# Annealing Simulation Algorithm Part

December 3, 2019

## 1 Introduction

Simulated annealing (SA) algorithm is another local search strategy we are going to use for this project. the simulated annealing algorithm is not guaranteed to get the optimal solution. However, it is useful to speed up convergence at the beginning of the process with a high temperature, and when it cools down, the probability of optimal solution increases. As it closes to 0, it has the highest probability to get the optimal solutions [1]. In this case, when T is large, it is easy to accept neighboring solutions even it is worse than current solution, but it will be strict when T is getting small. For Simulated annealing algorithm, it is important to know how to update temperature and what should be stop criterion. Sometimes, it is also needed to restart the algorithm.

## 2 General Procedure

*Overview* :
The Annealing simulation algorithm follows a basic procedure:
1. Set the initial temperature and a cooling rate.
2. Create a random initial solution.
3. Begin looping till the stop condition is met. Usually either the system temperature is cool enough or the good solution is found.
4. Inside of the loop, we select the neighbor by making small change to the current solution.
5. Then we decide whether we accept the neighbor solution or not.
6. Finally we decrease the temperature and continue the looping.

# 3 Psudocode

*Heuristics*
The maximum iteration number is the $\frac{initialTemperature}{coolingDegrees}$. A common acceptance method is always to accept the better solution and accept the worse solution with a probability of $P(accept) \leftarrow exp(\frac{e-e'}{T})$, where T is the current temperature, e is the cost of current solutio and $e'$ is the cost of the candidate solution.
When the temperature decreased, the T becomes smaller and the probability of acceptance decrease as well. Therefore the acceptance criteria is getting more stringent.
Restarting the procedures sometimes can help to improve outcomes.

*Pseudocode*
Pseudocode :

**Algorithm 1** Simulated Annealing

```
    Input:ProblemSize, iterations_max, temp_max
    Output:S_best
    function SimulatedAnnealing(ProblemSize)
        S_current ← ScreateInitialSolution(ProblemSize)
        S_best ← S_current
 4:     for (i = 1  to  iterations_max) do
            S_i ← createNeighborSolution(S_current)
            temp_current ← calculateTemperature(i, temp_max)
            if cost(S_i) ≤ cost(S_current) then
 8:             S_current ← S_i
                if cost(S_i) ≤ cost(S_best) then
                    S_best ← S_i
                end if
12:         elseIf  Exp((CostS_current − CostS_i) / temp_current) > Rand()
                S_current ← S_i
            end if
        end for
16: return S_best
    end function
```

# 4 TSP Implementation

# 5 Complexities

# 6 Design

# 7 Empirical evaluation

*Strength and Weakness* :