Project2 CARP

1. CARP Introduction

Capacitated Arc Routing Problem

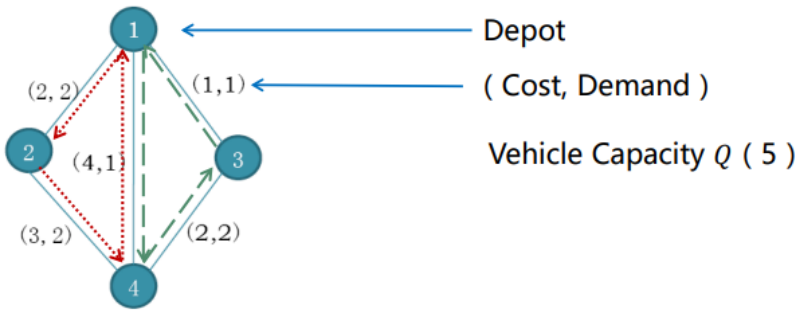
Problem formulation

- consider an undirected connected graph $G = (V, E)$
- a vertex set V
- an edge set E
  - each edge $e \in E$ incurs a cost $c(e)$, whenever a vehicle travels over it or serves it
- a set of required edges(tasks) $T \subseteq E$
  - each requiret edge(task) $t \in T$ has a demand $d(t) > 0$ associated with it
- a fleet of identical vehicles, each of capacify Q
  - based at a designated depot vertex $v_0 \in V$

Objective

determine a set of routes for the vehicles to serve all the tasks with minimal costs while satisfying

- each route must start and end at depot v_0
- total demand serviced on each route must not exceed Q
- each task must be served exactly once
  - the corresponding edge can be traversed more than once



Solution Representation

we have a solution to CARP as

$$s = (R_1, R_2, \dots R_m)$$

- m is the number of routes (vehicles)
- the k th root $R_k = (0, t_{k1}, t_{k2} \dots t_{kl_k}, 0)$
  - 0: the dummy task are both 0 and its two endpoints are both v_0

    - t_{kt}: the t th task
    - l_k the number of tasks served in R_k
- the cost and the demand of the dummy task are both 0

since each task here is an undirected edge and it can be served from either direction, so each task in R_k must be specified from which direction it will be served

- $t_{kt} = (head(t_{kt}), tail(t_{kt}))$

Objective Function

- Total cost of all routes
    - $minimize\ TC(s)$
    - $TC(s) = \sum_{k=1}^{m} RC(R_k)$
        - $RC(R_k)$ is the route cost of route R_k, which can be computed as:
        - $\sum_{i=1}^{l_k} c(t_{ki}) + dc(v_0, head(t_{k1})) + \sum_{i=2}^{lk} dc(tail(t_{k(i-1)}), head(t_{ki})) + dc(tail(t_{kl_k}), v_0)$
        - $dc(v_i, v_j) > 0$ is the cost of the shortest path from v_i to v_j $(i \neq j)$
- Each task is served exactly once
    - $\sum_{k=1}^{m} l_k = |T|$
    - $t_{k_1 i_1} \neq t_{k_2 i_2}\ \ \forall (k_1, i_1 \neq k_2, i_2)$
    - $t_{k_1 i_i} \neq inv(t_{k_2, i_2})\ \ \forall (k_1, i_1 \neq k_2, i_2)$
- Capacity constraint
    - $\sum_{i=1}^{l_k} d(t_{ki}) \leq Q\ \ \forall k = 1, 2, \ldots, m$
- Definition of t
    - $t_{ki} \in T$

2. DDL

- Solver description：2021-11-30
- Code CARP_solver.py：2021-11-30
    - Python 3.9.7
    - numpy

3. Input

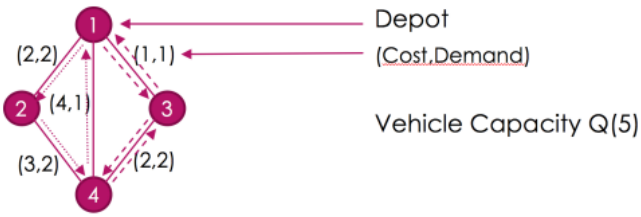
Command

```
python CARP_solver.py < CARP instance file> -t <termination> -s <random seed>
```

- `<CARP instance file>` the absolute path of the test CARP instance file
- `<termination>` the termination condition of your algorithm
    - range from [60s, 600s]
    - although our tests rely on your solver's internal time measurement, we will still measure your solver's runtime from external
- `<random seed>` the random seed used in this run

- the random seed controls all the stochastic behaviors of your solver
- If your solver is deterministic, it still needs to accept them, but can just ignore them

4. Output



Your solver must print messages to the standard output

Your solver has to print two lines

Solution line

begins with a lower case “s” followed by a space

the best solution your solver has found for the CARP problem instance must be printed in this line

```
s 0,(1,2),(2,4),(4,1),0,0,(4,3),(3,1),0
```

- 0：original part
- (1,2)：you choose to first finish task between vertex 1 and 2
- 0：back to the original

Quality line

begins with a lower case “q” followed by a space

solution quality (i.e., the total cost) of your best found solution must be printed in this line

```
q 15
```

- 15: the cost you find during the search

5. Format of CARP problem instance

```
NAME : gdb1
VERTICES : 12
DEPOT : 1
REQUIRED EDGES : 22
NON-REQUIRED EDGES : 0
VEHICLES : 5
CAPACITY : 5
```

```
TOTAL COST OF REQUIRED EDGES : 252
NODES       COST         DEMAND
1   2   13       1
1   4   17       1
1   7   19       1
1   10   19       1
1   12   4       1
2   3   18       1
2   4   9       1
2   9   2       1
3   4   20       1
3   5   5       1
5   6   7       1
5   11   20       1
5   12   11       1
6   7   4       1
6   12   3       1
7   8   8       1
7   12   18       1
8   10   3       1
8   11   10       1
9   10   16       1
9   11   14       1
10   11   12       1
END
```

Specification part

Line Number	Keyword	Value	Explanation
1	NAME	String	the name of the instance
2	VERTICES	Number	the number of vertices
3	DEPOT	Number	the depot vertex
4	REQUIRED EDGES	Number	the number of required edges
5	NON-REQUIRED EDGES	Number	the number of non-required edges
6	VEHICLES	Number	the number of vehicles
7	CAPACITY	Number	the vehicle capacity
8	TOTAL COST OF REQUIRED EDGES	Number	the total cost of all tasks

Data part

Data is given explicitly in this section

Line 9	NODES		COST	DEMAND
10	0	1	437	0
...	...	...	...	...
last row	END			

- 1st and the 2nd number are the node indices ("0" and "1")
- 3rd number is the cost of the edge
- 4th number is the demand of the edge ("0" indicates it is a non-required edge)

Please note that each edge appears only once, and each edge has two directions with same cost in each direction since the graph is an undirected graph

An "END" is added to the end of the file after all edges are listed

6. Evaluation

Usability test 50%

Once your solvers have passed a test, you will get all the corresponding scores

- 25%: Your solvers can satisfy the input and the output requirements
  - cannot handle the inputs
  - the output messages are undesirable
    - missing 's' line or 'q' line
    - the format of the output solution is incorrect
    - the total cost of the output solution is not correctly calculated
- 25%: the best found solution output by your solver is a feasible solution
  - infeasible solutions violate at least one constraints of CARP

Efficacy test 50%

The scores you get in this test depend on the rankings your solvers obtain. Note only those solvers that have been through usability test will be tested in efficacy test

select different CARP instances with different sizes from existing CARP benchmarks to test your solvers

for each instance, we will set a common cutoff time for all participant solvers and compare their solution quality

Robustness test 20%

The scores you get in this test depend on the rankings your solvers obtain. Note only those solvers that have been through usability test will be tested in robustness

be applicable to a wide range of problem instances

- we will generate a fixed number of instances that are very hard for it
- we will put all the newly generated instances into a pool, forming a test set
- we will set a common cut-off time for all participant solvers and compare their solution quality

7. Test Environment

- Operation System: Debian 10
- Server CPU：2.2GHz*2, 8-core total
- Python version: 3.9.7
    - You can use multithread, but the number of thread shall not exceed 8

8. Online Judge and Reference

- Paper reference link: http://scholar.google.com
- A website for you to submit your code and online judge: http://10.20.99.199/