 4.

Step 4. Find $i$ from the set M that maximizes $s_{ik}$, or $j$ from M that maximizes $s_{ij}$. If $s_{ik} \geq s_{ij}$ (or $j$ does not exist), the city $i$ is inserted into the route before the city $k$ and if $q_i > V'$ then the city $i$ is deleted from the set M. Let s be increased by one and $k$ equal $i$. If , the city $i$ is not deleted from M and $q_i = q_i - V'$. The route is closed.

Find $j$ that maximizes $s_{ij}$ when inserting $j$ into the route after the city $i$. If $s_{ik} \geq s_{ij}$ (or $i$ does not exist), the city $j$ is inserted into route after the city $i$ and if $q_j > V'$, city $j$ is deleted from M. Let $m$ be increased by one. If $q_j > V'$, the city $i$ is not deleted from M and $p_j = q_j - V'$. The route is closed.

If neither $i$ nor $j$ does exist, the route ends and method continues by the step 1.

## 2.4 ACO metaheuristics

Ant colony optimization (ACO) is a metaheuristics which has been invented in quite recent time. The idea was first introduced by Marco Dorigo 15 years ago. Since then it has been applied on many known problems and significant progress has been made in optimizing this metaheuristics by improving its attributes, finding right parameters values and especially combining it with another metaheuristics, e. g. local search (Dorigo & Schützle, 2004).

The main idea of this metaheuristics lies in simulating the behavior of natural systems, in this case ant colonies. The ants usually search for the food for survival by exploring their environment. The most important part of this process is pheromone deposition, which every ant does when he moves outside the anthill. The more ants move on one path, the more pheromones they deposit and attract other ants to use the path. They also deposit pheromones when moving from the food source to the anthill, so the richer the food source is, the more ants are attracted to it. This is achieved also by depositing more pheromone when returning from richer food sources.

This idea has been transformed into ant colony optimization by transforming the food-search problem into mathematical problems, mostly represented by graphs, and by defining artificial ants. Artificial ants are quite similar to the real ants. The difference is that they usually move on the edges of the graphs and construct feasible solutions of a problem. They are also often enhanced by memory and other features needed for the solution construction. When constructing the solution they follow the pheromone and heuristic information.

The main framework of the algorithm is as follows:

0. Initialization – computing the initial pheromone and heuristic information.
1. Solution construction – $m$ solutions are built by $m$ ants.
2. Pheromone evaporation – pheromone laid on the paths evaporates every iteration with specified rate.
3. Pheromone deposition – ants with w best solutions deposit pheromone on their paths.

Steps 1-3 are repeated until the stop condition is fulfilled. The stop condition can be set as a fixed number of iterations or as the solution improvement no longer occurs after a predefined number of iterations.

Our paper reports about the attempt to apply the ACO metaheuristics on SDVRP, which has been never done before.

Solution construction is done by an artificial ant by sequential selection of the nodes and collection of the goods in the graph until its capacity is full. Thus, it creates the node strings representing the solution. To accomplish this, we need first to define the heuristic information $\eta_{ij}$:

$$\eta_{ij} = d_{i1} + d_{1j} - d_{ij} \text{ for } i, j \in M; i \neq j \tag{11}$$

This heuristic information is the same as the one used in the C-W savings method introduced before. We also need the pheromone information $\tau_{ij}$, which is initially set to a value corresponding approximately to the value of pheromone, which will be laid by the ants in one iteration (see further). In every node $i$, the ant computes the probability of the next move for each node:

$$p_{ij} = \frac{\left[\tau_{ij}\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum_{k \in N'}\left[\tau_{ij}\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}} \text{ for } j \in M' \tag{12}$$

where $M'$ is a set of the nodes with unsatisfied demands in this solution so far. Parameters $\alpha$ and $\beta$ were set to 1 and 2 respectively, as they had given the best experiments results. After the probability computation the ant randomly chooses (with respect to the given probabilities) the next node. This procedure is repeated until all demands are satisfied.

After the solution is constructed, pheromone evaporation takes place. Pheromone evaporation greatly helps with the speed of solution convergence. It is described by the following formula:

$$\tau_{ij} = (1 - \rho).\tau_{ij} \text{ for } i, j \in M; i \neq j \tag{13}$$

where $\rho$ is the pheromone evaporation rate and has been set to 0.01 in the experiments (1% of the pheromone evaporates every iteration).

The last step of an iteration is pheromone deposition. The ant lays pheromone on every edge its solution consists of. The amount of pheromone laid (added to each edge) is computed as

$$\Delta\tau = 1/T \tag{14}$$

where $T$ is the total length of the constructed solution. This allows the ants with better solutions to lay more pheromone than those with worse solutions.

### 2.4.1 Computational complexity analysis

Computational complexity of the algorithm implementing metaheuristic ant colony optimization method can be expressed from the computational complexities of the partial steps.

When constructing the solution, each ant in certain node has to make decision for the move to the next node. Considering $n$ as a number of nodes, this activity requires computing the probability for each node by (12). This requires $n$ steps and the whole activity is repeated as many times as many vertex walks there are in the graph. The number of these walks depends on an instance's values and may vary from $n$ to the sum of all demands of all customers (that is the case when the vehicle's capacity $V$ is set to 1 which is highly unlikely). In real data instances it is possible to presume that there will be no need to make more than $2n$ walks, which means that the number of split deliveries will not exceed double of the number of nodes (that is when each node is split once in average). With this presumption we can expect that the number of operations will be a quadratic function with $n$ as argument. That means that this step will run with quadratic amortized computational complexity. The asymptotic computational complexity, however, is pseudopolynomial and depends on the value of all demands and vehicle's capacity.

Pheromone laying requires walking over the found solution (or the path rings of the solution) and updating the pheromone values of the edges included in the solution. Using the same presumptions as in the previous paragraph, we can except this operation to require number of steps from $n$ to sum of all demands of all customers. Therefore, we can assume this operation to have a linear amortized computational complexity, but again, the asymptotic complexity is pseudopolynomial.

All in all, the single iteration of one ant takes $O(n^2)$ operations in average. Having m ants, we have the computational complexity of single iteration of the whole colony $O(m.n^2)$.

Pheromone evaporations takes $n^2$ steps but it is conducted only once per a colony's iteration, so it has negligible effect on the overall complexity and so has the initialization phase of the algorithm.

## 3. Numerical experiments

Let us have 20 cities from Slovakia and Czech Republic. Bratislava (Slovakia) is a depot of all routes. The minimal distances between the cities or between the cities and the depot are known and the transportation is done in the public road network. The modified heuristic methods were applied on 10 studies with different sets of cities. The demands of all customers are given by the size of the cities and they are the same for each study. In the first experiment the capacity of a vehicle is constant.

For the ACO metaheuristics the number of ants was set to 100, but only the half of them (those with the best solutions) in each iteration were allowed to lay the pheromone. The number of iteration was fixed to 1000. These settings were found as best in the number of experiments.

After application of modified heuristic methods on different studies of SDVRP we found out that we can achieve the best result by the savings method, when considering only the heuristic methods. The nearest neighborhood method is not suitable for SDVRP, see the result table 1.

As for the ACO metaheuristics, we can see that even though it uses the same heuristic information as the savings method does, it generates better solutions in most of the input examples. In addition, there is a lot of room for improving this implementation, while it does not use any other optimization of the generated solutions, which may include local search metaheuristics or improving the strings by removing unnecessary splits at the end of some strings. Hence, the ACO metaheuristics seems to be really a promising method not only for the SDVRP to solve. It has a potential to become very efficient method to solve NP-hard problems and has good possibilities to be extended as a distributed algorithm for parallel systems.

Table 1

| | Number of cities | routes | Nearest neighb. method (km) | Savings method (km) | Insertion method (km) | ACO method (km) |
|---|---|---|---|---|---|---|
| 1 | 11 | 4 | 2555 | 1944 | 2245 | 1962 |
| 2 | 10 | 4 | 3063 | 2520 | 2457 | 2427 |
| 3 | 13 | 5 | 3804 | 3596 | 3471 | 3559 |
| 4 | 11 | 4 | 2688 | 2398 | 2419 | 2402 |
| 5 | 15 | 5 | 3463 | 3199 | 3150 | 3044 |
| 6 | 14 | 6 | 4521 | 3558 | 3865 | 3531 |
| 7 | 17 | 6 | 4123 | 3746 | 3911 | 3750 |
| 8 | 18 | 7 | 5163 | 4058 | 4476 | 4040 |
| 9 | 19 | 7 | 4692 | 4320 | 4538 | 4236 |
| 10 | 20 | 7 | 4667 | 4456 | 4725 | 4417 |

In the second experiment the capacity of a vehicle is not constant. We studied dependence of a total traveled distance on the capacity of a vehicle. We applied the three modified heuristic methods on more than 4000 SDVRPs with a different capacity of a vehicle and we marked down their dependence between the total traveled distance and the capacity of vehicle. The representative graph 1 represents application of a modified savings method on the SDVRP. This analysis can be used operatively, when the decision of the used vehicles capacity is to be taken.

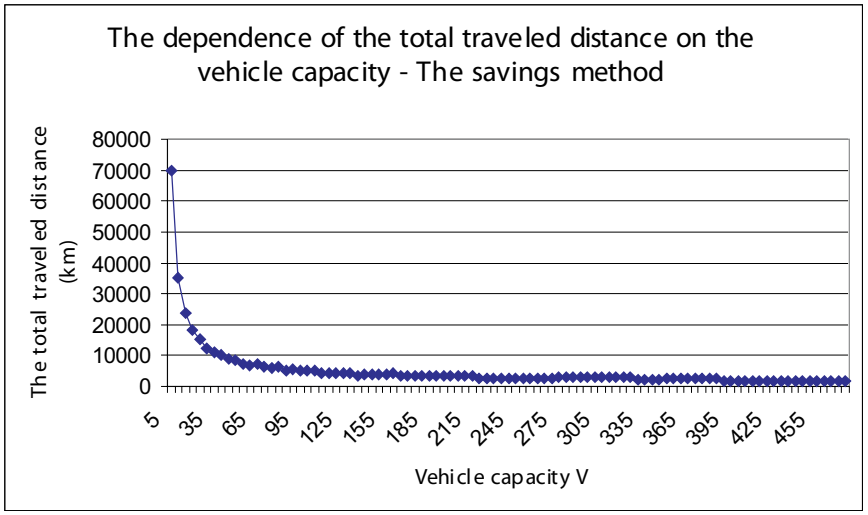


*Fig. 1 Results of the second experiment*

## References


[1] BODIN, L., GOLDEN, B.: *Classification in Vehicle Routing and Scheduling.* Networks, Vol. 11, 1981
[2] DANTZIG, G.B.; RAMSER, J. H.: *The Truck Dispatching Problem.* Management Science 6 (1): 80–91, 1959
[3] DORIGO, M., SCHÜTZLE, T.: *Ant Colony Optimization.* MIT Press, 2004, ISBN 0-262-04219-3
[4] DROR, M., TRUDEAU, P.: *Split Delivery Routing.* Naval Reasearch Logistics, 1990
[5] EVANS J. R., MINIEKA E.: *Optimization Algorithms for Networks and Graphs.* Marcel Dekker, Inc., New York, 1992
[6] FABRY, J.: *Dynamic Round and Cartage Problems (in Czech),* Dissertation thesis. Praha: VSE-FIS, 2006
[7] GUTIN, G., PUNNEN, A. P.: *The traveling salesman problem and its variations.* Kluver 2002. ISBN 1-4020-0664-0
[8] JANACEK, J.: *Mathematical Programming (in Czech),* EDIS ZU, Zilina, 1999, ISBN 80-7100-573-8
[9] JANACEK, J.: *Optimalization in Transport Networks (in Czech)*, EDIS ZU, Zilina, 2003, ISBN 80-8070-031-1
[10] PELIKAN, J.: *Discreet Models in Operating Research (in Czech),* Professional Publishing, 2001, ISBN 80-86419-17-7.