Travelling Salesperson – Simulated Annealing

Sudarshan Rao Mundkur

Computer Science Department, Purdue University

Fort Wayne, IN, 46805

[munds01@pfw.edu](mailto:munds01@pfw.edu)

**Problem Statement**: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?

Given this, there are two important rules to keep in mind:

* Each city needs to be visited exactly one time.
* We must return to the starting city, so our total distance needs to be calculated accordingly.

**Simulated annealing** is a probabilistic technique for approximating the global optimum of a given function. The algorithm begins with a high temperature, and slowly cools down to a low temperature. Cooling down is done simply by having a loop on a temperature variable, and by multiplying this variable by a number between 0 and 1 at every iteration.

**Approach:**

1. Create the initial list of cities by shuffling the input list (i.e., make the order of visit random).
2. At every iteration, two cities are swapped in the list. The cost value is the distance traveled by the salesman for the whole tour.
3. If the new distance, computed after the change, is shorter than the current distance, it is kept.
4. If the new distance is longer than the current one, it is kept with a certain probability.
5. We update the temperature at every iteration by slowly cooling down.

**Working of the program:**

|  |  |
| --- | --- |
| Function Name | Functionality |
| Main() | Function used to initialize all parameters, create Symmetric Distance Matrix and Create a random route. Call simulatedAnnealing() function. |
| simulatedAnnealing(max\_temparature,stop\_temparature, cooling\_rate, stop\_iteration, distMatrix, curr\_solution) | Calculate the distance of random route, create another route by swapping cities using 2-OPT method. Call accept function which returns the weight for the new route.  Increase iteration and reduce the temperature. |
| accept(candidate, distanceMatrix, current\_weight, min\_weight, temp) | Calculate distance for new route. Check if the distance is less than the previous distance to accept, otherwise accept the distance based on the weighted probability of difference in weights by temperature. |
| plotLearning(weights\_list) | Plot graph for number of iterations versus distance calculated in each iteration. |

**Observation / Test-Case:**

* Test case - 1

Please enter starting temperature: 1000

Please enter stopping temperature: .00000001

Please enter cooling rate: 0.995

Please enter threshold/no of iterations: 1000

Please enter number of cities: 80

Intial weight: 8365

Minimum weight: 8048

Chart, histogram

Description automatically generated

* Test case - 2

Please enter starting temperature: 10000

Please enter stopping temperature: .00000001

Please enter cooling rate: 0.5

Please enter threshold/no of iterations: 10000

Please enter number of cities: 100

Initial weight: 10038

Minimum weight: 9782

Chart, histogram

Description automatically generated

* Test case – 3

Please enter starting temperature: 1500

Please enter stopping temperature: .00000001

Please enter cooling rate: 0.90

Please enter threshold/no of iterations: 250

Please enter number of cities: 120

Initial weight: 11751

Minimum weight: 11425

Chart, histogram

Description automatically generated

* Test case – 4

Please enter starting temperature: 1000

Please enter stopping temperature: .00000001

Please enter cooling rate: .995

Please enter threshold/no of iterations: 50000

Please enter number of cities: 100

Initial weight: 9247

Minimum weight: 8913

Chart, histogram

Description automatically generated

* Test case - 5

Please enter starting temperature: 100

Please enter stopping temperature: .01

Please enter cooling rate: .995

Please enter threshold/no of iterations: 1800

Please enter number of cities: 100

Initial weight: 9353

Minimum weight: 9009

Chart, histogram

Description automatically generated

* Test case – 6

Please enter starting temperature: 50000

Please enter stopping temperature: .00000001

Please enter cooling rate: .995

Please enter threshold/no of iterations: 2100

Please enter number of cities: 100

initial weight: 10188

Minimum weight: 9810

Chart

Description automatically generated

**Conclusion:**

After observing the above output results, it seems that the starting, ending temperatures and cooling rate control the convergence pattern. The temperature is a variable whose value reduces during progress of the optimization and therefore, the probability that SA algorithm will accept a worse solution decreases as iteration number increases.

**Comparison of Genetic Algorithm and Simulated Annealing:**

**Simulated Annealing**, when the algorithm begins iterating, T is very large, causes the algorithm to move to every newly created candidate solution, whether better or worse than the current best solution, it is a random selection in the solution space. As iteration number increases and the temperature cools, the algorithm's search of the solution space becomes restrictive to accept only the better candidates.

**Genetic Algorithm**, it generates not a single candidate solution but an entire population of candidates. Genetic algorithm calls the cost function on each candidate solution of the population. It then ranks them, from best to worse. From these ranked values the next population is created. The next population is generated using two methods. 1. Elitism, refers to just taking the highest ranked candidate solutions and passing them to the next generation. 2. Mutation, change in one element in a candidate solution. 3. Crossover operation creating is a new child vector from two parent vectors, whose elements are comprised of some from each of the two parents.

Genetic algorithm returns a better solution, a value close to the solution space's global minimum, but the time and computation costs are higher.

Simulated annealing algorithm provides an optimal solution in less time and at lower computation costs. Depending on the problem and trade-offs (time and computation power), efficient selection for the algorithm can be made.