Triggered Memory-Based Swarm Optimization in Dynamic Environments

Hongfeng Wang[1], Dingwei Wang[1], and Shengxiang Yang[2]

[1] School of Information Science and Engineering, Northeastern University
Shenyang 110004, P.R. China
{hfwang,dwwang}@mail.neu.edu.cn
[2] Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, United Kingdom
s.yang@mcs.le.ac.uk

Abstract. In recent years, there has been an increasing concern from the evolutionary computation community on dynamic optimization problems since many real-world optimization problems are time-varying. In this paper, a triggered memory scheme is introduced into the particle swarm optimization to deal with dynamic environments. The triggered memory scheme enhances traditional memory scheme with a triggered memory generator. Experimental study over a benchmark dynamic problem shows that the triggered memory-based particle swarm optimization algorithm has stronger robustness and adaptability than traditional particle swarm optimization algorithms, both with and without traditional memory scheme, for dynamic optimization problems.

1 Introduction

In recent years, there has been an increasing concern from the evolutionary computation community on problem optimization in dynamic environments since many real-world problems are dynamic optimization problems (DOPs), where stochastic changes may occur regarding the optimization goal, the problem instance, or some restrictions. For DOPs, the goal of evolutionary algorithms (EAs) is no longer to find a satisfactory solution, but to track the trajectory of the moving optimum in the search space. This poses a great challenge to traditional EAs. To address this challenge, several approaches have been developed into EAs to improve their performance in dynamic environments [2,3,9,10,13,16]. A comprehensive survey can be found in [4].

The genetic algorithm (GA) was the first evolutionary computation approach used to explore DOPs. Recently, particle swarm optimization (PSO), as another evolutionary computation technique, has been applied to DOPs. In this paper, a triggered memory-based approach is introduced into the PSO to improve its performance in dynamic environments. The triggered memory scheme enhances traditional memory scheme with a triggered memory generator. Experimental study over a benchmark dynamic problem shows that the triggered memory scheme efficiently improves the performance of PSOs in dynamic environments.

M. Giacobini et al. (Eds.): EvoWorkshops 2007, LNCS 4448, pp. 637–646, 2007.

The rest of this paper is outlined as follows. The next section briefly describes the PSO and surveys the literature on PSOs for DOPs. Section 3 proposes a variety of triggered memory-based methods for PSOs to handle dynamic environments. Then, the experimental settings and results are reported in Section 4 and Section 5 respectively. Finally, Section 6 concludes this paper with some suggestions for future work.

2 Particle Swarm Optimization in Dynamic Environments

Particle swarm optimization was first introduced by Kennedy and Eberhart [7,12]. PSO simulates the social behaviour among particles that "fly" through a solution space. Each particle accomplishes its own updating based on its current velocity and position, the best position seen so far by the particle, and the best position seen so far by the population (or by the local neighbourhood in the local version of PSOs. In this paper, only the global version is discussed). The behaviour of a particle can be described as follows:

$$v_i(t+1) = \omega v_i(t) + c_1\xi(p_i(t) - x_i(t)) + c_2\eta(p_g(t) - x_i(t)) \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

where $v_i(t)$ and $x_i(t)$ represent the current velocity and position of particle i at time t respectively, $p_i(t)$ is the position of the best solution discovered so far by particle i, $p_g(t)$ is the position of the best solution found so far by all particles, ω is the inertia weight that controls the degree a particle's previous velocity will be kept, c_1 and c_2 are individual and social learning factors, and ξ and η are random numbers in the range [0, 1].

As a kind of robust optimization technique, PSO has been widely used for stationary optimization problems where the fitness landscape does not change during the course of the computation. Recently, the application area of PSOs has been extended to time-varying systems. When applied for DOPs, traditional PSOs face a big problem, that is, the whole population will eventually converge to a small area, from which it is very difficult for PSOs to jump out to follow the changes. This is a challenge to traditional PSOs for DOPs. Recently, researchers have developed a number of PSO approaches for DOPs, which are briefly reviewed below.

Eberhart and Shi [8] put forward the first work, where the PSO was investigated to track a single peak that varies spatially. Based on a parabolic function $f(\cdot) = x^2 + y^2 + z^2$, they observed that the tracking errors achieved by the standard PSO are several order of magnitude less than those achieved by comparable GA-based approaches.

Hu and Eberhart [11] introduced an adaptive PSO, which automatically tracks various changes in a dynamic system. They tested different environmental detection and re-randomization strategies, which effectively respond to a wide variety of changes, and reported and analyzed the experimental results on the parabolic function and Rosenbrock's benchmark function with various severities of environmental changes.

Parott and Li [14] investigated a PSO model for tracking multiple peaks in a continuously changing dynamic environment. Multiple parallel subpopulations were constructed by a form of speciation and encouraged to simultaneously tracking multiple peaks by preventing overcrowding at peaks in this model. The experiments in dynamic multimodal environments indicated that the technique was capable of tracking multiple changing peaks simultaneously.

A method of adapting PSO for dynamic environments was present by Carlisle and Dozier [6]. In their PSO, each particle can reset the record of its best position and avoid making direction and velocity decisions based on outdated information as the environment changed. Two resetting methods were examined and experimental results show that both were able to improve the performance of PSO in both static and dynamic environments.

Blackwell and Branke [1] proposed several new variants of PSOs specifically designed for non-stationary environments, where the single population PSO and charged PSO (CPSO) were extended by constructing interacting multi-swarms. In addition, a new multi-quantum swarm optimizer, which broadens the implicit atomic analogy of CPSO to a quantum model, was also introduced. Their experimental study on the Moving Peaks Benchmark problem indicates that the multi-swarm optimizers significantly outperform single population PSOs.

3 The Triggered Memory-Based PSO for DOPs

Among the approaches developed for EAs in dynamic environments, the memory scheme is a major approach that has proved beneficial for many DOPs [15]. In memory enhanced EAs, good individuals from the population can be stored into a memory at regular interval during the course of evolution and can be retrieved once a change occurs in the environment. Intuitively, when an optimum reappears in a previous location or nearby, the memory can remember that location and guide the population to move to that optimum. Memory can also help maintain the population diversity and adapt to environmental changes quickly because useful past information has been saved and can be reused. In this section, we discuss how a triggered memory mechanism can be applied to the PSO in order to make it suitable for dynamic problems.

A disadvantage of the memory schemes is that memory might mislead evolution and prevent the population from exploring new peaks in the search space, though it might be propitious to the exploitation of knowledge gained in the past. Intuitively, restarting evolution from scratch once a change in the environment has occurred will have a chance to find new peaks. However, it may be too time-consuming to reach the new optima. In [2], a tri-island model has been proposed for the memory-enhanced GA in dynamic environments and proved an efficient way to maintain the tradeoff between exploration and exploitation.

The idea of the tri-island memory model can be combined into the PSO for DOPs. For the memory-enhanced PSO with the tri-island model, the whole population is also divided into three parts: “explore”-population, memory and “exploit”-population, which are respectively used to explore the search space,

store good solutions, and exploit the memory in this paper. The memory is also used to detect the environmental changes. A change is detected to have occurred whenever the fitness of at least one solution in the memory has changed. Once an environmental change is detected, all individuals will be re-evaluated. In order to enforce exploration, the "explore"-population needs to be randomly re-initialized frequently. Thus, the re-initialization period, that is, when the "explore"-population should be re-initialized, becomes an important parameter. A simple scheme is to re-initialize the "explore"-population after every environmental change. However, there may be some problems for this scheme. For example, if the environment changes slowly, the population might always stay in a peak and might not jump out to search for other peaks for a long time.

In order to solve the above problem, we introduce a new triggered generator for the memory-based PSO, where the re-initialization of the "explore"-population will be immediately initiated once a peak has been found. Thus, the triggered generator can be more efficient in exploring the search space than the simple re-initialization method, especially when the environment does not change frequently. The next question is how to judge that a peak has been found by the "explore"-population. Here, we deem that a peak has been found if some restrictions on its performance are fulfilled. For the re-initialization conditions, we consider the following two alternatives:

First, compute the running average of the fitness of the best individuals over a period of five generations. If the increase degree of the running average of the best-of-generation fitness is less than a threshold $b1$, the re-initialization is started (this method will subsequently be termed *averfit*). The increase of the running average of the best-of-generation fitness is calculated as follows:

$$e(t) = \frac{\frac{1}{5}\sum_{i=0}^{4} f_b(t-i) - \frac{1}{5}\sum_{i=0}^{4} f_b(t-i-1)}{\frac{1}{5}\sum_{i=0}^{4} f_b(t-i)} = \frac{\sum_{i=0}^{4} f_b(t-i) - \sum_{i=0}^{4} f_b(t-i-1)}{\sum_{i=0}^{4} f_b(t-i)} \quad (3)$$

where $f_b(t)$ denotes the fitness of the best solution achieved in generation t.

Second, compute the Euclidean distance between the best individual and the worst individual in the "explore"-population. If the value is below a threshold $b2$, the population will be re-initialized. This method will subsequently be termed *maxdist* in this paper.

With respect to which individuals should be stored in the memory, we only consider the best individuals achieved by the "explore"-population. And since the memory space is fixed and limited, we need to consider which solutions in the memory should be replaced to make space for new ones. In this paper, in order to maintain the diversity of the memory, we apply the replacement strategy where the less fit solution of the two solutions that are closest to each other in the memory is removed from the memory.

For further questions as to when and how to use the memory, we propose two memory retrieval schemes: called *memory-based resetting* and *memory-based immigrants*. Both schemes happen at the same time when the "explore"-population

is re-initialized. The memory-based resetting scheme is to reset the record of the best global position using the memory, where all the solutions in the memory will be re-evaluated and the best one will be chosen as the new best global solution in the "exploit"-population if it is better than the old one. In this scheme, all particles together share a memory base that stores the best peaks achieved by the "explore"-population in the past. Therefore, the particles can adjust the direction of flying according to the memorial information. For the memory-based immigrants retrieval scheme, all the solutions in the memory are re-evaluated and injected into the "exploit"-population to replace the same amount of worst individuals in the "exploit"-population. Compared to the memory-based resetting scheme, this scheme seems more efficient since the solutions in the memory are explicitly immigrated to the population.

4 Experimental Settings

For the experiments, the Moving Peaks Benchmark by Branke [5] is used as the dynamic test problem. The base landscape of the moving peaks function consists of m peaks defined in the n-dimensional real space as follows:

$$F(\boldsymbol{x}, t) = \max_{i=1,\ldots,m} \frac{H_i(t)}{1 + W_i(t) \sum_{j=1}^{n} (x_j(t) - X_{ij}(t))^2} \tag{4}$$

where $W_i(t)$ and $H_i(t)$ are the height and width of peak i at time t respectively, and $X_{ij}(t)$ is the j-th element of the location of peak i at time t. Each peak can independently change its height and width and move its location around in the search space.

The parameter settings of the Moving Peaks Benchmark used in this paper correspond to Scenario 1 as specified on the benchmark website [5]. The test function has 5 peaks defined on a 5-dimensional real space. Every Δe generations, the height and width of each peak are changed by adding a random Gaussian variable and the location of each peak is moved by a shift vector $\boldsymbol{v_i}$ of fixed length s. More formally, a change of a single peak can be described as follows:

$$\begin{cases} \sigma \in N(0,1) \\ H_i(t) = H_i(t-1) + 7 \cdot \sigma \\ W_i(t) = W_i(t-1) + 0.01 \cdot \sigma \\ \boldsymbol{X}_i(t) = \boldsymbol{X}_i(t-1) + \boldsymbol{v}_i(t), \end{cases} \tag{5}$$

$$\boldsymbol{v}_i(t) = \frac{s}{|\boldsymbol{r} + \boldsymbol{v}_i(t-1)|}((1-\lambda)\boldsymbol{r} + \lambda \boldsymbol{v}_i(t-1)) \tag{6}$$

where the shift vector $\boldsymbol{v}_i(t)$ is a linear combination of a random vector $\boldsymbol{r}$ and the previous shift vector $\boldsymbol{v}_i(t-1)$ and is normalized to length s. The random vector $\boldsymbol{r}$ is created by drawing random numbers for each dimension and normalizing its length to s. Hence, the parameter s allows controlling the severity of changes and Δe determines the frequency of changes. The parameter λ allows controlling whether changes exhibit a trend (λ is always set to 0.5 in our experiments).

The experiments are designed to investigate the performance of different memory-based algorithms: a simple PSO model (SPSO) where each change is regarded as the arrival of a new optimization problem and is solved from scratch, a traditional memory-based PSO model (SMPSO) that is adapted from Branke's tri-island memory model for GAs [2], the triggered memory-based PSO models both with the resetting scheme (TMRPSO) and the immigrants scheme (TMIPSO). For all PSO models, the learning factors c_1 and c_2 are set to 2.0 and the inertia weight ω is initialized to 0.5, decreases linearly over the first 100 generations till 0.2, and then remains at 0.2 till the end of a run. The total number of particles is 50: the size of both the "explore"-population and "exploit"-population is set to 20. Unless stated otherwise, the size of memory is always 10.

To measure the performance for the algorithms, an offline performance function e^*, which is the average fitness error between the optimal fitness of the current environment and the best-of-generation fitness at each generation, is reported here since for DOPs a single, time-invariant optimal solution does not exist, the goal is not to find the optima but to track their progression through the space as closely as possible. The average fitness error at time t can be calculated as follows:

$$e^*(t) = \frac{1}{t} \sum_{i=1}^{t} |f_b^* - f_b(t)| \tag{7}$$

where f_b^* is the fitness of the optimum and $f_b(t)$ is the fitness of the best solution achieved at generation t. If several populations are used, $f_b(t)$ is the best solution over all populations at generation t.

5 Experimental Results

The preliminary experiments were first carried out on TMIPSO under the two triggered generators with different settings, where s is set to 1.0 and Δe is set to 100. The maximum generation is set to 1000, which equals 10 environmental changes. Each experimental result is averaged over 100 runs with the same set of different random seeds. The experimental results are shown in Fig. 1.

From Fig. 1, it can be seen that the approach of restricting the running average fitness (*averfit*, see the lower 3 curves) performs significantly better than the approach of restricting the maximum distance (*maxdist*, see the upper 3 curves) in the competition between the two different triggered generators. And the effectiveness of varying the threshold $b2$ settings in the *maxdist* scheme seems to be very small since the performance curves always wind together. That is, varying $b2$ does not affect the performance of algorithms much. However, the situation is different for the *averfit* scheme. It can be seen that the performance curves almost superpose together when $b1$ is small (i.e., 0.001 or 0.01), but the performance declines clearly when $b1$ becomes too large (i.e., 0.1). Therefore, $b1$ should not be set too large. For our following experiments, the *averfit* triggered generator will be used and the threshold $b1$ is always set to 0.001.

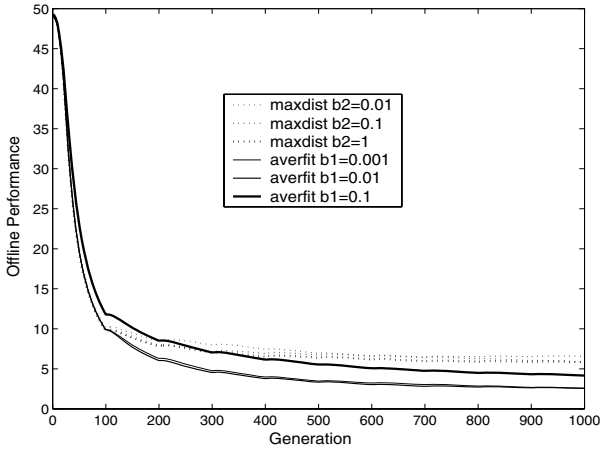
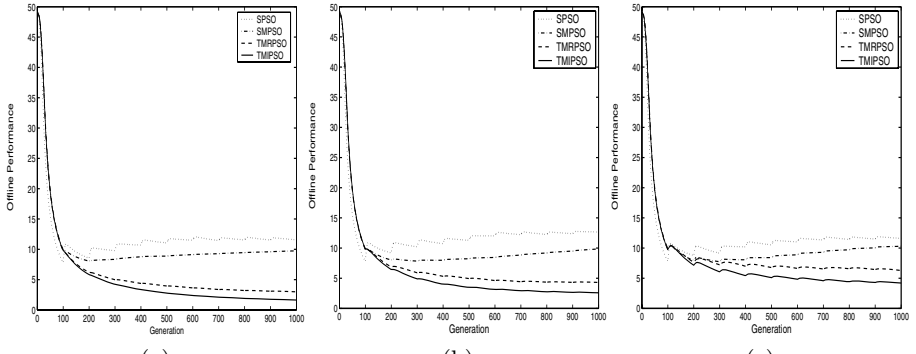


Fig. 1. Experimental results on the triggered memory-based immigrants PSO with different triggered generators in the dynamic environments



(a) (b) (c)

Fig. 2. Experimental results on the triggered memory-based immigrants PSO with different triggered generators in the dynamic environments with $\Delta e = 100$ and different severities of changes: (a) $s = 0.5$, (b) $s = 1.0$, and (c) $s = 2.0$

Fig. 2 plots the results of PSOs on the dynamic problems with different severities and Δe=100. From Fig. 2, several results can be observed.

First, SPSO slightly outperforms the memory-enhanced PSOs just for the stationary period (i.e., the first environment), but the memory-based PSOs always perform much better than SPSO for the dynamic periods. In the stationary period, all PSO models randomly search for the optimum in the solution space because the memory is empty and scanty. Hence, SPSO can more easily find a peak in the original fitness landscape since it has a single population whose size is much larger than each one of the populations in the memory-based PSOs.

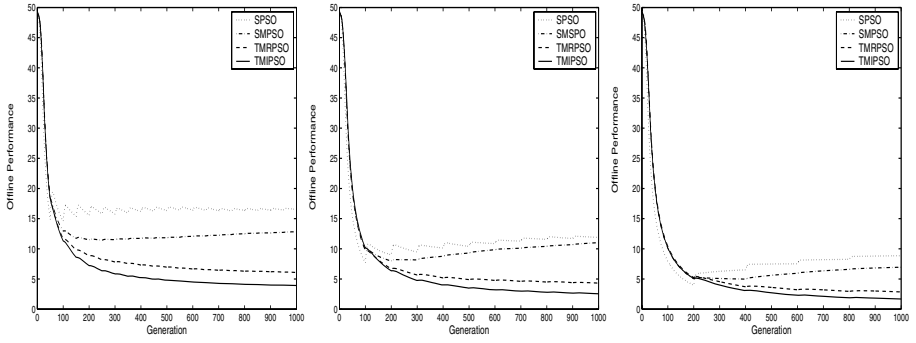


Fig. 3. Experimental results on four different PSO models in the dynamic environments with $s = 1.0$ and different frequencies of changes: (a) $\Delta e = 50$, (b) $\Delta e = 100$, and (c) $\Delta e = 200$

But in the dynamic periods, memory can help the population remember the past information and restart from the promising area closer to the new optimum. But SPSO will restart evolution from scratch here, which makes the population take a long time to reach an optimum (even a local optimum in most cases). This is also the reason why the performance curve of SPSO seems a little oscillatory.

Second, the triggered memory-based PSOs perform better than traditional memory-based PSO. In the triggered memory methods, the "explore"-population could contribute its best solution to the memory once a peak is affirmed to be found and the "exploit"-population is also injected with the new memory information at once. However, in the traditional memory method, the "explore"-population will contribute its solution and be re-initialized until a change is detected, which means that it could have stayed in a peak, if the peak is found, for a long time and lost the chance of finding a higher peak. Hence, the triggered memory methods can explore the solution space more efficiently and more quickly than the traditional memory method.

Third, the triggered memory-based resetting scheme performs worse than the triggered memory-based immigrants scheme over all the periods except at the beginning of evolution. In the memory-based resetting scheme, the information in the memory is reintroduced into the "exploit"-population just as an alternative of the global solution. This seems to just contribute an attractor to the population. However, whether the population could reach the neighbourhood of the attractor and keep the correct evolution in direction is not clear. Compared to the resetting scheme, the immigrants scheme employs a more efficient way where all the solutions in the memory are explicitly replaced into the "exploit"-population. On the other hand, the injection of more memorial information also helps the population maintain a high diversity in the immigrants scheme than in the resetting scheme.

Fourth, the change severity, which is one aspect of the environmental dynamism, affects the performance of all PSOs. And it seems natural that for

a fixed value of Δe, the performance of PSOs decreases when the value of s increases.

The experimental results on PSOs in the dynamic problems with different frequencies of environmental changes and $s = 1.0$ are plotted in Fig. 3. Similar results can be observed from Fig. 3 as from Fig. 2. The frequency of changes is another aspect of the environmental dynamism, and also naturally, when the frequency of change decreases, i.e., when Δe increases, the performance of all PSOs increases.

6 Conclusions

This paper investigates the application of PSOs with the tri-island memory model for DOPs. For this memory-based PSO model, a traditional way is to re-initialize the "explore"-population and retrieve the memory whenever an environmental change is detected. However, there may be some problems for this scheme. When the environment changes slowly, the population might have always stayed in a peak instead of searching for other peaks for a long time. In order to solve this problem, a new triggered memory scheme is proposed for the memory-based PSO in dynamic environments, where a triggered memory generator is designed for the retrieval period of memory. In this scheme, whenever the "explore"-population finds a peak, it will be immediately re-initialized and the memory will be retrieved. In order to determine that a peak has been found by the "explore"-population, two measures are also proposed. To retrieve the memory this paper proposes two strategies: the memory-based immigrants scheme replaces the solutions in the memory explicitly into the "exploit"-population and the memory-based resetting scheme only resets the record of the best global solution for the "exploit"-population using the best re-evaluated solution in the memory.

Based on the Moving Peaks Benchmark function [5], experiments were carried out to compare the performance of several PSOs including the proposed triggered memory-based PSOs in dynamic environments. From the experimental results, we can draw the following conclusions on the dynamic test problems. First, the memory mechanism can improve the performance of PSOs in dynamic environments. Second, the triggered memory method is more efficient than the traditional memory method in exploring the solution space. Hence, the triggered memory-based PSOs have stronger robustness and adaptability than the traditional memory-based PSO and simple PSO in dynamic environments, especially when the environment does not change frequently. Third, the memory-based immigrants scheme is more efficient than the memory-based resetting scheme for enhancing the performance of the triggered memory-based PSO in dynamic environments.

For future work, it is valuable to examine the performance of hybrid approaches that combine the triggered memory method and other approaches already known from the literature, e.g., the random immigrants scheme, for PSOs

in dynamic environments. In addition, it is also interesting to construct new triggered generators and examine them under the same framework.

References

1. T. Blackwell and J. Branke. Multi-swarm optimization in dynamic environments. In *Applications of Evolutionary Computing*, LNCS vol. 3005, pp. 489-500, 2004.
2. J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In *Proc. of the 1999 IEEE Congress on Evolutionary Computation*, Washington, DC, USA, vol. 3, pp. 1875-1882, 1999, IEEE Press.
3. J. Branke, T. Kaußler, C. Schmidth, and H. Schmeck. A multi-population approach to dynamic optimization problems. *Proc. of the 5th Int. Conf. on Adaptive Computing in Design and Manufacturing*, pp. 299–308, 2000.
4. J. Branke. *Envolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2002.
5. J. Branke. The moving peaks benchmark website. Online, http://www.aifb.uni-karlsruhe.de/jbr/MovPeaks.
6. A. Carlisle and G. Dozier. Adapting particle swarm optimization to dynamic environments. In *Proc. of the 2000 Int. Conf. on Artificial Intelligence*, Las Vegas, USA, pp. 429-434, 2000.
7. R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, Nagoya, Japan, pp. 39-43, 1995, IEEE Press.
8. R. Eberhart and Y. Shi. Tracking and optimizing dynamic systems with particle swarms. In *Proc. of the 2001 IEEE Congress on Evolutionary Computation*, Seoul, Korea, vol. 1, pp. 94-100, 2001, IEEE Press.
9. D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In *Proc. of the 2nd Int. Conf. On Genetic Algorithms*, pp: 59-68, 1987.
10. J. J. Grefenstette. Genetic algorithms for changing environments. In R. Maenner and B. Manderick, editors, *Parallel Problem Solving From Nature 2*, pp. 137-144, 1992.
11. X. Hu and R. Eberhart. Adaptive particle swarm optimization: Detection and response to dynamic systems. In *Proc. of the 2002 IEEE Congress on Evolutionary Computation*, Hawaii, USA, pp. 1666-1670, 2002. IEEE Press.
12. J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proc. of the 1995 IEEE Int. Conf. on Neural Networks*, Perth, Australia, vol. 5, pp. 1942-1948, 1995, IEEE Press.
13. R. W. Morrison and K. A. De Jong. Triggered hypermutation revisited. In *Proc. of the 2000 IEEE Congress on Evolutionary Computation*, San Diego, USA, pp. 1025-1032, 2000, IEEE Press.
14. D. Parrott and X. Li. A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. In *Proc. of the 2004 IEEE Congress on Evolutionary Computation*, Portland, USA, pp. 98-103, 2004, IEEE Press.
15. S. Yang. Memory-based immigrants for genetic algorithms in dynamic environments. *Proc. of the 2005 Genetic and Evolutionary Computation Conference*, vol. 2, pp. 1115–1122, 2005.
16. S. Yang and X. Yao. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing*, **9**(11): 815-834, 2005.