Applied Soft Computing xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

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

A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization☆

Yi Xiang[a,b], Yuren Zhou[a,b,*]

[a] School of Data Science and Computer, Sun Yat-sen University, Guangzhou 510275, PR China
[b] Collaborative Innovation Center of High Performance Computing, Sun Yat-sen University, Guangzhou 510275, PR China

ARTICLE INFO

Article history:
Received 29 October 2014
Received in revised form 1 May 2015
Accepted 23 June 2015
Available online xxx

Keywords:
Multi-objective optimization
Multi-colony model
Artificial bee colony algorithm
Migration strategy
Friedman test

ABSTRACT

This paper suggests a dynamic multi-colony multi-objective artificial bee colony algorithm (DMCMOABC) by using the multi-deme model and a dynamic information exchange strategy. In the proposed algorithm, K colonies search independently most of the time and share information occasionally. In each colony, there are S bees containing equal number of employed bees and onlooker bees. For each food source, the employed or onlooker bee will explore a temporary position generated by using neighboring information, and the better one determined by a greedy selection strategy is kept for the next iterations. The external archive is employed to store non-dominated solutions found during the search process, and the diversity over the archived individuals is maintained by using crowding-distance strategy. If a randomly generated number is smaller than the *migration rate* R, then an elite, defined as the intermediate individual with the maximum crowding-distance value, is identified and used to replace the worst food source in a randomly selected colony. The proposed DMCMOABC is evaluated on a set of unconstrained/constrained test functions taken from the CEC2009 special session and competition in terms of four commonly used metrics EPSILON, HV, IGD and SPREAD, and it is compared with other state-of-the-art algorithms by applying Friedman test on the mean of IGD. The test results show that DMCMOABC is significantly better than or at least comparable to its competitors for both unconstrained and constrained problems.

1. Introduction

In many real-world optimization applications, decision makers (DMs) often have to handle problems with multiple objectives that are conflicting to each other and should be optimized simultaneously, and they are usually called multi-objective optimization problems (MOPs) which are more difficult than one-objective ones since there is no single solution available for them but a set of Pareto-optimal solutions (PS) or non-dominated solutions that represent a trade-off among the objectives.

With the help of some useful techniques, such as objective weighting, method of distance functions or method of min–max formulation, a MOP can be transformed into a single objective problem, and then traditional mathematical programming methods can be applied to deal with it. However, these methods are usually restrained since some objectives may involve noise, discontinuity, concavity and uncertainty in their search space [1]. Alternatively, evolutionary algorithms (EAs) and swarm intelligence (SI) algorithms have been widely extended to find several members of the *PS* of the MOPs in a single run. Among them, Deb's NSGAII [2], Zitzler's SPEA2 [3], Zhang's MOEA/D [4], Tseng's MTS [5], Liu's DMOEADD [6], Liu's LiuLiAlgorithm [7], Kukkonen's GDE3 [8] and Akay's S-MOABC/NS [1] enjoyed more attention.

The island model or multiple-deme model is a very popular scheme used in many EA or SI algorithms [9]. In this model, the algorithms consist of several sub-populations which evolve independently most of the time and exchange members occasionally. The exchange of individuals is called *migration* and often controlled by some parameters [10]. Over the past decades, many algorithms based on island models have been proposed for single-objective optimization problems [11–16]. It was shown by the experimental results that these algorithms not only decreased the processing time but also performed a much broader search than single-population ones.

Recently, modeled as a Markov dynamic system, the island model was theoretically proved to be effective in solving single objective problems [17], and can also be applied to the multi-objective case. In fact, some island model based multi-objective

☆ This paper is supported by the National Natural Science Foundation of China (Grant nos. 61472143 and 61170081).
* Corresponding author at: School of Data Science and Computer, Sun Yat-sen University, Guangzhou 510275, PR China.

http://dx.doi.org/10.1016/j.asoc.2015.06.033
1568-4946/

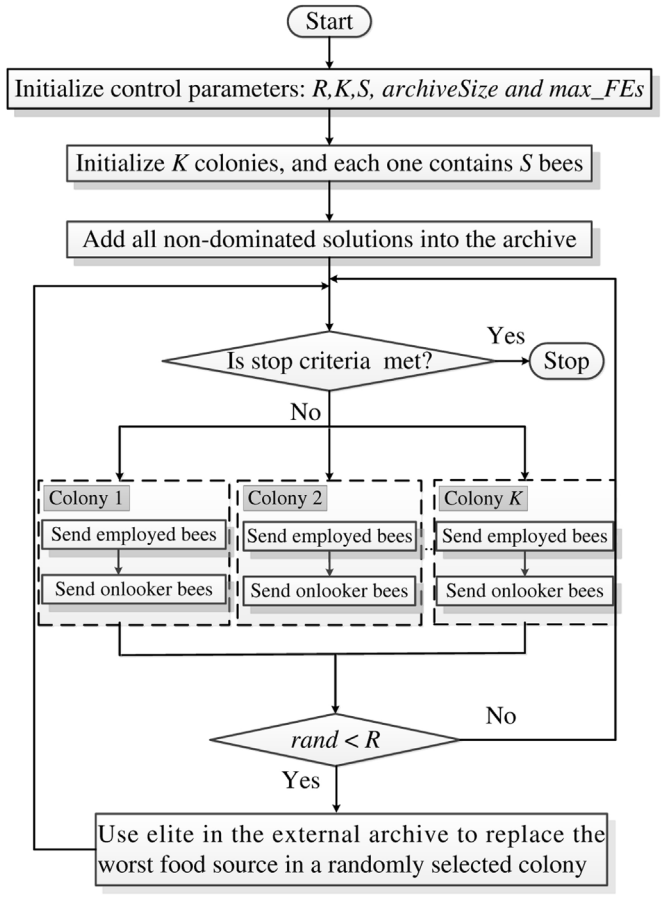


Fig. 1. The flow chart of the DMCMOABC algorithm.

evolutionary or SI-based algorithms have already been proposed. Montaño et al. [18] introduced a novel island model based multi-objective evolutionary algorithm: pMODE-LD+SS, where differential evolution (DE) operators were used and two parallel schemes were designed to improve effectiveness as well as efficiency. Cheshmehgaz et al. [19] proposed an effective multiple multi-objective evolutionary algorithms (MOEAs) by using a novel island model where a central island was assisted by multiple VIP islands and a sophisticated immigration strategy was designed to exchange information. Shang et al. [20] suggested a multi-population cooperative coevolutionary algorithm (MPCCA) for Capacitated Arc Routing Problem (CARP). In this algorithm, multiple subpopulations are used to search different objective subregions simultaneously and share individuals with their adjacent subpopulations cooperatively. Zhang et al. [21] proposed a multi-objective parallel evolution algorithm (MPEA) where the migration frequency is dynamically changed according to the diversity of the parent solutions. The algorithm was applied to effectively solve flight assignment problems. Some other island model based multi-objective algorithms can be found in Refs. [22–25].

Some of the above algorithms (e.g., [18,20,25]) divide the whole objective space into several sub-regions by a set of uniformly distributed weight vectors or direction vectors. Then each sub-region is assigned with a sub-population. However, it is not always easy to generate a set of uniformly distributed weight/direction vectors, especially when high-dimensional objective space is considered, and the generation of these vectors needs a recursive procedure [25]. The migration strategy used in most of these algorithms is static, i.e., a sub-population only exchanges individuals with its adjacent sub-populations in terms of the used topological

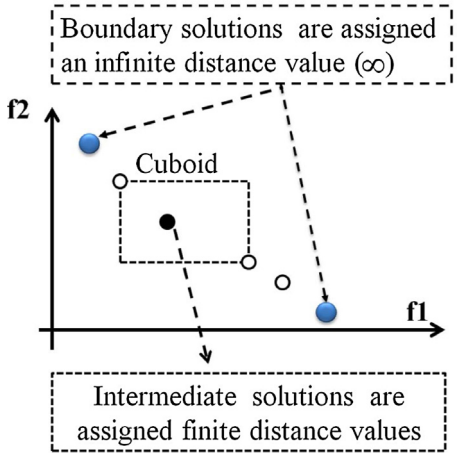


Fig. 2. Crowding distance calculation diagramed in a two-dimensional objective space.

structure. Meanwhile, some immigration strategies (e.g., [18,23–25], etc.) involve frequent re-division operations (central re-collection and re-distribution of all solutions from/to sub-populations [19]). This may be a main drawback since a re-division in distributed systems may affect the efficiency of the algorithm when a high level of parallelization is expected. Contrarily, this paper suggests a new island model where each island is equivalent and shares individuals with other islands dynamically. In this model, the migration is realized by using an elite pool (also known as the archive where non-dominated solutions are stored during the search process) and an island exchanges individuals with all other islands randomly. What is more, the proposed model involves no central re-collection, re-division and re-distribution operations.

Based on the proposed island model, a dynamic multi-colony multi-objective artificial bee colony algorithm (DMCMOABC) is suggested in this paper. For each island, the artificial bee colony (ABC) algorithm first proposed by Karaboga [26], is selected as the

Send employed bees in each colony

1. **For** $i \leftarrow 1$ **to** FN
2. For $\vec{X}_i$, randomly select a neighbor $\vec{X}_k$
3. Calculate the position $\vec{V}_i$ by using $\vec{X}_i$ and $\vec{X}_k$ with Eq. (4)
4. Work out $\vec{F}(\vec{V}_i) = (f_1(\vec{V}_i), f_2(\vec{V}_i), \ldots, f_M(\vec{V}_i))$
5. /* Select a candidate between $\vec{V}_i$ and $\vec{X}_i$ by using *domination comparator* */
6. **If** $(\vec{V}_i \prec \vec{X}_i)$ // $\vec{V}_i$ dominates $\vec{X}_i$
7. $\vec{X}_i \leftarrow \vec{V}_i$; $\vec{F}(\vec{X}_i) \leftarrow \vec{F}(\vec{V}_i)$
8. *archive*.add $(\vec{V}_i)$
9. **ElseIf** $(\vec{V}_i \triangleq \vec{X}_i)$ // $\vec{V}_i$ and $\vec{X}_i$ are non-dominated
10. **If** (*archive*.add $(\vec{V}_i) = true$)
11. $\vec{X}_i \leftarrow \vec{V}_i$; $\vec{F}(\vec{X}_i) \leftarrow \vec{F}(\vec{V}_i)$;
12. **Else**
13. Do nothing
14. **End If**
15. **Else** $(\vec{X}_i \prec \vec{V}_i)$ // $\vec{V}_i$ is dominated by $\vec{X}_i$
16. $\vec{V}_i$ is discarded
17. **End If**
18. **End For**
19. Call *computeFitness*() to get fitness value of each food source

End of Send employed bees

Fig. 3. The pseudo code of *send employed bees*.

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033

Send onlooker bees in each colony

1. Calculate selection probability using Eq. (5)
2. $i = 1, t = 1$
3. **While** $(t \leq FN)$
4. **If** $(rand < prob_i)$
5. $t++$
6. For $\vec{X}_i$, randomly select its neighbor $\vec{X}_k$
7. Calculate the position $\vec{V}_i$ using Eq. (4)
8. Work out $\vec{F}(\vec{V}_i) = (f_1(\vec{V}_i), f_2(\vec{V}_i), \ldots, f_M(\vec{V}_i))$
9. **If** $(\vec{V}_i \prec \vec{X}_i)$ // $\vec{V}_i$ dominates $\vec{X}_i$
10. $\vec{X}_i \leftarrow \vec{V}_i$; $\vec{F}(\vec{X}_i) \leftarrow \vec{F}(\vec{V}_i)$;
11. *archive*.add $(\vec{V}_i)$
12. **ElseIf** $(\vec{V}_i \triangleq \vec{X}_i)$
13. **If** (*archive*.add $(\vec{V}_i) = true$)
14. $\vec{X}_i \leftarrow \vec{V}_i$; $\vec{F}(\vec{X}_i) \leftarrow \vec{F}(\vec{V}_i)$;
15. **Else**
16. Do nothing
17. **End If**
18. **Else** $(\vec{X}_i \prec \vec{V}_i)$ // $\vec{V}_i$ is dominated by $\vec{X}_i$
19. $\vec{V}_i$ is discarded
20. **End If**
21. **End If**
22. $i++$
23. **If** $(i = FN + 1)$
24. $i \leftarrow 1$
25. **End If**
26. **End While**

End of Send onlooker bees

Fig. 4. The pseudo code of *Send onlooker bees*.

search engine. The ABC algorithm was very competitive to EAs and other global optimization algorithms [27,28], and was successfully applied to many practical problems, such as IIR filters [29], team orienteering problem [30], portfolio optimization problem [31], etc. In order to enhance the efficiency and effectiveness of the ABC algorithm, the parallelization approaches were studied in [32–34]. In recent years, the extension of ABC algorithm for multi-objective optimization has caught much attention from the evolutionary multi-objective (EMO) communities. Some representative works were available, such as MOABC [35], S-MOABC/NS [1], dMOABC [36], MOABC/D [37], etc. However, island models were not utilized in any of these algorithms. Very recently, Chen et al. [38] presented a multi-hive multi-objective bee algorithm (M^2OBA) for optimal power flow (OPF) in power systems by combining external archive, comprehensive learning, greedy selection, crowding distance, and cooperative search strategies. In the algorithm, the concept of Pareto dominance and comprehensive learning mechanism were used to determine the flight trajectory of a bee, and non-dominated solutions were kept in an external archive whose diversity was maintained by crowing distance strategies. By constructing a colony-level interaction topology and an information exchange strategy, the single population ABC has been extended to interacting multi-hive model.

In this work, the proposed DMCMOABC employs K colonies and each one contains S bees. In each colony, there are only two kinds of bees, i.e., the employed and the onlooker bees. These K colonies simultaneously optimize their own food sources by using neighbor information in one single run, and share information when a randomly generated number between 0 and 1 is smaller than a predefined control parameter R which is called the *migration rate*. In each migration, an elite, defined as the intermediate individual with the maximum crowding-distance value [2] in the external archive, is identified and used to replace the worst food source in a randomly selected colony. Here, the so-called worst food source refers to a solution with the maximum fitness value used in SPEA2 [3]. It is worthy to notice that the fitness value in SPEA2 is to be minimized. As we can see from the above information exchange mechanism, one colony may be able to share

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033

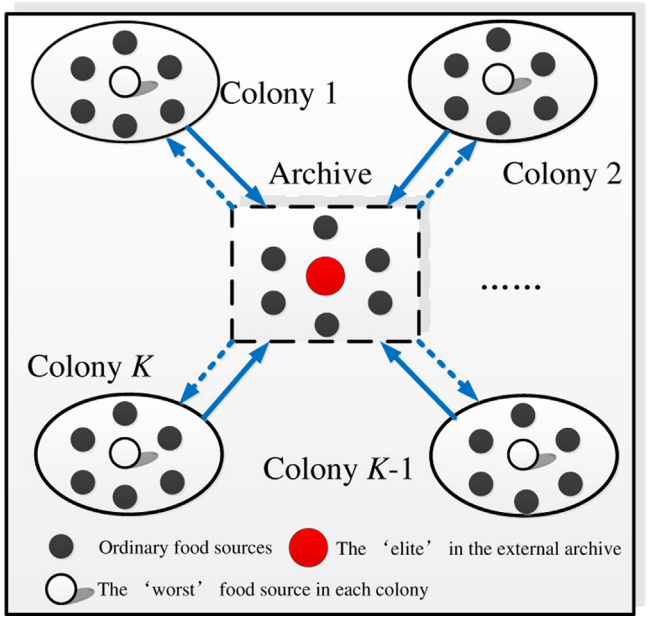


Fig. 5. The schematic diagram of information exchange used in DMCMOABC algorithm.

information with any other colonies since the elite in the archive could be found by any colony and the reception colony is also chosen randomly. In other words, the migration direction is not static. The DMCMOABC will be terminated when the number of function evaluations reaches to the maximum allowable function evaluations(max_FEs).

Although the M^{2}OBA and DMCMOABC share some commonalities, for example, they both use crowding distance to maintain the external archive and apply the greedy selection mechanism to decide which solution enters into the archive, they are distinguished in terms of the following aspects.

- The learning strategy used in each algorithm. In M^{2}OBA, the comprehensive learning strategy is used. Specifically, some dimensions of the new position for each food source $\bar{X}_i$ learn from a non-dominated solution EA_k which is randomly selected from the archive, while the remaining dimensions are produced by using another solution EA_l ($l \neq k$). In DMCMOABC, however, the basic learning strategy is adopted. That is, each bee generates the new position by using only neighboring information. Here, the neighbor of $\bar{X}_i$ is defined as a random food source $\bar{X}_k$ in the same colony. Although $\bar{X}_k$ is determined stochastically, it has to be different from $\bar{X}_i$. This learning mechanism ensures all colonies search independently in the whole search space. Since K colonies have their own search trajectories, it is good for the exploration process. In the basic learning strategy, only one dimension is modified for each food source that provides a nice exploitation in the sub-space where each colony locates. Thus, a balance between exploration and exploitation might be well kept.
- The information exchange mechanism adopted by each algorithm. In M^{2}OBA, the interaction of bees occurs in a two-level hierarchical topology. For the colony-level interaction, the ring or star topology is employed, while the topology of the individual-level is always using the star topology [38]. But, DMCMOABC uses a relatively simple information exchange strategy. First, an elite in the archive is identified by using crowding-distance value. Then, it will replace the worst food source in a randomly selected colony. Here, the elite in the archive is defined as the intermediate solution with maximum crowding-distance value, while the worst food source for each colony is the solution with the worst (or largest) fitness value used in SPEA2. The potential benefits of this mechanism lie in that it guarantees a better exploitation in the least crowded region where the elite locates. This is because the elite is used to replace the worst food source in a selected colony, and then it will go into a series of improvements executed by its employed and onlooker bees. Thus, more non-dominated solutions may be found in the sparsest region and the diversity over all archived solutions is naturally kept.
- The number of control parameters. Both algorithms employ some common control parameters, such as the number of colonies, the size of each colony, the size of the external archive, the maximum number of function evaluations and the parameter controlling migration rate. However, M^{2}OBA uses three additional parameters, such as the integer m controlling the dimensions to be changed in the updating equation of the food source, the size of the sending list K and the exchange factor δ ($0 < \delta < 1$) [38].

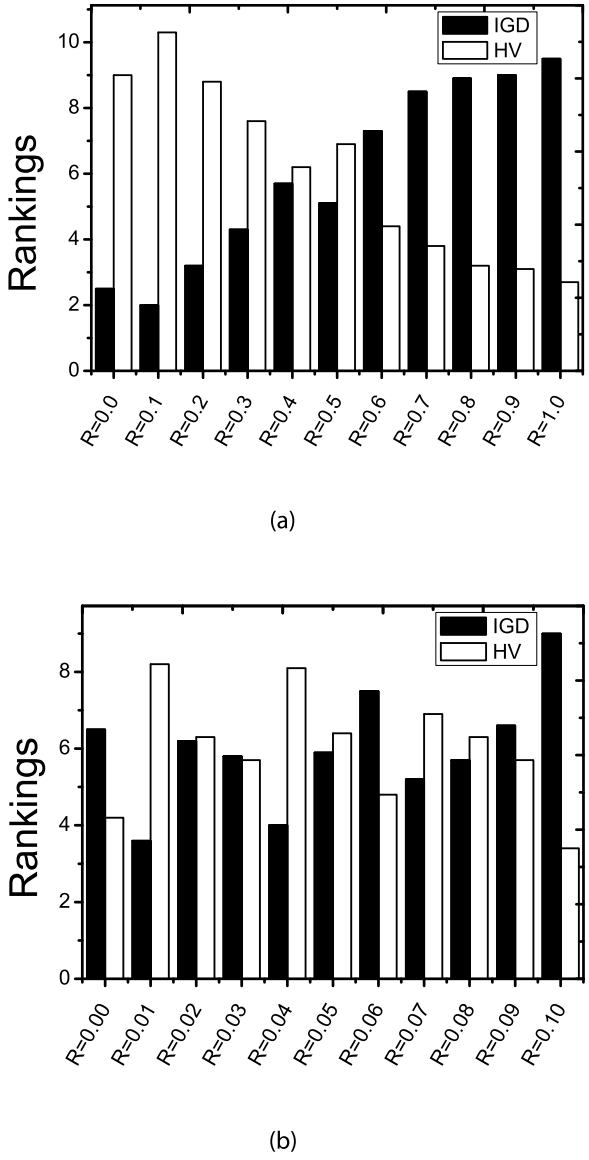
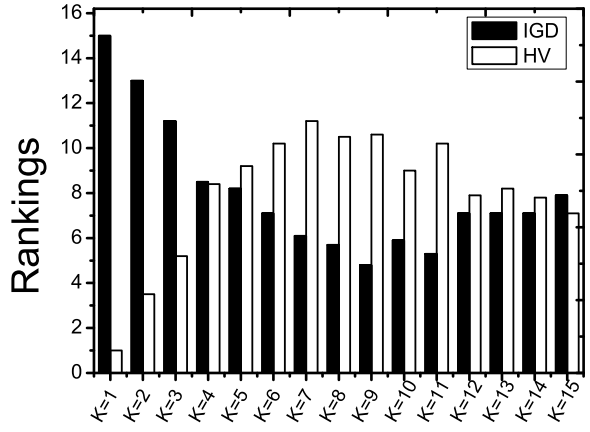
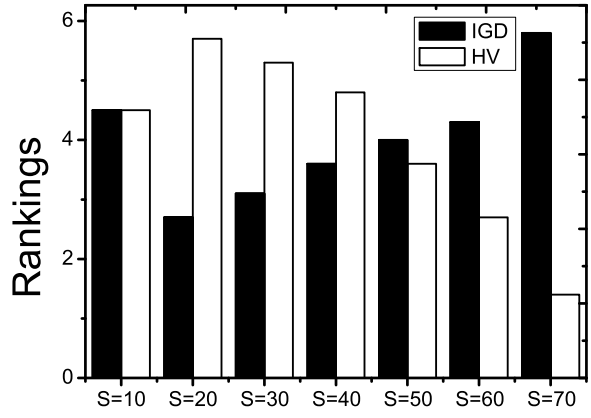


Fig. 6. Average rankings of HV and IGD when R is set to different values.

The rest of this paper is organized as follows. Section 2 gives some preliminaries on the multi-objective optimization and the basic artificial bee colony algorithm. Section 3 describes details of



(a)



(b)

Fig. 7. Average rankings of HV and IGD when K and S are set to different values.

the proposed DMCMOABC algorithm. The experimental study is presented in Section 4. Finally, Section 5 concludes the paper.

2. Preliminaries

2.1. Principles of multi-objective optimization

In general, there are two kinds of multi-objective problems (MOPs) in the optimization field: one is unconstrained problems and the other is constrained ones. A general unconstrained MOP consists of a number of conflicting objectives to be optimized simultaneously and a set of decision variables which are restricted to a limited interval. It can be formulated as follows:

$$\text{Minimize} \quad \vec{F}(\vec{X}) = (f_1(\vec{X}), f_2(\vec{X}), \ldots, f_M(\vec{X})), \tag{1}$$

where $\vec{X} = (x_1, x_2, \ldots, x_D)$ is called the decision vector, and $\Omega = \prod_{i=1}^{D}[LB_i, UB_i]$ is the decision space. Here, D is the dimension of the problem to be optimized; LB_i and UB_i are the lower and upper bounds of the ith decision variable, respectively. $\vec{F} : \vec{X} \to R^M$ consists of M ($M \geq 2$) real-valued objective functions, and $\Phi = \{\vec{F}(\vec{X}) | \ \vec{X} \in \Omega\}$ is known as the objective space.

If a MOP given by formula (1) is associated with a number of equality and inequality constraints, then it becomes a constrained multi-objective problem (CMOP) which can be formulated as below [38]:

$$\begin{aligned} \text{Minimize} \quad & \vec{F}(\vec{X}) = (f_1(\vec{X}), f_2(\vec{X}), \ldots, f_M(\vec{X})), \\ \text{subject to:} \quad & g_i(\vec{X}) = 0, i = 1, 2, \ldots, p, \\ & h_j(\vec{X}) \geq 0, j = 1, 2, \ldots, q, \\ & LB_d \leq x_d \leq UB_d, d = 1, 2, \ldots, D, \end{aligned} \tag{2}$$

where p and q are the numbers of equality and inequality constraints, respectively.

For MOP (1), let $\vec{X}$ and $\vec{V}$ be two vectors in the decision space, $\vec{X}$ is said to dominate $\vec{V}$ (denoted as $\vec{X} \prec \vec{V}$) if and only if $f_i(\vec{X}) \leq f_i(\vec{V})$ for all $i = 1, 2, \ldots, M$ and there exits at least one index j such that $f_j(\vec{X}) < f_j(\vec{V})$. If neither $\vec{X}$ dominates $\vec{V}$, nor $\vec{V}$ dominates $\vec{X}$, then $\vec{X}$ and $\vec{V}$ are non-dominated with each other denoted as $\vec{X} \triangleq \vec{V}$. A point $\vec{X}^*$ is called Pareto optimal if there is no $\vec{X}$ in the decision space dominating $\vec{X}^*$. Pareto optimal solutions are also called efficient, non-dominated, and non-inferior solutions. The set of all Pareto optimal solutions is called the Pareto set (PS), and all Pareto optimal objective vectors constitute the Pareto front (PF), namely, $PF = \{\vec{F}(\vec{X}) | \vec{X} \in PS\}$.

2.2. The artificial bee colony algorithm

The ABC algorithm is a relatively new optimization algorithm which mimics the foraging behavior of honey bees [1,26]. In the algorithm, each employed bee is associated with a particular food source and explores a new position around that food source by learning from its neighbors. The onlooker bees then make a decision to choose which food source to exploit by using a probability-based selection process according to the quality of food sources provided by employed bees. The bees of the exhausted food sources become a scout and start to search a random food source. In ABC algorithm, the whole colony consists of equal number of employed and onlooker bees and each food source corresponds to only one employed bee [39].

In ABC algorithm, *limit* is an important control parameter that is used to determine the exhausted food sources. A food source that cannot be improved for *limit* trials is abandoned. The algorithm starts with randomly generating some food source positions that correspond to the solutions in the decision space. And the counters storing the numbers of trials for solutions are set to 0. After initialization, the population is subjected to repeated cycles of the search processes of the employed bees, onlooker bees and scout bees [40]. The algorithm will be terminated once a maximum number of function evaluations (max_FEs) is reached.

For each food source $\vec{X}_i$, its employed bee will find a neighboring solution $\vec{V}_i$ depending on local information (visual information) in her memory. Then, $\vec{X}_i$ and $\vec{V}_i$ are compared with each other. If $\vec{V}_i$ outperforms $\vec{X}_i$, then the employed bee memories the new position and forgets the old one. Otherwise, the original food source is kept in the memory. If $\vec{X}_i$ can not be improved, then its counter holding the number of trials will increase by 1, otherwise, the counter is reset to 0 [41].

After every food source is optimized by its employed bee, the onlooker bees will choose food sources using the roulette wheel selection scheme based on the selection probability. The larger the fitness value, the higher the selection probability. Once a food source has been chosen, the onlooker bees will improve it as employed bees do.

In each cycle, the algorithm checks to find if there is any food source to be abandoned when the employed and onlooker bees complete their search. If the value of the counter of a food source

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033

Table 1
The statistics of EPSILON obtained by DMCMOABC over 30 runs.

Test function	Best	Worst	Median	Mean	SD
UF1	0.013516	0.022818	0.017497	0.017682	0.002571
UF2	0.010462	0.037354	0.015887	0.018598	0.007437
UF3	0.071641	0.255715	0.148660	0.151174	0.041419
UF4	0.032113	0.059275	0.039509	0.039831	0.005047
UF5	0.069297	0.173858	0.106295	0.104604	0.020220
UF6	0.053077	0.097291	0.063702	0.065632	0.010302
UF7	0.024408	0.054643	0.037824	0.038068	0.007751
UF8	0.098940	0.230489	0.130655	0.142389	0.033002
UF9	0.078222	0.180383	0.099226	0.106402	0.023512
UF10	0.228529	0.568752	0.355023	0.367040	0.094393
CF1	0.024054	0.076238	0.035633	0.037985	0.010876
CF2	0.012111	0.064653	0.019023	0.025894	0.016522
CF3	0.069201	0.120004	0.091722	0.092370	0.013487
CF4	0.019062	0.057829	0.031179	0.031450	0.008986
CF5	0.023505	0.079729	0.049434	0.049423	0.013957
CF6	0.013653	0.093085	0.034177	0.044388	0.021850
CF7	0.032185	0.080402	0.050807	0.052050	0.013688

Table 2
The statistics of HV obtained by DMCMOABC over 30 runs.

Test function	Best	Worst	Median	Mean	SD
UF1	0.660063	0.657418	0.659294	0.659161	0.000538
UF2	0.661056	0.658455	0.660532	0.660359	0.000614
UF3	0.606611	0.516884	0.573863	0.573470	0.017666
UF4	0.293560	0.287794	0.291522	0.291432	0.001164
UF5	0.424067	0.388401	0.403854	0.405504	0.009513
UF6	0.391320	0.368956	0.381770	0.382482	0.005518
UF7	0.492383	0.486399	0.490120	0.489945	0.001341
UF8	0.394648	0.363178	0.385482	0.384044	0.007510
UF9	0.736801	0.716691	0.729823	0.728194	0.005435
UF10	0.319223	0.252586	0.297996	0.295566	0.012387
CF1	0.451116	0.440822	0.447018	0.446538	0.002471
CF2	0.611683	0.604342	0.609275	0.608924	0.001700
CF3	0.225928	0.185427	0.204750	0.204820	0.009395
CF4	0.506589	0.500010	0.503986	0.503773	0.001853
CF5	0.503163	0.482316	0.496367	0.495763	0.004669
CF6	0.634173	0.616193	0.632720	0.632118	0.003148
CF7	0.626349	0.606432	0.620307	0.619598	0.004218

exceeds *limit*, then it will be abandoned and replaced with a random solution produced by the scout. In basic ABC, only one scout bee is allowed in each cycle.

3. The proposed DMCMOABC algorithm

The proposed algorithm uses a multi-colony model and an information exchange strategy to make the algorithm useful. The algorithm maintains the following control parameters: the migration rate (R), the number of colonies (K), the size of each colony (S), the size of the external archive (*archiveSize*) and the maximum number of function evaluations (max_FEs). Similar to single objective ABC algorithm, the number of food sources (denoted as FN) is fixed to half of the colony size, i.e., $FN = S/2$. The effects of parameters R, K, S will be experimentally studied later in Section 4.2. The flow chart of the algorithm is shown in Fig. 1. Main components of DMCMOABC are initialization, send employed bees, send onlooker bees, archive maintenance, and information

Table 3
The statistics of IGD obtained by DMCMOABC over 30 runs.

Test function	Best	Worst	Median	Mean	SD
UF1	0.004814	0.006670	0.005252	0.005333	0.000361
UF2	0.004189	0.006915	0.004778	0.004959	0.000704
UF3	0.034518	0.088183	0.053424	0.054412	0.010824
UF4	0.024070	0.028128	0.025089	0.025383	0.000909
UF5	0.039333	0.063891	0.053345	0.052676	0.006581
UF6	0.024834	0.062801	0.037692	0.039270	0.008339
UF7	0.004902	0.008920	0.006362	0.006539	0.000898
UF8	0.059719	0.074901	0.065685	0.066528	0.003298
UF9	0.033198	0.040923	0.036825	0.036804	0.001631
UF10	0.094768	0.136533	0.111567	0.111888	0.010661
CF1	0.020585	0.031188	0.023468	0.023948	0.002582
CF2	0.002400	0.048464	0.003905	0.009023	0.013827
CF3	0.044597	0.073777	0.061739	0.061316	0.007371
CF4	0.007015	0.011801	0.008805	0.008906	0.001236
CF5	0.009244	0.026996	0.013958	0.015169	0.004006
CF6	0.005611	0.028026	0.007456	0.008486	0.004038
CF7	0.010188	0.024047	0.015228	0.015274	0.002950

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033

Table 4
The statistics of SPREAD obtained by DMCMOABC over 30 runs.

Test function	Best	Worst	Median	Mean	SD
UF1	0.345926	0.642882	0.440152	0.452176	0.056196
UF2	0.184939	0.308447	0.246230	0.244891	0.030946
UF3	0.775819	1.076424	0.979322	0.982897	0.066581
UF4	0.515510	0.716345	0.661025	0.650423	0.051974
UF5	0.413256	0.736677	0.587696	0.590902	0.082529
UF6	0.724949	1.040922	0.885697	0.884847	0.078356
UF7	0.286238	0.546328	0.400303	0.406621	0.069065
UF8	0.572101	0.733455	0.667645	0.669816	0.040753
UF9	0.820136	1.008253	0.905609	0.903536	0.044956
UF10	0.834864	1.087500	0.964753	0.958296	0.058926
CF1	0.161777	1.159021	0.713708	0.681339	0.223872
CF2	0.975971	1.254019	1.092057	1.091611	0.072299
CF3	0.765496	1.086121	0.983518	0.968002	0.080803
CF4	0.676324	0.991046	0.830124	0.824903	0.083497
CF5	0.977664	1.223084	1.108729	1.122022	0.066263
CF6	0.288225	0.532367	0.379578	0.376643	0.048531
CF7	0.876067	1.235319	0.986519	0.994491	0.076517

exchange. The following subsections will describe them in more details.

3.1. Initialization

In initialization phase, *FN* food sources are randomly generated in the decision space for each colony. The position of the *i*th food source $\bar{X}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ is given by the following equation.

$$x_{id} = LB_d + r \cdot (UB_d - LB_d), \tag{3}$$

where r is a random number uniformly distributing over interval [0,1]; LB_d and UB_d are the lower and upper bounds of the dth decision variable, respectively. Here $d = 1, 2, \ldots, D$, where D is the dimension of the problem to be optimized. For the ith food source, the global variable $trial_i$ is used to hold the number of unsuccessful trials and it is initialized as 0. After all food sources have been produced, those non-dominated individuals will be added into the external archive. In our algorithm, all colonies share the same archive and the maintenance of it is depicted in the next subsection.

3.2. Archive maintenance

The external archive is a common technique employed in many multi-objective evolutionary or swarm-based algorithms which is used to keep non-dominated solutions found by an algorithm during the search process. An effective maintenance method is often required since the size of an archive is usually fixed. In DMCMOABC, a candidate $\vec{S}$ is directly discarded if there exists an identical individual or $\vec{S}$ is dominated by any member in the archive. And it will be added into the archive if and only if one of the following conditions is satisfied.

(a) The archive is empty.
(b) $\vec{S}$ dominates any member in the archive.
(c) $\vec{S}$ is non-dominated with each archived member, and the archive is not full.
(d) $\vec{S}$ is non-dominated with each archived member, but the archive is full.

An important task for the usage of the external archive is to keep a proper diversity over a set of non-dominated solutions. In case (a)

Table 5
The mean of IGD values obtained by each algorithm on unconstrained test functions (UF1–UF10).

Test function	DMCMOABC	dMOABC	MOEA/D	S-MOABC/NS	MTS
UF1	**5.333E−03**	5.941E−03	**4.350E−03**	2.537E−02	6.467E−03
UF2	**4.959E−03**	**5.673E−03**	6.790E−03	8.963E−03	6.158E−03
UF3	5.441E−02	5.085E−02	**7.420E−03**	2.680E−01	5.311E−02
UF4	**2.538E−02**	3.520E−02	6.385E−02	3.829E−02	**2.356E−02**
UF5	5.268E−02	1.003E−01	1.807E−01	3.964E−01	**1.489E−02**
UF6	**3.927E−02**	1.107E−01	**5.870E−03**	3.347E−01	5.918E−02
UF7	**6.539E−03**	1.015E−02	**4.440E−03**	1.639E−02	4.079E−02
UF8	**6.653E−02**	1.428E−01	**5.840E−02**	1.444E−01	1.125E−01
UF9	**3.680E−02**	1.143E−01	7.896E−02	2.188E−01	1.144E−01
UF10	**1.119E−01**	2.188E−01	4.742E−01	5.022E−01	**1.531E−01**

Test function	DMOEADD	LiuLiAlgorithm	GDE3	NSGAII	SPEA2
UF1	1.038E−02	7.850E−03	5.342E−03	1.038E−01	1.248E−01
UF2	6.791E−03	1.230E−02	1.195E−02	3.858E−02	4.522E−02
UF3	3.337E−02	**1.498E−02**	1.064E−01	1.581E−01	1.942E−01
UF4	4.269E−02	4.350E−02	2.651E−02	5.123E−02	5.205E−02
UF5	3.145E−01	1.619E−01	**3.928E−02**	8.010E−02	3.802E−01
UF6	6.673E−02	1.756E−01	2.509E−01	2.580E−01	2.749E−01
UF7	1.032E−02	7.301E−03	2.523E−02	1.854E−01	1.699E−01
UF8	6.842E−02	8.235E−02	2.486E−01	2.213E−01	1.995E−01
UF9	**4.897E−02**	9.392E−02	8.248E−02	3.159E−01	2.040E−01
UF10	3.221E−01	4.469E−01	4.333E−01	3.855E−01	3.248E−01

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033

Table 6
The mean of IGD values obtained by each algorithm on constrained test functions (CF1–CF7).

Test function	DMCMOABC	dMOABC	MTS
CF1	2.395E−02	3.953E−02	1.919E−02
CF2	9.023E−03	7.332E−03	2.678E−02
CF3	**6.132E−02**	1.096E−01	1.045E−01
CF4	8.906E−03	**7.268E−03**	1.110E−02
CF5	**1.517E−02**	1.805E−02	2.078E−02
CF6	**8.486E−03**	**7.030E−03**	1.617E−02
CF7	**1.527E−02**	3.402E−02	2.470E−02

Test function	DMOEADD	LiuLiAlgorithm	GDE3
CF1	**1.131E−02**	**8.590E−04**	2.940E−02
CF2	**2.100E−03**	**4.203E−03**	1.598E−02
CF3	**5.631E−02**	1.829E−01	1.275E−01
CF4	**6.995E−03**	1.423E−02	7.991E−03
CF5	**1.577E−02**	1.097E−01	6.800E−02
CF6	1.502E−02	1.395E−02	6.200E−02
CF7	**1.905E−02**	1.045E−01	4.169E−02

and (c), $\vec{S}$ is directly put into the archive. And the members dominated by $\vec{S}$ will be removed from the archive in case (b) so as to ensure all archived solutions are non-dominated with each other. When the archive is full and a candidate $\vec{S}$ is to be added (case (d)), the crowding distances of all members together with $\vec{S}$ are re-calculated, and the one with the minimum crowding distance value will be removed from the archive. The crowding distance introduced in Ref. [2] is used to estimate the density of solutions in the external archive. A solution with a larger crowding distance value locates in a less crowded region, indicating that this solution has more potential to be exploited in terms of the population diversity. Hence, solutions with a larger crowding distance value are preferred.

The computation of crowding distance needs the following steps: (1) sort the solutions in the external archive according to each objective function value in ascending order of magnitude; (2) then for each objective, an infinite distance value is assigned to the boundary solutions, i.e., solutions with smallest and largest function values, and all other intermediate solutions are assigned a distance value which is equal to the absolute normalized difference in the function values of two adjacent solutions; (3) the above calculation is repeated with other objective functions. The overall crowding distance value is calculated as the sum of individual distance values corresponding to each objective [2].

It should be noted here that each objective function is normalized before calculating the crowding distance. Fig. 2 shows the crowding distance calculation in a two-dimensional objective space. The crowding distance of the solution marked with solid black circle is the average side length of the cuboid (shown with a dashed box).

Table 7
Average rankings of each method by applying Friedman test on unconstrained problems.

Algorithm	Ranking
DMCMOABC	2.2
dMOABC	4.35
MOEA/D	3.85
S-MOABC/NS	8.1
MTS	3.95
DMOEADD	4.65
LiuLiAlgorithm	5.5
GDE3	5.7
NSGAII	8.1
SPEA2	8.6

3.3. Send employed bees

In this phase, an employed bee is sent to its corresponding food source to explore a temporary position by using neighbor information. For each colony, the pseudo-code of *send employed bees* is given in Fig. 3. Given food source $\vec{X}_i$, its temporary position $\vec{V}_i$ is calculated through the following equation.

$$v_{id} = x_{id} + \phi_{id} \cdot (x_{id} - x_{kd}), \tag{4}$$

where d is a random integer representing the index of the dimension to be changed; ϕ_{id} is a real number distributed uniformly over $[-1, 1]$; k is an integer randomly chosen from the set $\{1, 2, \ldots, FN\}$. Although k is determined stochastically, it has to be different from i. Usually, we call $\vec{X}_k$ the neighbor of $\vec{X}_i$.

After the generation of $\vec{V}_i$, it is then evaluated and compared with $\vec{X}_i$ by using *domination comparator*. Next, a greedy selection mechanism is applied to $\vec{V}_i$ and $\vec{X}_i$ so as to determine which one to be kept for the next iterations: if $\vec{V}_i$ dominates $\vec{X}_i$ ($\vec{V}_i \prec \vec{X}_i$), then replace $\vec{X}_i$ by $\vec{V}_i$ and update objective values, finally try to put $\vec{V}_i$ into the archive (see lines 7 and 8 in Fig. 3); if they are non-dominated with each other ($\vec{V}_i \triangleq \vec{X}_i$), and $\vec{V}_i$ can be successfully added into the archive (i.e. *archive*.add ($\vec{V}_i$) = *true*), then $\vec{X}_i$ is also replaced by $\vec{V}_i$. In the case where $\vec{V}_i$ can not be added, the procedure will do nothing (line 13 in Fig. 3)); if $\vec{V}_i$ is dominated by $\vec{X}_i$, $\vec{X}_i$ is kept as the candidate and $\vec{V}_i$ will be discarded.

Finally, the function *computeFitness*() is called to obtain the fitness value of each food source. In DMCMOABC, the fitness assignment method is the same as in SPEA2. That is, the fitness $fit(\vec{X}_i)$ contains two parts: one is the raw fitness value $R(\vec{X}_i)$, and the other is the density $D(\vec{X}_i)$. Mathematically, $fit(\vec{X}_i) = R(\vec{X}_i) + D(\vec{X}_i)$. The raw fitness of $\vec{X}_i$ is determined by summing the strengths (representing the number of solutions $\vec{X}_i$ dominates) of its dominators in both archive and the colony, while the density is incorporated to discriminate between solutions having identical raw fitness values [42]. It's worth noting that the fitness used in DMCMOABC is to be minimized.

Table 8
Adjusted p-values obtained through Friedman test on unconstrained test functions (DMCMOABC is the control method).

i	Algorithm	Unadjusted p	p_{Bonf}	p_{Holm}	$p_{Hochberg}$	p_{Hommel}	$p_{Holland}$	p_{Rom}	p_{Finner}	p_{Li}
1	SPEA2	0.000002	0.000021	0.000021	0.000021	0.000021	0.000021	0.000020	0.000021	0.000003
2	S-MOABC/NS	0.000013	0.000118	0.000105	0.000092	0.000092	0.000105	0.000088	0.000059	0.000017
3	NSGAII	0.000013	0.000118	0.000105	0.000092	0.000092	0.000105	0.000088	0.000059	0.000017
4	GDE3	0.009740	0.087661	0.058441	0.058441	0.048701	0.057036	0.055568	0.021782	0.012380
5	LiuLiAlgorithm	0.014801	0.133209	0.074005	0.074005	0.074005	0.071847	0.070378	0.026484	0.018693
6	DMOEADD	0.070382	0.633437	0.281528	0.222994	0.222994	0.253176	0.222994	0.103693	0.083057
7	dMOABC	0.112313	1.010821	0.336940	0.222994	0.222994	0.300514	0.222994	0.142021	0.126291
8	MTS	0.196198	1.765779	0.392395	0.222994	0.222994	0.353902	0.222994	0.217845	0.201600
9	MOEA/D	0.222994	2.006945	0.392395	0.222994	0.222994	0.353902	0.222994	0.222994	0.222994

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033

Table 9
Average rankings of each method by applying Friedman test on constrained problems.

Algorithm	Ranking
DMCMOABC	2.5714
dMOABC	3.2857
MTS	4.1429
DMOEADD	1.8571
LiuLiAlgorithm	4.2857
GDE3	4.8571

3.4. Send onlooker bees

In each colony, onlooker bees will select food sources to exploit according to the quality of each food source advised by employed bees. The selection probability for the ith food source is given by Eq. (5).

$$prob_i = 1 - \frac{fit(\vec{X}_i)}{\sum_{m=1}^{FN} fit(\vec{X}_m)}, \tag{5}$$

where $fit(\vec{X}_m)$ returns the fitness value of $\vec{X}_m$. As shown in Eq. (5), a food source with lower fitness value will be assigned a larger selection probability, thus a higher chance it is chosen by onlooker bees.

The pseudo-code of *send onlooker bees* is shown in Fig. 4. In this phase, a roulette wheel method is employed by onlooker bees to choose food sources based on the selection probabilities [43]. Once a food source is selected, the corresponding onlooker bee will try to optimize it just like the employed bee does. A new position is generated by Eq. (4), and is compared with the original food source by using domination comparator. The better one is kept for further improvements in the next iterations. The above procedure will repeat *FN* times and more potential food sources (with higher selection probability) may be exploited more than once.

3.5. Information exchange

In DMCMOABC algorithm, K colonies search simultaneously in the whole decision space, and non-dominated solutions found by them are put into the same archive. If there is no information exchange among the colonies, then it will be a multi-deme algorithm with these colonies searching in parallel. Hence, an information exchange strategy is introduced here to enable the migration among colonies. The schematic diagram of information exchange strategy is provided in Fig. 5. First, an elite in the archive is identified by using crowding-distance value (see Section 3.2). Then, it will replace the worst food source in a randomly selected colony. Here, the elite in the archive is defined as the intermediate solution with maximum crowding-distance value, while the worst food source for each colony is the solution with the largest fitness value used in SPEA2 (see Section 3.3).

In Fig. 5, the full lines indicate that non-dominated solutions found by each colony are added into the archive, while the dotted lines mean that the elite just replace the worst food source from only one randomly chosen colony. As we can see from this mechanism that one colony may communicate with all other colonies since the elite could be found by any colony and the reception colony is also chosen stochastically. That is to say, we here employ a dynamic information exchange strategy.

The elite is located in the least crowded region in the objective space, and it is used to substitute the worst food source in a picked colony. Hence, the region around elite will be exploited more nicely since it may be improved by employed or onlooker bees through some iterations. As a result, the final non-dominated solutions are expected to achieve better qualities in terms of both convergence and diversity.

Colonies in DMCMOABC evolve separately most of the time and exchange individuals occasionally. The information exchange phase is controlled by the *migration rate R*. As pointed out in [10] that it is an important parameter in multi-deme algorithms. The effect of R on the quality of returned solutions will be shown in Section 4.5 with some simulation results.

3.6. Constraint-handling method

To deal with CMOP (2), a constraint-handling method is imported from NAGA-II [2], which simply modifies *domination comparator* between two solutions $\vec{X}_i$ and $\vec{X}_j$ to *constrained-domination comparator*. A solution $\vec{X}_i$ is said to constrained-dominate solution $\vec{X}_j$, if and only if one of the following conditions is true.

- Solution $\vec{X}_i$ is feasible (without violating any constraint), while solution $\vec{X}_j$ is not.
- Both $\vec{X}_i$ and $\vec{X}_j$ are infeasible, but solution $\vec{X}_i$ has a smaller overall constraint violation.
- Both $\vec{X}_i$ and $\vec{X}_j$ are feasible, and solution $\vec{X}_i$ dominates $\vec{X}_j$.

In the comparison of two solutions $\vec{X}_i$ and $\vec{X}_j$, there are at most three cases to be considered: (1) both are feasible; (2) one is feasible, while the other is not; (3) both are infeasible. For the first case, the

Table 10
Adjusted p-values obtained through Friedman test on constrained test functions (DMOEADD is the control method).

i	Algorithm	Unadjusted p	p_{Bonf}	p_{Holm}	$p_{Hochberg}$	p_{Hommel}	$p_{Holland}$	p_{Rom}	p_{Finner}	p_{Li}
1	GDE3	0.00270	0.013499	0.013499	0.013499	0.013499	0.013426	0.012837	0.013426	0.005117
2	LiuLiAlgorithm	0.015158	0.075792	0.060634	0.060634	0.045475	0.059269	0.057815	0.037466	0.028066
3	MTS	0.022271	0.111355	0.066813	0.066813	0.066813	0.065336	0.066813	0.037466	0.040698
4	dMOABC	0.153127	0.765637	0.306255	0.306255	0.306255	0.282807	0.306255	0.187595	0.225826
5	DMCMOABC	0.475051	2.375253	0.475051	0.475051	0.475051	0.475051	0.475051	0.475051	0.475051

Table 11
The runtime of some algorithms (in milliseconds, and the best values are shown in boldface).

Fun.	DMCMOABC	dMOABC	MOEA/D	NSGAII	SPEA2
UF1	4.27E+04	7.87E+04	**1.39E+04**	1.66E+04	6.91E+04
UF2	4.88E+04	9.62E+04	**1.49E+04**	1.73E+04	7.96E+04
UF3	**1.03E+04**	1.69E+04	1.76E+04	2.07E+04	6.49E+04
UF4	4.06E+04	7.81E+04	**1.54E+04**	1.69E+04	7.86E+04
UF5	**5.16E+03**	5.82E+03	1.50E+04	1.79E+04	4.95E+04
UF6	**5.52E+03**	5.61E+03	1.51E+04	1.78E+04	6.12E+04
UF7	4.23E+04	8.16E+04	**1.39E+04**	1.67E+04	8.11E+04

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033

domination comparator can be directly applied, and for the second one, the feasible one is preferred. For the third case, the overall constraint violations need to be calculated for both solutions. The overall constraint violation of solution $\vec{X}$ is given by the following equation [44].

$$CV(\vec{X}) = \sum_{i=1}^{p} |g_i(\vec{X})| + \sum_{j=1}^{q} \langle h_j(\vec{X}) \rangle, \quad (6)$$

where the bracket operator $\langle \alpha \rangle$ returns the negative value of α, if $\alpha < 0$; otherwise, it returns zero. The solution with smaller overall constraint violation is preferred. This constraint-handling method is very simple and proved experimentally to be quite effective when used in real-coded NSGA-II and NSGA-III algorithms. Thus, it could be an alternative approach to handle constraints in other multi-objective evolutionary or swarm-based algorithms.

4. Experimental study

4.1. Performance metrics and test functions

The DMCMOABC is evaluated and compared with other state-of-the-art algorithms in terms of four commonly used metrics on seventeen test functions taken from the CEC2009 special session and competition [45]. The four metrics are EPSILON [46], HV [46], IGD [45] and SPREAD [46]. In multi-objective optimization, the convergence and diversity of the approximated front yielded by an algorithm are usually used to assess the performance of a meta-heuristic. In the above four metrics, EPSILON and SPREAD are intended to measure the convergence and diversity, respectively. While HV and IGD take both criteria into consideration [46]. It should be noted here that all metrics but HV are to be minimized.

For the purpose of performance comparison, the DMCMOABC, together with other state-of-the-art multi-objective algorithms, is evaluated on test functions UF1–UF10 and CF1–CF7. The definition and mathematical representation of them are available in the technical report of CEC2009 [45]. The CEC2009 test suite provides a set of complicated functions: UF1–UF7 are unconstrained two-objective problems, and UF8–UF10 are unconstrained three-objective ones, while the CF1–CF7 represent constrained problems with two objectives. Pareto fronts of these test problems have different characteristics, for example, some of them are convex while the other ones are concave, or a part of them are continuous but the others are discontinuous [35]. Usually, a successful multi-objective algorithm can well handle problems with different types of Pareto fronts.

4.2. Parameter selection

Apart from two conventional control parameters *archiveSize* and *max_FEs*, there are three additional parameters R, K and S that should be tuned beforehand. To do this, the DMCMOABC is evaluated on the UF1–UF10 test functions with *max_FEs* being 300,000, and the *archiveSize* is set to 100 and 150 for two- and three-objective functions, respectively. To investigate the best value of the control parameter R, the following experiment is designed. First, K and S are assigned with initial values 10 and 20, respectively. Then R is varied from 0.0 to 1.0 with a step size 0.1. For each value of R, DMCMOABC is run 15 times one each function and the mean values of HV and IGD are calculated. Finally, by applying Friedman test [46,47] to the recorded mean values, the average rankings for IGD and HV under each parameter configuration are obtained, which are shown in (a) of Fig. 6. The Friedman test assumes that the smaller the average ranking is, the better the computed fronts will be. This

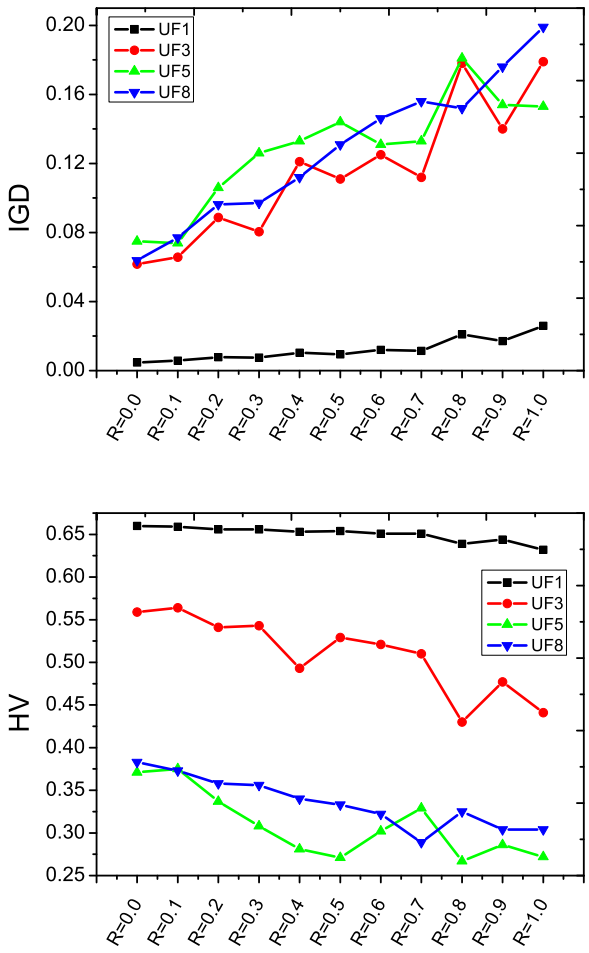


Fig. 8. The broken-line graphs of metrics when parameter R is varied from 0.0 to 1.0.

is true in all cases but HV since it is a metric to be maximized. In this case, it should be interpreted accordingly.

The bar graph in (a) of Fig. 6 shows that the performance tends to decrease in terms of both HV and IGD with the increment of R when its value is larger than 0.1. Compared with the value 0.0, $R = 0.1$ also gives better results. Considering the above experimental results, we may carefully conclude that the information exchange indeed improves the performance of the DMCMOABC algorithm, but it prefers a lower *migration rate*. It is suggested by the simulation results that the best value of R could be found in the interval (0.0, 0.1].

Another experiment is conducted to further study the parameter selection of R by changing its value from 0.0 to 0.1 with a smaller step size 0.01. Average rankings in different cases are presented in the form of a bar graph as shown in (b) of Fig. 6. It is revealed by the graph that values of R between 0.0 and 0.1 produce similar results when taking an overall consideration of HV and IGD. However, the performance is relatively better if R is set to 0.01 since DMCMOABC achieves the best rankings in terms of both metrics under this parameter setting. Thus, $R = 0.01$ is suggested for general usage in our algorithm.

To determine the best value of K, a similar experiment is conducted by varying the value of K from 1 to 15 with a step size 1. In this experiment, R and S are set to 0.01 and 20, respectively. It is observed from the experiment results, shown in (a) of Fig. 7, that the DMCMOABC performs better when K is set to a value between 6 and 11. Compared against other considered values, these ones could provide relatively better average rankings in terms of both HV and

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033

IGD. The bar graph also reveals that $K = 9$ gives the best and second best rankings in terms of IGD and HV, respectively. Therefore, this value is recommended for common usage. Finally, it is found from the graph that all values of K larger than 1 give better performance than $K = 1$, indicating a real performance improvement with the introduction of the multi-colony model. The effect of this model will be shown with more experimental results in Section 4.5.

The results of parameter selection experiment on S are presented in (b) of Fig. 7. In this experiment, $R = 0.01$ and $K = 9$. From the bar graph we find that the best performance is obtained when S is set to 20.

As the summary of this subsection, we list all suggested values of the control parameters as below: $R = 0.01$, $K = 9$ and $S = 20$. These values are determined by experiments where two parameters are fixed and the third one is varied over an interval with a step size. The average rankings obtained by Friedman test on HV and IGD are used as the criterion for parameter selection.

4.3. Computational results and performance comparison

Under parameter configurations described in the previous subsection, DMCMOABC is evaluated on seventeen test functions UF1–UF10 and CF1–CF7. On each function, the algorithm is independently run 30 times and terminated when the number of function evaluations reaches 300,000 for each run. The *archiveSize* is set to 100 and 150 for problems with two and three objectives, respectively. The statistics (the best, worst, median, mean and standard deviation) of each metric over 30 runs are computed and listed in Tables 1–4. The metrics EPSILON, HV and SPREAD are calculated by jMetal[1] software package, an object-oriented Java-based framework for multi-objective optimization with meta-heuristics, which was developed by Durillo and Nebro [46,48]. The IGD is a widely used metric for performance evaluation and comparison, and it is calculated exactly the same as described in the technical report of CEC2009 [45].

The DMCMOABC algorithm is compared with other state-of-the-art multi-objective algorithms in terms of the mean value of IGD. These algorithms include dMOABC [36], MOEA/D [4], S-MOABC/NS [1], MTS [5], DMOEADD [6], LiuLiAlgorithm [7], GDE3 [8], NSGAII [2] and SPEA2 [3]. The computational results for unconstrained and constrained test functions are provided in Tables 5 and 6, respectively. In these tables, the data of algorithms MOEA/D, S-MOABC/NS, MTS, DMOEADD, LiuLiAlgorithm and GDE3 are directly taken from original works, while those of NSGAII and SPEA2 are obtained by running jMetal software package. The DMCMOABC and dMOABC are implemented by our JAVA codes. To make the comparison process much easier, the best results for each function are bolded and underlined, while the second best ones are only bolded. It is shown by Tables 5 and 6 that DMCMOABC is very competitive compared to other well-known algorithms concerning the number of the cases where it obtains the best and the second best results.

To scientifically assess each algorithm, the Friedman test is applied to the obtained results, and average rankings of each method on unconstrained test functions are listed in Table 7. In this test, the Friedman statistic (distributed according to chi-square with 9 degrees of freedom) is 44.978182, and the associated p-value is 0.000001, indicating that there exits significant differences over the whole multiple comparison among all algorithms. Table 7 shows that our DMCMOABC gets the best average ranking outperforming MOEA/D and MTS, which are followed by dMOABC and DMOEADD. The performance of LiuLiAlgorithm and GDE3 are

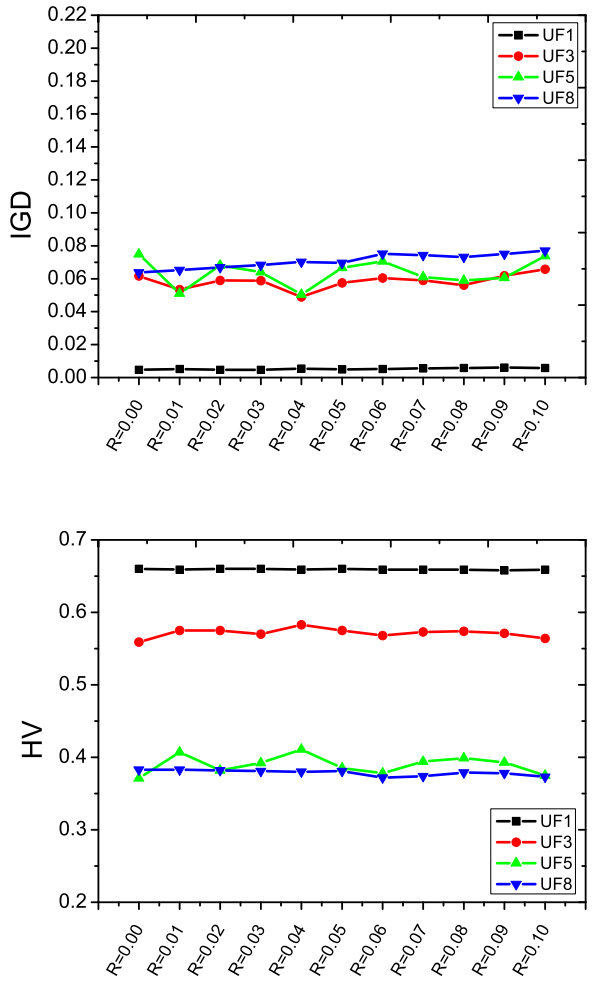


Fig. 9. The broken-line graphs of metrics when parameter R is varied from 0.00 to 0.10.

comparable, while S-MOABC/NS, NSGAII and SPEA2 get worse rankings compared to their counterparts.

Using the rankings computed by the Friedman test, the unadjusted/adjusted p-values are obtained with the application of a set of post hoc procedures, including the Bonferroni, Holm, Hochberg, Hommel, Holland, Rom, Finner and Li tests [47]. These p-values are shown in Table 8 which are computed by the KEEL Software Tool[2] [49,50]. As we can see in the table, the Friedman test shows a significant improvement of DMCMOABC over SPEA2, NSGAII and S-MOABC/NS for all considered post hoc procedures, with a level of significance $\alpha = 0.01$. The proposed algorithm is significantly better than GDE3 with $\alpha = 0.1$ for the Bonferroni, Holm, Hochberg, Holland and Rom tests, and with $\alpha = 0.05$ for the Hommel, Finner and Li test procedures. Compared to the LiuLiAlgorithm, the Friedman test presents a significant improvement with $\alpha = 0.1$ for all post hoc test procedures but the Bonferroni. The Finner and Li tests exhibit the most powerful behavior, reaching the lowest p-values in this pairwise comparison. It is revealed by the Li post hoc test procedure that the DMCMOABC obviously outperforms DMOEADD with a level of significance $\alpha = 0.1$. For the remaining pairs of comparisons (i.e., DMCMOABC vs. dMOABC, DMCMOABC vs. MTS, DMCMOABC vs. MOEA/D), the performance differences are not detected for any post hoc procedure, indicating the fact that our algorithm is comparable to these especially competitive algorithms.

[1] http://jmetal.sourceforge.net/

[2] http://www.keel.es/

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033

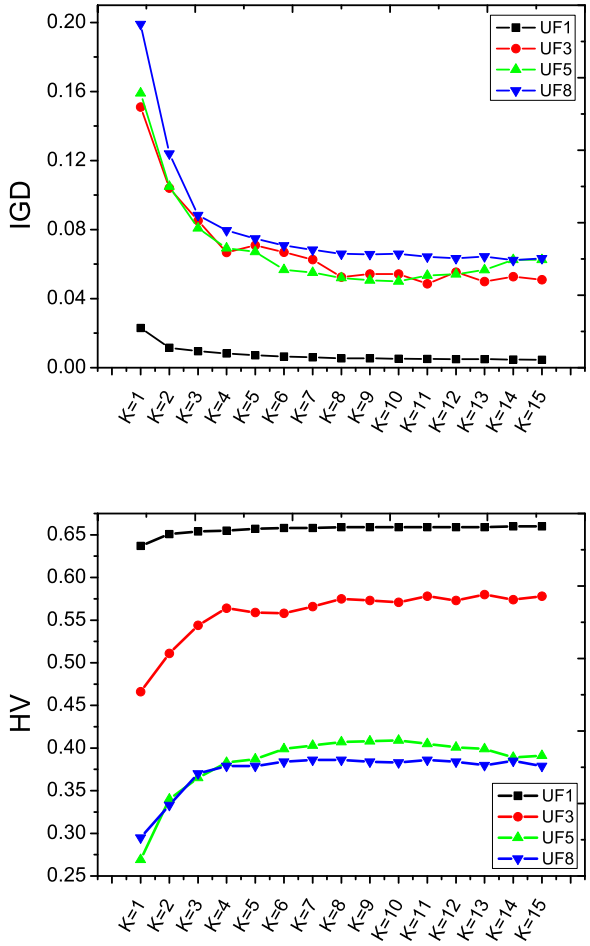


Fig. 10. The broken-line graphs of metrics when parameter K is varied from 1 to 15.

Tables 9 and 10 show the average rankings and adjusted p-values obtained by Friedman test on constrained test functions. In this test, the control algorithm is DMOEADD (i.e., the method with the best rankings) and our algorithm gets the second best ranking. As revealed in Table 10, DMOEADD, generally accepted as the best algorithm for solving constrained problems in CEC2009 competition, shows no significant improvement over DMCMOABC since the adjusted p-values for all post hoc procedures are large enough to accept the null hypothesis H_0 that is defined as the statement of no effect or difference between these two algorithms.

To evaluate the speed of the proposed DMCMOABC, the runtime of our algorithm, together with MOEA/D, dMOABC, MOEA/D, NSGAII and SPEA2, is recorded and listed in Table 11. MOEA/D is a very competitive algorithm for unconstrained optimization problems, and dMOABC is a latest multi-objective algorithm based on the ABC method. The DMCMOABC uses the same crowding distance strategy as in NSGAII to maintain the external archive, and the same fitness assignment procedure as in SPEA2. Therefore, they are chosen as the comparing algorithms. This experiment is conducted on a computer with the following environments: Pentium (R) Dual-Core CPU, T4500 2.30 GHz 2.29 GHz, 1.99 GB memory. From Table 11, we can find that DMCMOABC achieves the best runtime on UF3, UF5 and UF6, while MOEA/D is the fastest algorithm on the remaining test problems. Generally, DMCMOABC is slower than MOEA/D and NSGAII, but runs faster than dMOABC and SPEA2. Similar results are observed on other test problems. In our algorithm, once a better individual is found, the procedure will try to add it into the archive which may involve the re-calculation of crowding distances of all non-domination members (see Figs. 3 and 4, and Section 3.2). This mechanism is beneficial for the improvement of the quality of the returned non-dominated solutions, however, it may also introduce more computational efforts. In dMOABC, the external archive is maintained by a self-adaptive grid that involves a recursion procedure when dividing the objective space. Hence, the runtime of dMOABC naturally increases. This may be the reason why our algorithm is slower than some algorithms, such as MOEA/D, but faster than dMOABC.

As the summary of this subsection, we list some findings through the usage of Friedman test. The computed adjusted p-values show that DMCMOABC is significantly better than some state-of-the-art algorithms such as SPEA2, NSGAII, S-MOABC/NS, GDE3 and LiuLiAlgorithm, and achieves comparable performance when compared to the most competitive algorithms such as MOEA/D, DMOEADD, MTS and dMOABC. When comparing algorithms in terms of the runtime, DMCMOABC also outperforms some other algorithms, such as dMOABC and SPEA2.

4.4. The comparison of approximated Pareto fronts

To intuitively compare the performance of DMCMOABC and its competitors, the best Pareto fronts with lowest IGD value over 30 runs are plotted. For space limitation, these fronts are provided in Figs. A.1–A.6 in Appendix A. In these figures, MOEA/D and DMOEADD are chosen as the comparing algorithms since they are the most competitive algorithms in CEC2009 competition for unconstrained and constrained test functions, respectively. The approximated Pareto fronts of these two algorithms are borrowed from [4] and [6]. In the comparison process, two properties of the Pareto fronts are taken into consideration: one is the convergence, and the other is the diversity.

It is shown in Fig. A.1 that the best Pareto fronts produced by DMCMOABC are comparable to those of MOEA/D on UF1 and UF2 in terms of both convergence and diversity. But DMCMOABC fails to approximate a well converged and properly distributed Pareto front on UF3, thus the performance of the algorithm is significantly worse than that of MOEA/D.

As we can see in Fig. A.2, both the Pareto fronts on UF4 distribute nearly uniformly, but our algorithm is better than MOEA/D in terms of the convergence of the produced front. For test functions UF5 and UF6, the number of non-dominated solutions returned by MOEA/D is larger than that found by DMCMOABC algorithm. However, the front approximated by our algorithm has better convergence on UF5.

Next, we take a look at Fig. A.3. For UF7, both the Pareto fronts converge to the true one well, but the diversity of MOEA/D is slightly better than DMCMOABC since a small part of the top-left corner of the produced front is not successfully covered by our algorithm. For UF8 and UF9, the fonts approximated by MOEA/D are better than those of DMCMOABC in terms of the diversity, but are slightly worse when considering the convergence of the fronts. The DMCMOABC shows significant improvement over MOEA/D on UF10 because the computed front found by the former is much better when both criteria are taken into consideration (see (a) and (b) in Fig. A.4). Hence, the IGD value obtained by DMCMOABC naturally decreases.

For constrained functions (CF1–CF7), the quality of the Pareto fronts produced by DMCMOABC is comparable to that found by DMOEADD concerning both convergence and diversity (see Figs. A.4–A.6).

In this subsection, the approximated Pareto fronts produced by DMCMOABC and its counterparts are intuitively compared, and the results show that DMCMOABC has the ability to generate a set of well converged and properly distributed non-dominated solutions.

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033

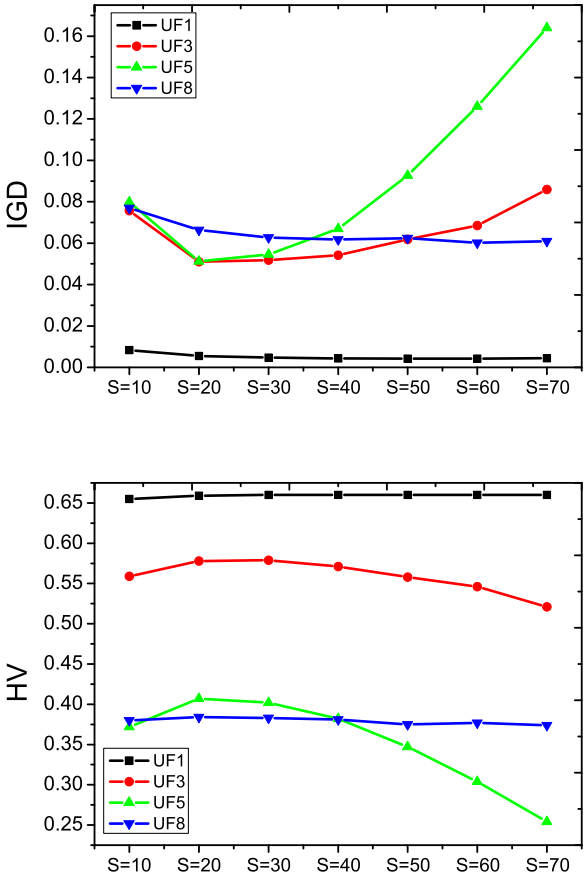


Fig. 11. The broken-line graphs of metrics when parameter S is varied from 10 to 70.

4.5. Discussion on the effect of control parameters

In this subsection, the effect of some control parameters is investigated by observing the variation tendency of IGD and HV on test functions UF1, UF3, UF5 and UF8. The main reason for choosing them is that they are the most representative functions of the categories to which they belong, where UF1 and UF3 are representatives of test functions with two objectives whose Pareto fronts are continuous. The UF5 is an example with discontinuous fronts, and UF8 represents a set of three-objective functions.

The broken-line graphs of the mean values of IGD and HV over 15 runs are shown in Figs. 8–11. It is observed from Fig. 8 that the performance of DMCMOABC tends to decrease as R increases from 0.2 to 1.0 in terms of both considered metrics on all selected test functions. The best value may distribute between 0.0 and 0.1, however, differences of each metric are not significant when R is changed from 0.0 to 0.1 with a step size 0.01 (see Fig. 9). We could also find in this figure that DMCMOABC produces relative better results when $R = 0.01$, which is consistent to our previous finding in Section 4.2.

Next we turn to analyze the effect of parameter K. Fig. 10 shows that the performance of DMCMOABC decreases sharply as K increases from 1 to 4, and changes slightly when K is larger than 5. Especially, the worst values of both IGD and HV are observed when K is set to 1, indicating that DMCMOABC is significantly better than the algorithm with only one colony and that the proposed multi-colony model indeed improves the performance of our algorithm. In fact, the diversity is more important than convergence in multi-objective algorithms for deal with some MOPs [51]. In DMCMOABC, K colonies search in different parts of the decision space and share information occasionally. This mechanism could enable the algorithm to explore more possible sub-spaces, thus, the diversity of the final solutions may be well kept.

Finally, we can find from Fig. 11 that the best overall performance is obtained by setting S to 20. As we all know, an important issue we should consider in the design of multi-objective algorithms, is to keep a balance between exploration and exploitation. Too small S is not good for the exploration process since only a small number of new food sources are generated in each iteration, while too large S will result in insufficient iterations or exploitation. Thus, a balance should be kept by setting S to a proper value. In this study, it seems to be quite suitable to set S to 20 for general usage.

5. Conclusion

In this paper, we suggest a new multi-objective artificial bee colony algorithm by using a multi-colony model and a dynamic information exchange strategy. In the proposed algorithm, K colonies search in their own sub-spaces independently most of the time and share information occasionally that is controlled by a parameter named migration rate. In each migration, an elite is selected from the external archive and used to replace the worst food source in a randomly chosen colony. The migration direction is dynamic since the elite may be found by any colony and the received colony is also determined stochastically. The control parameters of our algorithm are determined by an experiment and it is recommended to set R to 0.01, K to 9 and S to 20 for general usage, respectively.

The proposed algorithm is evaluated on a set of unconstrained and constrained problems from CEC2009 special session and competition in terms of four commonly used metrics EPSILON, HV, IGD and SPREAD, and it is compared with other state-of-the-art algorithms by applying Friedman test on the values of metric IGD. The results of post hoc procedures show that our algorithm is significantly better than SPEA2, NSGAII, S-MOABC/NS, GDE3 and LiuLiAlgorithm, and is highly competitive to the famous MOEA/D, DMOEADD, MTS and dMOABC for unconstrained test problems. As for constrained ones, the adjusted p values indicate that there is no significant difference between the proposed algorithm and DMOEADD which is generally accepted as the most effective algorithm in CEC2009 competition. When assess algorithms in terms of the runtime, our proposed algorithm also outperforms some other algorithms such as dMOABC and SPEA2.

The approximated Pareto fronts obtained by DMCMOABC are intuitively compared with those found by other competitively algorithms. It is revealed by the results that our algorithm has the ability to generate a set of well converged and appropriately distributed non-dominated solutions.

Our further work will focus on the extension of the algorithm to handle many-objective optimization problems, or use it in some real-world applications in the field of industrial control systems, logistics location, optimal power flow (OPF) in power systems, etc.

Acknowledgements

The authors would like to thank the anonymous reviewers for providing valuable comments to improve this paper, and add special thanks to X.X. Zheng for her help in the experimental study of this paper.

Appendix A. Pareto fronts obtained by DMCMOABC and its competitors on CEC2009 test problems

In the appendix, the approximated Pareto fronts obtained by our proposed DMCMOABC algorithm are intuitively compared with

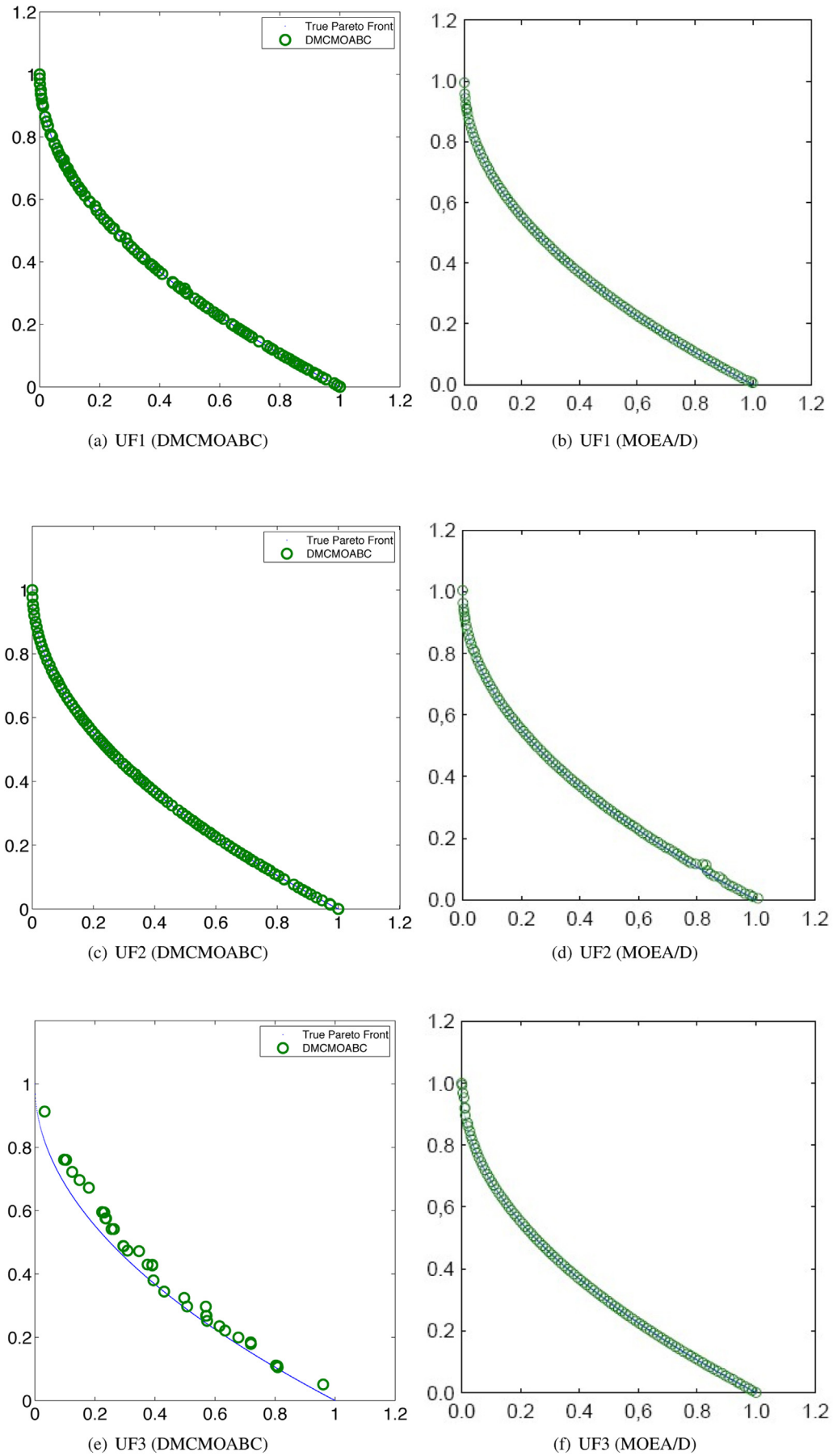


Fig. A.1. The best Pareto fronts obtained by DMCMOABC and its competitor on UF1–UF3.

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033

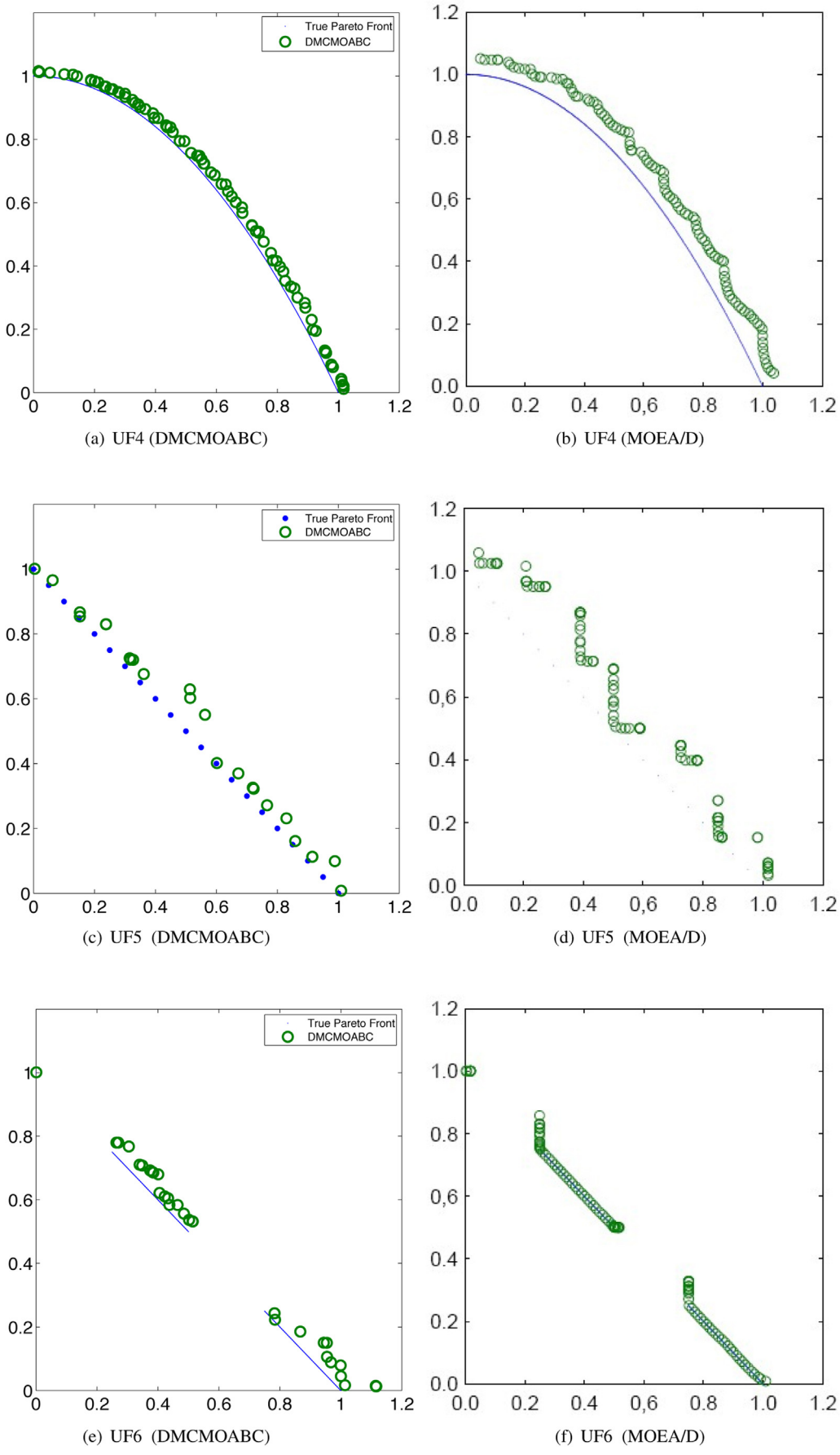
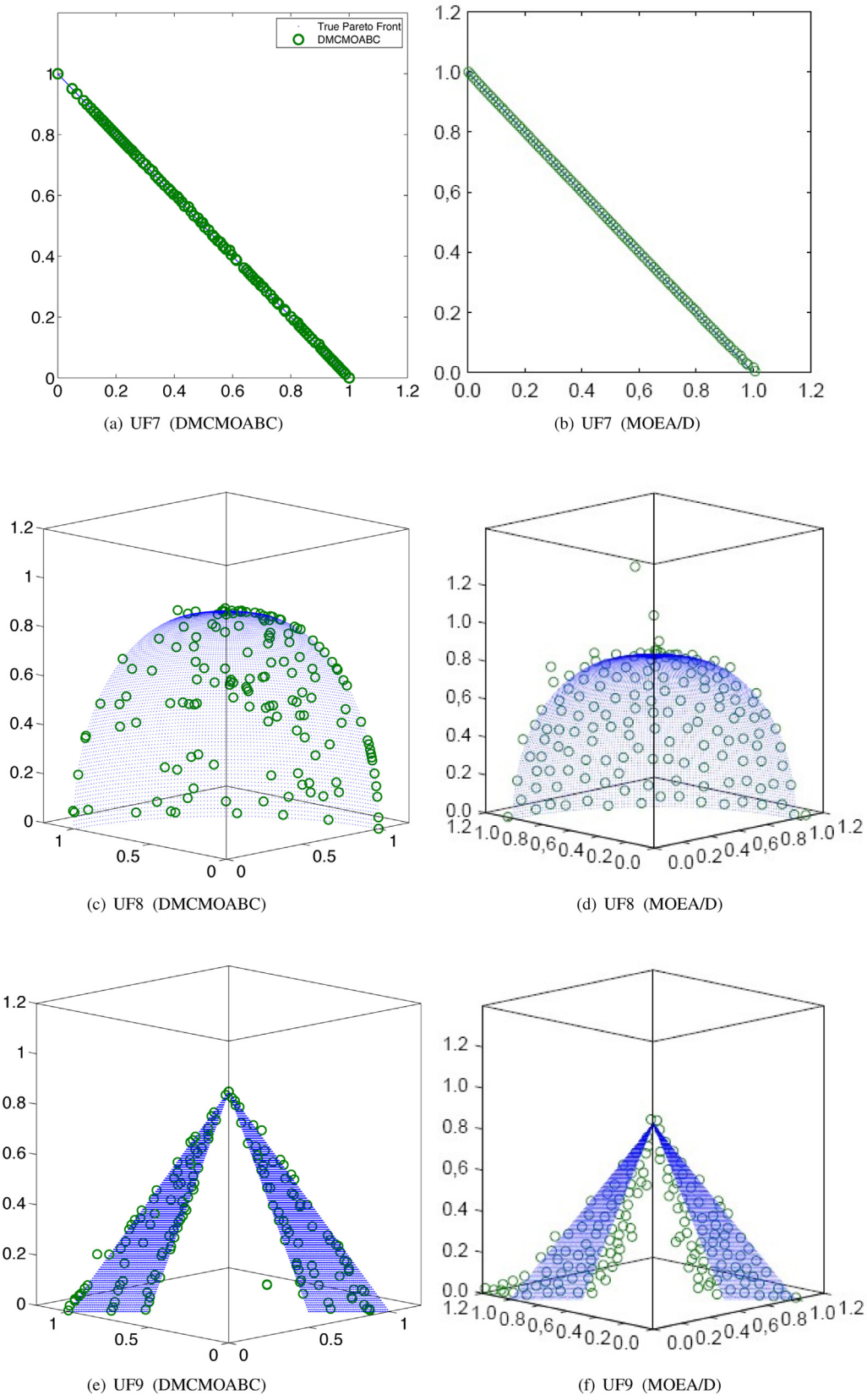


Fig. A.2. The best Pareto fronts obtained by DMCMOABC and its competitor on UF4–UF6.

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033



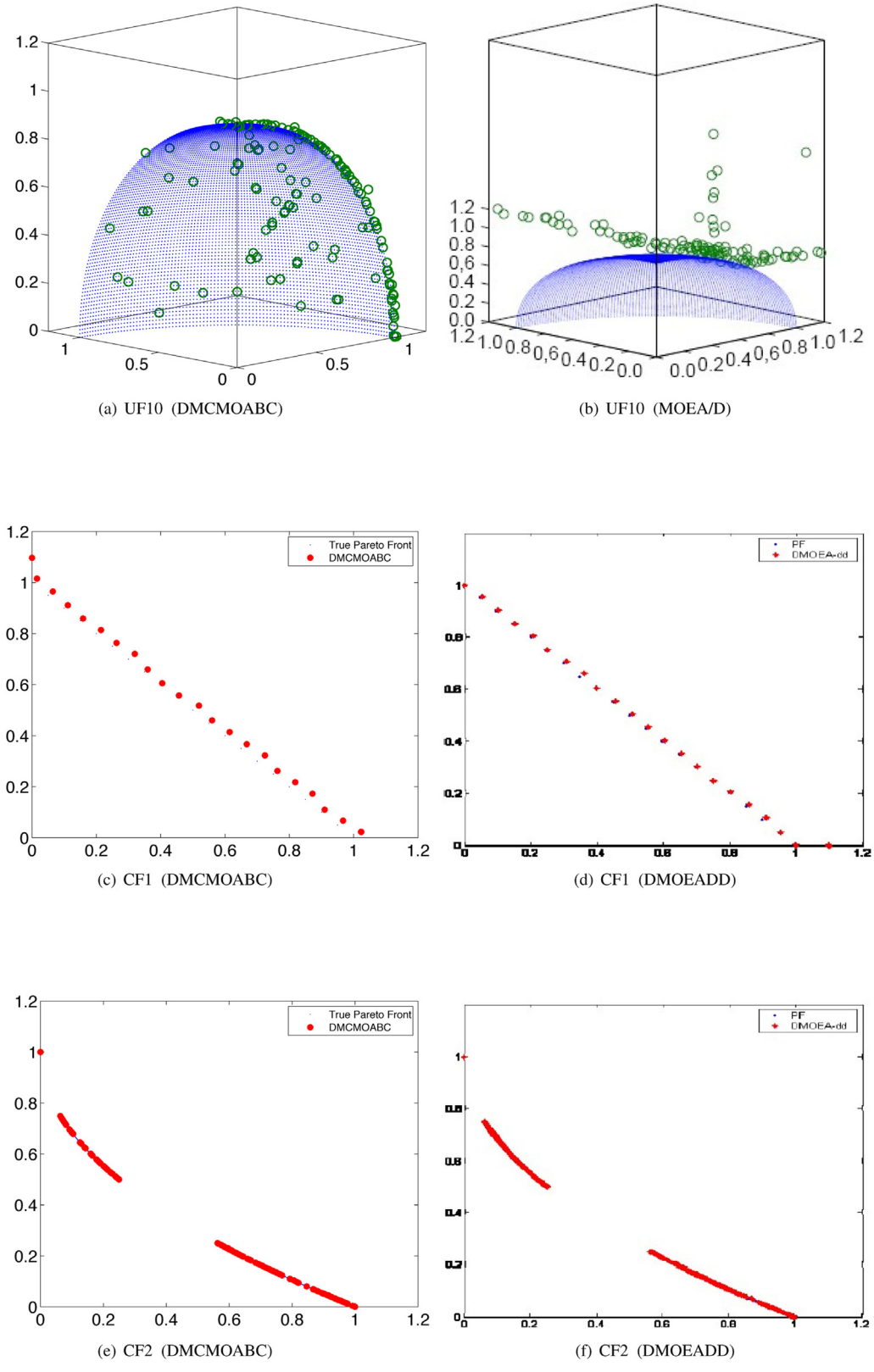
(a) UF7 (DMCMOABC) (b) UF7 (MOEA/D)

(c) UF8 (DMCMOABC) (d) UF8 (MOEA/D)

(e) UF9 (DMCMOABC) (f) UF9 (MOEA/D)

Fig. A.3. The best Pareto fronts obtained by DMCMOABC and its competitor on UF7–UF9.

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033



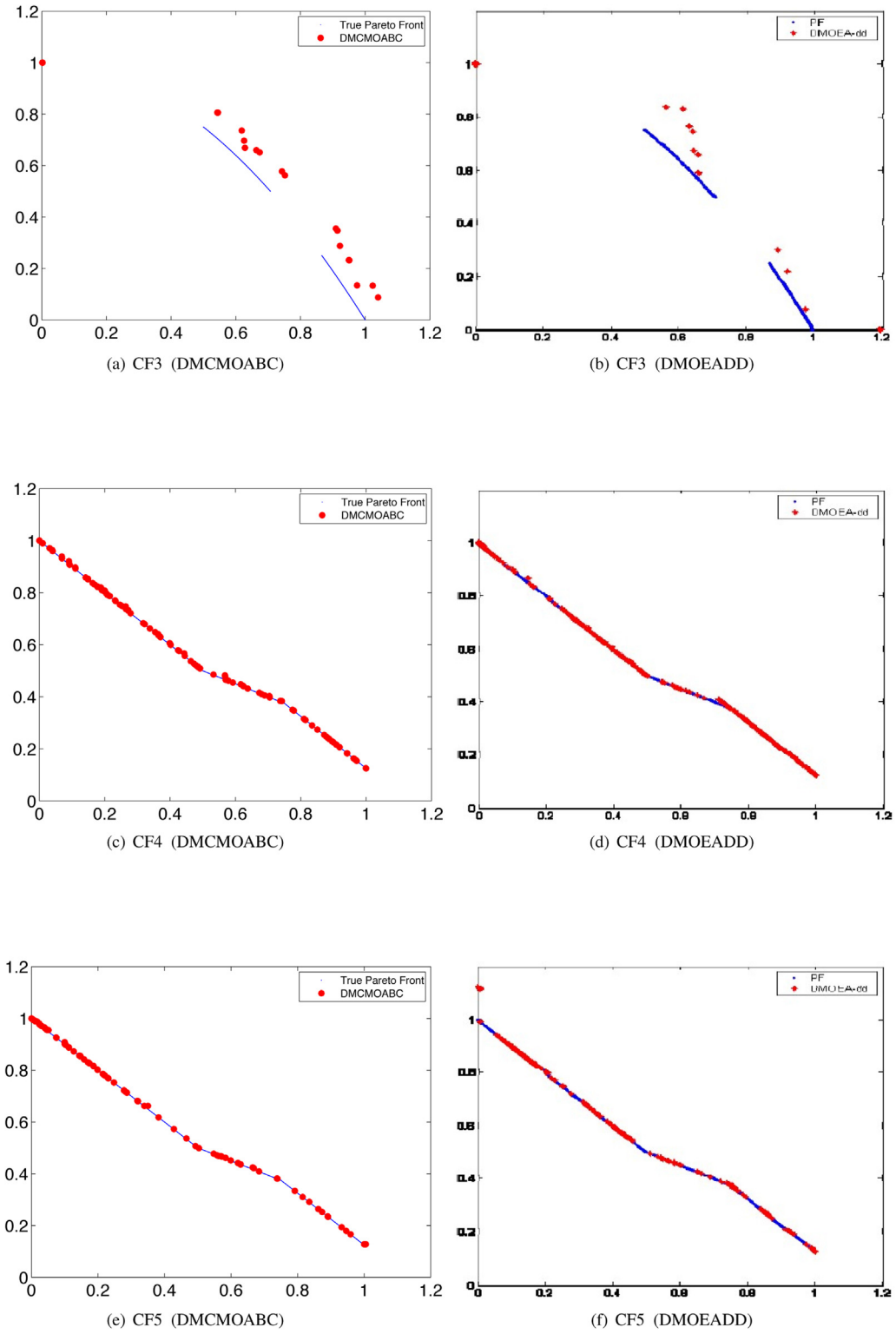
(a) UF10 (DMCMOABC) (b) UF10 (MOEA/D)

(c) CF1 (DMCMOABC) (d) CF1 (DMOEADD)

(e) CF2 (DMCMOABC) (f) CF2 (DMOEADD)

Fig. A.4. The best Pareto fronts obtained by DMCMOABC and its competitor on UF10, CF1 and CF2.

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033



(a) CF3 (DMCMOABC) (b) CF3 (DMOEADD)

(c) CF4 (DMCMOABC) (d) CF4 (DMOEADD)

(e) CF5 (DMCMOABC) (f) CF5 (DMOEADD)

Fig. A.5. The best Pareto fronts obtained by DMCMOABC and its competitor on CF3–CF5.

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033

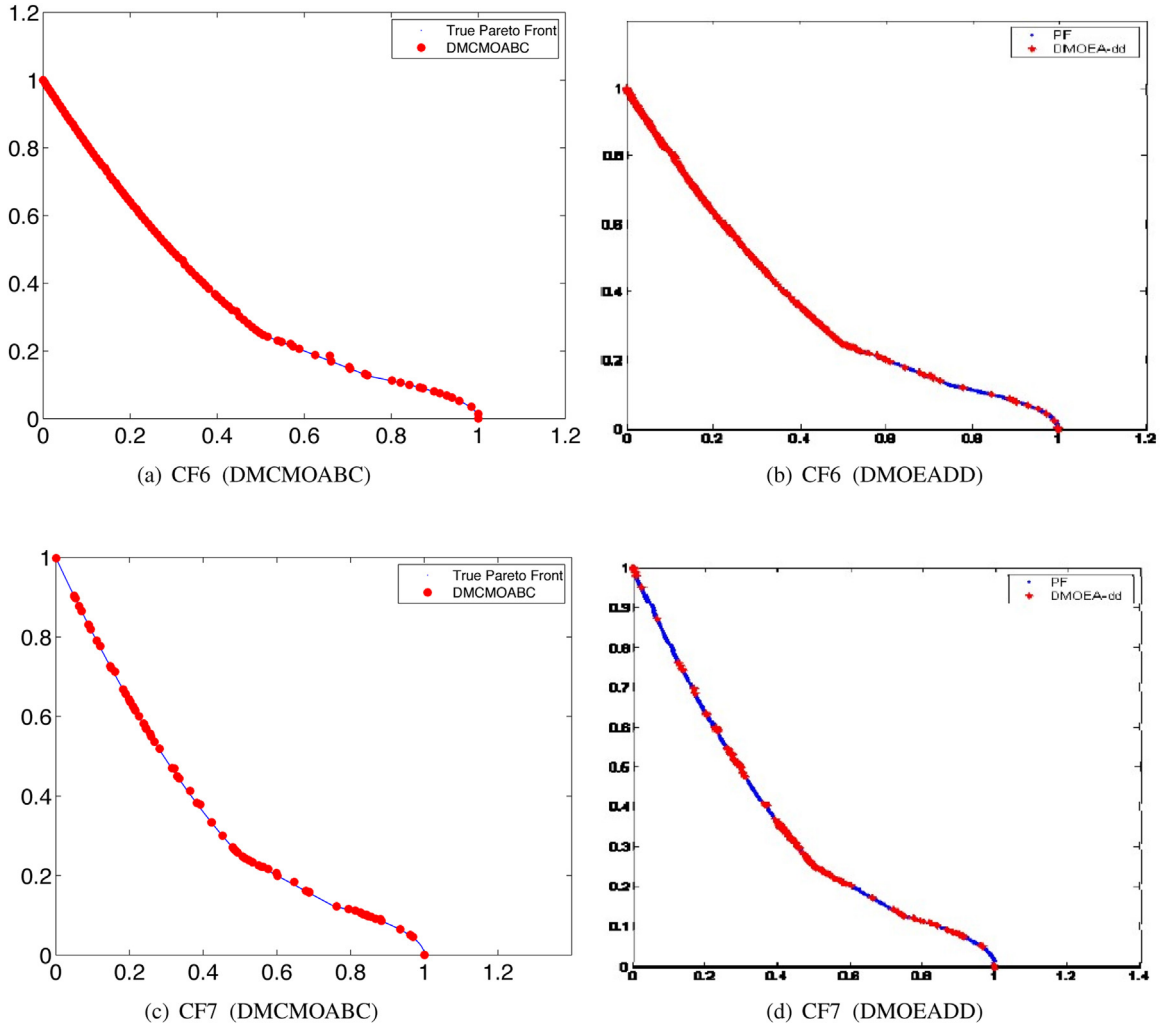


Fig. A.6. The best Pareto fronts obtained by DMCMOABC and its competitor on CF6 and CF7.

those given by MOEA/D and DMOEADD on unconstrained and constrained problems, respectively.

References

[1] B. Akay, Synchronous and asynchronous pareto-based multi-objective artificial bee colony algorithms, J. Glob. Optim. 57 (2) (2013) 415–445.
[2] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evolut. Comput. 6 (2) (2002) 182–197.
[3] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm, Tech. Rep., Computer Engineering and Networks Laboratory, Department of Electrical Engineering, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland, 2001.
[4] Q. Zhang, W. Liu, H. Li, The performance of a new version of MOEA/D on CEC09 unconstrained MOP instances, in: IEEE Congress on Evolutionary Computing (CEC), Trondheim, 2009, pp. 18–21.
[5] L.-Y. Tseng, C. Chen, Multiple trajectory search for unconstrained/constrained multi-objective optimization, in: Proceeding of Congress on Evolutionary Computation, CEC'09, IEEE, 2009, pp. 1951–1958.
[6] M. Liu, X. Zou, Y. Chen, Z. Wu, Performance assessment of DMOEA-DD with CEC 2009 MOEA competition test instances, in: Proceeding of Congress on Evolutionary Computation, CEC'09, IEEE, 2009, pp. 2913–2918.
[7] H.-L. Liu, X. Li, The multiobjective evolutionary algorithm based on determined weight and sub-regional search, in: Proceeding of Congress on Evolutionary Computation, CEC'09, IEEE, 2009, pp. 1928–1934.
[8] S. Kukkonen, J. Lampinen, Performance assessment of generalized differential evolution 3 with a given set of constrained multi-objective test problems, in: Proceeding of Congress on Evolutionary Computation, CEC'09, IEEE, 2009, pp. 1943–1950.
[9] C.A. Coello Coello, G.B. Lamont, D.A.V. Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd ed., Springer Science + Business Media, LLC, New York, NY, 2007.
[10] E. Cantú-Paz, A survey of parallel genetic algorithms, Calculateurs paralleles, reseaux et systems repartis 10 (2) (1998) 141–171.
[11] S.-C. Lin, W. Punch III, E.D. Goodman, Coarse-grain parallel genetic algorithms: categorization and new approach, in: Proceedings of Sixth IEEE Symposium on Parallel and Distributed Processing, IEEE, 1994, pp. 28–37.
[12] M. Middendorf, F. Reischle, H. Schmeck, Multicolony ant algorithms, J. Heuristics 8 (3) (2002) 305–320.
[13] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proceedings of 2005 IEEE Swarm Intelligence Symposium, 2005, pp. 124–129.
[14] H. Chen, Y. Zhu, K. Hu, Multi-colony bacteria foraging optimization with cell-to-cell communication for RFID network planning, Appl. Soft Comput. 10 (2) (2010) 539–547.
[15] W. Xu, R. Wang, L. Zhang, X. Gu, A multi-population cultural algorithm with adaptive diversity preservation and its application in ammonia synthesis process, Neural Comput. Appl. 21 (6) (2012) 1129–1140.
[16] M.A. Al-Betar, M.A. Awadallah, A.T. Khader, Z.A. Abdalkareem, Island-based harmony search for optimization problems, Expert Syst. Appl. 42 (4) (2015) 2026–2035.
[17] R. Schaefer, A. Byrski, M. Smołka, The island model as a Markov dynamic system, Int. J. Appl. Math. Comput. Sci. 22 (4) (2012) 971–984.
[18] A.A. Montaño, C. Coello Coello, E. Mezura-Montes, pMODE-LD+SS. an effective and efficient parallel differential evolution algorithm for multi-objective optimization, in: Parallel Problem Solving from Nature, PPSN XI, vol. 6239 of Lecture Notes in Computer Science, Springer, Berlin; Heidelberg, 2010, pp. 21–30.
[19] H.R. Cheshmehgaz, M.I. Desa, A. Wibowo, An effective model of multiple multi-objective evolutionary algorithms with the assistance of regional multi-objective evolutionary algorithms: VIPMOEAs, Appl. Soft Comput. 13 (5) (2013) 2863–2895.
[20] R. Shang, Y. Wang, J. Wang, L. Jiao, S. Wang, L. Qi, A multi-population cooperative coevolutionary algorithm for multi-objective capacitated arc routing problem, Inf. Sci. 277 (2014) 609–642.
[21] X. Zhang, X. Guan, Y. Zhu, J. Lei, Strategic flight assignment approach based on multi-objective parallel evolution algorithm with dynamic migration interval, Chin. J. Aeronaut. 28 (2) (2015) 556–563.

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033

[22] D.A. Van Veldhuizen, J.B. Zydallis, G.B. Lamont, Considerations in engineering parallel multiobjective evolutionary algorithms, IEEE Trans. Evolut. Comput. 7 (2) (2003) 144–173.
[23] L. Bui, H. Abbass, D. Essam, Local models – an approach to distributed multi-objective optimization, Comput. Optim. Appl. 42 (1) (2009) 105–139.
[24] P.C. Chang, S.H. Chen, The development of a sub-population genetic algorithm II (SPGA II) for multi-objective combinatorial problems, Appl. Soft Comput. 9 (1) (2009) 173–181.
[25] L. Jiao, H. Wang, R. Shang, F. Liu, A co-evolutionary multi-objective optimization algorithm based on direction vectors, Inf. Sci. 228 (2013) 90–112.
[26] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, Tech. Rep., Engineering Faculty, Computer Engineering Department, Erciyes University, 2005.
[27] P.-W. TSai, J.-S. Pan, B.-Y. Liao, S.-C. Chu, Enhanced artificial bee colony optimization, Int. J. Innov. Comput. Inf. Control 5 (12) (2009) 5081–5092.
[28] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Appl. Math. Comput. 214 (1) (2009) 687–697.
[29] N. Karaboga, A new design method based on artificial bee colony algorithm for digital IIR filters, J. Frankl. Inst. 346 (4) (2009) 328–348.
[30] T. Cura, An artificial bee colony algorithm approach for the team orienteering problem with time windows, Comput. Ind. Eng. 74 (2014) 270–290.
[31] W. Chen, Artificial bee colony algorithm for constrained possibilistic portfolio optimization problem, Physica A 429 (2015) 125–139.
[32] C.M. Vargas Benítez, H.S. Lopes, Parallel artificial bee colony algorithm approaches for protein structure prediction using the 3DHP-SC model, in: M. Essaaidi, M. Malgeri, C. Badica (Eds.), Intelligent Distributed Computing IV, vol. 315 of Studies in Computational Intelligence, Springer, Berlin; Heidelberg, 2010, pp. 255–264, http://dx.doi.org/10.1007/978-3-642-15211-5_27
[33] R.S. Parpinelli, C.M. Vargas Benítez, H.S. Lopes, Parallel approaches for the artificial bee colony algorithm, in: B. Panigrahi, Y. Shi, M.-H. Lim (Eds.), Handbook of Swarm Intelligence, vol. 8 of Adaptation, Learning, and Optimization, Springer, Berlin; Heidelberg, 2011, pp. 329–345.
[34] S. Biswas, S. Das, S. Debchoudhury, S. Kundu, Co-evolving bee colonies by forager migration: a multi-swarm based artificial bee colony algorithm for global search space, Appl. Math. Comput. 232 (2014) 216–234.
[35] R. Akbari, R. Hedayatzadeh, K. Ziarati, B. Hassanizadeh, A multi-objective artificial bee colony algorithm, Swarm Evolut. Comput. 2 (2012) 39–52.
[36] Y.-B. Zhong, Y. Xiang, H.-L. Liu, A multi-objective artificial bee colony algorithm based on division of the searching space, Appl. Intell. 41 (4) (2014) 987–1011.
[37] M.A. Medina, S. Das, C.A.C. Coello, J.M. Ramírez, Decomposition-based modern metaheuristic algorithms for multi-objective optimal power flow a comparative study, Eng. Appl. Artif. Intell. 32 (2014) 10–20.
[38] H. Chen, M.L. Bo, Y. Zhu, Multi-hive bee foraging algorithm for multi-objective optimal power flow considering the cost, loss, and emission, Int. J. Electr. Power Energy Syst. 60 (2014) 203–220.
[39] W. Zou, Y. Zhu, H. Chen, B. Zhang, Solving multiobjective optimization problems using artificial bee colony algorithm, Discret. Dyn. Nat. Soc. 2011 (2011) 1–37.
[40] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Inf. Sci. 192 (1) (2012) 120–142.
[41] Y. Xiang, Y. Peng, Y. Zhong, Z. Chen, X. Lu, X. Zhong, A particle swarm inspired multi-elitist artificial bee colony algorithm for real-parameter optimization, Comput. Optim. Appl. 57 (2) (2014) 493–516.
[42] Y. Xiang, Y. Zhou, H. Liu, An elitism based multi-objective artificial bee colony algorithm, Eur. J. Oper. Res. 245 (1) (2015) 168–193.
[43] M. Mernik, S.-H. Liu, D. Karaboga, M. Črepinšek, On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation, Inf. Sci. 291 (0) (2015) 115–127.
[44] H. Jain, K. Deb, An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach. Part II: Handling constraints and extending to an adaptive approach, IEEE Trans. Evolut. Comput. 18 (4) (2014) 602–622.
[45] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, S. Tiwari;1; Multiobjective optimization test instances for the CEC 2009 special session and competition, University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special session on performance assessment of multi-objective optimization algorithms, technical report.
[46] J.J. Durillo, A.J. Nebro, jMetal: a java framework for multi-objective optimization, Adv. Eng. Softw. 42 (2011) 760–771.
[47] J. Derrac, S. Garcia, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evolut. Comput. 1 (7) (2011) 3–18.
[48] J.J. Durillo, A.J. Nebro, E. Alba, The jMetal framework for multi-objective optimization: design and architecture, in: CEC 2010, 2010, pp. 4138–4325.
[49] J. Alcalá-Fdez, L. Sánchez, S. Garcí a, M.J. del Jesús, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, et al., KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput. 13 (3) (2009) 307–318.
[50] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. Garcí a, L. Sánchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Multiple-Valued Log. Soft Comput. 17 (2–3) (2011) 255–287.
[51] H.-L. Liu, F. Gu, Q. Zhang, Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems, IEEE Trans. Evolut. Comput. 18 (3) (2014) 450–455, http://dx.doi.org/10.1109/TEVC.2013.2281533.

Please cite this article in press as: Y. Xiang, Y. Zhou, A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization, Appl. Soft Comput. J. (2015), http://dx.doi.org/10.1016/j.asoc.2015.06.033