Implementation and Analysis of Heuristic Approach on Traveling Salesman Problem

Abstract

In this work, I utilize some random optimzation algorithm on TSP, an NP-hard problem, to achieve an acceptably good results within bounded time. Further, I analyze the performance between different combinations of policy, drawing the conclusion from experiment results that with sufficient iterations, simulated annealing will converge to stationary distribution of Markov Chain, turning out to be a optimal solution, even a global optimal for simpler cases.

1 Problem Restatement

Given undirected graph$G = (V, D)$, for all the edges $D = [d_{ij}]$, we have:

- $d_{ij} > 0$
- $d_{ii} = 0$
- $d_{ij} = d_{ji}$
- $d_{ij} + d_{jk} \geq d_{ik}, 1 \leq i, j, k \leq n$

Instead of defining some decisive variables to formulate the integer programming problem, we utilize permutation to describe the solution of TSP, i.e., a permutation of 1, 2, ..., n.

We define ith vertices as π_i, thus we have $\pi = (\pi_1, \pi_2, ..., \pi_n)$, representing a loop path: $\pi_1 \rightarrow \pi_2 \rightarrow ... \rightarrow \pi_n \rightarrow \pi_1$.

Let S be the set of all feasible solutions, where $|S| = \frac{(n-1)!}{2}$.

Thus we have the optimization problem formulated as below:

$$f(\pi) = \sum_{i=1}^{n} d_{\pi_i, \pi i+1},\ where\ \pi_{n+1} = \pi_1$$

$$\min f(\pi^{(i)}), \pi^{(i)} \in S$$
$$s.t.\ \pi^{(i)}\ is\ a\ permutation\ of\ \pi_1 ... \pi_n$$

We define a neighborhood $\mathcal{N}$ of 2-exchange operation.

2 Data Structure and Organization.

- **Input Data**: ($2 \times Cities$, representing the x, y coordinates of certain cities).
- **Exchange cities**: directly change the target row, or change the index sequence using randperm() function (sort of that).
- **Output Value**: optimal objective function value $f(\pi_{best})$, as well as the sequence of π_{best}.

3 Implementation of Heuristic Algorithm

3.1 Algorithm Description

Solution could be devided into three main parts: solution space, objective function and initial (feasible) solution:

- **Initialization**: Set the initial temperature T_0, which is high enough. Find out a random permutation of all cities as our initial solution (which satisfies the is restriction and is always workable in this TSP problem). We need to set the max iterations on every temperature (max_iter) as well as the aggregate temperature points (N) based on certain annealing policy.
- Pseudo code is illustrate below:

Algorithm 1 Simulated Annealing Algorithm

```
Require: Initialization as above, $\pi_{best} = \pi_0$.
1: while t=1, 2, ... N do
2:     while i=1, 2, ... max_iter do
3:         Generate new solution $\pi^{(i)}$
4:         $\Delta f = f(\pi^{(i+1)}) - f(\pi_{best})$
5:         if $\Delta f < 0$ then
6:             $\pi_{best} = \pi^{(i+1)}$
7:         else
8:             if $rand < exp(-\frac{\Delta f}{T})$ then
9:                 $\pi_{best} = \pi^{(i+1)}$
10:            end if
11:        end if
12:    end while
13:    $T = \alpha * T$ // Annealing Policy
14: end while
```

3.2 Genetic Algorithm

The basic procedure of GA could be summarized as 6 phases[1]:

- **Initialization**: set the improvement threshold (maximum steps to stop with no improvement).
- **Parental selection**: Select parental chromosome based on the current distribution.
- **Mating**: Mate the chromosome to generate the children. Two parents generate two children.
- **Variation**: Random variation on individuals based on given probability.
- **Evaluation**: Evaluate the fitness function of each individuals within the new generation.
- **Compare**: Update the best solution π_i.

3.3 Related Parameter Selection

- Simulated Annealing:
  - **Initial Temperature**: T_0 should be relatively high, set as 100.
  - **Max Iteration**: 500, 1000, 2000 for comparison, turns out with a more stable convergence when iterations go up.
  - **Steps of temperature**: the temperature should go through as many groups of value as possible so that the annealing process would go down to a sufficiently low-energy state (hopefully the global optimal solution).
- Genetic Algorithm:
  - **Variation probability and Cross probability**: 0.3-0.4 for cross and 0.6-0.8 for variation would be ideal, but it depends on the concrete problem after all. Still, there is a clear trade-off between **exploitation and exploration**, i.e. we should utilize the past results while maintaining the ability to generate new solutions, which could lead to even better ones.
  - **Steps of convergence**: if the result does not improve in the recent 3000 stpes, then we stop the global iterations of GA.
  - **Population size**: we select 20000 individuals as the entire population. Usually larger population leads to slower population and better diversity (which is very important to generate the optimal result).

4 Analysis on the Distribution of Solution

To analyze the probability distribution of optimal solution, we run the code for 2000 times, and calculate the distribution as below:

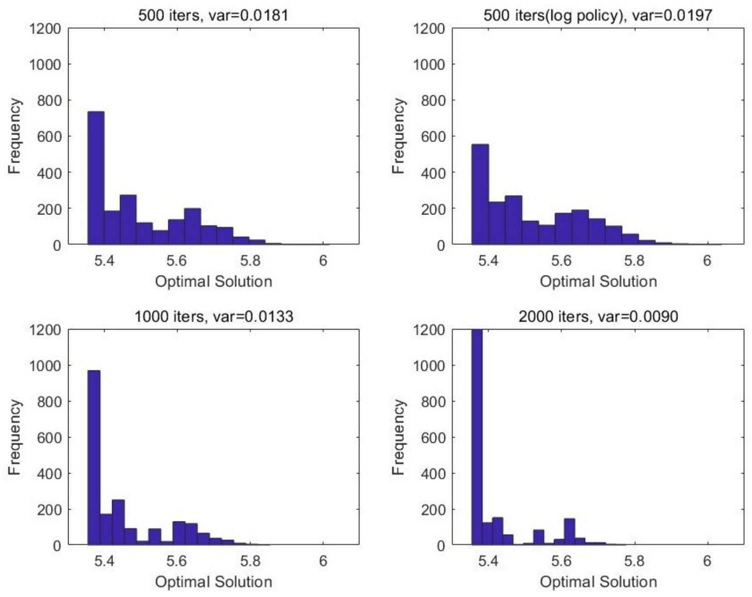


Figure 1: Distribution of Solutions of SA

Here we observe that when iterations go on, the optimal function value would tend to converge on the global optima, while a local-optimal solution between 5.6-5.7 is clearly observed. Generally it looks like a Boltzman Distribution with most solutions lying at state with the lowest energy. When iterations go on, more solutions would be stationary distribution.

When we rise the iterations more than 2000, we would get most of the value close to the global optimal value. This is conherent with the stationary distribution of Homogeneous Inreducible Markov Chain[2], where

$$\lim_{t\to\infty}\pi = \lim_{t\to\infty}(\pi_1, \pi_2, ..., \pi_n) = (1, 0, ..., 0)$$

Therefore, we have almost all the value to optimal one if we repeat for enough times here with proper parameter selection.

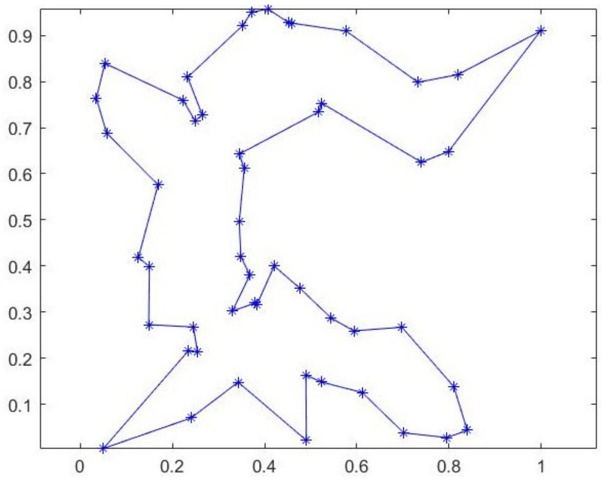
5 Comparison on Results

On the dataset of x50 and x500, we get the respective results for both approaches:

Dataset	SA1	SA2	SA3	GA
x50	5.3550	5.3550	5.3550	6.0112
x500	18.7751	17.8089	17.2368	—-

SA1	Simulated Annealing with policy that $t_{i+1} = \alpha t_i$
SA2	Simulated Annealing with policy that $t_{i+1} = t_i(1 - \frac{1}{50+log(i)})$
SA3	Simulated Annealing with policy that $t_{i+1} = t_i(1 - \frac{1}{50+log(10+log(i))})$
GA	Fixed Prob.: cross=0.3, variation=0.7

Final effect of best results:


(a) 50 cities

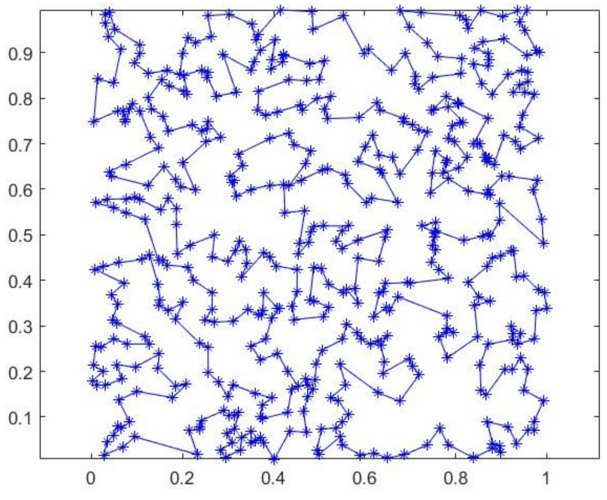

(b) 500 cities

Figure 2: We could easily get global opimal with SA on x50.mat, but x500.mat could hardly be evaluated with our eyes, still the current optimal is not far from the opimal result.

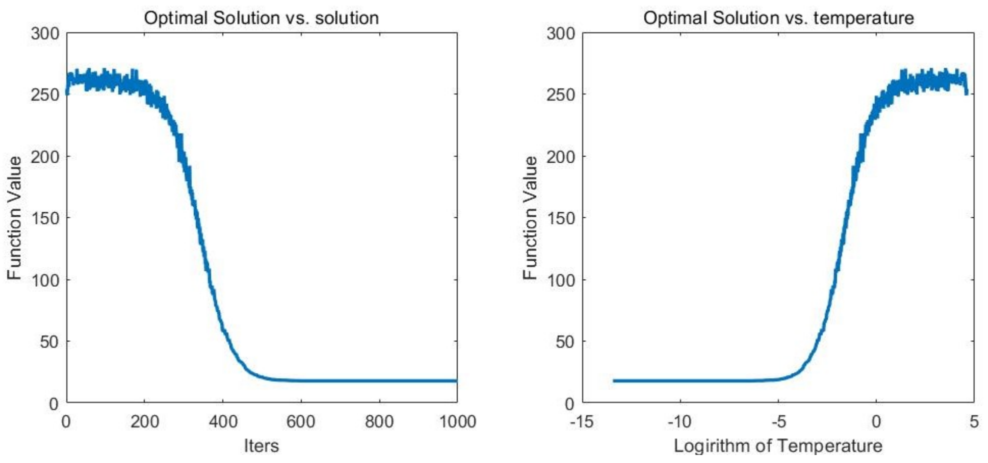


Figure 3: SA training record.

6 Reflection

GA does not work pretty well, it turns out that tuning the probability of crossing and variation is a big problem. Fine-tune takes a lot of time with considerably long computation time! So it's a wise approach to utilize adaptive stratgy to tune the probabilities. When I look into reference it's a little bit late (after my final of another course...). Still I'd like to illustrate it here without realizing (why not try it in the future):

- Strategy1: Whole-race adaptive, we evaluate the distribution of the entire race and fine-tune the variation and cross-over probability dynamically:

$$P(variantion) = K_v \frac{f_{max} - f_{avg}}{f_{max} - f_{min}}, \qquad P(cross) = K_c \frac{f_{max} - f_{avg}}{f_{max} - f_{min}}$$

- Strategy2: Individual-wise adaptive, while the formula seems pretty similar, we do the individual parameter tuning. For item i, we define the probability as:

$$P^{(i)}(variantion) = K_v \frac{f_{max} - f^{(i)}}{f_{max} - f_{min}}, \qquad P^{(i)}(cross) = K_c \frac{f_{max} - f^{(i)}}{f_{max} - f_{avg}}$$

Generally speaking, Simulated Annealing Approach works better than Genetic Algorithm with faster and better convergence of my current param. selection.

7 Reference

[1] Grewal S. (2011) Heuristic Approach. In: Manufacturing Process Design and Costing. Springer, London

[2] 陈小刚等，(1993) 模拟退火法及其收敛性，中科院光电所，Opto-Electronic Engineering, Vol. 20, No.3., 12-18

[3] Wen-yang Lin et.al. (2003) Adapting crossover and mutation rates in genetic algorithms. Journal of Information Science and Engineering 19, 889-903.

[4] Numerical Optimization, Jorge Nocedal et al., Second Edition, Springer, London