Practical work – Traveling salesman problem

Viet-Huy HA

April 21, 2023

1 Problem Introduction

The Traveling Salesman Problem is one of the classic and difficult problems in computer science. There have been many approaches to solving this problem since its inception, such as using linear, branch and bounded programming (posted on Informatics and Schools), but only limited to data sets. small material. Recently, evolutionary approaches, such as genetic algorithms, have been applied with better results. In this article, we would like to introduce a unique method based on simulating the behavior of real ants with the process of returning food to the nest in the wild to solve the problem of finding the shortest path for tourists. . This method is relatively difficult compared to the computer level of high school students, so in the article we emphasize on the idea, and installation instructions, as well as the simplest presentation. The authors hope that through the article, students who love Informatics in general and high school students who specialize in Information in particular will have a different perspective from the traditional ways of solving this problem.

Recalling the Traveler problem The Tourist problem, find the shortest path for a merchant (salesman), also known as a salesman, starting from a city, passing through all the cities in turn. at least once and return to the original city at the lowest cost, stated in the 17th century by two British mathematicians Sir William Rowan Hamilton and Thomas Penyngton Kirkman, and recorded in the textbook Theory Oxford's famous graph. It quickly became a difficult problem to challenge all over the world because the algorithmic complexity increased exponentially (in algorithms they are also called NP-hard problems). People began to try and publish the results of solving this problem on computers from 1954 (49 vertices), until 2004 the problem was solved with the number of vertices up to 24,978, and it is expected to continue to increase. again. The problem can be expressed in graph language as follows: Given a graph of n complete vertices with weight G=(V-set of vertices, E-set of edges) with or without direction. Find the Halmilton cycle with the smallest total weight.

2 Algorithm Introdcution

We propose in this project to design and implement two optimization methods allowing to calculate in an exact or approximate way a solution to the problem of the traveling salesman for any graph modeling the links between cities: One based on Genetic Algorithms and the other based on the principle of an Ant Colony

2.1 Genetic Algorithm

Genetic Algorithm (GA) is a technique that mimics the evolutionary adaptation of biological populations based on Darwin's theory. GA is a random optimal search method by simulating the evolution of humans or organisms. The idea of genetic algorithms is to simulate natural phenomena, to inherit and to struggle for survival.

GA belongs to the class of excellent algorithms but is very different from stochastic algorithms because they combine direct and random search elements. The important difference between GA search and other search methods is that GA maintains and processes a set of solutions, called a population. In GA, the search for a suitable hypothesis begins with a population, or initial selection of hypotheses. Individuals of the current population initiate the next generation by random breeding and mutation activities – sampled after biological evolutionary processes. At each step, the hypotheses in the current population are estimated relative to the fitness, with the most suitable hypotheses selected probabilistically as the seeds for the production of the next generation, called individual (individual). The individual that is more developed and adapted to the environment will survive and vice versa will be eliminated. GA can detect the new generation with better adaptability. GA solves mathematical programming problems through basic processes: crossover, mutation and selection for individuals in the population. Using GA requires determining: initial population initialization, function evaluating solutions according to the degree of fitness - objective function, genetic operators generating the reproductive function.

2.1.1 Basic properties of the algorithm

1. **Population**: Population is the subset of all possible or probable solutions, which can solve the given problem.
2. **Chromosomes**: A chromosome is one of the solutions in the population for the given problem, and the collection of gene generate a chromosome.
3. **Gene**: A chromosome is divided into a different gene, or it is an element of the chromosome.
4. **Allele**: Allele is the value provided to the gene within a particular chromosome.
5. **Fitness Function**: The fitness function is used to determine the individual's fitness level in the population. It means the ability of an individual to compete with other individuals. In every iteration, individuals are evaluated based on their fitness function.
6. **Genetic Operators**: In a genetic algorithm, the best individual mate to regenerate offspring better than parents. Here genetic operators play a role in changing the genetic composition of the next generation.
7. **Selection**: After calculating the fitness of every existent in the population, a selection process is used to determine which of the individualities in the population will get to reproduce and produce the seed that will form the coming generation. We have some types of selection styles available:
   - Roulette wheel selection
   - Elitism Selection
   - Tournament Selection

2.1.2 How it works

1. **Initialization**
   - An initial population will have certain individuals with different characteristics, these characteristics will determine the ability to reproduce, survive, and respond to environmental conditions of each individual.
   - The process of a genetic algorithm starts by generating the set of individuals, which is called population. Here each individual is the solution for the given problem. An individual contains or is characterized by a set of parameters called Genes. Genes are combined into a string and generate chromosomes, which is the solution to the problem. One of the most popular techniques for initialization is the use of random binary strings.
2. **Evaluation Phase**: A fitness function is a particular type of objective function which takes as input a candidate solution and outputs the quality of this solution, therefore the fitness function makes it possible to evaluate the candidate solutions
3. **Selection**: Selection is the process of selecting parents to generate the Child we call it also offspring that will be a part of the next generation.
4. **Crossover**: The crossing operation is the Process of reproduction of new chromosomes from the parent chromosomes (parents are selected from the old population Using A Selection Method)
5. **Mutation**: The mutation operator inserts random genes in the offspring (new child) to maintain the diversity in the population. It can be done by flipping some bits in the chromosomes

2.2 Ant Colony optimization algorithms

In the natural world, the first ants to find their way randomly choose a path, and find food back to the nest while finding their way, leaving their pheromones along the way to mark. If other ants look for the same path, they don't have to move randomly, but instead follow a pre-existing path, going back and reinforcing if they find their way. eat.

However, over time the pheromone stored along the way will begin to evaporate, thus reducing its concentration. The longer the pheromone they leave on the previous road, the longer it will take for the pheromone to completely evaporate. By comparison, shorter paths will store higher pheromone concentrations than longer paths due to more ants traversing. If the pheromone is completely evaporated, the paths chosen by the first ants will no longer be attractive to later ants. In this case, the effect of pheromone evaporation might not really affect the ant colony's pathfinder, but it has important implications for artificial systems.

As a result, when the ants find the best route to the food source from their nest, the other ants will follow that path and give positive feedback on the path leading to the ants. will also follow a single path. The idea of the ant colony algorithm is based on behavior with ants that we assume they will walk around the graph to see the problem to be solved.

We can describe how it works using the following stages:

1. Stage 1: All ants are in their nest. There is no pheromone content in the environment. (For algorithmic design, residual pheromone amount can be considered without interfering with the probability)

2. Stage 2: Ants begin their search with equal (0.5 each) probability along each path. Clearly, the curved path is the longer and hence the time taken by ants to reach food source is greater than the other.

3. Stage 3: The ants through the shorter path reaches food source earlier. Now, evidently they face with a similar selection dilemma, but this time due to pheromone trail along the shorter path already available, probability of selection is higher.

4. Stage 4: More ants return via the shorter path and subsequently the pheromone concentrations also increase. Moreover, due to evaporation, the pheromone concentration in the longer path reduces, decreasing the probability of selection of this path in further stages. Therefore, the whole colony gradually uses the shorter path in higher probabilities. So, path optimization is attained.

3 Results and Comparison

3.1 Genetic Algorithm Results

When applying the GA algorithm, we use several different versions at the Selection and Mutation steps to try to optimize the algorithm. When applying the GA algorithm, we use several different versions at the Selection and Mutation steps to try to optimize the algorithm. The results obtained are as follows: (**We are considering the test with list of given cities coordinate: (63, 8), (33, 34), (27, 21), (78, 19), (30, 77), (9, 34), (23, 8), (55, 7), (73, 56), (10, 71)**).

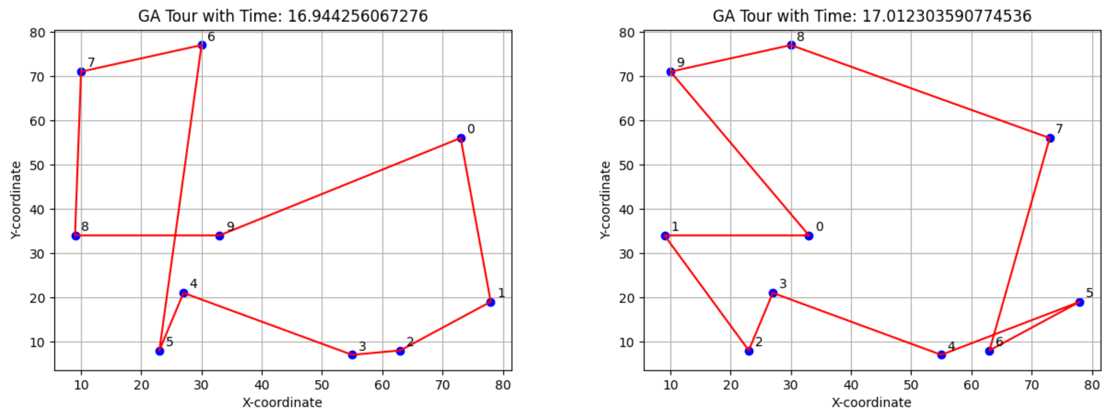
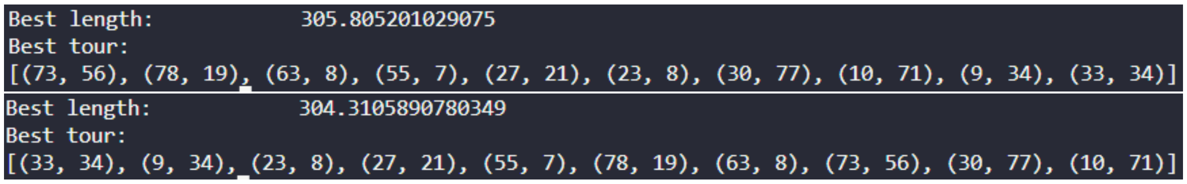


Figure 1: Genetic Tour and Time when using Roulette Wheel Selection and Rank Selection

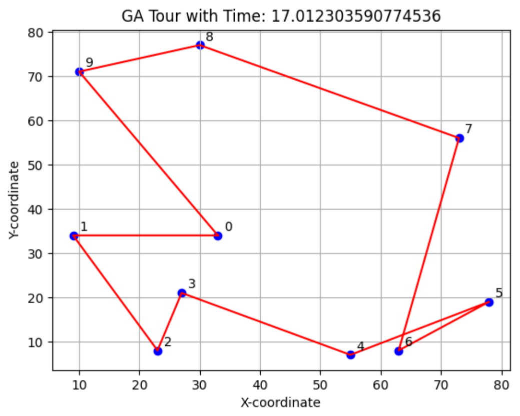


```
Best length:              305.805201029075
Best tour:
[(73, 56), (78, 19), (63, 8), (55, 7), (27, 21), (23, 8), (30, 77), (10, 71), (9, 34), (33, 34)]
Best length:              304.3105890780349
Best tour:
[(33, 34), (9, 34), (23, 8), (27, 21), (55, 7), (78, 19), (63, 8), (73, 56), (30, 77), (10, 71)]
```

Figure 2: Genetic Tour and Length when using Roulette Wheel Selection and Rank Selection

Through the obtained results, we can see that the selection method using Roulette Wheel Selection is slightly better in time, but slightly less accurate in terms of accuracy than Rank Selection. However, in terms of efficiency, we can see that they are equivalent

Next, we will consider results of two mutation types: Swap Mutation and Scramble Mutation.



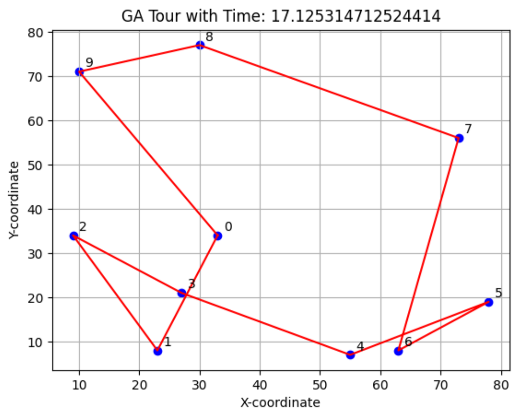
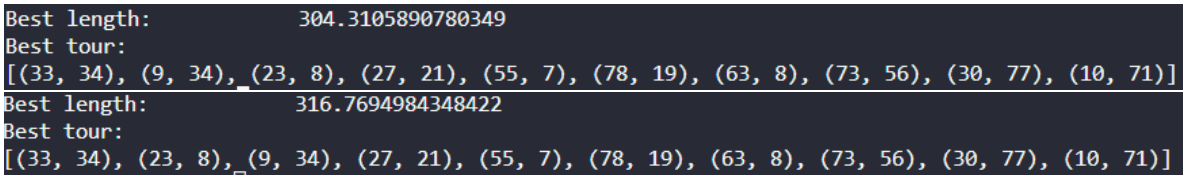


Figure 3: Genetic Tour and Time when using Swap Mutation and Scramble Mutation



```
Best length:              304.3105890780349
Best tour:
[(33, 34), (9, 34), (23, 8), (27, 21), (55, 7), (78, 19), (63, 8), (73, 56), (30, 77), (10, 71)]
Best length:              316.7694984348422
Best tour:
[(33, 34), (23, 8), (9, 34), (27, 21), (55, 7), (78, 19), (63, 8), (73, 56), (30, 77), (10, 71)]
```

Figure 4: Genetic Tour and Length when using Swap Mutation and Scramble Mutation

As we can see, in this test case, Swap Mutation has prevailed over Scramble

Analysis:

The reason Swap Mutation can yield better results than Scramble Mutation for a particular problem is that Swap Mutation are less disruptive and make smaller changes to the solution . It allows the GA to perform a more focused search in the local neighborhood of the solution, resulting in better exploitation of the search space. This can be especially useful for problems where the quality of the solution depends heavily on the order of the elements, as in the case of TSP.

3.2 Ant Colony Result

We also consider the Ant Colony algorithm with the above test case. **We are considering the test with list of given cities coordinate: (63, 8), (33, 34), (27, 21), (78, 19), (30, 77), (9, 34), (23, 8), (55, 7), (73, 56), (10, 71)**

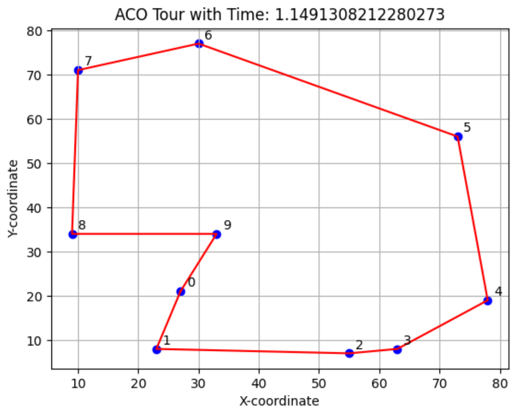
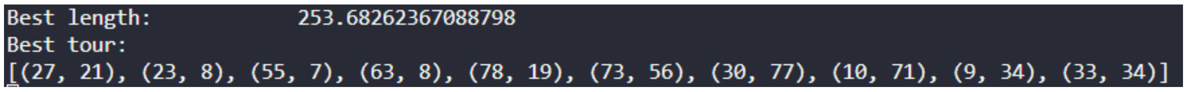


Figure 5: ACO algorithm



```
Best length:              253.68262367088798
Best tour:
[(27, 21), (23, 8), (55, 7), (63, 8), (78, 19), (73, 56), (30, 77), (10, 71), (9, 34), (33, 34)]
```

Figure 6: ACO best tour

Looking at the results obtained with this test case, we can see that the ACO algorithm used in this project is performing much better than GA. This can be traced back to the effective use of heuristic information and Pheromone updates

3.3 Genetic And Ant Colony Comparison

In this section, we will compare through many test cases with different values of n. In this section, we will compare through many test cases with different values of n. The results are shown through the two comparison charts below

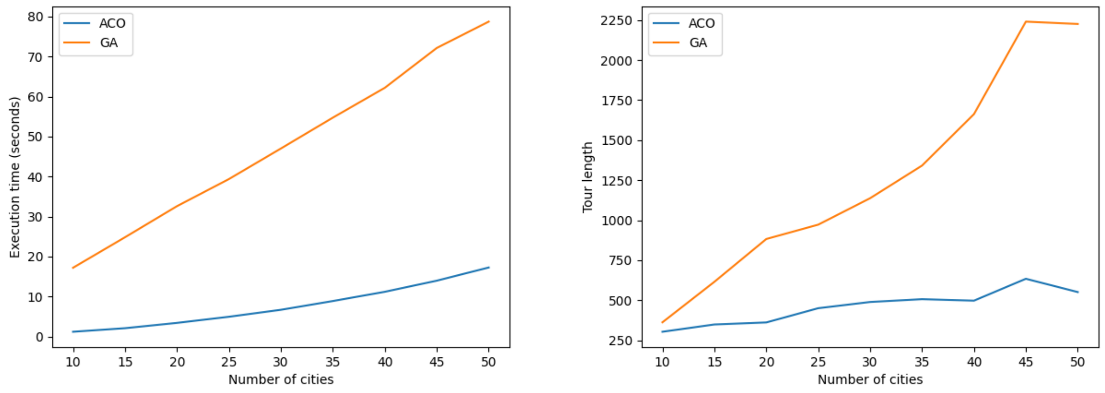


Figure 7: Execution Time and Best Length Found

Through two comparison charts, it is clear that the execution time of the ACO algorithm in this project is much better than that of GA. Obviously, the larger the test set, the larger the difference. Besides, the efficiency of ACO algorithm is also superior. The distance that the ACO algorithm finds is much better than that of GA.

3.3.1 Analysis:

The superior results of ACO compared to GA in the project can be traced back to the following reasons:

1. **Parameter settings:** The ACO algorithm has specific parameter settings (alpha, beta, rho, Q) that can be better tuned for this problem case than the GA parameters (mutation rate) , cross-sectional ratio, population size, etc.). Better parameter settings can lead to more efficient exploration and exploitation of solution spaces, leading to better solutions.

2. **Heuristic information**: The ACO algorithm uses heuristic information (in this case, the inverse of the distance between cities) to guide the search process. This can help the algorithm focus on more promising areas of the solution space, leading to better solutions.

3. **Pheromone updates:** The ACO algorithms use a global pheromone update mechanism that allows ants to share information about the good solutions they have found. This can help the algorithm to converge faster to a high-quality solution, while GA relies only on crossover and mutation operators to explore the solution space.