CARP Project
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY
November 10 – December 5, 2022, Shenzhen, Guangdong, China

CS303 - ARTIFICIAL INTELLIGENCE

SIMULATED ANNEALING FOR CAPACITATED ARC ROUTING PROBLEM

Liu Leqi (12011327)[1]

[1]Southern University of Science and Technology, Shenzhen, China

ABSTRACT

The Capacitated Arc Routing Problem (CARP) can be found for application in real life. The exact methods are only applicable to small intances. However, there is no method that we can use it to get the best answer for general CARP since cARP is a classical NP-hard arc routing problem. At this time, heuristic search algorithms can be utilized to generate an appropriate answer for CARP. In this project, I implemented a simulated annealing algorithm to find a high-quality feasible solution for CARP.

Keywords: CARP, Simulated Annealing

1. INTRODUCTION

The Capacitated Arc Routing Problem (CARP) constitutes an important class of vehicle routing problems. It has applications in areas such as household refuse collection, street sweeping, inspection of electrical and pipeline networks for faults, mail delivery, etc..

In CARP, we have a set of vehicles at depot and they need to serve a given set of road subjected to the vehicle's capacity constraints. During the service, there are costs on the road for vehicles. The goal of CARP is to identify a routes such that minimize the total cost of vehicles.

2. PRELIMINARY

NOMENCLATURE

G	The undirected connected graph
V	vertex set
E	Edge set
T	Task set
Q	Capacity of a vehicle
v_0	Depot, the origin and destination
r_i	The chosen route for vehicle i
$c(e_i)$	Cost of edge
$d(t_i)$	Demand of task

CARP can be described as follows: A road network represented as an undirected graph $G = (V, E)$ with a vertex set V with vertices v_i and an edge set E with edges e_i cost $c(e_i)$. The task set T of tasks t_i with demand $d(t_i)$ needs to be served by a fleet of identical vehicles associated with the same capacity Q. The objective is to determine a set of routes for the vehicles to serve all the tasks with a minimal cost, such that each vehicle trip starts and ends at the depot v_0, each task must be served in a trip, and the total demand handled by any vehicle must not exceed its capacity Q. The we can formulate CARP as below:

$$\begin{aligned}
&\min \sum c(r_i) \\
s.t. &\sum_j \sum_i d_j(r_i) = |T| \\
&t_{pi} \neq t_{qj}, \forall (p, i \neq q, j) \\
&t_{pi} \neq inv(t_{qj}), \forall (p, i \neq q, j) \\
&\sum_i d(t_{ki}) \leq Q, \forall k = 1, 2, ... \\
&t_{ki} \in T
\end{aligned}$$

3. METHODOLOGY

3.1 General Workflow

There are mainly 3 steps taken in simulated annealing for CARP: initialization, disturbance, optimal choice. We will always iterate to execute them until reach the time limit.

1. Initializaton: generate a set of solutions and take the one that costs least as initial current solution.
2. Disturbance: during simulated annealing, for every iteration, the next solution will be generated by taking a small disturbance on the current solution.
3. Optimal choice: if the next solution costs less than current or in a small probability, we will choose next solution to be the newer current solution.

The following 6 rules are taken in initialization.

1. random at all
2. maximize the cost from the task to the depot

Documentation for `asmeconf.cls`: Version 1.30, December 6, 2022.

3. minimize the cost from the task to the depot
4. maximize the term $dem(t)/sc(t)$
5. minimize the term $dem(t)/sc(t)$
6. use rule 2) if the vehicle is less than half-full, otherwise user rule 3).

where $dem(t)$ and $sc(t)$ are the demand and serving cost of task t, respectively.

3.2 Detailed Algorithm

- Simulated Annealing: Since the time is limited for this project, this algorithm should be cut off before time out. There is a timer to control the remaining time.

Algorithm 1 SIMULATED ANNEALING FOR CARP

```
function SIMULATEDANNEALING(list of tasks' demand)
    current←initSolution(list of tasks' demand)
    while within cooling do
        cool temperature
        while within the length of Markov chain do
            next←newSolution(current)
            if next costs less than current then
                current←next
            else if within a small probability then
                current←next
            end if
        end while
        if Going to be time out then
            break
        end if
    end while
    return current
end function
```

- Path Scanning: This algorithm is in order to get the initial solution. The 6 rules mentioned upon are all used in it.

- Solution Utils: Some functions that used in this project, for initial solution ans new solutions.

3.3 Analysis

The simulated annealing algorithm simulates the annealing process of solid. The key of it is that it will jump out of the trap of local optimal for some probability.
During the generation of new solution, I used two operators: flip and swap. The newer solution may be worse than the old one, but it might reach the global optimal when being stable. Therefore, we will choose the newer solution when it is better than the old one or for a probability as below (at k-th iteration of simulated annealing):

$$P_k = \exp\left(-\Delta E_k / T\right)$$

where ΔE_k is the cost difference between newer solution and the old one, T is the current temperature in simulated annealing.

Algorithm 2 PATH SCANNING

```
function PATHSCANNING(free: list of required arcs)
    while free is not empty do
        R ← ∅, load ← 0, cost ← 0, i ← depot
        while True do
            d̄ ← ∞
            for each u ∈ free and load+q(u) ≤ Q do
                if d(i, beg(u)) < d̄ then
                    d̄ ← d(i, beg(u))
                    ū ← u
                else if d(i, beg(u)) = d̄ and better(u, ū, rule)
then
                    ū ← u
                end if
            end for
            if d̄ = ∞ then
                break
            end if
            add ū at the end of route R
            remove arc ū and its opposite from free
            load ← load + q(ū)
            cost ← cost + d̄ + c(ū)
            i ← end(ū)
        end while
    end while
    return (list of routes, total cost)
end function
```

Algorithm 3 SOLUTION UTILS

```
function INITSOLUTION(list of tasks' demand)
    apply 6 rule on path-scanning and return the best one with
least cost
end function

function NEWSOLUTION(list of routes, total cost)
    take a small disturbance on the list of routes and then get a
new total cost.
end function
```

4. EXPERIMENTS

4.1 Setup

4.1.1 Environment.

- Windows 10 & Ubuntu 20.04
- Python 3.8.8
- NumPy 1.20.1

4.1.2 Data Set. I only used the 7 data set provided by course. It performed better than only using path-scanning, especially for the large graph, approximately 1200-1500 less cost.

4.1.3 Hyper-parameters. There are 4 hyper-parameters in this project.

- Number of times for random path-scanning
- Initial temperature
- the decay rate in simulated annealing
- The length of Markov chain

Besides the random rule, there are only 5 rules for generating initial solution. But to make the initial solution better, number of times of random path scanning is important. Here it was set 10 times.

To make simulated annealing starts from a state at enough randomness, initial temperature was set to be 10000. Apart from that, the decay rate of temperature was 0.95 and the length of Markov chain was 200. All of that were tested.

4.2 Results

All tests are with TERMINATION = 60 and SEED = 100. Long time for time out is in order to let the program run until it finishes. It was tested for 10 times and the result 1 is the average. The figure 1 is one of the tests.

Data Set	Time Consuming (sec)	Initial	Final
egl-e1-A	20.30	3716	2672
egl-s1-A	28.97	4356	2309
gdb1	10.32	353	353
gdb10	11.89	304	304
val1A	14.34	185	185
val4A	28.81	453	453
val7A	24.58	342	342

TABLE 1: RESULTS

4.3 Analysis

The degree of optimization of large graph is great (egl-e1-A, egl-s1-A), while there is hardly optimization on small graph (gdb1, gdb10, val1A, val4A, val7A). And the results are the same in all the 10 tests except for the time consuming (maybe the SEED are the same). The reason may be 1) the methods implemented for generating new solution is not effective enough to optimize 2) or the simulated annealing algorithm was stacked in local optimal or plateau.

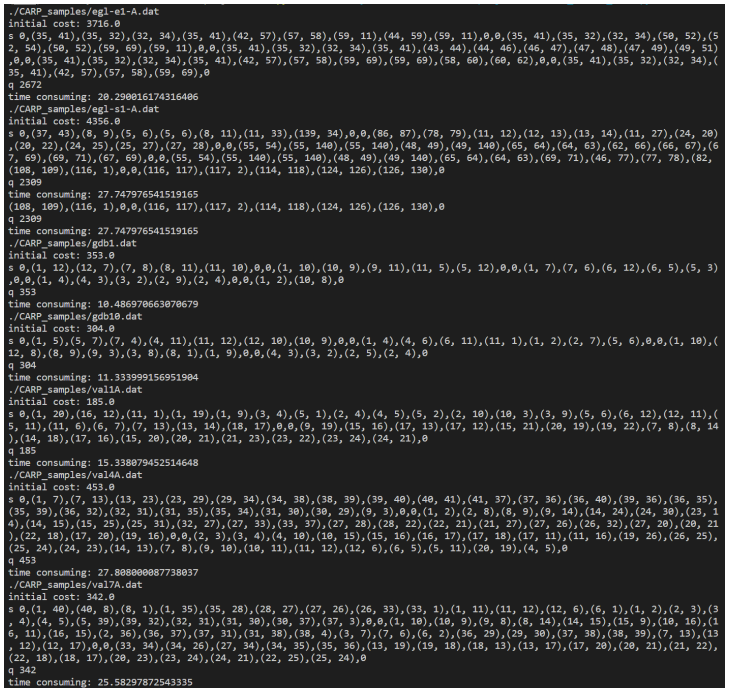

FIGURE 1: ONE OF THE TESTS

5. CONCLUSION

There are plenty of heuristic approaches for solving CARP since it was defined by Golden and Wong in 1981. The simulated annealing may not to be the best one but it earns its worth. It can provide a satisfactory answer of CARP. However, it performs bad for a large graph in a limited time and hardly optimization on small graph. Not only is it time consuming for cooling down, but also there are amount of local optimal solutions and plateaus in large graph. Being trapped into local optimal or plateau is a vulnerability of simulated annealing. It may be better if we utilize some more effective operators in generating new solution, which is the heart of jumping out of local optimal and plateau.

REFERENCES

[1] K. Tang, Y. Mei and X. Yao, "Memetic Algorithm With Extended Neighborhood Search for Capacitated Arc Routing Problems," in IEEE Transactions on Evolutionary Computation, vol. 13, no. 5, pp. 1151-1166, Oct. 2009.

[2] T. Yao, X. Yao, S. Han, Y. Wang, D. Cao and F. Wang, "Memetic algorithm with adaptive local search for Capacitated Arc Routing Problem," 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 2017, pp. 836-841.

[3] Min L, Ray T. Efficient Solution of Capacitated Arc Routing Problems with a Limited Computational Budget[J]. Australasian Joint Conference on Artificial Intelligence, 2012.

[4] Gündüz Ulusoy. The fleet size and mix problem for capacitated arc routing[J]. European Journal of Operational Research, 1985, 22(3):329-337.