**Q1: Solve shortest path problem manually**

Table 1 Label correcting algorithm calculation process

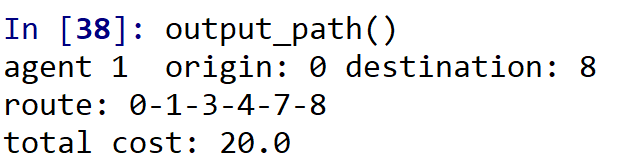
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| iteration | Node 0 | Node 1 | Node 2 | Node 3 | Node 4 | Node 5 | Node 6 | Node 7 | Node 8 |
| 0 | 0 | Inf | Inf | Inf | Inf | Inf | Inf | Inf | Inf |
| 1 | **0** | 6 (0) | 2 (0) | Inf | Inf | Inf | Inf | Inf | Inf |
| 2 | 0 | 6 (0) | 2 (0) | 8 (1) | Inf | Inf | Inf | Inf | Inf |
| 3 | 0 | 6 (0) | 2 (0) | 8 (1) | 12 (2) | Inf | Inf | 22 (2) | Inf |
| 4 | 0 | 6 (0) | 2 (0) | 8 (1) | 11 (3) | 15 (3) | Inf | 22 (2) | Inf |
| 5 | 0 | 6 (0) | 2 (0) | 8 (1) | 11 (3) | 13 (4) | Inf | 17 (4) | Inf |
| 6 | 0 | 6 (0) | 2 (0) | 8 (1) | 11 (3) | 13 (4) | 18 (5) | 17 (4) | Inf |
| 7 | 0 | 6 (0) | 2 (0) | 8 (1) | 11 (3) | 13 (4) | 18 (5) | 17 (4) | 22 (6) |
| 8 | 0 | 6 (0) | 2 (0) | 8 (1) | 11 (3) | 13 (4) | 18 (5) | 17 (4) | 20 (7) |

note: a (b) means node cost (pred node); node in red means current node; node with grey background means waiting scan node list

So, the shortest path from node 0 to node 8 is 0-1-3-4-7-8, and total cost is 20.

**Q2: Solve shortest path problem in python**

Set the origin and destination node id of agent 1 as 0 and 8 respectively, then run the python code. We can get the result as below, and the result is same as what is manually calculated.



**Q3: Shortest path problem**



Figure 1 Network for shortest path problem

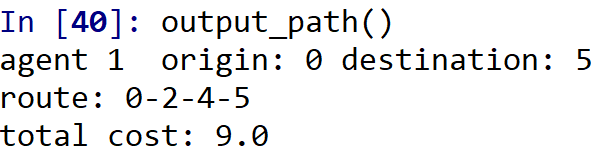
Table 2 Label correcting algorithm calculation process

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| iteration | Node 0 | Node 1 | Node 2 | Node 3 | Node 4 | Node 5 |
| 0 | 0 | Inf | Inf | Inf | Inf | Inf |
| 1 | 0 | 5 (0) | 3 (0) | Inf | Inf | Inf |
| 2 | 0 | 5 (0) | 3 (0) | 6 (1) | 10 (1) | Inf |
| 3 | 0 | 5 (0) | 3 (0) | 6 (1) | 5 (2) | Inf |
| 4 | 0 | 5 (0) | 3 (0) | 6 (1) | 5 (2) | 12 (3) |
| 5 | 0 | 5 (0) | 3 (0) | 6 (1) | 5 (2) | 9 (4) |

note: a (b) means node cost (pred node); node in red means current node; node with grey background means waiting scan node list

So, the shortest path from node 0 to node 5 is 0-2-4-5, and total cost is 9.

Now, set the origin and destination node id of agent 1 as 0 and 5 respectively, then run the python code. We can get the result as below, and the result is same as what is manually calculated.



**Q4: Vehicle routing problem (Group 2)**

The goal of vehicle routing problem (VRP) is planning routes of each vehicle to finish certain tasks (pick up passengers here) with some constraints (capacity constraints here) so that minimum total cost is reached. However, due to the complexity of VRP, it’s really hard to solve it directly especially for larger scale problem. So, we treat this VRP here as two phases like Fisher *(1)*, namely generalized assignment problem (GAP) and travelling salesman problem (TSP), where depots are assigned to specific vehicle in GAP and the route of each vehicle is solved in TSP. In this way, the origin problem is decomposed into two classic problems and the complexity can be significantly reduced.

First, we build the GAP formulation as follow.

(1)

(2)

(3)

(4)

where is the estimated cost if depot is assigned to vehicle (please refer to *(1)* for detailed calculation process, depot 2, 5,9 and 10 are selected as seed). One thing has to be mentioned here is that since the cost is an estimated value, the final results of this method may have tiny gaps compared with optimal solution of origin VRP. is a binary variable ( if depot is assigned to vehicle , otherwise 0). denotes the demand of depot and denotes the capacity of vehicle . Constraints (3) guarantees that all depots are assigned to vehicles. To obtain the optimal solution of the model above, we solve it using gurobi (version 8.0.1) in python environment. The results are shown as below.

Table 3 Assignment results of GAP

|  |  |  |  |
| --- | --- | --- | --- |
| Vehicle | Depot | Total demand | Capacity |
| 1 | 1,2,3,4 | 100 | 100 |
| 2 | 5,6,7 | 98 | 100 |
| 3 | 8,9,10 | 84 | 100 |
| 4 | 11,12,13,14,15 | 100 | 100 |

As can be seen from table 1, all depots have been assigned to vehicles with no capacity constraint violation. Now, what we should do is planning the route of each vehicle so that each vehicle can pass through all depots assigned to it and go back to origin depot, which can be viewed as a TSP for each vehicle. The formulation of TSP for each vehicle is given as below.

(5)

(6)

(7)

(8)

(9)

where denotes the distance between depot and depot . is a binary variable ( if vehicle chooses arc , otherwise 0). Constraint (8) guarantees no sub tours in the final solution, and is a larger enough value. Using gurobi solver in python again, vehicle routes are given in table 4.

Table 4 vehicle routes of TSP

|  |  |  |  |
| --- | --- | --- | --- |
| Vehicle | Route | Cost | Total cost |
| 1 | 0-4-3-2-1-0 | 55.75 | 231.4 |
| 2 | 0-7-6-5-0 | 45.56 |
| 3 | 0-10-9-8-0 | 64.95 |
| 4 | 0-12-11-13-14-15-0 | 65.14 |

To measure the gap between the result of the model above and the global optimal solution, we formulate the entire VRP here.

(10)

(11)

(12)

(13)

(14)

(15)

(16)

(17)

where is a binary variable ( if vehicle chooses arc , otherwise 0). Constraint (16) guarantees no sub tours in each vehicle’s route. By using gurobi in python, we can get the final results as below.

Table 5 vehicle routes of VRP

|  |  |  |  |
| --- | --- | --- | --- |
| Vehicle | Route | Cost | Total cost |
| 1 | 0-4-3-2-1-0 | 55.75 | 231.4 |
| 2 | 0-8-9-10-0 | 64.95 |
| 3 | 0-5-6-7-0 | 45.56 |
| 4 | 0-12-11-13-14-15-0 | 65.14 |

Obviously, the results are same with two-phase model, so the gap for this problem is 0%. Except solutions, we also record the CPU running time for these two models. Two-phase (GAP+TSP) model took 1.2 seconds while VRP model took 4.0 second. Although the difference here was tiny, there would be significant difference if the problem become more complex.

[1] Fisher M L, Jaikumar R. A generalized assignment heuristic for vehicle routing[J]. Network, 2006, 11(2):109-124.