UNIVERSITY OF ZAGREB

Faculty of Electrical Engineering and Computing

Master Programme

Heuristic Optimization Methods

REPORT – Project

Capacitated Vehicle Routing Problem with Time Windows

Erik Jusufi

Ac. year 2022/2023

Contents

1 Problem description

The given problem was to try and find best solutions for different instances of the capacitated vehicle routing problem with time windows. CVRP is classical combinatorial optimization problem. Given a central depot, a set of customers and a set of vehicles with equal capacity, problem is to choose a set of economic routes that allows delivering the amount of product requested by the customers without exceeding the vehicle capacity, where each customer is visited exactly once. All vehicles must start and finish their journey in the central depot.

2 Description of the implemented heuristic algorithm

For the given problem, greedy algorithm is used to create initial solution, and tabu search is used to optimize that initial solution.

Problem is defined with different classes.

Class Customer represents customer and contains customer data and function to calculate distance between 2 customers.

Class Vehicle represents vehicle and its route, it contains all customers that vehicle is going to visit, ordered by visiting order. It also contains functions to help generate neighborhood.

Class Solution represents specific solution with list of vehicles and their customers. Class solution has functions to calculate fitness or feasibility. It also contains function to define neighborhood by either moving random customer to different vehicle, moving random customer to new vehicle, swapping 2 customers from 2 vehicles, or reordering 2 different customers. Moving customer can result in reducing number of vehicles by 1.

Class problem contains problem specific information (parameters).

Using mentioned classes we try and solve the problem using combination of greedy and tabu search.

Greedy algorithm is creating feasible solution by putting customers in vehicles with respect to problem constraints. Initial solution is created by iterating through customers ordered by criteria: DUE_TIME – DISTANCE_TO_DEPOT. This heuristic is used because due time is most important customer parameter. If due time is low, vehicle has to be able to go to specific customer early on their route. I also subtracted due time with distance to depot because I wanted to give priority to customers which may have be far away, but have low due time.

Tabu search is searching for improving solutions in the neighborhood. It is generating wide neighborhood, and if there is an improving solution in the neighborhood, we expand search on that solution. If better solution is not found, we choose number of best solutions from neighborhood and randomly choose neighbor on which we will expand our search.

Objective function is a function I saw on the paper called *A Hybrid Approach for the Capacitated Vehicle Routing Problem with Time Windows by Ilya Bychkov and Mikhail Batsyn.* In the paper they use function that allows for solutions to be infeasible, but they exponentially increase the penalty for being in infeasible area. Objective function combines 2 target values, number of vehicles and distance, with priority given to number of vehicles. It also sums 3 infeasibility parameters: time over that represents sum of delays in respect to due time of the customers, capacity over that represents sum of excessive capacity and vehicles over that represents number of vehicles exceeding given constraints.

Termination criterion is number of iterations, counter for non-improving solution and timer.

3 Pseudocode

Pseudocode for greedy algorithm:

```
Iterate through all customers:
    Order customers by (due_time – distance_to_depot) ascending
While customers is not empty:
    Create new vehicle
    Iterate through remaining customers:
        Calculate new time
        If new_time < due_time and new_capacity < vehicle_capacity:
            Assign customer to vehicle and remove it from customer list
```

Pseudocode for tabu search algorithm:

```
Initialize best fitness, best solution, tabu tenure, tabu list and other parameters
Iterate for given number of iterations:
    If execution time or infeasible counter criteria is met, end the search
    Generate neighborhood
    If there is improving solution
        Solution = improving solution
    Else
        Choose random neighbor from list of best neighbors
    Add best neighbor to tabu list and remove latest if tabu list length is equal to tabu
    tenure
    If there is improving solution, update best fitness and best solution
```

4 Results

Instance 1

	Number of vehicles	Distance
1 minute	21	7298.18
5 minutes	21	7336.57
Unlimited	21	7336.57

Instance 2

	Number of vehicles	Distance
1 minute	50	18750.74
5 minutes	50	18510.41
Unlimited	50	18651.61

Instance 3

	Number of vehicles	Distance
1 minute	32	53921.33
5 minutes	32	48290.49
Unlimited	34	40233.21

Instance 4

	Number of vehicles	Distance
1 minute	101	60693.81
5 minutes	101	59780.88
Unlimited	101	60746.49

Instance 5

	Number of vehicles	Distance
1 minute	26	99440.82
5 minutes	26	99759.78
Unlimited	26	99486.82

5 Analysis

Implemented algorithm often gets stuck in local optima after only few iterations. Neighborhood size helps to diversify search, but not enough. If time constraint is 1 minute, it often helped to use smaller neighborhood sizes to give algorithm enough time to improve existing solution.

Tabu tenure and tabu list are not useful because of the size of search space. Same moves are rarely seen. Parameters for objective function are crucial.

Time constraints were not the problem because optimal solution is found pretty early in the search, and stopping criteria of the algorithm is usually non improving solution counter.

6 Conclusion

I should have defined neighborhood more carefully to diversify the search more, but also try and stay close to feasible area. Biggest problem is amount of infeasible solutions. If algorithm manages to end up in infeasible area, it can hardly find its way back to feasible area. Higher penalties for staying in infeasible area also didn't help.

Biggest improvement I would introduce is to define different neighborhood, not just changing and moving random customers, but trying to construct solutions from existing solutions, in a way that they can stay in feasible area or be very close to it.

Idea that came to mind is to define neighborhood that would somehow remove all customers from specific vehicle and try to rearrange them in existing vehicles and in that way reduce number of vehicles needed.

If I could start over, I would try and implement a variant of genetic algorithm.