# Travelling Salesman Problem

By Genetic Algorithm

## Overview

This report shows the pseudocodes and flowcharts of genetic algorithm and simulated annealing to solve the travelling salesman problem.

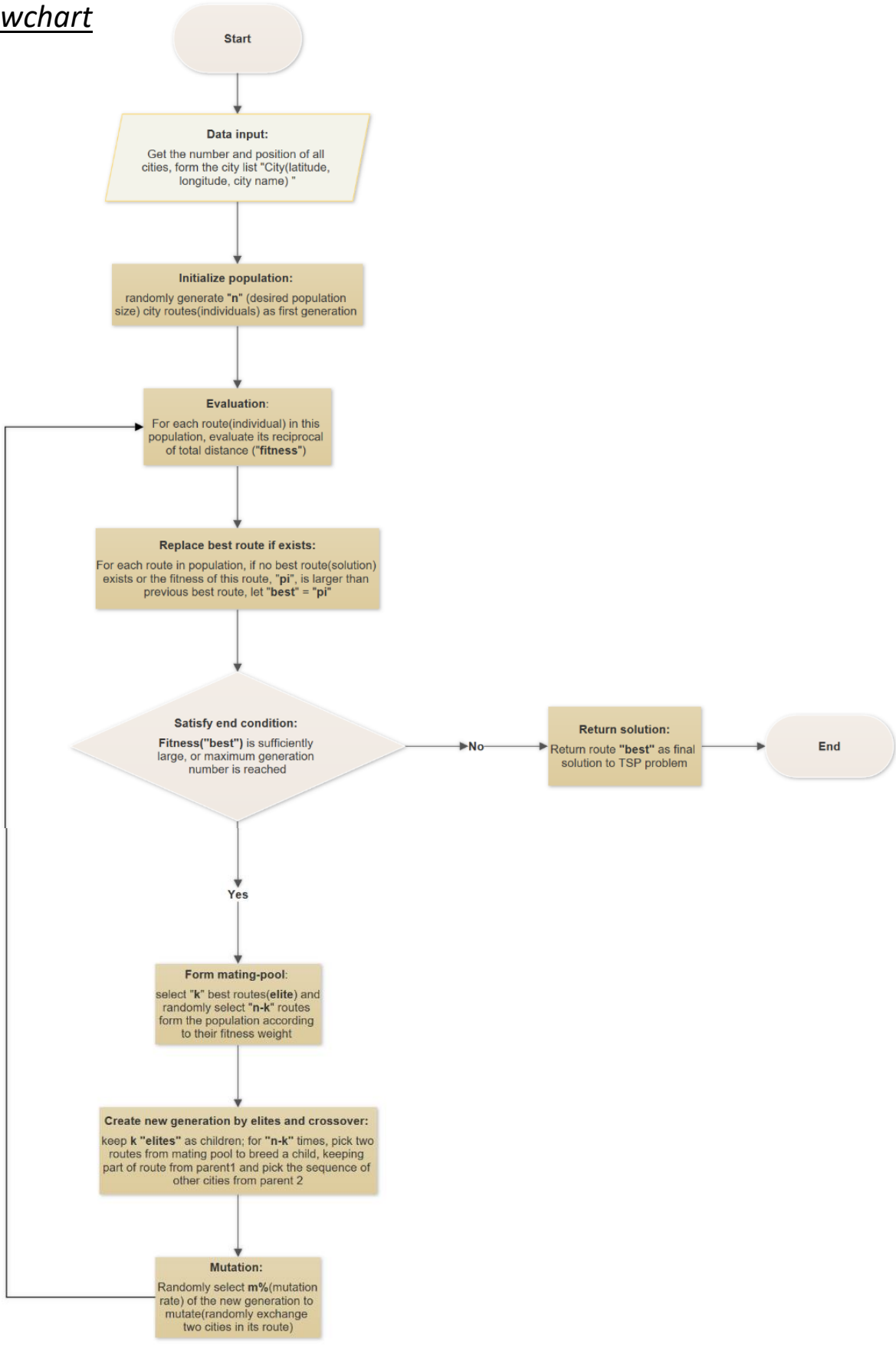
## Genetic Algorithm

### *Part 1: code*

```
best_route=0
n = desired population size

#generate n solutions as the first generation
for i in range(n):
    randomly generate a route (sequence of cities)
endfor


while fitness(best_route) is sufficiently large or maximum generation number is met:
# this is the end condition

  for i in range(generation):
      fitness[i] = 1/(total distance of route i)
      # use the reciprocal of total distance as fitness function
      # evaluate the fitness of every route in this generation

      if fitness[i]>best_route:
          best_route = route i
          #"best" represents the best existing route
      endif

  endfor

  elite = first k routes of this generation(sort fitness from largest to smallest)
  # elite is a group of good individuals in this generation
  # k is the target size of the elite group

  rest = randomly selected (n-k) routes according to their fitness weight
```

```
    # rest is a group of parents that are not elite
    # since the number of total parents should be n but only k elite individuals are selected, "rest" parents should be selected

    mating_pool = rest + elite
    # mating_pool is a group of parents for next generation, which are either from "elite" group or from "rest" group

    child1 = elite
    # child1: "elite" group
    # child2: children generated by crossover of two parents

    for i in range(n-k):
        parent1, parent2 = randomly pick 2 parents from mating pool
        child2[i] = crossover(parent1,parent2)
        #for (n-k) times, randomly choose two parents, and breed a child by crossover
        #the logic of crossover: keep part of route from parent 1 and pick the sequence of other cities from parent 2
    endfor

    child = child1 + child2
    # the next generation are either from "elite" group or from children generated by crossover of two parents

    generation = mutate(child)
    #mutate is to randomly select m%(a desired mutation rate) of the child, which are the routes, and randomly switch two cities of the selected routes
endwhile

solution = best_route
#best is the best route (solution)
```

## *Part 2: flowchart*



# Simulated Annealing

## *Part 1: code*

```
#At first, randomly generate a solution
initial_solution = randomly generate a route (sequence of cities)
current_distance = the total distance of this route
current_route = initial_solution
best_distance =0

#ending condition: the temperature is sufficiently low
while temperature > temperature_end:

    new_route = switch(current_route)
    # generate a new solution by switching two random cities on previous solution
    new_distance= distance(new_route)
    # the total distance of this new route

    probability = exp(-(new_distance-current_distance)/temperature)
    #this is the probability defined by simulated annealing algorithm
    a = a number randomly generated from 0 to 1

    if new_distance < current_distance or a<probabiltiy:
    # if the new route is shorter or the random number is less than the replacement
  probability, replace the old solution by current solution
        current_distance = new_distance
        current_route=new_route
    endif

    if current_distance<best_distance:
    # if the new route is shorter than the best existing route, replace the best route by this
  route
        best_distance = current_distance
        best_route = current_route
    endif

    decrease temperature
endwhile

solution = best_route
# best_route is the best solution to this problem
```

## *Part 2: flowchart*