Fitness Function in ABC Algorithm for Uncapacitated Facility Location Problem

Yusuke Watanabe(✉), Mayumi Takaya, and Akihiro Yamamura

Department of Computer Science and Engineering, Akita University, 1-1, Tegata-Gakuenmachi, Akita, Japan
m9014090@ie.akita-u.ac.jp

Abstract. We study the fitness function of the artificial bee colony algorithm applying to solve the uncapacitated facility location problem. Our hypothesis is that the fitness function in the artificial bee colony algorithm is not necessarily suitable for specific optimization problems. We carry out experiments to examine several fitness functions for the artificial bee colony algorithm to solve the uncapacitated facility location problem and show the conventional fitness function is not necessarily suitable.

Keywords: Swarm intelligence · Artificial bee colony algorithm · Uncapacitated facility location problem · Fitness function

1 Introduction

Efficient supply chain management has led to increased profit, increased market share, reduced operating cost, and improved customer satisfaction for many businesses [8]. For this purpose, it is getting more and more important in information and communications technologies to solve optimization problem such as the *uncapacitated facility location problem* (UFLP) which is a combinatorial optimization problem. The objective of the UFLP is to optimize the cost of transport to each customer and the cost associated to facility opening, when the set of potential locations of facilities and the customers are given, whereas UFLP is known to be NP-hard (see [6]). In the context of performing economic activities efficiently, various objects have been considered as facilities, such as manufacturing plants, storage facilities, warehouses, libraries, fire stations, hospitals or wireless service stations. Several techniques have been applied to the UFLP such as the swarm intelligence algorithms or a meta-heuristics algorithm; *particle swarm optimization* (PSO) [3], *ant colony optimization* (ACO) [5], *artificial bee colony algorithm* (ABC) [9], and *genetic algorithm* [7].

The method of simulating swarm intelligence is inspired by the movement and foraging behavior in herd animals [10]. As typical examples, particle swarm optimization was inspired by the behavior of groups of birds and fish, ant colony optimization was inspired by the foraging behavior of ants. These have been applied to a wide range of computational problems like data mining and image

I. Khalil et al. (Eds.): ICT-EurAsia 2015 and CONFENIS 2015, LNCS 9357, pp. 129–138, 2015.
DOI: 10.1007/978-3-319-24315-3_13

processing [1] in addition to numerous optimization problems like the traveling salesman problem and the flow shop scheduling problem.

Inspired by the foraging behavior of honey bee swarm, D. Karaboga [4] proposed the artificial bee colony (ABC) algorithm, which is an optimization algorithm developed for a function optimization. In the ABC algorithm model, a honey bee swarm consists of three types of bees which carry out different tasks. The first group of bees are called *employed bees*. They have a potential solution, evaluate its fitness value and keep the better solutions in their memory. The second group of bees are called the *onlooker bees*. They choose potential solutions on the basis of information provided by the employed bees. If a potential solution has high fitness value, many onlooker bees choose the solution. The third group of bees are called the *scout bees*. They explore new potential solutions randomly. An employed bee whose potential solution has been abandoned becomes a scout bee. The ABC algorithm is operated in an iterative manner by these three groups of artificial honey bees to search a better solution. The ABC algorithm is applied to many real-world problems, however, its mechanism has not been understood in detail. In this paper we examine the fitness function that is an important piece of the ABC algorithm by implementing several variants and comparing with the original fitness function.

The paper is organized as follows. In Sect. 2 we review the uncapacitated facility location problem and application of the ABC algorithm to the UFLP. In Sect. 3 we show the results of our experiments and we examine the fitness function of the ABC algorithm applied to the UFLP. In the last section, we summarize our findings.

2 Artificial Bee Colony Algorithm for the Uncapacitated Facility Location Problem

2.1 Uncapacitated Facility Location Problem

In the UFLP, a finite set F of facilities can be opened and a finite set D of customers are present. Each facility i in F has a fixed opening cost $f_i \in \mathbb{R}_+$, where $\mathbb{R}_+$ stands for the set of positive real numbers. For each pair of i in F and $j \in D$, a transport cost $c_{ij} \in \mathbb{R}_+$ for serving the customer j from facility i is specified. It is possible to open any number of facilities and to assign each user to any of the facilities opened that will serve for the user. The task of the UFLP is to minimize the sum of the opening costs of the facilities and the transport costs for each customer, that is, to minimize

$$\sum_{i \in X} f_i + \sum_{j \in D} c_{\sigma(j)j}$$

where $X \subseteq F$ is a subset of facilities to be opened and $\sigma : D \to X$ is an assignment of each customer to an appropriate facility.

2.2 Artificial Bee Colony Algorithm

In the UFLP with n potential facilities, that is $|F| = n$, for each employed bee k is given the *open facility vector* $Y_k = [y_{k1}, y_{k2}, y_{k3}, ..., y_{kn}]$ representing a potential solution, where $y_{ki} = 1$ if the i-th facility is open and $y_{ki} = 0$ otherwise.

For the open facility vector Y_k of an employed bee k, X is defined to be the set of facilities i in F such that $y_{ki} = 1$. Then the allocation $\sigma : D \to X$ for Y_k from each customer in D to an appropriate facility in X is uniquely determined for the transport costs c_{ij} as follows. For each j in D, $\sigma(j)$ is defined to be $i \in X$ so that c_{ij} is the smallest among $\{c_{hj} \mid h \in X\}$. If there are several such i, we choose one of them randomly. The total cost x_k for each employed bee k is computed as the sum of the fixed cost of opening facility determined by the open facility vector Y_k and the transport cost of each customer to the opened facilities.

Table 1. Open facility vector for an employed bee k

Facility		A	B	C	D	E
Open facility vector Y_k		1	0	0	1	1
Opening cost		10	8	4	7	3
Customer (Transportation costs)	a	1	4	3	10	12
	b	9	8	7	4	3
	c	8	12	6	5	7
	d	15	10	6	10	13

Let us see a concrete example shown in Table 1. In this example, 5 facilities A, B, C, D, E and 4 customers a, b, c, d are given and the open facility vector Y_k for an employed bee k is given as $[1, 0, 0, 1, 1]$. The total cost x_k for the employed bee k is calculated as follows:

x_k
= open facilities fixes costs
+ min (cost of supply from open facilities to customer)
= (10 + 7 + 3) + min(1, 10, 12)
+ min(9, 4, 3) + min(8, 5, 7) + min(15, 10, 13)
= (20) + (1 + 3 + 5 + 10)=20 + 19=39.

The fitness value $fitness(x_k)$ of the potential solution of the employed bee k is calculated from its total cost x_k by the formula below that is given in [4].

$$fitness(x_k) \begin{cases} \frac{1}{1+x_k} & if \quad x_k \geq 0 \\ 1 + abs(x_k) & if \quad x_k < 0 \end{cases} \tag{1}$$

where $abs(x_k)$ stands for the absolute value of x_k. We remark that a larger fitness value is better.

Step 1: Initialization. First, each entry in the open facility vector Y_k of each employed bee k is initialized to either 0 or 1 at random. We compute the fitness value $fitness(x_k)$ by the formula (1) for each employed bee k in the population B_1 of employed bees. We remark that we do not use the later part $1 + abs(x_k)$ in the formula (1) because x_k is always a positive number in case of the UFLP.

Step 2: Employed Bee Phase. In each iteration, each employed bee k obtains a new potential solution candidate by manipulating the current potential solution. The open facility vector for k is updated by randomly selecting one facility i in F and switching the bit corresponding to the selected facility (see Fig. 1). If the fitness value of the new open facility vector for k is larger than the fitness value of the previous vector for k, we update the open facility vector for k.

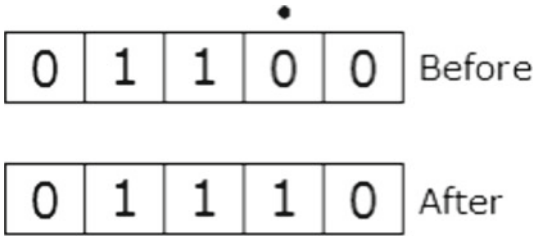

Fig. 1. Update of open facility vector

Step 3: Onlooker Bee Phase. To update onlooker bees, we select stochastically one of the potential solutions, that is, one of the open facility vectors of the employed bees, according to the probability p_k determined by the following formula:

$$p_k = \frac{fitness(x_k)}{\sum_{l=1}^{B_1} fitness(x_l)} \tag{2}$$

Note that the selection follows the probability above and so employed bees with larger fitness value have more bigger chance to be chosen. Like the process similar to Step 2, if the open facility vector of the selected employed bee has larger fitness value than the onlooker bee's current solution, then we replace the potential solution of the onlooker bee with the open facility vector of the selected employed bee. This procedure is repeated for all onlooker bees in the population B_2 of the onlooker bees.

Step 4: Scout Bee Phase. For a *cut limit* c which is determined in advance, if the fitness value of the open facility vector of an employed bee is not improved during c times iteration, then the employed bee is discarded. We harvest the employed bee whose open facility vector was discarded and turn it into a scout bee by giving a new random open facility vector. This operation prevents the search from falling into local optima.

Step 5: Termination Condition of Iteration. The termination condition is determined by the number N of iterations set in advance. When the program reaches N, it outputs the open facility vector whose fitness value is the largest and execution stops.

```
Begin
    Initialize solution randomly (Step1)
    Do
        For Each employ bee (Step2)
            search the new solution
            If the solution is improved
                Update the solution
        For Each Onlooker bee (Step3)
            Choice a solution by probability
            search the new solution
            If the solution is improved
                Update the solution
        For Each solution (Step4)
            If it is not updated
                search new solution randomly and update
    While (Maximum Iteration is not reached) (Step5)
End
```

Algorithm. Pseudocode of the ABC algorithm for UFLP.

3 Experimental Results

We examine the fitness function given by (1) when we implement the ABC algorithm for the UFLP to investigate the adequateness of the fitness function.

The ABC Algorithm was coded in C# language using Visual Studio and run on an Intel Core i3 3.07 GHz Desktop with 2.0 GB memory. We used 12 data sets (cap 71, 72, 73, 74, 101, 102, 103, 104, 131, 132, 133, 134) of benchmark problems from OR-Library [2] compiled by J.E. Beasley. UFLP is called Uncapacitated Warehouse Location Problem in the OR-Library. There are three groups of benchmark problems of size $m \times n$, where m is the number of customers and n is the number of facilities: 50×16 (cap 71, 72, 73, 74), 50×25 (cap 101, 102, 103, 104), 50×50 (cap 131, 132, 133, 134), and the optimal solutions for those instances are known. The performance of our program was evaluated by *average relative percent error* (ARPE), *hit to optimum rate* (HR) and *average computational processing time* (ACPU) that are introduced in [3].

ARPE is the average of the difference from the optimum expressed in percentages, and the lower ARPE shows it produces better solutions. HR represents the number that the algorithm finds the optimal solutions across all repetitions and the higher HR shows better performance. HR takes a value from 0.00 to 1.00 where 1.00 implies that the algorithm finds the optimal solution with probability 1. ACPU represents the time (in seconds) that the algorithm spends to output one solution and the lower ACPU shows better performance.

Table 2. Benchmark

Data set	Size(m × n)	Optimum
cap71	16×50	932615.75
cap72	16×50	977799.40
cap73	16×50	1010641.45
cap74	16×50	1034976.98
cap101	25×50	796648.44
cap102	25×50	854704.20
cap103	25×50	893782.11
cap104	25×50	928941.75
cap131	50×50	793439.56
cap132	50×50	851495.33
cap133	50×50	893076.71
cap134	50×50	928941.75

ARPE is defined by

$$ARPE = \sum_{i=1}^{R} \left(\frac{H_i - U}{U} \right) \times \frac{100}{R} \tag{3}$$

where H_i denotes the i-th replication solution value, U is the optimal value provided by [2], and R is the number of replications. In Table 2, we summarize the data sets we used for our experiments. It shows the size of data sets and the optimal solutions.

3.1 Experiment 1

We conducted an experiment to investigate 5 different candidates for fitness value as follows:

$$fitness(x_k) = 1 \tag{4}$$

$$fitness(x_k) = \frac{1}{1 + x_k} \tag{5}$$

$$fitness(x_k) = \frac{1}{1 + (x_k - x_*)} \tag{6}$$

$$fitness(x_k) = \frac{1}{1 + (x_k - x_*)^2} \tag{7}$$

$$fitness(x_k) = \frac{1}{1 + \sqrt{x_k - x_*}} \tag{8}$$

where x_* represents the best solution obtained so far. Note that the formula (5) is the original fitness function given in [4].

The results obtained in this experiment are shown in Figs. 2 and 3. For the benchmark problem cap131, we set a population of employed bees $B_1 = 50$, a population of onlooker bees $B_2 = 200$, and the number of repetitions $N = 100$ and the cut limit c ranges between 5 and 50. The x-axis of the graph represents the cut limit c and the y-axis of the graph represents HR in Fig. 2 and ARPE in Fig. 3, respectively. Comparing the formulas (4) and (5), we found that the two formulas show almost the same results. This implies that the formula (5) does not have a desired property as a fitness function. The formulas (6), (7) and (8) are designed so that the fitness value is affected largely by the value of x_k. If the cut limit c is set small, we obtain better results in the formulas (6), (7) and (8) than the formula (5), on the other hand, the cut limit c gets larger, then the formula (5) gives a better result. This indicates that we can obtain a better result in a shorter computation, although the search may fall in a local optimal.

Table 3. Experimental Results of the proposed fitness value

Problem	$fitness(x_k) = \frac{1}{1+x_k}$			$fitness(x_k) = \frac{1}{1000+(x_k-x_*)}$		
	ARPE	HR	ACPU	ARPE	HR	ACPU
cap71	0.000	1.00	0.1610	0.000	1.00	0.1674
cap72	0.000	1.00	0.1462	0.000	1.00	0.1490
cap73	0.000	1.00	0.1214	0.000	1.00	0.1215
cap74	0.000	1.00	0.1148	0.000	1.00	0.1173
cap101	0.000	1.00	0.2384	0.000	1.00	0.2482
cap102	0.000	1.00	0.2120	0.000	1.00	0.2231
cap103	0.000	1.00	0.2001	0.000	1.00	0.2061
cap104	0.000	1.00	0.1819	0.000	1.00	0.2065
cap131	0.349	0.06	0.5044	0.143	0.24	0.5357
cap132	0.214	0.03	0.4599	0.039	0.41	0.4976
cap133	0.173	0.04	0.4348	0.040	0.36	0.4501
cap134	0.039	0.73	0.4206	0.000	1.00	0.4360

3.2 Experiment 2

Considering the results of Experiment 1, we propose the following formula for the fitness value:

$$fitness(x_k) = \frac{1}{Q + (x_k - x_*)} \tag{9}$$

where Q is an arbitrary constant, given as a parameter to the algorithm.

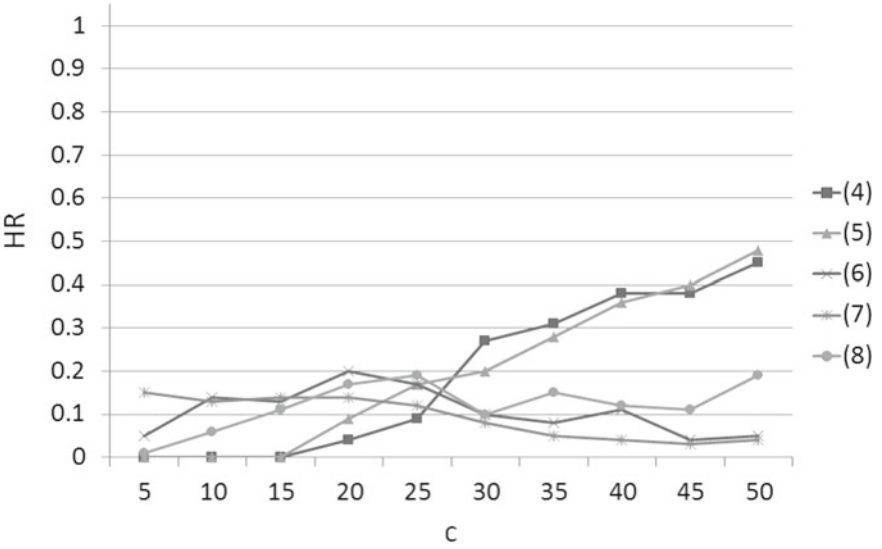

Fig. 2. Experiment 1: HR

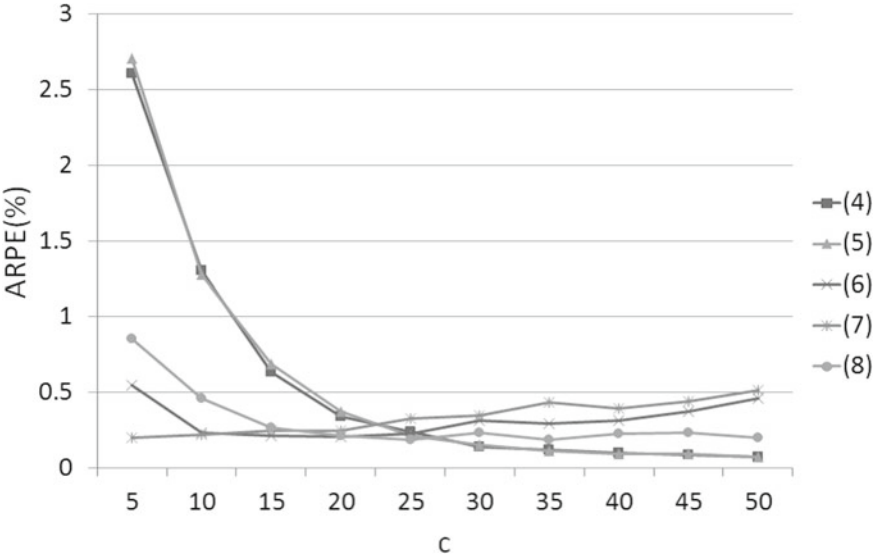

Fig. 3. Experiment 1: ARPE

Figures 4 and 5 show how the value of cut limit c between 5 and 50 affects HR and ARPE for the benchmark problem cap131 with bee populations $B_1 = 50$, $B_1 = 20$ and the number of iterations $N = 100$. If $Q = 10^6$ or 10^8, the fitness value of (5) is almost the same as the one for (9). If $Q = 1$, the fitness value of (9) is almost the same as the one for (6). If $Q = 10^4$, the fitness value of

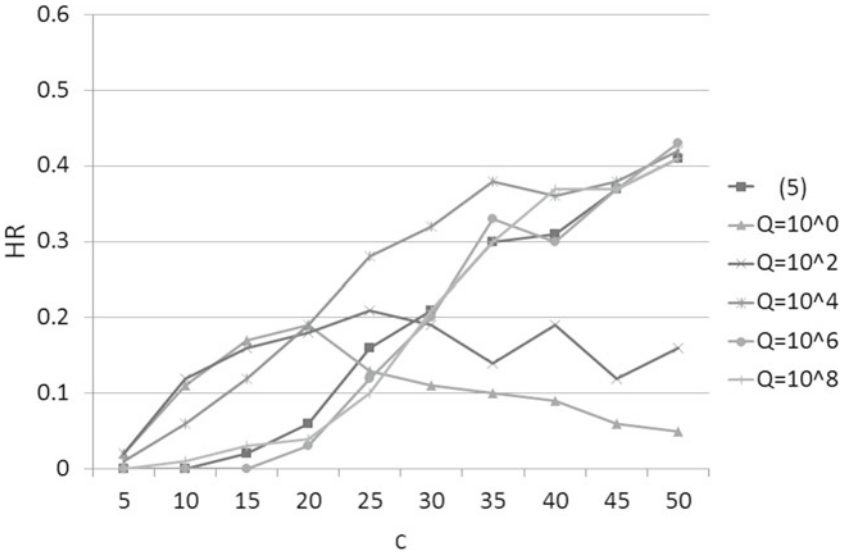

Fig. 4. Experiment 2: HR

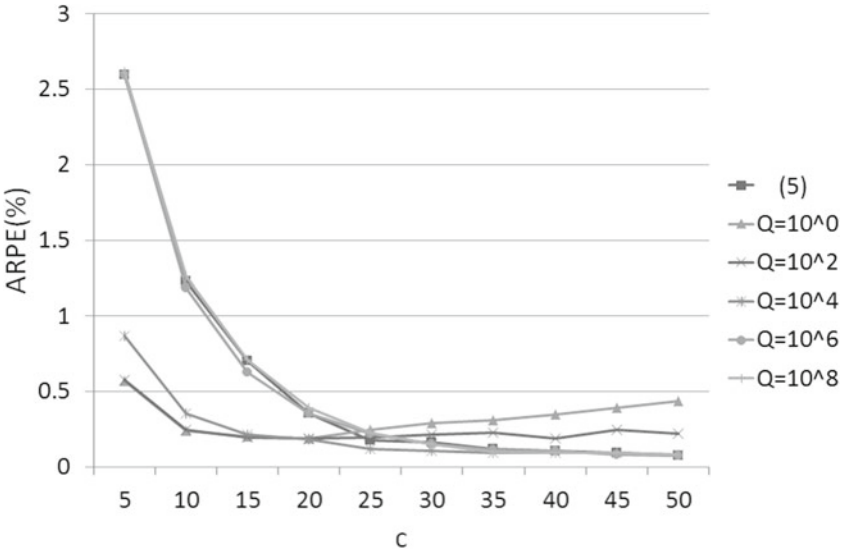

Fig. 5. Experiment 2: ARPE

(9) is better than (5). As a result of the experiment, we believe that the ABC algorithm operates more efficiently by the fitness formula (9) with $Q = 10^4$.

Finally, we show the result of performing the proposed fitness value and conventional fitness value for each set of benchmark problems in Table 3, where we set the population of employed bees $B_1 = 50$, the population of the onlooker bees $B_1 = 200$, the cut limit $c = 20$ and the number of repetitions $N = 100$.

4 Conclusions

In this paper we discussed the fitness function of the ABC algorithm for the UFLP by carrying out experiments of several variants and comparing with the original fitness function. Our experiment indicates the conventional formula (5) for the fitness value of the ABC algorithm is almost same as the fitness of a constant function given by (4) for the benchmark problems and so it seems inadequate for the UFLP. Therefore, we proposed a new formula (9) which can appropriately weight the fitness value according to problem instances. Using the formula (9) with a smaller value of Q, we can find more open facility vectors whose total costs are low and it makes the search efficient. However, we consider that the algorithm falls in a local optima when we use a too small value of Q and the algorithm does perform adequately. In our experiment, the ABC algorithm with the fitness function (9) with $Q = 10^4$ performs best and exceeds the original formula (5). The new candidate for the fitness function is worth being applied to other problems, although we do not know how to adjust the value of Q for specific optimization problem. It will be our future research to study how to adjust the value Q for a specific optimization problem such as TSP.

References

1. Abraham, A., Grosan, C., Ramos, V.: Swarm Intelligence in Data Mining. Springer-Verlag, Berlin Heidelberg (2006)
2. Beasley, J.E.: OR-Library (2005). http://www.brunel.ac.uk/mastjjb/jeb/info.html
3. Guner, A.R., Sevkli, M.: A discrete particle swarm optimization algorithm for uncapacitated facility location problem. J. Artif. Evol. Appl. **2008**, Article ID 861512, 9 p. (2008). http://dx.doi.org/10.1155/2008/861512
4. Karaboga, D.: An Idea Based on Honey Bee Swarm for Numerical Optimization. Erciyes University, Kayseri, Turkey, Technical report-TR06 (2005)
5. Kole, A., Chakrabarti, P., Bhattacharyya, S.: An ant colony optimization algorithm for uncapacitated facility location problem. Artif. Intell. Appl. **1**(1), 55–61 (2014)
6. Korte, B., Vygen, J.: Combinatorial Optimization: Theory and Algorithms. Springer, Heidelberg (2007)
7. Kratica, J., Tosic, D., Filipovic, V., Ljubic, I.: Solving the simple plant location problem by genetic algorithm. RAIRO Oper. Res. **35**, 127–142 (2001)
8. Simchi-Levi, D., Kaminsky, P., Simchi-Levi, E.: Designing and Managing the Supply Chain. Concepts, Strategies and Case Studies. McGraw-Hill, Boston (2000)
9. Tuncbilek, N., Tasgetiren, F., Esnaf, S.: Artificial bee colony optimization algorithm for uncapacitated facility location problems. J. Econ. Soc. Res. **14**(1), 1–24 (2012)
10. Yang, X., Cui, Z., Xiao, R., Gandomi, A.H., Karamanoglu, M.: Swarm Intelligence and Bio-Inspired Computation - Theory and Applications. Elsevier, Amsterdam (2013)