2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Bacterial Foraging Algorithm For Dynamic Environments

W. J. Tang, Q. H. Wu, *Senior Member, IEEE,* and J. R. Saunders

***Abstract*— Optimization in dynamic environments has received great attention in recent years [1]. Different from static optimization problems, its convergence and searching ability is cautiously desired. Over the last two decades, Evolutionary Algorithms (EAs), designed to solve the static optimization problems, have been comprehensively and intensively investigated. In recent years, as the emergence of another member of the EA family – bacterial foraging algorithm (BFA), the self-adaptability of individuals in the group searching activities has attracted a great deal of interests. In this paper, a BFA aiming for optimization in dynamic environments, called DBFA, is studied. A test bed proposed previously in [2] is adopted to evaluate the performance of DBFA. The simulation studies offer a range of changes in a dynamic environment. The simulation results show that DBFA can adapt to various environmental changes which occur in different probabilities, with both satisfactory accuracy and stability, in comparison with a recent work on bacterial foraging [3].**

I. Introduction

Static optimization problems have been the focus of evolutionary computation for a long time. However, there are many practical problems in various fields in the real word, which need optimization methodologies suitable for a changing environment. There have already been some previous research to tackle this issue [4]-[8]. One of these approaches detects a change in the environment, and then adjusts the algorithm parameters to increase the diversity or probability of mutation, which, on the other hand, destroys the information gained by previous search. Another maintains a certain diversity throughout the evolutionary process, for example, by introducing random immigrants; or taking the ages of individuals into account, which nonetheless, still disturbs optimization process. Recently, a new algorithm, called "Memory - Enhanced Approach" [9] [10], claimed to cope with periodically changing environments. The performance of this algorithm depends on memorizing the history of optimization process and maintaining the diversity of population. Therefore, the problem of convergence still exist. Inspired from the nature, there is a new group of approaches investigated recently, called "Multi-Population Approach", which gains much more concerns [12] [13]. It proposes the use of a number of subpopulation groups for covering possible solutions, and enables itself to detect new optima by maintaining a suitable diversity. "Self-organizing Scouts" is an example of this approach [4]. Furthermore, there are other ideas to deal with the dynamic problems, such as "thermodynamical GA"[14], ACO for dynamic problems [15] and varying population swarming [16]. However, the problems of confliction between convergence and diversity still exist in these algorithms.

For dynamic problems, rapid convergence, which is an important characteristic for algorithms used in static optimization problems, is not only desired, but also the ability of finding a global optimum is required. However these two requirements are contradictory to each other. There should be a compromise between the convergence and the diversity of the algorithm designed for solving a specific problem. Some of the reported results are encouraging. However, most of these methods were evaluated in the periodically changing environments or they possess intensive computation, as detection of environmental changes is required in each search step, which are either too hypothetic or unrealistic for the complexity of real-world problems.

However, the complex while organized activities exhibited in bacterial foraging patterns could inspire a new solution for dynamic problems. The underlying mechanism of the surviving of bacteria, especially *E.coli* in a changing environment has been reported by researchers in the area of biological sciences [17]. Inspired from these phenomena, an optimization algorithm, called BFA, was introduced in [3], which is known to be useful for applications in control [3] or parameter estimation [18]. Based on BFA, we propose a DBFA, which is especially designed to deal with dynamic optimization problems, combining the advantage of both local search in BFA and a new selection scheme for diversity generating.

We use the moving peaks benchmark (MPB) [2] as the test bed for experiments. The performance of the DBFA is evaluated in two ways. The first is concerned with the convergence of the algorithm in random - periodical changes in an environment, which are divided into three ranges from a low probability of changes to a higher one. The second is testing a set of combinations of the algorithm parameters which are largely related to the accuracy and stability of the algorithm. All results are compared with the existing BFA [3], and show the effectiveness of DBFA for solving dynamic optimization problems.

The remainder of this paper is organized as follows: Section II provides a brief introduction to the existing BFA. The DBFA is described in Section III. A short description of the test bed, the experiment design, as well as the criteria used to evaluate the performance of DBFA, is given in IV, followed by the simulation studies and discussions given in Section V. The conclusion is drawn in Section VI.

W. J. Tang and Q. H. Wu are with the Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, L69 3GJ, U.K.

J. R. Saunders is with School of Biological Sciences, The University of Liverpool, Liverpool, L69 3BX, U.K.

Corresponding author: Q. H. Wu, Tel: ++44 151 7944535; Fax: ++44 151 7944540; Email: qhwu@liv.ac.uk

0-7803-9487-9/06/$20.00/©2006 IEEE

II. Bacterial Foraging Algorithm

Bacterial foraging algorithm is inspired by the patten exhibited by bacterial foraging behaviours. Bacteria have the tendency to gather to the nutrient-rich areas by an activity called "chemotaxis". It is known that bacteria swim by rotating whip-like flagella driven by a reversible motor embedded in the cell wall. *E. coli* has 8-10 flagella placed randomly on a cell body. When all flagella rotate counterclockwise, they form a compact, helically propelling the cell along a helical trajectory, which is called *Run*. When the flagella rotate clockwise, they all pull on the bacterium in different directions, which causes the bacteria to *Tumble*.

A. Chemotaxis

Bacterial Chemotaxis is based on the suppression of tumbles in cells that happen by chance to be moving up in a gradient direction. Bacteria make decision according to their ambient environment. The motion of individual peritrichously flagellated bacteria can be described in terms of run intervals during which the cell swims approximately in a straight line interspersed with tumbles, when the organism undergoes a random reorientation.

In the existing BFA, a unit walk with random direction represents a *Tumble* and a unit walk with the same direction in the last step indicates a *Run*, as shown in Fig. 1. After one step move, the position of the ith bacterium can be represented as

$$\theta_i(j+1,r,l) = \theta_i(j,r,l) + C(i)\angle\phi(j) \qquad (1)$$

where $\theta_i(j,r,l)$ indicates the position of the ith bacterium at the jth chemotactic step in the rth reproductive loop of the lth elimination and dispersion event; $C(i)$ is the length of a unit walk, which is set to be a constant; and $\phi(j)$ is the direction angle of the jth step. When its activity is *Run*, $\phi(j)$ is the same with $\phi(j-1)$; otherwise, $\phi(j)$ is a random angle generated within a range of [0,2π].

With the activity of run or $tumble$ taken at each step of the chemotaxis process, a step fitness, denoted as $J_i(j,r,l)$, will be evaluated.

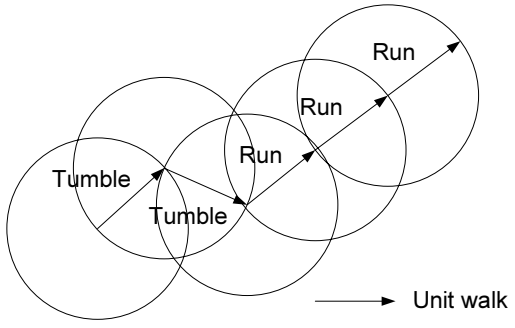


Fig. 1. *Tumble* and *Run*

B. Reproduction

The total fitness of each bacterium is calculated as the sum of the step fitness during its life, i.e. $\sum_{j=1}^{Nc} J_i(j,r,l)$ which is obtained after all chemotactic steps, where N_{c} is the maximum step in a chemotaxis process. All bacteria are sorted in reverse order according to their fitness. In the reproduction step, only the first half of population survive and a surviving bacterium splits into two identical ones, which occupy the same positions in the environment at 1st step. Thus, the population of bacteria keeps constant in each chemotaxis process.

C. Dispersion and elimination

The chemotaxis provides a basis for local search, and the reproduction process speeds the convergence which has been demonstrated by Passino [3]. While to a large extent, only chemotaxis and reproduction are not enough for global optima searching. Since bacteria may get stuck around the initial positions or local optima, it is possible for the diversity of BFA to change either gradually or suddenly to eliminate the accidents of being trapped into the local optima. In the BFA, the dispersion event happens after a certain number of reproduction processes. A bacterium is chosen, according to a preset probability p_{ed}, to be dispersed and moved to another position within the environment. These events may prevent the local optima trapping effectively, but unexpectedly disturb the optimization process. The detailed work can be found in [3].

III. BFA for dynamic environment (DBFA)

The performance of BFA in static environments has been reported in detail [3]. The process of "Chemotaxis" enables bacteria to obtain a satisfactory ability of local search. It is worth notice that the individuals in BFA could converge rapidly without information sharing between each other, which is different from most of EAs.

While in dynamic environments, a rapid convergence needs to be reconsidered as the environment is changing and a fast convergence may not lead to an effective trace of the global optimum in all possible directions. The reproduction process of BFA aiming to speed the convergence is suitable in static problems, but lack of adaptation in dynamic environments. Thus, in order to compromise between rapid convergence and high diversity, we propose a dynamic bacterial foraging algorithm (DBFA) in which a selection process is introduced using a more flexible scheme to enable a better adaptability in a changing environment. The basic idea of the DBFA is to maintain a suitable diversity for global search, while the local search ability is not degraded, and also consider changes in the environment. The scheme is described as follows:

$$J_i = \sum_{j=1}^{n} J_i(j,r) \tag{2}$$

$$rank_i = sort(J_i) \tag{3}$$

$$W_i = m\frac{(rank_i)^k}{\sum_{i=1}^{P}(rank_i)^k} + (1-m)\frac{J_i}{\sum_{i=1}^{P} J_i} \tag{4}$$

where n is the number of chemotactic steps (each step may contain a *Run* or *Tumble*) during a bacterium's life time, j is its index and P is the population size, m is the weight of diversity, and k is the exponent of $rank_i$. At rth selection step, the fitness of bacterium i, J_i, is still the sum of the step fitness during its life as indicated in equation (2), which has been redefined as $J_i(j,r)$, since there is no dispersion events. Thus, those experienced more nutrient-rich areas are more likely to be selected as a parent for next generation. However, this domination would not help diversity maintaining. Therefore, the combination of the solution and rank is chosen to prevent a rapid convergence which should be avoided to keep an adaptation ability of the DBFA for dynamic environments. This adaptation ability is improved as long as the rank of each individual functions as a fitness-independent factor. Thus, the whole population is sorted according to J_i using an operator *sort*, then $rank_i$ is allocated as the rank of bacterium i in equation (3). We introduce the parameter, m, which affects the diversity, to the selection process by combining the rank of the bacterium $(rank_i)^k$ with the fitness calculation J_i. The two factors are balanced by m. The survival probability of bacterium i, W_i, is given in equation (4), and

$$\sum_{i=1}^{P} W_i = 1$$

At last, the roulette wheel selection taken from the GA literature is adopted to generate the next generation.

As the diversity could be obtained in each generation, i.e. a chemotactic process which contains a number of chemotactic steps, the process of "dispersion and elimination" is not considered in this algorithm. The pseudo code of DBFA is described in Table I, where N_{s} indicates the number of selections, N_{c} represents the number of chemotactic steps in a bacterium's life time, $Jlast$ is a temporary variable in the process of *Run* and N_{r} is the maximum number of steps for a single activity of *Run*.

TABLE I

PSEUDO CODE FOR DBFA

Randomly initialize positions of bacteria in the domain;	
FOR (Selection $r = 1 : N_{\mathrm{s}}$)	
FOR (chemotactic steps per bacterium $j = 1 : N_{\mathrm{c}}$)	
Calculate:	Calculate the nutrient function of bacterium i as $J_i(j,r)$;
Tumble:	For bacterium i, set $J_i(j,r)$ as *Jlast*. Generate a random angle represented by an array Δ, where each element belongs to $[0,1]$; Move to a random direction $\frac{\Delta}{\sqrt{\Delta^T \times \Delta}}$ by a unit walk, the new position is calculated by equation (1); Start another chemotactic step.
Run:	For bacterium i, calculate the step fitness as $J_i(j,r)$. If $J_i(j,r) < Jlast$, take another unit walk of the same direction, set $J_i(j,r)$ as *Jlast*; otherwise, start another chemotactic step; Continue the **Run** until N_{r} steps before start another chemotactic step;
END FOR (chemotactic steps)	
Sum:	Set J_i as the sum of the step fitness over the life time of bacterium i using equation (2);
Sort:	Sort J_i in ascending values of fitness in the population;
Select:	Calculate the rank of bacterium i according to equation (3); Obtain W_i for bacterium i by equation (4); Select bacteria by using roulette wheel selection.
END FOR (Selection)	

IV. EXPERIMENT DESIGN

A. Environment setting

The experiment is set in a testing environment called Moving Peaks Benchmarks (MPB), which is also called DF1 [2]. The dynamic function introduced for general purposes is a "field of cones", as an objective function defined as follows:

$$Z = -\max_{i=1,\ldots,N}\{H_i - R_i\sqrt{(X-X_i)^2 + (Y-Y_i)^2}\} \tag{5}$$

where N is the number of cones in the environment. For the ith cone, H_i indicates its height, R_i is the slope control variable, and (X_i, Y_i) represents the coordinate of its center. The initialization of parameters is shown in Table II.

TABLE II

PARAMETER SETTINGS

Parameter	Value
N	15
H_1	0
R_1	0
X_1	0.5
Y_1	0.5
H_i	[1,10]
R_i	[8,20]
X_i	[-1,1]
Y_i	[-1,1]
i	$2,\ldots,N$

The parameters listed in Table II can be adjusted to change the environment. But in our simulation studies, the height and range of slope for each cone are set to be constant, and only the positions of the cones are changing. In this case, $x_{\mathrm{step}}, x_{\mathrm{step}}$ are the step sizes in x and y direction respectively. And for each step i, X_{i+1} and Y_{i+1} are calculated as

follows:

$$x_{\text{step}} = A_x \times x_{\text{step}}(1 - x_{\text{step}}) \quad (6)$$

$$y_{\text{step}} = A_y \times y_{\text{step}}(1 - y_{\text{step}}) \quad (7)$$

$$X_{i+1} = X_i + x_{\text{step}} Dx_i \quad (8)$$

$$Y_{i+1} = Y_i + y_{\text{step}} Dy_i \quad (9)$$

where A_x and A_y are both a constant, respectively. Dx_i and Dy_i can be assigned 1 or -1 with probability 0.5, respectively. An example of the dynamic environment, generated with DF1, in four steps, is illustrated in Fig. 2.

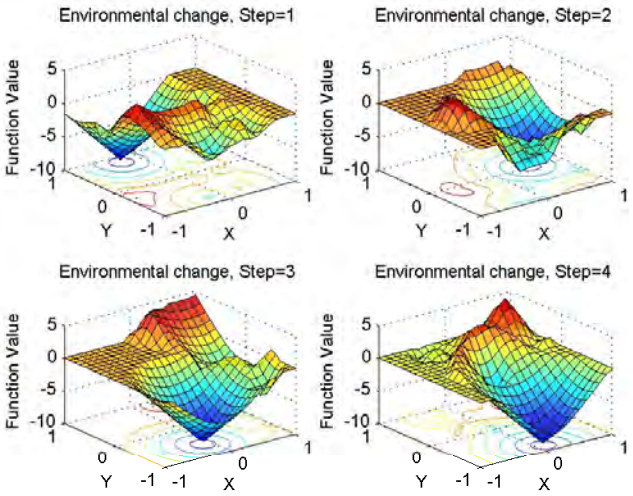


Fig. 2. An example of environmental changes

B. Selection of DBFA parameters

The experiments are designed to 1) evaluate the adaptability of DBFA for various dynamic environments; 2) adjust the algorithm parameters (k and m in equation (4)) of the algorithm to optimize its performance. Various environmental changes are used in our simulation studies, which are divided into three ranges [19]:

- Range I – Slow level of environmental changes
- Range II – Intermediate level of environmental changes
- Range III – High level of environmental changes

The level of changes is reflected by the frequency of changes in the environment, which is defined as a probability τ. For the environmental changes classified in Range I, $\tau \in [0, 0.01]$, in Range II, $\tau \in [0.05, 0.2]$ and in Range III $\tau \in [0.3, 0.8]$. In our simulation studies, τ indicates the probability of occurrence of environmental changes after each chemotactic step. The environmental changes are simulated as changes in the position of X_i and Y_i in equation (5), following the process discussed in Section IV-A.

The whole simulation process including the environmental changes is illustrated in Fig. 3. The "Environmental change database", shown in Fig. 3, stores and updates the parameters listed in Table II, which contains the characteristics of environmental changes for evaluation of the algorithm.

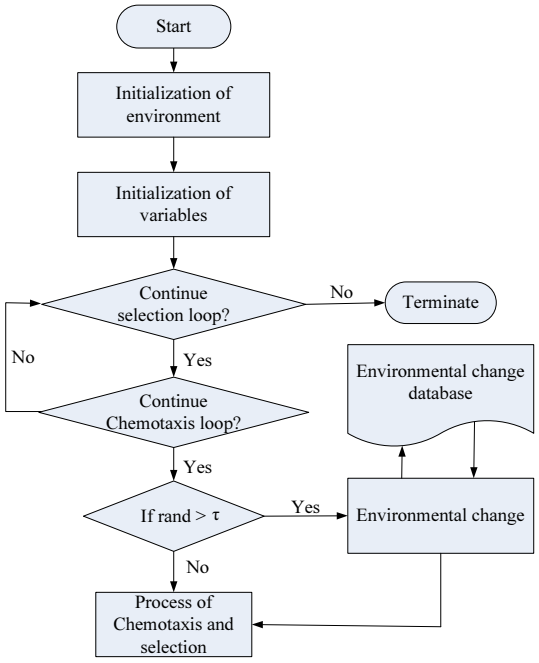


Fig. 3. Flow chart of DBFA

C. Criteria of performance evaluation

In our experiments, we use three different ways to evaluate the performance of DBFA. They are mainly concerned with the average best fitness found over a given period during the evolutionary process, accuracy and stability of the algorithm.

- Average Best over a Period (ABP)
  One of the most important factors in optimization is the ability of finding the global optimum. While in a dynamic environment, only comparing the best solution found after a certain number of generations is not sufficient, since the optimum might be varying over time. There is an alternative for reporting the performance of algorithm, which averages over the best solution found at each step during a period between two environmental changes. It is concerned with an average of the best values, denoted by Average Best, found over a period T_i, where T_i denotes the ith period. This average best over a period is denoted as ABP, which is similar to the "Best Fitness" for static environments, but only for a given period T_i. Let S_i be the first step of T_i, E_i be the last step. Thus, ABP is defined as:

$$ABP_{T_i} = \frac{1}{E_i - S_i} \sum_{t=S_i}^{E_i} f(t)^* \quad (10)$$

  where $f(t)^*$ is the best fitness found in each step, and $t = N_c \times r + j$.

- Accuracy
  To obtain the accuracy of algorithm A in function F, firstly, we calculate accuracy in each step t,

$$Accuracy_{\mathrm{F,A}}(t) = \frac{f_{\mathrm{F,A}}(t)^* - V_{\mathrm{wF,A}}(t)}{V_{\mathrm{oF,A}}(t) - V_{\mathrm{wF,A}}(t)} \quad (11)$$

  Then, the accuracy as a whole is defined as

$$Accuracy_{\mathrm{F,A}} = \frac{1}{N}\sum_{t=1}^{N} Accuracy_{\mathrm{F,A}}(t) \quad (12)$$

  where V_{w} and V_{o} are the worst and optimum value respectively, N is the number of steps.
- Stability
  Similar to the definition of accuracy, to algorithm A in function F, the stability is defined as follows:

$$Stability_{\mathrm{F,A}}(t) = Accuracy_{\mathrm{F,A}}(t) - Accuracy_{\mathrm{F,A}}(t-1) \quad (13)$$

$$Stability_{\mathrm{F,A}} = \frac{1}{N}\sum_{t=1}^{N} \max\{0, Stability_{\mathrm{F,A}}(t)\} \quad (14)$$

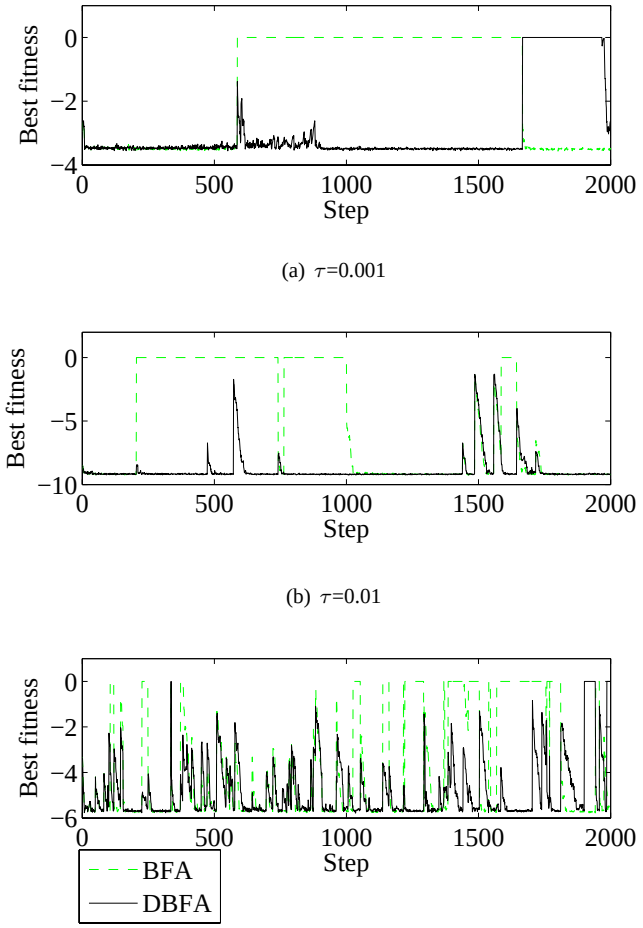
V. Simulation Results and Discussions

A. Simulation results

Both BFA and DBFA are evaluated using MATLAB. Each experiment consists of 10 runs of the algorithm program. The initial parameters, k and m of the algorithm, are set as 0 and 0.5, respectively. The comparison between BFA and DBFA is given in Figs. 4 and 5, where the best fitness and ABP in a single run are plotted. To demonstrate the performance in various environments, in the both figures, the dynamic environment with $\tau = 0.001$, $\tau = 0.01$ and $\tau = 0.05$ are selected, which fall into to Range I and II respectively. As it is shown in Fig. 4, the different parameters cause 3, 13 and 82 times of environmental changes in 2000 steps respectively. In this case, the performances of DBFA, for $\tau = 0.01$ and $\tau = 0.05$, are satisfactory, since the DBFA reacts to all environmental changes effectively. It is also capable to track 2 changes out of 3 when $\tau = 0.001$. In Fig. 5, the ability of searching for the optimum is evaluated by ABP. Compared with BFA, the DBFA is able to track the time-variant global optimum much more smoothly and effectively.

The DBFA has been evaluated in three ranges of dynamic environments, compared with the BFA, with respect to the comparison of accuracy as shown in Table III. $\tau = [0.001, 0.005, 0.01]$ is selected for Range I, $\tau = [0.05, 0.1, 0.2]$ and $\tau = [0.3, 0.5, 0.8]$ for Ranges II and III, respectively. The performance is most satisfactory when $\tau = 0.01$, while it degrades while τ is much smaller or larger. For Range III, it is not surprising that both the BFA and DBFA still obtain a high accuracy, since in this case, the environment changes rapidly and the diversity will play a more important role in contributing to the performance of the algorithm than the local research. While the difference between the accuracies of BFA and DBFA in the static environment is less than 1% when $\tau = 0$, which indicates that DBFA has the same ability of local search.

The stability in Table IV is related to the accuracy, as shown in equation (13). The most desired value of stability is 0, the smaller, the stabler. It is illustrated in the table that the values of the comparisons of two algorithms almost decrease monotonously as the severity of changing environment increases, which means that DBFA becomes stabler than BFA when the environment changes more frequently.



(a) τ=0.001

(b) τ=0.01

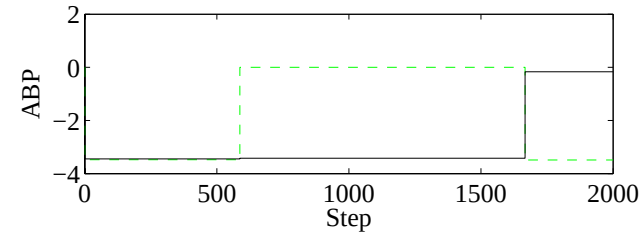
(c) τ=0.05

Fig. 4. Best fitness in the dynamic environment

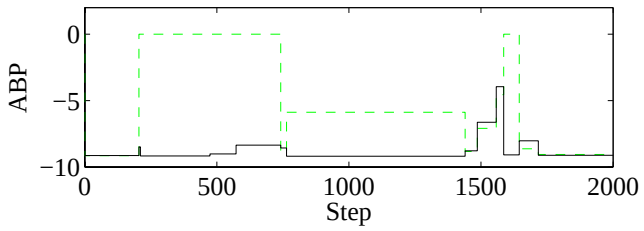
B. Discussion – The selection of parameters

To a large extent, the performance of DBFA depends on the two parameters, m and k, as given in equation (4). In order to get a better performance of DBFA, we evaluated the DBFA with $k = \{0.2, 0.5, 1\}$ and $m = \{0.2, 0.4, 0.7\}$ instead of $k = 0$ and $m = 0.5$ as the initial values. The DBFA was also tested with a set of environmental parameters with $\tau = \{0.005, 0.1, 0.5\}$ respectively. We compare the average accuracies of BFA and DBFA by 10 runs. The results are shown in Table V.

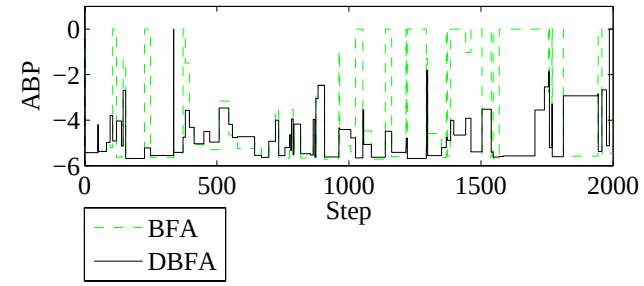
The DBFA with $m = 0.7$ performs worse than that when $m = 0.2$ and $m = 0.4$. It is also worse than that of BFA in 4



(a) τ=0.001



(b) τ=0.01



(c) τ=0.05

Fig. 5. ABP in the dynamic environment

cases out of 9 as listed in Table V, and it is slightly better in the other 2 cases. However, in general, it can be seen that for the 2 cases where $m = 0.2$, $\tau = 0.005$ and $m = 0.4$, $\tau = 0.1$, the DBFA offers the best performance. Table V indicates that when $k = 0.2$, the DBFA has the stablest performance. In this situation, the accuracy of DBFA is better than that of BFA in all cases except the ones with $m = 0.7$, where the convergence has been disturbed by a high diversity.

VI. Conclusion

A new DBFA for optimization in dynamic environments, based on bacterial foraging behaviours, has been presented in this paper. The existing BFA employs the basic foraging activities to mimicry "chemotaxis" while uses an artificial reproduction process to speed the convergence process. Due to the lack of diversity, it is not suitable for dynamic environments. The DBFA adopts a selection scheme which enables the bacteria to flexibly adapt to the changing environment.

We have used the dynamic environment generated by DF1 to evaluate the DBFA, and compared the DBFA with BFA in three aspects: the average best over a period, algorithm accuracy and stability. The simulation results show that in

TABLE III

Comparison of accuracy

Change severity	BFA(%)	DBFA(%)	Comparison(%)
0	98.75	98.59	-0.16
0.001	41.96	59.31	41.35
0.005	45.60	61.59	35.07
0.01	51.72	89.25	72.56
0.05	45.06	58.21	29.18
0.1	33.03	38.12	15.41
0.2	67.02	81.90	22.20
0.3	33.34	53.82	38.05
0.5	46.92	56.22	19.82
0.8	33.13	43.07	30.03

TABLE IV

Comparison of Stability

Change severity	BFA	DBFA	Comparison(%)
0	0.0017	0.0027	58.82
0.001	0.0014	0.0023	64.29
0.005	0.0033	0.0043	30.30
0.01	0.0033	0.0036	9.09
0.05	0.0107	0.0106	-0.93
0.1	0.0288	0.0259	-10.07
0.2	0.0475	0.0417	-12.21
0.3	0.0204	0.0198	-2.94
0.5	0.0917	0.0672	-26.72
0.8	0.1027	0.0683	-33.50

all three ranges of environmental changes, the DBFA is able to provide satisfactory performance, and can react to most of the environmental changes in time. The selection of the DBFA parameters has also been discussed.

It is worth mentioning that the diversity of DBFA changes after each chemotactic process rather than the dispersion adopted by the BFA after several generations. The DBFA utilizes not only the local search but also applies a flexible selection scheme to maintain a suitable diversity during the whole evolutionary process. It outperforms BFA in almost all dynamic environments. Furthermore, the detection of environmental changes is not necessary in the DBFA. The DBFA has the same computational complexity with that of BFA but offers the better performance, although the computation issue is not discussed in this paper.

It should be mentioned that the BFA and DBFA stemmed from a background which is totally different from that of the evolutionary computation techniques such as GA and PSO, etc. The bacteria based algorithms are still in the process of development and they are not mature yet. However, from the understanding of their essential behaviors, we can see the potential of methodologies in this kind, which are demonstrated in our paper and Passino's work [3]. On the other hand, it has been understood that GA and PSO were developed specifically for static optimization problems, although they have also been attempted for dynamic optimization problems. In this paper, we propose a bacteria based algorithm, for the first time, for dynamic optimization problems. DBFA will be compared with the other evolutionary algorithms, as the work proceeds further.

TABLE V

ACCURACY OF DIFFERENT PARAMETER COMBINATIONS(%)

		$m = 0.2$		$m = 0.4$		$m = 0.7$	
τ	k	BFA	DBFA	BFA	DBFA	BFA	DBFA
0.005	0.2	61.98	69.21	45.6	61.59	28.65	21.54
	0.5	58.1	65.4	23.56	11.48	13.46	6.43
	1	42.07	31.95	43.76	25.07	54.63	62.00
0.1	0.2	27.08	39.87	20.51	22.56	8.21	5.63
	0.5	32.32	40.77	37.21	53.25	19.53	18.41
	1	22.12	23.42	63.24	71.94	52.77	59.53
0.5	0.2	64.66	79.07	31.48	36.40	21.54	21.67
	0.5	69.13	76.51	26.74	26.32	42.60	56.44
	1	65.27	73.43	56.65	66.11	30.34	30.97

REFERENCES

[1] H. A. Abbass, K. Sastryy, and D. Goldberg, "Oiling the wheels of change: The role of adaptive automatic problem decomposition in Nonstationary environments", *IlliGAL Report No. 2004029*, 2004.

[2] R. W. Morrison and K. A. De Jong, "A test problem generator for non – stationary environments", *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, IEEE Press, pp. 2047–2053, 1999.

[3] K. M. Passino, "Biomimicry of bacterial foraging", *IEEE Control Systems Magazine*, pp. 52–67, June, 2002.

[4] J. Branke, *Evolutionary optimization in dynamic environments*, Kluwer Academic Publishers, Massachusetts USA, 2002.

[5] S. Yang, X, Yao, "Experimental study on population-based incremental learning algorithms for dynamic optimization problems", *Soft Computing*, Vol. 9, No. 11, pp. 815–834, 2005.

[6] R. Tinós and S. Yang, "Genetic algorithms with self-organized criticality for dynamic optimization problems", *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, IEEE Press, Vol. 3, pp. 2816–2823, 2005.

[7] Y. Jin, J. Branke, "Evolutionary optimization in uncertain environments – A survey", *IEEE Transactions on Evolutionary Computation*, Vol. 9, No. 3, pp. 303–317, 2005.

[8] J. Branke, "Evolutionary approaches to dynamic optimization problems – Updated survey", *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pp. 27–30, 2001.

[9] S. Yang, "Memory-enhanced univariate marginal distribution algorithms for dynamic optimization problems", *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, IEEE Press, Vol. 3, pp. 2560–2567, 2005.

[10] J. Branke, "Memory-enhanced evolutionary algorithms for dynamic optimization problems", *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, IEEE Press, Vol. 3, pp. 1875–1882, 1999.

[11] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems", in *Advances in evolutionary computing: theory and applications*, Springer - Verlag New York, Inc., New York, NY, 2003.

[12] T. Blackwell, J. Branke, "Multi-swarm optimization in dynamic environments", *Applications of Evolutionary Computing, Lecture Notes in Computer Science*, Springer, Vol. 3005 pp. 489–500, 2004.

[13] J. P. Li, M. Balazs, G. Parks, P. Clarkson, "A species conserving genetic algorithm for multimodal function optimization." *Evolutionary Computation*, Vol. 10, No. 3, pp. 207–234, 2002.

[14] N. Mori, H. Kita, and Y. Nishikawa, "daptation to a changing environment by means of the thermodynamical genetic algorithm", *Parallel Problem Solving from Nature, Lecture Notes in Computer Science*, Springer, Vol. 1141, pp. 512–522, 1996.

[15] M. Guntsch, M. Middendorf, and H. Schmeck, "An Ant Colony Optimization approach to Dynamic TSP", *Proceedings of the Genetic and Evolutionary Computation Conference 2001*, Morgan Kaufmann Publishers, pp. 860–867, 2001.

[16] C. Fernandes, V. Ramos, and A. C. Rosa, "Varing the population size of artificial foraging swarms on time varying landscapes", *Lecture Notes in Computer Science*, Vol. 3696, pp. 311–316, 2005.

[17] H. C. Berg, D. A. Brown, "Chemotaxis in *Escherichia coli* analyzed by three-dimensional tracking", *Nature*, Vol. 239, pp. 500–504, 1972.

[18] S. Mishra, "A hybrid Least Square - Fuzzy Bacterial Foraging Strategy for harmonic estimation", *IEEE Transaction on Evolutionary Computation*, Vol. 9, No. 1, pp. 61–73, 2005.

[19] R. Walker, " 'Niche Selection' and the evolution of complex behavior in a changing environment – a simulation", *Artificial Life*, Vol. 5, pp. 271–289, 1999.