Artificial Intelligence Lab Report 3

1st Dipean Dasgupta
202151188
BTech CSE
IIIT,Vadodara

2nd Shobhit Gupta
202151149
BTech CSE
IIIT,Vadodara

3rd Rahul Rathore
202151126
BTech CSE
IIIT,Vadodara

4th Rohan Deshpande
202151133
BTech CSE
IIIT,Vadodara

***Abstract*—The Travelling Salesman Problem (TSP) presents a challenging optimization task, particularly when applied to planning tours in vast geographical regions such as Rajasthan. In this context, the goal is to devise a tour route that efficiently covers a set of significant tourist locations while minimizing the overall travel cost. Utilizing Simulated Annealing, a non-deterministic search technique, this study aims to generate an effective tour itinerary for visitors exploring Rajasthan. By treating the problem as an optimization challenge where the cost of travel is proportional to distance, Simulated Annealing offers a randomized approach to navigating the extensive search space inherent in TSP. Through this methodology, the proposed solution seeks to balance exploration and exploitation, ultimately producing a cost-effective tour plan tailored to the diverse attractions of Rajasthan.**

I. Introduction

In computer science, one of the most well-known optimisation problems is the Travelling Salesman Problem (TSP). In this game, the objective is to determine the shortest path that passes through multiple cities and ends at the starting place. The TSP's task is to select the best option among a plethora of options, since the number of viable tours grows exponentially with the number of cities. Because of this, the TSP is an NP-hard problem, which implies that it may be computationally costly to discover an exact solution for large instances of the problem. The TSP has a wide range of practical uses, such as in scheduling, logistics, and vehicle routing.

A. Different Approaches For Solving TSP

The Travelling Salesman Problem (TSP) can be solved in a variety of methods, including heuristic algorithms. While heuristic algorithms are quicker, there is no guarantee that the result will be ideal. Heuristic algorithms encompass Ant-colony Optimisation and Simulated Annealing.
Exact algorithms are another option; however, they come with a high computational cost and no guarantee of an optimal solution. Branch Bound and dynamic programming are two examples.

B. Simulated Annealing

Simulated Annealing (SA), a heuristic optimization algorithm, can be used to solve the Traveling Salesman Problem (TSP). It is a probabilistic method inspired by the metal refining process of metallurgical annealing.

The TSP method begins with a random tour and iteratively modifies it with little random modifications. A probability function that strikes a balance between search space exploration and exploitation determines whether new trips are accepted. The technique calculates the probability of adopting a new solution that is worse than the existing one by progressively lowering the temperature. As a result, the algorithm can more successfully explore the search space and avoid local optima. When the temperature hits a particular point or a workable solution is discovered, the algorithm is ended.

SA has been used to a range of tasks, such as scheduling and vehicle routing, and has shown to perform effectively in numerous TSP scenarios. This has the benefit of enabling you to identify optimal solutions in a fair amount of time while striking a balance between exploration and utilisation of the search space. Nevertheless, SA might not always identify the optimal answer and can be computationally costly. The beginning temperature and cooling schedule are two examples of the parameters that can impact the algorithm's performance.

II. Pseudo Code For TSP using Simulated Annealing

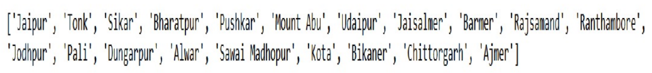
Here is a pseudo code for the problem using simulated annealing.

generate_random_tour() - is a function that takes the set of cities as input and generates a random tour
swap_tour() - is a function that swaps the order of 2 random cities in the current tour
tour_cost() - to calculate the total cost of the input tour

```
simulatedAnnealing (cities, initial_temp, cooling_rate, t_min) {

    present_tour = generate_random_tour(cities)
    best_tour = present_tour
    temp = initial_temp

    while (temp $\geq$ t_min) {
        new_tour = swap_tour(present_tour)
        delta_E = calculate_tour(new_tour) - calculate_tour(present_tour)
        if (delta_E $\leq$0) {
            present_tour = new_tour
            if (calculate_tour(present_tour) $\leq$ calculate_tour(best_tour))
```

```
                best_tour = present_tour
            }
            else {
                prob = $e^{-delta_E/temp}$
                if (random(0, 1) $\leq$ prob) {
                    present_tour = new_tour
                }
            }
            temp *= cooling_rate

        return best_tour
}
```

Results



```
['Jaipur', 'Tonk', 'Sikar', 'Bharatpur', 'Pushkar', 'Mount Abu', 'Udaipur', 'Jaisalmer', 'Barmer', 'Rajsamand', 'Ranthambore',
'Jodhpur', 'Pali', 'Dungarpur', 'Alwar', 'Sawai Madhopur', 'Kota', 'Bikaner', 'Chittorgarh', 'Ajmer']
```

Fig.1 :Calculated Travelling Sequence

$Jaipur- > Tonk- > Sikar- > Bharatpur- > Pushkar- > mountAbu- > Udaipur- > Jaisalmer- > Barner- > Rajsamand- > Ranthambore- > Jodhpur- > Pali- > Dungarpur- > Alwar- > SawaiMadhopur- > Kota- > Bikaner- > Chittorgarh- > Ajmer$

This is the calculated travelling sequence.

Conclusion

Since Simulated Annealing is a heuristic-based method, it is not guaranteed that we will arrive to the best answer right away. By using a greedy algorithm to identify a city tour and then applying simulated annealing to that tour, we can obtain a better solution more quickly. We can see that the trip gets more optimised as we increase the number of iterations. Therefore, increasing the number of iterations in simulated annealing can yield an optimal solution; however, the precise number of iterations needed to produce an optimal solution is unknown.

References

[1] A First Introduction to Artificial Intelligence By Prof. Deepak Khemani IIT Madras.