1-1-2010

Autonomous Bee Colony Optimization for Multi-objective Function

F Zeng

James Decraene

Malcolm Low

Philip Hingston
Edith Cowan University

C Wentong

See next page for additional authors

10.1109/CEC.2010.5586057

This is an Author's Accepted Manuscript of: Zeng, F., Decraene, J., Low, M., Hingston, P. F., Wentong, C., Suiping, Z., & Chandramohan, M. (2010). Autonomous Bee Colony Optimization for Multi-objective Function. Proceedings of IEEE Congress on Evolutionary Computation. (pp. 1279-1286). . Barcelona International Convention Centre, Barcelona, Spain. IEEE. Available here

Autonomous Bee Colony Optimization for Multi-objective Function

Fanchao Zeng, James Decraene, Malcolm Yoke Hean Low, Philip Hingston, *Senior Member, IEEE*,
Cai Wentong, Zhou Suiping, Mahinthan Chandramohan

***Abstract*—An Autonomous Bee Colony Optimization (A-BCO) algorithm for solving multi-objective numerical problems is proposed. In contrast with previous Bee Colony algorithms, A-BCO utilizes a diversity-based performance metric to dynamically assess the archive set. This assessment is employed to adapt the bee colony structures and flying patterns. This self-adaptation feature is introduced to optimize the balance between exploration and exploitation during the search process. Moreover, the total number of search iterations is also determined/optimized by A-BCO, according to user pre-specified conditions, during the search process. We evaluate A-BCO upon numerical benchmark problems and the experimental results demonstrate the effectiveness and robustness of the proposed algorithm when compared with the Non-dominated Sorting Genetic Algorithm II and the latest Multi-objective Bee Colony Algorithm proposed to date.**

I. Introduction

REAL world problems typically involve the simultaneous optimization of conflicting objectives [1, 2]. To solve these multi-objective combinatorial and/or numerical optimization problems, numerous nature-inspired algorithms have been proposed to locate their Pareto fronts. These algorithms can be classified into two main categories: evolutionary algorithms (EAs) and swarm intelligence based algorithms (SAs). Both families of techniques are inspired by real phenomena occurring in nature. EAs simulate natural evolution through the variation of genetic material and selection of fittest (from a phenotypical standpoint) candidate solutions. SAs exploit the collective intelligence emerging from the crowd behavior of social entities such as fish schools, bird flocks and insect colonies. Recent studies [1, 2] suggested that SAs are more suitable in terms of convergence speed to solve multi-objective optimization problems when compared with EAs.

Among SAs, the bee colony optimization (BCO) algorithm is characterized by specific behavioral features such as the waggle dance, task selection, collective decision making, and navigation. BCO has presented promising results when utilized to solve multi-objective problems [3, 4]. In BCO, the labor division properties (i.e., foraging behavior specifications) and self-organization dynamics are essential to achieve good performances. Specialized foraging-related tasks are conducted by differing types of bee whereas the self-organization dynamics are governed by four properties: positive feedback (waggle dance), negative feedback (food source exhaustion), fluctuations (random search) and social interactions (group discussion). Nevertheless, according to the taxonomy compiled by Karaboga and Akay [5], bee swarm algorithms are predominantly utilized to solve combinatorial problems such as the Traveling Salesman Problem, Job Shop Scheduling, Routing in Networks and Resource Allocation Problems. To date, only two studies [7, 8] have been reported in the literature for multi-objective numerical optimization.

In this paper, based on a self-adaptive foraging behavior of honey bees, an autonomous bee colony optimization (A-BCO) algorithm is proposed for multi-objective numerical functions. Three types of bees, i.e., elite, follower and scout bees are devised with distinct navigation patterns and specialized tasks. Elite bees are selected to perform the waggle dance to communicate foraging information with the other bees. The follower bees forage around flower patches of randomly selected elite bees. Scout bees carry out spontaneous searches in nearby areas. Unlike previous bee colony optimization algorithms, A-BCO may automatically determine a satisfactory number of search iterations. Moreover, both the bee colony structure (ratios of different types of bee) and flying patterns are subjected to variation during the search process. These dynamic changes are performed to optimize the balance between exploration (early phase of the search) and exploitation (later and final phase of the search). These self-adaptive properties are driven by a diversity-based running performance metric of best found solutions so far.

Indeed, A-BCO utilizes a diversity-based performance metric to dynamically assess the archive set. Using this diversity metric, A-BCO may determine the current stage of food foraging: either exploration or exploitation. During the exploration stage, the ratio of elite bees is relatively higher so that a wider exploration of the search space can be conducted. Also, the follower bees are allowed to expand their exploration around the flower patches found so far. During the exploitation stage, the bees attempt to improve/exploit the best found solutions. Here the ratio of elite bees is thus decreased; similarly the follower bees narrow even further the spread of their search. This final refinement stage aims at improving the distribution uniformity of solutions.

The remainder of the paper is structured as follows: An overview of related work is first provided. Then, a detailed description of A-BCO is given. Series of experiments involving multi-objective numerical optimization problems are conducted and finally discussed.

Fanchao Zeng, James Decraene, Malcolm Yoke Hean Low, Cai Wentong, Zhou Suiping and Mahinthan Chandramohan are with the School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639778 (email: fczeng@ntu.edu.sg).

Philip Hingston is with the School of Computer and Security Science, Edith Cowan University, Australia (email: phi@philiphingston.com).

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

II. Related Work

We first present several BCO techniques applied to both single and multi-objective numerical functions. Then we identify and discuss the limitations of these techniques. Karaboga and Basturk [4] proposed an Artificial Bee Colony (ABC) algorithm which relies on modeling the foraging behavior of honey bees. A bee colony structure, similar to the one described earlier, is employed. A single-objective multivariable function is used to compare the performance of ABC and other algorithms (Particle Swarm Optimization and Particle Swarm Inspired Evolutionary Algorithm). This study suggested that ABC can outperform the other algorithms when considering specific benchmark problems (e.g., the Rastrigin function). A similar method was developed by Sundareswaran and Sreedevi [6] and yielded the same conclusions. Akbari *et al.* [3] improved the previous bee colony algorithms by modifying the bee flying patterns. This method introduced two novel features, a repulsion factor & penalizing fitness (BSO-RP) and time-varying weights (BSO-TVW). Both features were introduced to address the stagnation, exploration and exploitation balancing problems. The experimental results showed that Akbari *et al.*'s technique can achieve competitive results and outperform ABC in terms of both success rate and convergence speed.

The above BCO algorithms were applied to single objective numerical functions. However, to our knowledge, only two BCO algorithms have been examined for multi-objective numerical functions. The earliest study on BCO and multi-objective optimization was conducted by Pham and Ghanbarzadeh [7] in 2007. A bee algorithm was proposed and evaluated over a multi-objective non-linear numerical engineering problem. Experimental results were compared with the Non-dominated Sorted Genetic Algorithm II (NSGA-II) and NSGA; it was shown that a similar Pareto front could be obtained using BCO. The most recent work, reported in the literature, to date was presented by Low *et al.* [8] where the Multi-objective Bee Colony Optimization (MOBCO) algorithm was proposed. A comparative study was conducted in which MOBCO presented comparable performance to NSGA-II when applied to the optimization of numerical functions (ZDT benchmarks).

Although previous bee colony algorithms demonstrated promising performances when applied to both single and multi-objective numerical functions, none of the algorithms included self-adaptive properties at both the macro (i.e., food forage duration) and micro level (i.e., the foraging behavior of bees). Self-adaptive properties may assist/relieve the user in setting the algorithm's parameters to best optimize a given problem:

1. The foraging/search duration is a key factor which may diminish the quality of solutions (when set too low) or waste computing resources (when set too high). Moreover the "optimal" search duration varies according to the optimization problem being addressed. We thus argue that the search duration should be self-adaptive (similarly to real bees that halt searching when they have discovered sufficient flower patches).
2. In existing BCO techniques, the food foraging behavior is determined by the composition of the bee swarm and flying patterns which are both typically fixed throughout the search process. We propose that these properties should be self-adaptive to promote efficient food foraging. This self-adaptation would exploit and dynamically optimize the balance between exploration and exploitation during the search [9].

The above self-adaptive methods proposed for BCO are described in detail in the next section.

III. Autonomous Bee Colony Optimization

This section introduces the behavior of real bees in nature and then provides an overview of A-BCO, finally the algorithm is detailed.

A. Honey Bees in Nature

The self-organized and collective behavior of colony insects enables them to solve multi-objective problems which cannot be addressed by single insects acting independently. In the case of honey bees, this behavior facilitates the search of flower patches in the environment. Information, with regards to the flower patches, is shared among the bees when they return to the hive. This foraging behavior remained mysterious until Von Frisch [10] decoded the language of the bee waggle dance. The latter is used as a communication medium to describe the quality, distance and direction of the flower patches to other bees in the hive. Bees are then sent to the patches according to their profitability ratings (determined by the nectar quality, nectar bounty and distance from the hive). Further details on the waggle dance can be found in [11, 12]. We utilize this waggle dance mechanism as an inspiration to drive the self-adaptive properties of our BCO. Here, through the waggle dance, our artificial bees communicate with each other to share information about the quality (i.e., the diversity-based performance metric) of found flower patches (i.e., solutions). This information is then employed to adjust the structure of the bee swarm and flying patterns as follows: when the diversity of solutions found is low then the spread of search is increased (i.e., the exploration stage). In contrast, when the diversity of solutions is high then the spread of search is decreased (i.e., the exploitation stage).

B. Overview of A-BCO

The A-BCO search is predominantly driven by the diversity-based performance of the archive set (which contains the best solutions found so far). This measure is used to dynamically adjust the structure and flying patterns of the bees (i.e., the micro level) and also to identify the current stage of the search (i.e., the macro level).

At the micro level, three types of bee are distinguished: elite, follower and scout bees. The flying patterns of elite and follower bees are varied through the use of crossover operators, whereas scout bees employ a mutation operator to update their foraging path. The partition of the bees is determined by their individual fitness which represents the

quality of food source being foraged by the bee. Unlike previous approaches, the ratio of each type of bee is dynamically adjusted. This is conducted in accordance with the diversity-based performance of the solutions found so far. Only a relatively smaller fraction of the bees are selected as scout bees.

As mentioned earlier, the diversity-based performance is also used to identify the current stage of the search. This macro level feature is utilized to adjust the search duration accordingly. Three stages are distinguished during which the search behavior is characterized as follows:

1. *Exploration*, a wide search for solutions throughout the search space is conducted.
2. *Transition*, this phase occurring between the exploration and exploitation stages is devised to determine whether the exploration phase has converged towards promising global optima. In other words this phase ensures that we do not proceed to the exploitation stage if the search has actually converged to local optima solution points.
3. *Exploitation*, optimization (i.e., increase the distribution uniformity) of best solutions found so far is carried out. Neighboring areas of best found solutions are explored further.

In the next section, the A-BCO algorithm is presented.

C. The Algorithm

The main steps of A-BCO are illustrated in Fig. 1.

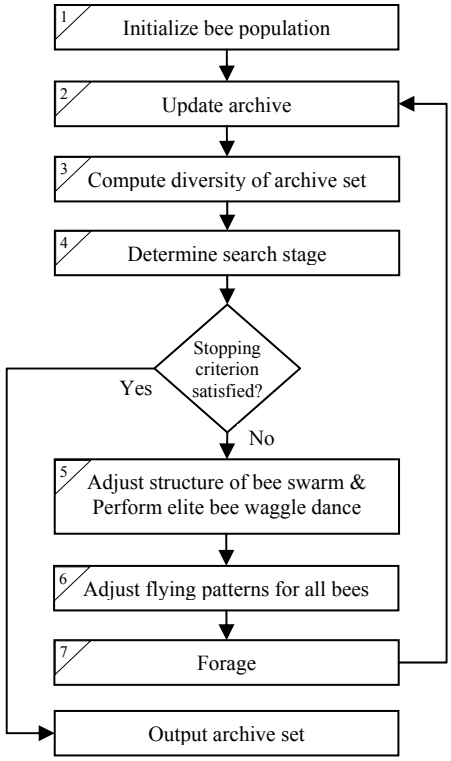

Fig. 1. A-BCO flowchart.

Each step of the A-BCO algorithm is now outlined:

1. The search process starts with the random initialization of the bee population.
2. According to the numerical objective functions being examined, the non-dominated solution sets are stored in the archive. The archive is used to store the best estimates of the Pareto front and is updated in each search iteration. The archive updating process contains two sub-steps:
   a. Firstly, the newly generated solution sets are combined with the non-dominated solution sets already stored in the archive. Then the dominated solutions are removed.
   b. Secondly, if the archive maximal size is reached, a recurrent truncation method based on crowding distance is utilized to remove the least "promising" non-dominated solutions, see Section D for details.
3. The diversity-based performance metric, given by $\alpha \in [0,1]$, of the solutions stored in the archive is calculated. α estimates the level of uniformity in the distribution of solutions in the archive set, i.e., if $\alpha=1$ then the solutions are uniformly distributed, whereas with $\alpha=0.6$ we may approximate that 40% of the solutions are not evenly distributed. Note that with $\alpha=0$, the archive set is empty, see Section E for details on computing α.
4. The current stage of food forage is determined according to the diversity of the archive set. Three stages or phases are distinguished: exploration, transition and exploitation. More details about the stage determination can be found in Section F.
5. The bee colony structure (i.e., ratios of elite, follower and scout bees) is adjusted according to α. This adjustment aims at maximizing α (i.e., increase the distribution uniformity of the solutions). The goal is to make the solutions in the archive set evenly distributed. Note that the archive size (K) is equal to the population size.

TABLE I
BEE COLONY STRUCTURE BASED ON DIVERSITY α.

Bee Type	Size
Elite	$(1-\alpha-s)K$
Follower	αK
Scout	sK, $s = 1/number\ of\ variables$

Table I lists the different bee type ratios which were devised according to the following considerations:

a. In typical experiments, the generated solution sets exhibit low diversity during the initial phase (i.e., α is low). In such cases the percentage of elite bees performing the waggle dance should be high (i.e., $1-\alpha$ to be high) so that exploration is emphasized. As the search proceeds, the archive set eventually becomes more diversified; the elite bee ratio should then be decreased to facilitate local fine tuning.
b. So according to the fitness (i.e., crowding distance)

of individual solutions, (1-α-s)*K* of the bees are selected as elite ones. After that, waggle dance is performed by the elite bees. Note that the number of scout bees is fixed throughout the search.

6. The flying patterns (i.e., the bees' search paths) are also subjected to variation. The scout bees use a polynomial mutation operator (promoting an increase in spread) to explore the search space further. The associated mutation probability is fixed. In contrast, elite and follower bees utilize the simulated binary crossover (SBX) method [14] to exploit the near-optimal generated solutions. The adjustment of flying patterns is achieved through the automated tuning of SBX's distribution index. This is performed in each search iteration. The diversity-based performance metric is again utilized to drive this adjustment. The implementation details are described in Section F.
7. Then, based on the adjusted flying patterns, the bees carry out food foraging.

Figure 2 depicts an overview of A-BCO where the interactions of the several techniques involved are depicted.

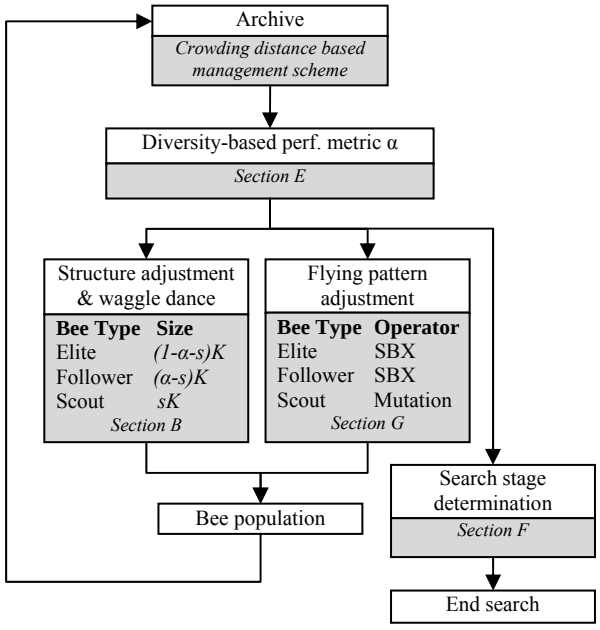

Fig. 2. Schematic overview of A-BCO and involved computational techniques.

D. Archive Management Scheme (AMS)

Archive is commonly used to store the approximation to the true Pareto front of the problem. Ideally, the size of the archive should be infinite; however, practically, this is not feasible. Therefore, a limit has to be set to define the maximal size of the archive. When this size limit is reached, the multi-objective optimization algorithm should be able to allow only better solutions to enter the archive, whereas the "least" valuable solutions are removed. In addition, to maintain a uniformly distributed set of solutions, the AMS ranks the solutions within the archive set to promote the most valuable solutions. We utilize the crowding distance method to rank the solutions. The crowding distance indicator was proposed by Deb et al. [15]. It serves as an estimation of the size of the largest cuboid enclosing the solution point. It could be regarded as a criterion to determine the value of the solution point. In this scheme, "boundary solutions" or highest and lowest objectives are given the maximum value in order to retain them. The crowding distance can be calculated by measuring the distance between the two immediate neighbors of a given point along each of the objective dimensions. Lastly, the "final crowding distance" is computed by adding the crowding distances obtained for each objective.

E. Diversity-based Performance Metric

A diversity-based performance metric is implemented to dynamically assess the quality of the archive. This metric is based on the running performance metrics proposed by Deb and Jain [13]. Two principal modifications are introduced:

1. The number of grids (approximating the diversity of the population, see Fig. 3) is derived by dividing the population size by the number of objectives (instead of requiring the user to manually define it as in [13]).
2. We argue that Deb and Jain's approach is limited by the requirement of *a priori* knowledge of the target solutions' distribution. Using this information, the number of grids can be determined/fitted. Nevertheless in real time/life optimization problems, this information is usually unavailable. Here the running metric does not refer to any pre-specified target set of solutions. Instead the running metric is employed to converge towards an ideal target set of solutions where each grid would possess a representative solution.

Given the minimal and maximal boundary values, the hyperplane is thus divided into a finite number of grids (population size divided by the number of objectives). The diversity performance metric is based on whether each grid contains a solution or not. The best diversity performance is achieved if all grids contain at least a solution. The steps to calculate the diversity are as follows:

Step 1: Calculate diversity array.

The number of integer variables in the diversity array is equal to the number of grids in the hyperplane. Each variable in the diversity array corresponds to one particular grid *i*. The value *h(i)* of the i^{th} elements is obtained using Eq. 1.

$$h(i) = \begin{cases} 1, if\ grid\ i\ contains\ a\ solution\ in\ the\ archive; \\ 0, otherwise. \end{cases} \quad (1)$$

Step 2: Assign a value m() to each grid i depending on its neighboring grids' h() values in the diversity array. The value of the i^{th} grid is calculated according to a mapping table (Table II).

For example let us consider the grid patterns p_1=|0|1|0| (i.e., h(i-1)=0, h(i)=1 and h(i+1)=0 and p_2=|1|0|1|. According to

Table II, we obtain $m(p_1) = m(p_2) = 0.75$ which represents a relatively good periodic spread pattern. Whereas if we consider p_3=|1|1|0|, we obtain $m(p_3)$= 0.67 meaning that p_3 covers a smaller spread.

TABLE II
MAPPING TABLE FOR $m()$.

$h(i\text{-}1)$	$h(i)$	$h(i\text{+}1)$	$m(\ h(i\text{-}1), h(i), h(i\text{+}1)\ )$
0	0	0	0.00
0	0	1	0.50
1	0	0	0.50
0	1	1	0.67
1	1	0	0.67
0	1	0	0.75
1	0	1	0.75
1	1	1	1.00

Step 3: For each objective, calculate the diversity measure d_m by averaging the m() values.

$$d_m = \frac{\sum_i^{NOG} m(\ h(i-1), h(i), h(i+1)\ )}{NOG} \qquad (2)$$

where NOG stands for the number of grids. To illustrate the procedure to calculate the diversity measure, an example is presented in Figure 3:

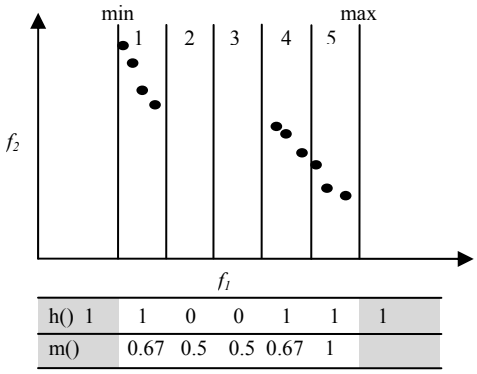

Fig. 3. Example of computing the diversity metric.

In this example, a two-objective (f_1 and f_2) minimization problem is examined. The solutions are marked as points. The $f_2 = 0$ plane is used as the reference plane and the range of f_1 values are divided into, supposing the population size is 10, 10/2 = 5 grids. Then, for each grid, the value of *h()* is calculated based on whether the grid contains a solution point or not. Then, the value of *m()* and the diversity measure are calculated based on a sliding window containing three consecutive grids. The *h()* values of the imaginary boundary grids are always 1 as shown in the shaded grids.

$$d_m(f1) = \frac{0.67 + 0.50 + 0.50 + 0.67 + 1}{5} = 0.668$$

Step 4: Calculate overall diversity performance metric, D_m by averaging the diversity measures of all objective spaces.

$$D_m = \frac{\sum_i^{NOO} d_m(i)}{NOO} \qquad (3)$$

where NOO means the number of objectives.

F. Food Forage Stages Determination

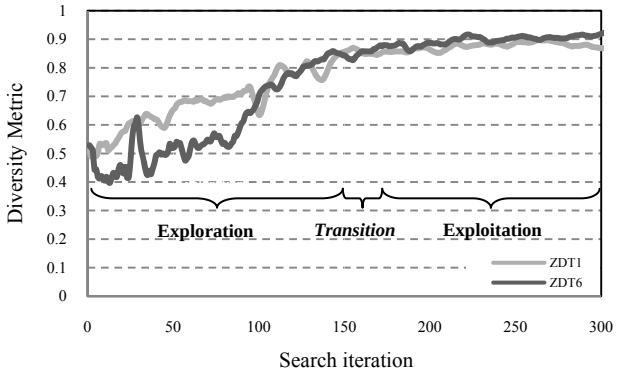

Fig. 4. Diversity metric dynamics for ZDT1 and ZDT6 using A-BCO.

Figure 4 illustrates the diversity metric obtained using A-BCO for the benchmark problems ZDT1 and ZDT6. This metric is used to return feedback about the search space. For ZDT1, from the 150th iteration, the diversity metric oscillates around a value of 0.90. With ZDT6, this diversity metric reaches steady state after 200 generations. In both cases, this convergence indicates that the search has reached global optima solutions. The transition (10 search iterations) and exploitation phases follow.

In general, there are two ways to identify the search stages (which are based upon reaching convergence):

1. The first approach is to define a temporal sliding window and upon the stabilization of diversity within this window, the transition and exploitation stages may initiate. However, this approach may mistakenly return local optima.
2. In A-BCO, we adopt the objective-oriented approach. We arbitrarily set the target transition diversity threshold to be 0.8. Once the diversity reaches 0.8 for at least 10 search iterations, then we assume that the search has converged and we may enter the transition stage. This "transition phase" is employed to prevent returning local optima. Once the search enters the exploitation stage, the number of search iterations is counted.

In A-BCO, the stopping criteria is either achieving α=1.0 or running *g*=200 search iterations during the exploitation stage. However, it is also possible that the Pareto front is discrete where a high diversity metric such α=0.8 is impossible to achieve (this issue is discussed in the experimental section). As a result, we also set a maximum search iteration g_{max} in A-BCO to prevent never-ending searches from occurring.

G. Flying Pattern Adjustment

Elite and follower bees do not follow identical search paths indicated by the waggle dances. Instead, the search path is updated using the simulated binary crossover (SBX) operator. The flying patterns of follower bees are dynamically adjusted to optimize the balance between exploration and exploitation during the different stages of the search. The working principles of SBX are now described to emphasize the importance of the distribution index η_c in generating offspring solutions. Then, we present the self-adaptive

mechanism (SAM) which can dynamically adjust the distribution index in SBX using the feedback information obtained from the diversity-based performance metric.

1) Simulated Binary Crossover (SBX): The SBX crossover operator [14] creates two offspring solutions (represented as real values) from two selected parent solutions. The procedure deriving offspring solutions $x_i^{(a,t+1)}$ and $x_i^{(b,t+1)}$ from the parent solutions $x_i^{(a,t)}$ and $x_i^{(b,t)}$ (t is the generation) is as follow.

A random number $u \in [0,1]$ is generated. Given a pre-specified probability distribution function (Eq. 4), the value of β_i (Eq. 8) is determined so that the area under the probability curve from zero to β_i is equal to u. The distribution index η_c is a non-negative real number. Figure 5 illustrates the probability density function for creating offspring solutions using the SBX operator from two example parents $x_i^{(a,t)}=3$ and $x_i^{(b,t)}=6$ with distribution indexes $\eta_c=2.0$ and $\eta_c=5.0$. Larger values of η_c are more likely to produce "near parent" solutions whereas smaller values of η_c lead to a more diverse search. After obtaining β_i from Eq. 5, the offspring solutions are calculated using Eq. 6 and 7.

$$f(\beta_i) = \begin{cases} 0.5(\eta_c + 1)\beta_i^{\eta_c}, \text{if } \beta_i < 1; \\ 0.5(\eta_c + 1)\dfrac{1}{\beta_i^{\eta_c+2}}, \text{otherwise.} \end{cases} \quad (4)$$

$$\beta_i = \begin{cases} (2u)^{\frac{1}{\eta_c+1}}, & u \le 0.5; \\ \left(\dfrac{1}{2(1-u)}\right)^{\frac{1}{\eta_c+1}}, & u > 0.5. \end{cases} \quad (5)$$

$$x_i^{(a,t+1)} = 0.5\left[(1+\beta_i)x_i^{(a,t)} + (1-\beta_i)x_i^{(b,t)}\right] \quad (6)$$

$$x_i^{(b,t+1)} = 0.5\left[(1-\beta_i)x_i^{(a,t)} + (1+\beta_i)x_i^{(b,t)}\right] \quad (7)$$

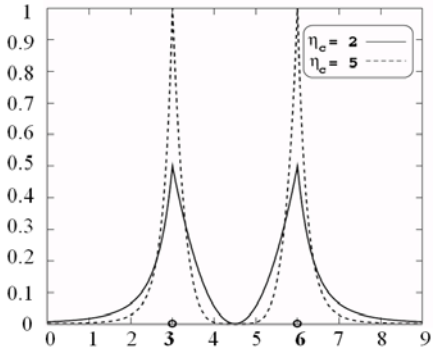

Fig. 5. The probability density function for creating offspring solutions with the SBX operator (adapted from [14]).

2) Self-Adaptive SBX: In most applications of SBX, the crossover distribution index η_c is fixed. Specifically, a fixed value of η_c=2.0 is typically chosen for single-objective optimization problems [16]. Whereas η_c=20.0 is commonly used for ZDT benchmark problem sets. Although using a fixed value of η_c can also lead to the implementation of self-adaptive techniques, past studies using the SBX operator with fixed distribution index could not solve multi-modal problems such as Rastrigin's function [14].

We propose a self-adaptive mechanism (SAM) to dynamically update η_c. Here we assume the optimal diversity performance (i.e., α=1.0) could only be achieved when the solution set is close to the optimal solution set. Hence, if optimal diversity performance is achieved, the distribution index η_c should be large enough to make the offspring solutions very similar to their parents. On the other hand, if the diversity performance is poor, strong crossover operation should be applied to "break" the clusters of solutions. At the beginning stage of the search process, relatively low diversity metric results in strong crossover operation to explore the search space and in the later stage, soft crossover operation is applied to exploit local near-optimal solutions. Thus, this diversity-driven mechanism can effectively exploit the concept of "explore first and exploit later".

The SAM mechanism is now detailed. A reference crossover distribution index η_c based on the diversity performance is computed. The spread β_i of the offspring solutions, with respect to the parent solutions, is obtained using Eq. 8. Based on β_i, the crossover operation can be classified into three classes, namely contracting crossover ($\beta_i < 1$), stationary crossover ($\beta_i = 1$), and expanding crossover ($\beta_i > 1$). The expanding crossover can "expand" the parent solutions to form more diverse offspring candidate solutions. Contracting crossover has the opposite effect of "contracting" the parent solutions. We define the value range of β_i from 0.9 to 1.1 as the close value range (CVR) where the generated offspring solutions are likely to be very similar to the parent solutions. This range was determined based on parametric studies carried out in [17].

$$\beta_i = \left| \frac{x_i^{(2,t+1)} - x_i^{(1,t+1)}}{x_i^{(2,t)} - x_i^{(1,t)}} \right| \quad (8)$$

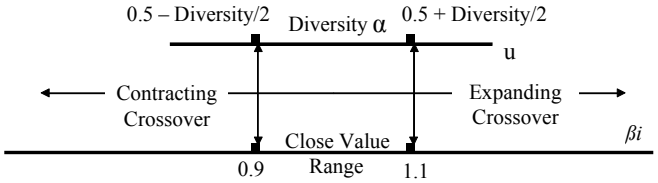

Fig. 6. Mapping between $\boldsymbol{\beta_i}$ and u value in SAM.

Here we determine the reference distribution index η_c so that the probability of β_i falling into the CVR (i.e., $\beta_i \in [0.9,1.1]$) equals to the diversity performance metric as illustrated in Figure 6. For example, if the diversity-based performance metric is α=0.70, then we should make sure that 70% of the time $\beta_i \in [0.9,1.1]$ is true. By mapping the random number u to β_i (using Eq. 5), we have $u \in [0.15,0.85]$. Then η_c can be calculated using Eq. 9 and 10:

$$\eta_c = \begin{cases} = \dfrac{\log 2u}{\log \beta_i} - 1, & u \le 0.5; \quad (9) \\ = -\left(1 + \dfrac{\log 2(1-u)}{\log \boldsymbol{\beta_i}}\right), & u > 0.5. \quad (10) \end{cases}$$

$\eta_c = \frac{\log 2 \times 0.15}{\log 0.9} - 1 = 10.42$ and
$\eta_c = -\left(1 + \frac{\log 2(1-0.85)}{\log 1.1}\right) = 11.63$ respectively.

The distribution indexes $\eta_c = 10.42$ and $\eta_c = 11.63$ are averaged and we obtain a reference crossover distribution index $\eta_c = 11.0$ to produce offspring solutions.

A randomly initialized population typically yields poor diversity performance and consequently lowers the probability of β_i falling into the CVR. During the later stage, the diversity performance stabilizes at a relatively higher value and the exploitation phase may start as the probability of β_i falling in the CVR is higher.

IV. EXPERIMENTS

The benchmark problems ZDT 1, 2, 3, 4, 6 [18] and DTLZ 2, 4 [19] are used to evaluate the performance of A-BCO. The latter is compared with NSGA-II [15] and MOBCO [8]. The following parameter setting is used for NSGA-II: η_c = 20.0, η_m = 50.0, p_c = 1.0, p_m = 1 / (number of variables). The parametric setting for MOBCO is the same as in [8]. Each set of experiments (where 100,000 fitness evaluations are conducted in each of them) is repeated ten times. For A-BCO, we set a population size of 100, archive size K=100, g=200 and g_{max}=1000. Therefore, the maximal number of fitness evaluation in A-BCO is 100,000. Two benchmark metrics, Inverted Generational Distance (IGD) and Spread are employed to measure the algorithms' performance. IGD uses the true Pareto front[1] as a reference and measure the distance of each of the solution points with respect to the front:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \qquad (13)$$

Where d_i is the Euclidean distance between the solutions and the respective closet members to the true Pareto front. n is the number of solutions contained in the Pareto front. When $IGD = 0$, it indicates that the solution set is in the true Pareto front. The Spread indicates the extent of spread among the obtained solutions throughout the Pareto front and is computed as follows.

$$Spread = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}} \qquad (14)$$

Where d_f and d_l are the Euclidean distances between the boundary solutions (of the obtained solution set). d_l is the Euclidean distance between consecutive solution points. Tables IV, V, VI and VII summarize the experimental results. As observed in Tables IV and V, A-BCO achieved lower means and standard deviations for both IGD and Spread diversity metrics in all benchmark problem sets compared to NSGA-II. When compared with MOBCO in Table VI and VII, A-BCO obtained better Spread performance for both means and standard deviation figures. In terms of IGD, A-BCO can still produce comparable results with MOBCO. *Note that no prior parameter-tuning was conducted for the runs using A-BCO.*

[1] The true Pareto front used in these experiments was taken from the jMetal website (http://jmetal.sourceforge.net).

TABLE IV
IGD RESULTS, A-BCO VS. NSGA-II.

Inverted Generational Distance (IGD) Metric				
	A-BCO		NSGA-II	
	Mean	Standard Deviation	Mean	Standard Deviation
ZDT1	**1.57E-04**	**6.70E-06**	1.91E-04	1.08E-05
ZDT2	**1.50E-04**	**2.76E-06**	1.88E-04	8.36E-06
ZDT3	**2.03E-04**	**1.77E-06**	2.59E-04	1.16E-05
ZDT4	**1.48E-04**	**4.59E-06**	1.84E-04	9.86E-06
ZDT6	**1.45E-04**	**1.17E-05**	1.59E-04	1.24E-05
DTLZ2	**3.51E-04**	**2.84E-06**	4.55E-04	3.62E-05
DTLZ4	**8.71E-05**	**5.19E-06**	1.04E-04	8.44E-06

TABLE V
SPREAD RESULTS, A-BCO VS. NSGA-II.

Spread Diversity Metric				
	A-BCO		NSGA-II	
	Mean	Standard Deviation	Mean	Standard Deviation
ZDT1	**1.45E-01**	**5.71E-03**	3.83E-01	3.14E-02
ZDT2	**1.33E-01**	**1.14E-02**	3.52E-01	7.25E-02
ZDT3	**7.04E-01**	**4.60E-03**	7.49E-01	1.49E-02
ZDT4	**1.57E-01**	**9.78E-03**	3.96E-01	2.94E-02
ZDT6	**1.27E-01**	**7.70E-03**	4.80E-01	4.49E-02
DTLZ2	**3.63E-01**	**3.07E-02**	4.80E-01	4.49E-02
DTLZ4	**1.77E-01**	**5.01E-02**	3.87E-01	5.44E-02

TABLE VI
IGD RESULTS, A-BCO VS. MOBCO.

Inverted Generational Distance (IGD) Metric				
	A-BCO		MOBCO	
	Mean	Standard Deviation	Mean	Standard Deviation
ZDT1	**1.57E-04**	6.70E-06	1.73E-04	**6.50E-06**
ZDT2	**1.50E-04**	**2.76E-06**	1.87E-04	1.73E-05
ZDT3	**2.03E-04**	**1.77E-06**	2.33E-04	1.86E-05
ZDT4	**1.48E-04**	**4.59E-06**	1.70E-04	1.92E-05
ZDT6	**1.29E-04**	**3.94E-06**	1.33E-04	5.66E-06
DTLZ2	**3.51E-04**	**2.84E-06**	3.72E-03	1.10E-03
DTLZ4	**8.71E-05**	**5.19E-06**	7.27E-04	1.04E-04

In the original study [8], the population size for MOBCO with ZDT6 was set to 200 and the search duration to 500 iterations. To compare with consistency, we also increase the population size of A-BCO to 200.

TABLE VII
SPREAD RESULTS, A-BCO VS. MOBCO.

Spread				
	A-BCO		MOBCO	
	Mean	Standard Deviation	Mean	Standard Deviation
ZDT1	**1.45E-01**	**5.71E-03**	1.65E-01	1.08E-02
ZDT2	**1.33E-01**	**1.14E-02**	1.88E-01	8.90E-03
ZDT3	**7.04E-01**	**4.60E-03**	7.34E-01	4.06E-03
ZDT4	**1.57E-01**	**9.78E-03**	1.89E-01	1.31E-02
ZDT6	**1.47E-01**	**1.05E-02**	1.51	1.73E-01
DTLZ2	**3.63E-01**	**3.07E-02**	5.94E-01	1.66E-01
DTLZ4	**1.77E-01**	**5.01E-02**	8.84E-01	1.34E-01

In A-BCO, the number of search iterations also varied with respect to the different optimization problems. The average number of search iterations for the benchmark problems is presented in Table VIII. For MOBCO and NSGA-II, the population size was set to 100 and consequently the search

iterations were set to 1000 (i.e., 100,000 fitness evaluations were conducted). In general, the A-BCO runs required a significantly lower number of search iterations when compared with NSGA-II and MOBCO; however, it still achieved better or comparable results with respect to both convergence and diversity.

TABLE VIII
NUMBER OF SEARCH ITERATIONS FOR A-BCO, MOBCO, AND NSGA-II

	Number of Search Iterations		
	A-BCO	MOBCO	NSGA-II
ZDT1	**332**	1000	1000
ZDT2	**349**	1000	1000
ZDT3	**1000**	**1000**	**1000**
ZDT4	**305**	1000	1000
ZDT6	**279**	500	1000
DTLZ2	**268**	1000	1000
DTLZ4	**237**	1000	1000

As shown in Table VIII, the number of search iterations for ZDT3 in A-BCO is the same as the other two. This is due to the discreteness feature of ZDT3's Pareto front. The diversity metrics for ZDT3 using A-BCO is depicted in Figure 7. We observe that the diversity metric stabilizes around the 100th search iteration with $\alpha \approx 0.60$. In this case, the diversity-based performance metric does not reach the threshold value of 0.8. Consequently the search continued until it had reached the maximal number of search iterations g_{max}.

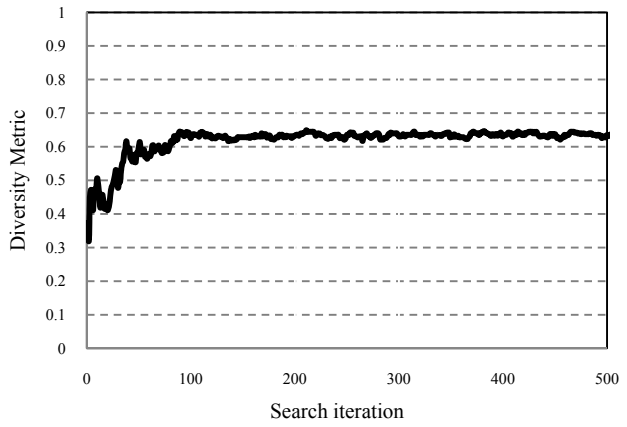

Fig. 7. Diversity metric dynamics for ZDT3 using A-BCO.

V. CONCLUSION

A novel A-BCO for multi-objective numerical functions was proposed. A-BCO's self-adaptive features were driven by a diversity-based performance metric. These self-adaptive features included the bee colony structure and their foraging patterns. This self-adaptation was conducted so as to exploit and optimize the balance between exploration and exploitation. Moreover, the near-optimal number of search iterations was also determined dynamically by A-BCO during the search process. When evaluated over several numerical benchmark problem sets, A-BCO was found to outperform both NSGA-II and MOBCO, without requiring the user to fine-tune the parameters. Finally, we believe that further investigations are required to evaluate A-BCO when applied to real-time problems where the Pareto front is unknown and dynamic.

ACKNOWLEDGMENT

This research is supported by the Defence Science and Technology Agency (DSTA, Singapore) grant POD0814214.

REFERENCES

[1] D.S. Liu, K.C. Tan, C.K. Goh, W.K. Ho, "A Multiobjective Memetic Algorithm Based on Particle Swarm Optimization," *IEEE Transactions on Systems, Man, and Cybernetics*, Part B: Cybernetics, vol. 37, , Feb. 2007, pp. 42-50.

[2] D.S. Liu, K.C. Tan, S.Y. Huang, C.K. Goh, W.K. Ho, "On Solving Multi-objective Bin Packing Problems Using Evolutionary Particle Swarm Optimization," *European Journal of Operational Research*, vol. 190, , October. 2008, pp. 375-382.

[3] R. Akbari, A. Mhammadi, K. Ziarati, "A Novel Bee Swarm Optimization Algorithm For Numerical Function Optimization," *Communications in Nonlinear Science and Numerical Simulation*, doi: 10.1016/j. cnsns.2009.11.003.

[4] D. Karaboga, B. Basturk, "A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm," *Journal of Global Optimization*, vol. 39, Nov. 2007, pp. 459-471.

[5] D. Karaboga, B. Akay, "A Survey: Algorithms Simulating Bee Swarm Intelligence," *Artificial Intelligence Review*, vol. 31, Jun.2009, pp. 64-85.

[6] K. Sundareswaran, VT Sreedevi, "Development of Novel Optimization Procedure Based on Honey Bee Foraging Behavior," *IEEE International conference on systems, man and cybernetics*, 2008, pp. 1220 - 1225.

[7] D. Pham and A. Ghanbarzadeha, "Multi-Objective Optimisation using the Bees Algorithm", *In proceedings of the 3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007)*, Whittles, Dunbeath, Scotland, 2007.

[8] M. Low, M. Chandramohan and C.S. Choo, "Application of Multi-Objective Bee Colony Optimization Algorithm to Automated Red Teaming," *In Proceedings of the 2009 Winter Simulation Conference*, pp 1798-1808.

[9] K.C. Tan, S.C. Chaim, A.A. Mamun, C.K. Goh, "Balancing Exploration and Exploitation with Adaptive Variation for Evolutionary Multi-objective Optimization," *European Journal of Operational Research*, vol. 197, Sep. 2009, pp. 701-713.

[10] V. Frisch, "Decoding the Language of the Bee," *Science*, vol. 185, 1974, pp. 663-668.

[11] F.C Dyer, "The Biology of the Dance Language," *Annual Review of Entomology*, vol. 47, 2002, pp. 917-949.

[12] J.C. Biesmeijer, T. D. Seeley, "The Use of Waggle Dance Information by Honey Bees throughout Their Foraging Careers," *Behavioral Ecology and Sociobiology*, vol. 59(1), 2005, pp. 133-142.

[13] K. Deb, S. Jain, "Running Performance Metrics for Evolutionary Multi-Objective Optimization," *In Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, volume 1, 2002, pp. 13-20.

[14] K. Deb, S. Agrawal, "Simulated Binary Crossover for Continuous Search Space," *Complex System*, 9(2), 1995, pp. 115-148.

[15] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, "A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, 6(2), 2002, pp. 182-197.

[16] K. Deb, A. Anand, D. Joshi, "A Computationally Efficient Evolutionary Algorithm for Real-parameter Optimization," *Evolutionary Computation Journal*, 10(4), 2002, pp. 371-395.

[17] F.C. Zeng, M. Y. H. Low, J. Decraene, S. Zhou, W. Cai, "Self-Adaptive Mechanism for Multi-objective Evolutionary Algorithms," International *MultiConference of Engineers and Computer Scientists 2010*, IAENG, pp. 7-12.

[18] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, 8(2), 2000, pp. 173-195.

[19] K. Deb, L. Thiele, M. Laumanns and E. Zitzler, "Scalable Multi-Objective Optimization Test Problems," *IEEE Congress on Evolutionary Computation 2002, IEEE Press*, pp. 825-830.