European Journal of Operational Research 245 (2015) 168–193

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Decision Support

An elitism based multi-objective artificial bee colony algorithm ☆

Yi Xiang [a,b], Yuren Zhou [a,*], Hailin Liu [c]

[a] School of Advanced Computing, Sun Yat-sen University, Guangzhou 510275, P.R. China
[b] Department of Advanced Mathematics, Guangdong Baiyun University, Guangzhou 510450, P.R. China
[c] Department of Applied Mathematics, Guangdong University of Technology, Guangzhou 510090, P.R. China

ARTICLE INFO

Article history:
Received 27 April 2014
Accepted 2 March 2015
Available online 6 March 2015

Keywords:
Heuristics
Multi-objective optimization
Elitism strategy
Artificial bee colony algorithm

ABSTRACT

In this paper, we suggest a new multi-objective artificial bee colony (ABC) algorithm by introducing an elitism strategy. The algorithm uses a fixed-size archive that is maintained based on crowding-distance to store non-dominated solutions found during the search process. In the proposed algorithm, an improved artificial bee colony algorithm with an elitism strategy is adopted for the purpose of avoiding premature convergence. Specifically, the elites in the archive are selected and used to generate new food sources in both employed and onlooker bee phases in each cycle. To keep diversity, a member located at the most crowded region will be removed when the archive overflows. The algorithm is very easy to be implemented and it employs only a few control parameters. The proposed algorithm is tested on a wide range of multi-objective problems, and compared with other state-of-the-art algorithms in terms of often-used quality indicators with the help of a nonparametric test. It is revealed by the test procedure that the algorithm produces better or comparable results when compared with other well-known algorithms, and it can be used as a promising alternative tool to solve multi-objective problems with the advantage of being simple and effective.

1. Introduction

Multi-objective optimization is a common problem faced by scientists and engineers, which concerns optimizing problems with multiple and often conflicting objectives. In principle, there is no single solution for a multi-objective optimization problem (MOP), but a set of Pareto-optimal solutions (Deb, Pratap, Agarwal, & Meyarivan, 2002a). This paper considers the following continuous MOP:

$$\begin{aligned} &\text{Minimize} F(x) = (f_1(x), f_2(x), \ldots, f_m(x)), \\ &\text{subject to } x \in \prod_{i=1}^{n} [lb_i, ub_i], \end{aligned} \tag{1}$$

where $\prod_{i=1}^{n} [lb_i, ub_i]$ is the decision space. $f : \prod_{i=1}^{n} [lb_i, ub_i] \to R^m$ consists of m real-valued continuous objective functions $f_1, f_2, \ldots, f_m$. R^m is called the objective space. Unlike single objective optimization, solutions of a MOP are in such a way that the performance of each objective cannot be improved without sacrificing the performance of at least another one. Hence, the solution to a MOP exists in the form of an alternate trade-off known as a Pareto-optimal set. The Pareto-optimal set is defined based on Pareto dominance (Akbari, Hedayatzadeh, Ziarati, & Hassanizadeh, 2012). Let $u = (u_1, u_2, \ldots, u_m)$ and $v = (v_1, v_2, \ldots, v_m)$ be two vectors in the objective space, u is said to dominate v if and only if $u_i \le v_i$ for each i and there exists at least one index such that $u_i < v_i$. If there is no x in the decision space such that $F(x)$ dominates $F(x^*)$, x^* is called a (globally) Pareto-optimal solution, and $F(x^*)$ is a Pareto-optimal objective vector. The set of all Pareto-optimal solutions is called the *Pareto set* (*PS*) and the set of all Pareto-optimal objective vectors is the *Pareto front* (*PF*) (Liu, Gu, & Zhang, 2014).

Over the past decades, a number of multi-objective evolutionary algorithms (MOEAs) have been suggested. Of them, Knowles and Corne's PAES (Knowles & Corne, 1999), Zitzler *et al.*'s SPEA2 (Zitzler, Laumanns, & Thiele, 2001) and IBEA (Zitzler & Künzli, 2004), Deb's NSGAII (Deb et al., 2002a), Zhang *et al.*'s MOEA/D (Zhang, Liu, & Li, 2009a), Kukkonen and Lampinen's GDE3 (Kukkonen & Lampinen, 2009), Nebro *et al.*'s MOCell (Nebro, Durillo, Luna, Dorronsoro, & Alba, 2009b) enjoyed more attention. Some particle swarm-based multi-objective algorithms, such as Sierra and Coello Coello's OMPSO (Sierra & Coello, 2005), and Nebro *et al.*'s SMPSO (Nebro et al., 2009a) were also well studied. Besides, hybrid algorithms such as Nebro *et al.*'s AbYSS (Nebro et al., 2008) and Durillo *et al.*'s CellDE (Durillo, Nebro, Luna, & Alba, 2008) were also suggested in corresponding literature. All the above multi-objective optimization algorithms were developed for finding multiple Pareto-optimal solutions to approximate the true PF in one single simulation run.

☆ This paper is jointly supported by the Natural Science Foundation of China (Grant nos. 61472143 and 61170081) and the Foundation for Distinguished Young Talents in Higher Education of Guangdong Province (Grant no. 2014KQNCX236).

* Corresponding author. Tel.: +86 20 39380288.
E-mail addresses: gzhuxiang_yi@163.com (Y. Xiang), yrzhou@scut.edu.cn (Y. Zhou).

http://dx.doi.org/10.1016/j.ejor.2015.03.005
0377-2217/

Algorithm: eMOABC

◇ **Input**:
- ◇ *CS*, the size of the bee colony;
- ◇ *AS*, the size of the crowding *archive*;
- ◇ *limit*, the abandonment criteria;
- ◇ *maxCycle*, the termination criteria.

◇ **Output**:
- ◇ *archive*.

1. Initialization:
1.1 Initialize the *colony* with the parameter *CS*;
1.2 Initialize crowding-distance *archive* with the parameter *AS*.
1.3 Add non-dominated solutions within the initial *colony* into the *archive*.
// Compute crowding-distances for the members in the initial *archive*.
1.4 crowdingDistanceAssignment (*archive*);
2. For (*t=0; t< maxCycle*-1*; t++*) {
2.1 SendEmployedBees (*colony*, *archive*);
2.2 crowdingDistanceAssignment (*archive*);
2.3 SendOnlookerBees (*colony*, *archive*);
2.4 crowdingDistanceAssignment (*archive*);
2.5 SendScoutBees (*colony*, *limit*).
}
3. Return *archive*

Fig. 1. The flowchart of the eMOABC algorithm.

Function 2: crowdingDistanceAssignment (*archive*)

1. $n = |archive|$ // number of solutions in the *archive*
// Initialize distance as 0.0 for each solution i
2. For each i, $archive[i]$.setCrowdingDistance (0.0)
3. **For** each objective m
// Sort *archive* using each objective value
3.1. $archive$ = sort ($archive$,m)
3.2. $archive[1]$.setCrowdingDistance (∞)
3.3. $archive[n]$.setCrowdingDistance (∞)
3.4. **For** i = 2 to n-1
$distance = archive[i+1].m - archive[i-1].m$
$distance = distance/(\text{objMax}.m - \text{objMin}.m)$
// (objMax.m-objMin.m), range of the mth objective
$archive[i]$.setCrowdingDistance ($distance$)
End For i
End For m

End of crowdingDistanceAssignment (*archive*)

Fig. 2. The pseudo code of crowdingDistanceAssignment (*archive*).

Function 1: SendEmployedBees (*colony*, *archive*)

Select the *elite* from the *archive*
For $i = 1$ **to** *FN*
Determine one dimension d to be modified randomly
Select a neighbor x_k from the *colony* stochastically
For food source x_i, calculate its new position v_i
$v_{id} = x_{id} + \phi_{id} \cdot (x_{id} - x_{kd}) + \varphi_{id} \cdot (x_{id} - elite_d)$
Evaluate v_i
If v_i dominates x_i
$x_i \leftarrow v_i$
$trial_i = 0$
Add v_i into the *archive*.
Else If x_i and v_i are non-dominated with each other
If v_i is successfully added into the *archive*
$x_i \leftarrow v_i$
$trial_i = 0$
Else
$trial_i = trial_i + 1$
End If
Else
$trial_i = trial_i + 1$
End If
End For
Use the function $calculateFitness$ (*colony*, *archive*) to compute fitness value of each food source

End of SendEmployedBees (*colony*, *archive*)

Fig. 3. The pseudo code of SendEmployedBees (*colony*, *archive*).

If $v_{id} < lb_d$
If $rand(0,1) \le 0.5$
$v_{id} = lb_d$
Else
$v_{id} = lb_d + 10^{-3} * rand(0,1) * (ub_d - lb_d)$
End If
End If
If $v_{id} > ub_d$
If $rand(0,1) \le 0.5$
$v_{id} = ub_d$
Else
$v_{id} = ub_d - 10^{-3} * rand(0,1) * (ub_d - lb_d)$
End If
End If

Fig. 4. The boundary treatment method used in the eMOABC algorithm.

The artificial bee colony (ABC) algorithm, proposed by Karaboga and Basturk (2007), is one of the most recently introduced swarm-based search methods (Akay & Karaboga, 2012; Szeto, Wu, & Ho, 2011). In ABC, there are three kinds of bees who do different tasks to make the algorithm useful. The employed bees will be sent to food sources and try to improve them by using neighbor information. Each onlooker bee will choose one of those food sources by a greedy strategy based on the quality information of food sources shared by employed bees, and then try to improve it. Finally, a scout bee will find a food source which has not been optimized in a limited number of cycles so as to re-initialize it to get rid of the poor solution. The ABC seems particularly suitable for multi-objective optimization mainly because it obtains high-quality solutions and shows fast speed of convergence for single-objective optimization (Akbari et al., 2012).

In recent years, extending basic ABC for handling multi-objective problems has received more attention from researchers. Some excellent works have been reported: Hedayatzadeh et al. proposed a multi-objective artificial bee colony (MOABC) which has adapted the basic ABC algorithm to multi-objective problems with a grid-based approach for maintaining and adaptively assessing the Pareto front. In this algorithm, the ε-dominance method was used to update the external archive (Hedayatzadeh, Hasanizadeh, Akbari, & Ziarati, 2010). The MOABC was compared with MOPSO (Coello Coello, Pulido, & Lechuga, 2004) and NSGAII (Deb et al., 2002a) on six test functions in terms of two performance metrics; Omkar et al. presented a method called the Vector-Evaluated ABC (or VEABC) for the multi-objective design of composite structures (Omkar, Senthilnath, Khandelwal, Naik, & Gopalakrishnan, 2011). In the VEABC, multiple populations were used to concurrently optimize the problem at hand. Zou et al. presented a novel artificial bee colony (ABC) algorithm for solving multi-objective optimization problems. In this algorithm, they