TSP Solver (Travelling Salesman Problem) using Simulated Annealing (SA)

Bhargavi kallam, EECS, University of Ottawa
bkall059@uottawa.ca

Objective: Implementing a Traveling Salesman Problem (TSP) solver using one of the metaheuristic search algorithms.

Problem Statement: What is the shortest possible route that visits each city exactly once and returns to the initial city, given a list of cities and calculating the distances between each pair of cities?

Abstract: Traveling salesman problem can act as best library to analyze the capacity of techniques over real-time applications as it is difficult to optimize solution over large number of cities. In this assignment simulated annealing metaheuristic is used over the TSPLIB library. The algorithm outputs the optimal cost and optimal tour of cities and to optimize the algorithm to obtain best fitness trail and error technique is used.

Dataset:

The travelling salesman problem library (TSPLIB) is a collection of diverse range of test problems gathered from many sources and with varying attributes. A brief description of each problem is also included. For this assignment, only files with TYPE: TSP and EDGE WEIGHT TYPE: EUC 2D over the symmetric data format.

Simulated Annealing:

Simulated annealing is an optimization approach for finding a reasonable approximation to a function's global minimum. The Simulated Annealing (SA) method starts with a random solution and builds a random close solution with each iteration.

The algorithm starts with a high temperature and gradually reduces it to a low temperature. Cooling down is as simple as creating a loop around a temperature variable and multiplying it by a value between 0 and 1 over each iteration.

```
Temperature = Initial temperature
While Temperature > Stopping_Temperature:
    Calculate the fitness and update the optimal tour solution.
    Temperature = Temperate*alpha
```

Calculating the fitness and to update the optimal solution, set of parameters are used to minimize the fitness over the process of annealing. The probability is determined on the difference between the current and previous cost values, as well as the temperature.

```
New_cost = cost function(solution)
If New_cost < current_cost:
    Update current_solution and current_cost to solution and New_cost
    If New_cost < Best_cost:
        Update Best_solution and Best_cost to solution and New_cost
Else:
    Difference = New_cost – current_cost
    If (random value of 0-1) < (exp (-difference / Temperature)):
        Update current_solution and current_cost to solution and New_cost
```

Motivation: To solve the TSP, I choose the Simulated Annealing Algorithm since it has various parameters that I can tweak to obtain a best solution - the purpose of simulated annealing is to reduce a system's energy (minimizing a cost function).

Implementation:

Simulated Annealing algorithm – Greedy search algorithm technique

Steps followed:

1. Begin with a starting solution, then generate a slightly different solution with each iteration.

2. We accept the iteration if it is distance cost is better. Otherwise, we'll accept it with a probability.

3. We iterate till we reach a stopping criterion.

Parameters used:

1. Initial Temperature: the temperature at which the process begins.
   (Temperature = math.sqrt(length of cities))
2. Stopping Temperature: minimum temperature at which to come to a halt.
   (stop_Temprature = 1e-5)
3. Alpha - a: it is the rate of learning (ranges between 0 and 1)
   (alpha=0.9995)
4. Iteration Stopping: Maximum Iteration
   (stop iter=100000)

Representation:

The neighbors of a problem state, which are new states created by conservatively altering a given state, are evaluated in the optimization of a solution. Each state in the travelling salesman problem is commonly specified as a permutation of the cities to be visited, and each state's neighbors are the set of permutations formed by swapping any two of these cities.

The method to generate random neighboring solutions is to choose two random positions of cities in permutation encoding representation and swap the elements of these permutations.

Used the Nearest Neighbors Heuristic to create an initial tour.

Operations:

While generating a candidate solution, it is quite important to consider that in simulate annealing the current solution should have lower energy comparative to that of random initial solution. So, candidate solutions to be chosen are that having similar energy as current solution. This heuristic is effective as worst moves are less likely to be chosen than good moves.

In this travelling salesman problem, the candidate solutions are randomly generated and the swapping of two consecutive cities to generate neighbor showed the best effect on tour length (shortest path).

Picking two random indices and reversing the order of the tour path between them.

Fitness:

For this Traveling salesman problem in simulated annealing - Fitness of a candidate solution as the tour length.

Evaluation of Solver:

The Evaluation of the tsp_solver can be clearly depicted from the fitness curve which is plotted with “fitness” on Y-axis and “Iterations” on X-axis. The initial and best cost function (length of tour) is indicated as red and green lines respectively.

Correctness:

It is clearly evident from the optimal cost(tour_length) and the fitness curve that the algorithm is optimizing the cost function over iterations. The space (O(n)) and run time complexity is of the algorithm is relatively fair.

The estimated time to execute the algorithm over a280.tsp file is 12secs.