Artificial Bee Colony Algorithm for TSP

Xinrao Li, *Student member, SUSTECH*, Weijie Xu. *Student member, SUSTECH*

***Abstract*—Traveling Salesman Problem is an important optimization problem of many fields and it is about finding a Hamiltonian path with minimum cost. To solve this problem, many researchers have proposed different approaches including metaheuristic methods,such as GA,PSO and so on. Artificial Bee Colony algorithm is a well-known swarm based optimization technique. In this project we choose to use ABC to solve Traveling Salesman Problem. Simulation results show that this Artificial Bee Colony algorithm can be used for combinatorial optimization problems.**

***Index Terms*—artificial bee colony,optimization,traveling salesman problem**

I. Introduction

Traveling Salesman Problem (TSP) is an NP-hard combinatorial optimization problem. In the literature a lot of metaheuristic algorithms have been applied to this problem for obtaining better results in acceptable computational times.

Artificial Bee Colony (ABC) is one of the most recently defined algorithms by Dervis Karaboga in 2005, motivated by the intelligent behavior of honey bees[1]. It is as simple as Particle Swarm Optimization (PSO) and Differential Evolution (DE) algorithms, and uses only common control parameters such as colony size and maximum cycle number. ABC as an optimization tool, provides a population-based search procedure in which individuals called foods positions are modified by the artificial bees with time and the bee's aim is to discover the places of food sources with high nectar amount and finally the one with the highest nectar. In ABC system, artificial bees fly around in a multidimensional search space and some (employed and onlooker bees) choose food sources depending on the experience of themselves and their nest mates, and adjust their positions. Some (scouts) fly and choose the food sources randomly without using experience. If the nectar amount of a new source is higher than that of the previous one in their memory, they memorize the new position and forget the previous one. Thus, ABC system combines local search methods, carried out by employed and onlooker bees, with global search methods, managed by onlookers and scouts, attempting to balance exploration and exploitation process.

II. Traverling Salesman Problem

The basic principle of TSP is that, the salesman starts from a point to his tour and he returns to this starting point as trying to obtain a closed tour with minimum cost[2]. When he takes his tour he must visit every point once. The cost of his tour directly depends on the tour length.

In this study, the distance between the city i and city (i+1) $d(T[i], T[i+1])$ is calculated as Euclidean distance by (1).

$$d(T[i], T[i+1]) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \quad (1)$$

Total tour length f can be expressed as follows (2)

$$f = \sum_{i=1}^{n-1} d(T[i], T[i+1]) + d(T[n], T[1]) \quad (2)$$

III. Artificial Bee Algorithm

ABC algorithm is a swarm based metaheuristic that simulates foraging behavior of honey bees. The artificial bee colony includes three kinds of bees considering the division of labor: employed bees, onlooker bees and scout bees. Each employed bee works on only one food source. An employed bee keeps a food source in her mind when she leaves from the hive and she shares the information about her food source with onlookers on dance area. Onlookers select a food source by watching the dances of the employed bees and try to improve this source. If a food source is abandoned, its employed bee becomes a scout to explore new food sources randomly. Basic steps of the ABC algorithm are given below:

1. Initialization. (Food source,population size)
2. Repeat:
   - Employed bees phase: for each employed bee
     - Produce a new solution v_i in the neighborhood of x_i via: $v_{ij} = x_{ij} + r(x_{ij} - x_{kj})$.
       i: Food source i.
       k: A randomly selected neighbor food source.
       j: A randomly selected position in solution x_i
       r: A random variable $\in [-1, 1]$.
     - Apply greedy selection between v_i and x_i
   - Calculate the probability values P_i for the solutions x_i by means of their fitness values by using (3)

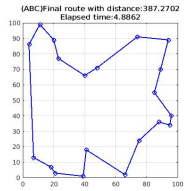
     $$P_i = \frac{0.9 * fit_i}{fit_{best}} + 0.1 \quad (3)$$

   - Onlooker bee phase: for each onlooker bee with probability P_i
     - Produce a new solution v_i in the neighborhood of x_i via: $v_{ij} = x_{ij} + r(x_{ij} - x_{kj})$.
     - Apply greedy selection between v_i and x_i
   - Determine the abandoned solution, if exists, replace it with a new produced solution for the scout.
   - Memorize the best solution achieved yet.
3. Until maxmimum iteration time.

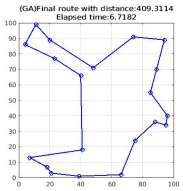
IV. Initial version of ABC for TSP

A. Baisc results

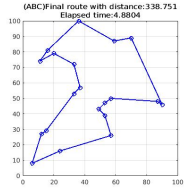
The ABC algorithm is implemented with MATLAB. For the first try, we create three different TSP maps, then use ABC algorithm and GA(given in the lecture PPT) to solve that. The



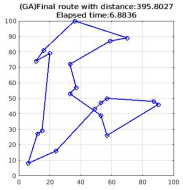
(a) ABC for problem 1



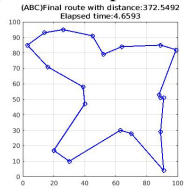
(b) GA for problem 1



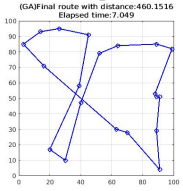
(c) ABC for problem 2



(d) GA for problem 2

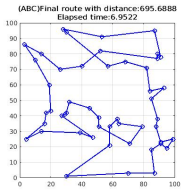


(e) ABC for problem 3

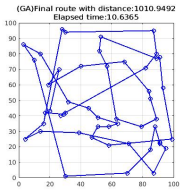


(f) GA for problem 3

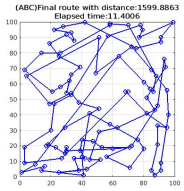
Fig. 1: Basic result of ABC, compared with GA



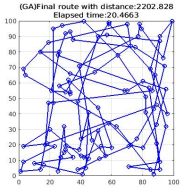
(a) ABC with city size=50



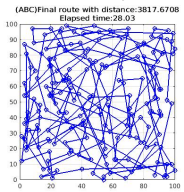
(b) GA with city size=50



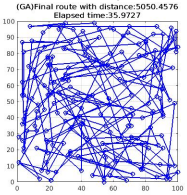
(c) ABC with city size=100



(d) GA with city size=100



(e) ABC with city size=200



(f) GA with city size=200

Fig. 2: Increased city size, compared with GA

population size of ABC and GA are the same. The results are illustrate in Fig.1.

As we can see from the results. The ABC performs better than the GA given in lecture PPT.

B. Varying Problem Size

Here we change the size of the problem, and run ABC and GA again.The population size of ABC and GA are kept the same. The results are shown in Fig.2.

As we can see from Fig.2. Considering the final optimized distance and consuming time for both ABC and GA. With same population size and same maxmimum iteration time, ABC acheives shorter distance with less time spent. Thus so far, ABC outperms GA.

C. Analysis

Here we breifly analyse why ABC outperfroms GA in the experiment.

1. As for distance:
   There exists re-optimization process in ABC:
   a. Food source with high fitness will be optimizied again.
   b. Food source with relative low fitness still have chance to be optimizied again.
2. As for time:
   Simple “swap mutation” operation saves time compared with “Crossover” and “Mutation” and many other operations in GA

D. Possible Improvements

Here we think of furthur improvements of the ABC algorithm. While doing the experiments, we find that the best solution at each iteration doesn’t decrease monotonically. This is shown in Fig.3. This may be due to the abandoment in Scout Bee Phase. While the scout bee abandons the bad solutions, the good solutions are abandoned as well. May a strategy like ”Elitism” can be added into the ABC to improve the algorithm. Also, the mutation in the ABC is too simple. Maybe a better mutation process(the process of find new food source) can be used in ABC.

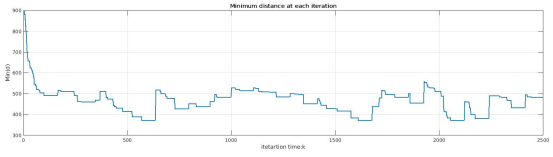


Fig. 3: Mimimum distance at each iteration

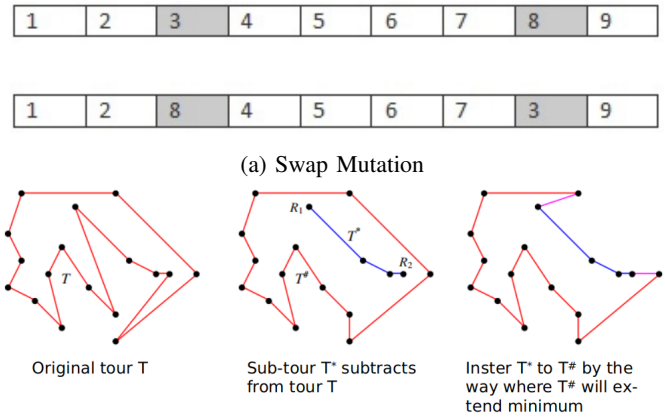
V. First improvement of ABC for TSP

In this part, we mainly focus on the improvement of mutation operation: after introducing a new mutation operation in ABC, we come up a adaptive mutation operation which can furthur improve the final distance while time cost increase much.

A. Adaptive Mutation Operation

In basic version of ABC, we use only use swap mutation, as shown in Fig.4(a). Swap mutation has the advantage of low computation cost, however, in late iteration, this simple mutation can hardly improve the food source. Thus we introduce minimum extension mutation[3], as shown in Fig.4(b).

In minimum extension mutation, firstly, from the original tour T,we randomly extract a sub-tour T^*, leaving a closed remaining tour $T^\sharp$. We then insert T^* back to $T^\sharp$ in a way where $T^\sharp$ get minimum extension. This kind of mutation can easily improve the food source, however, if we simply replace the swap mutation wiht the minimum extension mutation, the whole algorithm can take a great many time to do optimization. Thus we come up adaptive mutaiton operation.



(a) Swap Mutation

(b) Minimum Extension Mutation

Fig. 4: Mutation in ABC

The adaptive mutation operation works in the following way:

- Set a trial for best food source.
- For each iteration:
  - If best solution hasn't improved for certain time: Apply minimum extension mutation for each food source.
  - Else:Apply swap mutation for each food source.

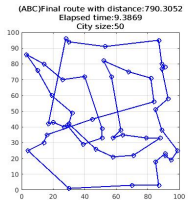
Thus the mutation operation alternate between swap mutation and minimum extension mutation.

B. Resutls

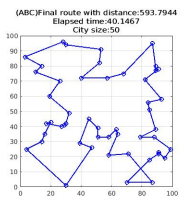
From Fig.5, we can see that although final distance get improved, it takes much longer time. Also, because we introduce the adaptive mutaion operation, the Scout Bee Phase deosn't take effect with the existence of adaptive mutation mechanism.

VI. Final version of ABC for TSP

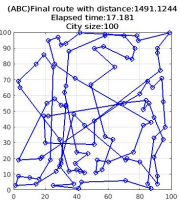
In final version of ABC for TSP, we only put the minimum extension to Scout Bee Phase and keep swap mutation in Employed Bee Phase and Onlooker Bee Phase.



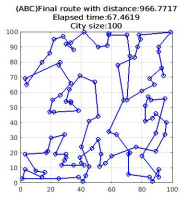
(a) Swap mutaion with city size=50



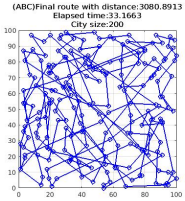
(b) Adaptive mutation with city size=50



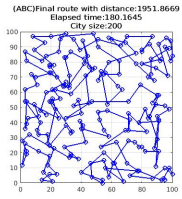
(c) Swap mutaion with city size=100



(d) Adaptive mutation with city size=100



(e) Swap mutaion with city size=200



(f) Adaptive mutation with city size=200

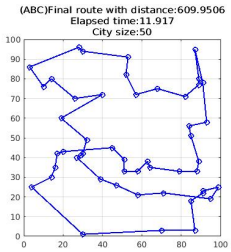
Fig. 5: Comparion of Swap mutation and Adaptive mutaion

A. Flowchart Of Final ABC For TSP

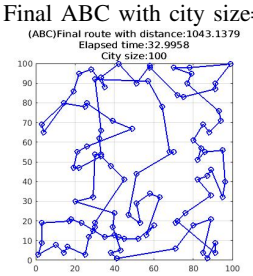
- Initialization.
- Repeat Until Maximum Itreration:
  - Employed Bee Phase
    - For each food source:
      - Generate new food source(solution) via swap mutation.
      - Calculate fitness for each food source.
  - Onlooker Bee Phase
    - With a probability,for each food source
      - Generate new food source(solution) via swap mutation.
      - Calculate fitness for each food source.
  - Scout Bee Phase
    - For food sources which have not been improved for certain time.
      - If it is a good solution, apply minimum extension mutation.
      - Else, randomly generate a new food source.

B. Results

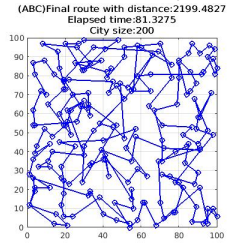
The result of final version ABC is shown in Fig.6, compared with Fig.5, we find that with little loss in final distance, the time cost reduced greatly.



(a) Final ABC with city size=50



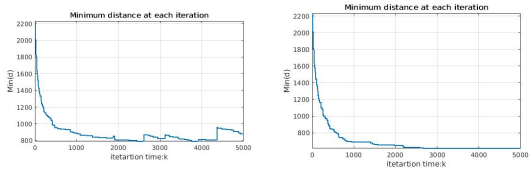
(b) Final ABC with city size=100



(c) Final ABC with city size=200

Fig. 6: Final version ABC

Also, the final version can solve the problem of scout bee phase in basic version ABC in which good results may also be abandoned. This improvement is shown in Fig.7: in basic version ABC, the best solution may be dropped, while in final version, the best solution decrease monotonously.

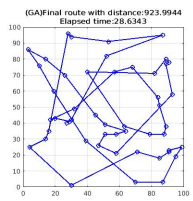


(a) Convergence in basic version ABC

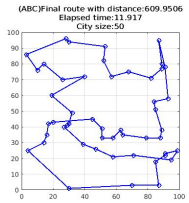
(b) Convergence in final version ABC

Fig. 7: Convergence comparison

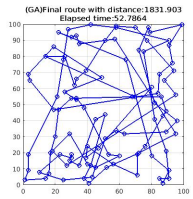
Finally we again compare the final version of ABC with GA, as shown in Fig.8. With less time, ABC achieves much more less distance than GA.



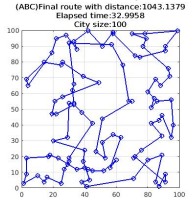
(a) GA,city size=50



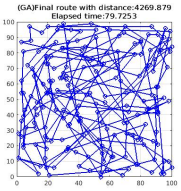
(b) Final ABC,city size=50



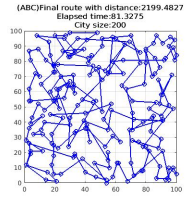
(c) GA,city size=100



(d) Final ABC,city size=100



(e) GA,city size=200



(f) Final ABC,city size=200

Fig. 8: Comparison of final version ABC and GA

VII. Conclusion

With new mutation operation and modified Scout Bee Phase, ABC algorithm gets much more improvements, it outperms GA greatly, especially when problem size is large. There is always a trade-off between time and final optimizied result in TSP. Also we leart that improvement of one module may lead to another module's failure, after improvement of one module, we should re- examine the whole procedure to avoid this.

References

[1] D. Karaboga and B. Gorkemli, "A combinatorial Artificial Bee Colony algorithm for traveling salesman problem," 2011 International Symposium on Innovations in Intelligent Systems and Applications, Istanbul, 2011, pp. 50-53.

[2] Shin Siang Choong, Li-Pei Wong, Chee Peng Lim,"An artificial bee colony algorithm with a Modified Choice Function for the traveling salesman problem",Swarm and Evolutionary Computation,Volume 44,2019,Pages 622-635.

[3] Allahverdi, N.: Development A New Mutation Operator to Solve the Traveling Salesman Problem by Aid of Genetic Algorithms. Expert Systems with Applications 38(3), 1313-1320