Study of the travelling salesperson problem

Li Miao
miao0044

April 27, 2021

1 Introduction

In this report, I will apply simulated annealing (SA) and genetic algorithm (GA) to travelling salesperson problem (TSP). TSP is a typical NP-hard problem, and hence many stochastic optimization algorithms have been proposed to solve it. Among the algorithms, SA and GA are the most popular and simple two. This report will discuss the implementation detail and performance.

2 Literature Review

2.1 Travelling Salesperson Problem

The travelling salesperson problem (TSP) says that, a salesperson, who makes a round trip visiting all the other cities and return home, wants to minimize the total distance. Note that each city must be visited only one time. Also we assume that the distance of each pair of cities is the shortest distance, i.e., the Euclidean distance, between them. TSP regards with many real applications in reality including distributing problem [1]. Hence, it is an important problem in operation research and computer science. However, it is a typical NP-hard problem, which cannot be solved in polynomial time.

From a broader view, TSP is a kind of 'combinatorial optimization' problems [2], in which a discrete set of choices result in a pay-off which is to be maximized or a cost which is to be minimized. There is a big class of stochastic optimization algorithms [3], which refers to a collection of methods for minimizing or maximizing an objective function when randomness exists. These algorithms includes Monte Carlo estimation, minimal spanning trees, linear programming, simulated annealing (SA), genetic algorithm (GA) and others. The key issue of them is how to define 'neighbour' in search procedure. A good definition can help to search the feasible space efficiently. Below we will work on SA and GA.

2.2 Simulated Annealing

SA is useful in approximating global optimum in the presence of large numbers of local optima. The word, 'Annealing', refers to an analogy with thermodynamics, specifically with the way that metals cool and anneal. In SA, the objective function is like the energy of a material. As the time of objective decreasing grows, the algorithm tends to stop searching.

[4] gives a detail discussion of the convergence property, time complexity and other analysis of SA. To be short, Implementation of SA is surprisingly simple. The algorithm is basically hill-climbing except instead of picking the best move, it accepts a random move according to some probability [4, 5]. Specifically speaking, if the selected move improves the solution, then it is always accepted; otherwise, the algorithm makes the move anyway with some probability depending on the present value of objective function. The probability decreases exponentially with the 'badness' of the move, which is the amount that the objective function worsened. The probability can be written mathematically as below:

$$p(\text{accept}) = \exp(-\frac{E' - E}{T}), \text{if } E' > E$$

in which T is the temperature controlled the probability. As the algorithms runs, T decreases and the probability to accept a worse solution is lower.

2.3 Genetic Algorithm

GA is a big family of algorithms which base on the natural selection and genetic mechanism in biology. It has wide applications including optimization, automation and machine learning [6, 8]. Its key components regard: coding (to represent the solution with machine codes), crossover (to generate a group of new solutions with two selected solutions, like the children of them), mutation (randomly change some part of children), and selection (select two best solutions for further genetic procedure). Repeat the four components will give a satisfy result. [7] gives a detail discussion of its time complexity.

The simplest version of GA has some shortcomings, like premature (convergence too quickly due to lack of diversity) and local minimum traps. Since the four components are flexible, GA can be adapted into many other versions. [9] compares different variants of how to select the best individual solutions; [10] proposes grouping version of GA; and [11] compares different cross-over and mutations rate. In a word, hyper-parameters of GA must be carefully chosen given the different tasks.

3 Implementation

Since SA and GA have many variants, in this section, I briefly introduce my version with pseudo-code. Then I discuss some details in implementation.

Pseudo-code of SA is listed in 1. It is composited by nested loop. With the outer loop going, the temperature decreases, and hence the accept ratio of a worse solution decreases until convergence.

Algorithm 1 TSP-SA($D, \alpha = 0.95, n = 1000$)

Require: D is a distance matrix of shape $N \times N$

```
T ← max(D)
T' ← min(D[D ≠ 0])
s ← permutation([1, ··· , N])
E ← tour(D, s)
E* ← E, s* ← s
while T ≥ T' do
  for i from 1 to n do
    s' ← generate_new_sol(s)
    E' ← tour(D, s')
    if E' < E then
      E ← E', s ← s'
      if E' < E* then
        E* ← E', s* ← s'
      end if
    else
      if exp(−(E' − E)/T) > rand() then
        E ← E', s ← s'
      end if
    end if
  end for
  T = T * α
end while
```

In SA, the hyper-parameters contain the initial temperature, the ending temperature, the maximum iterations at each temperature and the decay ratio. However, manually setting them for each dataset seems to be not that fancy. In this project, I set the initial temperature to be the maximum distance between two cities, and the ending temperature to be the minimum. Besides, I fix the decay ratio to be 0.95 and maximum iterations at each temperature to be 1000. In my experiment, I find them work well.

Pseudo-code of GA is listed in 2. It seems simpler. In each iteration, mutation, selection and crossover are executed sequentially.

Algorithm 2 TSP-GA($D, n = 200$)

Require: D is a distance matrix of shape $N \times N$

```
m ← 20 * N
s ← permutation([1, ··· , N])
E ← tour(D, s)
E* ← E, s* ← s
for i from 1 to n do
    for j from 1 to m do
        g[j] ← mutate(s)
    end for
    s1, s2 ← select(g)
    s ← crossover(s1, s2)
    s* ← bestOf(s1, s2, s)
    E* ← tour(s*, D)
end for
```

In GA, the hyper-parameters contain the maximum generations and the group size. For the same reason, I set the group size to be 20 times of the number of cities. Besides, I fix the maximum generations to be 200. In my experiment, I find them work well.

Note that the 'generate_new_sol()' function in SA and the 'mutate()' in GA is indeed the same. To modify from a current solution of TSP, there are three ways: randomly swap two cities, randomly reverse a sequence of cities, or randomly delete a city and insert it elsewhere. In SA, I choose the reverse method, and in GA, I choose the swap method.

Also note that 'select()' and 'bestOf()' in GA have similar implementation. They both requires to rank some solutions according to the total tour distance. In other words, the 'fitness' of each individual solution is defined to be the negative of total distance.

Besides, the 'crossover()' in GA takes two solutions as inputs and generate their 'children'. In detail, it randomly clips a sub-sequence from the first solution and fill the left spaces with the same order in the second solution (see figure 1).

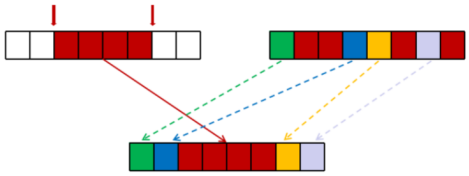


Figure 1: crossover

4 Experiments

In this project, I use python to implement the two algorithms. As for third party package, I only use 'Numpy' (for math calculation) and 'Matplotlib' (for plotting). Just execute 'python tsp.py tsp/five_d.txt' (the final arguments can be changed, any dataset that contains NxN distances can be solved by my script) to see the final result.

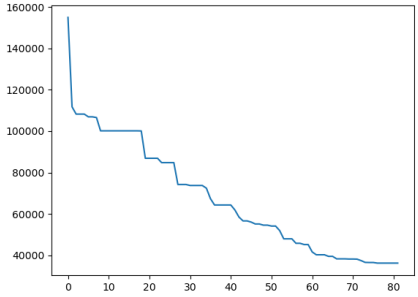
Datasets are from 'https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.html'.

The following table 1 listed the dataset together with given shortest tour distance and the solution I found.

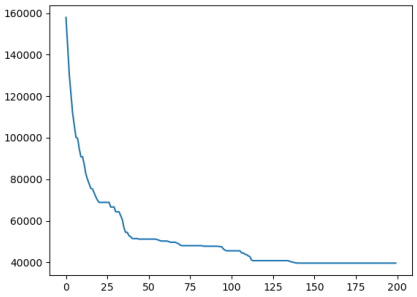
Table 1: TSP solved by GA and SA

dataset	gt	SA	GA
ATT48	33551	36077	39634
DANTZIG42	699	733	950
FIVE	19	19	19
FRI26	937	953	1164
GR17	2085	2088	2153
P01	291	303	307

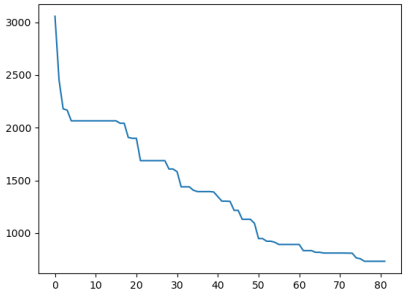
The searching curve of SA and GA for each datasets are shown below:



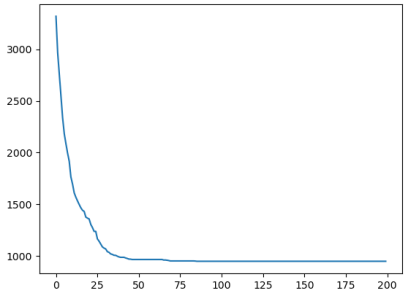
(a) ATT48-SA



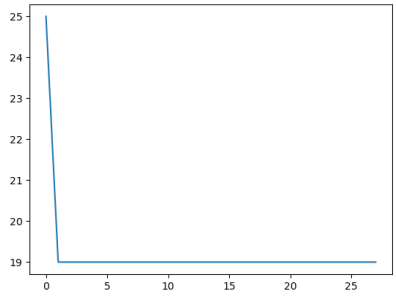
(b) ATT48-GA



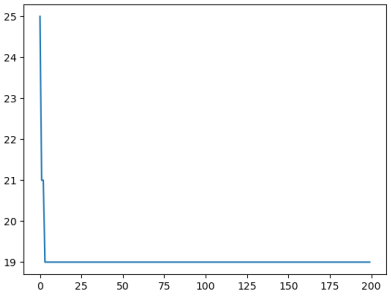
(c) DANTZIG42-SA



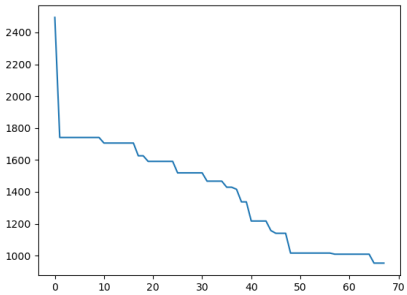
(d) DANTZIG42-GA



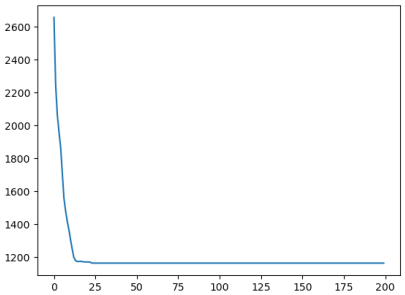
(e) FIVE-SA



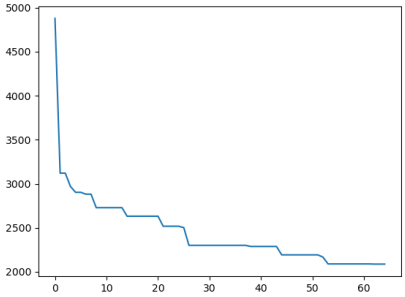
(f) FIVE-GA



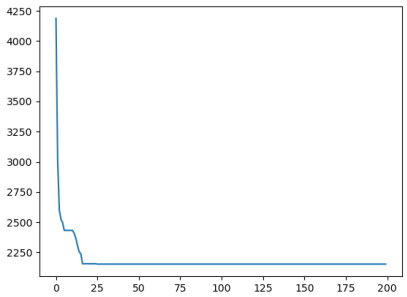
(g) FRI26-SA



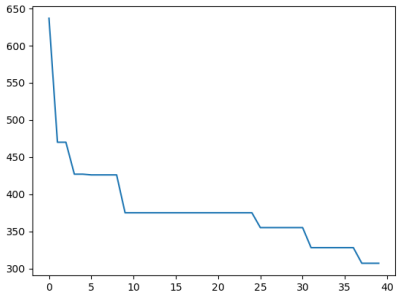
(h) FRI26-GA



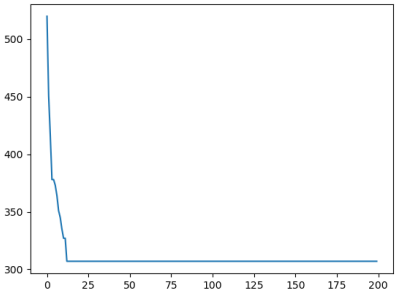
(i) GR17-SA



(j) GR17-GA



(k) P01-SA



(l) P01-GA

Figure 2: Searching curves

According to the experimental result, we can see that 1) Both SA and GA can find satisfied solution, and SA seems to be better than GA; 2) my method to set the hyper-parameters enable a careful and efficient search of solution space, and can reach the satisfied solutions.

References

[1] Van Buer M G, Woodruff D L, Olson R T. Solving the medium newspaper production/distribution problem[J]. European Journal of Operational Research, 1999, 115(2): 237-253.

[2] Lawler E L , Lenstra J K , Kan A , et al. The Traveling Salesman Problem; A Guided Tour of Combinatorial Optimization[J]. Journal of the Operational Research Society, 1985, 37(5):535-536.

[3] Kleywegt A J , Shapiro A . Stochastic Optimization. International Encyclopedia of the Social & Behavioral Sciences, 2001.

[4] Kirkpatrick S. Optimization by simulated annealing: Quantitative studies[J]. Journal of statistical physics, 1984, 34(5): 975-986.

[5] Aarts E H L, Van Laarhoven P J M. Simulated annealing: an introduction[J]. Statistica Neerlandica, 1989, 43(1): 31-52.

[6] Goldberg D E . Genetic Algorithm in Search, Optimization, and Machine Learning[M]. Addison-Wesley Pub. Co, 1989.

[7] Sutton A, Neumann F. A parameterized runtime analysis of evolutionary algorithms for the Euclidean travelling salesperson problem[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2012, 26(1).

[8] Kumar M, Husain M, Upreti N, et al. Genetic algorithm: Review and application[J]. International Journal of Information Technology, 2010, 2(2): 451-454.

[9] Shukla A, Pandey H M, Mehrotra D. Comparative review of selection techniques in genetic algorithm[C]//2015 international conference on futuristic trends on computational analysis and knowledge management (ABLAZE). IEEE, 2015: 515-519.

[10] Singh A, Baghel A S. A new grouping genetic algorithm approach to the multiple travelling salesperson problem[J]. Soft Computing, 2009, 13(1): 95-101.

[11] Patil V P, Pawar D D. The optimal crossover or mutation rates in genetic algorithm: a review[J]. International Journal of Applied Engineering and Technology, 2015, 5(3): 38-41.