The Traveling Salesman Problem: Solution with Simulated Annealing

Haina Wang

September 6, 2018

1 Introduction

The Traveling Salesman Problem is a classic excercise in computer science. The version considered in this report refers to the following question:

n cities are fixed on a square map. A salesman must visit every city exactly once and return to the starting city. What is shortest possible route given a particular configuration of cities?

The brute-force algorithm which lists down all possible routes has complexity $\mathcal{O}(n!)$. There are actually $(n-1)!/2$ possible routes for n cities, if we disregard the direction of a route.

The algorithm employed here is "simulated annealing", inspired by annealing in metallury. In general terms, it is a probabilistic technique to optimize a function, in which the solution space is explored step by step; moves that lead to better solutions are always accepted, whereas moves that lead to worse solutions are accepted with a certain probability. The probability for accepting bad solutions is originally set to be high, to allow for a thorough exploration of the solution space; it is slowly decreased to essentially zero to arrive at an approximation of the optimal solution.

2 Method

The access of the Java code used for the simulation is given in the Appendix. The essitial steps are as below.

1. Create a square 2D map with arbitrary dimensions 100×100. Put n cities on the map. The locations of the cities can be either pre-defined or randomly set according to the uniform 2D probability distribution function.

2. Construct a original random route and calculate the total route length. The total route length is termed "energy" in the program.

3. Randomly swap two cities along the route and re-calculate the energy. In this program, we *do not eliminate* identity swaps, *i.e.* cases where a city is swapped with itself or the swaps which simply reverse the direction of the traverse, *e.g.* $ABCDA$ and $ADCBA$ in the four-city scenario.

4. If the new route generated by the swap has an energy lower or equal to the previous route, it is always accepted. If the new route has higher energy, it is accepted with a Boltzmann probability

$$p = \exp(\frac{-(E_{\text{new}} - E_{\text{old}})}{T})$$

where T is an arbitrary parameter called "temperature". If a route is not accepted, the old route before the swap is resumed and another swap is attempted.

The swapping continues as the temperature drops. The initial temperature T_0 is initially set to be high and reduced linearly. **The "cooling rate" r is measured as the decrease of T per 100 accepted swaps.** The cooling is terminated when $T < 0.01$. The route at the end of cooling is regarded as the final solution.

5. The lowest-energy route that appears along the simulation procedure is also recorded and compared to the final solution.

3 Results and Discussion

3.1 Algorithm validation with five cities

On the 100×100 map, the algorithm is tested on 5 predefined cities $A(-20.0, 10.0)$, $B(30.0, 30.0)$, $C(20.0, 10.0)$, $D(20.0, -10.0)$, $E(-20.0, -10.0)$. It is possible to list all routes, as shown in Table 1.

Using the cooling schedule $T_0 = 100.01$, $r = 1.0$ (meaning T decreases by 1.0 per 100 accepted swaps), the optimal route with energy 156.2 is always achieved. Other configurations with five cities are tested and the algorithm is always capable of finding the shortest route.

3.2 Solutions for more cities

Fig. 1 shows minimum-energy solutions for 5, 25, 50 and 100 randomly located cities, respectively. The cooling schedule is $T_0 = 1000.01$, $r = 0.01$.

It should be noted that except for the case of 5 cities, the final route achieved at the end of cooling is *not* always the minimum-energy route encountered along the course of simulation. However, the energy difference is always $< 1\%$. Therefore, it is not unreasonable

Table 1: The energies of all routes for five predefined cities in Section 3.1.

Route	Energy
$ABCDEA$	156.2
$ABCEDA$	205.6
$ABDCEA$	179.8
$ABDECA$	203.3
$ABEDCA$	217.9
$ABECDA$	227.3
$ACBEDA$	211.1
$ACEBDA$	257.1
$ACDBEA$	185.2
$ACBDEA$	163.5
$ADBCEA$	173.0
$ADCBEA$	171.1

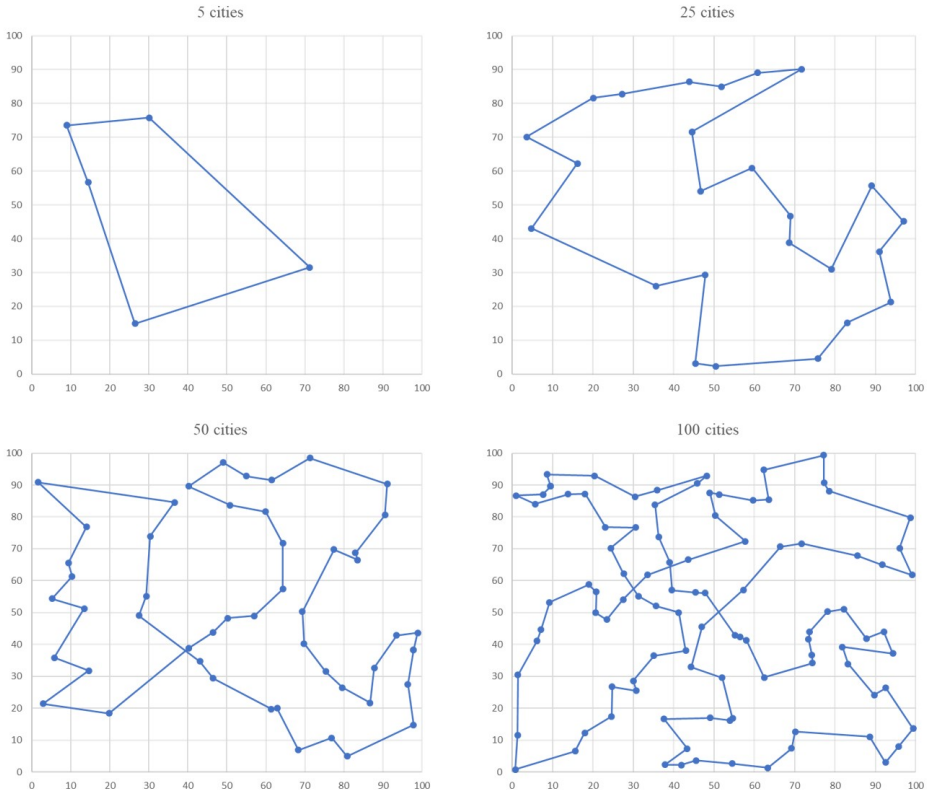

Figure 1: Solutions for 5, 25, 50 and 100 cities.

Table 2: The proportion of times f where the final route is the record minimum-energy route under the cooling schedule $T_0 = 1000.01$, $r = 0.01$.

Number of cities	f
5	1.00
25	0.57
50	0.48
100	0.62

Table 3: Run time for different number of cities under the cooling schedule $T_0 = 1000.01$, $r = 0.1$.

Number of cities	Run time (s)
12	0.489
25	0.717
50	1.26
100	2.33
200	4.45

to conclude that the final solution, which is a local optimum, is a good approximation of the global optimal solution.

Given different initial configurations, the proportion of times f that the final route is the record lowest energy route provides a measure of the effectiveness of the simulation. Table 2 gives this proportion, computed from 100 initial configurations for 5, 25, 50 and 100 cities, respectively, under the cooling schedule $T_0 = 1000.01$, $r = 0.01$.

f generally decreases as the number of cities increases, because the more complicated the solution space is, the less chance there is to arrive at the global minimum-energy solution, which can be occasionally visited during the simulation process. However, f is abnormally high for 100 cities, probably because in this case the solution space is so complicated that many low-energy routes, including the global optimum, are never visited.

Time complexity

Table 3 shows the run time for different number of cities under the cooling schedule $T_0 = 1000.01$, $r = 0.1$. Approximate linear time complexity can be observed.

3.3 Dependence on the cooling schedule

The initial temperature T_0 and the cooling rate r are expected to affect the effectiveness of the algorithm in finding minimum-energy solutions.

Experiments on 10 and 25 cities show that the value f is has little correlation with T_0 as long as $T_0 \geq 0.1d$, where d is the side length of the square map. As expected, f has a slight negative corrlation with r, *i.e.* the slower the cooling, the more likely that the global minimum is achieved.

The exact relationship between T_0, r and f is not mathematically clear and needs to be further investigated.

4 Conclusions

The simulated annealing algorithm with a Boltzmann probability function presented here is able to effectively find approximate optimum solutions for the Traveling Salesman Problem. If the "cities" in this problem refer to points in some interesting spaces (*e.g.* DNA sequences), this algorithm may find many practical applications.

Appendix

The .java files for this project are uploaded onto `https://github.com/Studio-Darboux-Carbonnier/Salesman`

The "main" function is contained in the file Salesman.java.