Applied Soft Computing 11 (2011) 2888–2901

Contents lists available at ScienceDirect

Applied Soft Computing

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

The best-so-far selection in Artificial Bee Colony algorithm

Anan Banharnsakun, Tiranee Achalakul, Booncharoen Sirinaovakul*

Department of Computer Engineering, King Mongkut's University of Technology Thonburi, Bangkok, Thailand

ARTICLE INFO

Article history:
Received 15 February 2010
Received in revised form 31 May 2010
Accepted 28 November 2010
Available online 4 December 2010

Keywords:
Artificial Bee Colony
Swarm intelligence
Optimization
Image registration
Mutual information

ABSTRACT

The Artificial Bee Colony (ABC) algorithm is inspired by the behavior of honey bees. The algorithm is one of the Swarm Intelligence algorithms explored in recent literature. ABC is an optimization technique, which is used in finding the best solution from all feasible solutions. However, ABC can sometimes be slow to converge. In order to improve the algorithm performance, we present a modified method for solution update of the onlooker bees in this paper. In our method, the best feasible solutions found so far are shared globally among the entire population. Thus, the new candidate solutions are more likely to be close to the current best solution. In other words, we bias the solution direction toward the best-so-far position. Moreover, in each iteration, we adjust the radius of the search for new candidates using a larger radius earlier in the search process and then reduce the radius as the process comes closer to converging. Finally, we use a more robust calculation to determine and compare the quality of alternative solutions. We empirically assess the performance of our proposed method on two sets of problems: numerical benchmark functions and image registration applications. The results demonstrate that the proposed method is able to produce higher quality solutions with faster convergence than either the original ABC or the current state-of-the-art ABC-based algorithm.

1. Introduction

Swarm Intelligence is a meta-heuristic method in the field of artificial intelligence that is used to solve optimization problems. It is based on the collective behavior of social insects, flocks of birds, or schools of fish. These animals can solve complex tasks without centralized control.

Researchers have analyzed such behaviors and designed algorithms that can be used to solve combinatorial and numerical optimization problems in many science and engineering domains. Previous research [1–4] has shown that algorithms based on Swarm Intelligence have great potential. The algorithms that have emerged in recent years include Ant Colony Optimization (ACO) [5] based on the foraging behavior of ants, and Particle Swarm Optimization (PSO) [6] based on the behaviors of bird flocks and fish schools.

Exploration and exploitation are the important mechanisms in a robust search process. While exploration process is related on the independent search for an optimal solution, exploitation uses existing knowledge to bias the search. In the recent years, there are a few algorithms based on bee foraging behavior developed to improve both exploration and exploitation for solving the numerical optimization problems.

The Artificial Bee Colony (ABC) algorithm introduced by D. Karaboga [7] is one approach that has been used to find an optimal solution in numerical optimization problems. This algorithm is inspired by the behavior of honey bees when seeking a quality food source. The performance of ABC algorithm has been compared with other optimization methods such as Genetic Algorithm (GA), Differential Evolution algorithm (DE), Evolution Strategies (ES), Particle Swarm Optimization, and Particle Swarm Inspired Evolutionary Algorithm (PS-EA) [8–10]. The comparisons were made based on various numerical benchmark functions, which consist of unimodal and multimodal distributions. The comparison results showed that ABC can produce a more optimal solution and thus is more effective than the other methods in several optimization problems [11–13].

Yang [14] introduced an algorithm called the Virtual Bee Algorithm (VBA) for solving engineering optimizations that have multi-peaked functions. In the VBA algorithm, the objectives or optimization functions are encoded as virtual foods. Virtual bees are used to search for virtual foods in the search space. The position of each virtual bee is updated via the virtual pheromone from the neighboring bees. The food with largest number of virtual bees or intensity of visiting bees corresponds to the optimal solution. However, the VBA algorithm was only tested using two-dimension functions.

An optimization algorithm inspired by the honey bee foraging behavior based on the elite bee method was proposed by Sundareswaran [15]. The bee whose solution is the best possible solution in each simulation iteration is considered to be the elite

* Corresponding author.
E-mail addresses: anan_cpe@yahoo.com (A. Banharnsakun), tiranee@cpe.kmutt.ac.th (T. Achalakul), boon@kmutt.ac.th (B. Sirinaovakul).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.11.025

```
Line  Initialization:
1     For i = 1 to n(FS)                          //do for all food sources
2       For d = 1 to D
3         x(FS, id) = x(min, d) + rand[0,1] * (x(max, d) − x(min, d))//initialize the feasible solutions in the search space
4       Next d
5     Next i
6     While (iteration ≤ MaxIteration)
7     For i = 1 to n(EB)                          //do for each employed bees
8       For d = 1 to D
9         x(EB, id) = x(FS, id)                   //assign the food source position to the employed bee
10      Next d
11      Select d_s                                //Randomly select the dimension of the solution
12      Select x_n(FS, d_s)                       //Randomly select the neighboring solution
13      x(EB, id_s) = x(FS, id_s) + rand[−1,1] * (x(FS, id_s) − x_n(FS, d_s))   //update the position of the employed bee
14      If f(x(EB, i)) < f(x(FS, i))              //compare and select the better solution between the old and the new solution
15        x(FS, id_s) = x(EB, id_s)               //Replace the old solution with the new solution
16        limit_count(x(FS, i)) = 0
17      Else
18        limit_count(x(FS, i)) + +
19    Next i
20    For i = 1 to n(EB)                          //do for all employed bees for selecting the best-so-far solution
21      If i = 1
22        For d = 1 to D
23          x_b(FS, d) = x(FS, id)
24        Next d
25        f(x_b(FS)) = f(x(FS, i))
26      Else
27        If f(x(FS, i)) < f(x_b(FS))
28          For d = 1 to D
29            x_b(FS, d) = x(FS, id)
30          Next d
31          f(x_b(FS)) = f(x(FS, i))
32    Next i
33    For i = 1 to n(OB)                          //do for each onlooker bee
34      Select d_s                                //Randomly select the dimension of the solution
35      Select x_s(FS, d_s)                       //Select the food source based on equation 2.3
36      For d = 1 to D
37        x(OB, id) = x_s(FS, d_s) + rand[−1,1] * fitness(x_b(FS)) * (x_s(FS, d_s) − x_b(FS, d_s))   //update the position of the onlooker bee
38      Next d
39      If f(x(OB, i)) < f(x_s(FS))               //compare and select the better solution between the old and the new solution
40        For d = 1 to D
41          x_s(FS, d) = x(OB, id)                //Replace the old solution with the new solution
42        Next d
43        limit_count(x_s(FS)) = 0
44      Else
45        limit_count(x_s(FS)) + +
46    Next i
47    For i = 1 to n(FB)                          //do for all food sources for abandoning the food source that cannot improve the further result
48      If limit_count(x(FS, i)) > limit
49        For d = 1 to D
50          x(SB, d) = x(FS, i) + rand[−1,1] * (ω_max − iteration/MaxIteration (ω_max − ω_min)) * x(FS, i)   //update the position of the scout bee
51        Next d
52        If f(x(SB)) < f(x(FS, i))               //compare and select the better solution between the old and the new solution
53          For d = 1 to D
54            x(FS, id) = x(SB, d)                //Replace the old solution with the new solution
55          Next d
56          limit_count(x(FS)) = 0
57        Else
58          limit_count(x(FS)) + +
59    Next i
```

Fig. 1. The pseudo-code of the Best-so-far ABC algorithm.

bee. A probabilistic approach is used to control the movement of the other bees, so majority of bees will follow the elite bee's direction while a few bees may fly to other directions. This approach improves the capability of convergence to a global optimum.

To improve the exploration and exploitation of foraging behavior of honey bees for numerical function optimization, Akbari et al. [16] presented an algorithm called Bee Swarm Optimization (BSO). In this method, the bees of the swarm are sorted according to the fitness values of the most recently visited food source and these sorted bees are divided into three types. The bees that have worst fitness are classified as scout bees, while the rest of bees are divided equally as experienced foragers and onlookers. Different flying patterns were introduced for each type of bee to balance the exploration and exploitation in this algorithm.

The old method for comparing the solutions	The objective-value-based comparison method

If $(Fitness(f_{new}(x)) > Fitness(f_{old}(x)))$
Replace the old solution with the new solution
Else
Keep the old solution
	If (Finding Minimum)
	If $(f_{new}(x) < f_{old}(x))$
	Replace the old solution with the new solution
	Else
	Keep the old solution
	Else If (Finding Maximum)
	If $(f_{new}(x) > f_{old}(x))$
	Replace the old solution with the new solution
	Else
	Keep the old solution

Fig. 2. The comparison between the old and the new method for comparing the solutions.

The experimental results from these algorithms show that the algorithms based on bee foraging behavior can successfully solve numerical optimization problems. However, in some cases the convergence speed can be an issue. This paper introduces a modified version of ABC in order to improve the algorithm's performance.

First, we test our modified algorithm using the same benchmarks as the previously cited research including the original ABC and the BSO algorithm. Then, we apply our method to optimize the mutual information value in order to measure similarity in an image registration process. We compare the registration results between the original ABC and our best-so-far method.

The paper is organized as follows. Section 2 describes the original ABC algorithm. Section 3 presents our best-so-far method. Section 4 describes the automated image registration problem. Section 5 presents the experiments. Section 6 compares and discusses the performance results of the best-so-far method with the original method and BSO algorithm. Finally, Section 7 offers our conclusions.

2. The original ABC algorithm

The ABC algorithm assumes the existence of a set of computational agents called honey bees. The honey bees in this algorithm are categorized into three groups: employed bees, onlooker bees and scout bees. The colony is equally separated into employed bees and onlooker bees. Each solution in the search space consists of a set of optimization parameters which represent a food source position. The number of employed bees is equal to the number of food sources. In other words, there is only one employed bee on each food source. The quality of food source is called its "fitness value" and is associated with its position. The process of bees seeking for good food sources is the process used to find the optimal solution.

The employed bees will be responsible for investigating their food sources and sharing the information about these food sources to recruit the onlooker bees. The onlooker bees will make a decision to choose a food source based on this information. The food source that has higher quality will have a larger chance to be selected by onlooker bees than one of lower quality. An employed bee whose food source is rejected as low quality by employed and onlooker bees will change to a scout bee to search randomly for new food sources. By this mechanism, the exploitation will be handled by employed and onlooker bees while the exploration will be maintained by scout bee. The details of the algorithm are as follows.

First, randomly distributed initial food source positions are generated. The process can be represented by Eq. (2.1). After initialization, the population is subjected to repeated cycles of three major steps: updating feasible solutions, selecting feasible solutions, and avoiding suboptimal solutions.

$$F(x_i), x_i \in R^D, \quad i \in \{1, 2, 3, \ldots, SN\}, \tag{2.1}$$

x_i is a position of food source as a D-dimensional vector, $F(x_i)$ is the objective function which determines how good a solution is, and SN is the number of food sources.

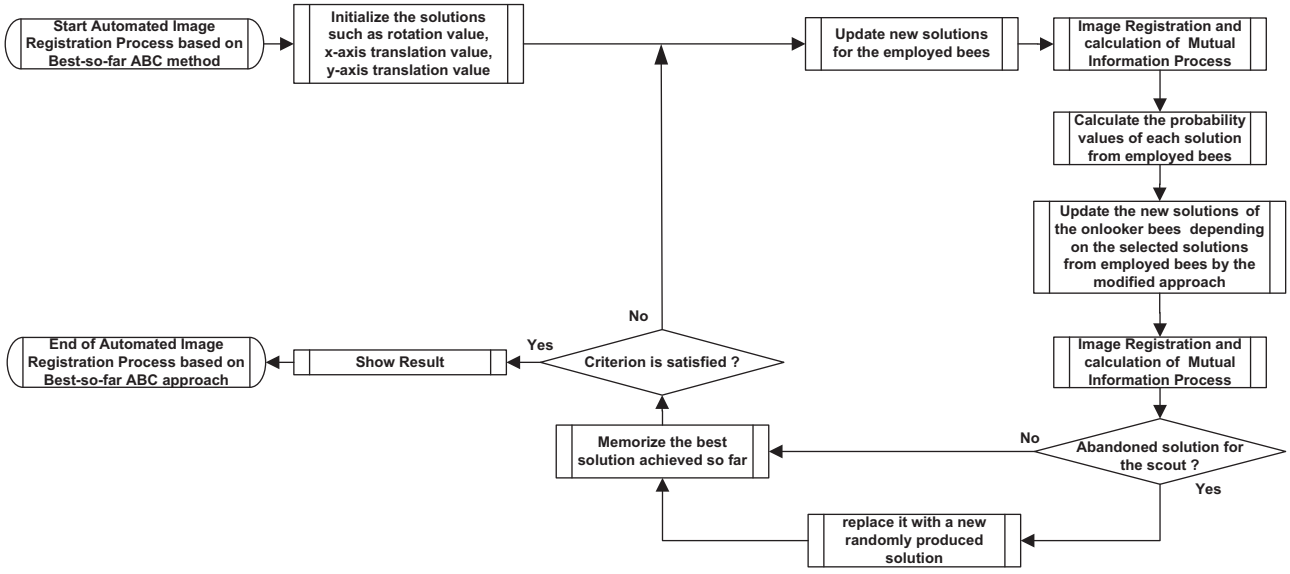

Fig. 3. Automated image registration based on the best-so-far ABC method.

Table 1
Numerical benchmark functions.

Function name	Function	Characteristic	Ranges	Global Minimum
Sphere	$f_1(\vec{x}) = \sum_{i=1}^{D} x_i^2$	Uni-modal, Separable	$-100 \le x_i \le 100$	0
Griewank	$f_2(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{D} x_i^2\right) - \left(\prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{1}}\right)\right) + 1$	Multi-modal, Non-Separable	$-600 \le x_i \le 600$	0
Rastrigin	$f_3(\vec{x}) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	Multi-modal, Separable	$-5.12 \le x_i \le 5.12$	0
Rosenbrock	$f_4(\vec{x}) = \sum_{i=1}^{D} 100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2$	Uni-modal, Non-Separable	$-30 \le x_i \le 30$	0
Ackley	$f_5(\vec{x}) = 20 + e - 20e^{\left(-0.2\sqrt{(1/D)\sum_{i=1}^{D} x_i^2}\right)} - e^{(1/D)\sum_{i=1}^{D}\cos(2\pi x_i)}$	Multi-modal, Non-Separable	$-30 \le x_i \le 30$	0
Schaffer	$f_6(\vec{x}) = 0.5 + \frac{(\sin\sqrt{x_1^2+x_2^2})^2 - 0.5}{(1+0.001(x_1^2+x_2^2))^2}$	Multi-modal, Non-Separable	$-100 \le x_i \le 100$	0

In order to update feasible solutions, all employed bees select a new candidate food source position. The choice is based on the neighborhood of the previously selected food source. The position of the new food source is calculated from equation below.

$$\upsilon_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{2.2}$$

In the Eq. (2.2), υ_{ij} is a new feasible solution that is modified from its previous solution value (x_{ij}) based on a comparison with the randomly selected position from its neighboring solution (x_{kj}). ϕ_{ij} is a random number between [−1,1] which is used to randomly adjust the old solution to become a new solution in the next iteration. $k \in \{1,2,3,\ldots, SN\}$ and $j \in \{1,2,3,\ldots, D\}$ are randomly chosen indexes. The difference between x_{ij} and x_{kj} is a difference of position in a particular dimension. The algorithm changes each position in only one dimension in each iteration. Using this approach, the diversity of solutions in the search space will increase in each iteration.

The old food source position in the employed bee's memory will be replaced by the new candidate food source position if the new position has a better fitness value. Employed bees will return to their hive and share the fitness value of their new food sources with the onlooker bees.

In the next step, each onlooker bee selects one of the proposed food sources depending on the fitness value obtained from the employed bees. The probability that a food source will be selected can be obtained from an equation below.

$$P_i = \frac{\mathrm{fit}_i}{\sum_{n=1}^{SN} \mathrm{fit}_n} \tag{2.3}$$

where fit_i is the fitness value of the food source i, which is related to the objective function value ($F(x_i)$) of the food source i.

The probability of a food source being selected by the onlooker bees increases as the fitness value of a food source increases. After the food source is selected, onlooker bees will go to the selected food source and select a new candidate food source position in the neighborhood of the selected food source. The new candidate food source can be expressed and calculated by Eq. (2.2).

In the third step, any food source position that does not improve the fitness value will be abandoned and replaced by a new position that is randomly determined by a scout bee. This helps avoid suboptimal solutions. The new random position chosen by the scout bee will be calculated from equation below.

$$x_{ij} = x_j^{\min} + rand[0,1] * (x_j^{\max} - x_j^{\min}) \tag{2.4}$$

where $x_j^{\min}$ is the lower bound of the food source position in dimension j and $x_j^{\max}$ is the upper bound of the food source position in dimension j.

The maximum number of cycles (MCN) is used to control the number of iterations and is a termination criterion. The process will be repeated until the output of the objective function reaches a defined threshold value or the number of iteration equals the MCN. A defined threshold value is set to be equal to the global minimum value or the global maximum value depending on the type of the optimization problems.

Although the activities of exploitation and exploration are well balanced and help to mitigate both stagnation and premature convergence in the ABC algorithm, the convergence speed still is the major issue in some situations in the ABC algorithm.

In our best-so-far ABC, the exploitation process is focused on the best-so-far food source. This food source is used in the comparison process for updating the new candidate food source to accelerate the convergence speed. The searching ability of the exploration process is also enhanced by the scout bee. It will randomly generate a new food source to avoid local optima.

3. The best-so-far ABC algorithm

To enhance the exploitation and exploration processes, we propose to make three major changes by introducing the best-so-far method, an adjustable search radius, and an objective-value-based comparison method.

The computational complexity is also addressed in order to evaluate the computational complexity of the best-so-far ABC with the original ABC and the BSO algorithms.

The pseudo-code of the best-so-far ABC algorithm is illustrated in Fig. 1. In this pseudo-code, the best-so-far method is shown in line number 37 where the onlooker bees will register the best selected food source position so far within the new candidate generation function. The adjustable search radius is performed by a scout bee in line number 50 and the objective-value-based comparison method is used in line number 14, 27, 39 and 52. The full details of each change are described in section 3.1–3.3.

Note that the pseudo-code in Fig. 1 is used to find the minimum objective value. To find the maximum objective value, the condition which is used to compare the old solution and the new solution, in line number 14, 27, 39 and 52, must be changed from

$$f(x_{new}(*)) < f(x_{old}(*))$$

Table 2
Result obtained in the ABC1 experiment.

Function	Dimensio	Max iteration	ABC				Best-so-far ABC			
			Convergence Iteration	Runtime (s)	Mean of objective values	SD	Convergence Iteration	Runtime (s)	Mean of objective values	SD
Sphere	10	500	500	0.234	1.426E−16	8.212E−17	500	0.281	3.303E−188	3.299E−187
	20	750	750	0.593	2.353E−12	2.198E−12	750	0.704	1.118E−239	1.109E−238
	30	1000	1000	1.109	2.649E−10	2.136E−10	1000	1.546	5.6413E−302	3.967E−301
Griewank	10	500	500	0.453	1.040E−03	2.741E−03	500	0.500	2.157E−214	2.092E−213
	20	750	750	1.203	4.734E−09	2.177E−08	750	1.344	2.920E−302	2.801E−300
	30	1000	1000	2.344	5.369E−09	2.664E−08	845	2.672	0	0
Rastrigin	10	500	500	0.344	4.911E−16	1.073E−15	500	0.391	1.414E−210	1.410E−209
	20	750	750	0.922	2.933E−11	5.728E−11	750	0.985	3.567E−300	3.569E−299
	30	1000	1000	1.782	3.593E−07	3.565E−06	863	1.953	0	0
Rosenbrock	10	500	500	0.469	0.07022	0.09035	462	0.465	0	0
	20	750	750	1.375	0.427111	0.611375	750	1.484	7.198E−22	4.941E−21
	30	1000	1000	2.704	0.681802	0.774655	1000	2.969	3.604E−15	3.585E−14
Ackley	10	500	500	0.390	1.400E−10	7.657E−11	63	0.047	0	0
	20	750	750	0.968	3.681E−07	1.302E−07	85	0.094	0	0
	30	1000	1000	1.782	5.246E−06	1.702E−06	109	0.234	0	0
Schaffer	2	2000	2000	0.672	7.895E−11	4.014E−10	947	0.468	0	0

to

$$f(x_{new}(*)) > f(x_{old}(*))$$

where $f(x(^*))$ is the objective function value.

3.1. The best-so-far method

In order to improve the efficiency of the onlooker bees, we proposed to modify the parameters that are used to calculate new candidate food sources. In the original algorithm, each onlooker bee selects a food source based on a probability that varies according to the fitness function explored by a single employed bee. Then the new candidate solutions are generated by updating the onlooker solutions as shown in Eq. (2.2). In our best-so-far method, from the pseudo-code line number 20 to 32, all onlooker bees use the information from all employed bees to make a decision on a new candidate food source. Thus, the onlookers can compare information from all candidate sources and are able to select the best-so-far position.

On the assumption that the best-so-far position will lead to the optimal solution, from the pseudo-code line number 33 to 46, we bias the solution direction by using the best-so-far solutions-based approach to update the new candidate solutions of onlooker bees. We then accelerate its convergence speed by using the fitness value of the best-so-far food source as the multiplier of the error correction for this solution update. The values in all dimensions of each food source are also updated in each iteration in order to increase the diversity of the feasible solutions in the search space.

The new method used to calculate a candidate food source is shown in Eq. (3.1)

$$v_{id} = x_{ij} + \Phi f_b(x_{ij} - x_{bj}) \tag{3.1}$$

where: v_{id} = The new candidate food source for onlooker bee position i dimension d, d = 1,2,3,...D; x_{ij} = The selected food source position i in a selected dimension j; Φ = A random number between −1 and 1; f_b = The fitness value of the best food source so far; x_{bj} = The best-so-far food source in selected dimension j.

This change should make the best-so-far algorithm converge more quickly because the solution will be biased towards best-so-far solution.

3.2. The adjustable search radius

Although our best-so-far method can increase the local search ability compared to the original ABC algorithm, the solution is easily entrapped in a local optimum. In order to resolve this issue, we introduce a global search ability for the scout bee.

From the pseudo-code line number 47 to 59, the scout bee will randomly generate a new food source by using Eq. (3.2) whenever the solution stagnates in the local optimum.

$$\upsilon_{ij} = x_{ij} + \phi_{ij}\left[\omega_{\max} - \frac{iteration}{MCN}(\omega_{\max} - \omega_{\min})\right] x_{ij} \tag{3.2}$$

where υ_{ij} is a new feasible solution of a scout bee that is modified from the current position of an abandoned food source (x_{ij}) and ϕ_{ij} is a random number between [−1,1]. The value of $\omega_{\max}$ and $\omega_{\min}$ represent the maximum and minimum percentage of the position adjustment for the scout bee. The value of $\omega_{\max}$ and $\omega_{\min}$ are fixed to 1 and 0.2, respectively. These parameters were chosen by the experimenter. With these selected values, the adjustment of scout bee's position based on its current position will linearly decrease from 100 percent to 20 percent in each experiment round.

Based on the assumption that the solution of scout bee will be far from the optimal solution in the first iteration and it will converge closely to the optimal solution in later iterations, Eq. (3.2) will

Table 3
Result obtained in the ABC2 experiment.

Function	Dimensio	Max iteration	ABC				BSO		Best-so-far ABC			
			Convergence Iteration	Runtime (s)	Mean of objective values	SD	Mean of objective values	SD	Convergence Iteration	Runtime (s)	Mean of objective values	SD
Sphere	10	5000	5000	2.390	9.472E−17	1.615E−17	8.475E−123	3.953E−122	833	0.578	0	0
	20	7500	7500	5.859	2.627E−16	8.796E−17	2.361E−115	7.348E−115	1003	1.234	0	0
	30	10000	10000	10.922	3.217E−16	5.142E−17	4.728E−102	1.372E−101	1115	2.141	0	0
Griewank	10	5000	5000	4.437	7.396E−05	7.396E−04	3.823E−46	6.679E−46	754	0.797	0	0
	20	7500	7500	12.037	2.054E−18	4.227E−19	8.402E−47	3.928E−46	828	1.735	0	0
	30	10000	10000	23.359	4.638E−18	7.744E−19	4.161E−47	7.523E−47	845	2.672	0	0
Rastrigin	10	5000	5000	3.453	9.934E−18	2.992E−18	4.171E−64	7.834E−64	767	0.672	0	0
	20	7500	7500	9.172	1.928E−17	3.579E−18	7.899E−61	2.215E−60	824	1.313	0	0
	30	10000	10000	17.406	2.962E−17	4.618E−18	6.228E−59	8.496E−59	863	1.953	0	0
Rosenbrock	10	5000	5000	4.672	0.00490144	0.0061382	3.617E−07	5.081E−05	462	0.465	0	0
	20	7500	7500	13.453	0.00541073	0.0090507	9.201E−07	1.102E−05	1191	2.469	0	0
	30	10000	10000	26.625	0.00771188	0.0151573	5.865E−06	8.519E−05	2169	6.469	0	0
Ackley	10	5000	5000	3.906	4.763E−15	1.636E−15	7.105E−19	5.482E−18	63	0.047	0	0
	20	7500	7500	9.657	1.546E−14	2.183E−15	2.131E−19	4.603E−18	85	0.094	0	0
	30	10000	10000	17.860	2.789E−14	2.843E−15	1.460E−18	8.130E−17	109	0.234	0	0
Schaffer	2	2000	2000	0.672	7.895E−11	4.014E−10	1.069E−49	3.170E−49	947	0.468	0	0

dynamically adjust the position of scout bee by allowing a scout bee in the first iteration to wander with a wider step size in the search space. As the number of the iteration increases, the step size for the wandering of a scout bee will decrease.

From the pseudo-code line number 7 to 19, the employed bees can thus still maintain diversity in producing a new food source (Eq. (2.2)). Using both the local and the global search methods, the convergence speed increases and solutions can be globally optimized more quickly.

3.3. Objective-value-based comparison method

In this work, we also focus on the method that is used to compare and to select between the old solution and the new solution in each iteration. Basically, the comparison of the new solution and the old solution is done by the fitness value. If the fitness of the new solution is better than the fitness of the old solution, we select the new one and ignore the old solution. The fitness value can be obtained from the following equation.

For finding the minimum objective value

$$Fitness(f(x)) = \begin{cases} \dfrac{1}{1+f(x)} & \text{if } f(x) \geq 0 \\ 1+|f(x)| & \text{if } f(x) < 0 \end{cases} \tag{3.3}$$

Based on Eq. (3.3), we can see that when $f(x)$ is larger than the zero but has a very small value, e.g. 1E−20, the fitness value of equation $1/(1+1E{-}20)$ is rounded up to be 1 (1E−20 is ignored). This will lead the fitness of all solutions to become equal to 1 in the later iterations. In other words, there is no difference between the fitness values that is equal to $1/(1+1E{-}20)$ and $1/(1+1E{-}120)$. Thus, a new solution that gives a better fitness value than the old solution will be ignored and the solution will stagnate at the old solution. In order to solve this issue, we directly use the objective value of function for comparison and selection of the better solution. The pseudo-code of the new comparison process is shown in Fig. 2.

3.4. Computational complexity

We can evaluate the computational complexity of the best-so-far ABC algorithm in comparison with the original ABC and the BSO algorithms. Let n be the total number of bees, d be the dimension of solutions, and m be the maximum iteration. The best-so-far ABC algorithm must spend time necessary to find the best-so-far solution in each iteration, so it uses the computational time equal to $(n/2)d + m(3nd)$. This is larger than the computational time of the original ABC algorithm which is equal to $(n/2)d + m(3/2nd + d)$. The BSO algorithm must spend time to sort all bees based on their fitness in each of iteration of algorithm, so the computational time is equal to $nd + m(n + nd\log n + 5/2nd))$. However, when we focus on the upper bound of the complexity of problems, both Best-so-far ABC and original ABC algorithms use computational time $O(mnd)$ while the BSO algorithm uses $O(mnd\log n)$, which is the highest complexity.

Although the computational complexity of the best-so-far ABC is higher than the original ABC at the same maximum iteration, the best-so-far ABC should find the optimal solution before it reaches the maximum iteration. Thus the actual computation time of best-so-far ABC may be reduced in relative to original ABC.

4. Using best-so-far ABC in image registration

We applied our best-so-far method to applications in the image registration domain. The following subsections provide a brief background on image registration and the adaptation of our best-so-far method to this problem.

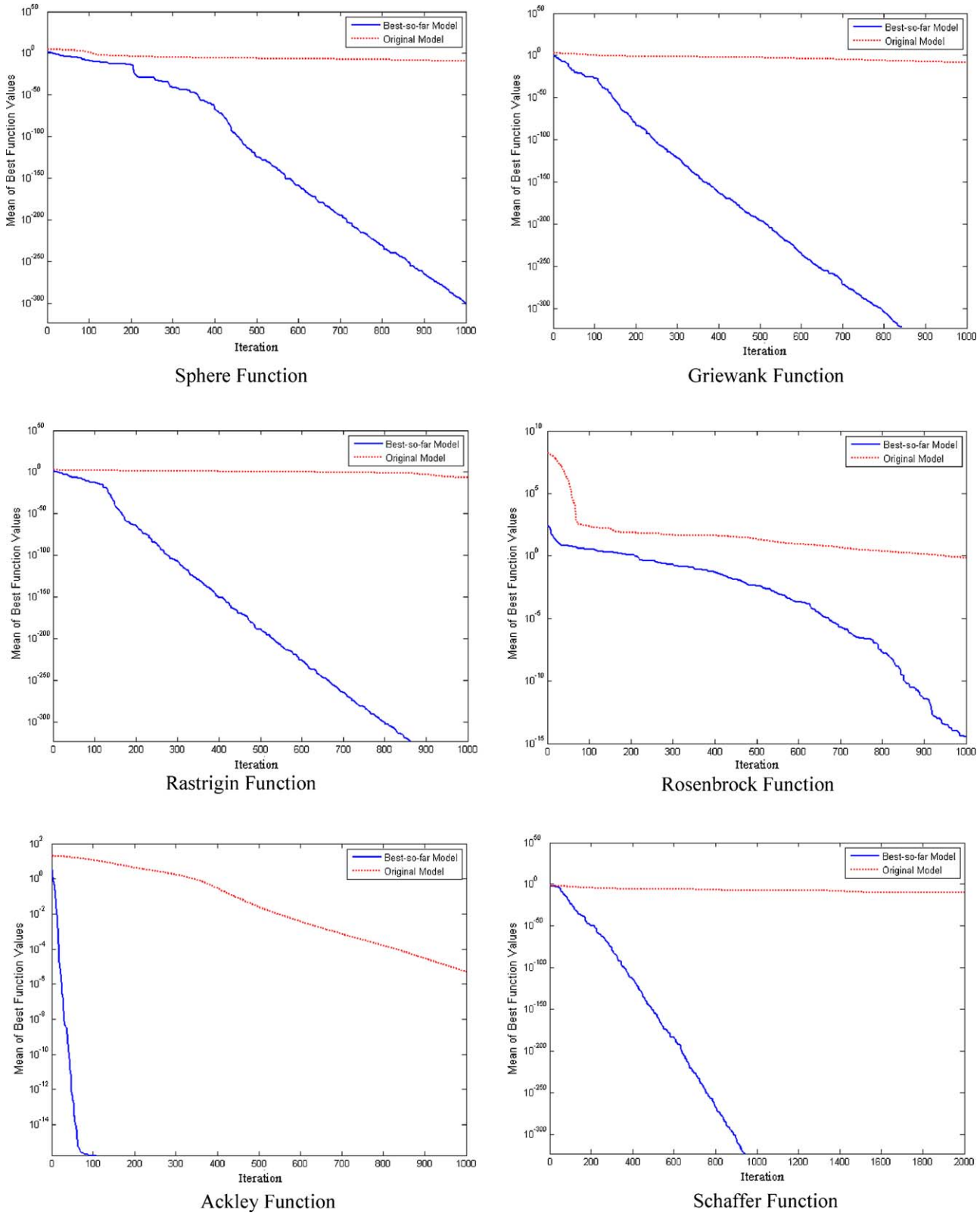

Fig. 4. Iterations to convergence for original and best-so-far algorithms when dimension = 30 (dimension of Schaffer Function = 2).

4.1. Background on image registration

Image registration is a process used to align two or more images of a scene. It plays an important role in many research fields such as the computer vision, remote sensing and medical imaging [17,18]. Usually a target image must be transformed to be geometrically coincident with a reference image. To do this, corresponding features must be identified in the two images or else the global intensity distributions must be compared. A similarity or fitness measure [19] is also necessary. The registration process searches for

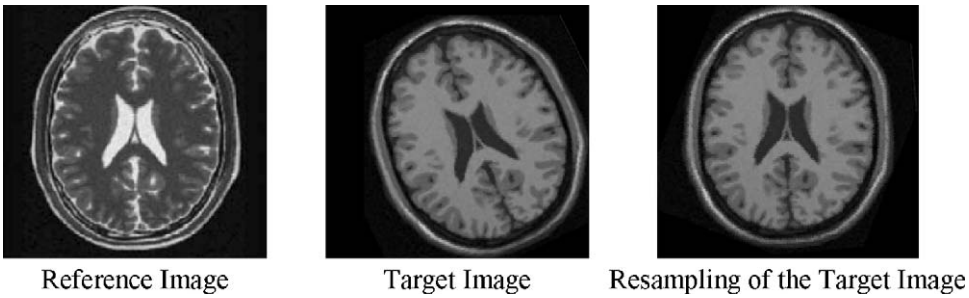

Fig. 5. Image Pair I: Brain image for registration.

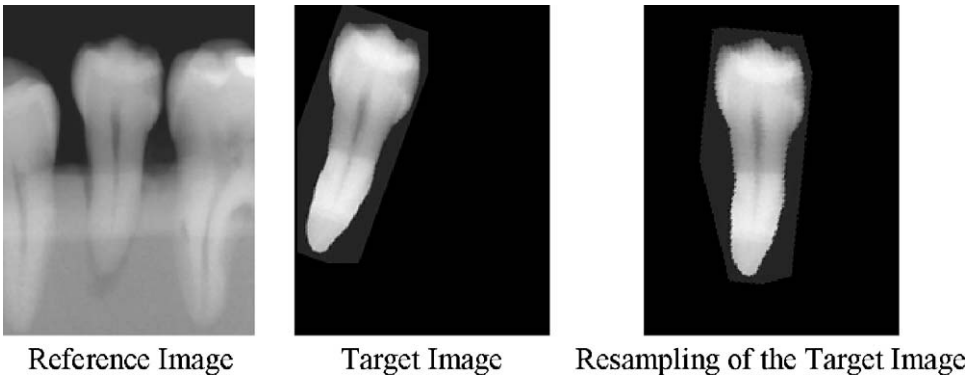

Fig. 6. Image Pair II: Dentistry image for registration.

a transformation from the target to the reference feature geometry that maximizes this similarity measure.

The global optimization of the similarity measurement is the key to automate the registration process. There are two common approaches to measure the similarity, which are geometric distance and statistical similarity. Hessian and Hausdorff geometric distance have been used to measure the distance in computer vision researches [20]. These methods measure the nearest neighbor of rank *k*th. The weakness of the methods is the fact that the presence of outliers may affect the accuracy. Mismatch [21] is then introduced. It is a new measure that makes use of a Gaussian distribution as a weight function in order to be tolerant of outliers.

Least Square Error is the method that statistically determines the smallest deviation or distance between two data sets, such as the reference and target images. The drawback of this measure is its sensitivity to the presence of outliers. Cross Correlation measures similarity of the two images by convolving them. Unfortunately, this measure is also not robust in the presence of noise or outliers. Currently, most of the researchers in this field work with Mutual Information (MI) as a measure of similarity. It is quite similar to correlation except that the concept of MI stems from conditional probability. It is a measure of the presence of the information from one set in another set. It is now becoming a standard to measure the similarity in image registration.

In this paper, we use a statistical similarity approach where the Mutual Information [22–24] is used for measuring the similarity of the two images. An optimization solution in the mutual information approach can be expressed mathematically as follows:

$$\alpha^* = arg\max_{\alpha}(MI(I_R, I_T)) \tag{4.1}$$

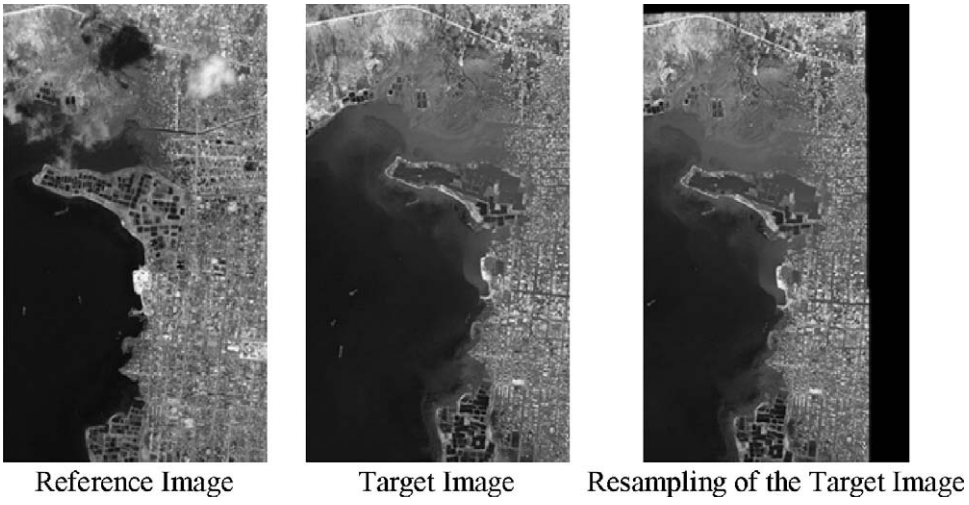

Fig. 7. Image Pair III: Satellite image for registration.

Fig. 8. Image Pair IV: Camera Man image for registration.

where α is the set of transformation parameters, *MI* is an objective function, I_R is a reference image and I_T is a target image.

The mutual information of I_R and I_T can be expressed in term of Shannon's entropy as follows:

$$MI(I_R, I_T) - H(I_R) + H(I_T) - H(I_R, I_T) \tag{4.2}$$

where $H(I_R)$ and $H(I_T)$ are the Shannon's entropies, and $H(I_R, I_T)$ is the joint entropy of the two images.

Shannon's entropy can be defined as:

$$H(I_R) = -\sum_a p_R(a)\,\log(p_R(a)) \tag{4.3}$$

$$H(I_T) = -\sum_b p_T(b)\,\log(p_T(b)) \tag{4.4}$$

$$H(I_R, I_T) = -\sum_a \sum_b p_{R,T}(a, b)\log(p_{R,T}(a, b)) \tag{4.5}$$

where $p_R(a)$ and $p_T(b)$ are the marginal probability mass functions of the reference and the target image respectively, and $p_{R,T}(a,b)$ is the joint probability mass function of two images.

The joint histogram of the image pair can be used to calculate these marginal probability mass functions:

$$p_{R,T}(a, b) = \frac{h(a, b)}{\sum_a \sum_b h(a, b)} \tag{4.6}$$

$$p_R(a) = \sum_b p_{R,T}(a, b) \tag{4.7}$$

$$p_R(b) = \sum_a p_{R,T}(a, b) \tag{4.8}$$

where $h(a,b)$ is a number of corresponding pairs having intensity value a in the reference image and intensity value b in the target image.

Various types of transformations can be applied in image registration. In our work, we used rigid transformations involving rotation and translation along the x-axis and y-axis. The transformation of images can be written as:

$$x' = x_c + (x - x_c) \times \cos\theta + (y - y_c) \times \sin\theta + \Delta x \tag{4.9}$$

$$y' = y_c + (x - x_c) \times \sin\theta + (y - y_c) \times \cos\theta + \Delta y \tag{4.10}$$

where x' and y' are new coordinates, x and y are the current coordinates, x_c and y_c are the center coordinates of the target image, Δx and Δy are the displacements of a target image, and θ is the rotation angle of the target image around its center.

4.2. Optimization solution with best-so-far ABC

We applied best-so-far ABC method to the registration problem described above. The goal is to find a global optimization of the similarity measure. In other words, we try to find the transformation variables that maximize the mutual information values of the images. The adopted algorithm is illustrated in Fig. 3.

In Fig. 3, initial solutions consisting of a rotation, and x and y-axis translation values are generated. These parameter sets are treated as the food sources for the employed bees. Each solution is used to transform the target image by re-sampling. Then we calculate the mutual information score between the transformed target and the reference image. The onlooker bees will then select the solutions that produce higher mutual information and update those solutions based on our best-so-far method. The process will be repeated until the mutual information reaches a threshold value or the number of iteration equals the MCN. Solutions that cannot improve the mutual information within a certain period will be abandoned and new solutions will be regenerated by the scout bee.

Eq. (2.2) is used by employed bees in order to update variables and improve the solution quality in each iteration. However, due to the independence among the transformation parameters in image registration application, Eq. (3.1), which uses a randomly chosen dimension, is slightly adjusted. A fixed dimension is used instead as shown in Eq. (4.11).

$$v_{id} = x_{id} + \Phi f_b(x_{id} - x_{bd}) \tag{4.11}$$

In our algorithm, Eq. (4.11) is introduced to improve the solutions in each iteration. When the solutions cannot be further improved, Eq. (2.2) is used to calculate the new candidate solutions. Moreover, Eq. (3.2) is used by scout bees to randomly generate the new solutions when the mutual information cannot be improved.

5. Experiments

We validated the performance of the proposed technique compared with the original technique in two sets of experiments: numerical and image registration applications. The following subsections describe the experimental methodology.

5.1. Numerical applications

The aim of this experiment is to compare the performance of the search process for the best-so-far method with the original algorithm and the Bee Swarm Optimization that is the state-of-the-art algorithm for solving numerical optimization based on behavior of bees proposed by R. Akbari [16]. Unimodal and multimodal benchmark functions as shown in Table 1 were used in this experiment. The objective of the search process is to find the solutions that can produce the minimum output value from these benchmark functions. In other words, the aim is to minimize $f_i(\vec{x})$ in Table 1. In order to make a performance comparison, we implemented the original ABC based on an algorithm given in [25] as well as our best-so-far method. Our original ABC code was also validated by comparing the benchmark results with the previous work proposed in [10]. Since we could not obtain the source code of the BSO algorithm, we only use the results reported in [16] to compare with our algorithm. As a

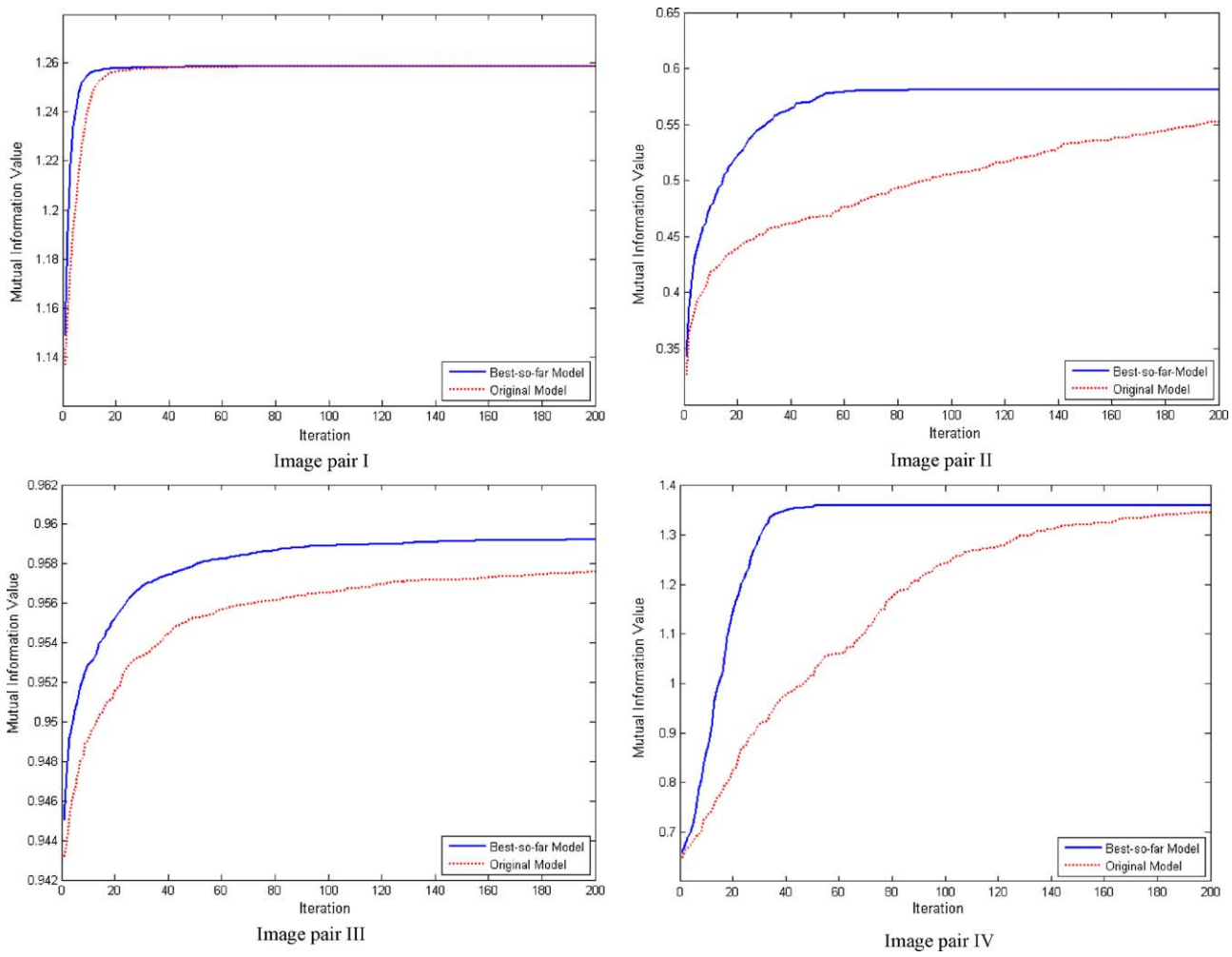

Fig. 9. The mean of the mutual information across iterations in Image Pair I–IV based on fixed initial solutions.

result, the BSO algorithm will not be included in some comparison experiments.

The experiments were conducted in a similar fashion as described in [10,16]. The number of employed and onlooker bees were set to 100. The values of $\omega_{\max}$ and $\omega_{\min}$ are set to 1 and 0.2, respectively. Each of the experiments was repeated 100 times with different random seeds. All the experiments in this paper were run on the same hardware (Intel Core 2 Quad with 2.4 GHz CPU and 4 GB memory). The number of iterations to convergence, the runtime (execution time), and the mean and standard deviation of the output values of benchmark functions were recorded. Note that lower mean objective values indicate better solutions.

In order to investigate the effect of the iteration increment on the output quality, we divided the experiments into two sets, called ABC1 and ABC2. The maximum numbers of iterations proposed in [10,16] were also used in our work. In the ABC1 experiment, for all benchmark functions with exceptions of Schaffer function, the numbers of maximum iterations were 500, 750 and 1000 for the dimensions of 10, 20 and 30, respectively. In ABC2, the maximum numbers of iterations were 5000, 7500 and 10,000 for the dimensions of 10, 20 and 30, respectively. For Schaffer function in both of ABC1 and ABC2 experiment, the numbers of maximum iterations was set to 2000. Note that one iteration of our method requires a longer execution time than the original method because the values in all dimensions of each candidate food source of onlooker bees are updated on every iteration and the algorithm spends the time to find the best-so-far solution in each iteration. However, the number of iterations to convergence is smaller. Thus, we will look at both the number of iterations and runtime when we consider convergence speed.

5.2. Image registration application

In our image registration experiments, once again we compare both the solution quality and execution performance between our best-so-far and original ABC implementation. The objective function of the image registration application is *Mutual Information* or MI. MI is used to measure the similarity of the two input images after registration. Thus, the higher the MI value, the more accurate the registration process.

In our experiments we used four sample image pairs. For each pair the experiments were repeated 30 times. The number of employed and onlooker bees used were 50 and the maximum number of iterations was 200. The search space of each variable for rigid transformation process in term of rotation degree (θ), translation in x-axis (x) and y-axis (y) were defined as shown below. The ranges of these parameters were chosen based on a visual estimate of the maximum amount of change required.

Image Pair	(θ)	(x)	(y)
I	$-25^\circ \le \theta \le 25^\circ$	$-20 \le x \le 20$	$-20 \le y \le 20$
II	$0^\circ \le \theta \le 50^\circ$	$-210 \le x \le 210$	$-160 \le y \le 160$
III	$-1^\circ \le \theta \le 1^\circ$	$-10 \le x \le 10$	$-50 \le y \le 0$
IV	$-90^\circ \le \theta \le 90^\circ$	$-256 \le x \le 256$	$-256 \le y \le 256$

We tested the algorithms with two cases. In the first case, we used a controlled initial solution with the fixed initial values of θ, x, and y. We offset all the initial values from the lower bound displayed in the table above. However, the initial values may accidentally be fixed close to the optimum solution. Thus, in the second test case, we used randomly generated initial values.

In both test cases we investigated the solution quality of the two algorithms when the runtime is approximately the same as well as studying the algorithm performance when the solution quality is approximately the same. In other words, we would like to know which algorithm uses less computation time to create a solution of the equivalent quality.

6. Results and discussion

6.1. Numerical results and discussion

The results obtained with the original and the best-so-far method based on the numerical benchmark functions are shown in Table 2. Note that there is no result of the BSO algorithm to compare in the ABC1 experiment.

The "Convergence Iteration" column shows the number of iterations needed for each algorithm to converge towards the optimal solution (value not to exceed the maximum number of iterations indicated in section 6.1). The "Runtime" column is the average runtime needed to reach the convergence state across a hundred runs. The "Mean" column is the average output values of benchmark functions. The "SD" column shows the standard deviation of the results. Mean and SD values which are smaller than 1E−304 are considered as 0.

In Table 2, we compare the results using approximately the same runtime value. The best-so-far method gives better solutions than the original algorithm in all cases of benchmark functions. Especially, the global minimal values of Ackley and Schaffer functions were found by the best-so-far method in ABC1 experiment.

The results from the ABC1 experiment showed that several benchmark functions did not converge within the specified number of iterations. The maximum number of iterations was then increased in the ABC2 experiment. The results are shown in Table 3.

In the ABC2 experiment in Table 3, when the number of iterations was increased, the solution quality was also improved in the original ABC as shown in mean of objective values column. However, our best-so-far method was still able to generate a better solution on all the benchmark functions. The average improvement on runtime for the best-so-far method when compared with the original ABC was 82%, and the maximum and minimum runtime improvement were 99% and 30% for Ackley and Schaffer function, respectively. For several benchmark functions, the original ABC never reached the mean value achieved by our best-so-far method even when the runtime was increased.

Table 4
Rate of convergence in the ABC2 experiment.

Function	Dimensions	Rate of convergence	
		ABC	Best-so-far ABC
f1 Sphere	10	0.9265	0.5856
	20	0.9942	0.6691
	30	0.9954	0.7124
f2 Griewank	10	0.9966	0.5358
	20	0.9890	0.6032
	30	0.9908	0.5829
f3 Rastrigin	10	0.9270	0.5622
	20	0.9662	0.6073
	30	0.9787	0.6246
f4 Rosenbrock	10	0.9965	0.8705
	20	0.9972	0.9492
	30	0.9978	0.9722
F5 Ackley	10	0.9928	0.5683
	20	0.9948	0.6791
	30	0.9964	0.7445
f6 Schaffer	2	0.9907	0.6849

The results from the BSO method reported in [16] were also used to compare with the Best-so-far method in the ABC2 experiment. The results showed that the mean value of Best-so-far method is better than the BSO method at the same number of the iterations on all benchmark functions.

Fig. 4 shows the comparison of the convergence speed (Iterations) in term of number of iterations between the original and the best-so-far method. Notice that the best-so-far method converges substantially faster with a much smaller number of iterations needed.

Note that there is no result of the BSO algorithm to illustrate the convergence speed, so the comparison with the BSO algorithm is not included in here.

The rate of convergence is used to indicate the speed at which a converging sequence approaches its limit. A smaller value of the rate of convergence implies that the solution converges to an optimal value more quickly. Suppose that the sequence $\{x_k\}$ converges to the number L, so the rate of convergence can be calculated from equation as follows

$$\textit{rate of convergence} = \lim_{k\to\infty} \frac{|x_{k+1} - L|}{|x_k - L|} \tag{6.1}$$

Table 5
The results obtained from the original ABC and best-so-far ABC method based on fixed initial solutions at approximately the same runtime value.

Image pair	Image size ($H \times W$)	ABC				Best-so-far ABC			
		Convergence Iteration	Runtime (s)	Mean (MI)	SD (MI)	Convergence Iteration	Runtime (s)	Mean (MI)	SD (MI)
I	253 × 255	186	687.699	1.25875	9.55E−05	179	684.106	1.25883	8.46E−05
II	210 × 160	199	413.281	0.552284	3.31E−02	200	411.116	0.581292	1.76E−02
III	299 × 176	200	636.353	0.957611	9.28E−04	196	635.344	0.959218	2.37E−04
IV	256 × 256	200	723.533	1.3449	3.00E−02	188	719.947	1.36046	7.57E−05

Table 6
The results obtained from the original ABC and best-so-far ABC method based on fixed initial solutions at approximately the same mean MI value.

Image pair	Image size ($H \times W$)	ABC				Best-so-far ABC			
		Convergence Iteration	Runtime (s)	Mean (MI)	SD (MI)	Convergence Iteration	Runtime (s)	Mean (MI)	SD (MI)
I	253 × 255	186	687.699	1.25875	9.55E−05	108	412.756	1.25875	8.53E−05
II	210 × 160	199	413.281	0.552284	3.31E−02	32	65.778	0.553137	1.81E−02
III	299 × 176	200	636.353	0.957611	9.28E−04	44	142.628	0.957646	3.14E−04
IV	256 × 256	200	723.533	1.3449	3.00E−02	37	141.691	1.34501	7.66E−05

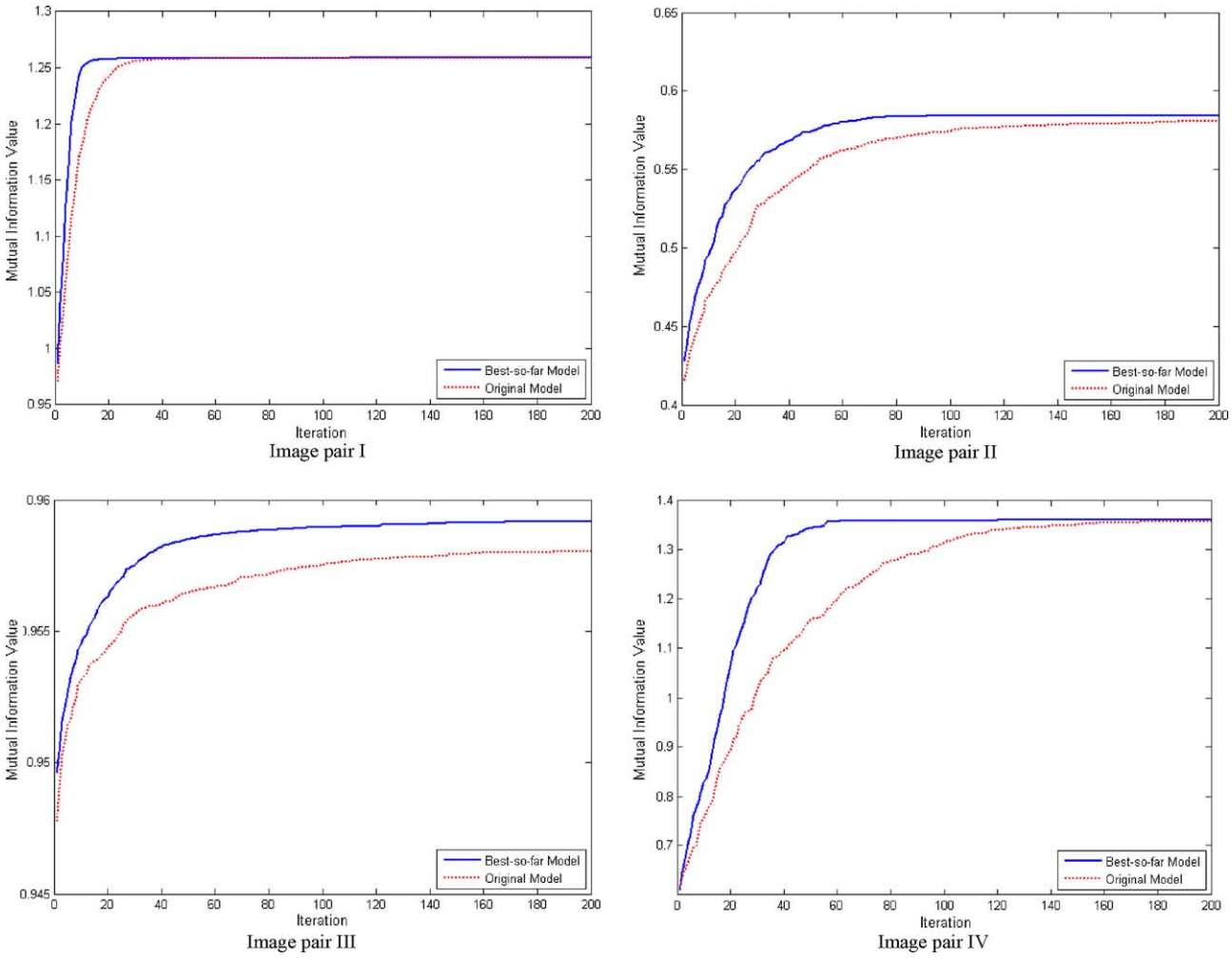

Fig. 10. The mean of the mutual information in Image Pair I–IV based on random initial solutions.

The results in Table 4 show that both the ABC and the Best-so-far ABC converge linearly to the limit (global minimum). However, when we compare the rates of convergence, we found that the Best-so-far ABC gives better results than the ABC on all benchmark functions, especially in the Griewank and Rastrigin functions.

In summary, the results in the numerical experiments indicate that the best-so-far ABC method can converge to the optimal solution more quickly on almost all benchmark functions when it is compared with the original ABC method. The results also show that the best-so-far ABC method can produce better solutions than the BSO algorithm that is the state-of-the-art algorithm. In other words, fewer iterations were needed to converge and thus, less computational time was required. Notice that the SD is relatively low, which implies consistency among experimental runs.

6.2. Image registration results and discussion

In this experiment, we used both the original and the best-so-far methods to register four sets of images. The two methods produced results that appeared indistinguishable to human eyes. However, analysis of MI for registered images suggests that the best-so-far method offered advantages.

The registered images are shown in Figs. 5–8. However, if we investigate the resulting numbers closely, the best-so-far produces better results faster as discussed next.

First, we discuss the results of the experiments for the fixed initial solution. Table 5 and Fig. 9 show the results in a case where the runtime of both methods is approximately the same. In Table 6, the means of MI values are fixed to be approximately the same.

In Table 5, we compare the results using approximately the same runtime value. The best-so-far method gives slightly better solutions than the original algorithm in all cases of sample image pairs, with somewhat faster convergence. Table 6 shows the convergence speed at approximately the same mean value of MI. The results indicate that the best-so-far method solutions converged to an optimal solution more quickly than the original method in all image pairs. The maximum runtime improvement of 84% was found in the experiment with image pair II and the minimum of 39% was with image pair I. The average runtime improvement for all image pairs was 70%.

For the random initial solution experiments, in some cases, if the initial solution is close to the optimal solution, the results from the first few iterations of the original algorithm generate better results. However, when the number of iterations increases, the best-so-far method will adjust the solutions and converge to the optimal solutions more quickly. The results of these experiments are shown in Tables 7 and 8 and Fig. 10.

In the random initial solutions experiment, although the MI values from the original method are equivalent to the MI values from the best-so-far method at approximately the same runtime value as shown in Table 7, the results in Table 8 show that the solutions of the same case still converge to an optimal solution faster than the original algorithm when the approximate mean value is the same. The maximum and minimum percentage of the runtime improvement,

Table 7
The results obtained from the original ABC and best-so-far ABC method based on random initial solutions by using approximately the same runtime value.

Image pair	Image size ($H \times W$)	ABC				Best-so-far ABC			
		Convergence Iteration	Runtime (s)	Mean MI	SD MI	Convergence Iteration	Runtime (s)	Mean MI	SD MI
I	253×255	189	714.005	1.25867	9.53E-05	196	712.789	1.25878	1.13E−04
II	210×160	199	400.781	0.580997	2.89E−03	187	399.604	0.58439	3.26E−04
III	299×176	199	641.630	0.958047	8.09E−04	200	629.072	0.959175	6.32E−04
IV	256×256	199	759.200	1.35800	5.38E−03	200	758.391	1.36049	4.54E−05

Table 8
The results obtained from the original ABC and best-so-far ABC method based on random initial solutions by using approximately the same mean MI value.

Image pair	Image size ($H \times W$)	ABC				Best-so-far ABC			
		Convergence Iteration	Runtime (s)	Mean MI	SD MI	Convergence Iteration	Runtime (s)	Mean MI	SD MI
I	253×255	189	714.005	1.25867	9.53E−05	82	298.207	1.25867	1.17E−04
II	210×160	199	400.781	0.580997	2.89E−03	65	138.899	0.581029	3.44E−04
III	299×176	199	641.630	0.958047	8.09E−04	38	119.524	0.958108	6.56E−04
IV	256×256	199	759.200	1.35800	5.38E−03	59	223.725	1.35838	4.62E−05

found on the experiment of image pair III and image pair I, were equal to 81% and 58%, respectively. The average improvement on the runtime for the best-so-far method compared with the original model on all experiments of image pairs in this case was 68%.

7. Conclusion

In this paper, a best-so-far method for solution updates in the Artificial Bee Colony algorithm was proposed. We have replaced the neighboring solutions-based approach with the best-so-far technique in order to increase the local search ability of the onlooker bees. The searching method based on a dynamic adjustment of search range depending on the iteration was introduced for scout bees. The comparison and the selection of the new solution were changed from a fitness-based comparison to an objective-value-based comparison that can help to resolve round up issues in the computation of the floating point "goodness" value.

In our best-so-far method, onlooker bees compare the information from all employed bees to select the best-so-far candidate food source. This will bias the solution handled by onlooker bees towards the optimal solution. Moreover, if the solution appears to stagnate in a local optimum, the scout bee can randomly generate a new position in order to maintain the diversity of new food sources.

The performance of the best-so-far ABC method was then compared with the original ABC algorithm and the BSO algorithm using a set of benchmark functions. The computational complexity and the rate of convergence on both best-so-far ABC method and the original ABC method were addressed. The results from the experiments provide evidence that the best-so-far ABC outperforms all the mentioned algorithms in both quality and convergence rate. We further applied the best-so-far method to optimize the mutual information in an image registration application. Several image pairs were used in the experiments. The results showed that our algorithm can arrive at the convergence state more quickly. Lastly, the mutual information value produced by the best-so-far method is higher implying that the registration quality is also enhanced. Thus, we can conclude that our best-so-far ABC is efficient from both the perspective of solution quality and algorithm performance. The algorithm can serve as an alternative in several application domains in the future.

Acknowledgments

This work is supported by the Thailand Research Fund through the Royal Golden Jubilee Ph.D. Program under Grant No. PHD/0038/2552.

References

[1] M.H. Aghdam, N. Ghasem-Aghaee, M.E. Basiri, Text feature selection using ant colony optimization, Expert Systems with Applications: An International Journal 36 (2009) 6843–6853.
[2] B. Yagmahan, M.M. Yenisey, Ant colony optimization for multi-objective flow shop scheduling problem, Computers & Industrial Engineering 54 (2008) 411–420.
[3] R.E. Perez, K. Behdinan, Particle swarm approach for structural design optimization, Computers and Structures 85 (2007) 1579–1588.
[4] P.Y. Yin, Particle swarm optimization for point pattern matching, Journal of Visual Communication and Image Representation 17 (2006) 143–162.
[5] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, USA, 2004.
[6] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the 1995 IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.
[7] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, Turkey, 2005.
[8] D. Karaboga, B. Akay, A comparative study of Artificial Bee Colony algorithm, Applied Mathematics and Computation 214 (2009) 108–132.
[9] D. Karaboga, B. Basturk, On the performance of Artificial Bee Colony (ABC) algorithm, Applied Soft Computing 8 (2008) 687–697.
[10] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization 39 (2007) 459–471.
[11] A. Singh, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, Applied Soft Computing 9 (2009) 625–631.
[12] F. Kang, J. Li, Q. Xu, Structural inverse analysis by hybrid simplex artificial bee colony algorithms, Computers and Structures 87 (2009) (2009) 861–870.
[13] N. Karaboga, A new design method based on artificial bee colony algorithm for digital IIR filters, Journal of the Franklin Institute 346 (2009) 328–348.
[14] X.S. Yang, Engineering optimizations via nature-inspired virtual bee algorithms, in: Lecture Notes in Computer Science, Springer (GmbH), 2005, pp. 317–323.
[15] K. Sundareswaran, V.T. Sreedevi, Development of novel optimization procedure based on honey bee foraging behavior, in: IEEE International Conference on Systems, Man and Cybernetics, 2008, pp. 1220–1225.
[16] R. Akbari, A. Mohammadi, K. Ziarati, A novel bee swarm optimization algorithm for numerical function optimization, Communications in Nonlinear Science and Number Simulation 15 (2010) 3142–3155.
[17] H. Chen, P. Varshney, M. Arora, Mutual information based image registration for remote sensing data, International Journal of Remote Sensing 24 (2003) 3701–3706.
[18] W. Jacquet, E. Nyssen, P. Bottenberg, B. Truyen, P. de Groen, 2D image registration using focused mutual information for application in dentistry, Computers in Biology and Medicine 39 (2009) 545–553.
[19] A. Goshtasby, 2-D and 3-D Image Registration for Medical, Remote Sensing, and Industrial Applications, John Wiley & Sons, New Jersey, USA, 2005.
[20] D.M. Mount, N.S. Netanyahu, J. LeMoigne, Efficient algorithms for robust point pattern matching and applications to image registration, Pattern Recognition 32 (1999) 17–38.
[21] S. Ratanasanya, D.M. Mount, N.S. Netanyahu, T. Achalakul, Enhancment in Robust Feature Matching, ECTI-CON, 2008, pp. 505–508.

[22] T.M. Cover, J.A. Thomas, Elements of Information Theory, John Wiley & Sons, New York, USA, 2006.
[23] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, P. Suetens, Multimodality image registration by maximization of mutual information, IEEE Transactions on Medical Imaging 16 (1997) 187–198.
[24] P. Viola, W.M. Wells, Alignment by maximization of mutual information, International Journal of Computer Vision 24 (1997) 137–154.
[25] D. Karaboga, Artificial Bee Colony (ABC) Algorithm Homepage [Online], Available: http://mf.erciyes.edu.tr/abc [2009].