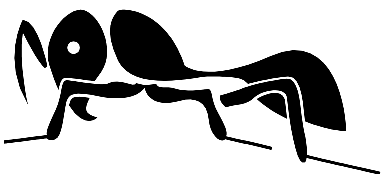
ELECTRONIC, ELECTRICAL AND COMPUTER ENGINEERING

UNIVERSITY OF BIRMINGHAM

Object Oriented Programming Using C# UML

Assignment 2014-15



Swarm Intelligence

Name: DONGLIN YANG

Student ID Number: 1458881

I Introduction

In nature, the ant’s food source is always randomly scattered around the nest, by people’s careful observation, it can be found that ants can always find a shortest path from their nest to food source after a period of time. Meanwhile, the route between the nest and the food source is almost like a straight line, rather than circular or other curved shapes. The ant colony not only can find the shortest route, but also to adapt to the change of environment. For example, in the ant movement path, there is an obstacle suddenly appearing. Initially the distribution of each ant is uniform, the probability of each path by ants selected is same, regardless of the length of path. Ants can leave pheromone on their passed path, and also be able to perceive the pheromone intensity, which in order to guide their direction of motion, ants tend to move toward the direction of a higher concentration of pheromone. At the same time, there will be more pheromone left on the shorter path, select the shorter path ants also increase. It is not difficult to see that, the collective behavior of a large number of ants is a kind of feedback phenomenon, which is more the numbers of ants walk through a path, higher probability for the later ants choose. Individual ants search for food by this method and finally find the shortest path.

Ant colony optimization (ACO) algorithm is a kind of simulation optimization algorithm by imitating ants foraging behavior. It is first proposed by Italian scholar Dorigo M et al in 1991, after that he systematically researched the basic principles and mathematical models of ACO algorithm. Additionally compared with TSP optimization problem, genetic algorithms, tabu search algorithm, simulated annealing algorithm, hill climbing and other simulation experiments, these researches has laid the foundation of ACO algorithm, and has caused the worldwide attention and study.

Earlier ACO algorithm was successfully used to solve the well-known traveling salesman problem (TSP), the algorithm uses distributed positive feedback computation and parallel computing, easy to combine with other methods, but also has strong robustness. However the long searching time and easy to fall into local optimal solution is its obvious defects. This paper will focus on the application of ACO algorithm in TSP, in order to describe in detail how the ACO algorithm applied, it should be introduced TSP first.

II Background

1. The Traveling Salesman Problem

The traveling salesman problem (TSP) refers to a salesman to visit multiple locations, how to find the shortest path which through each site once then return to the starting point. Rules are simple, but the solution may become extremely complex after increasing the number of sites. Generally considering, the most basic solution is to list each alternative routes (that means making permutations and combinations for these sites). Then calculate the total distance of each route, choose the shortest one as the optimal solution. As shown in Figure 1, given four cities A, B, C, D, all of combinations can be represented by a state space diagram. There are six alternative routes, from which it is easy to find the shortest path, additionally due to the symmetric reasons, the total of alternative route changed to 3 finally. It is not difficult to see that when the number of city is N, there are (N-1)! / 2 possible closed tours. With the increasing of city numbers, the route number will become the law of exponential growth that reaches the incalculable point. Currently assuming the city number is 20, there are $(20-1)! / 2 \approx 6.082 \times 10^{16}$ available paths.

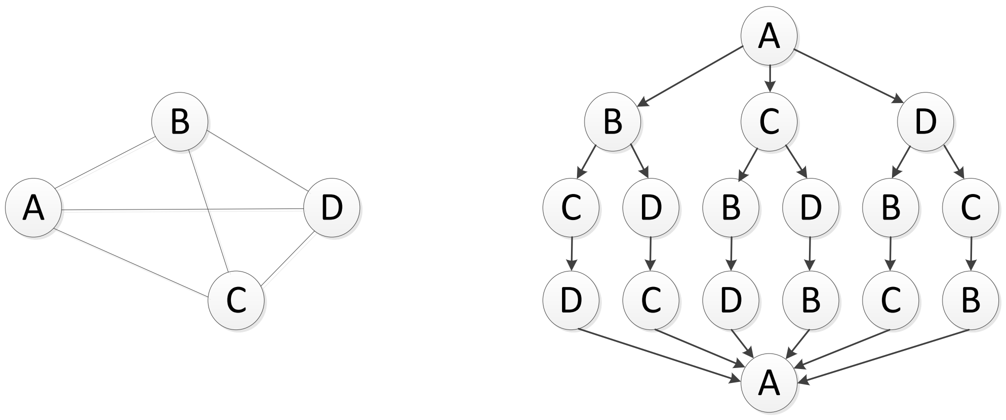


Figure 1 State space Diagram of Four Cities Routes Combinations

2. ACO algorithm applied in TSP

ACO algorithm is a kind of iterative algorithm, a part of ants were released out to look for solutions in each iteration, the city refers to ant in TSP. In this problem, each ant makes a probabilistic choice for the route and they communicate by pheromone. Therefore, the higher concentration of pheromone path, the more ants choose.

ACO used for solving N cities TSP, d_{ij} is defined as the distance between two cities *i* and *j*, *m* is defined as the number of ants in each iteration, $\tau_{ij}(t)$ indicates the amount of pheromone in the arc (*i*, *j*) at time *t*, $\tau_{ij}(0)$ = C which means the amount of pheromone at initial time on each path is C (C is a constant). After the time *n*, the amount of pheromone can be expressed by the following equation (1):

$$\tau_{ij}(t+n) = (1-\rho)\cdot\tau_{ij}(t) + \Delta\tau_{ij} \qquad (1)$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad (2)$$

where $\rho \in (0,1]$ is the amount of pheromone volatilization and $\Delta\tau_{ij}$ is the pheromone of all of passing arc (*i, j*) ants deposited in this iteration. Meanwhile, $\Delta\tau_{ij}^{k}$ represents the quantity of pheromone by each ant *k* left on the arc (*i*, *j*) in this iteration. $\Delta\tau_{ij}^{k}$ can be expressed by the following equation:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q / L^{k} & \text{if } (i, j) \in T^{k} \\ 0 & \text{if } (i, j) \notin T^{k} \end{cases} \quad k = 1..m \tag{3}$$

where Q is the pheromone intensity which is a constant, T^k is the tour by ant k in this iteration and L^k is its length.

$tabu_k (k = 1, 2, \cdots, m)$ used to record the ant k passed city, and $p_{ij}^{k}(t)$ is defined as the probability of ant k choosing to go from city i to city j at time t. $p_{ij}^{k}(t)$ can be expressed as the following equation:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}(t)}{\sum\limits_{s \in allowed_k} \tau_{is}^{\alpha}\eta_{is}^{\beta}(t)} & j \in allowed_k \\ 0 & others \end{cases} \tag{4}$$

where $allowed_k = \{C - tabu_k\}$ represents the ant k's selection of allowed city in its next step. α is the heuristic coefficient which indicates the relative importance of the track. β is the expectation coefficient which determines the influence of arc length over accumulated pheromone. η_{ij} is heuristic function which equals to $1/d_{ij}$. Finally, $NC_{\max}$ is defined as the maximum number of iterations.

In order to do comparison with the ACO algorithm, this program introduces the greedy algorithm which simply chooses the nearest unvisited city as the next one until the ant has passed all cities. The greedy algorithm is a time saving method, but it cannot find the optimal solution.

III Design of Unified Modeling Language

1. Introduction

Unified Modeling Language (UML) is a commonly used in object-oriented modeling approach, which considers from different point of view defining a set of graphs, such as use case diagram, class diagram, object diagram, state chart diagram, sequence diagram, interaction diagram, etc. Use case diagram describes the system function from a user perspective, and points out the function of the operator. Class diagram describes the static structure of the class in the system. Object diagram is an instance of the class diagram and shows the relationship between different objects. State chart diagram describes the states and their controls, commonly used in modeling dynamic characteristic. Sequence diagram describes the dynamic cooperative relations between objects, emphasizing the sequence of sending messages, at the same time shows the interaction between objects. Interaction diagram is similar as the sequence diagram, used to describe the partnership between objects. Then, these diagrams will be used in Use-Case Model, Analysis Model, and Design Model to solve the traveling salesman problem (TSP).

In this paper, the program “TSP-Solution” used to find solutions of TSP. When the user opening this program, there are three menus are “File”, “TSP Solver”, and “About” can be clicked. Meanwhile, the data information, calculate status, program information and a button “Close” can be seen in the right side of interface, also a large blank area displayed in the center. The menu “File” used to load and select the TSP test data, “TSP Solver” used to choose greedy algorithm or ACO algorithm to find solutions, “About” used to show the compiler information. The data information displays the file name, optimum tour length and dimension (the number of cities). The calculate status shows current iterative count which can be set by the user, also the current tour length showed in this area. The program information will display the algorithm progress when the user running this program. If the user wants to exit this program, just click the button “Close”. The large blank area will display a map as the final solution when the program runs out.

2. Use-Case Model

2.1 Use Case View Diagram (Figure 2)

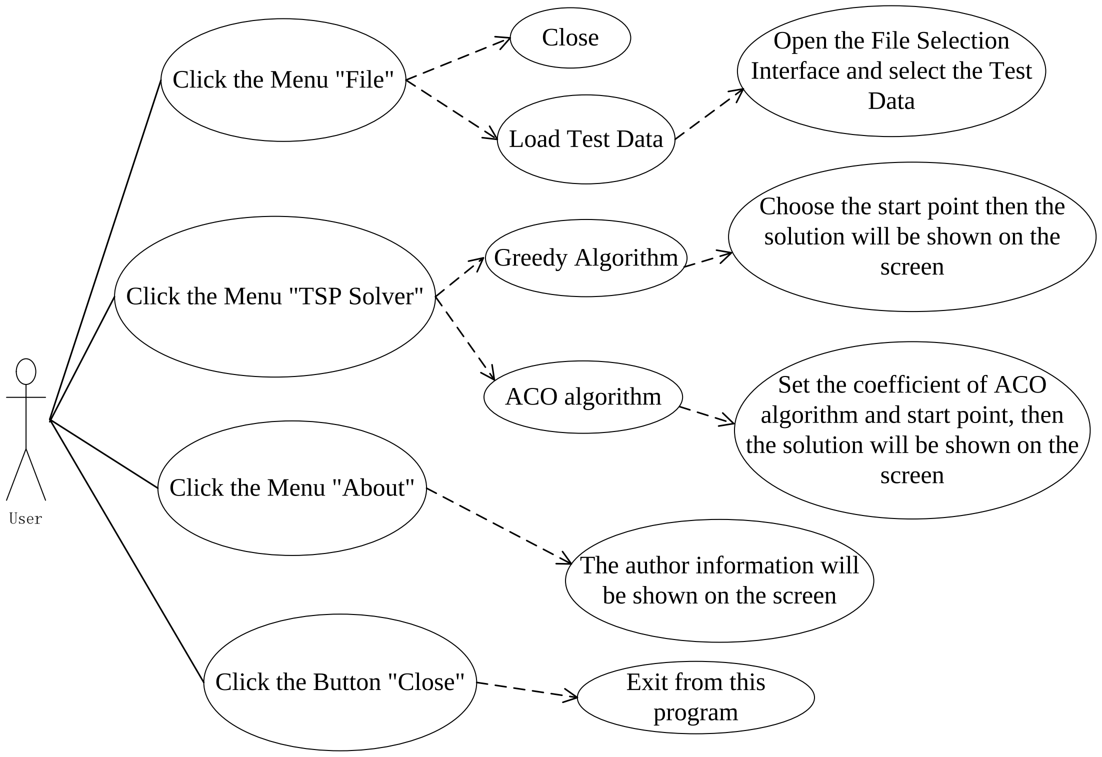

Click the Menu "File"
Close
Load Test Data
Open the File Selection Interface and select the Test Data
Click the Menu "TSP Solver"
Greedy Algorithm
Choose the start point then the solution will be shown on the screen
ACO algorithm
Set the coefficient of ACO algorithm and start point, then the solution will be shown on the screen
User
Click the Menu "About"
The author information will be shown on the screen
Click the Button "Close"
Exit from this program

Figure 2 TSP-Solution Use-case Diagram

2.2 Scenario Descriptions

The user click the menu “File” then the drop-down menu will show two selections “Close” and “Load Test Data”. When the menu “Load Test Data” is clicked the file selection interface will appear on the screen and the user can choose the test data. After that, the selected test data will be traced out as several points in the central blank area, meanwhile in the right side of this program interface will show the data information which includes the file name, optimum tour length and dimension. Below that there is an interface named algorithm progress which displays “Data have read successfully”. If the menu “Close” is clicked, this drop-down menu will be closed.

The user click the menu “TSP Solver” then the drop-down menu will show two selections “Greedy algorithm” and “ACO algorithm”. When the menu “Greedy algorithm” is clicked, there will is an interface which let the user choose the start point then click the button “Start Greedy algorithm”, also the progress bar at the bottom of the screen will show the progress. After a period of time the solution will be displayed as a map in the center, the right side of this program interface will show the current tour length. When the menu “ACO algorithm” is clicked, there will is an interface which let the user set the coefficient of ACO algorithm and select “Use Random” or “Set City Index” which set the start point as the user like. Then click the button “Begin ACO Resolve” and the progress bar at the bottom of the screen will show the progress, also the current iterative count will increased from 1 to NcMax (which is the count of iteration as a kind of coefficient set before) in the right side. After a period of time the solution will be displayed as a map in the center, the right side of this program interface will show the current tour length.

The user click the menu “About” then the author’s information will be shown in the center of screen. The user click the right side button “Close” then an interface “Are you sure to exit” displayed in the center, the user can choose “Yes” to exit from this program or “No” to close this interface.

Note: The user should load TSP test data first then choose a kind of TSP solver to find solutions. Otherwise, the drop-down menu of “TSP Solver” will be grayed which means the user cannot select an algorithm.

2.3 Class Identification

Nouns

User, menus, program, interfaces, test data, blank area, information, algorithm, start point, progress bar, solution, map, coefficient.

Stereotypical classes

Boundary: Menus, interface, information, coefficient, solution

Entity: User, test data, map, author information

Control: Program (greedy algorithm and ACO algorithm belong to program)

Conclusion

Reconsidering the prompts from the list of Nouns and stereotypical classes, a single TSP Window class (which contain the form of setting coefficient of two algorithms and the interface of author information) is sufficient. Therefore, the final class diagram is shown in Figure 3.

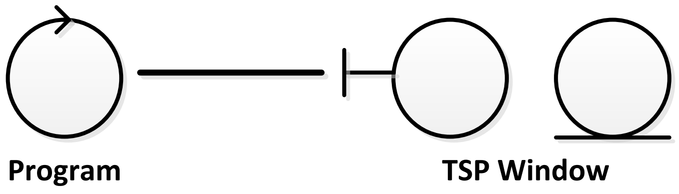


Figure 3 Class Diagram (Stereotypes)

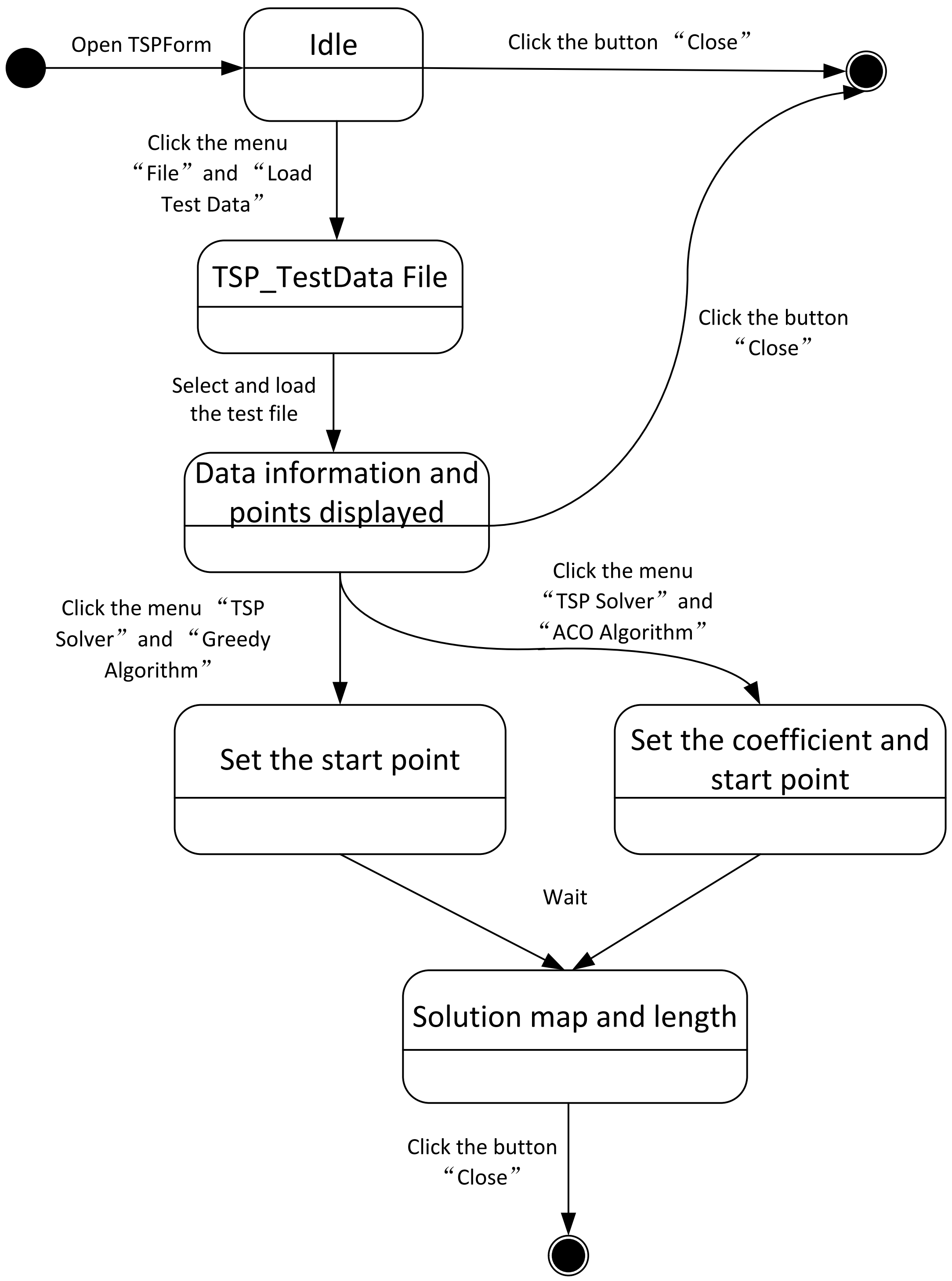
2.4 CRC Cards

Table 1 Class Responsibility Collaboration Cards

Class: TSP Window	
Responsibilities	**Collaborator**
The TSP Window class is responsible for showing the menus, “Close” button, data information, calculate status, algorithm progress, progress bar, selecting and loading the test files, setting the coefficient and start point, tracing out test data’s points in the blank area, and drawing a map as visual for the final solution.	Program

Class: Program	
Responsibilities	**Collaborator**
The program class is responsible for searching for solutions by Greedy algorithm and ACO algorithm respectively, receiving coefficient information and start point from TSP Window class, sending information of solution to TSP Window class.	TSP Window

2.5 Statechart Diagram



Open TSPForm
Idle
Click the button “Close”
Click the menu “File” and “Load Test Data”
TSP_TestData File
Click the button “Close”
Select and load the test file
Data information and points displayed
Click the menu “TSP Solver” and “ACO Algorithm”
Click the menu “TSP Solver” and “Greedy Algorithm”
Set the start point
Set the coefficient and start point
Wait
Solution map and length
Click the button “Close”

Figure 4 Initial Statechart

2.6 Interaction Diagram

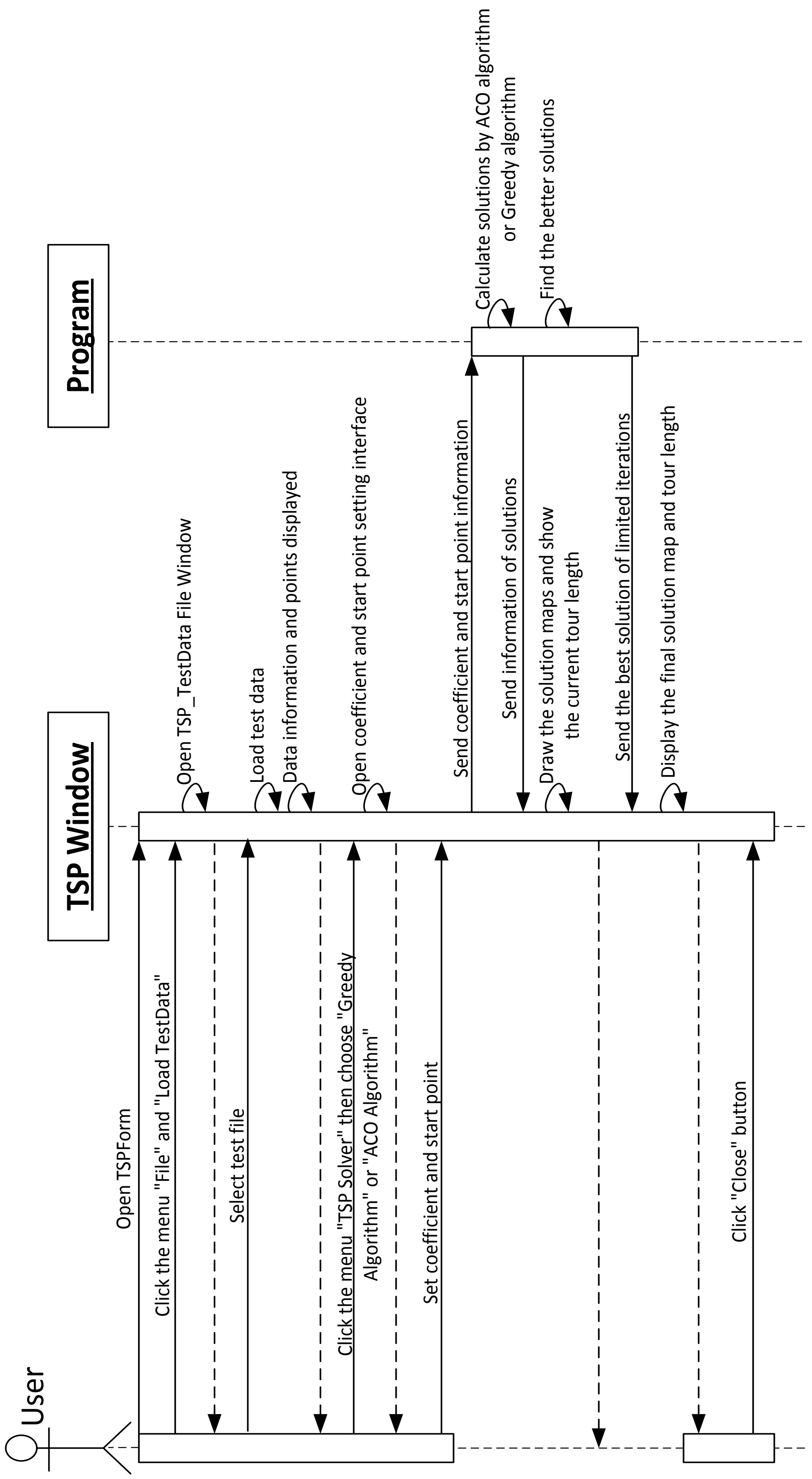

User
TSP Window
Program
Open TSPForm
Click the menu "File" and "Load TestData"
Open TSP_TestData File Window
Select test file
Load test data
Data information and points displayed
Click the menu "TSP Solver" then choose "Greedy Algorithm" or "ACO Algorithm"
Open coefficient and start point setting interface
Set coefficient and start point
Send coefficient and start point information
Calculate solutions by ACO algorithm or Greedy algorithm
Find the better solutions
Send information of solutions
Draw the solution maps and show the current tour length
Send the best solution of limited iterations
Display the final solution map and tour length
Click "Close" button

Figure 5 Interaction Diagram

3. Analysis Model

3.1 Attributes

Class	**Attribute**	**Comment**
TSP Window	strTestDataPath	File path of test data
	FrmACO	Form of parameter set of ACO Algorithm
	FrmGreedy	Form of start point set
	FrmAbout	Form of author information
	Bitmap	Map of point of test data
	Graphics	Map of finding solution
	BestRoute	Record the best hints of city point
	xCoordinateArray	Store x coordinate of city
	yCoordinateArray	Store y coordinate of city
	im	Ant count
	dAlpha	Heuristic coefficient
	dBeta	Expectation coefficient
	dRho	Information volatilization:0<=ρ<1
	dQ	Pheromone intensity
	iNcMax	Iteration count
	iCityCount	City count
	CurrentNc	Current iteration count
	dXMax	The max value of x coordinate of city
	dYMax	The max value of y coordinate of city
Program	N	City count
	M	Ant count
	Nc	Current iteration count
	inittao	Initial amount of pheromone
	[ ] x	x coordinate of N cities
	[ ] y	y coordinate of N cities
	[,] distance	Matrix of city distance

[,] tao	Matrix of the amount of pheromone
[,] eta	Matrix of heuristic function
[,] detatao	Matrix of the amount of pheromone increment
alpha	Heuristic coefficient
beta	Expectation coefficient
rho	Information volatilization:0<=ρ<1
Q	Pheromone intensity
Nc_max	Maximum count of iteration count
[,] tabu	Matrix of recording the ant k passed city
[ ] bestRoute	Array of storing cities belongs to best route

3.2 Methods

Class	**Method**	**Comment**
TSP Window	LoadTestData()	Load test data of TSP
	ReadTSPData(string strFilePath)	Read The TSP file "strFilePath">file name of TSP test data
	DrawCityVertex()	Draw the city vertex
	RestPicBox()	Reset the picture of form Graphics
	DrawMinRoadLine()	Draw current minimum road
	PaintPoint()	Draw the city point
	WriteMsgToList()	Write the message of algorithm progress
	OutPutLength()	Output current minimum length of tsp
	ACOlgorithmToolStripMenuItem_Click()	Open the parameters set form of ACO
	BeginACOCalcate()	Start ACO calculate
	StartGreedyCalculate()	Start calculation of greedy algorithm
	SetProgressBar()	Set progress bar
	BtnClose_Click()	Close this TSP Window

	GreedyAlgorithmToolStripMenuItem_Click()	Open the start point set form of Greedy
	BtnBeginACO_Click()	Begin ACO calculate-code
	BtnBeginGreedy_Click()	Begin greedy calculate-code
Program	Init()	Initial information of the ACO algorithm or greedy algorithm
	FindOptimalPath()	Find the best path of TSP using ACO algorithm or greedy algorithm
	EvaluateSolution()	Calculate the minimum distance of the TSP road by using ACO or greedy algorithm

3.3 Analysis Model Class Diagram

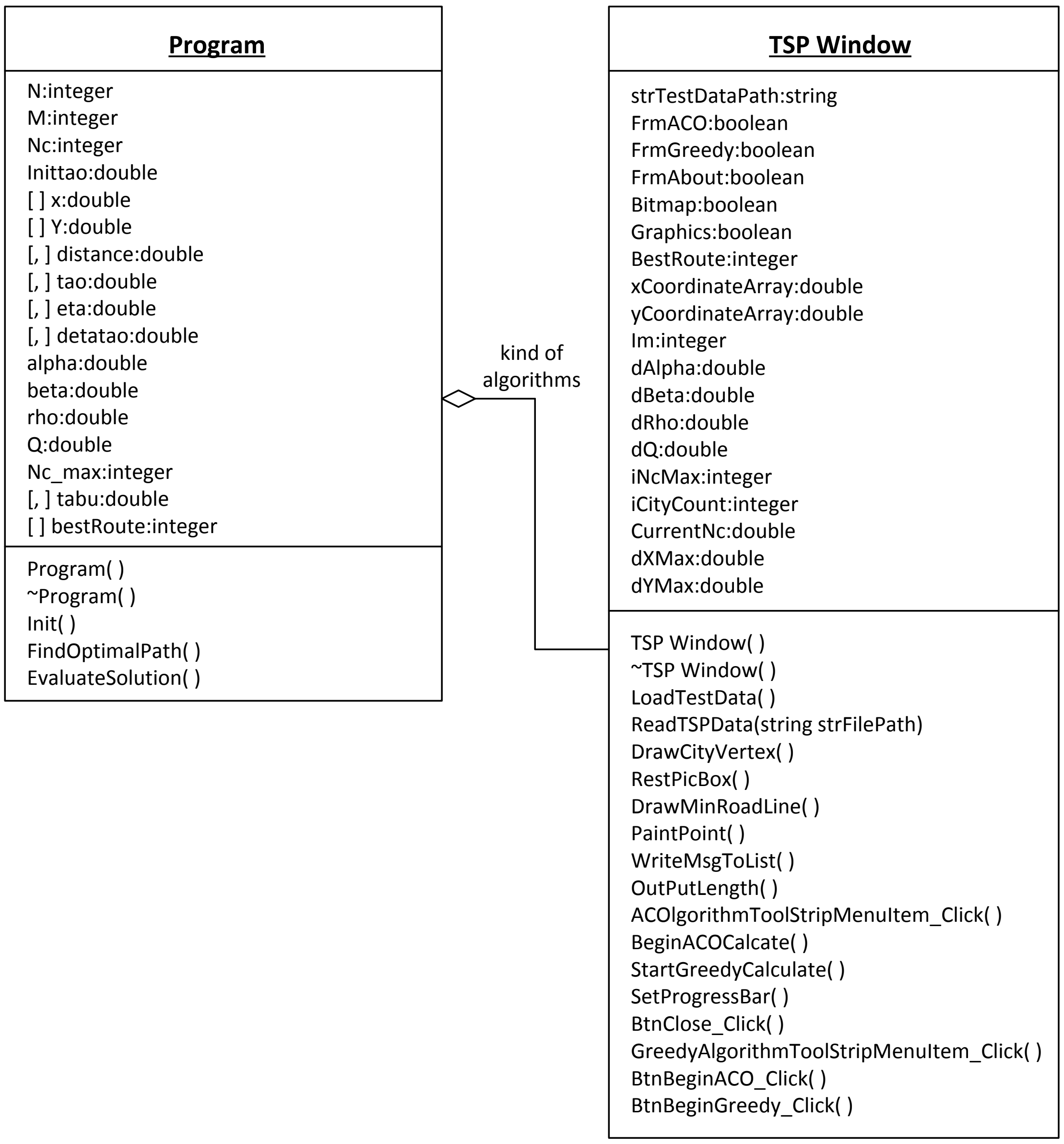


Figure 6 Analysis Model Class Diagram

3.4 Analysis Model Statechart

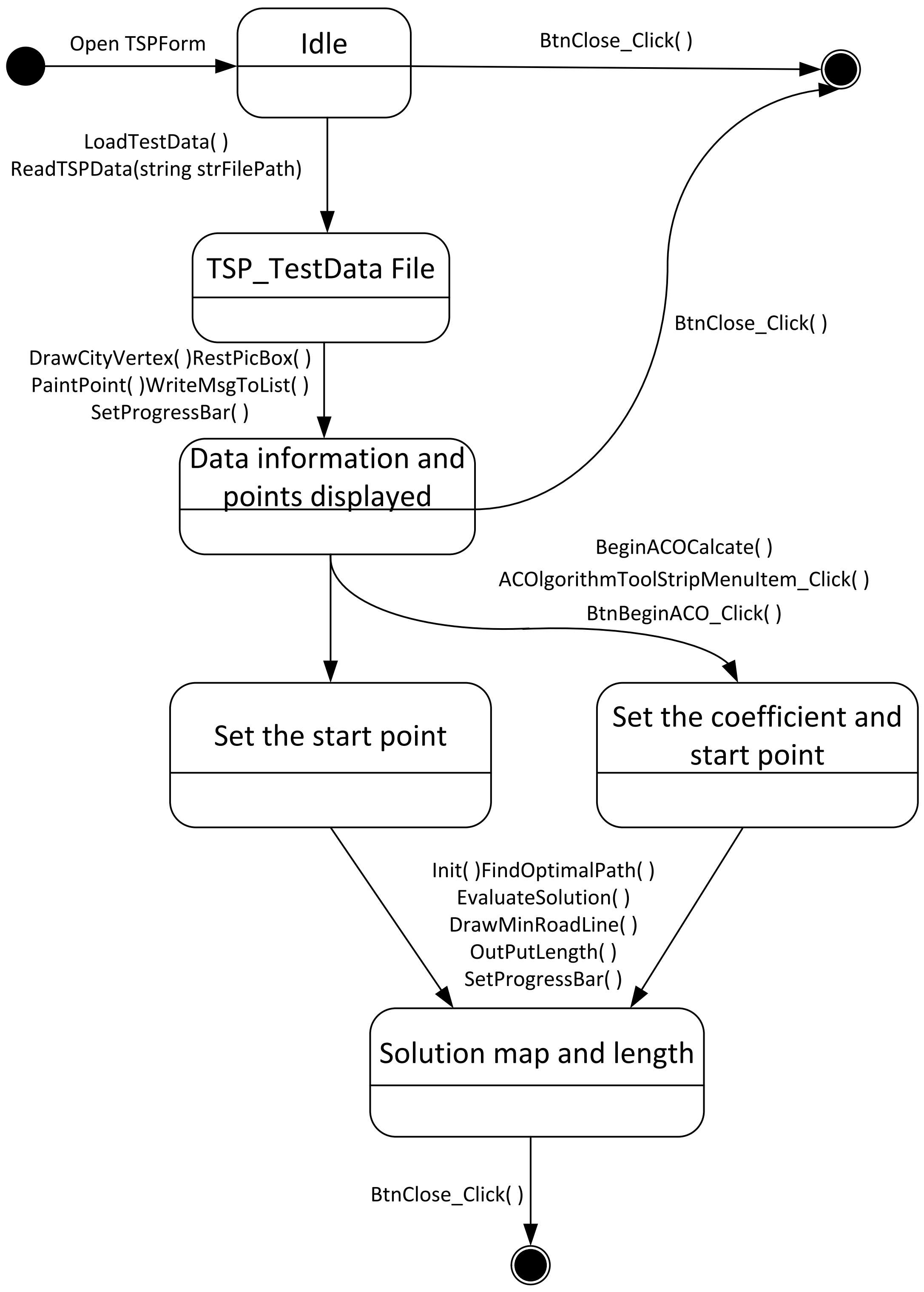


Figure 7 Analysis Model Statechart

3.5 Analysis Model Interaction Diagram

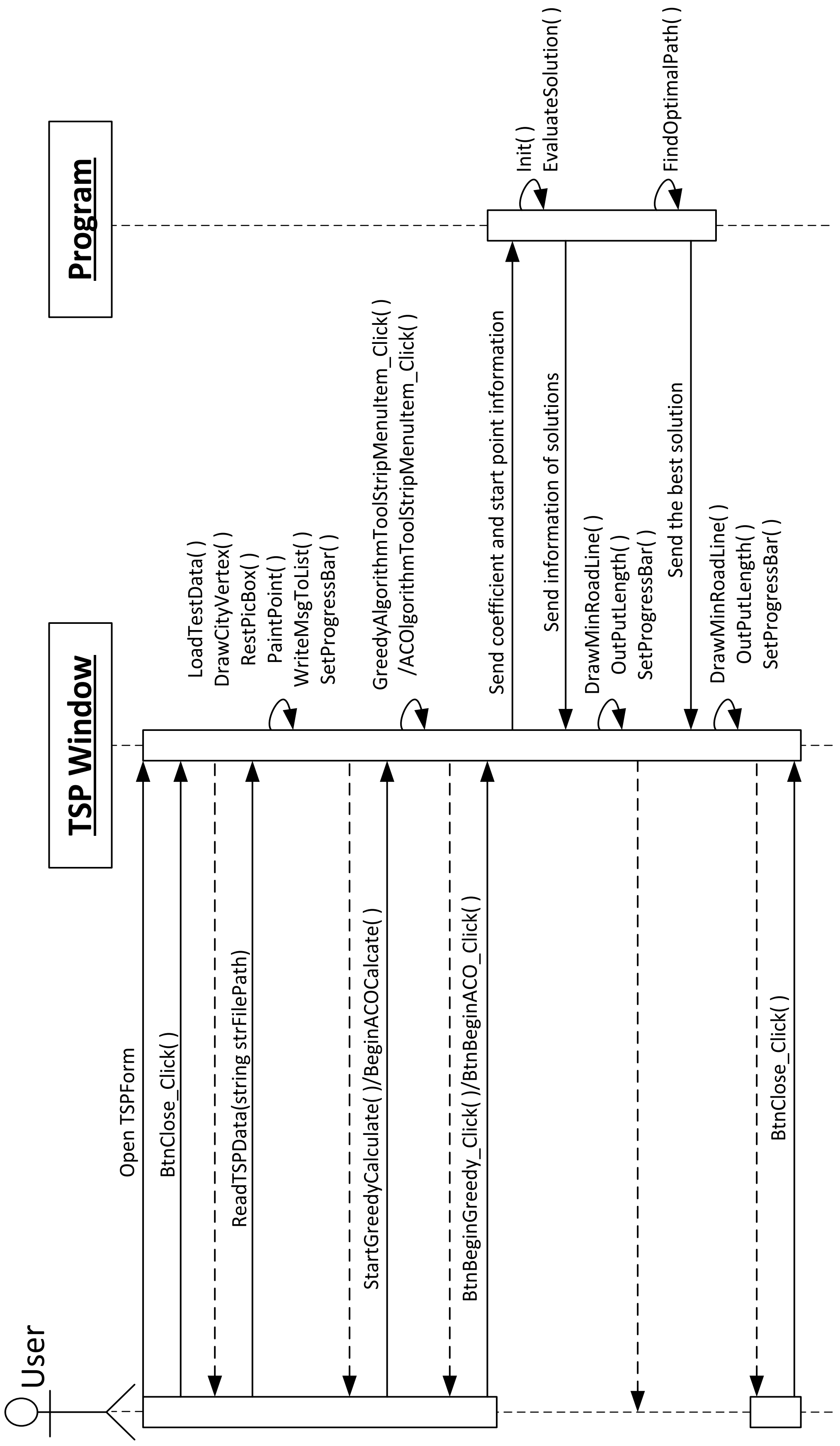


Figure 8 Analysis Model Interaction Diagram

4. Design Model

4.1 Revisit Use-Case Model

The Design meets the requirements.

4.2 Sequence Diagram

The principal qualifiers have already been added.

4.3 Textual Description of Object to Object Interaction

This is a simple system and the various diagrams give a clear description. The following description is intended for introduction.

The TSP Window class is responsible for interacting with the user which includes showing the menus, “Close” button, data information, calculate status, algorithm progress, progress bar, selecting and loading the test files, setting the coefficient and start point, tracing out test data’s points in the blank area, and drawing a map as visual for the final solution.

The Program class is responsible for initializing information of the ACO algorithm or greedy algorithm, calculating solutions and searching for the best path of TSP by using these two algorithms.

4.4 Subsystems

No subsystems are required as this is a simple design.

4.5 Implementation of Non-functional Requirements

No Non-functional Requirements.

4.6 Deployment Model

This program is implemented on a single processor.

4.7 Legacy Issues

No legacy issues.

4.8 Reconsider the Attributes

Incorporated in the Class Diagram.

4.9 Reconsider the Associations

The use of Aggregation remains appropriate.

4.10 StateChart

No further revisions required.

4.11 Class Diagram Showing Visibility

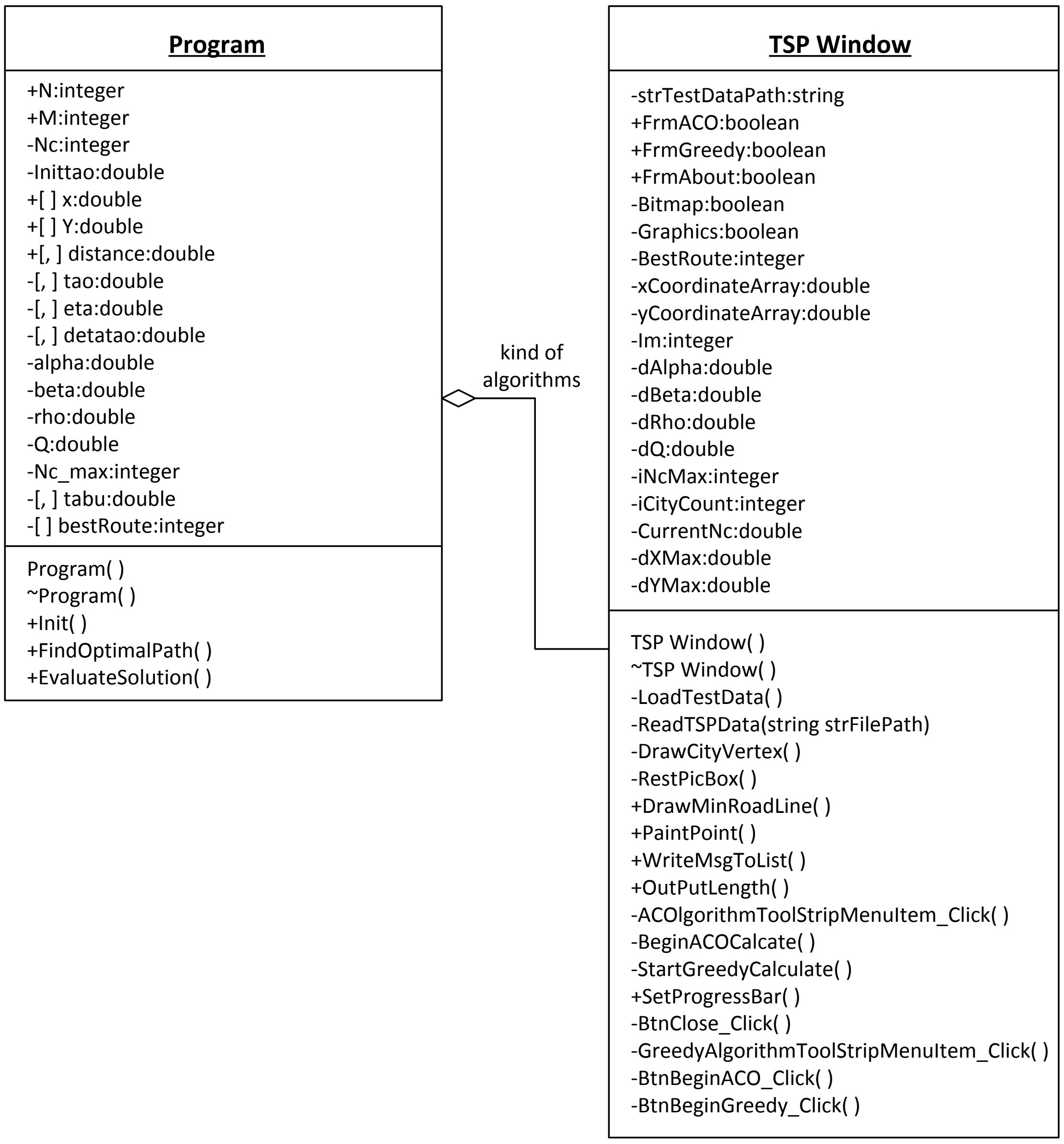
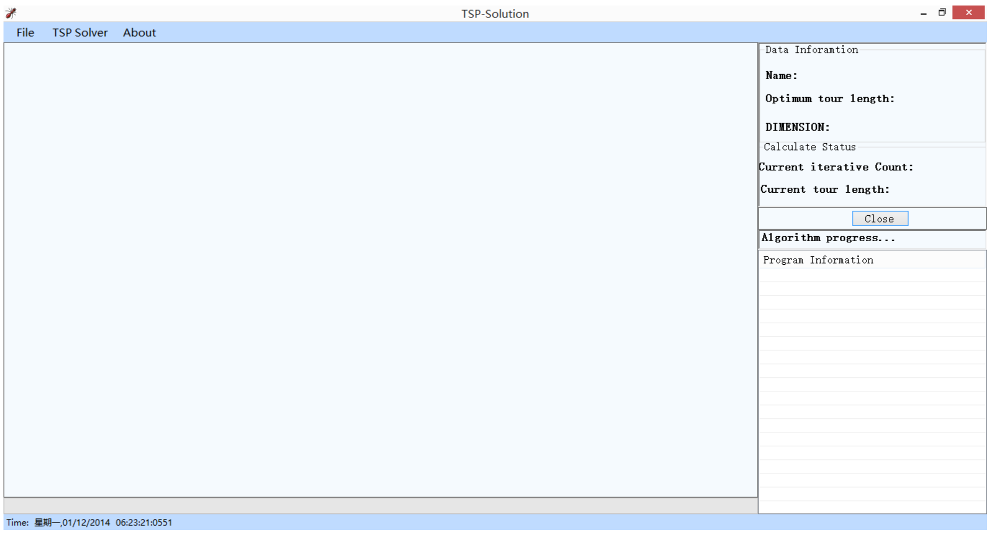


Figure 9 Design Model Class Diagram Showing Visibility

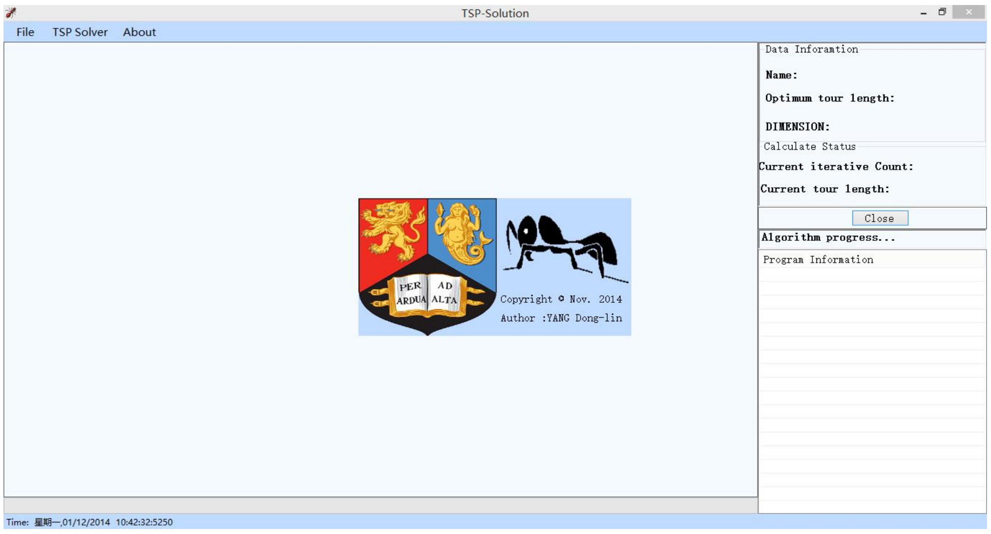
IV Implementation and Test

When the user open the program “TSPForm” the initial interface will be shown as following Figure 10. If the user want to get information about the compiler, just click the menu “About” (Figure 11).



Time: 星期一,01/12/2014 06:23:21:0551

Figure 10 The Initial Interface

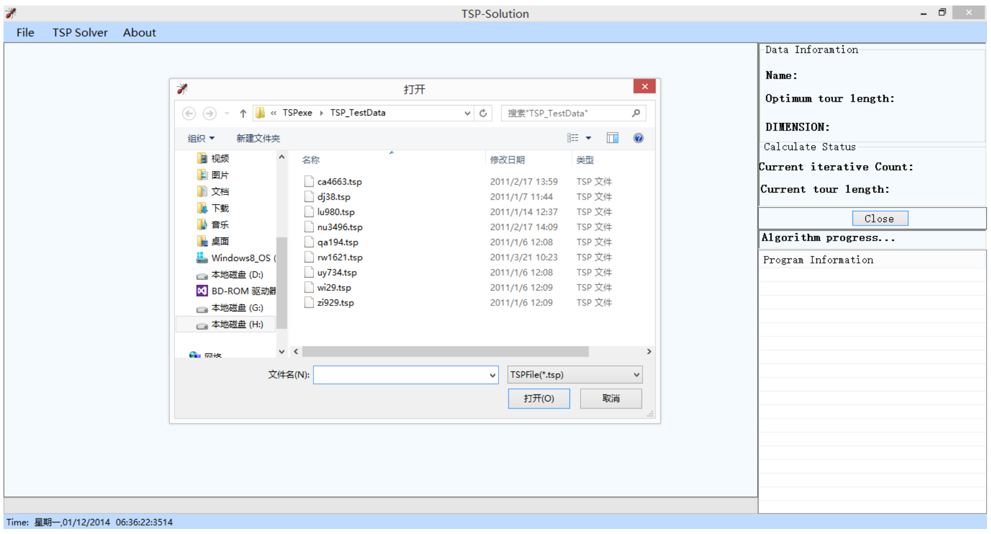


Copyright © Nov. 2014
Author :YANG Dong-lin

Time: 星期一,01/12/2014 10:42:32:5250

Figure 11 “About” Form

Clicking the menu "File" and select "Load TestData", the test data file selection interface (Figure 12) will appear which for user choose test file of TSP.



名称	修改日期	类型
ca4663.tsp	2011/2/17 13:59	TSP 文件
dj38.tsp	2011/1/7 11:44	TSP 文件
lu980.tsp	2011/1/14 12:37	TSP 文件
nu3496.tsp	2011/2/17 14:09	TSP 文件
qa194.tsp	2011/1/6 12:08	TSP 文件
rw1621.tsp	2011/3/21 10:23	TSP 文件
uy734.tsp	2011/1/6 12:08	TSP 文件
wi29.tsp	2011/1/6 12:09	TSP 文件
zi929.tsp	2011/1/6 12:09	TSP 文件

Data Inforamtion

Name:

Optimum tour length:

DIMENSION:

Calculate Status

Current iterative Count:

Current tour length:

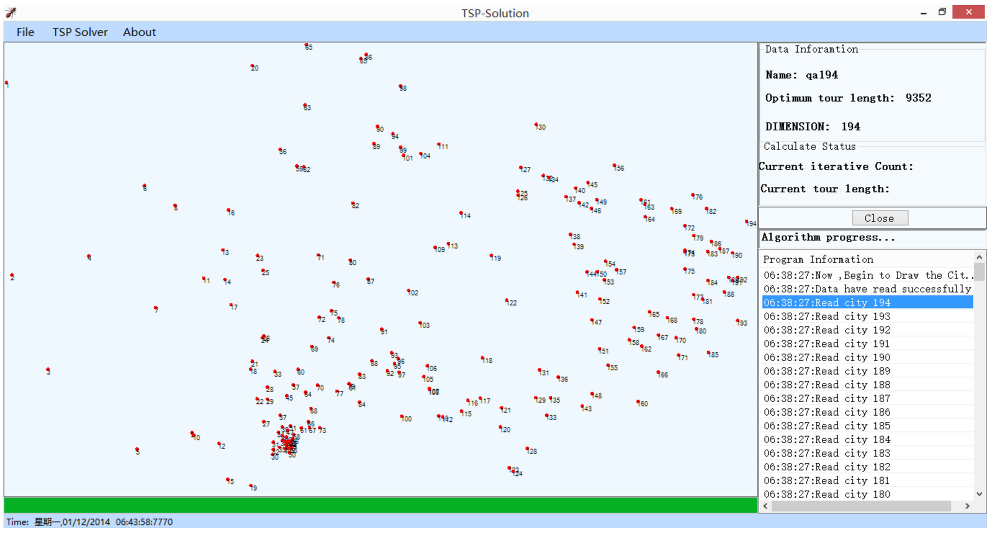
Algorithm progress...

Program Information

Time: 星期一,01/12/2014 06:36:22:3514

Figure 12 Test Data File Selection Interface

After that, the selected test data will be traced out as several points in the central blank area, meanwhile in the right side the data information will be updated which includes the file name, optimum tour length and dimension. Below that there is an interface named algorithm progress which displays the data have read successfully (shown as Figure 13).



Data Inforamtion

Name: qa194

Optimum tour length: 9352

DIMENSION: 194

Calculate Status

Current iterative Count:

Current tour length:

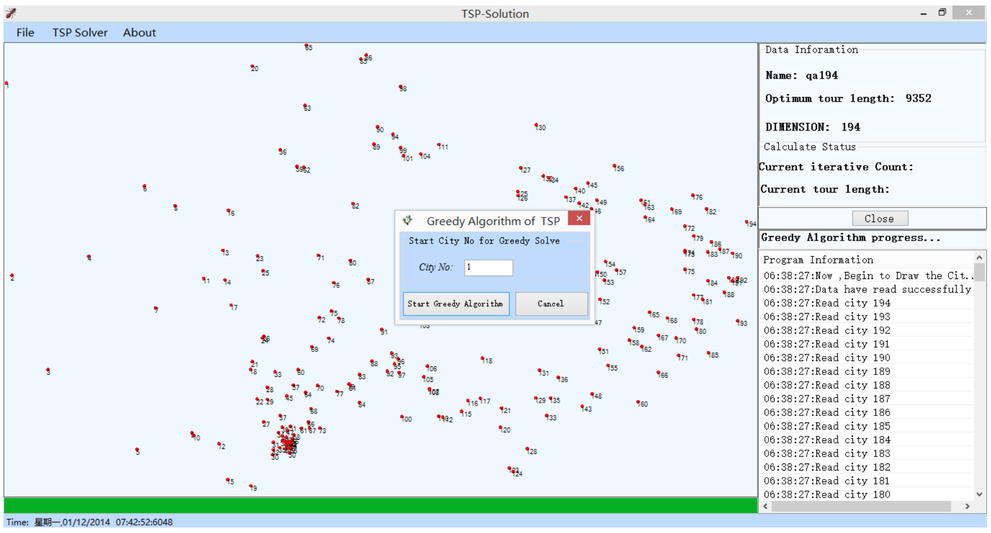
Algorithm progress...

```
Program Information
06:38:27:Now ,Begin to Draw the Cit..
06:38:27:Data have read successfully
06:38:27:Read city 194
06:38:27:Read city 193
06:38:27:Read city 192
06:38:27:Read city 191
06:38:27:Read city 190
06:38:27:Read city 189
06:38:27:Read city 188
06:38:27:Read city 187
06:38:27:Read city 186
06:38:27:Read city 185
06:38:27:Read city 184
06:38:27:Read city 183
06:38:27:Read city 182
06:38:27:Read city 181
06:38:27:Read city 180
```

Time: 星期一,01/12/2014 06:43:58:7770

Figure 13 Data Loaded Interface

Next, the user click the menu “TSP Solver”, then choose “Greedy Algorithm” or “ACO Algorithm”. When “Greedy Algorithm” is selected, the “Start City No for Greedy Solve” form (Figure 14) comes out. When “ACO Algorithm” is selected, the “Parameters Set for ACO Algorithm” form (Figure 15) comes out.



Greedy Algorithm of TSP

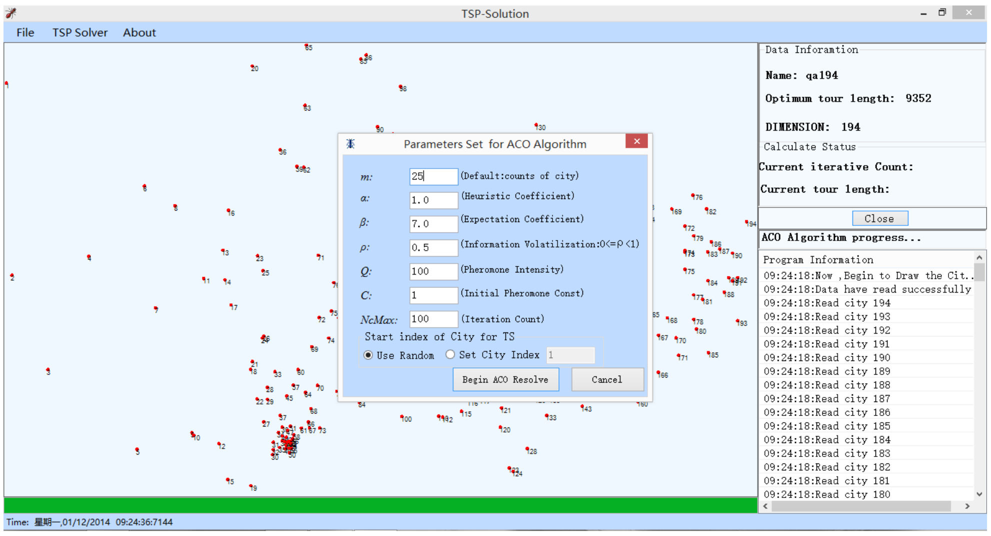
Start City No for Greedy Solve

City No: 1

```
Data Inforamtion
Name:  qa194
Optimum tour length:  9352
DIMENSION:  194
Calculate Status
Current iterative Count:
Current tour length:
Greedy Algorithm progress...
Program Information
06:38:27:Now ,Begin to Draw the Cit..
06:38:27:Data have read successfully
06:38:27:Read city 194
06:38:27:Read city 193
06:38:27:Read city 192
06:38:27:Read city 191
06:38:27:Read city 190
06:38:27:Read city 189
06:38:27:Read city 188
06:38:27:Read city 187
06:38:27:Read city 186
06:38:27:Read city 185
06:38:27:Read city 184
06:38:27:Read city 183
06:38:27:Read city 182
06:38:27:Read city 181
06:38:27:Read city 180
```

Time: 星期一,01/12/2014 07:42:52:6048

Figure 14 “Start City No for Greedy Solve” Form



Parameters Set for ACO Algorithm

m:	25	(Default:counts of city)
α:	1.0	(Heuristic Coefficient)
β:	7.0	(Expectation Coefficient)
ρ:	0.5	(Information Volatilization:0<=ρ<1)
Q:	100	(Pheromone Intensity)
C:	1	(Initial Pheromone Const)
NcMax:	100	(Iteration Count)

Start index of City for TS

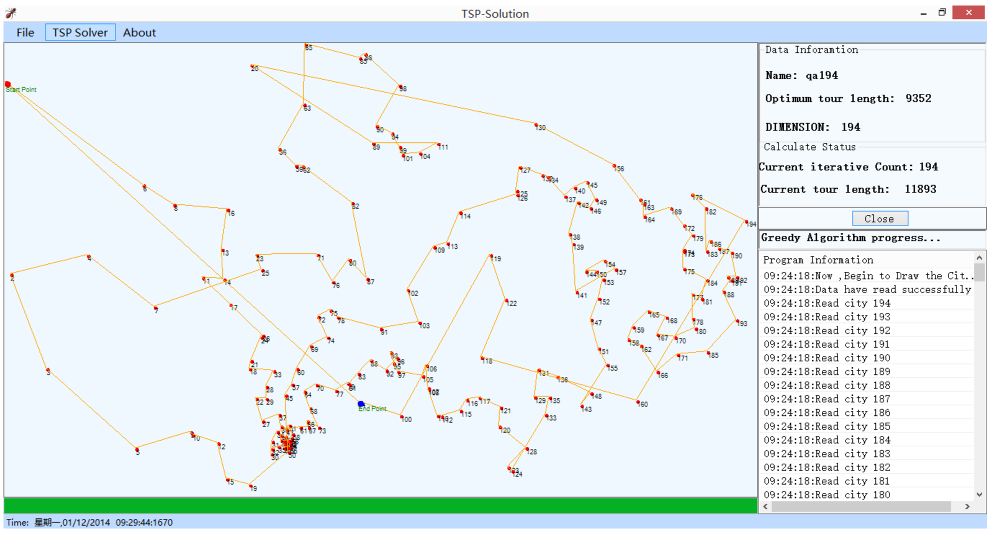
[✓ Use Random] Set City Index 1

```
Data Inforamtion
Name:  qa194
Optimum tour length:  9352
DIMENSION:  194
Calculate Status
Current iterative Count:
Current tour length:
ACO Algorithm progress...
Program Information
09:24:18:Now ,Begin to Draw the Cit..
09:24:18:Data have read successfully
09:24:18:Read city 194
09:24:18:Read city 193
09:24:18:Read city 192
09:24:18:Read city 191
09:24:18:Read city 190
09:24:18:Read city 189
09:24:18:Read city 188
09:24:18:Read city 187
09:24:18:Read city 186
09:24:18:Read city 185
09:24:18:Read city 184
09:24:18:Read city 183
09:24:18:Read city 182
09:24:18:Read city 181
09:24:18:Read city 180
```

Time: 星期一,01/12/2014 09:24:36:7144

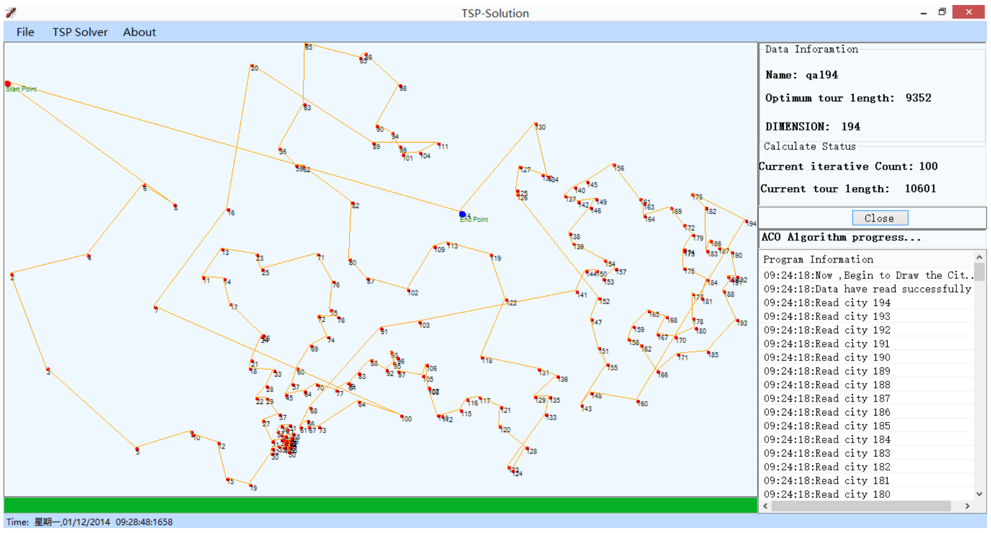
Figure 15 “Parameters Set for ACO Algorithm” Form

After setting the start point of greedy algorithm, click the button “Start Greedy Algorithm” and the solution (Figure 16) will be displayed quickly. After setting the start point and parameters of ACO algorithm, click the button “Begin ACO Resolve” and the best solution (Figure 17) will be shown after a period of time, the waiting time will become longer with the count of iteration increasing.



```
Data Inforamtion

Name:  qa194

Optimum tour length:  9352

DIMENSION:  194
Calculate Status
Current iterative Count: 194
Current tour length:   11893

Greedy Algorithm progress...
Program Information
09:24:18:Now ,Begin to Draw the Cit..
09:24:18:Data have read successfully
09:24:18:Read city 194
09:24:18:Read city 193
09:24:18:Read city 192
09:24:18:Read city 191
09:24:18:Read city 190
09:24:18:Read city 189
09:24:18:Read city 188
09:24:18:Read city 187
09:24:18:Read city 186
09:24:18:Read city 185
09:24:18:Read city 184
09:24:18:Read city 183
09:24:18:Read city 182
09:24:18:Read city 181
09:24:18:Read city 180

Time: 星期一,01/12/2014 09:29:44:1670
```

Figure 16 Solution by Using Greedy Algorithm



```
Data Inforamtion

Name:  qa194

Optimum tour length:  9352

DIMENSION:  194
Calculate Status
Current iterative Count: 100
Current tour length:   10601

ACO Algorithm progress...
Program Information
09:24:18:Now ,Begin to Draw the Cit..
09:24:18:Data have read successfully
09:24:18:Read city 194
09:24:18:Read city 193
09:24:18:Read city 192
09:24:18:Read city 191
09:24:18:Read city 190
09:24:18:Read city 189
09:24:18:Read city 188
09:24:18:Read city 187
09:24:18:Read city 186
09:24:18:Read city 185
09:24:18:Read city 184
09:24:18:Read city 183
09:24:18:Read city 182
09:24:18:Read city 181
09:24:18:Read city 180

Time: 星期一,01/12/2014 09:28:48:1658
```

Figure 17 Solution by Using ACO Algorithm

V Conclusion

From figure 16 and figure 17, for the TSP of 194 cities, the current tour length of greedy algorithm is 11893, however the current tour length of ACO algorithm is 10601. The optimum tour length for 194 cities TSP is 9352. It is not difficult to see that the solution of ACO algorithm is more close to the optimum length than greedy algorithm's solution. Therefore, comparing with greedy algorithm, ACO algorithm is more efficient.

Additional, ACO algorithm compared to the optimal solution there is still a certain gap. This basic ACO algorithm can be improved to get an advanced version, which may be closer to the optimal solution.

Reference

[1] M. Spann, “Object Oriented Programming Using C# Course Slides,” 2014 - 15.

[2] David Pycock, “Object - Oriented Software Design UML: Unified Modelling Language”, Issue 2.2, 2014 - 15.

[3] H.M. Deitel, P.J. Deitel, “Visual C# 2010. How to Program,” 3rd Edition, ISBN 0-13-701183-0.

[4] P. Stevens, R. Pooley, “Using UML: Software Engineering with Objects and Components,” 2nd Edition, ISBN 0-321-26967-5.

[5] Duan Haibin, Wang Daobo, Zhu Jiaqiang, Huang Xianghua, “Development on ant colony algorithm theory and its application,” *Control and Decision*, vol. 19(22), pp. 1321-1327, Dec. 2004.

[6] Duan Haibin, “Ant Colony Algorithm: Theory and Applications,” 1st Edition, ISBN 7-03-016204-8.