1 Introduction

1.1 Abstract

From the perspective of testing goal, normal software testing engineering concentrates on defects detecting. In this process, crucial loop will be tested; logic error (run time error) will also be detected. However, these processes are omitted in this particular testing engineering (at component testing stage). Because the Cyclomatic Complexity will increase dramatically when test the drive loop of the algorithm On the other hand, it is impossible to test every path in the each crucial loop because the randomly of the algorithm. However, as these algorithms have very high invoking rate in the shortest path finding subsystem, it has a very high probability that every path in each crucial loop will be reached so that the logic error (and some other defects) can be fixed according to the algorithm design and programmer experience when performing integration testing. In this testing document, discussion about defects detecting and (static/logic) error fixing are omitted and more concentration and made on the relation between algorithm parameters and its performance. Moreover, discussion about algorithm parameters and the performance of shortest path finding subsystem are also concentrated.

1.2 Test target

In this software algorithm testing engineering process, three components and an integrated subsystem are tested:

- **Ant Colony Algorithm**

One of the system main algorithm components which responsibility is generates an optimal travel sequence on a given undirected weighted complete graph.

- **Generic Algorithm**

One of the system main algorithm components which responsibility is generates an optimal travel sequence on a given undirected weighted complete graph.

- **Path generator algorithm**

It has the responsibility to generate a path given a sequence of nodes on the real library map.

- **Shortest path finding subsystem.**

This is the sub system which integrates the previous three components. It has the responsibility to interact with the database to access data for use and provide computation results for the web server.

The reference documents when perform testing process is the design document of each test target.

1.3 Test objective

1.3.1 Objective One

Clear the relationships between the two algorithms and their parameters. Test the performance of

the algorithms according to different values of parameters. And then represent the relationships intuitionally or abstractly, along with some rational analysis. Draw forecast conclusion between parameters and performance according to their relationships. And on the basis of this conclusion and experimental data setting rational parameters' values for algorithms. Repeat the test again, compare and analyze the results to expected outcome. Finally, analyze the performances of the two algorithms and give explanation and prospect.

1.3.2 Objective Two

Testing the path generator algorithm and trying to find potential logic errors. Setting special situation according to the algorithm's logic to test and verify the implementation of the algorithm's logic is correct. Test other cases to find limitations of the algorithm. Then compare to the expected results and give explanation and prospect.

1.3.3 Objective Three

Connecting all the components of the optimal path system to test and verify the correctness, security, consistency, and usable of connections. At the same time argue the details hidden and methods encapsulation.

1.3.4 Objective Four

Connect database and web server to test and verify the correctness, security, consistency, and usable of connections. In the meantime, test the joint system for a few times to get the final visual path and then give analysis and prospect.

1.3.5 Objective Five

Test the capability of the whole system.

1.4 Algorithm performance description

In this section, several algorithm performance identifiers are described in details.

1.4.1 Compute capability

Compute capability is a very important algorithm performance identifier because it determines the boundary of problem scale that it can deal with. Higher compute capability means the algorithm can solve larger problem and consequently it potentially means that this algorithm can be applied to wider application area. However, higher compute capability doesn’t means everything, an algorithm with a high her compute capability does not mean it also has a high compute accuracy or low computing complicity. So, other algorithm performance identifiers are also need to be measured.

1.4.2 Algorithm complicity

It always important to notice that even though the algorithm compute capability is very high, it is still unacceptable that the algorithm will terminate in a unreasonable time or the time (or space) complicity increase dramatically when the problem scale increase linear. The algorithm complicity identifier given an approximate evaluation of the algorithm complicity. Through the testing process, the algorithm complicity will be measured and a mathematical formula will also be given. Moreover, the algorithm complicity also limits compute capability because even though the algorithm can generate an optimal solution given a particular problem, it is still unacceptable if this process takes an unreasonable time.

1.4.3 Algorithm accuracy

It is obvious that the algorithm accuracy is one of the most important algorithm performance identifier that is needed to be measured or observed. Normally, this identifier is not represented by mathematical formula or quantified in accurate values. Instead, this parameter is observed from the comparison between the output solution and the expected solution. In this testing engineering, this strategy is also employed and some comparisons are made to given a full evaluation of this algorithm performance parameter.

1.4.4 Convergence rate

Another algorithm identifier that is very important for algorithm performance measurement is the algorithm convergence rate. It is a measure of how fast the candidate solution evaluates itself and approach to better solution. Basically, it should be viewed as the most important performance parameter because it has influence on both algorithm complicity and algorithm accuracy. In specific, when the convergence rate is decrease, either the algorithm complicity increased or the algorithm accuracy decreased. On the other hand, when the convergence rate increased, both algorithm accuracy and algorithm complicity may decrease. That means, when the convergence rate increased algorithm can generate better solution faster but may fall into local optimal solution. On the other hand, if the convergence rate decreased, it needs takes more time to generate a accuracy enough solution or the solution accuracy will decreased. However, the algorithm convergence rate is very hard to capture into a mathematical formula. There are three main reasons: the first reason is the randomly propriety of the algorithm, the second reason is there are many algorithm parameters can influence the convergence rate and the third reason is it is very hard to determine whether an algorithm parameter has a direct relation with the convergence rate or it only has an indirect influence. Consequently, it is impossible to quantify this algorithm performance parameter, the only reasonable and effective approach to evaluate it is observing other algorithm identifiers or output some media data to reflect it. Then by modifying some repellent algorithm parameters, the algorithm convergence rate is observed and controlled.

1.5 Expected result

The expected results contain the two algorithms performance and the path generator algorithm performance. And for each component, the representation of its performance has evaluation standards which analyzed respectively.

1.5.1 Expected results of the two algorithms

1.5.1.1 Computational capabilities

The computational capabilities of the two algorithms are different; Generic algorithm is about thirty books while the Ant Colony algorithm is sixty books.

1.5.1.2 Complexity

The complexities of the two algorithms are same which is $O(n^3)$ (n is the input problem scale)

1.5.1.3 Accuracy

According to the algorithms' logic and design.in typical problem scale, it has very high probability to find the shortest path.

1.5.2 Expected results of the path generator algorithm

1.5.2.1 Computational capabilities

The path generator algorithm can always generate the path to fetch book.

1.5.2.2 Complexity (run time)

The complexity of the algorithm is $O(n)$.

1.5.2.3 Accuracy

This algorithm can always generate the correct path.

1.5.2.4 Limitations

This algorithm can only handle maps that correspond to algorithm assumptions. For other maps, it cannot promise to generator correct path.

2 Test details

2.1 Test data description

The original problem can be abstract as the description below:

Given an undirected weighted complete graph $G = (V, E)$ which $V = \{1, 2 \cdots, n\}$ and $E = \{e_{ij} = (i, j) \mid i, j \in V, i \neq j\}$ is vertex set and edge set respectively. $d_{ij}(i, j \in V, i \neq j)$ is the distance between vertex i and j, where$d_{ij} > 0, d_{ij} \neq \infty \, and \, d_{ij} = d_{ji}$. The mathematical model for this problem can be constructed as below:

$$\min F = \sum_{i \neq j} (d_{ij} * x_{ij}) \quad (1)$$

$$st. x_{ij} = \begin{cases} 1, & edge\ e_{ij}\ is\ in\ the\ optimal\ path \\ 0, & edge\ e_{ij}\ is\ not\ in\ the\ optimal\ path \end{cases} \quad (2)$$

$$\sum_{i \neq j} x_{ij} = 1, i \in V \quad (3)$$

$$\sum_{i \neq j} x_{ij} = 1, j \in V \quad (4)$$

$$\sum_{i,j \in S} x_{ij} = |S|, S\ is\ a\ subgraph\ of\ G \quad (5)$$

In this model, |s| is the number of vertices in S. (1) is the objective function, which gives the shortest path that visits each vertex in G exactly once. (2) ~ (4) give a constraint of each vertex has only one in-edge and out-edge. (5) guarantee there is no sub circle in shortest path circle.

For the purpose to assign an appropriate algorithm parameters value for the both of these two algorithms, computer simulation experiment is employed. In this testing engineering, the following particular CTSP problem is designed and testing input:

- Problem scale n = 10;
- Distance matrix:

$$\begin{bmatrix} 0 & 98 & 99 & 27 & 89 & 175 & 96 & 26 & 64 & 49 \\ 98 & 0 & 54 & 98 & 73 & 18 & 77 & 89 & 63 & 64 \\ 99 & 54 & 0 & 61 & 36 & 69 & 46 & 53 & 13 & 81 \\ 27 & 98 & 61 & 0 & 86 & 75 & 15 & 87 & 21 & 87 \\ 89 & 73 & 36 & 86 & 0 & 46 & 19 & 87 & 65 & 46 \\ 175 & 18 & 69 & 75 & 46 & 0 & 26 & 53 & 21 & 23 \\ 96 & 77 & 46 & 15 & 19 & 26 & 0 & 46 & 36 & 54 \\ 26 & 89 & 53 & 87 & 87 & 53 & 46 & 0 & 75 & 65 \\ 64 & 63 & 13 & 21 & 65 & 21 & 36 & 75 & 0 & 54 \\ 49 & 64 & 81 & 87 & 46 & 23 & 54 & 65 & 54 & 0 \end{bmatrix}$$

2.2 Ant Colony Algorithm

2.2.1 Algorithm parameters description

2.2.2 Ant population size

The initial ant population size M is given according to the function below:

$$\mathrm{M} = \begin{cases} 0.5 * |\mathrm{V}| & n \geq 20 \\ 10 & n < 20 \end{cases}$$

2.2.3 Edge selection probability

In each iteration, for each ant, it will transfer from one vertex to another vertex. The transformation probability of an ant is determined by the following function:

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}{\sum_{l \subset U}^{n'} (\tau_{il}^{\alpha} * \eta_{il}^{\beta})} & j \in U \\ 0 & j \notin U \end{cases}$$

$$U = V - tabu_k = \{1, 2, 3 \cdots n'\}$$

In this formula, p_{ij}^{k} represents the transformation (from of vertex i to j) probability of ant k; $tabu_k$ is the tabu list of ant k which represents the vertices set that ant k has been visited; $U = V - tabu_k$ represents the available vertices set of ant k in the next transformation; τ_{ij} is the pheromone strength at edge e_{ij}; η_{ij} is the pheromone visibility at edge e_{ij}; α and β are two independent parameters which represent information heuristic factor and expectation heuristic factor respectively.

2.2.4 Pheromone matrix modification parameter

After the tabu list of each ant become empty, algorithm should modify the pheromone matrix according to the following function:

$$\tau_{ij} = (1 - \rho) * \tau_{ij} * \Delta\tau_{ij} \quad for\ e_{ij} \subset E$$

$$\Delta\tau_{ij} = \sum_{k=1}^{M} \Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & if\ ant\ k\ had\ passed\ edge\ e_{ij} \\ 0 \end{cases}$$

In this function, ρ is the pheromone volatilization correlation; $1-\rho$ is pheromone Residual factor; Q is pheromone intensity; L_k is the total cost of ant k generated candidate solution.

2.2.5 Relation between algorithm parameters and algorithm performance prediction

In this section, some prediction about relations between some algorithm parameters and algorithm performance are made.

2.2.5.1 Relation between algorithm performance and ant population size

According to the principle behind the ant colony algorithm, relations between algorithm performance and ant population size is listed as below:

- It has no influence on the algorithm compute capability. Actually, the population is determined by the input problem scale
- For the same impute problem, when the population size increased, the algorithm complicity may or may not increase. However, the real operation time consumed by the implemented algorithm program will definitely increase.
- For the same impute problem which large enough that the algorithm cannot always generate an optimal solution, increase the population size may increase the algorithm accuracy. That means, larger ant population size in the algorithm will improve algorithm accuracy.
- It has no influence on the algorithm convergence rate because it has no influence on the algorithm operators (ant transaction or pheromone matrix modification)

2.2.5.2 Relation between algorithm performance and parameters in edge selection probability formula

According to the mathematical formula and principle behind the ant colony algorithm, relations between algorithm performance and parameters in the edge selection probability formula is listed as below:

- It has no influence on the algorithm compute capability.
- There is no clear relation (or say it is hard to analysis from the principle behind the algorithm) between algorithm parameters such as pheromone visibility η_{ij} , information heuristic factor α and expectation heuristic factor β and the other algorithm performance parameters. The only way to measure these parameters is to generate s serious of testing data and observe the relation between these parameters and other algorithm performance parameters.

2.2.5.3 Relation between algorithm performance and parameters in pheromone matrix modification formula

- It has no influence on algorithm compute capability and algorithm complicity.
- When the pheromone intensity Q increased, the algorithm convergence rate will also increase because already found "good" candidate solution will have a higher selected probability. However, the algorithm accuracy may decrease because the algorithm may fall into local optimal solution.
- When the pheromone intensity Q decreased, the algorithm convergence rate will decrease. If the algorithm complicity remained (that means ant population size and iteration times does not changed), the algorithm accuracy may decrease because the algorithm may lose some already found "good" candidate solutions.
- Pheromone volatilization ρ has similar effect on the algorithm performance parameters but in an opposite way.

2.2.5.4 About algorithm compute capability

From the discussion above, it seems that there is no algorithm parameters can affect this algorithm performance parameter. Theoretically, the algorithm compute capability is infinite. From the perspective of programming implementation, the only factor the can affect the compute capability is the value representation accuracy of the programming language.

2.2.6 Relation between algorithm parameters and algorithm performance

2.2.6.1 Pheromone trail persistence

2.2.6.1.1 Parameter propriety analyze

Like most of evolution simulation algorithm, ant colony algorithm also has the disadvantages such as low convergence rate or easy fall into local optimal solution; and the pheromone residual $1-\rho$ has a direct influence (mentioned above) on the algorithm search capability (algorithm accuracy) and convergence rate: under the influence of $1-\rho$, when the input problem scale is large, it has a high probability that pheromone on the path that has never been searched decrease to zero. Consequently, the global searching capability decreased. More specifically, when $1-\rho$ becomes too large, the algorithm randomly characteristic will be influenced because the previous searched path has higher probability too be searched again. On the other hand, if the pheromone residual is too small, the randomly characteristic is guaranteed by sacrifice algorithm convergence rate.

2.2.6.1.2 Testing result representation

In this particular test, pheromone residual is set as independent variable and the other algorithm

parameters are set as constant. In specific:

- Ant population size **M = 5**
- Pheromone intensity **Q = 1**
- Information heuristic factor $\boldsymbol{\alpha = 0.9}$
- expectation heuristic factor $\boldsymbol{\beta = 1.0}$
- iteration terminate condition: average candidate solution cost difference between two generations is less than **0.001**
- pheromone trail persistence $\boldsymbol{\rho \in \{0.3, 0.5, 0.7, 0.9, 0.95, 0.99\}}$

The testing result is represented as below:

ρ	Length of best path	Iteration times
0.3	379.7287	17
0.5	379.7303	26
0.7	379.7321	44
0.9	379.7547	115
0.95	379.8639	165
0.99	379.2012	567

Table. 1 The influence to the Ant Colony algorithm properties for different pheromone trail persistence ρ

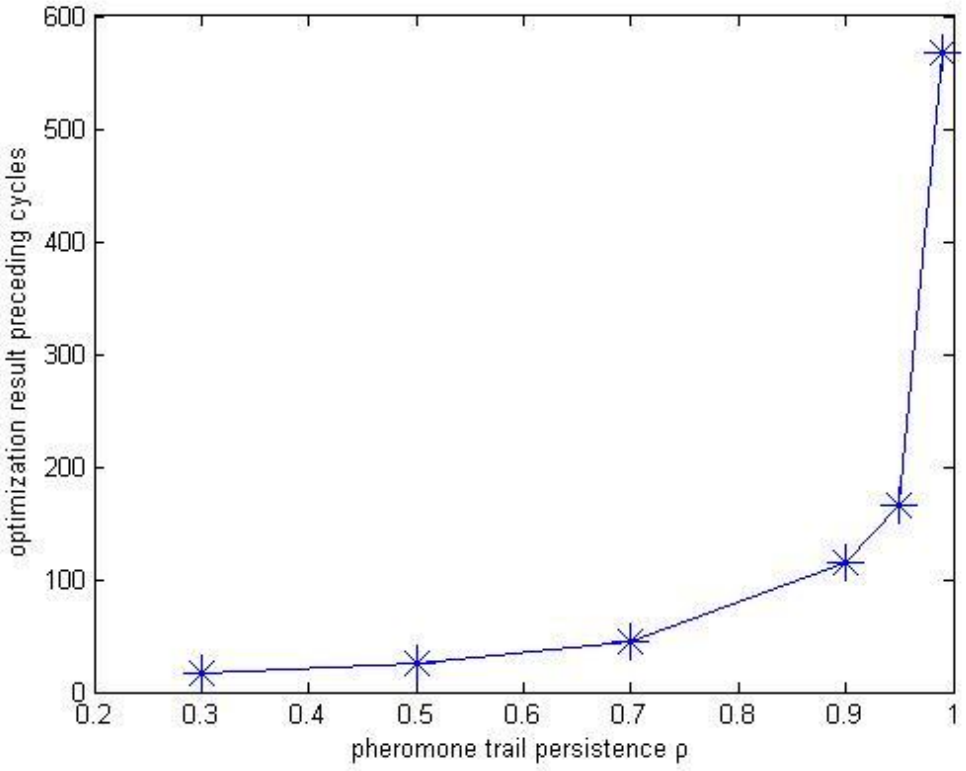

Fig. 1 The relationship of the pheromone trail persistence ρ and the optimization result preceding cycles

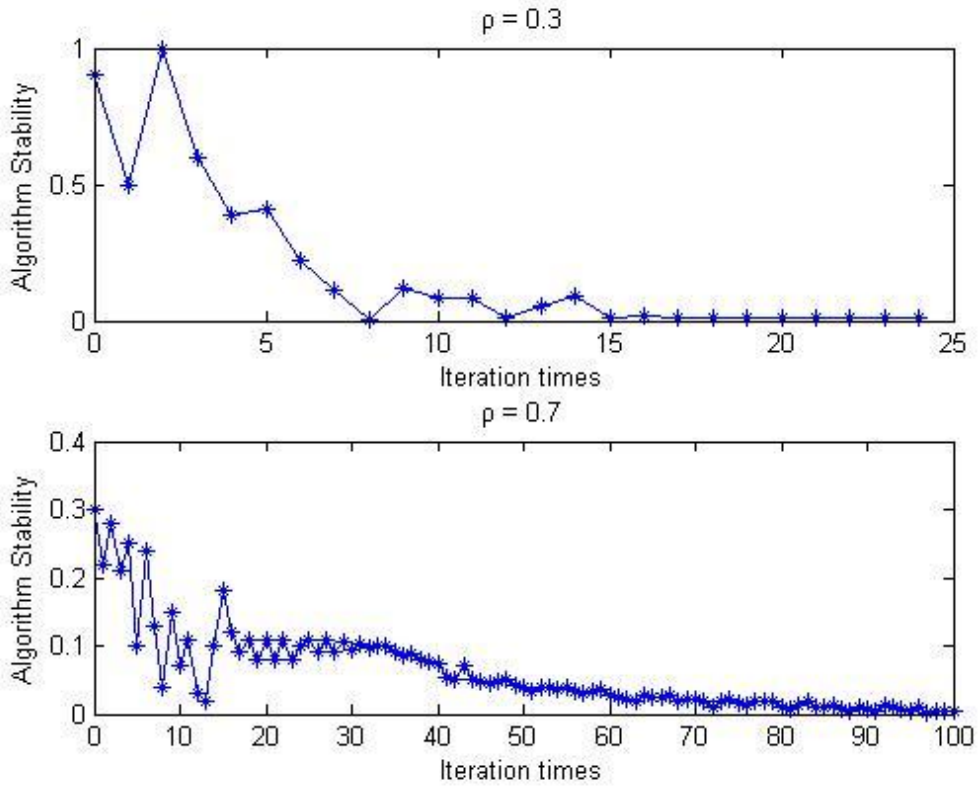

Fig. 2 The relationship of the pheromone trail persistence ρ
and the stability of the Ant Comoly Algorithm

2.2.6.1.3 Testing result analysis

From the simulation testing result, it is clear that, when all the other parameters are constants, the influence of the pheromone residual has a great influence on algorithm convergence rate. In particular, when $1-\rho$ is small, because the residue information on every edge is in dominate place, information positive feedback effect is relatively week and algorithm searching randomly stronger; consequently, algorithm convergence rate is low. On the other hand, when $1-\rho$ is relatively large and the information positive feedback effect dominates the evolution direction and algorithm search random characteristic is week, algorithm convergence rate increased. However, it has a relatively higher probability that the algorithm fall into a local optimal solution.

According to the discussion above, it is essential to consider both searching capability and the convergence rate when assigning a particular value for ρ. For this particular subsystem we trying to construct, it is predictable that the number of books need to search in typically smaller than ten. Form the figure 1 and figure 2, when $\rho = 0.5 \sim 0.7$, algorithm performance is stable and consistent, search capability and algorithm convergence rate are also relatively acceptable. So, according to the algorithm testing data and corresponding analysis above, when $\mathbf{1-\rho=0.3}$ $\mathbf{(\rho=0.7)}$

2.2.6.2 Population size

2.2.6.2.1 Parameter propriety analyze

Like most of evolution simulation algorithm, Ant Colony Algorithm is a randomly searching algorithm. Through the process the evolution of the whole population, which consists of a serious of candidate solutions (a subset of all possible solutions), the optimal solution has a high probability to be found. The evolution process not only needs the self-adapt capability of each individual but also call for cooperation of the whole population. It is important to notice that, the information exchange between individual plays a very important role in the complex but ordered behavior of ant population.

For CTSP, the ant path in a iteration can be viewed as a candidate solution, m ants path in one iteration contribute to a candidate solution set. Obviously, the bigger the candidate solution set the higher global searching capability, so is the algorithm stability. However, when the ant population size increased, it will make the pheromone on most ever searched path has less difference. The positive information feedback becomes weak. Although the searching is more randomly, the convergence rate decreased. On the contrail, if the candidate solution set becomes smaller, especially when the input problem scale is large, it will makes the pheromone on the path that have never been searched decrease approximately to zero. Algorithm random searching capability becomes weak even though the convergence rate increased. It will also weak the algorithm global searching capability and make it converge too early.

2.2.6.2.2 Testing result representation

In this particular test, ant population size is set as independent variable and the other algorithm parameters are set as constant. In specific:

- Pheromone intensity **Q = 1**
- Information heuristic factor $\boldsymbol{\alpha = 0.9}$
- expectation heuristic factor $\boldsymbol{\beta = 1.0}$
- iteration terminate condition: average candidate solution cost difference between two generations is less than **0.001**
- pheromone trail persistence $\boldsymbol{\rho = 0.7}$
- ant population size $\mathbf{m} \in \{2, 3, 4, 5, 6, 7\}$

The testing result is represented as below:

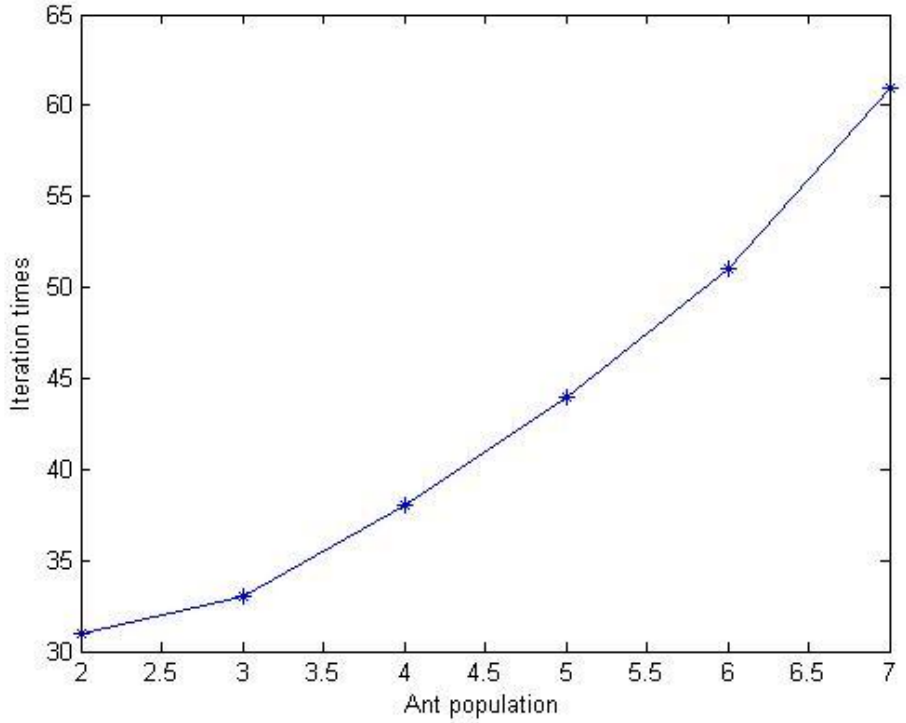

Fig. 3 The relationship of ant population size m and the optimization result preceding cycles

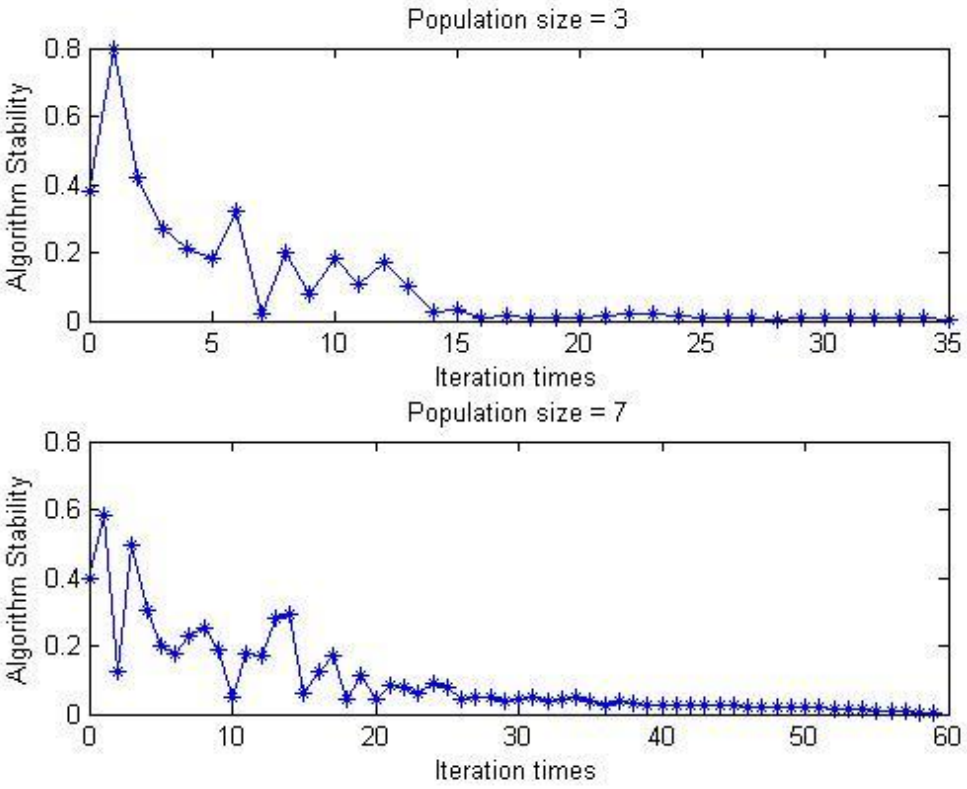

Fig. 4 The relationship of ant population size m and the stability of Ant Colony Algorithm

2.2.6.2.3 Testing result analysis

From the testing result, it is clear that the ant population size has an approximately positive linear influence on the iteration times (convergence rate). When the ant population size is overlarge (for example approximate to population size), the algorithm convergence rate sacrificed even though the global searching capability and algorithm stability improved.
To choice an appropriate ant population size, global searching capability and convergence rate should also be considered. For different particular input problem scale, appropriate or compromise choice should be made. According to the testing result above, appropriate population size $\mathbf{n} = \mathbf{n} \sim {}^{\boldsymbol{n}}/_{\mathbf{2}}$ (**n is the problem sclae**) .

2.2.6.3 Heuristic parameter

2.2.6.3.1 Parameter propriety analyze

From the ant translation formula:

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}{\sum_{l \subset U}^{n'} (\tau_{il}^{\alpha} * \eta_{il}^{\beta})} & j \in U \\ 0 & j \notin U \end{cases}$$

$$U = V - tabu_k = \{1, 2, 3 \cdots n'\}$$

The Information heuristic factor α indicate the relevant importance of pheromone on each path when an ant trying to select a vertex to move. Expectation heuristic factor β indicate the relevant importance of heuristic information (information expectation η_{ij}) when an ant trying to select a vertex to move. Expectation heuristic factor β reflect the apriority in path searching and the influence strength of certainty factor. The higher its value is, the ant has a higher probability to choice a local shortest path. Consequently, the algorithm convergence rate increased however the random searching capability decrease and the probability that fall into a local optimal solution increased. On the other hand, heuristic factor α reflect the influence strength of random on path searching. The larger it value is, the ant will has a higher probability to choice path that had ever been searched and consequently algorithm random search capability becomes weak and it will easier to fall into local optimal solution. To guarantee global optimal searching capability, the precondition is to guarantee the random searching capability and on the other hand, the algorithm high convergence rate also calls for relatively higher certainty search. According to the discussion above, information heuristic factor α and expectation heuristic factor β has a high relativeness.

2.2.6.3.2 Testing result representation

In this particular test, information heuristic factor α and expectation heuristic factor β are set as independent variables and the other algorithm parameters are set as constant. In specific:

- ant population size $\mathbf{m = 4}$
- Pheromone intensity **Q = 1**
- iteration terminate condition: average candidate solution cost difference between two generations is less than **0.001**
- pheromone trail persistence $\boldsymbol{\rho = 0.7}$

The testing result is represented as below:

∂	β	Best Cost	Iteration Times
0.1	1.0	379.8219	194
0.1	0.5	337.1086	75
0.5	1.5	325.3601	9
1.0	1.5	325.3508	6
3.0	1.5	325.3493	5
5.0	1.5	325.3508	4
10.0	5.0	325.0034	2
10.0	10.0	325.0006	2

Table. 2 The influence of the Ant Colony Algorithm proprities for different aombition of the heruistic factor ∂ and β

2.2.6.3.3 Testing result analysis

From the testing result represented above, different combination values of information heuristic factor α and expectation heuristic factor β has a great influence on algorithm performance.

- **Bad search result** and algorithm **terminate too early** (both α and β are too large)

When information heuristic factor α is too large, pheromone information will lead the searching direction. That means, when ant is trying to select a vertex to move, it will completely depend on pheromone information τ_{ij} on path. At the same time, if the expectation heuristic factor is also relatively large, it will make the information positive feedback becomes very strong. Algorithm will definitely terminate too early. Local optimal solution has a very high probability to be generated when the input problem scale is large.

- **Bad search result** and algorithm **terminate too early** (both α and β are too small)

When the information heuristic factor α is too small, pheromone information τ_{ij} has weaker influence on ant path selection. That means, when ant is trying to select a vertex to move, it will completely depend on expectation information η_{ij}. At the same time, if the information expectation factor is also small, it will make the algorithm fall into infinitely and completely random searching iteration (that means the algorithm will never terminate). Under this condition, optimal solution is very hard to find.

- **Optimal Approximated search result** (appropriate α and β value)

For different input problem scale, combination of α and β are little bit different. For this particular (and some similar scale input problem) $\alpha = 0.5 \sim 5$ and $\beta = 1.5$. In the condition, algorithm can generate a good candidate (typically optimal) solution and algorithm also has a low iteration times.

2.2.6.4 Pheromone intensity

2.2.6.4.1 Parameter propriety analyze

In this particular Ant Colony Algorithm, total pheromone information Q left by an ant when it passes a particular edge. Normal understanding of Q is: the larger the Q is, the faster pheromone accumulated and the information positive feedback becomes stronger. This will help to increase the algorithm convergence rate. As the high coupling rate of each algorithm parameters, influence of total pheromone information Q is actually depending value of other parameters.

2.2.6.4.2 Testing result representation

In this particular test, total information Q is set as independent variable and the other algorithm parameters are set as constant. In specific:

- pheromone trail persistence $\boldsymbol{\rho = 0.7}$
- ant population size $\mathbf{m = 4}$
- Pheromone intensity $\mathbf{Q} \in \{\mathbf{1}, \mathbf{10}, \mathbf{100}\}$
- Information heuristic factor $\boldsymbol{\alpha = 0.9}$
- expectation heuristic factor $\boldsymbol{\beta = 1.0}$
- iteration terminate condition: average candidate solution cost difference between two generations is less than **0.001**

The testing result is represented as below:

Pop size m	Best Cost	Iteration Times
1	327.7546	32
10	327.7538	34
100	327.7542	31

Table. 3 The influence of the Ant Colony Algoirhtm proprties
for different quantity of trail Q laid by ants

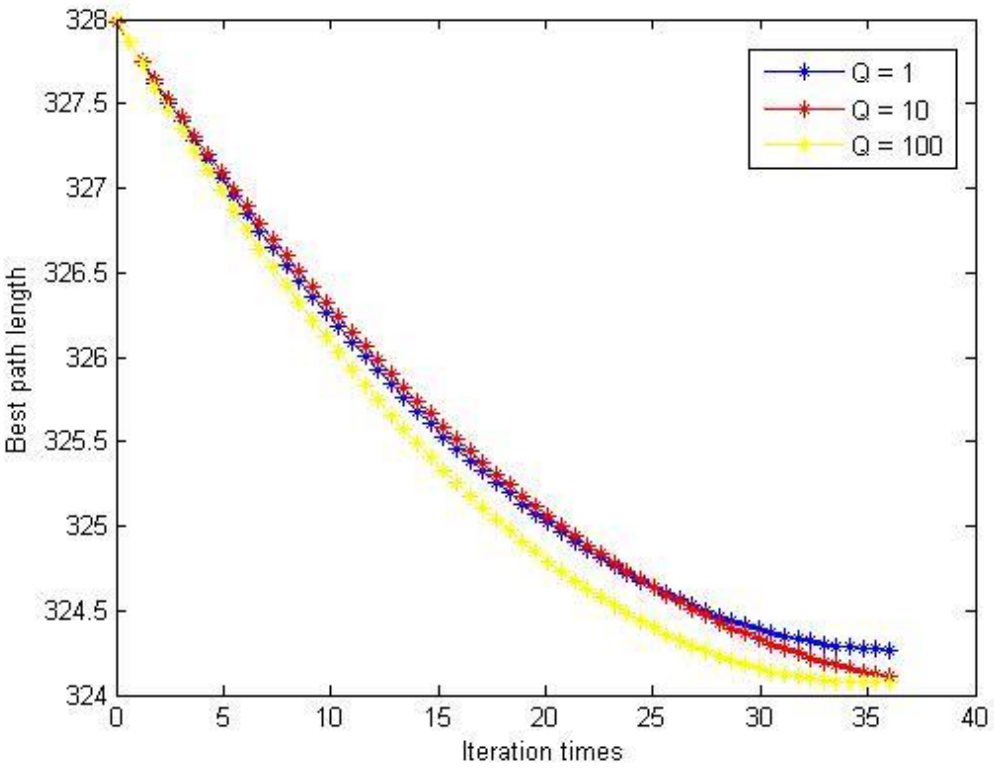

Fig. 5 The influence of the Ant Colony Algoirhtm
properties for the quantity of trail Q

2.2.6.4.3 Testing result analysis

For the testing result above, it is clear that the total information Q has on clear influence on algorithm performance. So, the value of this parameter needs not to be assigned too carefully.

2.2.7 Conclusion

Ant colony is a very powerful biological optimal searching algorithm. It base on the group intelligence heuristic behavior. Through the testing engineering process, the suggestion parameters value group for this particular application subsystem component is:

- $m = n \sim {}^{n}/_{2}$ (n is the input problem scale)
- $\partial = 1 \sim 5$
- $\beta = 1 \sim 5$
- $\rho = 0.7$
- $Q = 100$

2.3 Generic Algorithm

2.3.1 Varying parameters statement

During the implementation of the algorithm, a total of four varying parameters are involved in to help decision-making. They have abilities to determine the gene pool population size, generate times, crossover probability, and variation probability respectively. These four parameters will describe in details below.

2.3.1.1 Initial gene pool size

The population size (pop_size) of gene pool relates to the number of book to search. The initial population size is calculated by formula (n is the input problem scale):

$$\text{pop\_size} = \begin{cases} \text{n} * (\text{n} - 1) & n \geq 25 \\ 500 & n < 25 \end{cases}$$

2.3.1.2 2. Generation times

This parameter is varies for different ranges of books. Generation times T determined by the following formula (n is the input problem scale):

$$\text{T} = \begin{cases} 5 * n & n \geq 40 \\ 200 & n < 40 \end{cases}$$

2.3.1.3 Crossover rate

Crossover is not always occurred to avoid jump into local minimum solution, there is a function to calculate the probability to determine the occurrence. This probability relates to not only the fitness of the two parents genes but also all the candidate genes. The crossover probability (pc) can be described as below:

$$\text{K} = 1$$

$$Pc = \begin{cases} K & f' > f \\ K * \dfrac{f' - f_{fit}}{f - f_{fit}} & f' \leq f \end{cases}$$

$$f' = \left(f(s_1) + f(s_2)\right)\Big/_2$$

2.3.1.4 Variation rate

Similarly, variation is not performing all the time to guarantee the convergence; it also has an occurrence probability. And the probability relates to not only the particular gene but also all the

other candidate solutions. The crossover probability (Pm) can be described as:

$$Pm = \begin{cases} p * (f_{best} - f')/(f_{best} - f) & f' \geq f \\ p & f' < f \end{cases}$$

$$f' = f(\mathrm{s})$$

$$p = 0.5$$

f_{best} : the minimum $f(\mathrm{i})$ in the candidate solution set

f : the average $f(\mathrm{i})$ in the candidate solution set

2.3.2 Relationship prediction between varying parameters and algorithm performance

2.3.2.1 Initial gene pool size

The pop_size represents the size of the gene pool. To get a relatively stable result, it is necessary to provide a large size of gene pool to expand searching space. In the case of small size of gene pool, even though the convergence rate and run time is fast, it cannot guarantee to find the shortest path because it contains a small part of genes which may miss some good genes or fragments of good genes. So a speculative conclusion is that with the increasing of gene pool size the obtain solution is more stable and accuracy. But for the reason of running time and convergence rate, it should not too large. The precise value of the gene population will be confirmed after further analysis and amount of experiments.

2.3.2.2 Generation number

Generation time not only is a necessary condition to get a stable and accurate solution but also has ability to terminate the algorithm. Like the real nature a relatively stable and good gene must through a series of selection, crossover, and variation processes. This program also needs to iterate lots of times to make the convergence constrains to a specific tendency which closing to the optimal solution. The selection of generation number should be paid attention because if the generation number is too small, it is possible that this algorithm cannot find the optimal solution. If the generation number is too big, it is inefficiency because it needs more time to run the program. More serious phenomenon is that it breaks the original generated good solutions and produces some bad solution. So the generation number should satisfy two conditions which are finding the optimal solution and voiding the situation of breaking original good solutions. The precise value of the generation number will be confirmed after further analysis and amount of experiments. Meanwhile, the generation number can terminate the algorithm by executing generation number time of iterations.

2.3.2.3 Crossover probability

Crossover process has ability to produce new child solutions by cross gene fragments. Different from nature crossover process which is achieved by random, in this program, the crossover process is implemented by calling greedy method which always chooses the best fragments of the parents' genes in two directions consisting of two child genes. This method can promise cover more genes and it accelerates convergence rate. In this respect, it seems that it is a good choice to improve crossover probability. However, High crossover probability can produce the local optimal solution because it always choose the better solution and leave out gene fragments which is not a better choice in previous generations but can produce optimal solution in other generations. So to avoid to this situation, the crossover probability should be limited. Meanwhile, to accelerate convergence rate, the crossover probability should not too small. The precise value of the crossover probability will be confirmed after further analysis and amount of experiments.

2.3.2.4 Variation probability

Variation section has ability to generate new genes by reversing the original gene fragments which can expand searching space at the same time. In this aspect, it seems it better to improve the variation probability. However, if the Variation probability is too high; it is possible to break the original good solutions also because the variation is randomly. Moreover, it slows the convergence rate or even no convergence. In the case of the variation probability is too small; the search space is narrow which may cause local minimum solution. So the selection of variation needs to satisfy two conditions which are avoiding local optimal solution and guarantee convergence. The precise value of the crossover probability will be confirmed after further analysis and amount of experiments.

2.3.3 Analyze the effects of parameters(according to specific data)

Generic algorithm is a searching algorithm with generation and detection process. Colony is the object for crossover and variation operations. With the increasing of population size search space expands and the probability to obtain globally optimal solution amplifies. However, in the case of the population size is too large, it not only multiplies the complexity of the algorithm but also may drop the probability that high fitness genes been selected. Figure6 and figure7 show the generic algorithm performances with the changing of population size and generation number.

2.3.3.1 relationships between pop_size and algorithm performance

Pop size n	Best Cost	Average Cost	Convergence Time
100	342	468.24	259
200	314	460.32	506
300	303	466.97	833
400	302	459.745	1185
500	293	458.678	1471
600	293	468.098	2146
700	293	459.551	2688

Table. 4 The relationship between initial gene pool size and the algorithm cost and convergence time

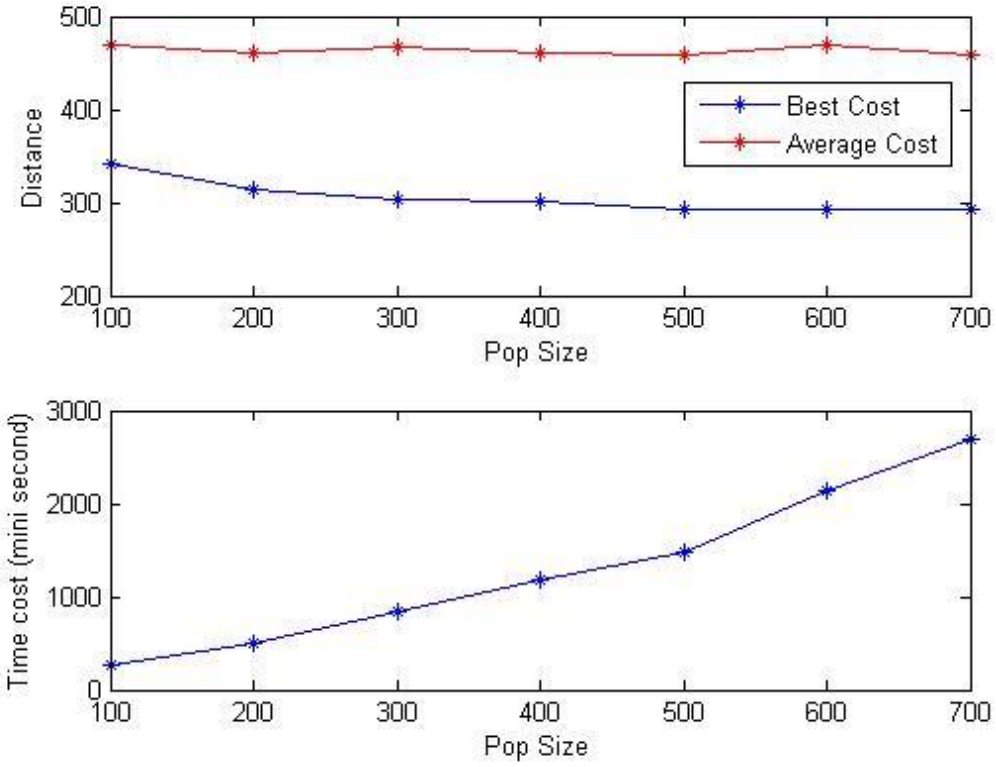

Fig. 6 The relationship between initial gene pool size and the algorithm cost and convergence time

From figure6 drawn by matlab according to data of table 4, it is apparent that with the increasing of population size, shortest distance and average distance decrease which means the solution is closing to the optimal solution.

An obvious phenomenon is when population size equals and large than 500 the shortest distance is fixed which is 293. And the average distance change range is not too much. This indicates that in this scale of this specific problem, 500 or more than 500 population sizes is required to obtain

optimal solution. However, running time represents up trend which implies that the program applies more time on executing generic algorithm with larger size of colony population. To guarantee the conditions that obtain the optimal solution and reduce the complexity of the problem which is shorten running time, value 500 is been considered to be the best choice.

2.3.3.2 relationships between generation number and algorithm performance

Iteration times	Best Cost	Average cost	Convergence time
100	320	458.524	366
200	310	465.898	697
300	303	461.792	937
400	293	463.662	1315
500	293	459.098	1621
600	304	458.718	1762
700	315	466.926	2055

Table. 5 Relationship between cost, convergence time and iteration times

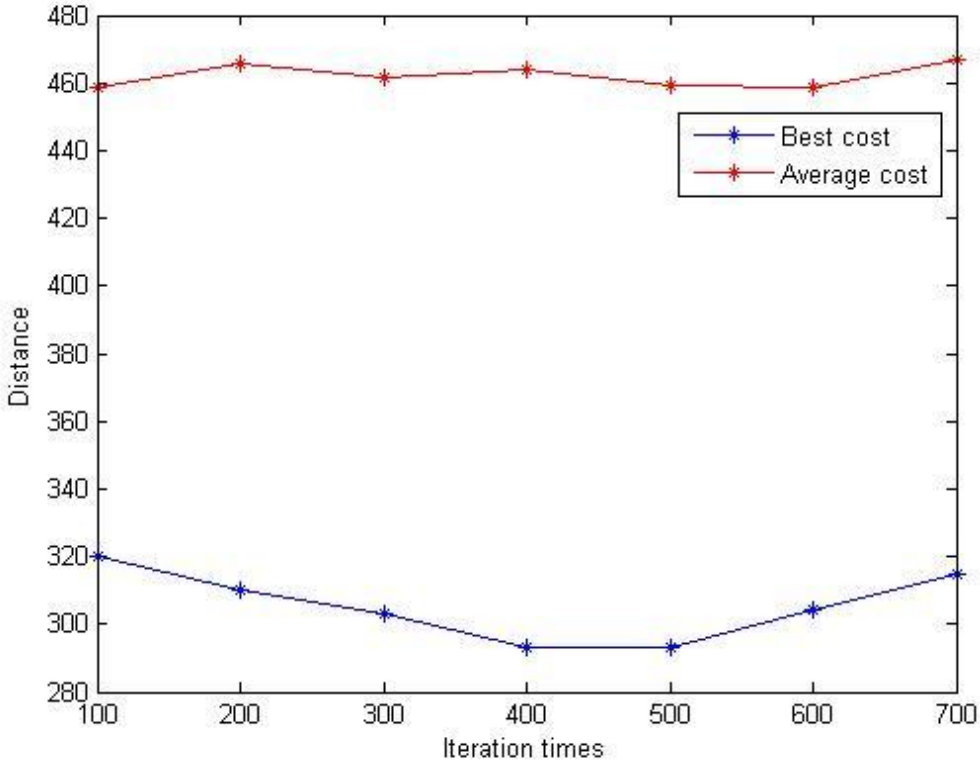

Fig. 7 Relationship between distance and iteration times

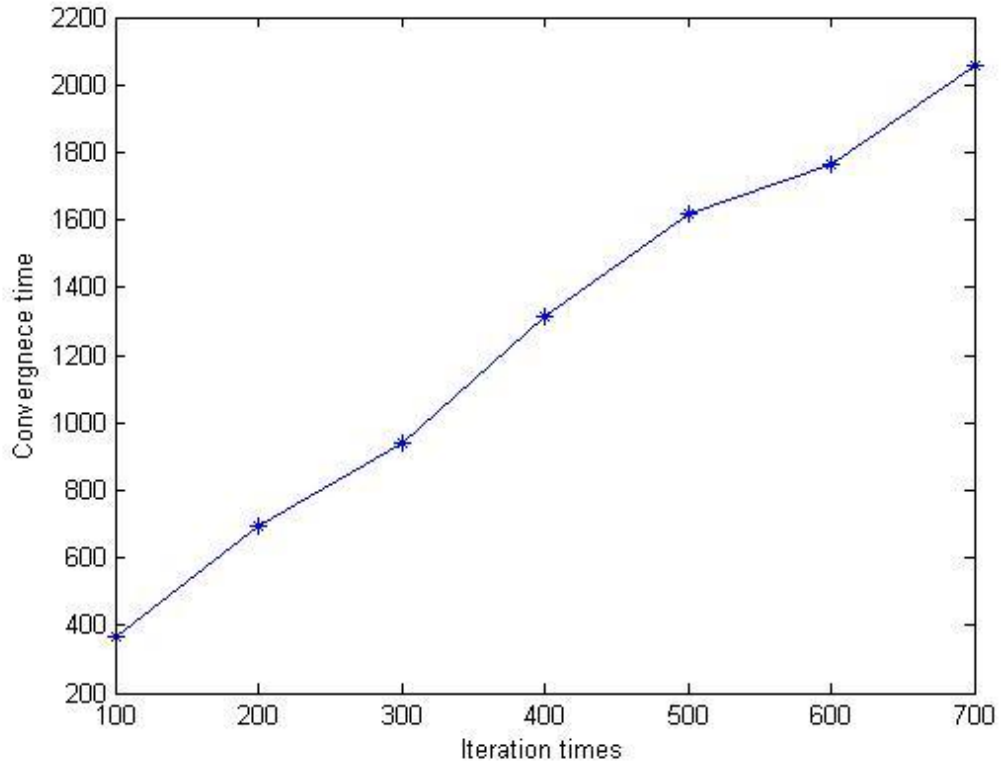

Fig. 8 Relationship between convergence time and iteration times

From figure7 drawn by matlab according to table 5 shows the relationships between generation number and algorithm performance. Different from population size, with the increasing of generation number, the shortest distance and average distance represent downtrend first and then uptrend. When generation number less than or equals to 500, the solution close to the best solution. While if generation number more than 500, the solution far away from the best solution. The reason for appearing this phenomenon is when generation numbers two large, it broke the original good solutions during variation section and produce bad genes. From figure8, The same as population size, with the increasing of generation number, run time aggrandizing because it requires more time to execute the additional generations. So to save time and obtain optimal solution, 500 is a fit value for this problem under the particular scale.

Crossover and Variation operations have abilities to guarantee variety of colony and help generic algorithm jump out of local minimum solution situation. Figure 3 and Figure4 shows the relationship between parameters contained in formula of calculating crossover probability and variation probability and distances including shortest distance and average distance. Both of the two operations will be illustrate bellow.

2.3.3.3 relationships between parameter k and algorithm performance

k	Best Cost	Average Cost
0.1	315	549.87
0.2	304	526.724
0.3	302	484.188
0.4	315	479.594
0.5	293	507.026
0.6	302	463.114
0.7	303	464.184
0.8	293	447.404
0.9	293	434.49
1	293	419.402

Table. 6 Relationships between k and distance

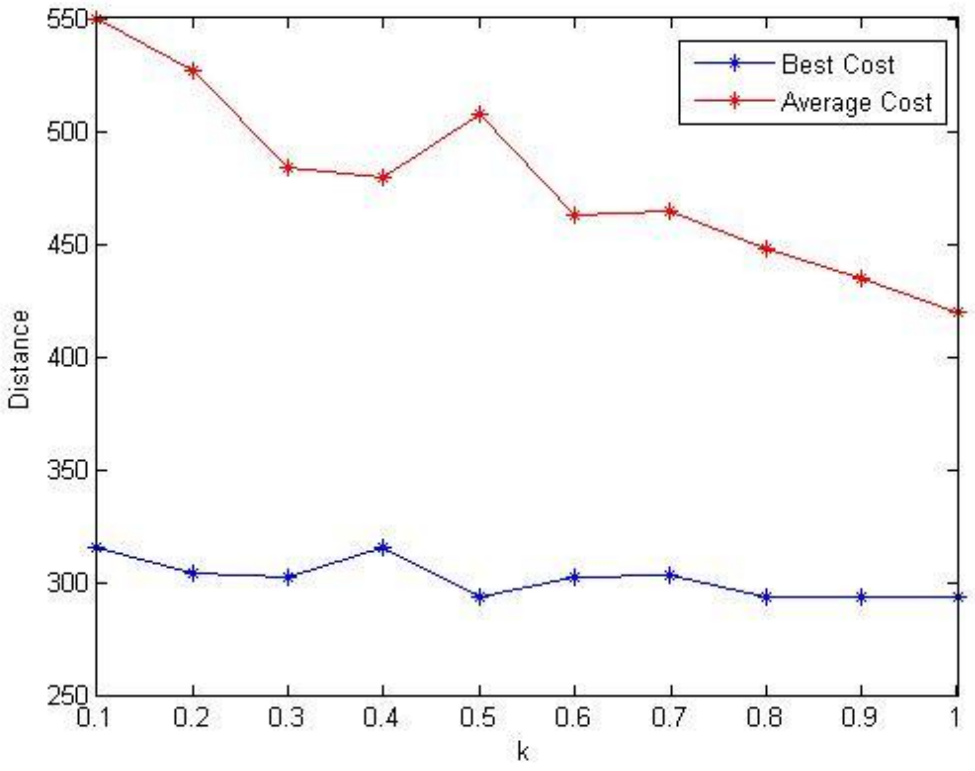

Fig. 9 relationships between k and distance

Parameter k in formula of calculating crossover probability has determined effect on crossover probability of the generic algorithm. And k represents proportional relationship to crossover probability. Figure 9 drawn by matlab according to data in table 6 indicates the relationships between parameter k and the algorithm performance. From Figure 9, it is clear that with the

increasing of value k, the shortest and average distances represent decreasing trend. Apparent this phenomenon because of the crossover method is greedy algorithm which always chooses the best solutions to generate child genes like described above. This means when doing crossover, it chooses the shorter distance of two parent genes as fragments of children genes. So with high crossover probability, the algorithm can find the shortest path quickly, this is fit for the table6 that when value k equals or more than 0.8, the shortest distance equals to the optimal solution. In other cases, the shortest distance larger than the optimal solution. In this sighs, it seems that high crossover probability is good. However, the bad effect of high crossover probability is that it is easy to lead to local optimal solution phenomenon because the searching space is narrow which may leave out better gene fragments. So to avoid this situation, a better resolver is selecting a lower crossover probability. In this program, it sets 0.2 as the crossover probability after a series of experiments. It may delay rum time but the overall algorithm is efficiency. This value can promise find the best solution without falling into local optimal solutions.

2.3.3.4 relationships between parameter p and algorithm performance

p	shortest distance	average distance
0.01	293	349.714
0.05	293	361.168
0.1	293	397.188
0.15	293	433.436
0.2	302	441.758
0.3	302	465.59
0.5	326	527.178
1	358	571.406

Table. 7 Relationships between p and distance

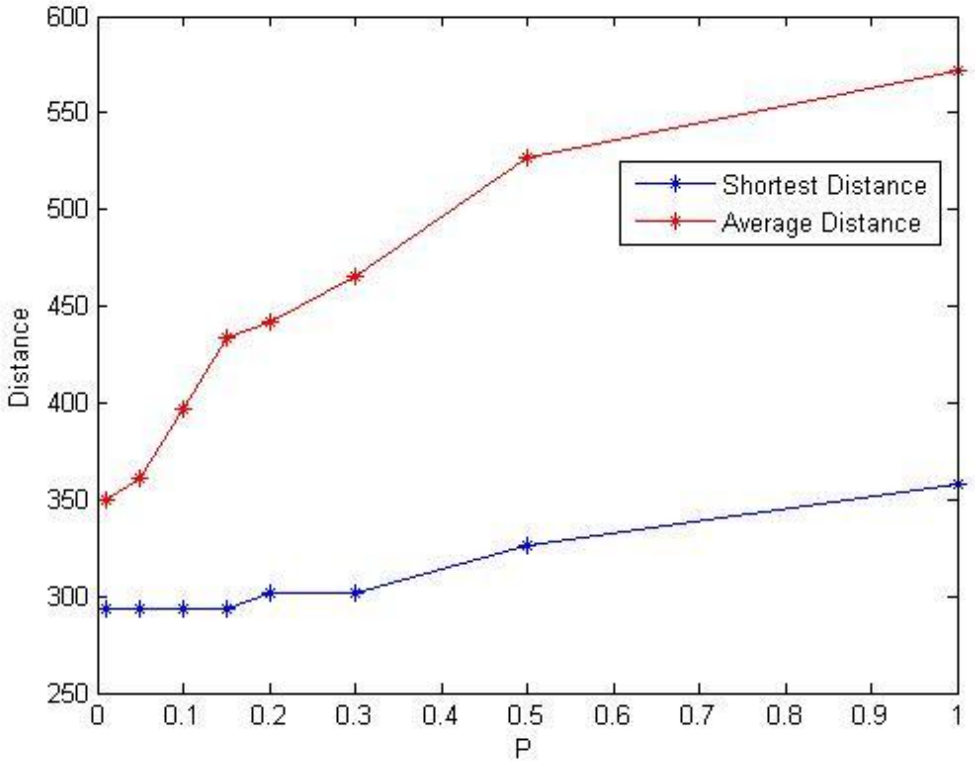

Fig. 10 Relationships between p and distance

Parameter p in formula of calculating variation probability has ability to determine the variation probability of the generic algorithm. And parameter p represents benefice relationship to variation probability according to table 7 shows. That is with the increasing of value p, the shortest distance and average distance reduce. Look at the table, it is clear that when value p large than 0.5 the shortest distance is too far away from the best distance. The reason of this phenomenon is when variation probability is too high, generic operation tends to random search which slows convergence rate or even no convergence. However, if variation probability is too low, it may fast the convergence rate but minimize searching space which may fall into local minimum solution. Observe the figure 10 drawn by matlab according to data in table 7; an obvious tendency is that the head part of the figure10 is straight line which equals to the best solution. Like table4 shows, when value p less than 0.15 it can find the best solution under the specific scale of problem. But to guarantee it can find the optimal solution in condition of wide searching space and not slow the convergence rate. In this program, 0.05 was chosen to act as the factor of formula of determines variation probability.

2.4 Algorithms comparison

In the best path finding subsystem, Ant Colony Algorithm and Generic Algorithm are executed simultaneously and both of them will generate a candidate solution. Then these two solutions are passed to path generator (another subsystem component) and both (sub)optimal solution cost will

be calculated. After that, the solution with less cost will be selected and provided to the web server vie the subsystem connecting interface.

On typical input problem scale (normal $2 \leq n \leq 15$), both of these two algorithms will generate an optimal solution. However, some other algorithm performance parameters may quite different.

In this section, three algorithm performance parameters are compared on different input problem scales.

2.4.1 Algorithm computing capability

Problem scale n	Computable ? (Ant Colony Algorithm)	Computable ? (Generic Algorithm)
10	Y	Y
20	Y	Y
30	Y	Y
40	Y	N
50	Y	N
60	Y	N
70	N	N

Table. 8 Compute capability between ant Colony Algorithm and Generic Algorithm

From table eight shown above, it is clear that both of these two algorithms have their own compute capability. However, they have quite different compute capability. In specific, ant colony algorithm can compute as large as n = 60 problem scale however, generic algorithm can only compute n = 30 problem scale. However, it is not the algorithm theoretical limitations that cause the computation limit. It is programing language (JAVA) that implement these algorithm cause these computation limitations.

More specifically, the reason that limits the ant colony algorithm compute capability is the accuracy that can be represented by **float** type variable. On the other hand, the reason that limits the generic algorithm compute capability is the representation capability of **int** type variable (it can only represent at most 1000000).

These limitations can all be overcome by changing the variable types. However, for this particular shortest path finding subsystem, the compute capability is enough. For the point of view of resource saving, these variable types will not be changed unless future requirement issued.

2.4.2 Algorithm accuracy

Problem scale n	Optimal solution cost (Ant Colony Algorithm)	Optimal solution cost (Generic Algorithm)
5	367.6455	367.6455
10	656.1252	603.7433
15	978.5757	843.7728
20	1407.1785	1122.5687
25	1679.7755	1392.5862
30	2078.4197	1619.7217

Table. 9 Accurancy comparasion between Ant Colony Algorithm and generic Algorithm

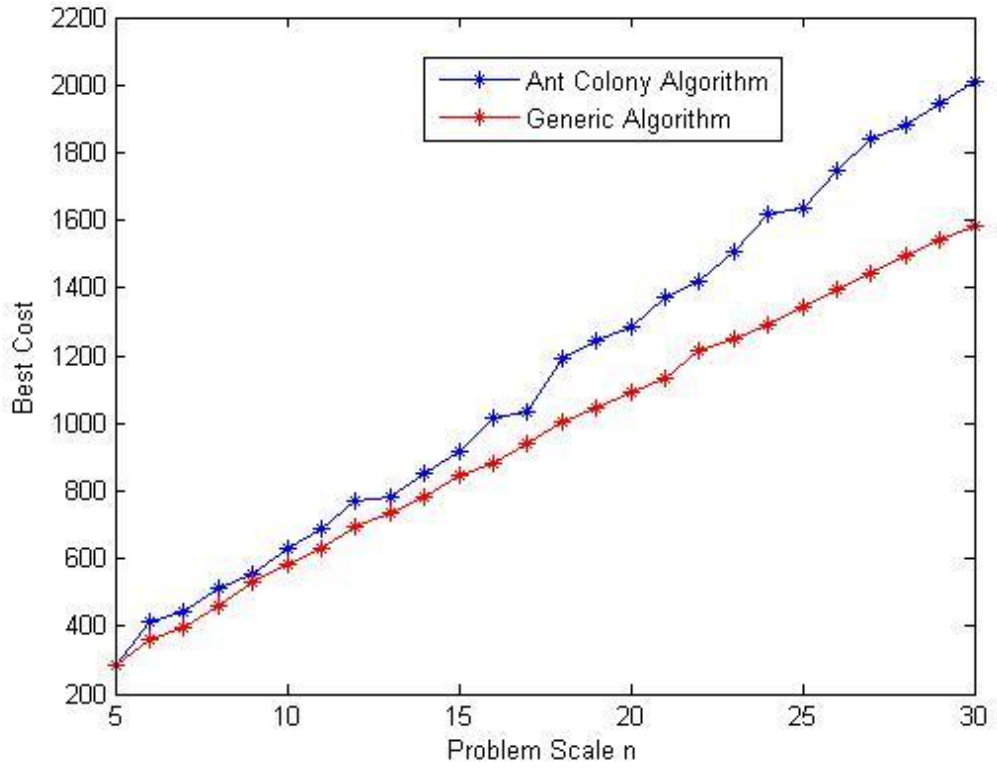

Fig. 11 Accurancy comparasion between Ant Colony Algorithm and generic Algorithm

From table nine and figure eleven, it can be seen that when the input problem scale is smaller than or equal to ten, both of these algorithm shows similar accuracy performance. However, the accuracy difference becomes more and more distinguishable with the linear increase of problem scale.

It is important to mention that the test data are all generated randomly and the result of integration shows that these two algorithm both show exactly the same performance (both solutions are optimal) when the input distance matrix is generated base on real data. However, from the standard accuracy performance point of view, generic algorithm shows a better accuracy

performance.

2.4.3 Algorithm convergence time

Problem scale n	Time cost (Ant Colony Algorithm)	Time cost (Generic Algorithm)
5	0	187
10	16	281
15	15	297
20	0	314
25	15	484
30	31	905

Table. 10 Convergence time cost comparasion Ant Colony Algorithm and Generic Algorithm

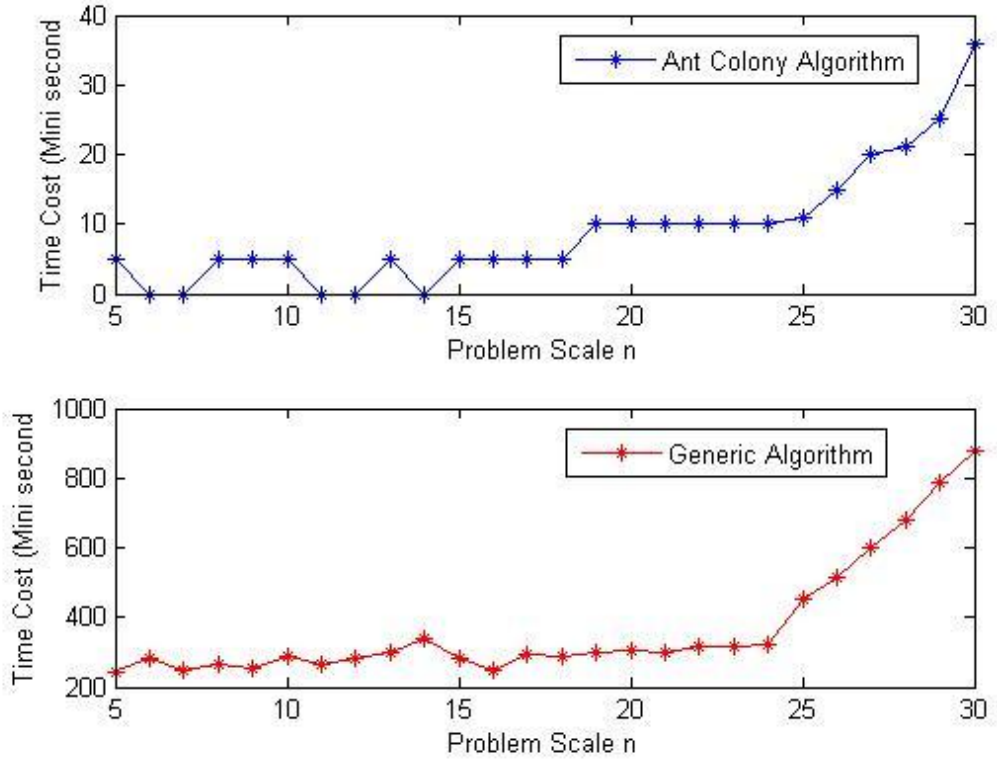

Fig. 12 Convergence time cost comparasion Ant Colony Algorithm and Generic Algorithm

From table ten and figure twelve, it can be seen that when the input problem scale is less than twenty-five both of these two algorithm shows a steady convergence time. However, when the problem scale is large than twenty-five and increase linear, the convergence also show a linear growth trends. From the comparison of the first figure and the second figure in figure seven, it is clear that the convergence time of generic algorithm is much bigger than that of ant colony algorithm. So, from the algorithm execution speed point of view, Ant Colony Algorithm shows a better convergence time performance.

2.5 Path Generator

The testing process of this subsystem component is much simple than that of testing the previous two algorithms because this component is respectively designed and implemented in simple algorithm logic and java implementation skills. Only one input data will be enough to test all the possible condition. However, from the view of its design assumption, this algorithm is not very robustness. Limitations of this algorithm will also discuss in this section.

2.5.1 Testing data

The testing data for this algorithm component is shown as below:

```
bi[0] = new bookInformation(11, 47, 6, 52, 11, 52, 6, 32, 11, 32);
bi[1] = new bookInformation(11, 37, 6, 52, 11, 52, 6, 32, 11, 32);
bi[2] = new bookInformation(17, 37, 17, 52, 22, 52, 17, 32, 22, 32);
bi[3] = new bookInformation(28, 37, 28, 52, 33, 52, 28, 32, 33, 32);
bi[4] = new bookInformation(22, 47, 17, 52, 22, 52, 17, 32, 22, 32);
bi[5] = new bookInformation(33, 73, 28, 78, 33, 78, 28, 58, 33, 58);
bi[6] = new bookInformation(50, 21, 50, 26, 55, 26, 50, 6, 55, 6);
bi[7] = new bookInformation(66, 11, 61, 26, 66, 26, 61, 6, 66, 6);
```

This testing data can test all the possible condition that the algorithm may came across.

2.5.2 Testing result

```
[12.2, 47.0]
[12.2, 37.0]
[15.8, 37.0]
[15.8, 30.8]
[26.8, 30.8]
[26.8, 37.0]
[23.2, 47.0]
[23.2, 53.2]
[34.2, 53.2]
[34.2, 73.0]
[34.2, 27.2]
[48.8, 27.2]
[48.8, 21.0]
[48.8, 4.8]
[67.2, 4.8]
[67.2, 11.0]
```

From the testing result, it can be seen that the generated path is exactly the right shortest path that can avoid obstacle and reach to every desired point. However, from the testing data result, it can hardly observe this conclusion. Only after representing this testing result visually that the correctness of this algorithm can be proved easily. In the integration testing phase, a visual representation of generated path will be provided.

2.5.3 Limitation analysis

It is known that there are infinite possible conditions when generate a path and avoid obstacle at the same time. For this shortest path finding subsystem is only implicated under the input of

library map, for the algorithm simplify consideration, only some simple basic assumption are considered. These assumptions are:

- All the bookshelves are line up and no crisscross between them.
- All the vertical space or horizontal space is all the same.

If the input library map data are not restricted to these two assumptions, the generated path may cross some bookshelves.

However, it is predictable that most library map may satisfy to these two assumptions and consequently this path generator component is still applicable for typical library map input.

3 Interface test

The interface testing process consists of two main phase:

- Component interface test
  - Ant Colony Algorithm component
  - Generic Algorithm component
  - Path generator component
- Subsystem interface test
  - Control algorithm component

All the components' interfaces are tested from the following respects:

- Interface correctness
- Interface safety
- Interface consistency
- Interface usability
- Information hiding
- Method encapsulation
- Component low coupling analysis

As most of testing details are either static program analysis or too complicate to describe and on the other hand, some of the interfaces testing activities are accomplished during the system component implementation. According to these reasons, the testing details are omitted to provide in this document.

In conclusion, the interface test result shows that all the respects mentioned above are precisely followed. On the other hand, the usability shown from the integrated subsystem testing is also approving this argument.

4 Result visualization

The following is the final representation of the whole shortest path finding subsystem. Below is a visualized integrating testing result: