CARP_Report

赵韦铭

11611004

1. Preliminaries

1.1 problem Description

This project is to solve a CARP, Capacitated arc routing problems (CARP) arise in distribution or collecting problems where activities are performed by vehicles, with limited capacity, and are continuously distributed along some predefined links (roads, streets) of an associated network.

1.2 problem Application

There are many applications for this problem. In the logistics and transportation industry, the use of carp can save oil costs. For example, plan the most suitable path for a snow removal truck or garbage truck to save on fuel costs.

2. Methodology

The method what I use is based on path scanning and LS.Use all this method can improve the algorithm highly. First, the Dijkstra algorithm is used to obtain the shortest distance between every twopoints, then the ellipse rule is used for Path scanning, and finally the local search is

used for optimization. The simulated annealing algorithm is used to prevent the entry into the local optimal solution.

2.1 Notation

- ned : Number of sides of all tasks
- td: total demand
- rvc: remained vehicle capacity

- w :the vehicle capacity
- Scur :the solution being constructed in the current iteration
- Sbest :the best solution obtained so far
- CScur:the total cost of solutions Scur
- CSbest :the total cost of solutions Sbest
- LB: the lower bound for the problem
- F: the set of feasible unserved edges that are at minimum distance from the last node added to Scur.
- S: a solution

2.2 Data structure

Graph(Adjacecy matrix): The relationship between edges and edges and the shortest distance between two points is the *Graph(adjacency matrix)* of data structure.

Array:when path scanning, the set of feasible unserved edges that are at minimum distance from the last node added to Scur is *Array*

Heap:when I optimize the Dijkstra algorithm, the *heap* need to use

2.3 Model design

2.3.1 formulate and solution

I formulate the question for three module,each of the module need some solutions.

FORMULATE	SOLUTION
Find shortest path	Dijkstra Algorithm
Path Scanning	Ellipse rule
Optimal	Reverse-operator merge-split

2.3.2 Description:

path scanning_ellipse rule

Because of what I use is ellipse rule ,so I did not use 5 rules which are on the slide.

The "ellipse rule" is defined as follows. Let ned be the number of edges with positive demand in the network, td the total demand to be collected, tc the total cost assigned to edges with positive demand, a real parameter, and [vh, vi] the last serviced edge on the route. If the remaining capacity of the vehicle is less than or equal to × td/ned (i.e., the average demand on the arcs), then the next edge to be serviced [vp, vj] must be the nearest edge to [vh, vi] (vi = vp, if the edges are adjacent) satisfying the condition:

$$SP(vi, vp) + cpj + SP(vj, v0)tc/ned + SP(vi, v0), (1)$$

where for example, SP(vi, vp) is defined as the shortest path cost between the nodes vi and vp, and vi and v0 are the foci of the ellipse. If no feasible edge (i.e., an edge with demand less than or equal to the remaining capacity of the vehicle) satisfies (1) then the vehicle returns directly to the depot.

local search (reverse operator,　merge-split)

The local search is composed of two parts: reverse operator and merge-split (MS).

The local search starts with a reverse operator similar to the one-way 2-opt operator. That is, it reverses the direction of the subroutine. Suppose the inverse operator is applied to a journey consisting of tasks. At each repetition, all possible subroutines are numbered and the length of the subroutine increases from 1 to t-1. In this course, once more low cost is solved, the current solution will be updated.

If the reverse operator can not improve the solution, then use the merge split (MS) operator to perform the best search. That is, at each step, to check all the solutions that the MS operator can carry out from the current solution, and choose the best and best solution to replace the current solution.

2.4 Detail of algorithm

Algorithm 1 Ellipse Rule

```
CSbest ← +∞
iter ← 1
while iter < some and (CSbest > LB)) do
    Scur ← v0
    rvc ← w
    for i = 1 to ned do //for all the edges with positive demand

        if (rvc > α × td/ned) then
            Determine F
        else
            Determine F satisfying equation (1)
        end if
        if (F = {}) then
            Scur ← Scur ∪ {v0} //the vehicle returns to the depot
            rvc ← w
        else
            Select [vp, vj] randomly from F //an additional edge is
            serviced in the solution being constructed
            Scur ← Scur ∪ {[vp, vj]} ∪ {vj}
            rvc ← rvc − qpj
        end if
    end for
    if (CScur < CSbest) then //if the current solution is better than the
        best so far
        Sbest ← Scur
    end if
    iter ← iter + 1
end while.
```

Algorithm 2 Local search

```
Input: solution s
Output: potentially improved solution s
while s remains changed
    for each sub-routes SR of each route R in s do
        reverse SR to obtain a new solution s
        if s is better than s then
            s ← s
            break
        end if
    end for
end while
if s is not updated then
    apply MS operator to improve s;
    if s is updated then
        while s remains changed
            for each sub-routes SR of each route R in s do
                reverse SR to obtain a new solution s
                if s is better than s then
                    s ← s
                    break
                end if
            end for
        end while
    end if
end if
return s;
```

3. Empirical Verification

I use the command line to test.

3.1 Dataset

First,I use the simple dataset to find bug,then I use the dataset from sakai.The simple dataset is there

```
NAME : simpledataset
VERTICES : 7
DEPOT : 1
REQUIRED EDGES : 5
NON-REQUIRED EDGES : 3
VEHICLES : 3
CAPACITY : 6
TOTAL COST OF REQUIRED EDGES : 12
NODES       COST        DEMAND
6   4       4           0
1   2       4           0
6   7       3           2
1   5       3           0
2   3       2           3
3   4       3           3
4   5       7           1
5   6       4           2
END
```

After that I use the dataset from carp webside **egl-e2-B val2B egl-s1-B**

3.2Performance

3.2.1 How to measure performance

I measure performance by time ,and I measure performance the difference of the cost of my result between lowest bound cost of the dataset.

3.2.1.1 measured by time

I think I can deal with problem under 60 vertice very well and find the lower bound cost result.However, for egl-s1-A and egl-e1-A,I can't find the best result even the local optimal solution will be found very fast.Besides, my Dijkstra use adjacency matrix, but the grape is sparse graph(稀疏图)，It will cost many time to find Dijkstra matrix.

3.2.1.2 measured by difference between result and lowest bound

The val1A val4A gdb10 val7A and gdb1 is lowest bound and difference is 0,and the eg1-sl-A has difference is 5267 - 5018 = 249 and egl-e1-A has difference is 3612 - 3548 = 64.

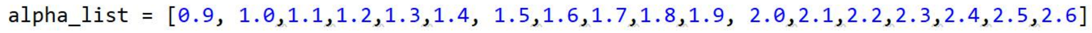
3.2.2 test envirment

operating system: windows10

cpu: inter core m3 7th Gen

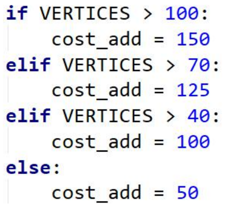
3.3 Hyperparameters

I use ellipse rule and the rule need a hyperparameters α .and the α is randomly choose from follow picture



```
alpha_list = [0.9, 1.0,1.1,1.2,1.3,1.4, 1.5,1.6,1.7,1.8,1.9, 2.0,2.1,2.2,2.3,2.4,2.5,2.6]
```

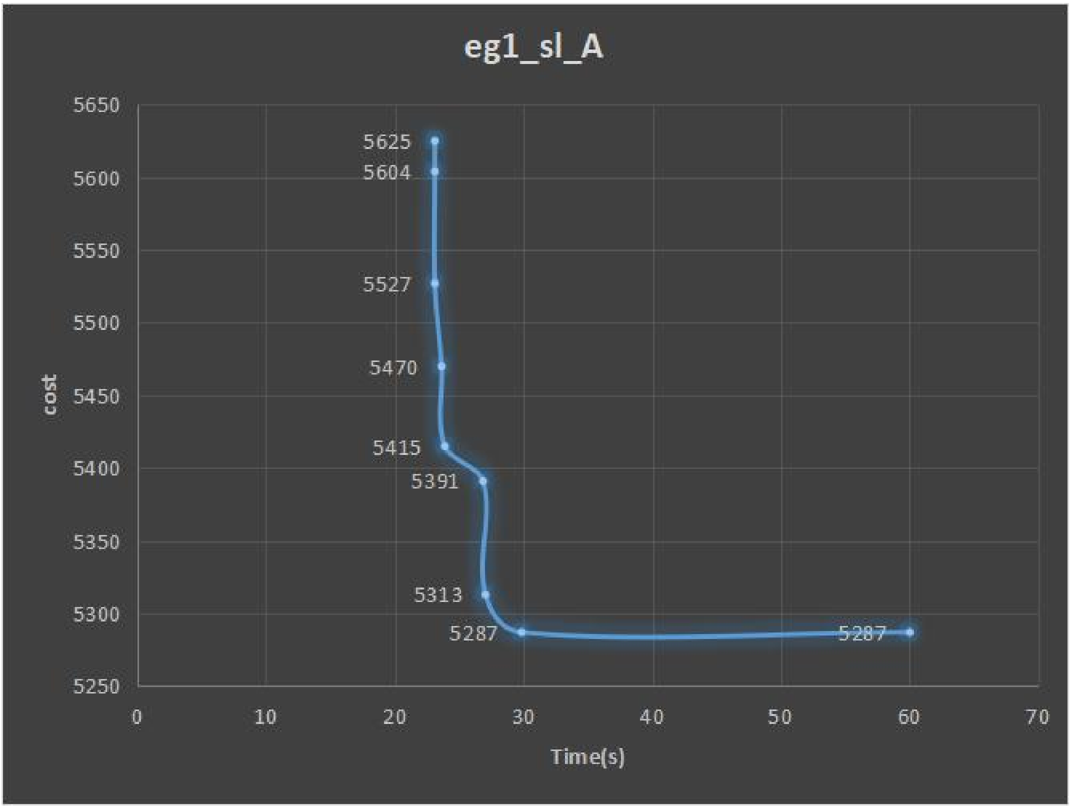
Besides,I use the hyperparameters to decide if the result is in local optimal can't continue, need to go back to a result which can not bigger than local optimal result is ***cost_add***.(Simulated Annealing Algorithm)The ***cost_add*** is 150 if num of vertice > 100 ... then can see follow picture.

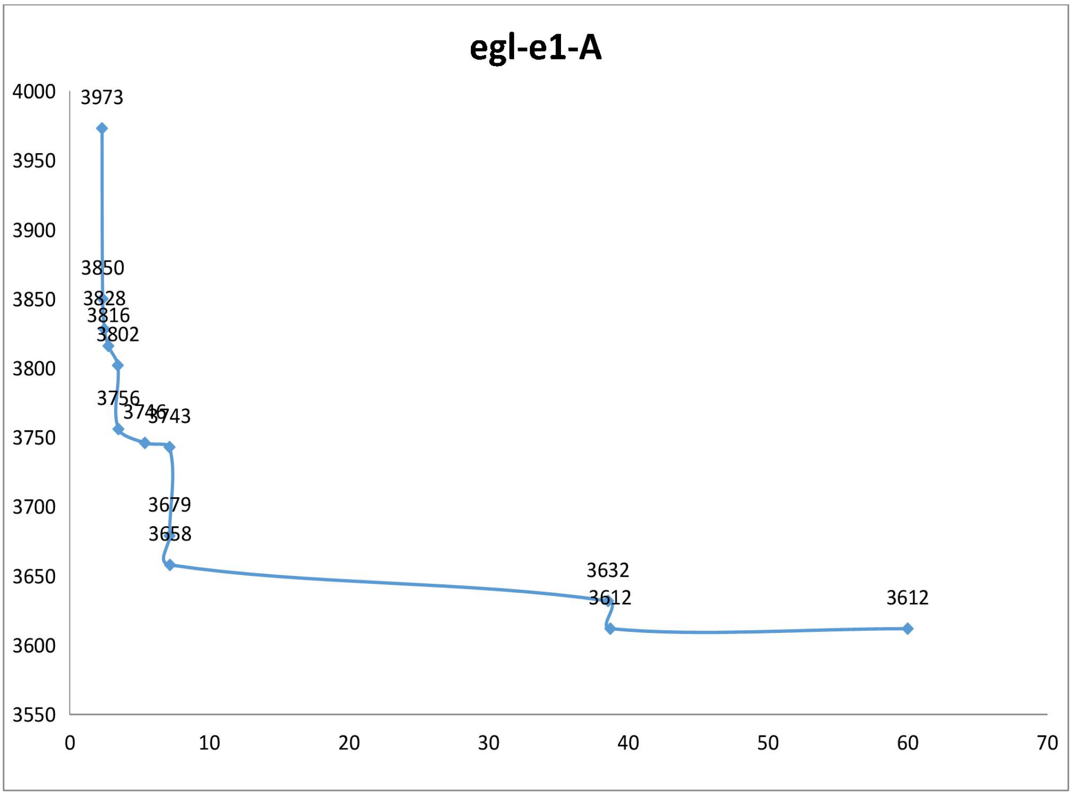


```
if VERTICES > 100:
    cost_add = 150
elif VERTICES > 70:
    cost_add = 125
elif VERTICES > 40:
    cost_add = 100
else:
    cost_add = 50
```

3.4 Experimental results

(for the egl_s1_A,the Dijkstra algorithm cost to many time, and it will cost over 20s)


eg1_sl_A
cost
Time(s)
5650
5600
5550
5500
5450
5400
5350
5300
5250
0
10
20
30
40
50
60
70
5625
5604
5527
5470
5415
5391
5313
5287
5287


egl-e1-A
4000
3950
3900
3850
3800
3750
3700
3650
3600
3550
0
10
20
30
40
50
60
70
3973
3850
3828
3816
3802
3756
3746
3743
3679
3658
3632
3612
3612

ALL THE RESULT IN TABLE

	60(S)	600(S)	LB
VAL1A	173	173	173
VAL4A	402	400	400
VAL7A	279	279	279
gdb1	316	316	316
gdb10	275	275	275
egl-e1-A	3612	3612	3548
egl-s1-A	5311	5267	3548
egl-e2-B	6473	6457	6301
Val2B	260	259	259
egl-s1-B	6850	6850	6388

3.5 Conclusion

3.5.1 advantage

I think the experimental results are meet my expectations.the local search and ellipse rules I used were very good for this carp task.I can find a local optimal result very fast.

3.5.2 disadvantage

The only regret is that there is too much time running the Dijkstra algorithm.Because sparse matrix is better to use adjacency list,But I use adjacency matrix.

4 References

[1] Ke Tang, Senior Member, IEEE, Juan Wang, Xiaodong Li, Senior Member, IEEE, and Xin Yao, Fellow, IEEE “A Scalable Approach to Capacitated Arc Routing Problems Based on Hierarchical Decomposition” 11, NOVEMBER 2017

[2] Luís Santos a,c, João Coutinho-Rodrigues a,c,* , John R. Currentb,c “An improved heuristic for the capacitated arc routing problem” 2009