Solution to CVRP problem with the Ant Colony optimization algorithm

Przemyslaw Chojecki
STUDENT NO. 298814

2022.03.14

Conspectus submitted for course
Artificial Intelligence Methods 2 at
The Faculty of Mathematics and Information Science,
Warsaw University of Technology

Contents

1 Context of this conspectus **2**
1.1 This paper 2

2 Introduction to the problem **3**
2.1 Simulations can only find the approximate solutiontion 3
2.2 Ant Colony 4
2.3 Model of CVRP 4

3 Related work **5**

4 Proposed solution - Ant Colony algorithm **6**
4.1 Overview 6
4.2 Probability of getting to the next node 6
4.3 Distribution of pheromone 8

5 Modifications **9**
5.1 Reduce graph 10
5.2 Divide graph 10

6 Hypotheses **12**

7 Methodology **13**

8 Schedule **14**

9 Softwere **15**

1 Context of this conspectus

This document is the conspectus of the work for the project in class **Artificial Intelligence Methods 2**.

The project will consist of the conspectus, the solution's code, and the report.

The code developed for the project will be developed and later available at GitHub [6].

1.1 This paper

In this conspectus, we will briefly introduce the work and propose the solution for the problem. We will reference and review the literature that references the problem. We will state the hypotheses for the project and explain the method that we will use to verify the hypotheses. Also, we will state the schedule for the project and list the software that we will use to accomplish our goal.

2 Introduction to the problem

The Capacitated Vehicle Routing Problem (CVRP) is widely encountered in transportation. Many companies need to find the optimal solution because transporting goods is a significant part of their costs.

The CVRP is a modificarion of VRP. According to [1]: "Capacitated Vehicle Routing Problem (CVRP) is a real-life constrain satisfaction problem in which customers are optimally assigned to individual vehicles (considering their capacity) to keep total travel distance of the vehicles as minimum as possible while serving customers".

The problem is to find the optimal path for every vehicle so that every customer will receive the delivery. Every day the company has deliveries to make and drivers to make those deliveries. The company needs to split the deliveries for the day between deliverers with the smallest cost possible.

2.1 Simulations can only find the approximate solutiontion

Every company can have its way of understanding the cost of the deliveries in a single day. It is hard to state it with a single mathematical function. Most of the authors of academic work about the problem are trying to find the optimal way of delivering the goods according to some simple formulae. One such formula can be the length of the route. Unfortunately, the reality is a lot more complicated, and the solution should also consider the possibility that one driver can go sick and not make deliveries that day.

Another thing that is impossible to take into account is the human factor. For example, the software shows the following location for the driver and the optimal path to this place. However, last week the driver has heard that on this specific roadway, there was an accident. Therefore he decides to use another road instead. If the software knew the driver's preferences, it would output a different solution.

Many random factors are hard to state and take into account. One cannot reasonably expect to find the actual best solution for the company. However, it is sensible to expect software to find decent solutions and show them to the person in charge. Let him decide which one will be used for the day of deliveries.

2.2 Ant Colony

As it was already stated, the problem is impossible to simulate precisely. However, it is pretty easy to simulate the model of the problem with great precision. Finding the optimal solution to the model problem is still computationally expensive. In [8] it is stated that the problem is NP-hard. However, one can successfully use Ant Colony algorithms to find the approximation of the solution.

Ant Colony algorithms are known to rarely find the optimal solution in reasonable times but also can find a decent solution in a reasonable time. This is great, especially in this setup. The actual optimal solution to the model will most likely not be optimal in the real world but will be close. Therefore the sub-optimality of the found solution is not a problem in this setting.

2.3 Model of CVRP

The CVRP is a model of the real problem. This model can be explained in those points:

- In CVRP, we consider a graph of nodes. A node represents a warehouse or a client. There is a single warehouse, and the rest of the nodes are clients.
- The graph is complete because a supplier can always get from one client to another. The edges have weights according to the distance between nodes.
- A single shipping of one supplier is a cycle in this graph that starts at the warehouse and ends at it (the supplier, after all deliveries, has to come back to the warehouse).
- Also, every client has his order. In the model, this is represented as a single number - property of a node. Those numbers are parameters of the dataset.
- A single cycle cannot be longer than s_{max} which is also a parameter of the dataset. (This is an additional restriction that is not typically referred to as a part of CVRP)
- After one cycle is finished, the supplier will start the next cycle. All the restrictions are restarted.
- The problem aims to find such a set of cycles that minimalize the whole length of the tour.

3 Related work

This chapter will name a few works that attempted to tackle the CVRP with the Ant Colony approach.

In [1] the main focus of this work is to modify different methods (called Sweep algorithm). Also, the authors compared the performance of a bunch of algorithms on solving the CVRP. The method described here starts with splitting the clients into groups and then solving every group as the standard TSP (Traveling Salesman Problem).

In [7] authors show the Ant Colony algorithms finds better solutions to CVRP than the simple heuristics approaches. Furthermore, they showed that one could enrich the Ant Colony algorithm with additional heuristics to obtain better results.

In [3] authors considered a modification of the problem where the track has to at first get into one of the warehouses, pick the goods from it and then deliver those to clients.

In [4] authors tried to improve the performance of Ant Colony by improving the communication between ants. In standard Ant Colony algorithms, the ants only communicate by their pheromone. Here, the authors tried to add the direct communication between ants.

In [2] authors considered some additional heuristics. One of them can be described like this: when routes for two cars are in the relationship that their routes are crossing at one point, they can exchange the deliveries that led them into the crossing. This way, they will not cross, and their routes will be shorter.

4 Proposed solution - Ant Colony algorithm

We will use the Computational Intelligence algorithm to solve this problem. The Ant Colony algorithm uses plenty of ants to find plenty of possible paths. Then the costs of paths are calculated. Then, the ants finding the best paths leave messages for future ants to follow.

4.1 Overview

In every iteration, every ant finds the whole path. At every point of the path, an ant can go to any other point. It chooses the point according to the discrete distribution. The probability of going to a node is bigger when the pheromone to this node is bigger. Also, the probability of going to a node is bigger when the node is closer.

At the beginning of optimization, the pheromone is distributed randomly. At the end of each iteration, when all the ants find their paths, they distribute pheromone. The better their path is, the more pheromone they leave.

The ants will restrict themselves only to consider such paths that the cycles from the warehouse through clients and back to the warehouse will not exceed the given constant s_{max}.

4.2 Probability of getting to the next node

In this text, we will assume the warehouse is node 0.

For the ant, the probability of going from node i to node j is according to the following formula:

$$p_{i,j} = \begin{cases} \frac{1}{c_i} \cdot (\tau_{i,j})^{\alpha} \cdot (\frac{1}{d_{i,j}})^{\beta}, & \text{if } j \in \{\text{possible nodes}\} \\ 0, & \text{otherwise} \end{cases}$$

Where:

1. $\tau_{i,j}$ is a pheromone on (i, j) edge at the moment
2. $d_{i,j}$ is the cost of the (i, j) edge (in the CVRP considered here, it is the Euclidean distance from node i to node j)
3. the $\alpha \geq 0$ and $\beta \geq 0$ are the parameters
4. the c_i is the normalizing constant for a given node i so that:

$$\sum_{j \in \{0,1,\ldots,n-1\}} p_{i,j} = \sum_{j \in \{\text{possible nodes}\}} p_{i,j} = 1$$

Therefore :

$$c_i = \sum_{j \in \{\text{possible nodes}\}} (\tau_{i,j})^{\alpha} \cdot (\frac{1}{d_{i,j}})^{\beta}$$

Set of possible nodes

The set of possible nodes always contains a 0-th node. Meaning an ant can always get back to the warehouse (unless it is in the warehouse, i.e. unless it is $i = 0$). So:

$$\forall_{i \in \{1,2,...,n\}} p_{i,0} > 0$$

and:

$$p_{0,0} = 0$$

The set of possible nodes for an ant at the beginning is a full set $\{\text{possible nodes}\} = \{0, 1, 2, ..., n\}$. Then, a node j will be removed from the set of possible nodes if at least one of those three things happen:

1. The ant already was in the node j in this iteration (this rule does not apply to $j = 0$). In particular, for every node i, the $p_{i,i} = 0$.

2. The distance $d_{i,j} + d_{j,0}$, is so long that if and ant would go from i to j and then to 0, then the cumulative distance from last visit in warehouse exceeds s_{max}.

3. The order of the j-th node is so big that after visiting the j-th node, this cycle's cumulative order will exceed the vehicle's capacity.

Those rules will guarantee that every path found by every ant will obey every restriction applied to the problem.

Note that the set of possible nodes from a client always includes $j = 0$. Sometimes, the $j = 0$ may be the only possible node. In such a case, the probability of getting back to the warehouse will be 100%.

However, the above rules do not imply that every ant will find the solution with the appropriate number of vehicles used. The $cars_{max}$ is a parameter of the problem, but not every found solution will obey this restriction. We believe this can be omitted in the optimization process because the Ant Colony will try to find the shortest path. We believe it will accidentally find the one with an appropriate number of cars used. If it is not the case, we will decide this problem is unsolvable by the proposed algorithm.

4.3 Distribution of pheromone

At the beginning of the optimization, the pheromone is distributed equally for every edge. This is the approach that can be found in [2], and [4]. Some other works suggest starting optimization with the random quantity of the pheromone. Other works suggest putting the initial pheromone according to the length of the edge and simultaneously reducing the β parameter. This work will not consider those approaches.

When every ant finds its path, the pheromone is updated. This happens once, at the end of each iteration.

The old, accumulated from the previous iteration, the pheromone is evaporating. Also, the new pheromone is added according to the heuristics: the better the path found with this edge, the more pheromone will be added.

Therefore, the pheromone is modified according to the equations:

$$\Delta\tau_{i,j} = \begin{cases} \frac{Q}{L_k}, & \text{if edge } (i,j) \text{ is a part of the path } L_k \\ 0, & \text{otherwise} \end{cases}$$

$$\tau_{i,j} := (1-\rho)\cdot\tau_{i,j} + \Delta\tau_{i,j}$$

Where:

1. $Q \geq 0$, $\rho \in [0,1]$ are parameters
2. L_k is a path that the r-th Ant just made

Note however, that sometimes the update is defined as follows:

$$\tau_{i,j} := (1-\rho)\cdot(\tau_{i,j} + \Delta\tau_{i,j})$$

This approach is equivalent to the previous one with a different Q.

5 Modifications

The basic version of the Ant Colony algorithm will optimize the complete graph as a whole. This approach will most likely not bring satisfactory results for problems with many vertices. The time needed to find a decent solution will be unacceptable.

We propose two possible solutions to this problem:

- **Reduce graph** - reducing the number of possible next steps for an ant. In the basic version, the ant can go from one stop to any other point. In most cases, such jumps are ridiculous. The forbiddance of such jumps will make the algorithm find a better solution faster.

- **Divide graph** - reducing the problem to smaller ones. The size of the problem is, in most cases, the biggest obstacle. It is an NP-hard problem, after all.
  We will adjust this problem according to the "Divide-and-conquer" design paradigm.

The **Divide graph** modification was motivated by the [1] paper. More information about this paper can be found in Section 3.

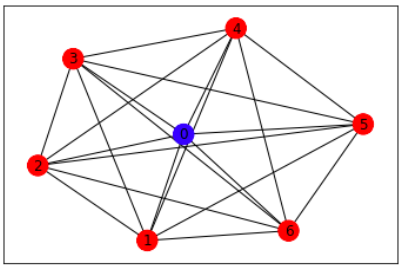

Figure 1: The original graph of neighbourhood - an Ant can go from any point to any other. The warehouse is tagged with blue. The clients are tagged with red.

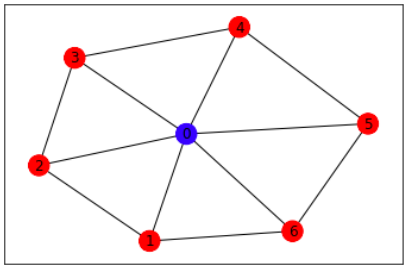

Figure 2: The modified graph of neighbourhood - an Ant can go from a point only to the close ones or to the warehouse. The warehouse is tagged with blue. The clients are tagged with red.

5.1 Reduce graph

In this work's basic version of Ant Colony, the graph of interest is the complete graph. We propose to reduce the number of edges. As the rule of thumb we will reduce the degree of a vertex from $n-1$ to $floor(\sqrt{n})+1$. We will achieve it by dropping the edges of the most distant vertices. Note, however, that the edges to the warehouse will not be eliminated - this is where the $+1$ in $floor(\sqrt{n})+1$ is from. The example reduction of this kind is visualized in Figure 5.1. Note that the modified graph can be directed, despite in the basic version, the graph was undirected.

5.2 Divide graph

In this work's basic version of Ant Colony, the graph of interest is the complete graph. We propose to split this graph into a smaller number of complete graphs.

The procedure we propose works as follows:

1. Take the whole graph S_n
2. Calculate $k = ceiling(ln(n))$
3. Use the 'KMeans' algorithm to split the n vertices into k groups
4. For every group, solve the smaller problem with basic Ant Colony algorithm

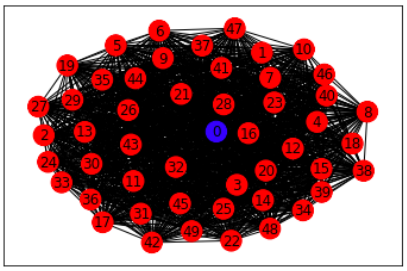

Figure 3: The original graph of neighbourhood - an Ant can go from any point to any other. The warehouse is tagged with blue. The clients are tagged with red.

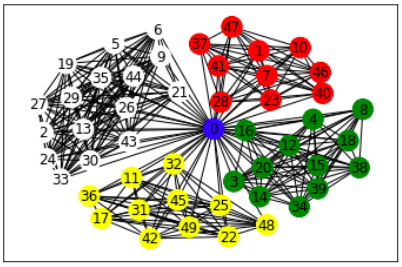

Figure 4: The clasterized graph of neighbourhood - an Ant can go from a point only to the ones that 'KMeans' algorithm qualified as the close ones. The warehouse is tagged with blue. The clients are tagged with different colours according to the group generated by 'KMeans' algorithm.

Keep in mind that the 'K-means' algorithm is randomized in of itself.

In Figure 5.2 there is plotted an example of the divided graph generated by this procedure.

The solution found with this procedure will almost surely not be the optimal one. However, it will be found fast and probably be good enough, specifically if the algorithm will be run multiple times for different 'K-means' runs.

Additional modification

Using the pheromone from the solution of Divideg Graph as a starting pheromone for the global optimization may be suitable. Note that the pheromone is distributed randomly at the start of the standard Any Colony algorithm. My idea is to run the "Divide the graph" algorithm for ~ 10 times and then distribute the pheromone proportionally to a found solution. Then the global optimization is expected to find a better optimum.

6 Hypotheses

In this section, the hypotheses for the work will be stated. Those hypotheses will be tested in the work.

1. The **basic version** will faster find better solutions than the Reduced graph and the Divided graph modifications for **smallest** graphs (up to 50 nodes).
2. The **Reduced graph** modification will faster find better solutions than the basic version and the Divided graph modification for **medium** graphs (from 50 to 100 nodes).
3. The **Divided graph** modifications will faster find better solutions than the basic version and the Reduced graph modification for **biggest** graphs (above 100 nodes).

7 Methodology

We will use two datasets to test the proposed algorithms. Those can be found in the widely-used online repository [5].

Those datasets will be used to determine the correctness of the algorithm and for testing the appropriate hypotheses.

From the set **Augerat 1995 - Set A**, four graphs will be used:

- A-n32-k05 (32 nodes) - small graph
- A-n44-k06 (44 nodes) - small graph
- A-n60-k09 (60 nodes) - medium graph
- A-n69-k09 (69 nodes) - medium graph

Graphs from **Uchoa et al. 2014** dataset will be used for testing the hypotheses for big graphs. This set contains graphs with a more significant number of notes (from 100 to 1000). The graphs are not specified and depend on the algorithm's speed and quality.

Take a note that the problem described in this work has additional restrictions over the CVRP described in [5]. In [5] the CVRP is a problem, where every car has a limited capacity. In this work, We additionally consider the limit on the length of the path of every car.

For every problem and algorithm, the calculations will be performed multiple times to see the differences in distributions of results.

For every such case, the algorithm will be running a fixed amount of time. In that, the results will be comparable.

The solutions will be compared to the greedy algorithm.

8 Schedule

The schedule of planned work can be found on Gantt diagram 5 that was generated with a free version of TeamGantt site.

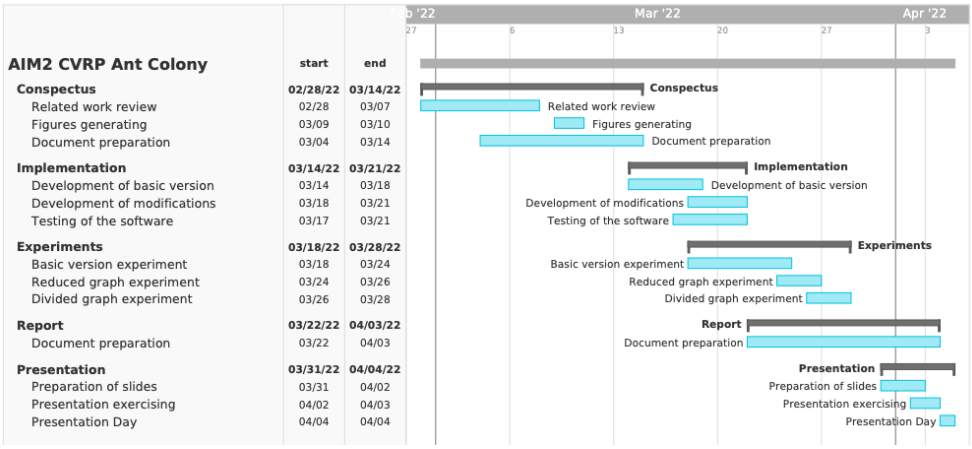

Figure 5: Gantt diagram that shows the schedule for the project.
It was generated with the free version of the site `prod.teamgantt.com`

9 Softwere

We will develop the software for this project in Python and use the ‘NumPy‘ library.

The visualizations will be provided with ‘Matplotlib‘ and ‘seaborn‘. For Graph visualization, the ‘networkx‘ package will be used.

The package ‘BeautifulSoup‘ was used to deal with ‘XML‘ files in Python. The ‘XML‘ is a filetype the datasets from section 7 are published in.

Implementing the ‘KMeans‘ algorithm is not part of this work. Therefore we will use the implementation from library ‘sklearn‘.

All the analysis will be available as ‘Jupyter Notebook‘s and will be available at ‘GitHub‘ page [6].

References

[1] M. A. H. Akhand, Zahrul Jannat Peya, Tanzima Sultana, and Al-Mahmud. Solving capacitated vehicle routing problem with route optimization using swarm intelligence. In *2015 2nd International Conference on Electrical Information and Communication Technologies (EICT)*, pages 112–117, 2015.

[2] Amir Hajjam, Lyamine Bouhafs, and A. Koukam. *A Hybrid Ant Colony System Approach for the Capacitated Vehicle Routing Problem and the Capacitated Vehicle Routing Problem with Time Windows*. 09 2008.

[3] Phan Nguyen Ky Phuc and Nguyen Le Phuong Thao. Ant colony optimization for multiple pickup and multiple delivery vehicle routing problem with time window and heterogeneous fleets. *Logistics*, 5(2), 2021.

[4] Michalis Mavrovouniotis. An ant system with direct communication for the capacitated vehicle routing problem. pages 14–19, 01 2011.

[5] J.E. Mendoza. VRP-REP: the vehicle routing community repository. `http://www.vrp-rep.org`. [Online; accessed 14-03-2022].

[6] Chojecki Przemysław. GitHub page with the code for this project. `https://github.com/PrzeChoj/MSI2/`. [Online; accessed 14-03-2022].

[7] W.F. Tan, Lai Soon Lee, Zanariah Abdul Majid, and Hsin-Vonn Seow. Ant colony optimization for capacitated vehicle routing problem. *Journal of Computer Science*, 8:846–852, 01 2012.

[8] Paolo Toth and Daniele Vigo, editors. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, USA, 2001.