Bees algorithm for multimodal function optimisation

Zhou, Zuokuan; Xie, Yong; Pham, Duc; Kamsani, Silah; Castellani, Marco

DOI:
10.1177/0954406215576063

Document Version
Peer reviewed version

Citation for published version (Harvard):
Zhou, Z, Xie, Y, Pham, D, Kamsani, S & Castellani, M 2015, 'Bees algorithm for multimodal function optimisation', Institution of Mechanical Engineers. Proceedings. Part C: Journal of Mechanical Engineering Science . https://doi.org/10.1177/0954406215576063

Bees Algorithm for Multimodal Function Optimisation

Z. D. Zhou[1], Y. Q. Xie[1*], D. T. Pham[2], S. Kamsani[2], M. Castellani[2]
[1]School of Information Engineering, Wuhan University of Technology, China
[2]School of Mechanical Engineering, University of Birmingham, U.K

Abstract: The aim of multimodal optimisation (MMO) is to find significant optima of a multimodal objective function including its global optimum. Many real-world applications are MMO problems requiring multiple optimal solutions. The Bees Algorithm (BA) is a global optimisation procedure inspired by the foraging behaviour of honeybees. In this paper, several procedures are introduced to enhance the algorithm's capability to find multiple optima in MMO problems. In the proposed Bees Algorithm for MMO, dynamic colony size is permitted to automatically adapt the search effort to different objective functions. A local search approach called balanced search technique (BST) is also proposed to speed up the algorithm. In addition, two procedures of radius estimation and optima elitism are added, to respectively enhance the Bees Algorithm's ability to locate unevenly distributed optima, and eliminate insignificant local optima. The performance of the modified Bees Algorithm is evaluated on well-known benchmark problems, and the results are compared with those obtained by several other state-of-the-art algorithms. The results indicate that the proposed algorithm inherits excellent properties from the standard Bees Algorithm, obtaining notable efficiency for solving MMO problems due to the introduced modifications.

Keywords: Swarm-based algorithms, multimodal optimisation, Bees Algorithm, balanced search, hill valley

* Corresponding author: xyqwhut0728@126.com

Nomenclature

MMO: multimodal optimisation

BA: Bees Algorithm

BST: balanced search technique

SOA: swarm-based optimisation algorithm

EA: Evolutionary Algorithm

GA: Genetic Algorithm

PSO: Particle Swarm Optimisation

IWO: Invasive Weed Optimisation

HV: hill valley

DE: Differential Evolution

SDE: Species-based Differential Evolution

1. Introduction

Swarm-based optimisation algorithms (SOAs) are usually employed to find the global solution to an optimisation problem, discarding any alternative solution of equal or comparable fitness. However, in a multimodal optimisation (MMO) task, the main purpose is to find multiple optimal solutions [1]. MMO problems are gaining increasing attention due to their frequent occurrence in scientific and engineering applications, such as object detection in machine vision, parameter tuning in varied-line-spacing holographic grating design, and protein structure prediction. MMO applies to those problems which have more than one global optimum in the feasible solution space, or one global optimum and several

local optima. As they represent alternative solutions, it is sometimes desirable to locate all the significant optima of a given fitness landscape. In addition, the knowledge of multiple optimal solutions may provide useful insight into the problem domain. A similarity analysis of multiple optimal solutions may bring about helpful innovative and hidden principles, similar to what is often observed in Pareto-optimal solutions in a multi-objective problem solving task [2].

Investigation of the performance of SOAs for MMO problems has been receiving growing interest in the SOA community. Evolutionary algorithms (EAs) are widely used to solve MMO problems due to their population-based searching ability. Niching, clustering, and speciation methods have been used to distribute the EA population on different peaks in the search region. Similarly, several modified versions of Particle Swarm Optimisation (PSO) and Invasive Weed Optimisation (IWO) have been used to search multiple optimal solutions. More details about methods for solving MMO problems are presented in the second section.

First proposed by Pham [3], the Bees Algorithm is a SOA that mimics the foraging behaviour of honey bees, a species which has been successfully surviving for hundreds of thousands of years in various kinds of natural environments. This paper will introduce several modifications to the basic Bees Algorithm with the aim to find multiple optimal solutions simultaneously in a single run. The modifications are necessary since the basic Bees Algorithm is designed to find only one optimum. First of all, unlike some SOAs that use a predefined clustering radius (or parameter with the similar function) for the peaks in the fitness landscapes, the proposed algorithm estimates the radius using an amended hill valley (HV) method. The second modification introduces a local search operator named balanced

search technique (BST) to search for the solutions of highest fitness in a fitness peak. Some algorithms calculate the gradient of the objective function. This can be very helpful in simulation but unfortunately many real world problems are non-differentiable. Compared to the purely random local search in the basic Bees Algorithm, this modification aims at improving the algorithm's search speed. Furthermore, the algorithm allows for variable colony size. That is, the population size in each generation is allowed to increase if more optima are detected, or decrease if only a small number of optima exist in the objective function. This is biologically plausible, since biological bees optimise the number of harvesting bees according to the abundance of food sources (i.e. nectar).

The remainder of this paper is organised as follows. In Section 2, a review is provided of related SOAs for solving MMO problems. Section 3 outlines the basic Bees Algorithm. The modifications introduced to perform MMO search are explained in Section 4. Thereafter, the individual impact of each of the new features on the search capability of the algorithm is highlighted in Section 5. In Section 6, the experimental results of the proposed algorithm and comparisons are presented. Finally, Section 7 concludes the paper and suggests topics for future work.

2. Related work

When applying SOAs to MMO problems, it is very important to consider two apparently contradictory requirements: preserving promising individuals from one generation to the next and maintaining the diversity of the population [4]. This section briefly reviews some recently developed techniques to address the above trade-off.

De Jong tried to solve the MMO problem using an EA for the first time in 1975 [5]. He used population crowding. Crowding encourages population diversity by eliminating from the parent population those individuals which are most similar to the offspring. Fitness sharing was proposed by Goldberg and Richardson in 1987 [6] to increase the chance of locating multiple optima. Instead of using an absolute fitness function, they designed a shared function which takes into account the genotypic or phenotypic similarity of the individuals. Since then, an increasing number of researchers explored different ways to deal with the population diversity problem. These methods include species conservation, pre-selection, elitism, and clearing.

The adaptive elitist-population search method was used in a Genetic Algorithm (GA) for MMO [7-8]. Vitela and Castanos proposed a sequential niching algorithm for MMO. They combined hill-climbing, a derating function, niching and clearing techniques within a GA for a multiple optima search [9]. In the literature [10-12], authors discussed a clustering genetic algorithm based on dynamic niching with niche migration. They studied the niching method intensively and claimed very little priori knowledge is required to determine the radius and the number of niches. Other techniques as the distance measuring method [13] and memetic algorithms [14] were used within a GA for MMO purpose.

Particle Swarm Optimisation (PSO) is another prominent member of the SOA family, and is now receiving a great deal of interests for MMO purpose. A memetic algorithm, along with a local search operator, was hybridised with PSO by Wang [15] for MMO. This hybrid PSO obtained excellent performance. Likewise, other practitioners [16-22] hybridised PSO with niching and clustering techniques to design different population topologies or fitness

evaluation methods to obtain several multiple optimal solutions. Some latest studies related to PSO for MMO are summarised in [23].

Li [24] utilised a SOA for determining species in conjunction with a basic Differential Evolution (DE) procedure named Species-based Differential Evolution (SDE). In [25], the principle of locality, a widely used concept in computing, was incorporated with differential evolution for MMO. Spatial locality and temporal locality were adopted in the proposed methods. Other learning algorithms such as artificial weed colony optimisation are continuously investigated for MMO [26-28].

3. The basic Bees Algorithm

The basic Bees Algorithm is inspired by the foraging behaviour of honeybees in nature, and was designed to search for the best solution to a given optimisation problem. A solution in the search space is thought of as a nectar source. Scout bees randomly sample the solution space and appraise the quality of the visited locations through the fitness function. Foragers are recruited to exploit the most promising *m* locations found by the scout bees. Each scout directs a number of foragers to the neighbourhood of the solutions found. The scouts that found the *e* top-rated locations recruit *nre* foragers, the scouts that found the remaining *m-e* most promising solutions recruit $nrb < nre$ foragers. The neighbourhood of a solution is regarded as a 'flower patch'. Overall, the original Bees Algorithm employs a combination of local exploitative and global exploratory search techniques. For the global search, scout bees are sent to random points of the search space to look for potential solutions. For the local search, foragers are sent to the neighbourhood of the most favourable solutions.

The parameters need to be set for the basic algorithm are: the number of scout bees (*n*); the number of patches selected for the local search (*m*); the number of top-rated patches (elite) in selected patches (*e*); the number of foragers recruited for the top patches (*nre*); the number of foragers to be recruited for the other selected patches (*nrb*); the initial size of each patch (*ngh*); and finally the stopping criteria. Since its original formulation, the Bees Algorithm has undergone many variations [29-34], and for their applications reader can refer to [35-39].

4. The proposed modified Bees Algorithm

The proposed algorithm includes a number of modifications to the basic Bees Algorithm to find multiple satisfactory solutions to an objective problem in a single run. This section details the proposed algorithm and presents the modifications. Without loss generality, it will be assumed in the rest of this paper that the optimisation problem requires the maximisations of a given fitness function.

4.1 Main procedures of the proposed algorithm

Before detailing the proposed algorithm, a number of terms will first be defined.

Definition 1. Field: a field defines an area in the search space that may potentially cover a fitness peak. It helps to differentiate one peak from another. Each field should cover one and only one fitness peak ideally. The size of a field is determined by its radius (refer to Definition. 3). Each field contains at least one scout.

Definition 2. Field centre: the field centre is the location of highest fitness found so far by a scout bee in the field. If there is only one scout in that field, its location automatically becomes the field centre.

Definition 3. Field radius: the field radius is the Euclidean distance from the field centre

to the border. It determines the size of field.

Definition 4. Neighbourhood: the neighbourhood is used to constrain the range for a refined local search within a field to locate the local optima. Only within the neighbourhood scouts and foragers are allowed to land. To enhance the search accuracy, the size of the neighbourhood shrinks if the search stagnates in a field.

The main body of the proposed algorithm is designed based on the framework of the basic version. Figure 1 summarises the proposed algorithm, and Table 1 lists the parameters to be initialised in Step 1. In Step 3, fields are allowed to merge or split according to the distance between them or the distribution of scouts. The fields' radii are updated through an estimation procedure. Local search is performed in Step 4 to look for the optimal solutions on detected fitness peaks. A balanced search strategy is employed in this step to enhance the search speed. This is followed by global search in Step 5. The global search tries to detect potential fitness peaks that have not been identified in the search space. The hill valley method is implemented in global search to enhance the possibility of finding unidentified peaks.

Input: Objective function

Step 1: Initialisation

Step 2: While (stopping criteria not met)

Step 3: Fields update

Step 4: Local search (neighbourhood search)

Step 5: Global search

Step 6: End while

Output: Optimal solutions

Figure 1. Pseudo code of main body of the proposed algorithm

Table 1. Parameters of the Bees Algorithm for MMO

ns	initial number of scouts in Fields
nr	initial number of random scouts
nre	number of foragers recruited by the scout at the field centre
nrb	number of foragers recruited by other scouts in a field (*nre*> *nrb*)
ngh	initial neighbourhood size
rad	initial field radius
stlim	limit of cycles to determine the stagnation of a field

4.2 Field update

Fields are allowed to merge and split when the certain conditions are met, and the radius of each field is made adaptive to the objective problem through an estimation procedure.

4.2.1 Distance between fields

Let P_i and P_j be two fields with radius R_i and R_j, and C_i and C_j be the respective centre. The distance between P_i and P_j is the Euclidean distance between C_i and C_j calculated as Equation (1):

$$Dis(P_i, P_j) = \sqrt{\left\| pos(C_i) - pos(C_j) \right\|^2} \quad (1)$$

where $pos(\cdot)$ returns the position of a specified point in the search space.

4.2.2 Rule for merging fields

P_i and P_j are allowed to merge when Relation (2) is satisfied:

$$Dis(P_i, P_j) < 0.7 \times (R_i + R_j) \quad (2)$$

Let P_k, C_k, R_k denote respectively the newly formed field, its centre and radius, then C_k is the one selected from C_i and C_j which has higher fitness, and R_k is the larger one of R_i and R_j. All the scouts of P_i and P_j are moved into the common field P_k. A restriction on the number of scouts in a field is placed since a field cannot sustain an overly dense population. If the number of scouts in a field exceeds a predefined upper limit (3 in this paper), only those at the fittest positions will be kept for the next generation, whilst the others are regarded as redundant and transferred to global search.

4.2.3 Rule for splitting a field

A field P_i is allowed to split when Relation (3) is satisfied:

$$Dis(C_i, S_i) > 1.4 \times R_i \tag{3}$$

where $S_i \in \{\text{scouts in } P_i \mid S_i \text{ is not at } C_i\}$ is one of the scouts in P_i. Each field gets only one chance to split in an evolving iteration. Denoting the two children fields as P_m and P_n, C_m, C_n, R_m, R_n are updated as $C_m = C_i$, $C_n = S_i$ and $R_m = R_n = 0.7 \times R_i$, and P_m inherits all the scouts from P_i except S_i.

4.2.4 Field radius estimation

The field radius is updated through a radius estimation algorithm, where an amended hill valley (HV) [40] is applied. Figure 2 helps to explain how the amended HV works on a one dimensional function. Nevertheless, the validity of the procedure extends to any dimensionality of the search space.

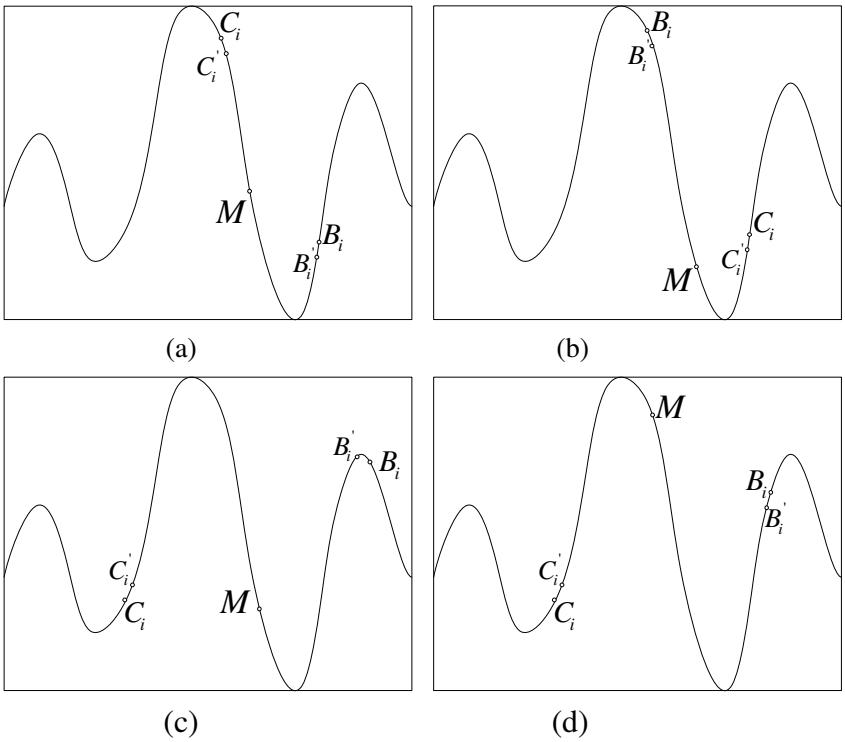


Figure 2. (a) (b) and (c) demonstrates the conditions (1), (2), and (3)respectively, (d) demonstrates a valley being omitted.

Figure 2 illustrates four possible cases in field radius estimation. Let C_i and R_i be the centre and estimated radius of the current field P_i. A particular bee B_i is created and sent to the position obtained by Equation (4).

$$pos(B_i) = pos(C_i) + D_r \cdot R_i \tag{4}$$

where D_r symbolises a normalised random direction, Then three sample points $C_i^{'}$, $B_i^{'}$ and M are created according to Equations (5).

$$\begin{cases} pos(C_i^{'}) = pos(C_i) + \delta \cdot [pos(B_i) - pos(C_i)] \\ pos(B_i^{'}) = pos(B_i) + \delta \cdot [pos(C_i) - pos(B_i)] \\ pos(M) = pos(C_i) + \dfrac{1}{2} \cdot [pos(B_i) - pos(C_i)] \end{cases} \tag{5}$$

where δ is an absolute small positive value. The basic motivation underneath the radius estimation is: if a valley is detected between B_i and C_i, the current radius ought to be reduced,

otherwise it should be increased. A valley is said to exist if at least one of the following conditions is met:

(1) $fitness(B_i^{'}) < \min\{fitness(B_i), fitness(C_i)\}$, as shown in Figure 2(a);

(2) $fitness(C_i^{'}) < \min\{fitness(B_i), fitness(C_i)\}$, as shown in Figure 2(b);

(3) $fitness(M) < \min\{fitness(B_i), fitness(C_i)\}$, as shown in Figure 2(c).

where the function $fitness(\cdot)$ returns the fitness value of a sampled position. Otherwise, it is assumed there is no valley between B_i and C_i. Figure 2(d) shows a case in which the conditions (1), (2) and (3) are not satisfied but actually a valley does exist between B_i and C_i. The HV method omits a valley between two end points with a probability inversely proportional to the number of adopted sample points. To reduce the chance of such an event, the amended HV is repeated k ($k > 0$) times (k equals to the dimension of the objective function in this paper) before determining the radius.

Figure 3 shows the pseudo code of field radius estimation algorithm where α $(0 < \alpha < 1)$ is introduced to control the radius alterability. Throughout this paper it is kept constant to $\alpha = 0.2$.

Input: current radius $R_i(t)$, k

Step 1: Send B_i according to Equation (4), $k = k$-1;

Step 2: Produce sample points according to Equations (5);

Step 3: Determine whether a valley exists according to conditions (1), (2) and (3),

if a valley exists, go to Step 4,

if a valley does not exist and $k = 0$, go to Step 5,

otherwise go to Step 1;

Step 4: $R_i(t+1) \leftarrow (1-\alpha) \cdot R_i(t)$, return;

Step 5: $R_i(t+1) \leftarrow (1+\alpha) \cdot R_i(t)$.

Output: updated radius $R_i(t+1)$

Figure 3. The pseudo code of field radius estimation

4.3 Local search

The basic Bees Algorithm and most of its variants implement local search in a neighbourhood using a random operator, as expressed by Equations (6).

$$\begin{cases} pos(F^j(t)) = pos(S(t)) + ngh(t) \cdot r_1 \\ fitness(S(t+1)) = \max\limits_{j=1,2,\ldots,n_F} \{fitness(S(t)),\ fitness(F^j(t))\} \end{cases} \tag{6}$$

where $F^j(t)$ stands for the jth forager recruited by the scout $S(t)$ in generation t, n_F denotes the number of foragers recruited by S, r_1 is a uniformly distributed random value in $(-1,1)$. The function $\max\{\cdot\}$ embodies the greedy selection strategy adopted in the local search, that is the scout is replaced by the recruited forager if the forager is landing at a position of higher fitness than the scout. The balanced search technique (BST) is developed to speed up the algorithm as described below.

4.3.1 Obtaining the guide

A gradient-like vector is obtained as a search guide for the recruited foragers. It does not require the function to be differentiable. The guide is calculated as follows:

$$G_i(t) = pos(C_i(t)) - pos(C_i(t-1)) \tag{7}$$

where $G_i(t)$ denote the guide of the field P_i in generation t. Equation (7) indicates that the guide for the current iteration depends on the position of the field centre in the last two iterations. The guide is thereafter normalised by dividing it by the norm $\| G_i(t) \|$ as in Equation (8). A search conducted by this guide is called guided search.

$$\hat{G}_i(t) = \frac{G_i(t)}{\| G_i(t) \|}, \text{if } G_i(t) \neq 0 \tag{8}$$

4.3.2 Formulating the balanced search

The basic principle of BST is to keep a balance between random and guided search, as shown in Equations (9).

$$\begin{cases} pos(F_i^j(t)) = pos(C_i(t)) + (\mu_i(t) \cdot \hat{G}_i(t) + (1-\mu_i(t)) \cdot D_r) \cdot ngh_i(t) \cdot r_2 \\ fitness(C_i(t+1)) = \max\limits_{j=1,2,\ldots,n_F} \{fitness(C_i(t)), fitness(F_i^j(t))\} \end{cases} \tag{9}$$

in which D_r symbolises a normalised random direction, r_2 is a uniformly distributed random value in $(0,1)$, and $\mu_i(t)$ is introduced as an adaptive weight to balance the influence of the two local search operators. The larger $\mu_i(t)$ is, the more local search depends on guided search. Conversely, the local search will rely more on randomness.

4.3.3 Updating the balance weight

The weight $\mu_i(t)$ is updated according to Equations (10).

$$\mu_i(t+1) = \begin{cases} 1.2 \times \mu_i(t), |\, angle(\hat{G}_i(t), \hat{G}_i(t-1)) \,| \leq \theta_1 \\ 0.8 \times \mu_i(t), |\, angle(\hat{G}_i(t), \hat{G}_i(t-1)) \,| \geq \theta_2 \\ \mu_i(t), other. \end{cases} \tag{10}$$

subject to $0 < \mu_i(t) \leq 1$. θ_1 and θ_2 are two thresholds that determine the size of $\mu_i(t)$, and the function *angle*(·) returns the angle between two vectors. Equations (10) indicate that if improvements in fitness are obtained in consecutive iterations without altering substantially the direction of the guide, μ_i will keep growing until it reaches its upper limit. According to Equations (9), an increase of μ_i will result in the dominance of the guided search. On the contrary, if the angle between two successive guides exceeds the threshold θ_2, μ_i will gradually decrease and then random search will take dominance.

4.3.4 Preserving stagnant fields

A field is considered stagnant if no improvement can be obtained after a predefined number of evolutionary cycles (*stlim*). Instead of abandoning the stagnated field, the proposed algorithm records the information of the field including its centre and radius. The position of the field centre is one of the optimal solutions located by the algorithm. All the scouts in the stagnated field except the one at the centre are released and become random scouts, so the search in this field is terminated.

4.4 Global search

Global search focuses on yet unknown areas of the solution space. It is initially carried out by a predefined number of scouts, called random scouts. To guarantee that a fitness peak discovered by a random scout has never been searched before, the regional evolution and the standard hill valley are combined. The regional evolution is used to prevent that a random scout which just began to climb a fitness peak is being neglected due to its current low fitness. A random scout is allowed to evolve a few times (equals to the dimension of the objective function in this paper) before competing with those scouts at the field centre. Normally, the HV would incur a fast increase of function evaluations, since it has to evaluate the fitness values of a number of sample points. In the proposed algorithm, this increase is restrained by restricting HV only to the fields in the vicinity of the peak under consideration. When a random scout discovers a new promising region in the fitness landscape, a new field is generated around this random scout. This random scout therefore become a scout in fields and is involved in the local search in the next iteration. The global search process consists essentially of the following steps in Figure 4:

Input: current fields, random scouts

Step 1: send a scouts randomly to the search space;

Step 2: implement the regional evolution;

Step 3: implement the HV to determine whether the random scout has found a new peak in fitness landscape;

Step 4: if a new fitness peak is identified, a new field covering this peak is formed and inserted into current fields

Step 5: go to Step 1 until all the random scouts are sent out

Output: updated fields

Figure 4. The pseudo code of global search

4.5 The mechanism of variable colony size

In the proposed algorithm, the variable colony size is achieved by transferring part of the scouts between local search and global search, and setting a range for the number of random scouts.

The rule governing how fields are merged has set a restriction on the number of scouts in a field. If the number of scouts grows above a specified level in a field, some scouts landing at low-fitness positions will be released and added to the random scouts. The number of random scouts is hence increased. However it cannot exceed an upper boundary, otherwise some surplus random scouts will be removed from the colony.

In the global search process, the random scouts that have discovered a potential fitness peak are becoming the scouts in fields and will be involved in the local search process in the following evolutionary iteration. In this situation the number of scouts in fields increases

while the random scouts decrease. A few random scouts will be created by the algorithm if the number of current random scouts falls below the lower boundary. The colony size is therefore increased. Through this way the algorithm ensures the number of random scouts falls within the allowed boundaries. Figure 5 helps explain the mechanism of the variable population size in a colony.

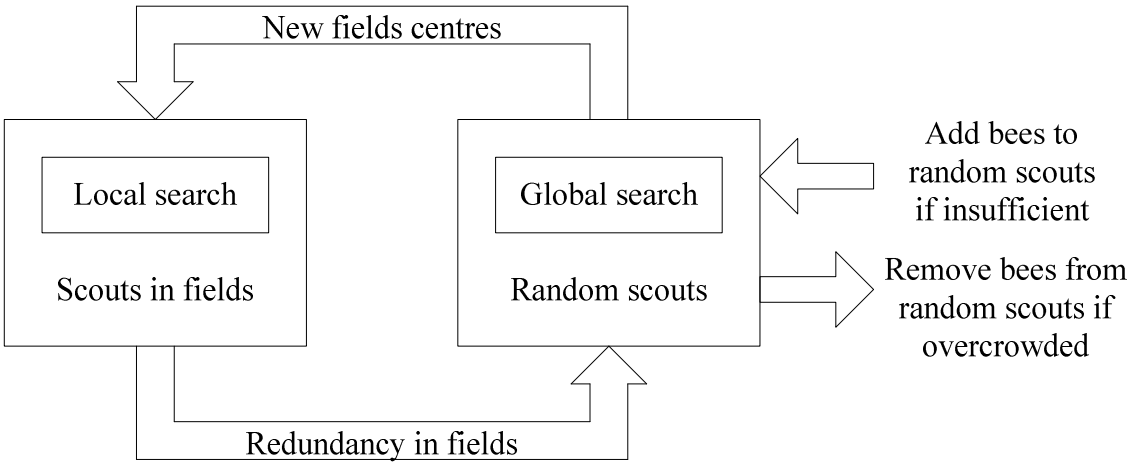


Figure 5. Variable population size in a colony

5. Effects of new features used

5.1 Evolution of the colony size

Two functions are used here to demonstrate how the colony size evolves during the optimisation process.

(1) Deb's function:

$$f_1(x) = \sin^6(5\pi x) \tag{11}$$

where $0 \le x \le 1$ and the five global maximal at $x = 0.1$, $x = 0.3$, $x = 0.5$, $x = 0.7$, and $x = 0.9$ respectively.

(2) Two dimensional multimodal function:

$$f_2(x) = x_1 \sin(4\pi x_1) - x_2 \sin(4\pi x_2 + \pi) + 1 \qquad (12)$$

where $-2 \le x_1, x_2 \le 2$. There are totally 100 optimal solutions (including local optima).

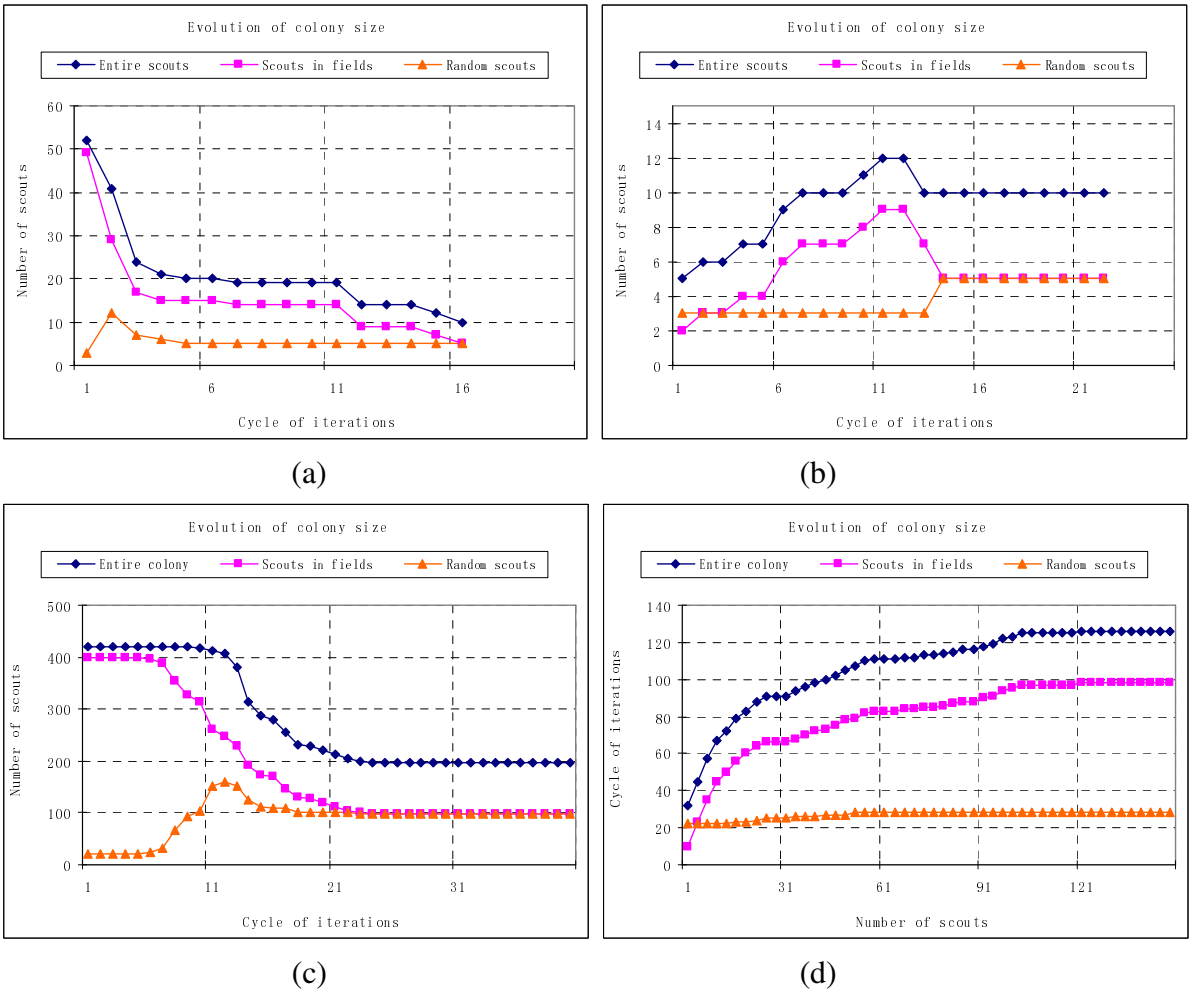


Figure 6. Evolution of the colony size during optimisation: in (a) and (b) the algorithm starts with a large and small population size respectively to search the fitness landscape of function (1); in (c) and (d) the algorithm starts with a large and small population size respectively to search the fitness landscape of function (2).

In the case of function (1), the algorithm starts with 50 field scouts and 5 random scouts, and then with 2 field scouts and 3 random scouts. As can be seen in Figure 6(a), the entire population drops dramatically when the size of the initial colony is unnecessarily large to find the five optima in the solution space. The number of random scout grows at the beginning

because the redundant scouts in fields are transferred to global search, and then declines due to upper boundary of the number of random scouts. On the contrary, when the colony size is insufficient for finding many optima at the beginning of the search (Figure 6(b)), new members are gradually added to the colony so the colony size keep growing until enough for exploring the fitness peaks. The developmental pattern in Figure 6(c) basically matches that of 6(a), and the pattern in Figure 6(d) with 6(b). The two cases show that the population in an evolving colony is able to adapt to the task, and finally run in a relatively stable state.

5.2 Effects of BST

Also, a set of well-known functions are used to demonstrate the effects of BST, as given in Table 2.

Table 2. Functions used for evaluating the effects of BST

(1) Deb's function (5 optima):

$f_1(x) = \sin^6(5\pi x), \quad x \in [0, 1]$;

(2) Deb's decreasing function (5 optima):

$f_2(x) = 2^{-2((x-0.1)/0.9)^2} \sin^6(5\pi x), \quad x \in [0, 1]$;

(3) Roots function (6 optima):

$f_3(x) = \frac{1}{1+|x^6-1|}, \quad x \in C, \quad x = x_1 + ix_2 \in [-2, 2]$;

(4) Two dimensional multimodal function (100 optima):

$f_4(x) = x_1 \sin(4\pi x_1) - x_2 \sin(4\pi x_2 + \pi) + 1, \quad x_1, x_2 \in [-2, 2]$;

(5) Eight dimensional multimodal function (256 optima):

$f_5(x_1, x_2, ..., x_8) = \frac{\sum_{i=1}^{8} |\sin(2\pi(1-x_i)^{3/5})|}{8}, \quad x_i \in [0, 1], i = 1,2,...,8$.

For each function, the algorithm starts with the same parameter configurations. For problem (1) and (2), the colony starts with 10 field scouts and 5 random scouts. The search

accuracy is set to be 0.000001. For function (3), (4) and (5), these configurations are 20, 10, 0.0001; 50, 30 0.001 and 130, 85, 0.01 respectively. Table 3 compares the results of the algorithms using random and balanced local search in terms of function evaluation, iteration cycles and the number of optima found. The function evaluation is the primary criterion for comparing the performance of the various algorithms. The statistical significance of the difference between the results is evaluated through student's *t*-tests. The *t*-tests are run with a confidence level of 95% and the *p*-values are listed in Table 4. The *p*-value below the significance level signal (0.05) indicates a statistically significant difference between the results obtained by the two algorithms.

. Table 3 shows that the BST allows the Bees Algorithm to find the optimal solutions using about 13.8%, 3.4%, 43%, 13.0% less function evaluations than the random local search based Bees Algorithm on functions (1), (3), (4), (5) respectively, and the corresponding *p*-values of 0.0014, 0.0051, 0.0001 0.0001 in Table 4 are below the acceptance value 0.05, suggesting that the improvement is statistically significant. For function (2), The Bees Algorithm using BST needs 8.2% less function evaluations but the *p*-value indicates the improvement is not significant. The data shows both algorithms are successful in finding all the optima. In these experiments, the BST based Bees Algorithm requires less function evaluations and less iteration cycles on the majority of functions than the basic Bees Algorithm. Therefore BST can be considered as a promising local search approach to speed up the algorithm.

Table 3. Comparison betweem random local search and BST

Test function	Local search method	Function evaluations(Std.)	Iteration cycles(Std.)	Optima found(Std.)

(1) Deb's function	Random	1,196(272)	19.3(5.3)	**5**(0)
	BST	**1,031**(228)	**16.2**(4.7)	**5**(0)
(2) Deb's decreasing function	Random	1,162(312)	19.5(6.8)	**5**(0)
	BST	**1,067**(254)	**17.6**(5.1)	**5**(0)
(3) Roots function	Random	10,310(619)	36.4(8.3)	**5.8**(0.4)
	BST	**9,960**(603)	**31.2**(8.1)	**5.8**(0.4)
(4) Multimodal function (2D)	Random	77,816(1,0142)	59.9(17.3)	94.9(1.75)
	BST	**44,037**(6,520)	**31.6**(5.3)	**96.7**(1.7)
(5) Multimodal function (8D)	Random	836,823(68,345)	49.2(4.38)	247.2(7.8)
	BST	**727,777**(66,564)	**47.6**(5.94)	**250.8**(6.7)

Table 4. *p*-values showing statistical significance

Test function	Function evaluations	Iteration cycles	Optima found
(1) Deb's function	**0.0014**	**0.0026**	-
(2) Deb's decreasing function	0.0982	0.1172	-
(3) Roots function	**0.0051**	**0.0020**	-
(4) Multimodal function (2D)	**<0.0001**	**<0.0001**	**<0.0001**
(5) Multimodal function (8D)	**<0.0001**	0.1285	**0.0150**

5.3 Evolution of field radius

Three benchmark functions are selected to demonstrate how the field radius evolves as the algorithm proceeds. For visualisation purposes, two benchmarks are chosen to be two dimensional (as plotted in Figure 7) whilst the other one is four dimensional, (Table 5).

Table 5. Functions used for demonstration of field radius evolvement

(1) Inverted Rastrigin (9 optima)

$$f_1(x) = -20 - \sum_{i=1}^{2} x_i^2 - 10 \cdot \cos(2\pi x_i), \quad -1.5 \le x_i \le 1.5;$$

(2) Five hills (Ursem function, 5 optima)

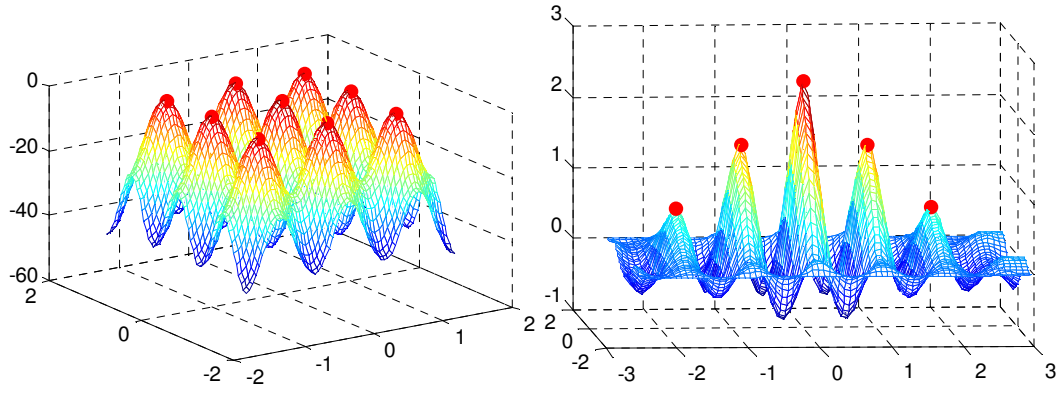
$$f_2(x) = \sin(2.2\pi x_1 + 0.5\pi) \cdot \frac{2 - |x_2|}{2} \cdot \frac{3 - |x_1|}{2} + \sin(0.5\pi x_2^2 + 0.5\pi) \cdot \frac{2 - |x_2|}{2} \cdot \frac{3 - |x_1|}{2},$$

$-2.5 \le x_1 \le 1.5, -2 \le x_2 \le 2$;

(3) Four dimensional multimodal function (16 optima)

$$f_3(x_1, x_2, ..x_3) = \frac{\sum_{i=1}^{3} |\sin(2\pi(1-x_i)^{3/5})|}{3}, \quad x_i \in [0,1], i = 1,2,3.$$

For each function, the algorithm is executed several times using different initial field radii, whilst all the other parameters are kept the same. For function (1), the radius is initialized at 1.5 and 0.1. For function (2) and (3), it is set to respectively 1.0 and 0.2, and 0.3 and 0.03. Figure 8 shows how the radius evolves as the search proceeds. The plots in Figure 8 show two trends, which can be summarised as follows: the field radius self-adapts at the beginning, then reaches a relatively steady value, and finally fluctuates around this value. Figures 8 (e) and (f) show that the field radius estimation method is also applicable to multi-dimensional problems. From this group of tests, it can be concluded that the radius estimation method makes the radius adaptive and less dependent on the preset initial value. However, the final phase of fluctuations indicates there is still room for enhancing the radius estimation accuracy.



(a) (b)

Figure 7. functions for estimating field radius (a) function 1; (b)function (2)

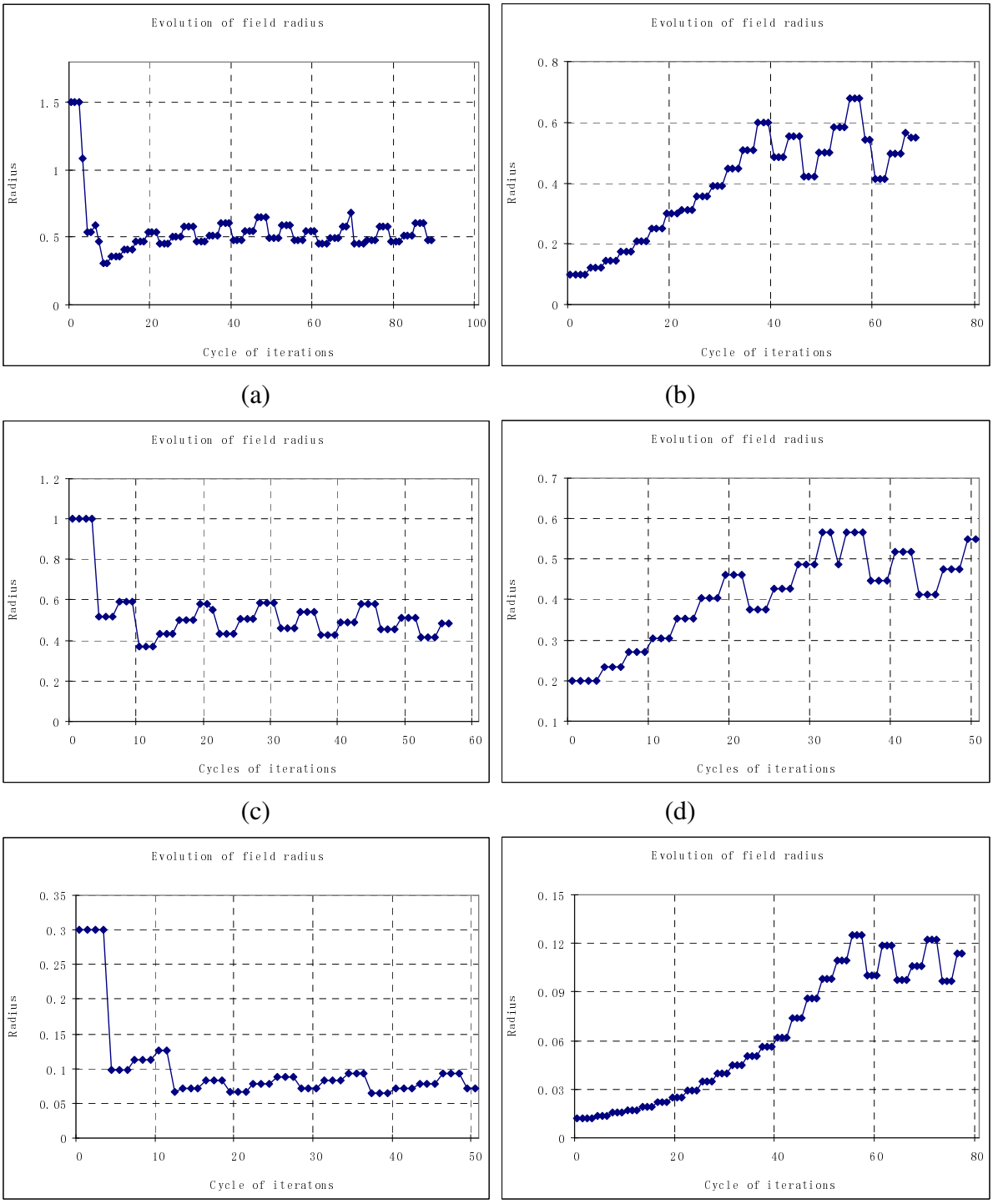


Figure 8. Evolution of the field radius during the optimisation process, (a)(c)(e) The algorithm starts with a large field radius for each function; (b)(d)(f) The algorithm starts with a small field radius for each function.

6. Evaluation of the Bees Algorithm for MMO

This section tests the proposed performance on optimising MMO problems. The purpose is to underline the Bees Algorithm's efficiency for MMO, rather than focusing on its superiority to other algorithms.

6.1 Experimental setup

To evaluate the performance of the proposed algorithm, some commonly used multimodal functions of various characteristics, such as irregular landscape, symmetric or equal distribution of optima, unevenly spaced optima, multiple global optima in the presence of multiple local optima, are employed as given in Table 6.

Table 6. Benchmark functions for evaluating the performance of the algorithm

f_1: Two-peak trap (1 global optima/1 local optima)

$$f(x)=\begin{cases}\frac{160}{15}(15-x), 0\le x<15\\ \frac{200}{5}(x-15), 15\le x<20\end{cases}, 0\le x\le 20 .$$

f_2: Central two-peak trap (1/1)

$$f(x)=\begin{cases}\frac{160}{10}x, 0\le x<10\\ \frac{160}{5}(x-15), 10\le x<15\\ \frac{200}{5}(x-15), 15\le x\le 20\end{cases}, 0\le x\le 20$$

f_3: Five-uneven-peak-trap (2/3)

$$f(x)=\begin{cases}80(2.5-x), 0\le x<2.5\\ 64(x-2.5), 2.5\le x<5\\ 64(7.5-x), 5\le x<7.5\\ 28(x-7.5), 7.5\le x<12.5\\ 28(17.5-x), 12.5\le x<17.5\\ 32(x-17.5), 17.5\le x<22.5\\ 32(27.5-x), 22.5\le x<27.5\\ 80(x-7.5), 27.5\le x<30\end{cases}, 0\le x\le 30$$

f_4: Equal maxima (5/0)

$$f(x) = \sin^6(5\pi x), 0 \le x \le 1$$

f_5: Decreasing maxima (1/4)

$$f(x) = \exp\left[-2\log(2)\cdot\left(\frac{x-0.1}{0.8}\right)^2\right]\cdot\sin^6(5\pi x), 0 \le x \le 1$$

f_6: Uneven maxima (5/0)

$$f(x) = \sin^6[5\pi(x^{3/4} - 0.05)], 0 \le x \le 1$$

f_7: Uneven decreasing maxima (1/4)

$$f(x) = \exp\left[-2\log(2)\cdot\left(\frac{x-0.08}{0.854}\right)^2\right]\cdot\sin^6[5\pi(x^{3/4} - 0.05)], 0 \le x \le 1$$

f_8: Himmelblau's function (4/0)

$$f(x) = 200 - (x_1^2 + x_2 - 11)^2 - (x_1 + x_2^2 - 7)^2, -6 \le x_i \le 6$$

f_9: Six-hump camel back (2/2)

$$f(\vec{x}) = -4[(4 - 2.1x_1^2 + \frac{1}{3}x_1^4)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2],$$

$$-1.9 \le x_1 \le 1.9, -1.1 \le x_2 \le 1.1$$

f_{10}: Shekel's foxholes (1/24)

$$f(\vec{x}) = 500 - \frac{1}{0.002 + \sum_{i=0}^{24}\frac{1}{1 + i + [x_1 - a(i)]^6 + [x_2 - b(i)]^6}}, \text{where } a(i) = 16(i \bmod 5 - 2),$$

$$b(i) = 16(\frac{i}{5} - 2); \quad -65.536 \le x_i \le 65.536$$

f_{11}: Inverted Shutert (18/many)

$$f(\vec{x}) = -\prod_{i=1}^{2}\sum_{j=1}^{5} j\cos[(j+1)x_i + j], -10 \le x_i \le 10$$

f_{12}: 1D inverted Vincent function (6/0)
f_{13}: 2D inverted Vincent function (36/0)
f_{14}: 3D inverted Vincent function (216/0)

$$f(\vec{x}) = \frac{1}{n}\sum_{i=1}^{n}\sin[10\cdot\log(x_i)], 0.25 \le x_i \le 10$$

The performance of Bees Algorithm for MMO is compared with that of r2psols, r3psols, r2psolhcls, r3psolhcls and IWO-δ-GSO. In the experiments, the parameter settings are matched as closely as possible with those used by the other algorithms under comparison.

However, due to the different mechanisms of the algorithms, some of the parameters are dimensionality depended, that is, nre=2·(D+1), nrb=D+1 and $stlim$=5·D. The initial ngh is correlated with the search range: ngh=2·range/ns. Table 7 lists the other parameter settings for the experiments.

Table 7. Initial colony size, initial field radius and search resolution

No.	*ns*	*nr*	*rad*	resolution
f_1-f_3	20	5	2	0.05
f_4-f_7, f_9	20	5	0.02	0.000001
f_8	20	5	0.02	0.005
f_{10}	20	20	3	0.00001
f_{11}	20	150	0.2	0.05
f_{12}	40	10	0.2	0.0001
f_{13}	150	50	0.2	0.001
f_{14}	300	400	0.2	0.001

6.2 Results and discussion

The success rate and average number of optima found are recorded and presented in Table 8 and Table 9 respectively, and again p-values are given in Table 10 to show the statistical significance of the performance differences between the compared algorithms and the proposed Bees Algorithm (with confidence level of 95%) in the number of optima. Some p-values cannot be obtained because the compared and the proposed algorithm can both reach 100% success rate in finding the optima. The best performance is reported in boldface. As can be seen from Table 8, the proposed Bees Algorithm and the IWO-δ-GSO generally perform better than others. The functions f_1, f_2, f_7 and f_{10} are the easiest, and all the algorithms can find the optima. The r3psolhcls is the only algorithm that cannot reach 100% success rate in finding the 5 optima of the function f_4, and the corresponding p-value of 0.0112 (below 0.05) indicates the proposed Bees Algorithm significantly differs from r3psolhcls. For function f_6, all the algorithms except r2psols can find the 5 optima with 100% success rate,

and the p-value of 0.0346 suggests the difference between the results is significant. For functions f_3, f_5 and $f_{12,}$ the proposed algorithm and IWO-δ-GSO show a distinctive advantage over the other algorithms, since they are capable of locating all the optima with 100% success rate while the other algorithms cannot. Furthermore, the p-values indicate that the differences in results are statistically significant, except for the r2psolhcls on function f_{12}. The success rate and the number of optima found show that the proposed Bees Algorithm outperforms the other algorithms on function f_8, and the corresponding p-values imply the results difference is significant except for the IWO-δ-GSO. However, the data from the three tables display the inferior performances of the IWO-δ-GSO, r3psols and r3psolhcls to the Bees Algorithm, r2psols and r2psolhcls on the function f_9. For $f_{11,}$ even though r2psolhcls generates the highest success rate, the corresponding p-values suggest all the algorithms except the r3psols produce statistically similar results. On functions f_{13} and f_{14} all algorithms fail to achieve a non-zero success rate. However, the average number of optima found, and the respective p-values, imply that the proposed Bees Algorithm and IWO-δ-GSO produce statistically resembling results, and that they outperform the other algorithms except for r3psols on f_{13}.

Table 8. Success rate

No.	BA (%)	r2pso ls(%)	r3pso ls(%)	r2pso lhcls(%)	r3pso lhcls(%)	IWO-δ-GSO (%)
f_1	**100**	**100**	**100**	**100**	**100**	**100**
f_2	**100**	**100**	**100**	**100**	**100**	**100**
f_3	**100**	52	36	80	48	**100**
f_4	**100**	**100**	**100**	**100**	96	**100**
f_5	**100**	16	4	64	16	**100**
f_6	**100**	96	**100**	**100**	**100**	**100**
f_7	**100**	**100**	**100**	**100**	**100**	**100**
f_8	**100**	76	64	68	72	88
f_9	**100**	**100**	96	**100**	96	60
f_{10}	**100**	**100**	**100**	**100**	**100**	**100**

f_{11}	70	64	52	**72**	60	-
f_{12}	**100**	92	96	96	92	**100**
f_{13}	0	0	0	0	0	0
f_{14}	0	0	0	0	0	0

Table 9. Average number of optima found

No.	BA	r2psols	r3psols	r2psolhcls	r3psolhcls	IWO-δ-GSO
f_1	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**
f_2	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**
f_3	**5.00**	3.96	3.24	4.72	3.92	**5.00**
f_4	**5.00**	**5.00**	**5.00**	**5.00**	4.96	**5.00**
f_5	**5.00**	1.98	1.24	4.52	2.80	**5.00**
f_6	**5.00**	4.96	**5.00**	**5.00**	**5.00**	**5.00**
f_7	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
f_8	**4.00**	3.76	3.60	3.68	3.72	3.88
f_9	**2.00**	**2.00**	1.96	**2.00**	1.96	1.60
f_{10}	**25.00**	**25.00**	**25.00**	**25.00**	**25.00**	**25.00**
f_{11}	17.70	17.56	17.36	**17.72**	17.60	-
f_{12}	**6.00**	5.92	5.96	5.96	5.88	**6.00**
f_{13}	**30.70**	26.64	26.92	27.28	27.44	29.60
f_{14}	**105.20**	77.20	76.28	78.40	77.99	102.60

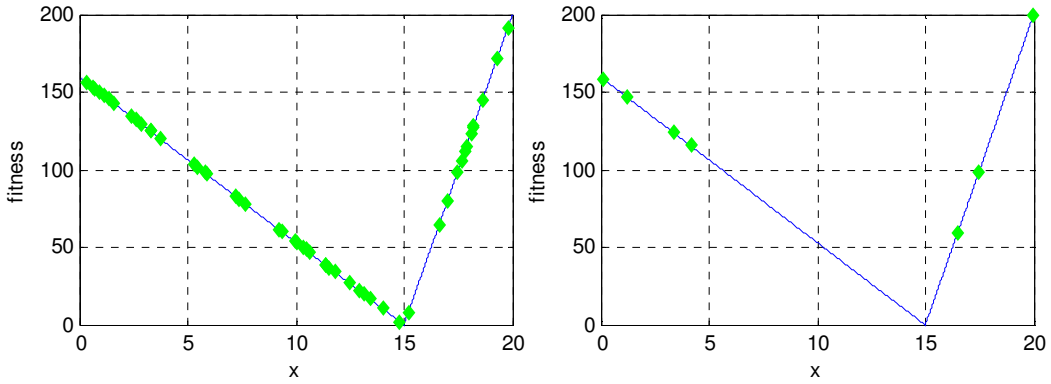
Table 10. *p*-values showing statistical significance

No.	r2psols	r3psols	r2psolhcls	r3psolhcls	IWO-δ-GSO
f_1	-	-	-	-	-
f_2	-	-	-	-	-
f_3	**< 0.0001**	**< 0.0001**	**0.0005**	**<0.0001**	-
f_4	-	-	-	**0.0112**	-
f_5	**< 0.0001**	**< 0.0001**	**0.0057**	**< 0.0001**	-
f_6	**0.0346**	-	-	-	-
f_7	-	-	-	-	-
f_8	**0.0326**	**0.0025**	**0.0135**	**0.0130**	0.1331
f_9	-	**0.0453**	-	**0.0210**	**<0.0001**
f_{10}	-	-	-	-	-
f_{11}	0.0972	**0.0005**	0.7715	0.1504	
f_{12}	**0.0002**	**0.0449**	0.0503	**0.0068**	-
f_{13}	**0.0017**	0.5718	**0.0115**	**0.0023**	0.3415
f_{14}	**0.0118**	**0.0235**	**0.0094**	**0.0340**	0.8240

6.3 Associated diagrams showing bees' distributions in search space

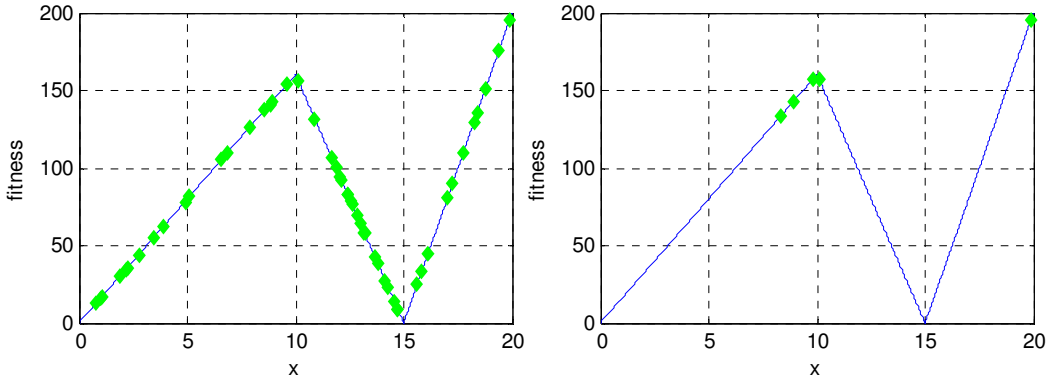
Function f1 has two peaks, of which one is global and the other is local. Figure 9 shows a simulation run of the proposed Bees Algorithm for MMO on f_1 starting from 50 bees. The

initial population is much larger than the necessary for finding only two optimal solutions. Most of the members in the colony become redundant and are removed. After 123 function evaluations, the bees get attracted towards the global optimum at x=20, and the local optimum at x=0. Finally both peaks are located by the bees.



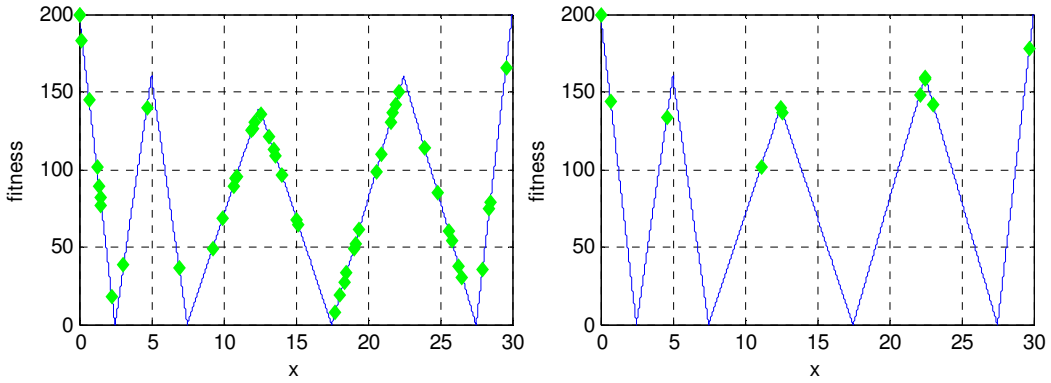
(a) initialisation (b) at 123 evaluations

Figure 9. Distribution of individuals in the search space during the evolution process for f_1



(a) initialisation (b) at 220 evaluations

Figure 10. Distribution of individuals in the search space during the evolution process for f_2

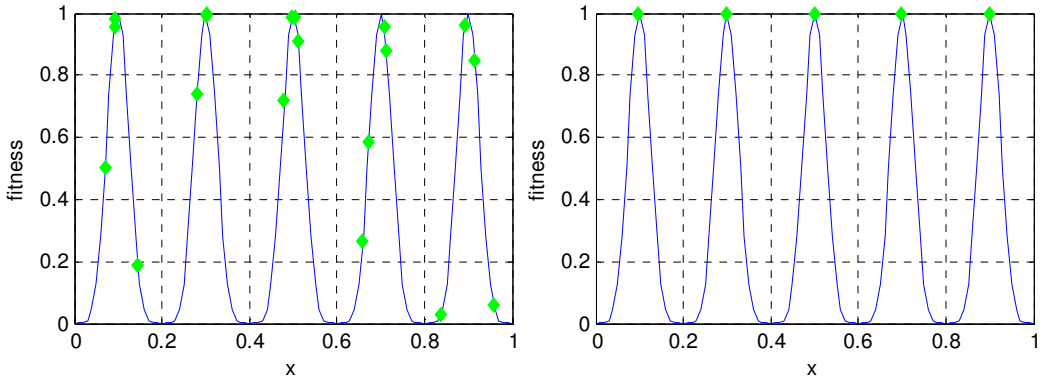


(a) initialisation (b) at 136 evaluations

Figure 11. Distribution of individuals in the search space during the evolution process for f_3

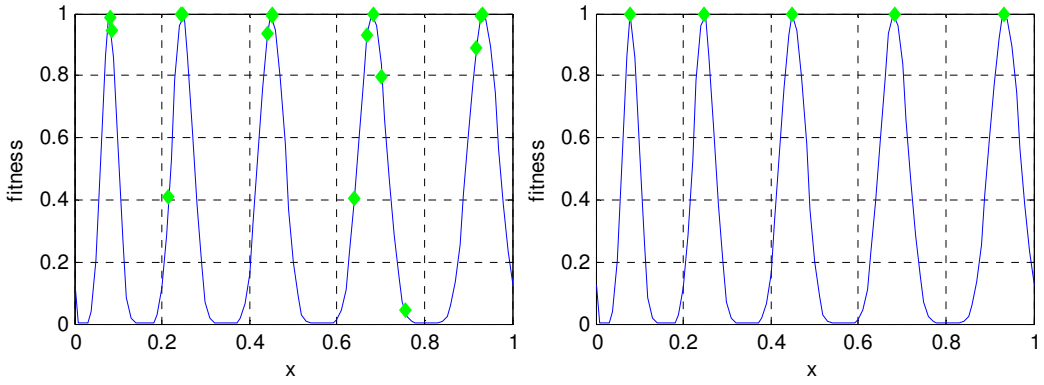
Also function f_2 has two peaks: one is the global peak and the other is the local one. Figure 10 shows two snapshots from a sample simulation run of the Bees Algorithm with an initial colony size of 50 bees, the first snapshot taken at initialisation and the other at 220 function evaluations.

f_3 has five peaks, of which two are global and three are local. Figure 11 shows two snapshots of a sample run of the Bees Algorithm on f3 at initialisation and after 136 function evaluations.



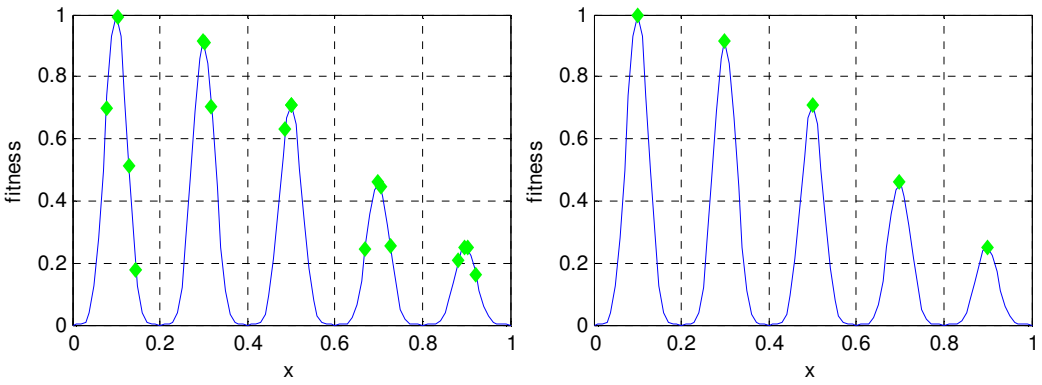
(a) at 205 evaluations (b) at 571 evaluations

Figure 12. Distribution of individuals in the search space during the evolution process for f_4



(a) at 208 evaluations (b) at 575 evaluations

Figure 13. Distribution of individuals in the search space during the evolution process for f_6



(a) at 208 evaluations (b) at 558 evaluations

Figure 14. Distribution of individuals in the search space during the evolution process for f_5

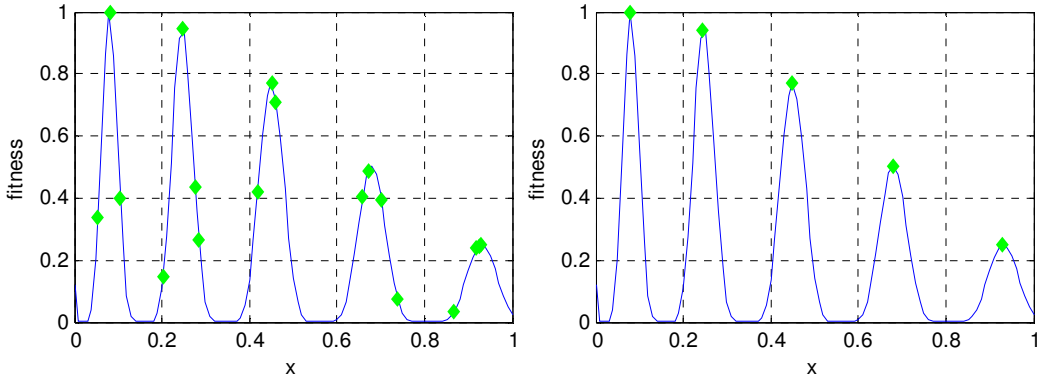
Functions f_4 and f_6 have five global peaks. The peaks of f_4 are evenly spaced, whereas the peaks of f_6 are unevenly spaced. Figure 12 and Figure 13 show two stages in a sample run on f_4 and f_6 respectively. It shows that the Bees Algorithm is able to find all the global and local peaks of f_4 and f_6 using very few function evaluations.

Functions f_5 and f_7 have one global peak and four local peaks. The peaks in f_5 are evenly spaced, whereas in f_7 they are unevenly spaced. Figure 14 and Figure 15 show a sample run of the Bees Algorithm on f5 and f7 respectively. The above figures show the colony distributions of the Bees Algorithm and illustrate the algorithm's ability to detect all the peaks.

Function f_8 has four global peaks. Figure 16 shows four stages of the search process of a simulation run of the Bees Algorithm on f8, using initially 50 bees, and sampled at: initialisation, after 495 evaluations, 1361 evaluations and 1736 evaluations.

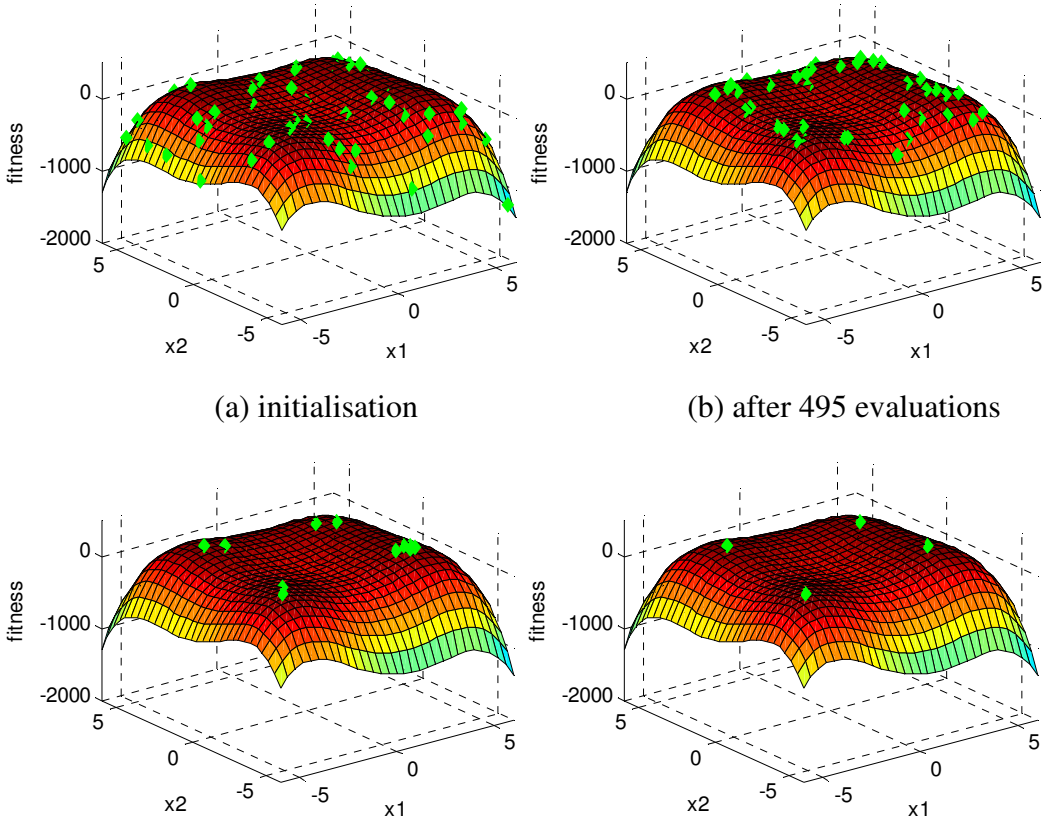
Function f_{10} has 25 evenly spaced peaks of unequal heights, of which one is the global peak whilst the others are local peaks. Figure 17 plots the evolution of the search process on function f_{10} at the initialisation stage, after 2191 function evaluations, 5735 function evaluations and 8253 function evaluations. Also on this function the algorithm locates all the

peaks successfully.



(a) at 209 evaluations (b) at 630 evaluations

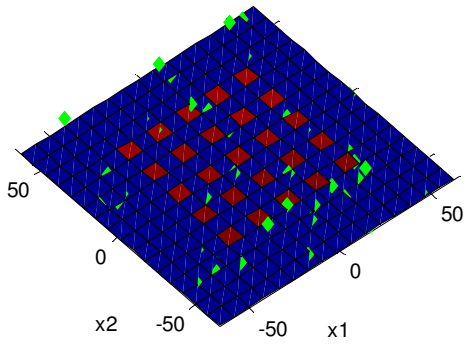
Figure 15. Distribution of individuals in the search space during the evolution process for f_7



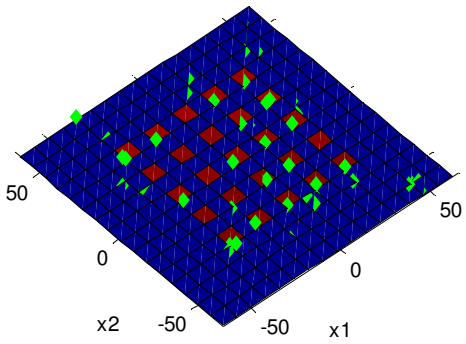
(a) initialisation (b) after 495 evaluations

(c) after 1361 evaluations (d) after 1736 evaluations

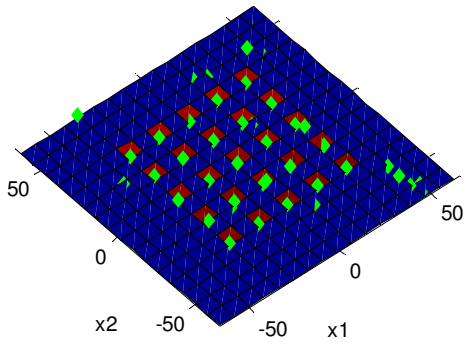
Figure 16. Distribution of individuals in the search space during the evolution process for f_8



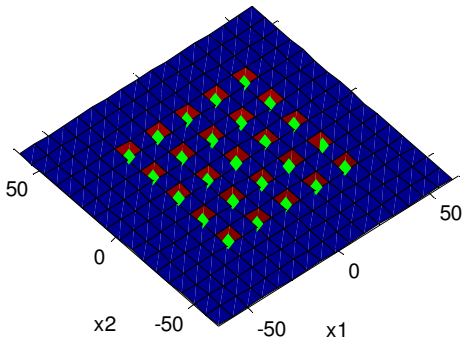
(a) initialisation



(b) after 2191 evaluations



(a) after 5735 evaluations



(b) after 8253 evaluations

Figure 17. Distribution of individuals in the search space during the evolution process for f_{10}

For visualisation reasons only a part of the benchmark functions showing how the bees distribute in the optimising process are illustrated. The results clearly indicate the high efficiency of the Bees Algorithm in solving MMO problems. The Bees Algorithm exhibits good explorative and exploitative abilities of locating global and local peaks. At the same time, the Bees Algorithm needs very few function evaluations to locate the optima. The speed of the Bees Algorithm is likely due to its adaptive colony size, in which surplus bees are removed.

7. Conclusion

This paper has proposed an evolutionary optimisation technique based on the basic Bees Algorithm for solving MMO problems. Some fundamental concepts of the basic Bees Algorithm are retained, such as the mechanism of combining exploration and exploitation,

and the “waggle dance” which allocates foragers according to the fitness of the discovered flower patches. The basic Bees Algorithm has been improved by adding several new procedures, which are also partly inspired by the honeybees’ natural behaviour. The experiments proved the ability of the proposed algorithm to solve MMO problems. This stems from the capability of the Bees Algorithm to adjust the number of individuals according to the complexity of the search space. This ability is also enhanced by the proposed local search method called balanced search technique (BST), which guides the foragers towards the fitness gradient whilst still retaining some randomness for explorative purposes. The HV is executed in the global search stage to detect good solutions in yet uncharted regions of the search space. The experimental results show that the proposed Bees Algorithm is competitive for the MMO problems compared with other algorithms.

A first step to extend the current work may be to stabilise the field radius. Currently, its fluctuation around an estimated value may degrade the accuracy of locating optimal solutions in the search space. Secondly, the BST can be further studied or modified for other optimisation problems like find one global optimum or tracking a dynamic optimum. Finally, a very important future research issue would be to develop a tool for estimating the ratio of the number of scouts in fields to the number of random scouts. The partition in scouts directly determines how the algorithm balances between exploitative and explorative search. The former decides the accuracy and speed of convergence when searching around an identified solution and the latter affects the ability of exploring potential spaces.

Acknowledgement

This research is supported by National Natural Science Foundation of China (Grant Nos. 51305319 and 51175389), and the Key Project of Natural Science Foundation of Hubei Province of China (Grant No. 2013CFA044).

References

[1] K. Deb. Multimodal Optimization Using a Bi-Objective Evolutionary Algorithm. Evolutionary Computation. Vol.20(1), pp.27-62, 2012.

[2] K. Deb, A. Srinivasan. Innovation: Innovative Design Principles Through Optimization. In Proceedings of the Genetic and Evolutionary Compuation Conference (GECCO 2006), pp.1629-1636, 2006.

[3] D.T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, M. Zaidi. The Bees Algorithm – A Novel Approach to Function Optimisation. Technical Note: MEC 0501, The Manufacturing Engineering Centre, Cardiff University, Queen's University UK, 2005.

[4] J.P. Li, M.E. Balazs, G.T. Parks, P.J. Clarkson. A species conserving genetic algorithm for multimodal function optimization, Evolutionary Computation, Vol.10(3), pp.207-234, 2002.

[5] K.A. De Jong. An analysis of the behavior of a class of genetic adaptive systems. Ph.D. thesis. University of Michigan Ann Arbor, MI, USA, 1975.

[6] D.E. Goldberg, J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In proceedings of the Second International Conference on Genetic Algorithms and their application. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, pp.41-49, 1987.

[7] K.S. Leung, Y. Liang. Adaptive Elitist-Population Based Genetic Algorithm for Multimodal Function Optimization, Genetic and Evolutionary Computation – GECCO 2003, Lecture Notes in Computer Science Volume 2723, Springer-Verlag, pp.1160-1171, 2003.

[8] Y. Liang, K.S. Leung. Genetic Algorithm with adaptive elitist-population strategies for multimodal function optimization. Applied Soft Computing. Vol.11(2), pp.2017-2034, 2011.

[9] J.E. Vitela, O. Castanos. A sequential niching memetic algorithm for continuous multimodal function optimization. Applied Mathematics and Computation, Vol.218(17), pp.8242-8259, 2012.

[10] D.X. Chang, X.D. Zhang, C.W. Zheng, D.M. Zhang. A robust dynamic niching genetic algorithm with niche migration for automatic clustering problem. Pattern Recognition, Vol.43(4), pp.1346-1360, 2010.

[11] A.E. Imrani, A. Bouroumi, H.Z. Abidine, M. Limouri, A. Essaid. A fuzzy clustering-based niching approach to multimodal function optimization. Cognitive Systems Research, Vol.1(2), pp.119-133, 2000.

[12] J. Gan, K. Warwick. Dynamic Niche Clustering: a fuzzy variable radius niching technique for multimodal optimization in GAs. In Proceedings of the 2001 Congress on Evolutionary Computation, Vol.1, pp.215-222, 2001.

[13] D.T. Vollmer, T. Soule, M. Manic. A Distance Measure Comparison to Improve Crowding in Multi-Modal Optimization Problems. 2010 3rd International Symposium on Resilient Control Systems (ISRCS), Idaho Falls, pp.31-36, 2010.

[14] E.J. Vitela, O. Castanos. A Real-Coded Niching Memetic Algorithm for Continuous Multimodal Function Optimization. IEEE Congress on Evolutionary Computation (CEC 2008), Hong Kong, pp2170-2177, 2008.

[15] H.F. Wang, I. Moon, S.X. Yang, D.W. Wang. A memetic particle swarm optimization algorithm for multimodal optimization problems. Information Science, Vol.197, pp.38-52, 2012.

[16] R. Brits, A.P. Engelbrecht, F.V. Bergh. Locating multiple optima using particle swarm optimization. Applied Mathematics and Computation Vol.189(2), pp.1859-1883, 2007.

[17] J. Zhang, D.S. Huang, T.M. Lok, M.R. Lyu. A novel adaptive sequential niche technique for multimodal function optimization. Neurocomputing, Vol.69(16-18), pp.2396-2401, 2006.

[18] X.D. Li. Niching Without Niching Parameters: Particle Swarm Optimization Using a Ring Topology. IEEE Transactions on Evolutionary Computation, Vol.14(1), 2010.

[19] B.Y. Qu, J.J. Liang, P.N. Suganthan. Niching particle swarm optimization with local search for multi-modal optimization. Information science 197, pp131-143, 2012.

[20] A. Passaro. Niching in Particle Swarm Optimization. Ph.D. thesis, University of Pisa, Department of Computer Science, 2007.

[21] J, Zhang. J.R. Zhang, K. Li. A Sequential Niching Technique for Particle Swarm Optimization. International Conference on Intelligent Computing, ICIC 2005, Hefei China, Part I, Vol.3644, pp.390-399, 2005.

[22] Y.F. Xu. A niching particle swarm segmentation of infrared images. 2010 Sixth

International Conference on Natural Computing (ICNC), Yantai, China, Vol.7, pp.3739-3742, 2010.

[23] Y. Liu, X.X. Ling, Z.W. Shi, M.W. Lv, J. Fang, L. Zhang. A survey on particle swarm optimization algorithms for multimodal function optimization. Journal of Software, Vol.6(12), pp.2449-2455, 2011.

[24] X.D. Li. Efficient differential evolution using speciation for multimodal function optimization. In Proceeding of the 2005 conference on Genetic and evolutionary computation (GECCO'05), New York, USA, pp.873-880, 2005.

[25] K.C. Wong, C.H. Wu, R.K.P. Mok, C.B. Peng, Z.L. Zhang. Evolutionary multimodal optimization using the principle of locality. Information Science Vol.194, pp.138-170, 2012.

[26] X. Zhao, Y. Yao, L.P. Yan. Learning algorithm for multimodal optimization. Computers & Mathematics with Applications, Vol.57(11-12), pp.2016-2021, 2009.

[27] M. Schoenauer, F. Teytaud, O. Teytaud. Simple tools for multimodal optimization. In Proceedings of the 13th annual conference companion on Genetic and evolutionary computation (GECCO'11), Dublin, Ireland, 2011.

[28] S. Roy, S.M. Islam, S. Das, S. Ghosh. Multimodal optimization by artificial weed colonies enhanced with localized group search optimizers. Applied Soft Computing Vol.13(1), pp.27-46, 2013.

[29] D.T. Pham, A. Ghanbarzadeh, E.Koc, S. Otri, S. Rahim, M. Zaidi. The Bees Algorithm – A Novel Tool for Complex Optimisation Problems. In Proceedings of the 2nd International Virtual Conference on Intelligent Production Machines and Systems

(IPROMS 2006), pp.454-459, 2006.

[30] D.T. Pham, M. Castellani. The Bees Algorithm: modelling foraging behavior to solve continuous optimization problems. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, Vol.223(12), pp.2919-2938, 2009.

[31] D.T. Pham, Q.T. Pham, A. Ghanbarzadeh, M. Catstellani. Dynamic Optimisation of Chemical Engineering Processes Using the Bees Algorithm. Proceedings of the 17th IFAC World Congress, the International Federation of Automatic Control Seoul, Korea, Vol.17, Part 1, pp.6100-6105, 2008.

[32] M.S. Packianather, M. Landy and D.T, Pham. Enhancing the speed of the Bees Algorithm using pheromone-based recruitment. 7th IEEE International Conference on Industrial Informatics (INDIN 2009), Cardiff, Wales, pp.789-794, 2009.

[33] D.T. Pham, A.H. Darwish. Fuzzy Selection of Local Search Sites in the Bees Algorithm. In the 4th International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007) [online], Available from http://conference.iproms.org.

[34] Q.T. Pham, D.T. Pham, M. Castellani. A modified bees algorithm and a statistics-based method for tuning its parameters. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering 226:287, pp.287-301, 2011.

[35] D.T. Pham, H.A. Darwish. Using the Bees Algorithm with Kalman Filtering to Train an Artificial Neural Network for Pattern Classification. Journal of Systems and

Control Engineering, Vol.224(1), pp.885-892, 2010.

[36] D.T. Pham, E. Koc. Design of a Two-dimensional recursive filter using the bees algorithm. International Journal of automation and computing. Vol.7(3), pp399-403, 2010.

[37] D.T. Pham, E. Koc, J.Y. Lee, J. Phruekanant. Using the Bees Algorithm to schedule jobs for a machine. In Proceedings of Eighth International Conference on Laser Metrology, CMM and Machine Tool Performance, pp.430-439, 2007.

[38] M.C. Ang, D.T. Pham, A.J. Soroka, K.W. Ng. PCB assembly optimization using the bees algorithm enhanced with TRIZ operations. 36th Annual Conference on IEEE Industrial Electronics Society (IECON 2010), Glendale, AZ, pp.2708-2713, 2010.

[39] D.T. Pham, M. Castellani, A.A. Fahmy. Learning the inverse kinematics of a robot manipulator using the Bees Algorithm. 6th IEEE International Conference on Industrial Informatics (INDIN 2008), Daejeon, pp493-498, 2008.

[40] R.K. Ursem. Multinational evolutionary algorithms. Proceedings of the 1999 Congress on Evolutionary Computation (ECE 99), Washington DC, Vol.3, pp.1633-1640, 1999.