SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

PROJECT

Capacitated Vehicle Routing Problem with Time Windows

Adrian Sušec & Matej Koščec

Zagreb, January 2024.

Contents

1. Introduction

Capacitated Vehicle Routing Problem with Time Windows (CVRPTW) is a problem belonging to Vehicle Routing Problems with additional constraints that address the real-life situations where we have to pay regards to the maximum capacity of a vehicle, as well as the situation where certain customers are able to receive the delivery within a predefined time period.

In CVRPTW, the primary objective is to design optimal routes for a fleet of vehicles originating from a central depot to service a set of customers.

This problem is not only theoretically intriguing but also has significant practical applications. Efficient solutions to CVRPTW can lead to substantial cost savings and operational efficiencies in various sectors like logistics, distribution, and public transportation.

The complexity of the problem, stemming from real-life constraints and combinatorial complexity, makes exact methods computationally infeasible for large instances. Consequently, heuristic and metaheuristic approaches are often used to find acceptable solutions within reasonable computational times.

2. Problem formulation

The problem is defined by certain parameters that are linked to vehicles and customers used in this instance. Constraints related to vehicles are:

- Number - maximum available vehicles
- Capacity - defined for a single vehicle

Constraints related to customers are:

- Demand - resources that have to be delivered to the specified customer
- Ready time - the earliest time at which the delivery can start
- Due time - the latest time at which the delivery can start
- Service time - time spent by the vehicle at the customer location

The CVRPTW in general specifies these constraints:

1. Each customer is served by exactly one vehicle/route, with the resource amounts that equal their demands
2. The demand on each route must not exceed the capacity of the vehicle.
3. The vehicle servicing a certain customer must arrive at the customer location within the interval given for that customer. The duration of the service can exceed the interval.
4. Each vehicle starts and finishes its route in node 0 (customer 0 location; depot), within the time interval given for customer 0.

3. Algorithm description

Two main algorithms were used to find feasible solutions that seem to be good enough. In the beginning we used only the Ant Colony Optimization. After we let the algorithm run on all the instances, it was obvious that the initial obtained solutions were far from wanted results, but they would become much better after the Ant Colony optimization algorithm was left to run for some time. Considering that, we decided to try use a Greedy algorithm and find an initial solution that would use a lot less vehicles, and we would use that solution to deposit more pheromones in the pheromone matrix for the Ant Colony Optimization which would further reinforce the good solutions. Those solutions would generally be found after a lot of iterations of the Ant Colony, which means that this reduces the time need to find good solutions.

Solutions are represented as a set of routes, each assigned to a vehicle. In case of Ant Colony Optimization, one vehicle can be likened to an ant. A route is a sequence of customers with associated delivery start times. The primary objective is minimizing the number of vehicles used to visit all the customers, with the secondary objective being the minimization of the distance travelled by those vehicles.

The Ant Colony Optimization algorithm iteratively improves the initial solution. The algorithm runs for a predetermined number of iterations or until a stopping condition (like the time limit) is met. A Greedy algorithm is used in the beginning to create an initial solution that minimizes the number of vehicles. This solution is not necessarily optimal but provides a good starting point.

Specifically, the greedy solution is constructed as follows: a vehicle starts from the depot and as the next location it can choose from the list of non-visited locations. Those locations are sorted by distance from the current location, followed by a sort in ready times. After selecting the next location in a greedy manner, that location is removed from the non-visited list. A vehicle returns to the depot when it would no longer be possible to visit any other location without breaking the due time constraint of the depot.

When a vehicle returns to the depot, another one is deployed in the already described manner. This is repeated until all the customers are visited.

Solutions obtained this way can be improved by running the Ant Colony Optimization algorithm. The key element for success is the pheromone matrix, which is of dimensions NxN, where N is the number of customers.

The specific implementation allows the initial amount of pheromones deposited to be separated in two categories: those that construct a solution found by the greedy algorithm, and those that are not a part of that solution. This allows us to choose the degree to which we want to reinforce the initial solution and guide the search according to it. Similar to the Greedy algorithm, one ant represents a vehicle. According to the probabilities calculated with the pheromone matrix, the ant selects the next location until it could no longer return to the depot in a viable manner, or until the capacity of the vehicle is used up. Main parameters guiding the search are alpha, beta and rho alongside the pheromone matrix, and they will be discussed in the later chapters.

4. Pseudocode

Algorithm 1 Greedy algorithm

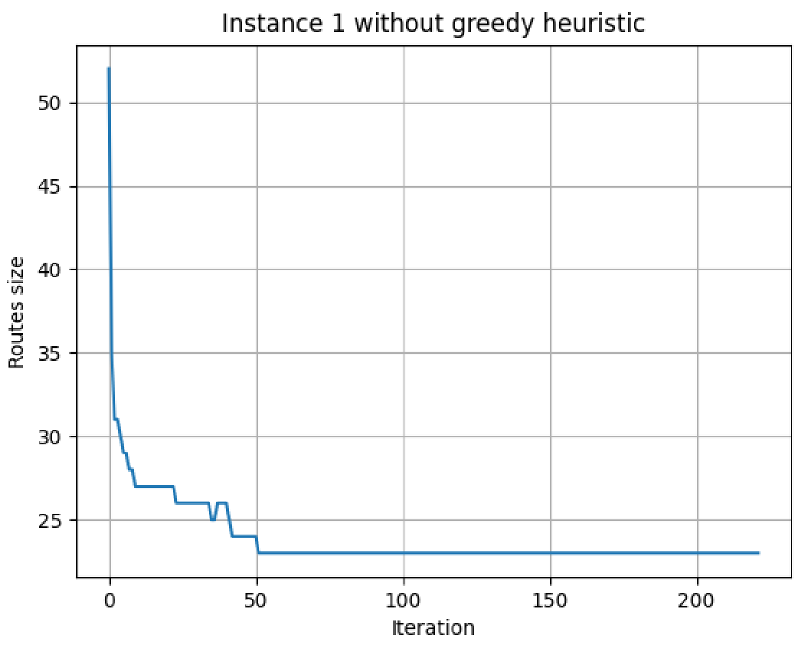
```
$notVisited \leftarrow allLocations$
$solution \leftarrow \{\}$
**while** $notVisited$ is not empty **do**
    $route \leftarrow [depot]$
    $nextLocation \leftarrow greedySelectNextLocation$
    **while** True **do**
        **if** can visit $nextLocation$ without breaking constraints **then**
            $notVisited \leftarrow notVisited - nextLocation$
            $route \leftarrow route + nextLocation$
        **end if**
        $nextLocation \leftarrow greedySelectNextLocation$
    **end while**
    $solution \leftarrow solution + route$
**end while**
$solution \leftarrow solution + depot$
return $solution$
```

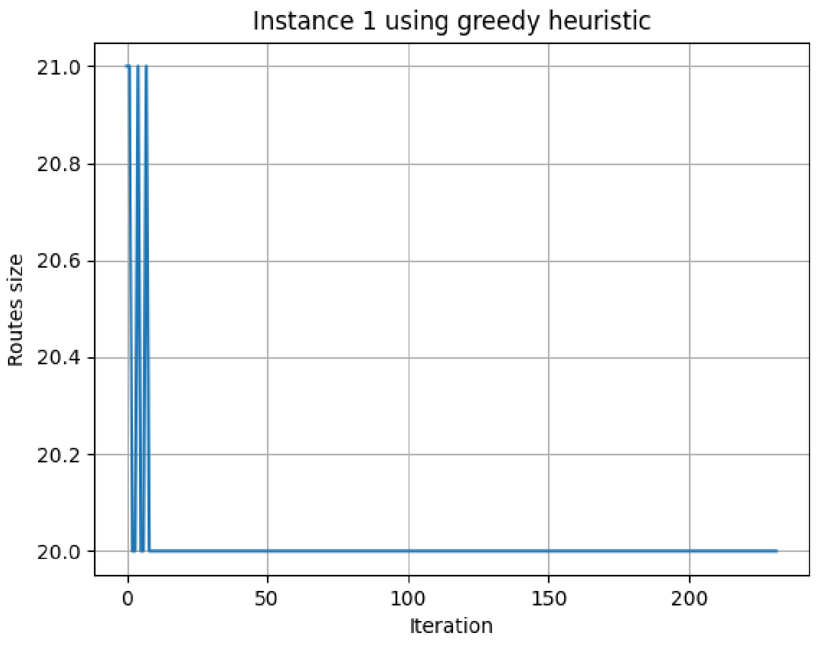
Algorithm 2 Ant Colony Optimization

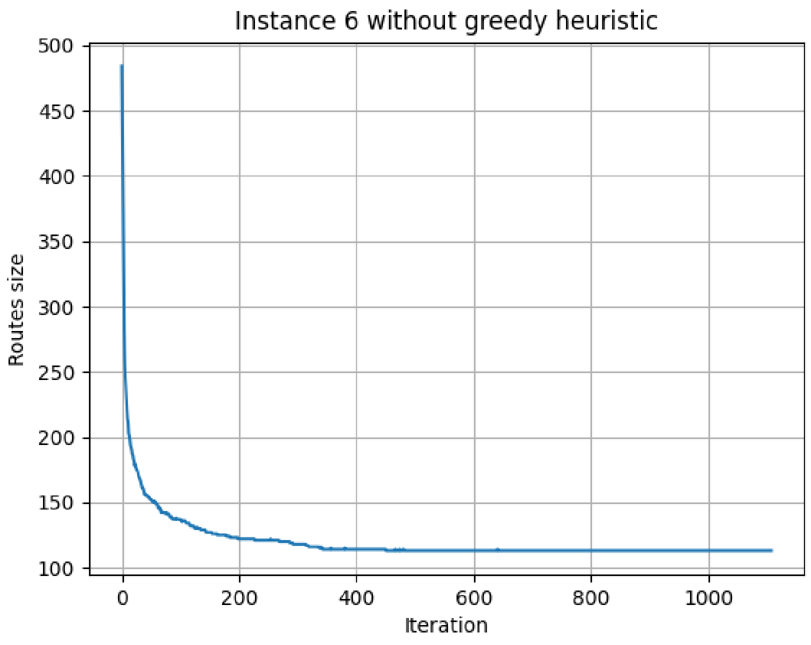
```
$bestSolution \leftarrow \{\}$
$pheromoneMatrix \leftarrow depositPheromones(greedySolution)$
**while** time limit not exceeded **do**
    $notVisited \leftarrow allLocations$
    $solution \leftarrow \{\}$
    **while** $notVisited$ is not empty **do**
        $route \leftarrow [depot]$
        $nextLocation \leftarrow select(pheromoneMatrix)$
        **while** True **do**
            **if** can visit $nextLocation$ without breaking constraints **then**
                $notVisited \leftarrow notVisited - nextLocation$
                $route \leftarrow route + nextLocation$
            **end if**
            $nextLocation \leftarrow greedySelectNextLocation$
        **end while**
        $solution \leftarrow solution + route$
    **end while**
    $solution \leftarrow solution + depot$
    **if** $goodness(solution) > goodness(bestSolution)$ **then**
        $bestSolution \leftarrow solution$
    **end if**
    $pheromoneMatrix \leftarrow evaporate(pheromoneMatrix)$
    $pheromoneMatrix \leftarrow reinforce(pheromoneMatrix, solution)$
**end while**
return $bestSolution$
```

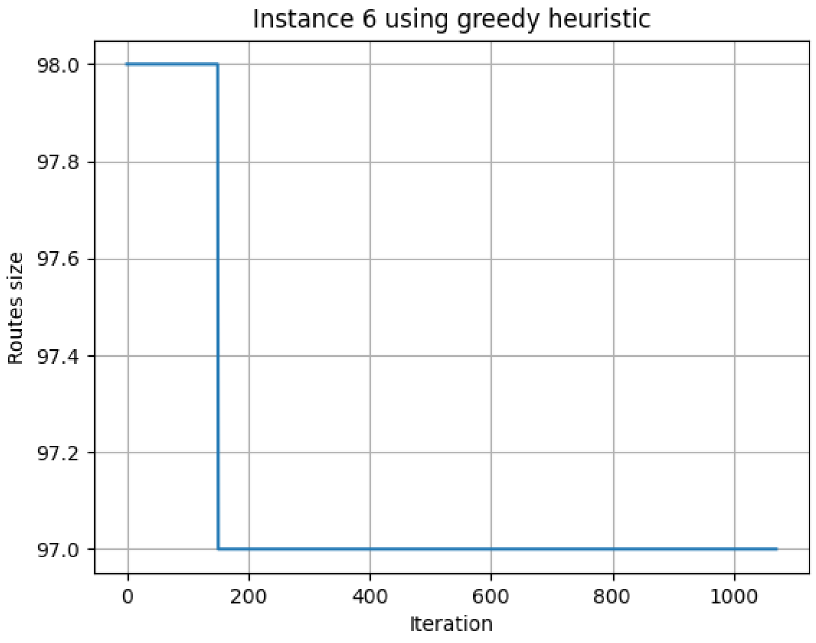
5. Analysis of results

Following charts show us the performance of generic Ant Colony Optimization compared to our hybrid heuristic algorithm where the solution found in a greedy manner is deposited with 100x more pheromones on transitions between locations that belong to that solution.




Instance 1 using greedy heuristic
Routes size
Iteration
21.0
20.8
20.6
20.4
20.2
20.0
0
50
100
150
200


Instance 6 without greedy heuristic
Routes size
Iteration
500
450
400
350
300
250
200
150
100
0
200
400
600
800
1000



Concretely, our best found solutions are as follows:

- instance 1: vehicles = 19, length = 5368.15
- instance 2: vehicles = 37, length = 13939.98
- instance 3: vehicles = 20, length = 12723.20
- instance 4: vehicles = 124, length = 61569.31
- instance 5: vehicles = 19, length = 74752.74
- instance 6: vehicles = 96, length= 87767.12

Other parameters impacting the search are $alpha$, $beta$ and rho. $Alpha$ and $beta$ impact the search through the probabilistic state transition rule (given a location i, the probability to select next city j is

$$p_{ij} = \frac{\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}{\sum_k \tau_{ik}^{\alpha} * \eta_{ik}^{\beta}}$$

Generally, we found that better solutions were found when $alpha$ was at least 3 times bigger than $beta$, and rho was at least 0.5, meaning the evaporation between iterations was noticeable. With $beta$ and $alpha$ being equal, in certain instances not enough emphasis was given to the initial greedy solution, which meant the search began with worse solutions.

Precalculation of the pheromone matrix and the probabilities for the transition to the next location was a big factor in the performance of our algorithm, which meant we managed to speed up the algorithm a lot. Our algorithm usually reaches the locally optimal solution within a few iterations ($t < 1s$) for smaller instances, and within a couple of seconds for bigger instances. All in all, we believe the solutions are quite satisfying considering the algorithm is not too complicated, and uses a few smart and noticeable improvements to reach a feasible solution. Run times of 5 minutes generally managed to find a shorter route, but with the same number of vehicles used up in the 1 minute run time.

6. Conclusion

In this project, we have presented a comprehensive study of the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW), a complex and practically relevant problem in the field of logistics and transportation. Our focus was on developing and analyzing a hybrid heuristic approach that combines the strengths of Greedy algorithms and Ant Colony Optimization to tackle the CVRPTW effectively.

We found that the hybrid heuristic approach, where the initial solutions from the Greedy algorithm are enhanced by the Ant Colony Optimization, significantly improves the performance compared to using generic Ant Colony Optimization alone. The use of a Greedy algorithm for initial solution generation proved to be an efficient strategy, as it reduced the number of vehicles required and made a solid foundation for further optimization by Ant Colony Optimization.

A higher weight on $alpha$, in comparison to $beta$, led to better solutions, emphasizing the importance of pheromone trails over heuristic information. The noticeable evaporation rate (rho) also played a vital role in escaping local optima and ensuring a diverse search.

While our algorithm shows promising results, it is not without limitations. Additionally, investigating other metaheuristic combinations or machine learning approaches could provide even better solutions, which should get us closer to the global optimum. Diving deeper into scalability and the algorithm's performance on larger datasets would also be an area worth researching.