CS 4633/6633 Programming Assignment 2 **Due October 18**

For this assignment, you will implement the 2-opt local search algorithm for solving the Traveling Salesperson Problem. Then you will improve the performance of this algorithm using either random restarts or simulated annealing.

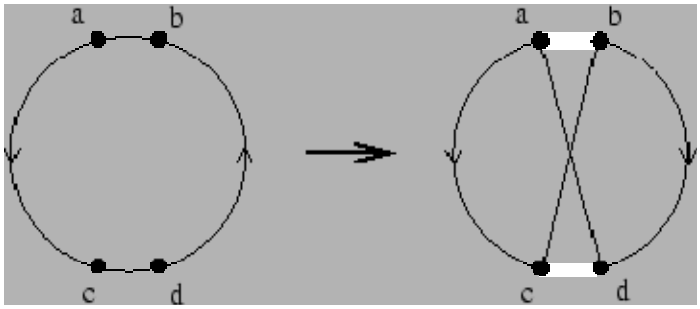
The Traveling Salesperson Problem is the problem of finding a minimum-length tour of N cities that are located in a two-dimensional space. The salesperson must visit every city once and only once, and then return to the starting city.

In principle, an optimal tour can be found using a systematic search algorithm such as A* or depth-first branch-and-bound search. For more than about 25 cities, however, this approach is not practical unless the admissible heuristic is very powerful. Even then, scalability is limited. The Traveling Salesperson Problem belongs to the class of NP-complete problems, and for these problems, it is computationally intractable to find optimal solutions, once the size of the problem grows beyond a certain point.

Local search is a widely-used approach to finding good, though not optimal, solutions to NP-complete search problems. A local search algorithm starts with an initial (possibly random) solution and then repeatedly makes small changes that improve the solution until some convergence criterion is satisfied.

2-opt

A well-known local search algorithm for the Traveling Salesperson Problem is called 2-opt. The basic step of 2-opt is to delete two edges from a tour and reconnect the remaining fragments of the tour by adding two new edges. See the figure below for an example. Once we choose the two edges to delete, we do not have a choice about which edges to add – there is only one way to add new edges that results in a valid tour.



Example of 2-opt

The 2-opt algorithm repeatedly looks for 2-opt moves that decrease the cost of the tour. A 2-opt move decreases the cost of a tour when the sum of the lengths of the two deleted edges is greater than the sum of the lengths of the two deleted edges. Note that a 2-opt move is the same as inverting a subsequence of cities in the tour. For example, in the picture above, the initial tour is *a, …, c, d, …, b, a* while the final tour is *a, …, c, b, …, d, a.* So the segment *d, …, b* is inverted in the modified tour.

Here is pseudcode for the 2-opt local search algorithm:

```
current_tour := create_random_initial_tour()
repeat
    modified_tour := apply_2opt_move (current_tour)
    if length(modified_tour) < length(current_tour)
    then current_tour := modified_tour
until no further improvement or a specified number of iterations
```

Although the 2-opt algorithm performs well and can be applied to Traveling Salesperson problems with many cities, it has a serious drawback – it can become stuck in local minima. One way to make local search less susceptible to getting stuck in local minima is called simulated annealing. Another is called random restarts. The second part of your assignment is to implement one or both of these.

Simulated annealing

In the original version of 2-opt, we only accept a modification of a tour if it decreases the length of the tour. In simulated annealing, we sometimes allow a modification that increases the length of the tour. We accept

a modification that creates a longer tour with a certain probability that decreases with the proposed increase in tour length. We also gradually reduce this probability over time, in order to rule out shorter and shorter path increases – hopefully allowing convergence to a path length close to the minimum. Here is pseudocode for the simulated annealing version of 2-opt.

```
set initial temperature
current_tour := create_random_initial_tour()
repeat
    modified_tour := apply_2opt_move (current_tour)
    if length(modified_tour) < length(current_tour)
    then current_tour := modified_tour
     else if random[0,1) < e^((length(current_tour) – length(modified_tour))/temperature)
     then current_tour := modified tour
    decrease temperature
until temperature is almost zero
```

In the above procedure, the procedure for setting the initial temperature and gradually reducing it is called the *cooling schedule* and it critically affects performance. If the cooling schedule is properly done, then in theory, simulated annealing converges towards a global minimum. In practice, however, it is not always easy to know what the best cooling schedule is.

For this assignment, it may be reasonable to set the initial temperature to 2000, decrease it by one percent each iteration, and stop when the temperature goes below 0.01. But feel free to experiment with other cooling schedules.

Random restarts

Another way to avoid getting trapped in local optima is called random restarts. Because the local optimum in which an algorithm gets trapped depends strongly on the initial solution with which the algorithm starts, restarting the algorithm with different initial solutions may result in a better final solution. The idea is to restart the local search algorithm from several different random initial solutions, and keep the best final solution. Here is the pseudocode for this approach.

```
set number_of_restarts to a reasonable number
best_tour_found_so_far := null
length_of_best_tour_found_so_far := infinity
repeat
        current_tour := create_random_initial_tour()
        repeat
           modified_tour := apply_2opt_move (current_tour)
           if length(modified_tour) < length(current_tour)
            then current_tour := modified_tour
        until no further improvement or a specified number of iterations
        if length(current_tour) < length_of_best_tour_found_so_far
        then best_tour_found_so_far := current_tour
             length_of_best_tour_found_so_far := length(current_tour)
until number_of_restarts has been exceeded
```

Evaluation and report

On MyCourses, you will find a number of test instances of the Traveling Salesperson Problem as well as a parser that reads a test instance and creates a data structure for the problem that can be used by the local search algorithm you implement.

Run your implementation on at least 10 instances of the Traveling Salesperson Problem having at least 50 cities. Compare the performance of 2-opt by itself to the performance of 2-opt using simulated annealing and/or random restarts. Plot the average performance of each on a X-Y graph that shows tour length on the Y-axis and time (or number of iterations) on the X-axis. Turn in a one-page report that describes what you did and analyzes the results. Also turn in a copy of your code.