ELECTRONIC, ELECTRICAL AND COMPUTER ENGINEERING





# Object Oriented Programming Using C#

**Assignment 2014-15**

******

***Swarm Intelligence***

**Ｎａｍｅ：　ＰＥＩ　ＳＨＩ**

**ＩＤ：　　　１３７９３９６**

1. **Introduction**

The aim of the programming assignment is to use the swarm intelligence to solve the travelling salesman problem (TSP).

* 1. **Swarm Intelligence**

The swarm intelligence is a term which used to describe a new type of simulated evolutionary algorithm and it is also named ant colony optimization (ACO). The name was created by Marco Dorigo in his doctoral thesis, ‘[Ant system: optimization by a colony of cooperating agents](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=484436&tag=1)’, in1992.[1] The swarm intelligence simulated the process of the natural ants foraging, through the interaction of pheromones to find the shortest path from the nest to food, so it is based on the positive feedback principle of information.

According to the observation of entomologist, the entomologist found that although the vision of the ants is not developed, but it can find the shortest path from food source to the nest without any hint, and can search a new best route while the surrounding environment is changed. Through a long term search, they found that when a single ant walks on the ground, in order to avoid get lost by itself, it will release a special kind of secretion, pheromones, and the ant is able to detect a certain range of pheromone secreted by other ants, then, the action of the ant will be affected. With the increase of the pheromones, the probability of choosing this path by subsequent ants is bigger and bigger. Then, the shortest path will be found by the ants. This selection process is the foundation of the swarm intelligence.



Fig.1 the principle of ants foraging

The Fig.1(a) shows that if the point A is the food and the point E is the ant nest,

the ants will not need to choose the shortest path while there does not have an obstacle between the two points. Because the shortest path is the straight line between the two points, so the ants will follow the straight line when they carry food.[2]

The Fig.1(b) shows that if there is an obstacle in the path, the ants will choose

to turn left or turn right in point B and point D. This select will be affected by the

consistence of the pheromones. For example, the ants from point A to point B,

because the path BCD is shorter than the path BHD, so the ant that selects the

path BCD will be faster to arrive the point D than the ants that selects the path

BHD. And if compare the consistence from point D to point B, the consistence

of the path DCB will be bigger than the path DHB, so if the ants go back to their

nest, most of the ants will select the path DCB. And this situation will lead to

the consistence increase.

The Fig.1(c) shows that the most of ants choose to carry food through path

BCD, and the path BCD is the shortest path.

In the practical application, at first, the programming creates some artificial ant.

Then, every single artificial ant searches the surrounding environment

randomly. With the running of the programming , the path of the ants will close

the optimum solution. At last, the optimum solution will be showed.

* 1. **The Travelling Salesman Problem**

The traveling salesman problem means that how to choose a shortest path while a salesman should visit some places, and every place should be visited and only could be visited once.

If a salesman should visit N (N is the numbers of cities) cities, the salesman will have (N-1)！ possible closed tours. With the increase of the numbers of cities, it will be hard to choose the shortest path.[3] This problem belongs to NP-hard problem which means no polynomial time solution, it is hard to solve through simple math formula.

The swarm intelligence is a good method to solve this problem. In the practical application, at first, we define  as the distance between the city I and city j :

 

Then, suppose  is the pheromone that deposited on the path from i to j at time t. Where m is the number of ants and p is the pheromone evaporation constant. So after the time n, the pheromone in the path has changed :

 ①

 ②

 means the ant k deposits the quantity of the pheromone in this loop, and  shows the sum of the quantity of the pheromone in the path between i and j in this loop.

 ＝ 　 ③

Where 。In the process of the ant （＝１，２，…，）moving, means the probability of the ant k moving from i to j :

 ＝ ④

At last, when ,the ants should be put in every cities, and suppose the . Every ant begins to move from city i to city j following the formula ④. is used to control the relative influence of the pheromone that deposits previously. After one loop, get the new  through the formula ①；②；③。Then, we will get the minimum path in this loop. With the new loop beginning, more and more ants will select the same path. If all of the ants choose the same path, the result will be the optimum solution.

Unified Modeling Language (UML) is a kind of OMG standard which built from 1997.[4] It is a kind of graphical language which supports system modeling and software development, and it could provide modeling and visualization support for all stages of software development. And the modeling process is the first step in the development of object-oriented design method.

1. **Use-Case Model**
	1. **Brain storm**

Potential requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Concept | Necessity | Priority | cost | Risk |
| Users open the windows | HIGH | HIGH | LOW | LOW |
| Users choose different maps | HIGH | HIGH | LOW | LOW |
| The points of the maps shows in the windows | HIGH | HIGH | LOW | LOW |
| Users choose different intelligence to solve the DSP | MED | HIGH | LOW | LOW |
| The system begins to compute the optimum solution | MED | MED | LOW | LOW |
| The windows shows the result of the system in every loop | HIGH | LOW | LOW | LOW |

Fig.2 Potential requirements

The fig.2 shows that the risk and cost are not great associated with the whole TSP. Therefore, all of the requirements would be considered.





Fig.3 Use-Case Diagram

* 1. **Survey Descriptions**

Windows: The user chooses the maps and algorithm through the windows. And the windows show the result of the system while the system is running. Through the windows, the user could get the length of the path and the times of the loops.

Note When the maps change, the windows will show different point to user after reset the data.

The system:

 The system includes the ACO algorithm and the Greedy algorithm. The system will follow the needs of user to choose different algorithm to solve the problem and send the result to the windows.

The map:

 The map provides the location of the points. Users could choose the map that accord with their demand, and get the optimum result of the map that selected by users.

* 1. **CRC Cards**

CRC cards means class responsibility collaboration cards, it is used to describe the responsibilities and collaborators of the windows and the system in this project.

|  |  |
| --- | --- |
| **Class: windows** |  |
| **Responsibilities** | **Collaborators** |
| The windows is responsible for choosing the maps and algorithm. And it will send these information to system to do operation. When the system gets the result, the windows will show the result to users. When the map or the algorithm has some changes, the windows will reset the data, and shows the new information for users. | system, map |

|  |  |
| --- | --- |
| **Class: the system** |  |
| **Responsibilities** | **Collaborators** |
| The system is responsible for controlling the running of programming and analysis the final result. When the system receives the information of maps and the algorithm from the windows, it will run the program that designed by designer and send the result to the windows. While the program is running in one and one loop, it will compare the minimum length of the path and send the minimum length to the windows. | windows, map |

|  |  |
| --- | --- |
| **Class: map** |  |
| **Responsibilities** | **Collaborators** |
| The map is responsible for providing the points that users want to get the optimum solution. After the user choose the map, the map will send the location of points to the windows. | windows |

Fig.4 CRC Cards

* 1. **Simple Interaction**

A simple diagram is important to complete the Use-Case Model. It is likely to be a sequence diagram.

Fig.5 Interaction Diagram

* 1. **Statechart Diagram**

The class with the most and the more significant state transitions is the windows, the windows is the controller of the whole project. The states are: select information sending, program running, user selecting, map points sending and the show of windows.

Fig.5 Statechart Diagram

* 1. **Class Diagram**

The class diagram shows class, interface, and the static structure and the relationship between them. In this part, there are three classes: ant, controller and map. It is showed in Fig. 6.

Fig.6 Class Diagram

After the design of the Use-Case Model, the technical specification should be designed. So in the next part, Analysis Model should be built.

1. **Analysis Model**

From Windows Class:

Potential attributes: algorithm selection, map selection, the exhibition of result

Potential methods: algorithm(), map(), path(), shortest length(), current length()

From System Class:

Potential attributes: algorithm running, distance calculation, distance comparation

 Potential methods: ACO algorithm(), Greedy algorithm(), distance calculation(), length comparation()

 From Map Class:

 Potential attributes: different maps

 Potential methods: choose map(), points location()

* 1. **Attributes**

**Class Attribute Comment**

**Windows** Algorithm selection Status

 Map selection Status

 Path exhibition Number

 Length exhibition Number

 Loop times exhibition Number

**System** algorithm running algorithm

Distance calculation Number

 Distance comparation Number

**Map** Maps Number

Fig.7 Attributes

* 1. **Methods**

**Class Method Comment**

**Windows** AlgorithmSelection()

 MapSelection() Sends the information to system

Path()

ShortestLength()

CurrentLength() Receive the information from system

**System** ACO()

Greedy () Follow the selection of user to do one algorithm

DistanceCalculation()

LengthComparation() Send the information to windows

**Map**  MapSelection()

PointsLocation() call the map and send the coordinates to the system

Fig.8 Methods

* 1. **Sequence Diagram**

In this part, the triggers and parameters should be added. When some information have changed, it will lead to some details changes. And it will give the specific methods to every sequence.

Fig.8 Analysis Model Sequence Diagrams

* 1. **State Chart Diagram**

The State Chart Diagram is mainly used to describe the dynamic behavior of an object and shows the state sequence of the object, the state transition event and the action by the state transition.

Fig.9 Analysis model Statechart

* 1. **Non-functional Requirements**

The operation of the system can be stable and efficient. And when the users wants to change the map when the system is running, the system and the windows will reset the data and begin to operate the new one.

1. **Design Model**
	1. **Attributes and methods**

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Attributes** | **Data types** | **Methods** |
| **Windows** | Algorithm selection | Status | AlgorithmSelection() |
|  | Map selection | Address | MapSelection() |
|  | Path exhibition | Number | Path() |
|  | Length exhibition | Number | ShortestLength() |
|  | Loop times exhibition | Number | CurrentLength() |
| **System** | algorithm running | algorithm | ACO()/ Greedt() |
|  | Distance calculation | Number | DistanceCalculation() |
|  | Distance comparation | Number | LengthComparation() |
| **Map** | Maps | Number | MapSelection() |
|  |  |  | PointsLocation() |

Fig.9 Analysis of the attributes and the methods

* 1. **Interaction Diagram**

The principle qualifiers have already been added in Fig.8. To the ACO, the interaction diagram is Fig.10. The program is the controller of the system.

Fig.10 the interaction diagram of the ACO

* 1. **Statechart**

No further revisions required for the whole project. To the ACO, the statechart is the Fig.11.

The class with the most and the more significant state transitions is the programming, and it is the controller of the whole system.

 The states are : ant running, ant detecting, ant releasing, controller and map points changing.

Fig.11 The statechart of the ACO

* 1. **Non-functional requirement**

when the users wants to change the map when the system is running, the system and the windows will reset the data and begin to operate the new map. It should have a void reset() and give it defined as high priority.

1. **Test Models**

The Fig.12 shows the process of running with the Greedy algorithm in the situation of 3496 cities.



Fig.12 The Greedy algorithm

The Fig.13 shows the process of running with the swarm algorithm in the situation of 194 cities



Fig.13 ACO

1. **Conclusion**

 The aim of this project is achieved at the final, although the speed of running is not very quick and the final result keeps a little change after a large number of loops.

**Referrnence:**

**[1] Wikipedia, “Ant colony optimization algorithms”, [Online], http://en.wikipedia.org/wiki/Ant\_colony\_optimization\_algorithms,[Last Modified: 17 Nov 2014, 11:30].**

**[2] James F. Kennedy, “Swarm Intelligence”, Information Technology—General and others, Morgan Kaufmann, 2001, pp.5-15.**

**[3] Gillett B E, Miller L R, “ A heuristic algorithm for the vehicle-dispatch problem[J]”, Operations research, 1974, 22(2),pp. 340-349.**

**[4] Simon Bennett, “Object-oriented systems analysis and design using UML”, London: McGraw-Hill, 2002, pp.13-14.**