CSE-404: Task-2 (Group: A1)

SOLVING TSP USING LOCAL SEARCH

The Travelling Salesman Problem (TSP):

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city? It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science. Often instead of providing the distances between the pairs directly, the Euclidean co-ordinates of the cities are given.

In this assignment you will be solving TSP using local search.

Simulated Annealing

You can find the details about Simulated Annealing in your text book.

However, the provided pseudocode in the book is for maximization problems, while TSP is a minimization problem. In this assignment you will be implementing it in the following manner.

```
current = Randomly generated initial state
for (t = 1; ; t++)

    T = schedule (t)
    if (T == 0) return current

    ran = random number between [0,1]
    if (ran <0.5)
            next <- a successor of current generated by “heuristic1”
    else
            next <- a successor of current generated by “heuristic2”

    delE = cost(next)-cost(current)

    if next better than current i.e., delE < 0
            current = next
    else
            current = next only with probability exp(-delE/kT)
```

Scheduling function

The scheduling function (**schedule**) should provide larger T at the beginning and gradually decrease it to 0. If T is lowered slowly enough, Simulated Annealing will find a global

optimal solution with probability approaching 1. However, finding such function is very hard in practice.

You can find many scheduling function in the literature. Some are given below:

Linear cooling scheme: schedule (t) = $T_0 - \eta t$

Exponential cooling scheme: schedule (t) = $T_0 \alpha^t$, here $0<\alpha<1$

Logarithmic cooling scheme: schedule (t) = c / log (1+t)

In this assignment you will be implementing the **linear cooling function.**

Heuristics for generating neighbors

In literature one can find many well-known heuristics to generate neighbors of TSP. Some of them are greedy and some are random. You will implement the following heuristics in this assignment.

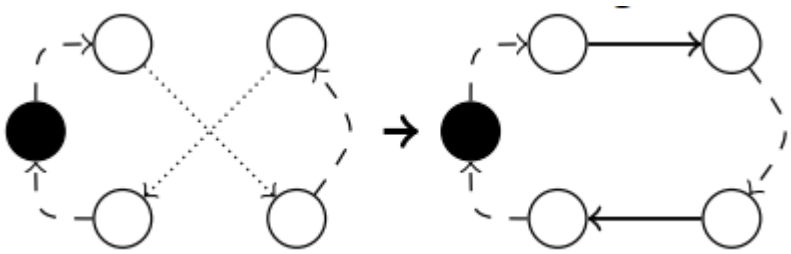
1-1 Exchange

Randomly pick two cities from the route and exchange them. A pseudocode is given below.

```
oneOneExchange(route)
{
      n = length of route;
      ran = random integer between [1, n]
      ran2 = random integer between [1, n] and not equal to ran
      swap the cities at ran and ran2
      return route
}
```

2 – Opt

A 2-opt move removes two edges from the route and reconstructs it such that it does not cross over itself. Try to loop over all pair of edges, until one 2-opt move gives you a better solution. This strategy is called First Improvement Strategy.



```
twoOpt(route)
{
      for each possible pair of edges
            remove the edges and reconstruct route with 2-opt move to get
            new_route
            if new_route is better than route
                  return new_route
      return route
}
```

THE TASKS

1. Implement Simulated Annealing with the heuristic specified above.
2. Run your algorithm with different values of the parameter T_0,η, and k. Each time you will run the algorithm 10 times, compute some statistics and present them as shown in the table. You must prepare the result before submission and show it in hard/soft copy at the time of evaluation.

Problem Instance	T_0	η	k	Average Cost	Minimum Cost	Gap= BKS-Average Cost
att48	val1					
att48	val2					
att48	val3					
burma14	val4					
burma14	val5					
burma14	val6					
burma14	..					

The symbols bear the meaning as discussed in lecture.

Input Format

The first seven lines contain some information about the problem. You need to read the 4th line to get the number of cities *n*. Starting from 8th line, the co-ordinates of the cities are given. The input file ends with a line with the string "EOF".

Output Format

Print the cost (length of the route) of your solution in the first line. The second line will print the solution as a sequence of integers. **Show the resultant route with GUI for bonus marks.**

References:

1. TSP LIB: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
2. TSP Data set: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/
3. Best Known Solutions (BKS): http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html