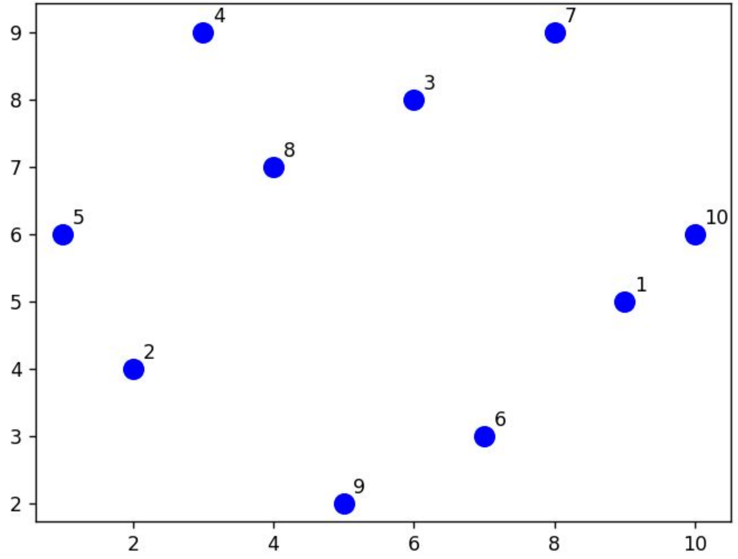
Ant Colony Optimization for a solving a Dynamic Travelling Salesman Problem

CSI 5165 - Project Presentation

Daniel Lobo (300319498)
12 April 2023

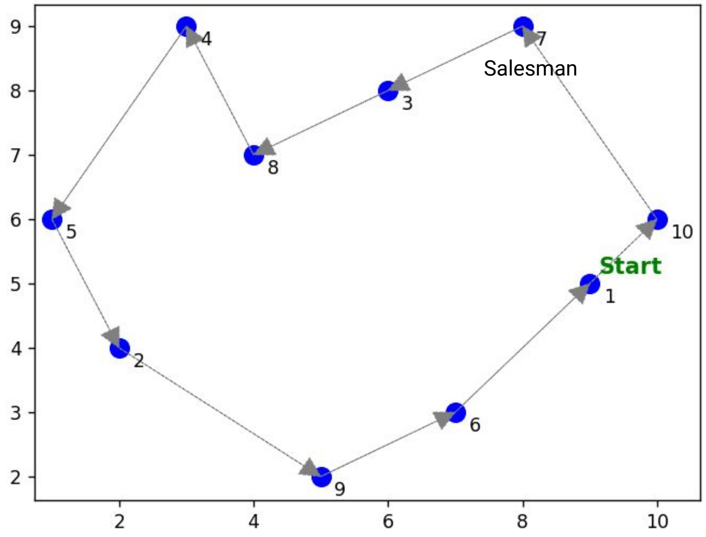
Problem Definition

- **Basic Travelling Salesman Problem**
  - **Given:** A set of cities and their coordinates
  - **Goal:** Find the shortest path that visits each city exactly once.



Problem Definition

- **Dynamic Travelling Salesman Problem**
  - Starts with an initial best path
  - As the salesman progresses, traffic may cause the initial path to not be ideal.
  - Recalculate an optimal sub-path with the following constraints.
    - Initial path start city
    - Visited cities
    - Current city
    - Initial path end city
- **Goal:** Minimize the path cost by dynamically optimizing the initial path based on changing traffic conditions



Methodology - ACO Algorithm

Ant colony algorithm for the TSP problem (ACO–TSP).

Initialize the pheromone information ;
Repeat
  For each ant **Do**
    Solution construction using the pheromone trails:
      $S = \{1, 2, \ldots, n\}$ /* Set of potentially selected cities */
      Random selection of the initial city i ;
      Repeat
        Select new city j with probability $p_{ij} = \frac{\tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}}{\sum_{k \in S} \tau_{ik}^{\alpha} \times \eta_{ik}^{\beta}}$;
        $S = S - \{j\}$; $i = j$;
      Until $S = \emptyset$
  End For
  Update the pheromone trail:
    For $i, j \in [1, n]$ **Do**
      $\tau_{ij} = (1 - \rho)\tau_{ij}$ /* Evaporation */ ;
    For $i \in [1, n]$ **Do**
      $\tau_{i\pi(i)} = \tau_{i\pi(i)} + \Delta$ /* π: best found solution */ ;
Until Stopping criteria
Output: Best solution found or a set of solutions.

Methodology - ACO Algorithm

Ant Colony Optimization - Solution Construction

- **Probability of selecting city j from city i:**

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}}{\sum_{k \in S} \tau_{ik}^{\alpha} \times \eta_{ik}^{\beta}}, \quad \forall j \in S$$

- **Pheromone trails:** τ_{ij} = desirability to have the edge (i, j) in the solution.
- **Problem dependent heuristic:** $\eta_{ij} = 1/d_{ij}$, d_{ij} = distance between cities i and j
- α and β are the relative influence of the pheromone values and the problem-dependent heuristic values.

Methodology - ACO Algorithm

Ant Colony Optimization - Pheromone Update

Evaporation Phase

- All the pheromone trails are reduced by a fixed quantity.

$$\tau_{ij} = (1 - \rho)\tau_{ij}, \quad \forall i, j \in [1, n]$$

τ_{ij} is the pheromone value between nodes i and j.

ρ is the rate of reduction (between 0 and 1)

Methodology - ACO Algorithm

Ant Colony Optimization - Pheromone Update

Reinforcement Phase

- Pheromone values are updated based on the solution generated.

$$\tau_{i\pi(i)} = \tau_{i\pi(i)} + \Delta, \qquad \forall i \in [1, n] \quad \Delta = 1/f(\pi)$$

f (π) is the cost of the best solution found

Methodology - Dynamic TSP Algorithm

- Important Components:
  - **City info:** coordinates of each city
  - **Cost matrix:** cost to travel from city i to city j is the distance between i and j
  - **traffic_factors:** traffic factors to be randomly applied to the cost matrix
    - 1 - No traffic
    - 1.05 - Low traffic
    - 1.2 - Medium traffic
    - 1.5 - High traffic
  - **Updated_cost_matrix:** multiplying each value in the cost matrix with a random traffic factor

Methodology - Dynamic TSP Algorithm

Algorithm

- Calculate the initial cost matrix
- Find the initial path and its cost by running the ACO algorithm
- Start the tour with the 1st city in the initial path
- For each remaining city in the initial path
    - Calculate the remaining cost based on the initial cost matrix
    - Apply the traffic factors to the cost matrix
    - Calculate the remaining cost based on the new cost matrix
    - If the change in cost > 10 %
        - recalculate a new sub path using the ACO algorithm
    - Update the initial path with this new sub path

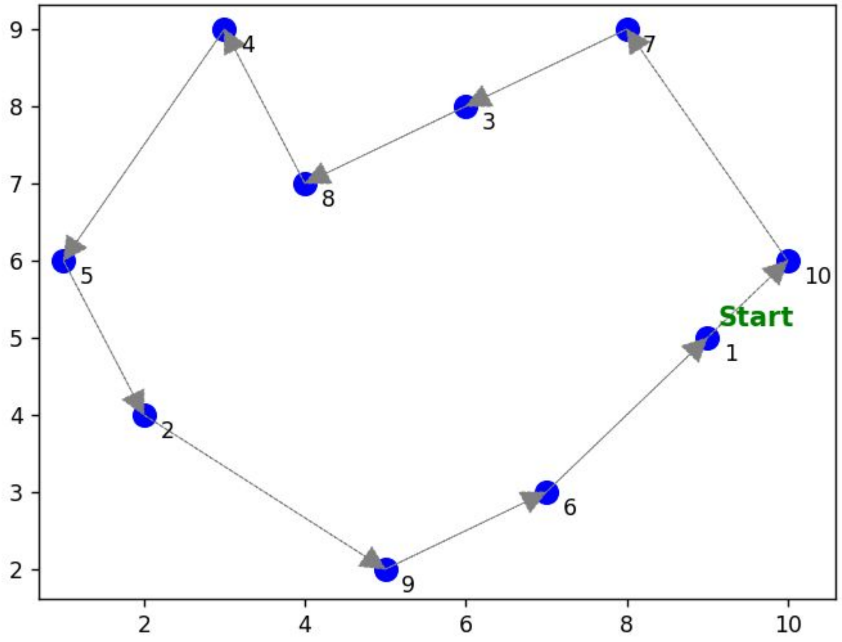
Methodology - Dynamic TSP Algorithm

- ACO parameters used for finding the initial path
  - **num_ants** : 100
  - **max_iterations** : 1000
  - **α** : 1.5
  - **β** : 5
  - **ρ** : 0.75

- ACO parameters used for finding a sub path
  - **num_ants** : 50
  - **max_iterations** : 300
  - **α** : 1.5
  - **β** : 5
  - **ρ** : 0.75

Example

Initial path



Initial Cost Matrix (Distance between 2 cities)

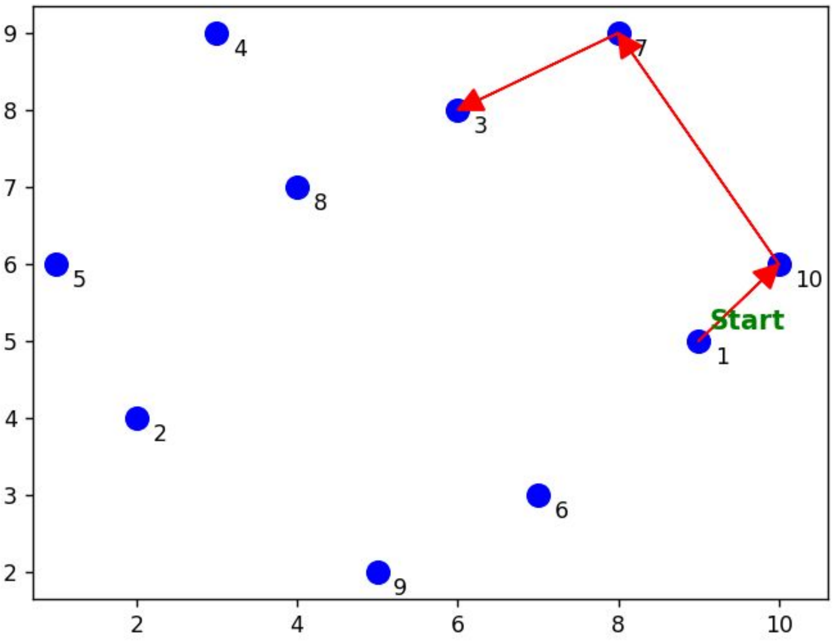
	1	2	3	4	5	6	7	8	9	10
1	0.00	7.07	4.24	7.21	8.06	2.83	4.12	5.39	5.00	1.41
2	7.07	0.00	5.66	5.10	2.24	5.10	7.81	3.61	3.61	8.25
3	4.24	5.66	0.00	3.16	5.39	5.10	2.24	2.24	6.08	4.47
4	7.21	5.10	3.16	0.00	3.61	7.21	5.00	2.24	7.28	7.62
5	8.06	2.24	5.39	3.61	0.00	6.71	7.62	3.16	5.66	9.00
6	2.83	5.10	5.10	7.21	6.71	0.00	6.08	5.00	2.24	4.24
7	4.12	7.81	2.24	5.00	7.62	6.08	0.00	4.47	7.62	3.61
8	5.39	3.61	2.24	2.24	3.16	5.00	4.47	0.00	5.10	6.08
9	5.00	3.61	6.08	7.28	5.66	2.24	7.62	5.10	0.00	6.40
10	1.41	8.25	4.47	7.62	9.00	4.24	3.61	6.08	6.40	0.00

Initial Path: [1, 10, 7, 3, 8, 4, 5, 2, 9, 6, 1]

Initial path cost = 26.23

Example

Starting the tour. At city 3, remaining cost has increased due to traffic.



Updated Cost Matrix (Affected by traffic)

	1	2	3	4	5	6	7	8	9	10
1	0.000	7.4235	6.360	10.815	8.463	2.830	4.120	6.468	6.000	1.692
2	7.4235	0.000	5.660	5.100	2.240	6.120	7.810	3.610	3.610	9.900
3	6.360	5.660	0.000	4.740	5.6595	5.355	3.360	2.240	6.080	4.470
4	10.815	5.100	4.740	0.000	4.332	7.210	7.500	3.360	7.644	8.001
5	8.463	2.240	5.6595	4.332	0.000	7.0455	7.620	3.160	5.660	9.000
6	2.830	6.120	5.355	7.210	7.0455	0.000	6.384	5.250	2.240	4.452
7	4.120	7.810	3.360	7.500	7.620	6.384	0.000	5.364	7.620	3.610
8	6.468	3.610	2.240	3.360	3.160	5.250	5.364	0.000	5.100	6.384
9	6.000	3.610	6.080	7.644	5.660	2.240	7.620	5.100	0.000	6.400
10	1.692	9.900	4.470	8.001	9.000	4.452	3.610	6.384	6.400	0.000

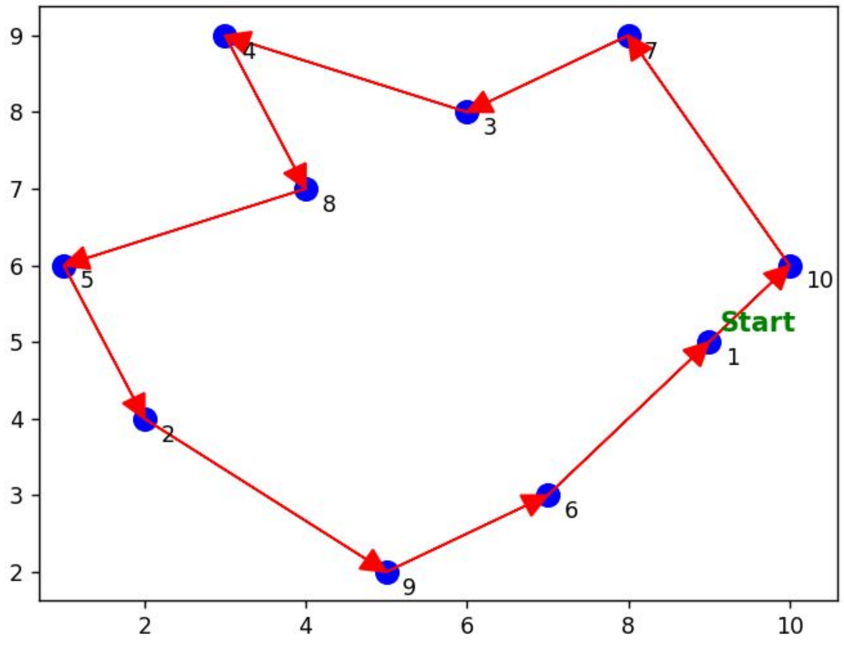
Current Path: [1, 10, 7, **3**, 8, 4, 5, 2, 9, 6, 1]

Initial remaining cost = 19.01

Remaining cost with traffic = 26.39

Example

Recalculated path



Initial path = [1, 10, 7, 3, 8, 4, 5, 2, 9, 6, 1]
New path = [1, 10, 7, 3, 4, 8, 5, 2, 9, 6, 1]

Initial path cost (without traffic) = 26.23
Initial path cost (with traffic) = 30.03
New path cost (with traffic) = 28.89

Experiments and Results

Number of cities	Initial path cost (without traffic)	Initial path cost (with traffic)	Optimized path cost (with traffic)	Number of recalculations
10	26.23	30.03	28.89	3
20(berlin20.tsp)	5928.85	6053.03	5973.52	12
52(berlin52.tsp)	7873.96	9368.43	9001.96	28

The “.tsp” instances are from TSPLIB [6]

Further Work

- Dynamically updating the ACO parameters used for finding sub-paths
  - Progress in path => fewer cities for recalculating sub-paths

- Optimizing the number of times a recalculation for a sub-path is done
  - More recalculations => better quality solutions (less efficient)
  - Less recalculations => More efficient (lower quality solutions)

Conclusion

- Dynamic TSP presents additional challenges compared to static TSP. The problem complexity increases due to the need for continuously updating the optimal tour
- Ant Colony Optimization (ACO) is a powerful metaheuristic algorithm that has been successfully applied to solve various combinatorial optimization problems.
- In this project, I proposed a solution to the dynamic TSP problem using ACO and obtained positive results.
- The algorithm's performance is affected by various factors, such as the pheromone decay rate, the number of ants, and the frequency of the traffic changes.
- Fine-tuning these parameters can further improve the algorithm's performance.

References

[1] M. Mavrovouniotis and S. Yang, ‘Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors’, Appl. Soft Comput., vol. 13, no. 10, pp. 4023–4037, Oct. 2013.

[2] E.-G. Talbi, Metaheuristics: From design to implementation. John Wiley & Sons, 2009.

[3] M. Dorigo, M. Birattari and T. Stutzle, "Ant colony optimization," in IEEE Computational Intelligence Magazine, vol. 1, no. 4, pp. 28-39, Nov. 2006, doi: 10.1109/MCI.2006.329691

[4] M. Guntsch, M. Middendorf, “Pheromone modification strategies for ant algorithms applied to dynamic TSP.”

[5] E. Bonabeau, M. Dorigo, G. Theraulaz, “Swarm Intelligence: From Natural to Artificial Systems”, Oxford University Press, New York (1999)

[6] TSPLIB. (n.d.). Retrieved February 23, 2023, from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/index.html

Thank You