Solving Travelling Salesman Problem Using Stochastic Local Search Algorithms

Yi Liu - 1605486

In this Assignment, I was using MATLAB as programming language to implement the Simulated Annealing and Tabu Search algorithms for solving travelling salesman problem. The 2-OPT algorithm was used as the local search method among them.

1.1 Introduction

Simulated Annealing

The Simulated Annealing algorithm is designed to avoid the drawbacks of the randomized search and local search. The randomized search is good at exploration but bad at exploitation, especially bad for problems where good solutions are a very small portion of all possible solutions. The local search is good at exploitation but bad at exploration, it often gets stuck into local optimum.

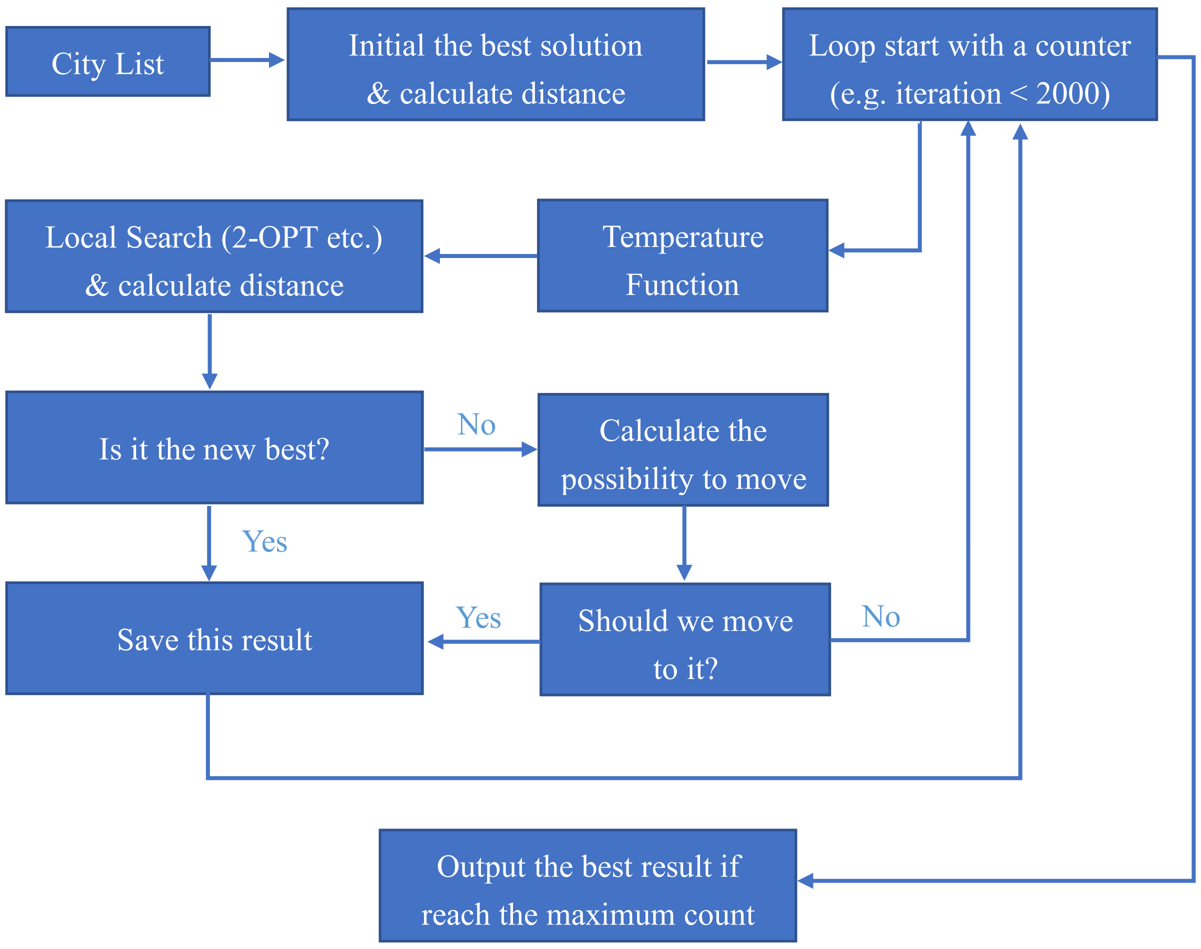
Simulated Annealing algorithm introduced an escape strategy (temperature function & annealing) to escape the local optimum with random non-improving step. The main idea has two parts: firstly, generate a temperature function, the temperature will reduce as the times of iterations increase. Secondly, set a probability to move. If new solution e_{new} is better than current solution e, then the probability is 100% (set it as the new best and continue to climb). If new solution is worse than current solution, then introduce the annealing: set probability as $exp(e-e_{new}/t)$ and determine whether to move according to that probability. This allows algorithm to explore more of the possible search space of solutions.

Tabu Search

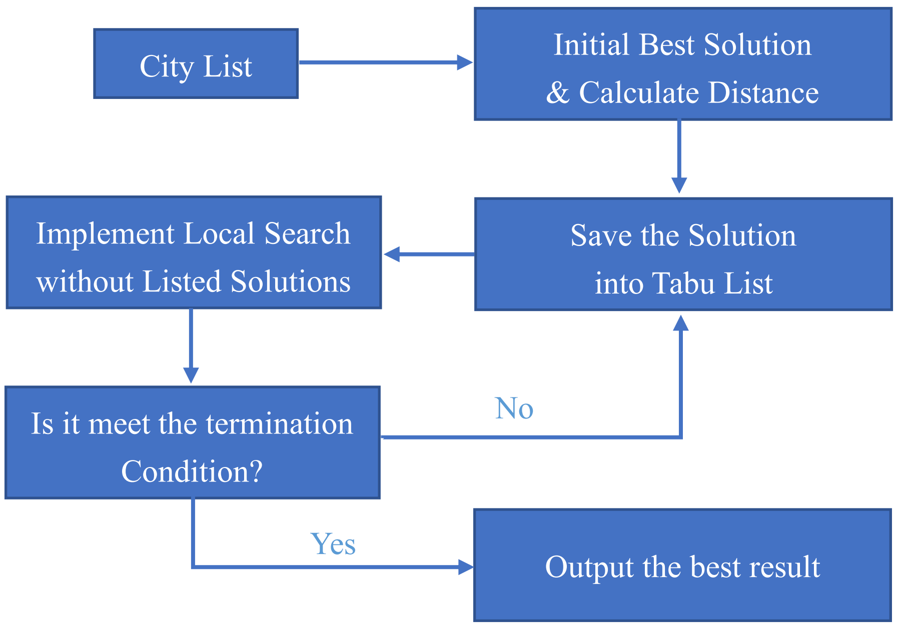
The Tabu Search is an improved version of local search algorithm, it helps algorithm to avoid the local optima. In this assignment, the Tabu Search I implemented is the simplest version with the simplest Tabu list which only remembers recently visited solutions. The main idea of this Tabu Search is use memory to guide local search process away from local optima. It maintains a Tabu list that records previously visited solutions, it forbids the local search algorithm immediately return to previously visited solutions. The main advantage is the quality of solution will improve apparently in the initial stage of search.

1.2 Flow Chart

Simulated Annealing


City List
Initial the best solution & calculate distance
Loop start with a counter (e.g. iteration < 2000)
Local Search (2-OPT etc.) & calculate distance
Temperature Function
Is it the new best?
No
Calculate the possibility to move
Yes
Save this result
Yes
Should we move to it?
No
Output the best result if reach the maximum count

Tabu Search


City List
Initial Best Solution & Calculate Distance
Implement Local Search without Listed Solutions
Save the Solution into Tabu List
Is it meet the termination Condition?
No
Yes
Output the best result

1.3 Pseudo Code

Simulated Annealing

```
// Distance & 2-opt functions are simplified, see the details in the sa.m
begin
    city_list = [cities] // Initial solution.
    distance = $f$(city_list) // Calculate the distance of a list.
    $city\_list_{best}$ = city_list // Initial best solution.
    for (i = 0; i < MAX_ITERATION; i++)
        t = temperature($t_0$) // Temperature calculation.
        $city\_list_{new}$ = opt2(city_list) // Pick some neighbour (2-opt)
        $distance_{new}$ = $f$($city\_list_{new}$) // Calculate the distance
        // Calculate the probability to move to this solution.
        if P(EXP(distance-$distance_{new}$/t)) > R(0,1)
            city_list = $city\_list_{new}$
            distance = $distance_{new}$
        end
        // If solution is the new best, save it.
        if $distance_{new}$ < $distance_{best}$
            $city\_list_{best}$ = $city\_list_{new}$
            $distance_{best}$ = $distance_{new}$
        end
    end
    Output $city\_list_{best}$
end
```

Tabu Search

```
begin
    city_list = [cities] // Initial solution.
    tabu_list = [] // Initial tabu list
    While(terminationflag != true)
        Determine city_list is a non-tabu neighbours // Use ismember()
        $city\_list_{new}$ = opt2(city_list) // Pick some neighbour (2-opt)
        tabu_list = [tabu_list $city\_list_{new}$] // Update tabu list
        if(best solution found || maximum iteration reached)
            terminationflag = true
        end
    end
end
```

2. Parameters

Simulated Annealing

The main parameters of simulated annealing are the initial temperature, terminate temperature and the speed of annealing. These three parameters will influence the number of iterations and possibility of EXP(distance – distance_new / temperature). After I tested the different combinations of those three parameters, the following phenomenon were found:

1. The initial temperature determines the possibility of accepting the inferior solution at the early stage. If I set a very high initial temperature, there's a high possibility to accept the inferior solution at the early stage. If I set a very low initial temperature, the diversity of solutions in search space will be reduced.
2. The terminate temperature determines the stability of the convergence at the end of iteration. If I set a very high terminate temperature, the algorithm continues to accept inferior solution at the end stage.
3. The speed of annealing determines the speed of convergence. If I set a very fast speed of annealing, (e.g. 0.9) the number of iteration will be reduced dramatically, and the algorithm ends before the better solution found.

According to those discovered phenomenon, I set a higher initial temperature as 1500 to accept the higher diversity of solutions, and a terminate temperature as 5 because a good convergence is necessary at the end stage. I set the speed of annealing as 0.995, it makes sure the algorithm won't converge rapidly. The temperature was 9.68 after 1000 iterations, which is good enough to find a satisfied solution.

Tabu Search

The main parameters of Tabu search are the types of arguments in the tabu list and length of the tabu list. I recorded the order of recently visited cities as the type of arguments into tabu list, therefore the recorded route won't appear again before it is eliminated by the constrain of tabu list length. The length of tabu list is a very important parameter in Tabu search. The following phenomenon were found after my test:

1. If the tabu list is too short, the algorithm has a higher probability to get stuck into local optimum (e.g. tabu list < 5).
2. If the tabu list is too long, a large amount of solutions will be banned. It will cost longer computation time. The algorithm may not be able to continue (e.g. tabu list > 200).

According to those discovered phenomenon, I set the length of tabu list as 20. It will record the recent 20 visited routes and forbid the algorithm repeat to visit them in this 20 iterations.

3. Average Results & Standard Deviations

Simulated Annealing

7181	7173	7013	7013	7188	7013
7116	7116	7092	7083	7163	7150
7325	7120	7083	7174	7190	7201
7326	7087	7128	7146	7146	7092
7115	7092	7116	7105	7013	7013

Average (30 runs) = 7126

Standard Deviation (30 runs) = 78.09

Tabu Search

7092	7092	7190	7201	7312	7092
7116	7326	7087	7239	7190	7312
7201	7013	7100	7182	7087	7354
7174	7181	7219	7219	7013	7239
7092	7013	7105	7195	7239	7312

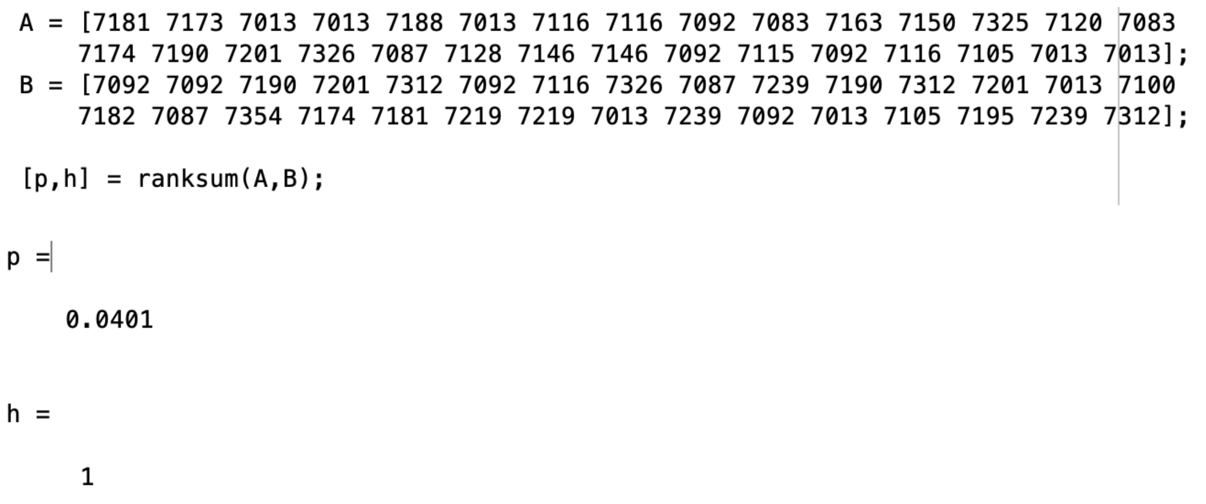
Average (30 runs) = 7172

Standard Deviation (30 runs) = 95.63

4. Comparison

Wilcoxon Signed-Rank Test

If there are two independent samples, we can use Wilcoxon signed rank test as the nonparametric test for two populations. For the two data sets of this assignment, I used ranksum() function in MATLAB to help me do this test. This function outputs two values, *P*-value and hypothesis.



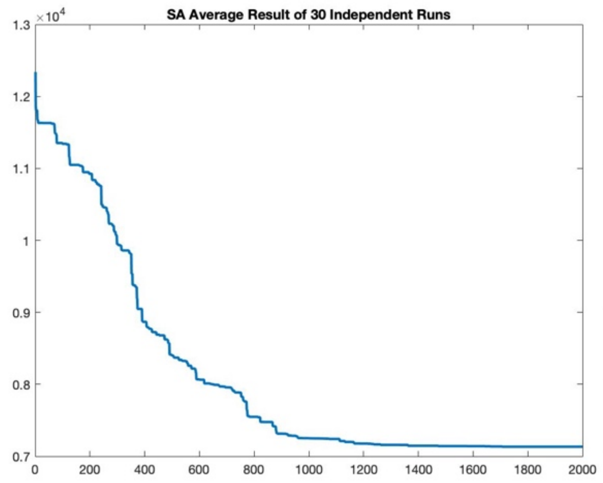
```
 A = [7181 7173 7013 7013 7188 7013 7116 7116 7092 7083 7163 7150 7325 7120 7083
     7174 7190 7201 7326 7087 7128 7146 7146 7092 7115 7092 7116 7105 7013 7013];
 B = [7092 7092 7190 7201 7312 7092 7116 7326 7087 7239 7190 7312 7201 7013 7100
     7182 7087 7354 7174 7181 7219 7219 7013 7239 7092 7013 7105 7195 7239 7312];

 [p,h] = ranksum(A,B);

p =

    0.0401


h =

     1
```

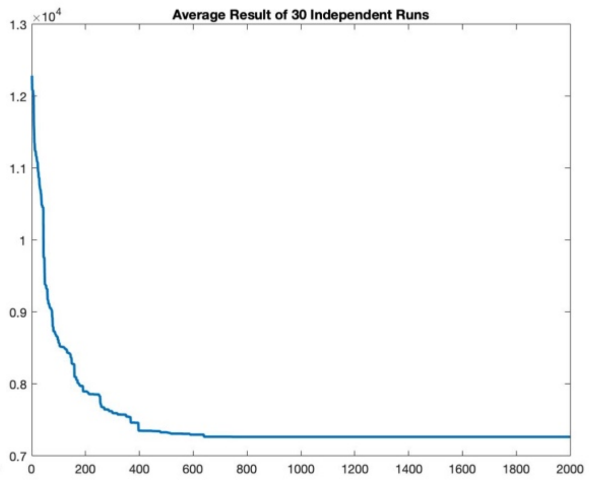
The result were *P*-value = 0.0401 (less than 0.05 significance level) and hypothesis = 1, which means we have to reject the null hypothesis of equal medians.

According to the average and standard deviation of two different algorithms, the Simulated Annealing algorithm has a little bit lower average value, and lower standard deviation than Tabu search. These two phenomena show the Simulated Annealing from my implementation has a higher probability to get a better solution, and more likely to approach the correct solution at the end stage. The appearance of the correct solution (7013) from Simulated Annealing is more than Tabu search. The Simulated Annealing has 5/30 appearances of 7013 and Tabu search has only 2/30 appearances. The reason for the above phenomena happened is partially due to local optimal stalemate of Tabu search, because this is only the simplest version of Tabu search and the length of Tabu list is very limited. However, if the Tabu list is too large, the computation time will be significantly affected.

Average Speed of Convergence

Apart from those two data sets, I also plotted two figures of average speed of convergence in 30 runs.





The first figure is the average speed of convergence of Simulated Annealing, and the second one is the average speed of convergence of Tabu search. According to these two figures, and previous observation, the Tabu Search has faster convergence than Simulated Annealing at early stage.

How to Run The Programme

Please read the README file inside /src folder. Thanks for your reading.