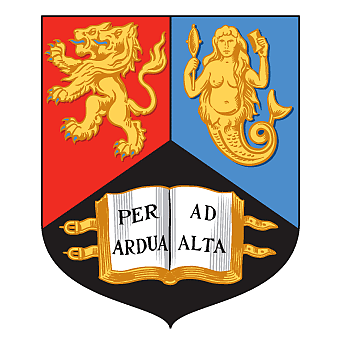
ELECTRONIC, ELECTRICAL AND COMPUTER ENGINEERING



University-of-Birmingham-logo

# Object Oriented Programming Using C#

**Assignment 2014-15**

***AN04208_***

***Swarm Intelligence***

**Name: Yechen Tian**

**Student Number: 1337236**

1. Introduction

This project is to resolve the Travelling Salesman Problem. C# is adopted to program, Unified Modeling Language (UML) is used to model this problem, and the design idea comes from Swarm Intelligence, specifically the ant colony optimization (ACO) algorithm. Besides the ACO algorithm, the Greedy algorithm can be used to design the path, but it is not the best choice.

The ACO algorithm is an algorithm used to find the optimal path in the graph. It was proposed by Marco Dorigo in the year 1992 in his PhD thesis. [1] His inspiration came from a fact that ant seek the path to get their food. The ant colony algorithm is a novel simulated evolutionary algorithm which shows many good properties. Each ant starts to seek food without any information about the position of the food. When one finds the food, it will release volatile secretions which can be called pheromone to the circumstance. The pheromone will gradually volatilize with time lapse, that is to say, the density of the pheromone represents the distance of the path. So it can attract other ant to find the food, and an increasing number of ant will find the food. But some of the ant will not follow the path which has been found, they will seek food through other paths. If the new path is shorter than the path before, they will release more pheromone to attract other ant, and the ant will get food though the shorter path. After a period of time, the ant can probably find the shortest path to get food.

The Greedy algorithm is described as when a big problem need to be solve, no matter what to do, the best solution will adopted based on the present situation. That is to say, the solution is not the best for the whole problem, to some extent, it is only the best choice for part problem. So it cannot provide the optimal solution to all problems, but it can provide approximately optimal solution to some big problems. The Greedy algorithm is a stage treatment solution. [2]

TSP is one of the famous problems in the mathematic area. It is assumed one travelling salesman needs to visit number N cities, he has to choose the path to visit all of the cities. The requirement is anyone of the cities can be visited only once, and through the whole path, the travelling salesman must go back to the initial city where he starts the travel, and the distance of the path should be the shortest among all of the possible paths.

The ACO is used to resolve the TSP. Set value to measure the distance between city i and city j and. Set a value to represent the amount of ant in city i when the time is t, so the whole amount of ant, represents the residual amount of information on the line between city i and city j, assume when the travel start, all the information left in all lines. Parameter represents the information retention, after time n, the information on the line updates to (), in this formula, represents the information which is left by the number k ant in the ij line and represents the information left by all ant which go through the ij line. When k ant go through the line, and when no ant go through the line, . Define The number k ant in its moving process, and represents the probability for the ant k to move from position i to position j, and , a set can be used to record the cities that ant k has passed, and the set means the cities where ant k will probably go. The value represents the sum of the distance which the ant k has passed. Based on the application of the ACO algorithm to the TSP. UML should be created firstly to model this problem. Then according to the UML, the program will be finished to make the functions come true.

UML started from an Object Management Group (OMG) standard, it is a graphic language which supports modelling and software system development. And it can provide modelling and visualization supports to the development of system, including from requirement analysis to specifications and from structure to configuration. Object Oriented Analysis and Design was very popular between 1980s and 1990s, and it gave birth to the UML. It not only unified the expression of Booch, Rumbaugh and Jacobon, but also made a further development, and it became the standard modelling language which is accepted by most people. [3]

In this project, based on the UML, the use-case model, analysis model and design model should be set to finish the program.

1. Use-case Model

2.1. Brain storm

Potential requirements

Table 2.1.1. Prioritization of Potential requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Concept | Necessity | Risk | Cost | Priority |
| Users can open an operation interface to operate. An operation interface should be set. | High | Med | Med | High |
| Data can be typed in or chosen in the operation interface. | Medium | Med | Med | Med |
| City points can be shown in the graph when the data are chosen. | Low | Low | Med | Low |
| The choice of algorithm, according to different algorithm, different path will be shown. | Medium | Low | Low | Med |
| The path will be formed in the graph. | High | Low | Med | High |

Because there is no high risk in all conceptions, all the functions and requirements can be fulfilled.

The use-case diagram is shown in figure 2.1.2. .

Figure 2.1.2. Use-Case Diagram



2.2. Scenario Descriptions

The user starts the operation interface. The user can load the data which he or she wants to use, the data have been saved into the files, and the user just needs to pick one of data. Then the system generates the graph of all points in it. All the positions of the points represent the cities position, user can see the cities position directly. The user can choose the algorithm to calculate the path, different algorithm forms different paths. Finally the path will be shown in the graph according to the algorithm.

* 1. Class Identification

Nouns:

User, cities data, path, operation interface, traveler.

Stereotypical classes:

Boundary Operation interface

Entity Path

Control algorithm

* 1. CRC Cards

CRC cards describe the responsibilities and collaborators of each class. They are vital in the Analysis stage but should be drafted during User Modelling as an aid to identifying attributes, methods and inter-class relationships. The CRC cards is shown in table 2.4.1. .

Table 2.4.1. Class Responsibilities Collaborators Cards

|  |  |
| --- | --- |
| Class: operation interface |  |
| Responsibilities | Collaborators |
| The operation interface is responsible for operating the system and showing the result of the system, and it can show the status of the system. When the operation interface is opened, user can choose data to calculate the path, and user also can change the data to get another path. How the path is formed is based on the algorithm, via choosing which algorithm to use, user can finally get the path they want. | Algorithm, Path |

|  |  |
| --- | --- |
| Class: algorithm |  |
| Responsibilities | Collaborators |
| Algorithm is responsible for path generation. A traveler start from one city, based on the algorithm, he or she choose the next destination. User can choose different algorithm to reach their goal. Different algorithm can form different path, via selecting algorithm, different path will be shown on the interface. | Operation interface, Path. |

|  |  |
| --- | --- |
| Class: path |  |
| Responsibilities | Collaborators |
| The path is the result that user wants. It is optimal path of all the cities. Traveler starts from one city, and go to each city without repetition, finally he or she gets back to the initial city. How the traveler travels is based on the algorithm, that is to say, algorithm decides which way to go, and after the calculation, the result will be shown on the interface. | Operation interface, algorithm. |

* 1. Interaction Diagram

Interaction Diagram is most likely to be a sequence diagram since the sequencing of the messages is most apparent from the user requirements. This diagram shows the interaction relationship among the three classes. The interaction diagram will be shown in figure 2.5.1. .

Figure 2.5.1. Interaction Diagram



* 1. State Chart Diagram

The Class with most and the more significant state transitions in the system. The states are: Idle, date selected, algorithm selected, points formed in the graph, path formed. Through different operations, the user gets the result. The state chart diagram will be shown in figure 2.6.1. .

Figure 2.6.1. State Chart Diagram



* 1. Class Diagram

A simple class diagram showing linkage between class methods and attributes is expected. This diagram should be a first pass. It should not consider the privacy or visibility of methods and attributes and should not show inheritance. It should show composition or aggregation and other relationships. The class diagram will be shown in figure 2.7.1. .

Figure 2.7.1. Class Diagram



1. Analysis Model

From Operation Interface Class

Potential attributes: Interface open, Interface close, Points on the graph, Path on the graph.

Potential methods: Open the Interface, Choose data, Alter data, Select an algorithm, and Alter the algorithm.

From Algorithm Class

Potential attributes: ACO algorithm, Greedy algorithm

Potential methods: Program an ACO algorithm into the system, Program a Greedy algorithm into the system

From Path Class

Potential attribute: Path formed on the graph, Path saved as a file or a picture.

Potential methods: The path connected all cities is the shortest. To each city, the path from and to the city is the shortest. Save the path as a file or picture.

3.1. Attributes

The attributes are listed in the table 3.1.1. . All attributes belong to different classes.

Table 3.1.1. Attributes

|  |  |  |
| --- | --- | --- |
| Class | Attribute | Comment |
| Operation Interface Class | Interface open | Status |
|  | Interface close | Status |
|  | Points on the graph | Graph/number |
|  | Path on the graph | Graph/number |
| Algorithm Class | ACO algorithm | Program |
|  | Greedy algorithm | Program |
| Path Class | Path calculated | Process |
|  | Path saved as file or picture | Saving |

3.2. Methods

Then the methods will be shown in figure 3.2.1. .

Figure 3.2.1. Methods

|  |  |  |
| --- | --- | --- |
| Class | Method | Comment |
| Operation Interface Class | Open the Interface() | These operations can be done by users to find the optimal path. |
|  | Choose data() |  |
|  | Alter data() |  |
|  | Select an algorithm() |  |
|  | Alter the algorithm() |  |
|  | Close the Interface() |  |
| Algorithm Class | Program an ACO algorithm() | Choose one of the algorithms to design the path |
|  | Program a Greedy algorithm() |  |
| Path Class | Form the shortest path() | According to different algorithms, different paths are formed. |
|  | Traveler chooses the nearest city() |  |
|  | Save the path as a file or picture() | The path can be saved, and the user can use it. |

* 1. Sequence Diagram

Triggers and parameters are added. Some details are added. Then sequence diagram will be shown in figure 3.3.1. .

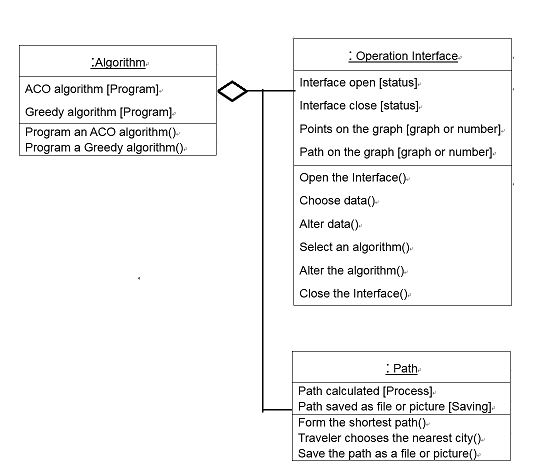
Figure 3.3.1. Sequence Diagram



* 1. Class diagram

The class diagram will be shown in figure 3.4.1. .

Figure 3.4.1. Class diagram



* 1. State Chart Diagram

The State Chart Diagram should be more detailed and added to some specific actions, and it will be shown in figure 3.5.1. .

Figure 3.5.1. State Chart Diagram



* 1. Non-functional Requirements

When the calculation is finished, the graph should stop running. When error occurs, the operation interface should restart automatically. When data or algorithm has some problem, there should be a window reminding user the wrong information.

1. Design Model
   1. Revisit Use-case Model

The design meets the requirements. User can use operation interface to get the optimal path. User can choose the data about the cities, and the cities represented as points are formed in the graph, then choose the algorithm, the path will be generated. Users can saved the path as picture or file.

* 1. Sequence Diagram

The sequence diagram in design model is the same as painted in analysis model. It is shown in figure 3.3.1. .

* 1. Textual Description of Object to Object Interaction

The Operation Interface Class provides the operation platform to users to reach their goal, that is to say, users use the interface to get the optimal path.

The Algorithm Class helps to generate different paths which users want, algorithm calculate the path and finally forms a path in the graph shown on the interface.

The Path Class is the result of all operations, it can be shown on the interface and it can be saved to users.

* 1. Implementation of Non-functional Requirements

Non-functional requirements are addressed with the whole system, and it can be embodied with the Operation Interface Class.

* 1. Deployment Model

The whole functions are realized in C#.

* 1. Reconsider the Attributes

The attributes are the same as mentioned in analysis case. They are shown in table 3.1.1. .

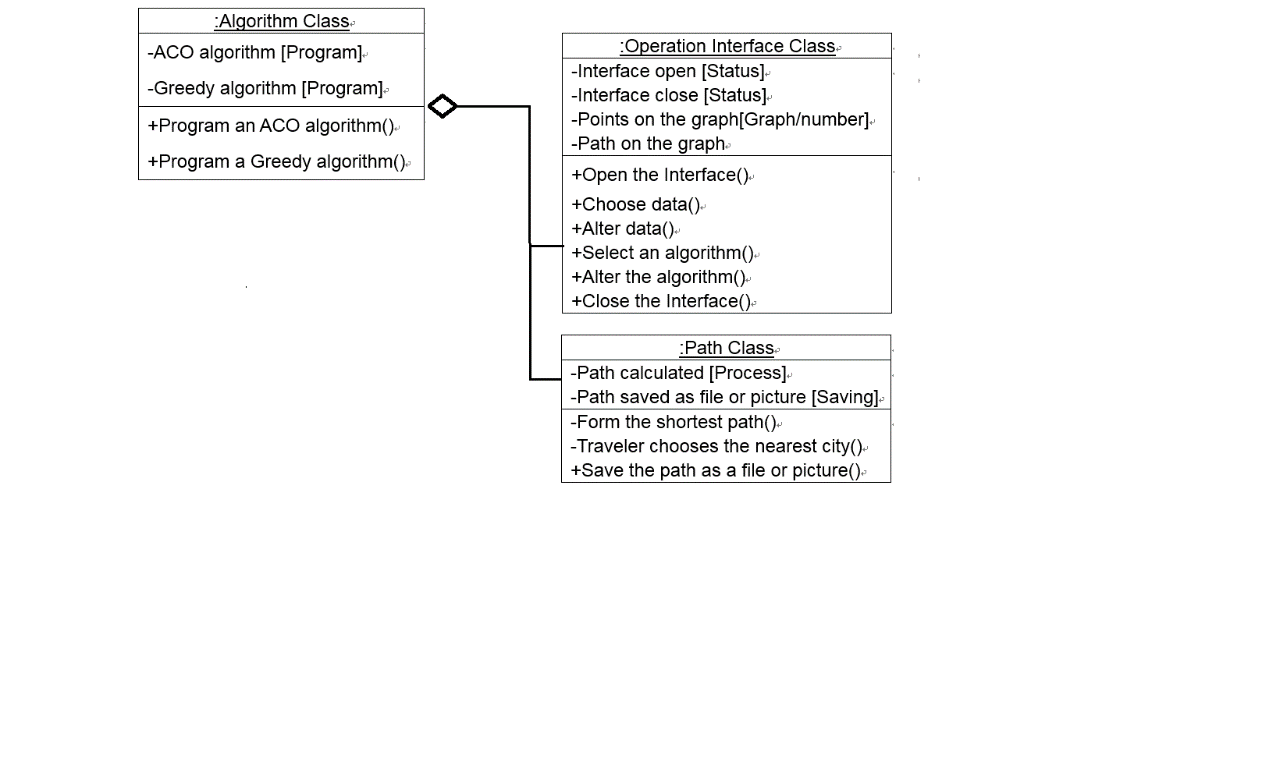
* 1. State Chart

Because the attributes do not change, the state chart remains as it shown in 3.5.1. .

* 1. Class Diagram Showing Visibility

The diagram will be shown in figure 4.8.1. .

Figure 4.8.1. Class Diagram Showing Visibility



1. Test

Three tests are done in the operation interface. Three groups of data are used to generate the path.

Figure 5.1 shows 29 cities shown as points. Figure 5.2 shows the path generated via Greedy Algorithm, and figure 5.3 shows the path generated via ACO Algorithm.

Figure 5.1 29 cities

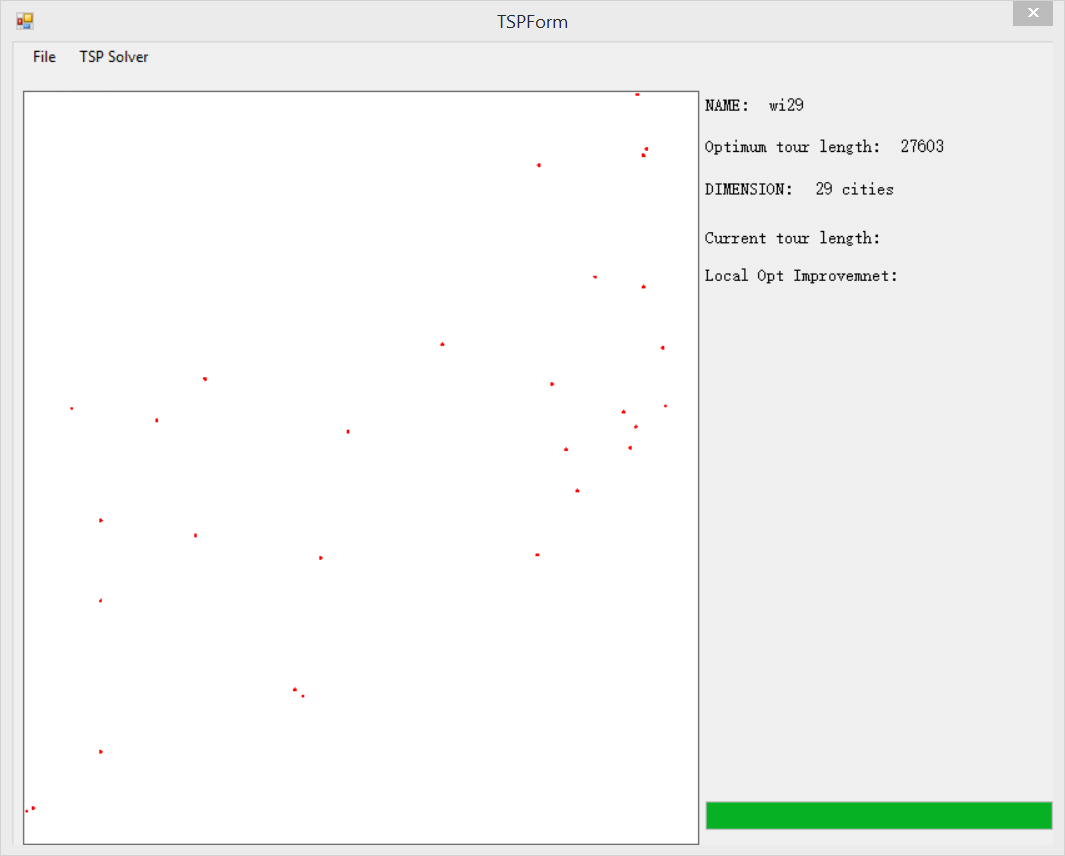


Figure 5.2 29 cities via Greedy Algorithm

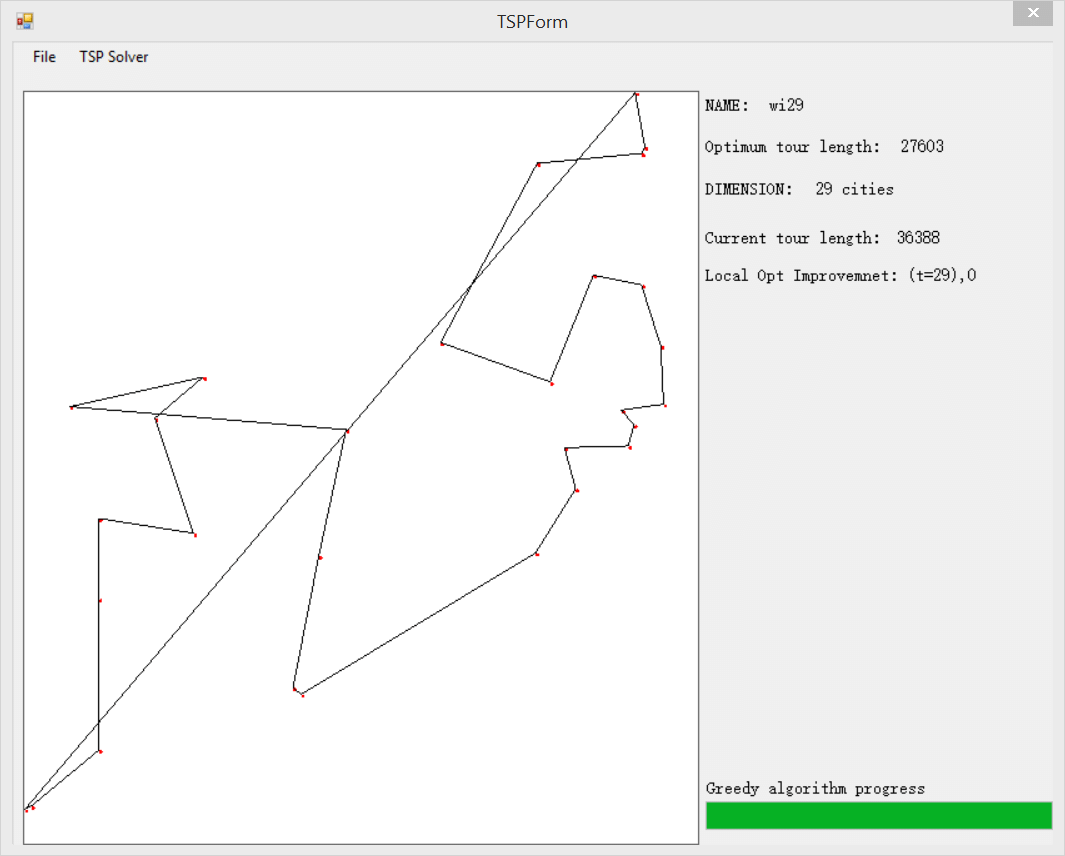


Figure 5.3 29 cities via ACO Algorithm

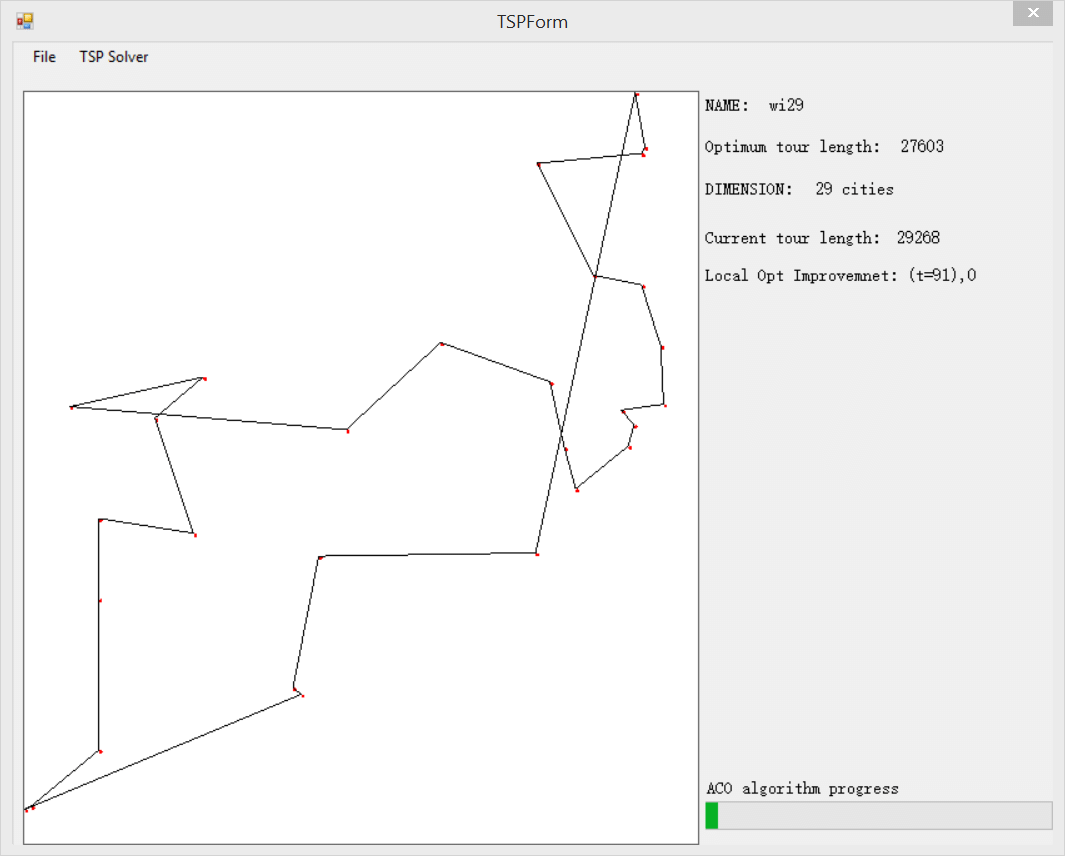


Figure 5.4 shows 194 cities shown as points. Figure 5.5 shows the path generated via Greedy Algorithm, and figure 5.6 shows the path generated via ACO Algorithm.

Figure 5.4 194 cities

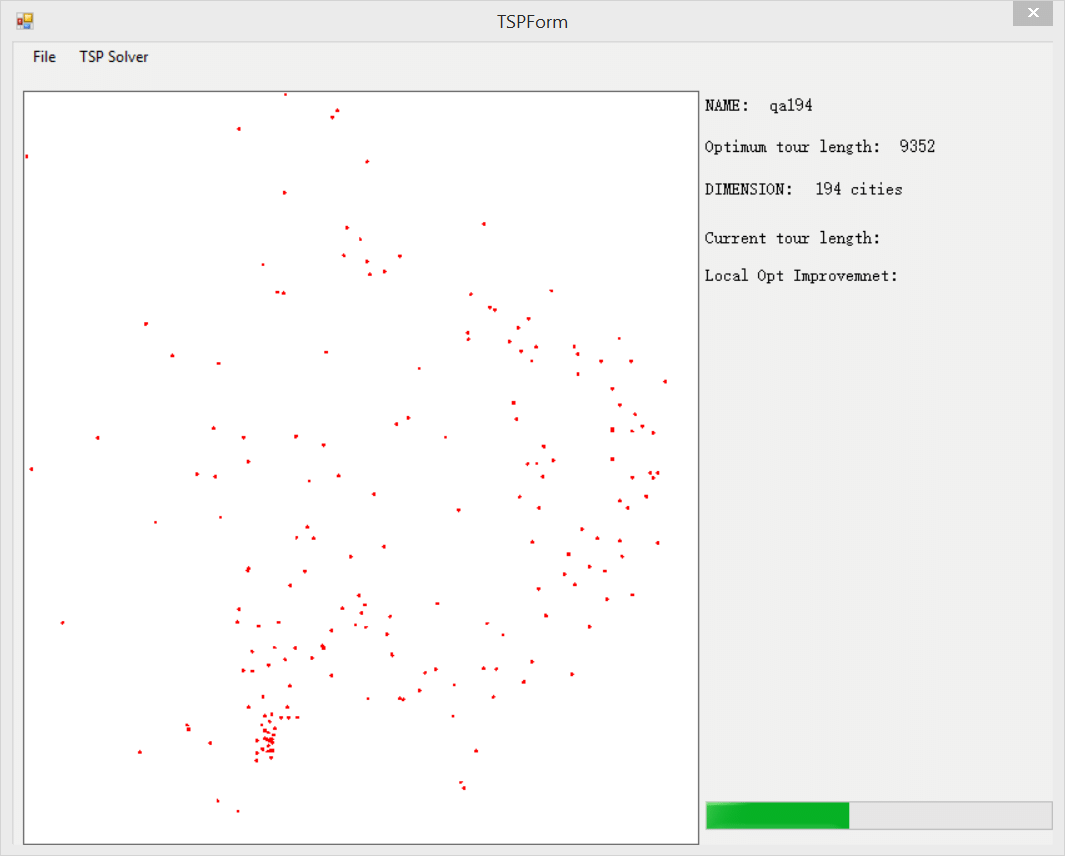


Figure 5.5 194 cities via Greedy Algorithm

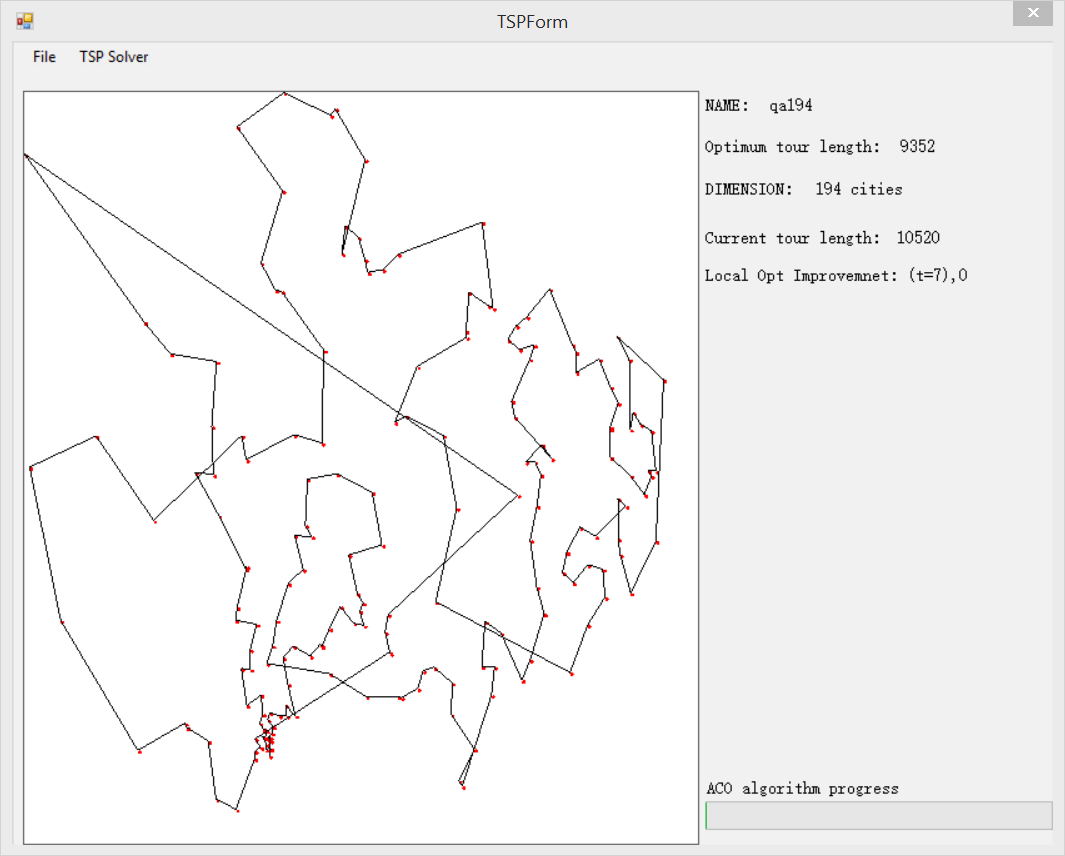


Figure 5.6 194 cities via ACO Algorithm

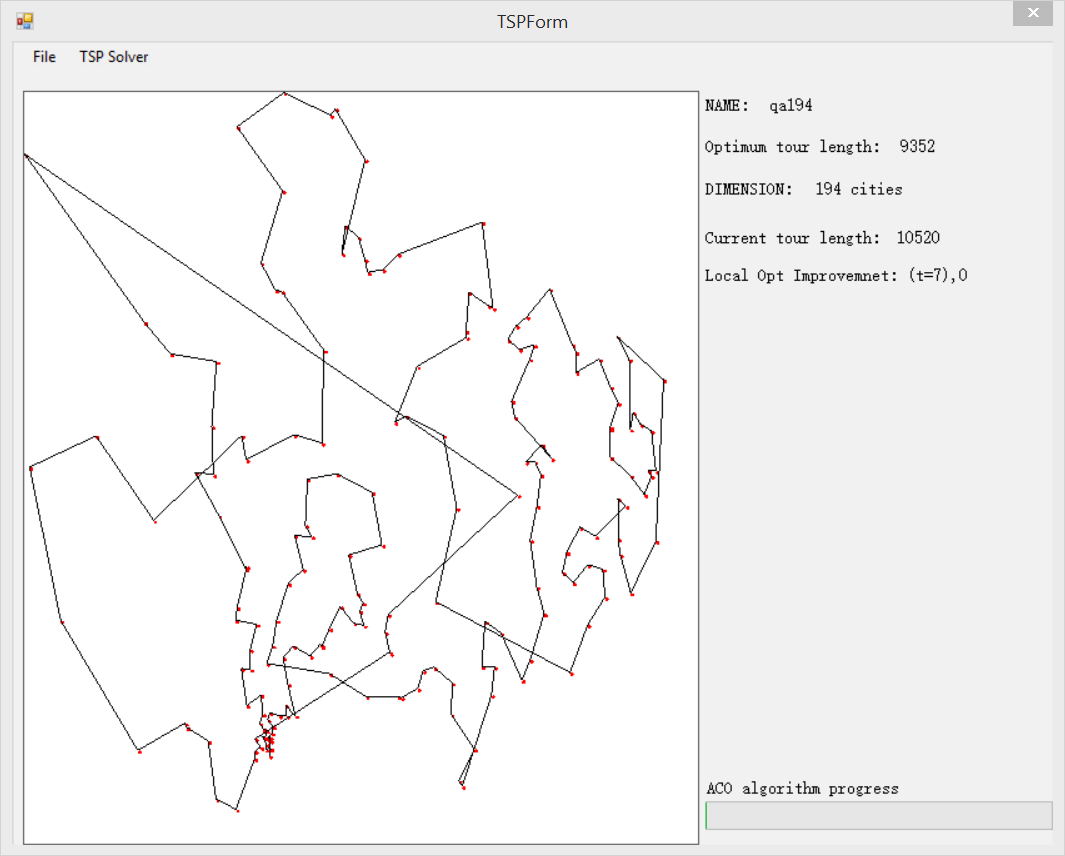


Figure 5.7 shows 3496 cities shown as points. Figure 5.8 shows the path generated via Greedy Algorithm, and figure 5.9 shows the path generated via ACO Algorithm.

Figure 5.7 1496 cities

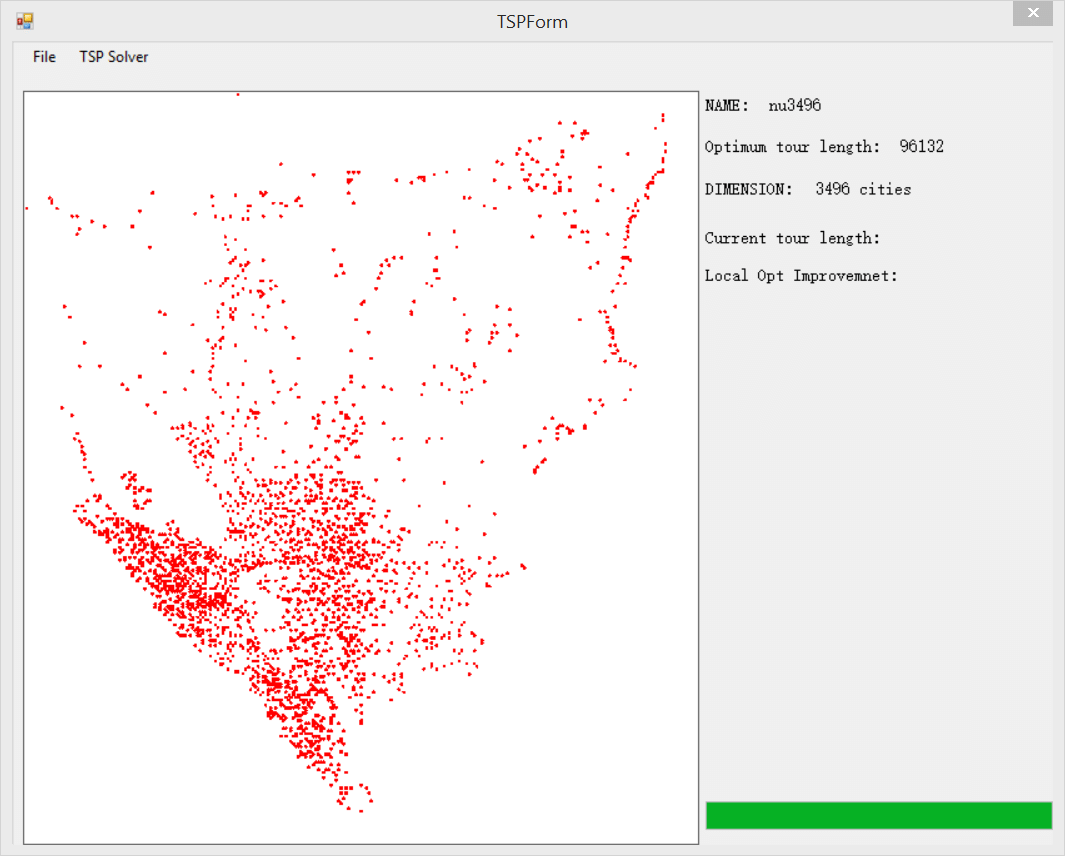


Figure 5.8 3496 cities via Greedy Algorithm

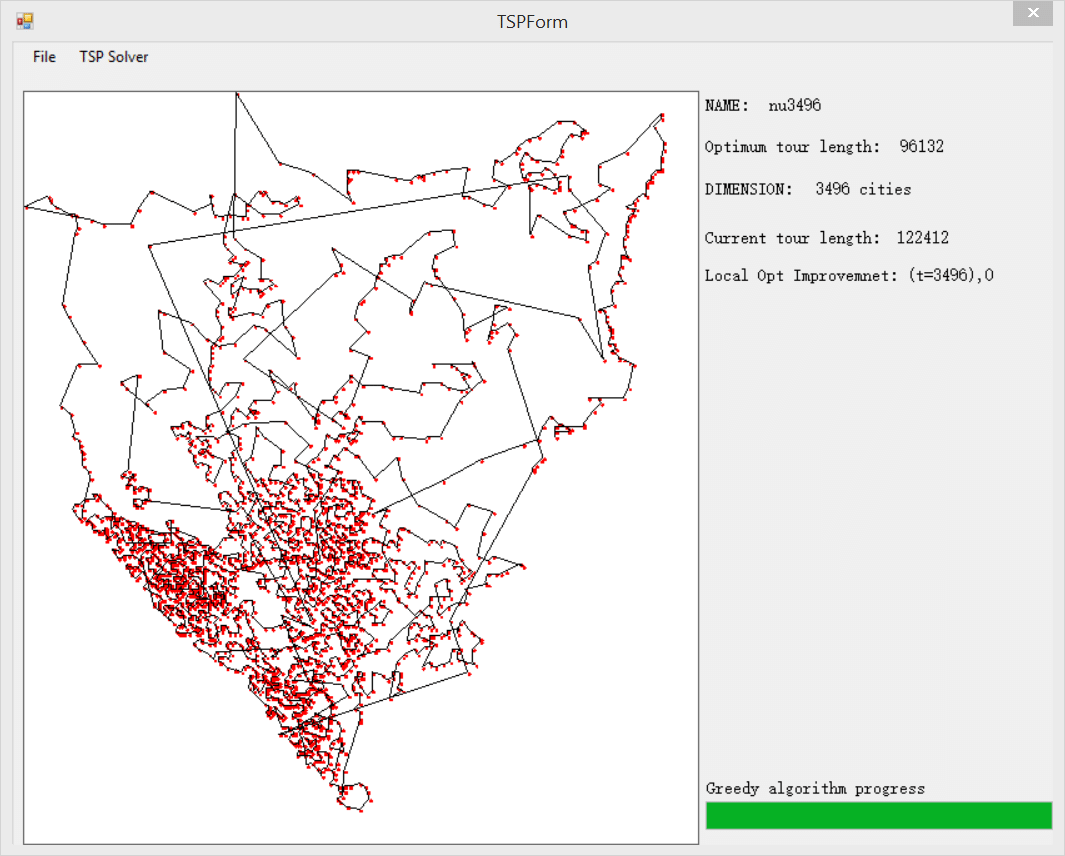
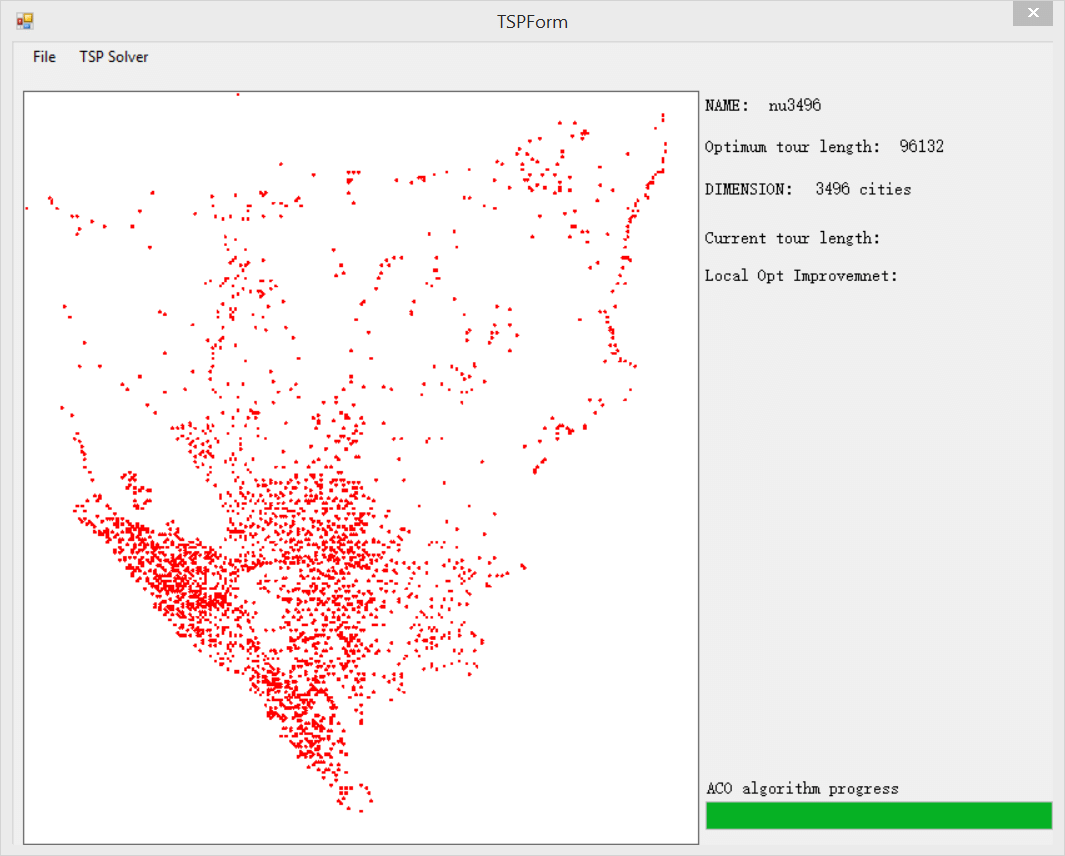


Figure 5.9 3496 cities via ACO Algorithm



1. Conclusion

Through the project, users can start an operation interface to select data and algorithm to get the optimal path.

Through the whole project, the UML and C# can be used well. The Greedy algorithm and the ACO algorithm are acknowledged. UML is used to model the TSP, but the UML should be more expertly used and designed.

The ACO algorithm can calculate the optimal path, but it takes a great deal of time. Not only it can takes much time, but also it cannot generate the only one path, which means, when the calculation of ACO algorithm is finished, the path will change rapidly but the distances are nearly the same. Some of the function mention in the UML cannot be realized because it is too complex. When to solve the huge amount of cities problem, it will occur out time phenomenon by using the ACO algorithm, and the path cannot be generated. This should be improved.

1. References

[1] Gambardella L M, Dorigo M. Solving Symmetric and Asymmetric TSPs by Ant Colonies[C]//International conference on evolutionary computation. 1996: 622-627.

[2] Wikipedia, “Greedy algorithm”, <http://en.wikipedia.org/wiki/Greedy_algorithm>, 25-11-2014.

[3] Booch G, Rumbaugh J, Jacobson I. The unified modeling language user guide[M]. Pearson Education India, 1999.