Ant Colony Optimization (ACO) and a Variation of Bee Colony Optimization(BCO) in Solving TSP Problem, a Comparative Study

Muhammed Basheer Jasser*
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
43400 UPM Serdang, Selangor, Malaysia

Mohamad Sarmini*
Faculty of Electrical And Elecronic Engineering
University Of Aleppo

Rauf Yaseen
Faculty of Electrical And Elecronic Engineering
University Of Aleppo

ABSTRACT

Traveler sales man problem is known research problem which has a lot of industrial applications. A lot of algorithms has been proposed to solve TSP, some of Ant Colony Optimization (ACO) and Bee Colony Optimization (BCO) algorithms. BCO algorithm has variations and enhancements to improve the performance. In this paper, a experimental comparison study between the basic Ant Colony Optimization and enhanced Bee Colony Optimization algorithms is done. Both ACO and enhanced BCO have been implemented using MATLAB. The comparison study includes comparing the time consumed, solution quality and algorithmic complexity in order to prove the effictivness and efficiency of each. The experimental study showed that the basic ACO outperforms the enhanced BCO in the required consumed time to get the solution path, while enhanced BCO proved to provide better solution quality.

General Terms:

Swarm Intelligence, Evolutionary Computing

Keywords:

ACO, BCO, TSP

1. INTRODUCTION

Traveler sales man problem (TSP) [1] is very popular in the research community. TSP stands for finding the Hamiltonian path with the minimum cost. It has a lot of applications, this includes: enhancing transportations, finding the best route in network and enhancing routing algorithm. So many algorithms have been proposed to solve TSP problem, some of these algorithms are derived from animal behavior like ant colony and bee colony optimization algorithms (ACO, BCO). These algorithms are considered optimizing in which each system evolution cycle the solution is enhanced and improved. In ACO [4], Ant starts searching for food source randomly, when finding a source, it deposits chemical acid on its way returning to the nest. When new ant starts the searching process it follows the acid trails left by other ants to minimize the cost and enhance solution. Similarly for BCO algorithm[11] where bee searches for flower nectars, returns back to the hive and do waggle dancing in order to guide other bees in their journey. This type of behavior for both ant and bees is optimizing, means that better solution and minimum cost is guaranteed in the future. BCO has a lot of variations to enhance the performance and solution quality. One of these variations is integrating the original BCO with 2-opt local search heuristic so that better solutions can be obtained for TSP problem[14]. In this work, both original ACO and enhanced BCO algorithms are implemented using Matlab, so the objective of this paper is to analyze the algorithm complexity and solution quality of both in the form of comparative study. The structure of this paper is as follows: section2 will discuss about background of TSP, ACO and BCO, section3 explains the enhanced BCO, section4 shows the experimental results, Discussion of the experimental results and algorithmic complexity analysis are explained in section5 and finally section6 concludes the work.

2. BACKGROUND

Traveler Sales Man problem (TSP)[1] is known research problem, a lot of algorithms could be tested against this problem. In TSP problem, a sales man is supposed to visit a set of cities and return back to the starting point without repetition. For any algorithm employed to solve this problem, the aim here is to obtain the shortest journey length with the minimal algorithm and running time. Many Algorithms families have been proposed to solve TSP, some of called swarm intelligence algorithms, other called evolutionary computation. ACO[4] and BCO[11] algorithms are considered swarm intelligence algorithms. Evolutionary computation family includes algorithms like genetic algorithm[13].

2.1 Ant Colony Optimization Algorithim

ACO algorithm was suggested by Marco Dorigo in his Ph.D thesis at 1992. ACO algorithm [4] simulates the ant behavior in searching

for food source and depends on ant population where each ant represents on solution and communicates with other ants to enhance the obtained solution. An ant starts randomly searching for food source, if a food source has been found, the ant leaves Formic acid tail on its way to the nest which attracts nearby ants so that they follow this tail. When returning back to the nest these ants also leave acid on the same way so that the concentration increases. If more than path is found between the nest and the food source, the shorter distance is chosen. The Acid concentration increases on the shortest path so that it becomes more attractive to ants. Over time the acid concentration decreases on the longer distances. Figure 1 shows how the pheromone is updated during time where yellow, green and red represents the acid concentration from lowest to highest.

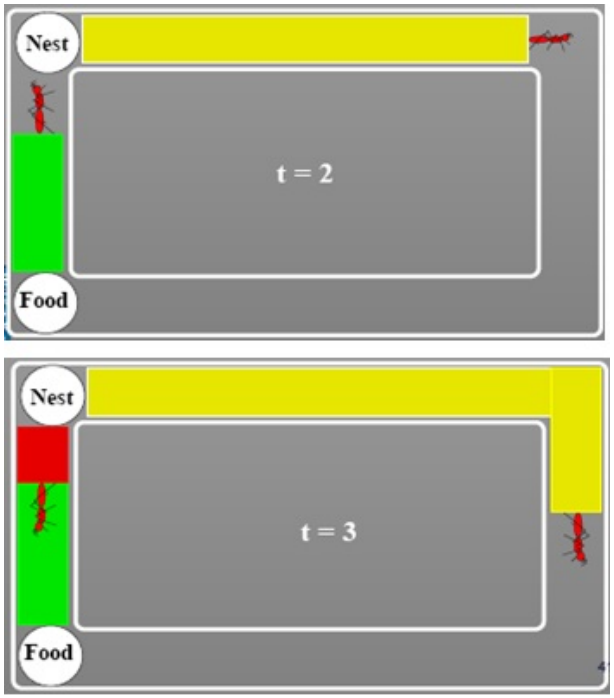

Fig. 1. ACO Pheromone Trail Update

The general pseudo code of ACO algorithm is shown in Figure 2. It starts with generating initial random solutions, then perform some actions in order to find better solutions and update the pheromone which means modify the solution space obtained for better future solutions enhancements. All these actions are performed iteratively for fixed times or until some solution quality level is obtained.

```
procedure ACO
    while(not_termination)
    generateSolutions()
    daemonActions()
    pheromoneUpdate()
    end while
end procedure
```

Fig. 2. ACO Algorithm Pseudo Code

2.2 Bee Colony Optimization Algorithm

Similar to ACO algorithm, BCO algorithm [11] is also considered of the family of swarm intelligence algorithm. BCO simulates the bees behavior in nature during the search process of flower nectar. Figure 3 shows the pseudo code of BCO algorithm where it is noticed that BCO requires a different number of parameters: n is the number of bees assigned to search for food (search for solution), m is the number of sites to visited by n bees, in TSP problem m represents the cities to be visited, and e which represents the best sites out of m sites. BCO algorithm starts by initializing the population with random solutions, then iteratively evaluates each solution fitness and obtains better solution from the neighborhood. BCO algorithm continues until no better solution can be obtained or considering other stopping criteria.

```
Procedure BCO
  Initialize population with random solutions
  Evaluate fitness of the population
  While (stopping criterion not met)
     // Forming new population
  Select sites for neighborhood and search
  Recruit bees for selected sites (more bees for
  best e sites) and evaluate fitness.
  Select the fittest bee from each path.
  Assign remaining bees to search randomly
  and evaluate their fitness
  End while
End procedure
```

Fig. 3. Basic BCO Algorithm Pseudo Code

2.3 Related Work

A lot of consideration has been given to solve TSP research problem. F.H Khan et al. [7] proposed a new technique to represent chromosomes in genetic algorithms using binary matrix and new fitness criteria in order to find the optimal solution for TSP. M. Mi et al. [9] proposes a new evolutionary algorithm similar to Genetic algorithm, but with different crossover technique in order to enhance the performance in finding the best solution. Z.Y. Quan. et al. [15]proposed a new swarm optimization algorithm called Discrete Glowworm swarm optimization algorithm. In their proposed algorithm, a new encoding and decoding schemas are introduced to represent the TSP problem characteristics. X. Geng et. al [5] proposes a local search algorithm based on greedy search and simulated annealing to solve the TSP, in order to obtain more accurate solutions, a combination of three mutation types are adopted with the basic simulated annealing algorithm. E. lizarraga et. al [8] proposes ant partition approach for Ant Colony Optimization (ACO) which uses different variations of ACO to evaluate the different partitions of ant set in the same iteration, this work is based on the divide and conquer idea with a stopping criterion if some solutions set is not performing well.

Several experimental studies have been carried out to compare the algorthims introduced to solve the TSP problem in order to evaluate the effectivness and efficiency of the different algorithms and approaches. W. Hui et. al [6] presents an experimental study to compate three different algorithms in solding TSP: basic genetic algorithm, Hopfield neural network and basic ant colony

algorithm. These algorithms are compared for time complexity, space complexity and the difficulty level of realization, the results showed that the time complexity of the basic genetic algorithm is the highest and the lowest time complexity is ant colony optimization, and the space complexity is the highest for the basic genetic algorithm and same for both Hopfield networks and ant colony algorithms. N.A.M. Zin et. al [16] performed an experimental study to compare exhaustive, heuristic and genetic algorithms where 25 cities are considered and the solution is written in prolog. The comparison has been made in the perspective of time taken to find the solutions and the optimal solutions obtained by each type of algorithm. It has been found that heuristic algorithms gives the solutions with very good time while genetic algorithm appeared to be promising in the perspective of the shortest path obtained among other algorithms. E. Osaba et. al [10] introduces a comparison study between a Tabu search based and memetic algorithm, according to this experimental study it has been found that Tabu search based algorithm gives the solution in better execution time while memetic algorithm provides a better solution quality in the term of path length, so if the running time is more important than the solution quality tabu search algorithm must be chosen otherwise if the solution quality is more important, memetic algorithm must be chosen. A. Chikhalikar et. al [3] introduces a comparative study between basic ACO and BCO algorithms, it was found that ACO requires more computational time compared to BCO, and ACO is more suitable for application where the search space is not so big while BCO is more suitable for larger search spaces. R. Sagayam et. al [12] performed a comparison study between ACO and BCO algorithms in solving spam host detection problem, it was found that ACO outperforms BCO in obtaining the optimal solution, that's why it was concluded in this study that ACO is better to be used in detecting spam host than BCO. M.K. Bedi et. al [2] discussed both ACO and BCO application in fault detection problem, the experimental study showed that BCO gives better solution quality than ACO in solving fault detection problem.

3. ENHANCED BCO ALGORITHM

According to the model proposed by L.P.Wong et. Al[14], A bee is allowed to explore complete tour path. Before leaving the hive, it observes other bees waggle dances in order to know which set of moves to follow in its expected journey, this path is called the preferred path which will guide the bee during its journey and represents one possible solution R. A solution R is a possible permutation of the cities to be visited.
During bee journey, a bee travel from one city to another until reaching the destination, this depends on decision made by the bee in each move which is based on heuristic rule. This rule consists of two factors arc fitness and heuristic distance. The factor arc fitness is computed for all the cities that can be visited by a bee at specific journey step, where higher fitness value is assigned to the arc which is part of the preferred path. On the other hand, bee tends to visit the next city which is closer to current one depending on heuristic distance factor.

BCO algorithm has been integrated with local search optimization technique to improve original algorithm performance. This is explained in the next following steps:

—1) Producea pseudorandom feasible solution, R.

—2) Performa transformation on R to produce R'.

—3) ReplaceR with R' if R' is found to be better than R.

—4) Repeatstep 2 until no improvement is observed. At this stage, R is said to be locally optimal.

—5) Repeatstep 1 to step 3 until a pre-defined computation time is exceeded or when a satisfactory result is gained.

R represents permutation of arcs connecting all cities in TSP problem. Applying previous steps guarantees that R will locally optimal means that a better solution can be obtained earlier than original BCO when using local search technique. The enhanced BCO model pseudo code is shown in Figure 4 where first, initial population is generated with equipped random preferred paths. This is because when the algorithm starts, no bee has ever did journey means that no waggle dance to be observed by other bees. The algorithm will be executed for a fixed number of times depending on some criteria which may be: no solution improving or limited time of iterations. This model has been implemented using MATLAB for further analysis in this experimantal study.

```
Procedure BCO
  Initialize population( )
  While stop criteria is not fulfilled do
    while all bees have not built a complete path do
      Observe_Dance( )
      Forage_ByTransRule( )
      Perform_2-Opt()
      Perform_Waggle_Dance( )
    End while
  End while
End procedure
```

Fig. 4. Enhanced BCO Algorithm Pseudo Code

4. EXPERIMENTAL RESULTS

Both ACO and enhanced BCO algorithms have been implemented using array data structure where the journey paths, set of moves, pheromone trails and waggle dances are represented using two dimensional arrays. Matlab is used to implement these algorithms since its main data structure used is array. For each model, diagrams are constructed which represents the solution of each iteration, best journey path and best time. These experiments has been carried out on an Intel Core i3- 2350M laptop with 2.30 GHz and a RAM of 4GB DDR3. Figure 5 shows an example of one iteration solution where it is represented by nodes (cities) and arc (edges).

To carry out the experiments, an array of 30 rows and 2 columns was used which represents 30 cities including its Coordinates X and Y. Depending on this array the distances array is generated and calculated which has 30 rows and columns. The experiments were executed three times for both algorithms, this covers iteration 10,50 ,200 and 1000. In each execution, best length obtained and execution time were registered in order to serve the comparison procedure. Figure 6 shows the experiments results.

Figures 7, 8 show the best obtained solution for BCO and ACO reseptively for iteration number equal to 200 where it is noticed that Enhanced BCO gave better solution than ACO.

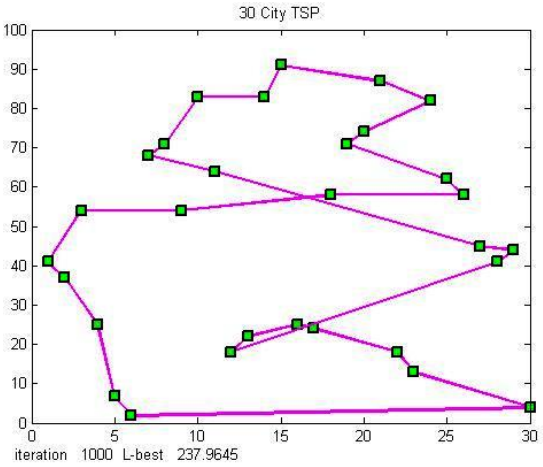

Fig. 5. An Example of Solution

No. City	No. Iteration	Best Length		Execution Time/Sec	
		BCO	ACO	BCO	ACO
30	10	**237.64**	**251.72**	**7.94**	**3.21**
	50	**237**	**247.64**	**20.61**	**14.91**
	200	**239.15**	**247.64**	**65.97**	**60.92**
	1000	**237.96**	**247.63**	**307.95**	**298**

Fig. 6. Experimental Results

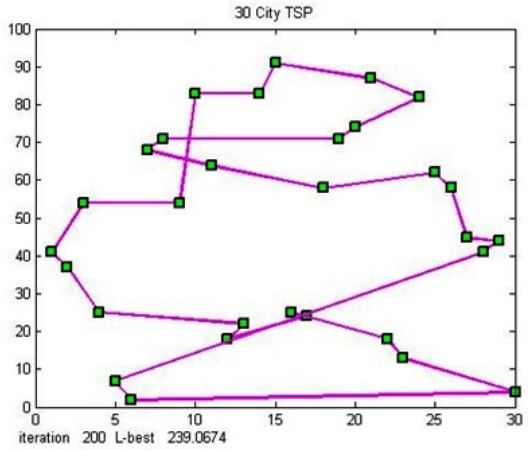

Fig. 7. EnhancedBCO Solution

Figure 9, 10 shows the graph of best obtained journey path length for each iteration for both EBCO and ACO respectively.

5. DISCUSSION

It is noticed from Figure 6 that enhanced BCO produces better solution quality in the term of journey path length compared to ACO,

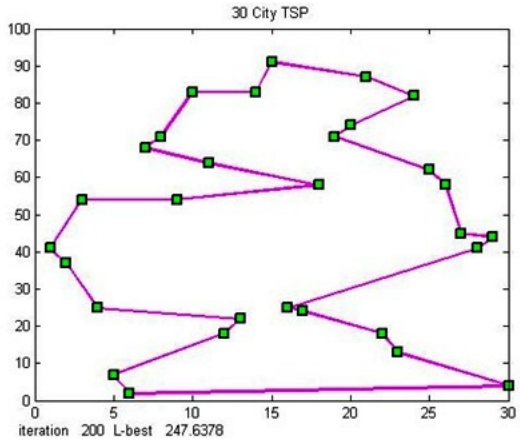

Fig. 8. ACO Solution

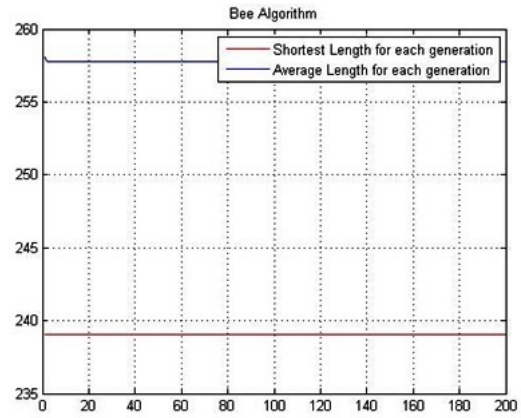

Fig. 9. EBCO Journey Paths Lengths Graph

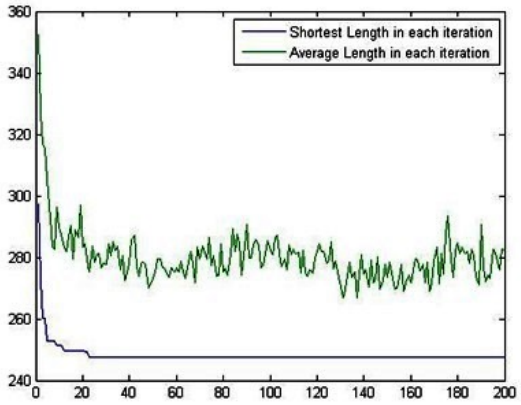

Fig. 10. ACO Journey Paths Lengths Graph

comparing to what has been found in the experimental study in [12] where ACO outperformed BCO in obtaining better solution quality in solving spam host detection problem, that's why in our experiment ACO is compared with the enhanced BCO where enhanced BCO outperforms ACO in finding the solution quality for

solving the TSP problem. ACO execution time is less than its rival means that it costs less time than BCO for small number of iterations, but as the number of iterations is increasing the execution time difference between ACO and enhanced BCO decreases, in other words, for big search space/ cities number and iterations, both algorithms are approximately equal in the computational time, while for smaller search space/ cities number and less interations number ACO gives better execution time than enhanced BCO. So it is possible to say that if it's wanted to get better solution quality from ACO, more iterations must be considered means that more computational time which may be the same or more than the computational time required from enhanced BCO to find a better qualified solution. This means if the objective is to get better execution time for small search space without giving too much care about the solution quality, then ACO must be chosen, otherwise for big search space, enhanced BCO gives approximately the same computational time and better solution quality so enhanced BCO must be chosen. Figures 7, 8 show the solution obtained by enhanced BCO and ACO respectively, this solution as obvious is represented by a set of nodes connected by a set of arcs. The solution best path length is calculated by summing arcs length. Journey paths lengths are calculated for each iteration and a graph representing the how the solution is improved is shown in Figures 9, 10. It is noticed in Figure 9, which represents the solution improvement rate for enhanced BCO, that the best solution is found at the early iterations, this is explained by the local optimal solution obtained by BCO model, this is considered as disadvantage of this model in a way of not being able to improve the solution in future system evolution. On the contrary, it is obvious in Figure 10 that ACO starts with a solution with high path length and improves with iteration, this actually reflects optimization concept in these models.

5.1 Algorithmic Complexity Analysis

In this section, big-theta for ACO and enhanced BCO will be analyzed depending on the number of execution for each algorithm step. To get the most accurate big-o analysis, matlab profiling function has been used to find lines of code hit calls and the running time, this will help in reflecting theoretical analysis in the real algorithm implementation using specific processor and memory.

The complexity in this situation mostly affected by three main factors which are: The number of bees or ants m, the number of cities n and the number of iterations nc. Considering m equals n, two factors remain in the analysis: m and nc. Since the implementation of ACO and EBCO depends on the iteration number in improving the solution, the number of execution will be the same for each experiment, in other words, there is no need to consider the big-omega algorithmic. Some optimization algorithms implementation depends on stopping criteria when the solution obtained is better than a specific limit, while in this case, a number of iteration nc is tested for each experiment no matter if the best solution is obtained or not.

The interest here is the code part of the algorithm implementation, means that the complexity of other functions calls like open GUIs or functions that do the same functionality with the same calls number for both ACO and EBCO is not considered, because it wont affect the comparison analysis and its complexity can be accumulated later.

In ACO algorithm, analysis of the main while iteration loop is analyzed considering m equals to 30 and NCmax equals to 10. Table 1 shows the ACO statements call hit of the main algorithm code. It is noticed from the table that the execution in most of its parts is straight forward, means that the statement has a fixed time to be executed . In other words, there is no big-omega algorithmic complexity to be considered.
Discarding the small complexity parts and considering only the major parts results in concluding that big-o for ACO is O(NCmax*m^3). Similarly for enhanced BCO, Tables 2 3 shows EBCO statements call hit of the main algorithm code.
Similar to ACO, the major part of complexity is only considered, this results the complexity of BCO is O(NCmax*m ^ 3).

6. CONCLUSION

In this work, two algorithms have been compared from three different viewpoints: the solution quality, algorithm complexity and running time. Its meant by solution quality the algorithm ability to obtain the best solution, in TSP problem it is represented by the shortest route. It has been found that the enhanced bee algorithm produces better solution especially at the early iterations. On the other hand it is found also that ACO algorithm costs less time compared to enhanced BCO when the search space and iteration numbers are small, but when the search space and iteraations number increase, the computational time for both ACO and enhanced BCO becomes aproximately similar. Regarding the algorithm complexity it is found that both algorithms have the same O(n) function. This result was obtained by theoretical analysis of each line of code and compare it with the results obtained from Matlab. In both algorithms, the array data structure has been used in the representation of individuals journey. The experiments were carried out for small number of cities, in this work 30 cities, in future greater number of cities and bigger search space may be tested and considered for deeper analysis.

7. ACKNOWLEDGEMENT

This work was funded by Malaysian Technical Cooperation Program (MTCP) fund from the malaysian government, reference number KPT.B.600-5/3 JILID 3. Corresponding authors: Muhammed Basheer Jasser, Mohamad Sarmini .

8. REFERENCES

[1] David L Applegate. *The traveling salesman problem: a computational study*. Princeton University Press, 2006.

[2] Mandeep Kaur Bedi and Sheena Singh. Fault detection techniques prioritization using bee colony optimization and then comparison with ant colony optimization. *International Journal of Computer Applications*, 69(17), 2013.

[3] Aditi Chikhalikar and Avanti Darade. Swarm intelligence techniques: Comparative study of aco and bco. *self*, 4:5, 1995.

[4] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1):53–66, 1997.

[5] Xiutang Geng, Zhihua Chen, Wei Yang, Deqian Shi, and Kai Zhao. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. *Applied Soft Computing*, 11(4):3680–3689, 2011.

[6] Wang Hui. Comparison of several intelligent algorithms for solving tsp problem in industrial engineering. *Systems Engineering Procedia*, 4:226–235, 2012.

Table 1. ACO Algorithmic Analysis

Statement	Calls Hit	Value	Time/Sec
while NC¡=NCmax	NCmax+1	11	
Randpos=[]	NCmax	10	
for i=1:(ceil(m/CityNum))	NCmax	10	
Randpos=[Randpos,randperm(CityNum)]	NCmax	10	
end	NCmax	10	0.01
Tabu(:,1)=(Randpos(1,1:m))'	NCmax	10	
for j=2:CityNum (The comparison operation)	NCmax * m	300	
for i=1:m (The comparison operation)	NCmax *(m-1)*(m+1)	8990	
visited=Tabu(i,1:(j-1));	NCmax *(m-1)*m	8700	0.04
J=setdiff(1:CityNum,visited)	NCmax *(m-1)*m	8700	2.7
P=J;	NCmax *(m-1)*m	8700	0.01
for k=1:length(J)(The comparison operation)	NCmax*(m-1)*m*(m+1)/2	134850	0.01
P(k)=(Tau(visited(end),J(k))^ Alpha)*(Eta(visited(end),J(k))^ Beta)	NCmax*(m-1)*m*(m)/2	130500	0.08
end	NCmax*(m-1)*m*(m)/2	130500	0.03
P=P/(sum(P))	NCmax *(m-1)*m	8700	0.12
Pcum=cumsum(P)	NCmax *(m-1)*m	8700	0.02
Select=find(Pcum¿=rand)	NCmax *(m-1)*m	8700	0.05
tovisit=J(Select(1))	NCmax *(m-1)*m	8700	0.01
Tabu(i,j)=tovisit	NCmax *(m-1)*m	8700	0.01
end	NCmax *(m-1)*m	8700	0.01
end	NCmax * (m-1)	290	0.01
if NC¿=2	NCmax	10	
Tabu(1,:)=Rbest(NC-1,:)	NCmax-1	9	
end	NCmax-1	9	
L=zeros(m,1)	NCmax	10	
for i=1:m (The comparison operation)	NCmax * (m+1)	310	
R=Tabu(i,:)	NCmax * m	300	
L(i)=CalDist(D,R);	NCmax * m	300	0.01
end	NCmax*m	300	0.01
Lbest(NC)=min(L)	NCmax	10	
pos=find(L==Lbest(NC))	NCmax	10	
Rbest(NC,:)=Tabu(pos(1),:)	NCmax	10	
Lave(NC)=mean(L)	NCmax	10	0.01
drawTSP(C,Rbest(NC,:),Lbest(NC),NC,0)	NCmax	10	1.17
NC=NC+1	NCmax	10	
DeltaTau=zeros(CityNum,CityNum)	NCmax	10	
for i=1:m	NCmax * (m+1)	310	
for j=1:(CityNum-1)(The comparison operation)	NCmax * m * m	9000	0.01
DeltaTau(Tabu(i,j),Tabu(i,j+1))=DeltaTau(Tabu(i,j),Tabu(i,j+1))+Q/L(i)	NCmax * m * (m-1)	8700	0.01
end	NCmax * m * (m-1)	8700	0.01
DeltaTau(Tabu(i,CityNum),Tabu(i,1))=DeltaTau(Tabu(i,CityNum),Tabu(i,1))+Q/L(i)	NCmax * m	300	0.01
end	NCmax * m	300	0.01
Tau=(1-Rho).*Tau+DeltaTau	NCmax	10	
Tabu=zeros(m,CityNum)	NCmax	10	
tauji(NC)=Tau(1,2)	NCmax	10	
end	NCmax	10	0.01
Pos=find(Lbest==min(Lbest))		1	
ShortestRoute=Rbest(Pos(1),:)		1	
ShortestLength=Lbest(Pos(1))		1	
Lbestant=Lbest		1	

[7] Fozia Hanif Khan, Nasiruddin Khan, Syed Inayatullah, and Shaikh Tajuddin Nizami. Solving tsp problem by using genetic algorithm. *International Journal of Basic & Applied Sciences*, 9(10), 2009.

[8] Evelia Lizárraga, Oscar Castillo, and José Soria. A method to solve the traveling salesman problem using ant colony optimization variants with ant set partitioning. In *Recent Advances on Hybrid Intelligent Systems*, pages 237–246. Springer, 2013.

[9] Mei Mi, Xue Huifeng, Zhong Ming, and Gu Yu. An improved differential evolution algorithm for tsp problem. In *Intelligent Computation Technology and Automation (ICICTA), 2010 In-*

Table 2. Enhanced BCO Algorithmic Analysis

Statement	Calls Hit	Value	Time/Sec
while(i¡=m)(The comparison operation)	m+1	31	
rand=randint(1,1,[1,n]);	The best case m / average 4m	30/120	0.07
found=ismember(rand,prefTabu(:,1));	The best case m / average 4m	30/120	0.04
if found == 0	The best case m / average 4m	30/120	0.01
prefTabu(i,1)= rand	m	30	
i=i+1;	m	30	
end;	m	30	
end	The best case m / average 4m	30/120	0.01
for i=1:m	m+1	31	
j=2	m	30	
while(j¡=n)(The comparison operation)	m*(m+1)	930	
rand=randint(1,1,[1,n])		3617	2.53
found=ismember(rand,prefTabu(i,:))		3617	0.51
if found == 0		3617	0.01
prefTabu(i,j)= rand		870	0.01
j=j+1		870	
end		870	0.01
end		3617	0.01
end	m	30	
Tabu(:,1)=prefTabu(:,1)		1	
while NC¡=NCmax (The comparison)	NCmax+1	11	
PFcolony=0;	NCmax	10	
PF=zeros(1,n)	NCmax	10	
LTrack=zeros(1,n)	NCmax	10	
LpTrack=zeros(1,n)	NCmax	10	
for i=1:m (The comparison)	NCmax * (m+1)	310	
Tau=zeros(n)	NCmax * m	300	
P=zeros(n)	NCmax * m	300	0.01
for j=2:n (The comparison)	NCmax * m * m	9000	
visited=Tabu(i,1:(j-1))	NCmax * m * (m-1)	8700	0.03
J=setdiff(1:n,visited)	NCmax * m * (m-1)	8700	2.76
if length(J)==1	NCmax * m * (m-1)	8700	0.02
Tau(Tabu(i,j-1),J(length(J)))=1	NCmax * m	300	0.01
P(Tabu(i,j-1),J(length(J)))=1	NCmax * m	300	
else	NCmax * m * (m-2)	8400	0.01
for K=1:length(J)(The comparison)	NCmax * m * (m-2)*(m+2)/2	134400	0.01
if J(K)==prefTabu(i,j)	NCmax * m * (m-2)*(m+1)/2	8009	0.17
Tau(Tabu(i,j-1),prefTabu(i,j))=lambda	NCmax * m * (m-2)*(m+1)/2	8009	0.01
else		122191	
Tau(Tabu(i,j-1),J(K))=(1 lambda)/(length(J)-1)		122191	0.09
end		122191	0.04
end	NCmax * m * (m-2)*(m+1)/2	130200	0.05
sum=0	NCmax * m * (m-2)	8400	0.01
for k2=1:length(J)(The comparison)	NCmax * m * (m-2)*(m+2)/2	134400	0.02
sum=sum+((Tau(Tabu(i,j-1),J(k2))^Alpha)*(Eta(Tabu(i,j-1),J(k2))^Beta))	NCmax * m * (m-2)*(m+1)/2	130200	0.21
end	NCmax * m * (m-2)*(m+1)/2	130200	0.19
for k2=1:length(J)(The comparison)	NCmax * m * (m-2)*(m+2)/2	134400	0.01
if (Tabu(i,j-1) = J(k2))	NCmax * m * (m-2)*(m+1)/2	130200	0.17
num=((Tau(Tabu(i,j-1),J(k2))^Alpha)*(Eta(Tabu(i,j-1),J(k2))^Beta))	NCmax * m * (m-2)*(m+1)/2	130200	0.29
P(Tabu(i,j-1),J(k2))=num/sum	NCmax * m * (m-2)*(m+1)/2	130200	0.11
end	NCmax * m * (m-2)*(m+1)/2	130200	0.02
end	NCmax * m * (m-2)*(m+1)/2	130200	0.03
end	NCmax * m * (m-2)*(m+1)/2	130200	0.01
[row,col]=find(P==max(P(Tabu(i,j-1),:)))	NCmax * m * (m-1)	8700	0.12
Tabu(i,j)=col(1)	NCmax * m * (m-1)	8700	0.01
end	8700	0.01	
for j=1:n-1 (The comparison)	NCmax * m * m	9000	0.02
LTrack(i)=LTrack(i)+D(Tabu(i,j),Tabu(i,j+1))	NCmax * m * (m-1)	8700	0.02
LpTrack(i)=LpTrack(i)+D(prefTabu(i,j),prefTabu(i,j+1))	NCmax * m * (m-1)	8700	0.01
end	NCmax * m * (m-1)	8700	0.01
if (LTrack(i) ¡ LpTrack(i))	NCmax * m	300	0.01
PF(i)=1/LTrack(i)	m	30	
PFcolony=PFcolony+(PF(i)/m)	m	30	
wagdur(i)=PF(i)/PFcolony;end	m	30	
if (PF(i)¡0.95*PFcolony)	NCmax * m	300	0.01
elseif (0.95*PFcolony¡=PF(i)¡0.975*PFcolony)	NCmax * m	300	
elseif (0.975*PFcolony¡=PF(i)¡0.99*PFcolony)	NCmax * m	300	
elseif (PF(i)¿0.99*PFcolony)	NCmax * m	300	
PFfollow(i)=0	m	30	
end	m	30	
end	NCmax * m	300	0.01

Table 3. Enhanced BCO Algorithmic Analysis

Statement	Calls Hit	Value	Time/Sec
[Lbest(NC,1),pos]=min(LTrack)	NCmax	10	0.01
Rbest(NC,:)=Tabu(pos,:)	NCmax	10	
Lave(NC,1)=mean(LTrack)	NCmax	10	0.01
drawTSP(C,Rbest(NC,:),Lbest(NC,1),NC,0)	NCmax	10	1.16
NC=NC+1	NCmax	10	
if NC¿=2	NCmax	10	
prefTabu=Tabu	NCmax	10	
end	NCmax	10	
mcc=NC/NCmax	NCmax	10	
end	NCmax	10	
[ShortestLength,Pos]=min(Lbest)		1	
ShortestRoute=Rbest(Pos,:)		1	
bee=Lbest		1	
Lbestbee=Lbest		1	

ternational Conference on, volume 1, pages 544–547. IEEE, 2010.

[10] Eneko Osaba and Fernando Díaz. Comparison of a memetic algorithm and a tabu search algorithm for the traveling salesman problem. In *Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on*, pages 131–136. IEEE, 2012.

[11] DT Pham, A Ghanbarzadeh, E Koc, S Otri, S Rahim, and M Zaidi. The bees algorithm-a novel tool for complex optimisation problems. In *Proceedings of the 2nd Virtual International Conference on Intelligent Production Machines and Systems (IPROMS 2006)*, pages 454–459, 2006.

[12] R Sagayam and Mrs K Akilandeswari. Comparison of ant colony and bee colony optimization for spam host detection.

[13] SN Sivanandam and SN Deepa. *Genetic Algorithm Optimization Problems*. Springer, 2008.

[14] Li-Pei Wong, Malcolm Yoke Hean Low, and Chin Soon Chong. Bee colony optimization with local search for traveling salesman problem. *International Journal on Artificial Intelligence Tools*, 19(03):305–334, 2010.

[15] Yong-Quan Zhou, Zheng-Xin Huang, and Hong-Xia Liu. Discrete glowworm swarm optimization algorithm for tsp problem. *Dianzi Xuebao(Acta Electronica Sinica)*, 40(6):1164–1170, 2012.

[16] Nur Ariffin Mohd Zin, Siti Norul Huda Sheikh Abdullah, Noor Faridatul Ainun Zainal, and Esmayuzi Ismail. A comparison of exhaustive, heuristic and genetic algorithm for travelling salesman problem in prolog. *International Journal on Advanced Science, Engineering and Information Technology*, 2(6):49–53, 2012.