Advanced Engineering Forum *Online: 2012-09-26*
ISSN: 2234-991X, Vols. 6-7, pp 445-451
doi:10.4028/www.scientific.net/AEF.6-7.445

A Novel Multi-Objective Artificial Bee Colony Algorithm for the QoS Based Wireless Route Optimization Problem

Changsheng Zhang[1], Mingkang Ren[1], Bin Zhang[1*]

[1]College of Information Science & Engineering, Northeastern University, Shenyang 110819, China

[a]email:zcs820@yahoo.com.cn, [b]email: 935627579@qq.com, [c]email:paper_820@yahoo.com.cn

Keywords: Artificial Bee Colony, Multi-Objective Optimization, Wireless Route Optimization.

Abstract. In this paper, an efficient multi-objective artificial bee colony optimization algorithm based on Pareto dominance called PC_MOABC is proposed to tackle the QoS based route optimization problem. The concepts of Pareto strength and crowding distance are introduced into this algorithm, and are combined together effectively to improve the algorithm's efficiency and generate a set of evenly distributed solutions. The proposed algorithm was evaluated on a set of different scale test problems and compared with the recently proposed popular NSGA-II based multi-objective optimization algorithm. The experimental results reveal very encouraging results in terms of the solution quality and the processing time required.

Introduction

The QoS based route optimization is a key issue of the wireless network, which is a NPC-hard multi-objective optimization problem[1]. For a given wireless network which is modeled as a undirected graph G=(N,E), where N is the set of nodes and E is the set of links, the QoS based route optimization problem is to find a path from the source node $s \in N$ to the destination node $t \in N$ that meets different optimization criteria and satisfies the specified constraints. The attribute delay, cost and free space loss related criteria are considered simultaneously in this paper, and the problem is formulated as follows:

$$\text{Min: } f(p) = \left(f_1(p), f_2(p), f_3(p)\right) \tag{1}$$

where

$$f_1(p) = \sum_{(i,j)\in E} d_{ij} \times x_{ij}$$

$$f_2(p) = \sum_{(i,j)\in E} w_{ij} \times x_{ij}$$

$$f_3 = \max\left(\left(\frac{4\pi l_{ij}}{\lambda_{ij}}\right)^2 \times x_{ij}\right)_{(i,j)\in E}$$

Subject to the constraints:

$$\sum_{(i,j)\in E} x_{ij} = 1,\ i = s$$

$$\sum_{(i,j)\in E} x_{ij} = 1,\ j = d$$

$$\sum_{(i,j)\in E} x_{ij} - \sum_{(j,i)\in E} x_{ji} = 0,\ i \neq s, j \neq d$$

The f_1, f_2 and f_3 are the objective functions related to the delay, cost and free space loss attributes respectively. The p denotes the path from node s to d. The w_{ij} and d_{ij} denote the cost and delay time of the link $(i,j) \in E$ respectively. The wavelength and the distance between the antennas i and j are represented as λ_{ij} and l_{ij}. The x_{ij} is a binary variable, that takes the value of 1 if the link $(i,j) \in E$ is used to transport data stream from the source node s up to the destination d; otherwise, it takes the value of 0.

Due to the computational complexity, the most related researches are concentrated on heuristic-based algorithms especially the meta-heuristic approaches aiming to find near-optimal solutions[1][2]. As a novel meta-heuristic approach, the ABC algorithm is defined by Dervis Karaboga[3], motivated

by the intelligent behavior of honey bees. It has been applied to solve many problems and obtained satisfying results[4]-[8]. But, the most existing ABC based algorithms focus on single objective optimization problem and little research has been done to apply this algorithm to discrete multi-objective optimization problem.

In this paper, the ABC algorithm is extended and a Pareto-based multi-objective discrete artificial bee colony algorithm called PC_MOABC is proposed to solve the QoS based multi-objective route optimization problem. The concepts of Pareto strength and crowding distance are introduced and combined together effectively to improve the algorithm's efficiency and generate a set of evenly distributed solutions. The proposed algorithm was evaluated on a set of different scale test problems and compared with the recently proposed NSGAII based multi-objective optimization algorithm[2]. This paper is organized as follows. In Section 2, we give the formalization of PC_MOABC algorithm including its model and concrete algorithms description. The experiments and comparative studies are given in section 3. Finally, in section 4, we summarize the contributions of this paper along with some future research directions.

PC-MOABC Algorithm

Since the basic ABC algorithm is proposed and used to solve the continuous single objective optimization problems, it can not be directly applied to solve the discrete multi-objective route optimization problem. In order to apply it to solve this problem, we have used an encoded variable integer array to represent a candidate solution, which denotes a path from the source node s up to the destination t expressed as p=(s, n_1, n_2,..... n_s, t), where $n_i, i \in [1,s]$ is the node number existing in this path. In order to generate a feasible path during search, a random Depth-first searching process called *Search_Path* is proposed and used to find a path from the node s to the destination node t. In this process a mark array is used to store some nodes which have been added to the path. Then a node that connects with the current node n_i is selected randomly. If there is no nodes connecting with n_i, then delete n_i from the path, mark the n_i as a no access node and mark the link between n_i and n_{i-1} in the path disconnected. Because the network is connected, the algorithm will be able to find a path eventually. The detail description of this process is as follows**.**

Algorithm Search_Path(*s*,*t*)
Input:
N: The number of the networks nodes
Adj[*N*][*N*]: The adjacent matrix
Output:
path: A path from the node *s* up to the node *t*
Begin
Mark[*s*]=1; //The mark array, marks the nodes have
//been added to the path
while(path.last_node!=*t*) ***do*****{**
node=path.last_node;
for *i*=0 to *N*-1 ***do*****{**
if(Mark[*i*]==0 && Adj[*node*][*i*]==1){
SelectSet.add(*i*); /*SelectSet stores the nodes connecting with the current node and out of the path*/
}***endif***
}***endfor***
if(SelectSet.size==0) **{**
node1=path.last_node;
path.pop(*node1*);
node2=path.last_node;
Mark[*node1*]=0;
Adj[*node1*][*node2*]=Adj[*node2*][*node1*]=0;
} ***else*****{**

```
        pos=rand(0, SelectSet.size-1);
        Mark[SelectSet [pos]]=1;
        path.add(SelectSet [pos]);
    } endif
  }endwhile
End.
```

Furthermore, since more than one objective is considered, the way to compare two feasible solutions in the original ABC algorithm can not be used. In order to solve this problem, the dominance concept[9] is introduced and defined as follows:

***Definition1* (*Dominance*):** For the two feasible solutions *X and Y* of a minimization optimization problem: *min F*=($f_1, f_2, f_3, \ldots f_k$) , say X dominate Y, *iff* $\forall i$, $f_i(X)<=f_i(Y)$, and $\exists i$, $f_i(X)<f_i(Y)$, $i\in[1,k]$.

Based on this concept, the dominance strength and the crowding distance are introduced and used to evaluate the quality of a food source located in θ_i. The dominance strength of a food source located in θ_i is defined as follows:

$$Strength(\theta_i)=\frac{1}{L(\theta_i)} \tag{2}$$

$L(\theta_i)$ represents the front layer of θ_i which is calculated base on the dominance concept. To compute the crowding distance which can reflect the exploitation degree around a food source, the objective values must be normalized. In this paper, the crowding distance of a food source located in θ_i will be computed as follows:

$$d(\theta_i)=\frac{\sum_{\theta_j\neq\theta_i}\sum_{k=1}^{3}\left(\frac{f_k^{\theta_i}}{\max_k}-\frac{f_k^{\theta_j}}{\max_k}\right)^2}{3\times(S-1)} \tag{3}$$

$\max_k$ represents the maximum value of the *kth* objective in all food sources, S represents the amount of food sources, i represents the ith food source, $i\in[1, S]$.

Based on the two quality indicators defined above, two feasible candidates *X and Y* for this problem can be compared use the following *greedy rules*: If their dominance strengths are different, then the one with bigger dominance strength is better; If they have the same dominance strengths, then the one with bigger crowding distance is better; If their dominance strengths and crowding distances are both the same, then they possess the same quality.

Furthermore, onlookers select food sources from the sharing information of employed bees. The choosing probability used by onlookers with the food source located at θ_i in the original ABC algorithm can not be directly used. In this paper, the choosing probability used by onlookers with the food source located at θ_i is redefined as follows.

$$P(\theta_i)=\frac{Strength(\theta_i)+d(\theta_i)}{\sum_{j=1}^{S}\left(Strength(\theta_j)+d(\theta_j)\right)} \tag{4}$$

Obviously, the dominance strength and the crowding distance are both considered. Through this way, the colony diversity can be increased, and the obtained solutions can also be made distributed more evenly.

Moreover, a variable archive is used to hold the best solutions ever found. During each generation, the new discovered food sources are used to update the archive. The way of updating is based on the dominance strength. In order to produce a candidate food position based on the current food source θ_i for the employed bees and onlookers, the PC_MOABC algorithm uses the following strategy: Selecting a node n_i randomly from the path that corresponding to the food source θ_i, reserving the nodes before n_i, and continuing to search a new path starting from the node n_i using the *Search_Path* process. If the new food source is better than the old one, the new one will replace the old one, and its trial will be changed to 0; If the old one is better than the new one, the employed bee will continue to use the old one and its trial will be increased by 1; If they possess the same quality, the new one will replace the old one, but its trial will not be changed. If a scout discovered a

rich food source, it would be employed. To simulate this behavior, the following strategy is used: when a candidate can not be improved further through a predetermined number of cycles which is called "limit" for abandonment, the food source is assumed to be abandoned and the corresponding employed bee becomes a scout for exploration. Since the scout does not have any guidance while looking for food, the scout will find a path from the source node *s* up to the destination node *t* using the random *Search_Path* process. Based on the above descriptions, The PC_MOABC algorithm can be detailed as follows:

Step 1: Initialization

Step 1.1 Set the control parameter values.

Step 1.2 Use the *Search_Path* process to randomly generate a food position for each employed bees.

Step 1.3 ***for*** each food source ***do***
Calculate its dominance strength using Eq. (2);
Calculate its crowding distance using Eq. (3)
endfor

Step1.4 Add the Pareto front into the external archive.

Step 2: Preferences computation for the food sources

for each food source ***do***
Calculate its choosing probability using Eq. (4); ***endfor***

Step 3*: *Employed bees exploitation

for each employed bee e_i ***do***
produce new solution and update the food source;
if(e_i.*limit*>T)
it becomes a scout for exploration.
endif
endfor

Step 4: Scouts exploration

for each employed bee e_i ***do***
Use the *Search_Path* process to discover a new food source *fs*.
Make it to become and employed bee related to *fs*. ***endfor.***

Step5*: *Onlookers exploitation

for each onlooker bee o_i ***do***
Choose a food source *fs* based on its preferences;
Produce a new candidate *fn* around *fs*;
update the *fs* using the defined greedy rules; ***endfor.***

Step 6*: *Compute the dominance strength and the crowding distance for each food source

Step 6.1 Merge the food sources remembered by employed bees and contained in the archive into a temporary set called *Temp_Set*.

Step 6.1 ***for*** each food source remember by employed bee ***do***
Calculate its dominance strength using Eq. (2) based on the Temp_Set.
Calculate its crowding distance using Eq. (3) based on the Temp_Set. ***endfor***

Step 7*: *Check the termination criteria

if the termination condition is satisfied,
Output the food sources in the archive;
Stop the algorithm running;
else ***go to step 2***; ***endif***.

Experimental result

In order to evaluate the effectiveness of the proposed PC_MOABC algorithm, we compared it with the recently proposed NSGA-II based multi-objective optimization algorithm[2] on different scale test instances of the QoS based route optimization problem. We compare them from the solving efficiency and solution quality aspects respectively. The used four different scale test instances are

generated using the rules in [1], and are called G1(20, 72), G2(40,140), G3(50,200), G4(70, 280) and G5(80,400), G6(100, 568) respectively. For conveniently, if not specially pointed, during comparisons for all test instances, the swarm size and maximum iteration number for both algorithms are set as 200 and 3000, the value of the threshold "*limit*" of the PC_MOABC algorithm is set as 4, the crossover and mutation probability of the used NSGA-II algorithm are set as the same as in paper 2. Both algorithms run twenty times for each test instance on a Core(i7), 2.93GHZ, 2GB RAM computer and are implemented in C++ language. The obtained maximum, minimum and average time for each algorithm spent on each test instance is given in Fig. 1.

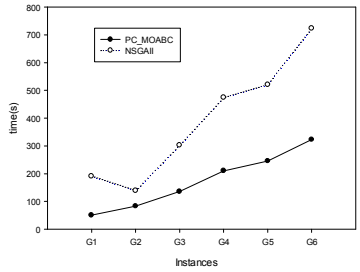

Fig.1 Experimental Results of consumed time

We can see that the PC_MOABC algorithm is greatly faster than the compared NSGA-II algorithm for all test instances. This is mainly because there are two populations with the same size, the parent population and offspring population in the NSGA-II algorithm, and the computing of non-dominate sorting and crowding distance are both based on the merging result of this two populations. So, when the swarm size is set the same of the two algorithms, The NSGA-II will consume more time and space.

In order to provide a quantitative assessment for the performances of these algorithms, the metrics called IDG and C-metric which is proposed as a complementary of the IDG are used[10]. The solutions obtained by PC_MOABC algorithm for each test instance are merged together first. Then compute their dominance strengths and make the solutions with the biggest dominance strength as the Pareto fronts obtained by this algorithm for this test instance and denoted as *P*. The same processing is applied to the solutions obtained by NSGA-II algorithm and the obtained Pareto front is denoted as *N*. For each test instance, in order to approximate its ideal Pareto front, we run each algorithm ten times and set the maximum generation number as 10000 for both of the two algorithms. The obtained results are all merged together, compute their dominance strengths and make the solutions with the biggest dominance strength as the ideal Pareto fronts. for each test instance which are showed as in Fig. 2, the values of IDG and C-metric are computed. The comparative results based on the two metrics are given in table-1 and table-2.

Table 1 Experimental Results of IGD values

Algorithm	G1	G2	G3	G4	G5	G6
PC_MOABC	0.00	9.40	6.59	22.92	10.46	6.84
NSGA-II	0.00	10.92	9.43	26.14	15.66	12.58

Table 2 Experimental Results of C-metric values

C-metric	G1	G2	G3	G4	G5	G6
$C(N \prec P)$	0.50	0.42	0.25	0.40	0.32	0.26
$C(P \prec N)$	0.50	0.55	0.28	0.60	0.54	0.63

From the table-1 and table-2, we can see that the IDG values obtained by PC_MOABC algorithm are all smaller than the IDG values obtained by NSGA-II algorithm, and the values of $C(P \prec N)$ are all bigger than the values of $C(N \prec P)$ for all the test instances except G1. For the small test

instance G1, not only the IDG values obtained by the two algorithms are the same, but also the values of $C(P \prec N)$ and $C(N \prec P)$ are also both the same.

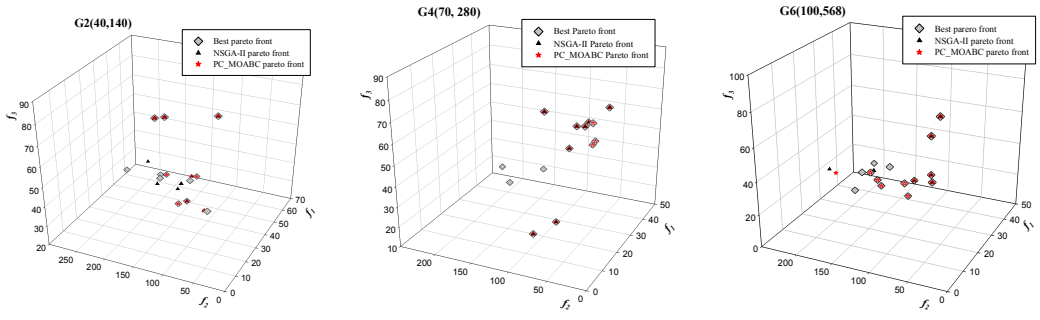

Fig.2 Experimental Results of Obtained Pareto Fronts

So, we can say that the PC_MOABC algorithm is more efficient and has acquired more effective solutions than the compared NSGA-II algorithm.

Conclusions

Modeling the behavior of social insects, such as ants, birds or bees, for the purpose of search and problem solving has been the emerging area of swarm intelligence. In this paper, an artificial bee colony based multi-objective optimization algorithm is developed to solve the QoS based route optimization problem. To evaluate the performance of this algorithm, it is compared with the recently proposed NSGA-II based multi-objective optimization algorithm and the experimental results reveal very encouraging results in terms of the quality of solution and the processing time required. There are a number of research directions that can be considered as useful extensions of this research. We can combine it with some local search strategy or hybrid it with other meta-heuristic algorithms properly, and apply it to solve other multi-objective optimization problems.

Acknowledgments: This work was supported by NSFC Major Research Program(61100090, 61073062), and the Special Fund for Fundamental Research of Central Universities of Northeastern University(110204006), and the Foundation for new Teachers of Education Ministry (2010 0042120040).

References

[1] Y. Donoso, R. Fabregat, multi-objective optimization in computer networks using Metaheuristics, Auerbach Publishers, New York (2007).

[2] C. Chitra, P. Subbaraj, A non-dominated sorting genetic algorithm solution for shortest path routing problem in computer networks, Expert Systems with Applications, 39(1):1518-152, 2012.

[3] D. Karaboga, B. Basturk. On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing, 8(1), 687–697, 2008.

[4] M. Fatih Tasgetiren, et al. A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops, Information Sciences, 181(16):3459-3475, 2011.

[5] M. H. Horng. Multilevel thresholding selection based on the artificial bee colony algorithm for image segmentation. Expert Systems with Applications, 38(11):13785-13791, 2011.

[6] D. Karaboga, C. Ozturk. A novel clustering approach: Artificial Bee Colony (ABC) algorithm,Applied Soft Computing, 11(1):652-657, 2011.

[7] W. C. Yeh, T. J. Hsieh. Solving reliability redundancy allocation problems using an artificial bee colony algorithm. Computers & Operations Research, Volume 38(11):1465-1473, 2011.

[8] W.Y. Szeto, Y. Z. Wu, Sin C. Ho. An artificial bee colony algorithm for the capacitated vehicle routing problem. European Journal of Operational Research, 215(1):126-135, 2011

[9] K. Deb, A. Pratap and S. Agarwal. A fast and elitist multi-objective genetic algorithm: NSGAII, IEEE Transactions on Evolutionary computation, 6(2):182-197, 2002.

[10] X. Masip-Bruina, M. Yannuzzib, J. Domingo-Pascual, etc., Research challenges in QoS routing, Computer Communications, 29(5):563-581, 2006