ICIC Express Letters ICIC International ©2016 ISSN 1881-803X
Volume **10**, Number **11**, November **2016** pp. **2631–2638**

SELF-ADAPTIVE NEIGHBORHOOD SEARCH PARTICLE SWARM OPTIMIZATION IN DYNAMIC ENVIRONMENTS

Dingcai Shen[1,2] and Shengzhou Hu[1]

[1]Key Laboratory of Jiangxi Province for Numerical, Simulation and Emulation Techniques
Gannan Normal University
Economic & Technological Development Zone, Ganzhou 341000, P. R. China
dcshensa@gmail.com

[2]School of Computer and Information Science
Hubei Engineering University
No. 272, Traffic Avenue, Xiaogan 432000, P. R. China

Received March 2016; accepted June 2016

Abstract. *Many real-world optimization problems are dynamic. The ability to track a changing optimum over time is concerned in these problems. In this study, a new variant of Particle Swarm Optimization (PSO) combined with self-adaptive neighborhood search strategy is proposed. The self-Adaptive Neighborhood Search PSO (ANSPSO) divides the individuals of the population into three types of roles, and the individual with different roles adopts a different update strategy during the evolution, and the search radius adaptively changes within each environmental change cycle. A comparative study with several algorithms with different characteristics on a common platform by using the moving peaks benchmark and various problem settings is presented in this study. The results indicate that the proposed algorithm can track the changing optimum in each circumstance effectively on the selected benchmark function.*

Keywords: Neighborhood search, Particle swarm optimization, Dynamic optimization, Moving peaks benchmark

1. Introduction.

Evolutionary Algorithms (EAs) have been widely employed to deal with stationary optimization problems. However, in real-world applications, there exists another class problem which changes conditions over time: the newly added artifacts in production scheduling, machines wear out or unexpected maintenance result in the decline in production capacity, urban traffic problems at different times, market factors uncertain changes in financial trading models, etc. These changes include the fitness functions, search space, and/or constraints. The dynamic behavior of Dynamic Optimization Problems (DOPs) constitutes new challenges to the evolutionary algorithms. The main challenge of the canonical techniques in dynamic environments is the loss of diversity which arises due to the convergency of the evolutionary population after some iteration steps; however, in dynamic environments, finding the optimum solution(s) in the problem space is no longer the only goal of the optimization algorithm, but rather to continuously track the changing optimum during the evolutionary process [1, 2].

The dynamic property of problems poses major challenges to canonical EAs [3]. Some additional strategies are needed to be introduced into the dynamic optimization algorithms. During the last two decades, there has been a growing interest in DOPs, and many works have been achieved in EAs community in this field. A simple way to solve DOPs is to restart the algorithm once the detection of environmental changes. However, the primitive restart strategy is often impractical in realistic application [4]. In order to track the optimum effectively in a dynamic environment, a key factor is to maintain the diversity of the evolutionary population. However, the diversity of the population is difficult to guarantee after the evolutionary operator's operation. Therefore, the main goal of dynamic optimization algorithms is to adopt some strategies to increase or maintain the diversity of the evolutionary population. The hyper-mutation strategy [5] and the *random immigration* strategy [6] are two straight-forward approaches to increase the diversity. Other schemes include memory-based methods [7, 8] and multi-population approaches [9, 10].

In the current study, on account of the nature of easily losing population diversity of conventional EAs, a self-adaptive neighborhood search particle swarm optimization (ANSPSO), combining with multiple roles in the process of the evolution, is proposed. The individuals in the population are assigned different roles according to its fitness value, and different update strategies are adopted in the updating process, so as to better maintain the diversity of the population, providing a guarantee for the algorithm to track the changing trajectory of the optima. The ability of the algorithm to track the changing trajectory of the optima was verified by the numerical experiments on moving peak benchmark under different environmental cycle and different environmental change intensity.

The arrangement of this paper is as follows. In Section 2, the basic PSO algorithm is briefly introduced. The newly proposed approach is described in detail in Section 3. In Sections 4 and 5, ANSPSO experimental results are compared with those of existing dynamic optimization algorithms and analysis is given. Finally, Section 6 concludes the paper and outlines future work.

2. Background and Related Work.

Particle Swarm Optimization (PSO) [11], which is a population-based stochastic optimization algorithm for continuous optimization, was first developed by Eberhart and Kennedy in 1995. The algorithm is inspired by the social interaction behavior of birds flocking and fish schooling, and it can be easily implemented and has been proven to be both effective and fast when applied to many global optimization functions [12].

Similar to other population-based EAs, PSO is initialized with a population of random particles which distributed uniformly in the search space. Each particle (individual) of the population "flies" through the d-dimensional search space, adjusting its position according to its own experience and that of neighboring particles. Therefore, in the original PSO, all particles have a tendency following the previous best position visited by itself and its neighbor particles.

Assuming in a population with d-dimensional solution space of N individuals $(\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_N)$, the position and the velocity of the ith particle of the swarm can be represented by a d-dimensional vector, $\boldsymbol{X}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})$ and $\boldsymbol{V}_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,d})$, respectively. The best previously visited position (*pbest*) of the ith particle is denoted as $\boldsymbol{P}_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,d})$, and the best position of the population (*gbest*) is represented by $\boldsymbol{P}_g = (p_{g,1}, p_{g,2}, \ldots, p_{g,d})$. For generation t of PSO algorithm, $\boldsymbol{V}_i$ and $\boldsymbol{X}_i$ are updated as follows:

$$\boldsymbol{V}_i^{t+1} = \boldsymbol{V}_i^t + \varphi_1 \left(\boldsymbol{P}_i^t - \boldsymbol{X}_i^t\right) + \varphi_2 \left(\boldsymbol{P}_g^t - \boldsymbol{X}_i^t\right) \tag{1}$$

$$\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t + \boldsymbol{V}_i^{t+1} \tag{2}$$

where:

$$\varphi_1 = c_1 r_1,\ \varphi_2 = c_2 r_2,$$

and c_1 and c_2 are the positive acceleration constants used to scale the contribution of cognitive and social components, typically set to 2.05. r_1, r_2 are uniformly distributed random number in the interval $[0.0, 1.0]$.

According to the definition of the neighborhood of each particle, there are two main kinds of neighborhood models for PSO [13]. The first one is called *lbest* model, where each particle is mutated using the best position found so far in its several fixed neighbors and not in the entire population. On the other hand, the second one, referred to as the *gbest*, the neighborhood of a particle is selected in the entire population for the current generation. Some researchers pointed out [14, 15] that the *gbest* model converged faster and would be more likely to get stuck in local optima than that of *lbest*. On the other hand, the *lbest* model is converged slower with less vulnerability to the attraction of local optima than that of the *gbest* model.

The PSO algorithm performs repeated applications of the update process until a specified number of iterations has been reached, or the number of maximum evaluations has been exceeded.

3. **Proposed ANSPSO Algorithm.** In a dynamic environment, maintaining population diversity is a key factor for the algorithm to track environmental change. Various strategies have been introduced to increase the population diversity during the evolution. Inspired by [16] and Artificial Bee Colony (ABC) algorithm [17] that categorize the population into three groups and each group using different update strategy during the evolution, in ANSPSO, the individuals are also divided into three categories: *Leader*, *Follower*, *Scouter*. After the initialization process, the fitness of individuals in population $\boldsymbol{P}$ are calculated and sort the values in descending order. The formerly ranked P_L ($P_L \geqslant 1$) individuals are chosen as *Leader*. The next P_F ($P_F \geqslant 1$) individuals followed on the *Leader* individuals are designated as *Follower*, and the remainder individuals are assigned the role as *Scouter*.

The *Leader* individual(s) explores better solutions around the neighborhood of the randomly selected one among them, the update of *Leader* using the following expression.

$$\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t + \phi_i\left(\boldsymbol{X}_L^t - \boldsymbol{X}_i^t\right) \tag{3}$$

where $\boldsymbol{X}_L^t$ is random selected *Leader* different from i, ϕ_i is random number drawn from $[-1, 1]$ which is used to randomly weight the influence of the old solution to become a new solution in the next iteration.

The *Follower* individual searches better solutions around the *Leader* within a certain range, in order to gain a better exploration ability, and we change the search range adaptively using expression as follows:

$$\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t + \phi_i R_i\left(\boldsymbol{X}_L^t - \boldsymbol{X}_r^t\right) \tag{4}$$

where $\boldsymbol{X}_L^t$ is random selected *Leader* individual, ϕ_i is as Equation (3). The index r is a random integer drawn from the *Follower* indices and is different from i. R_i is adaptively adjusted using the following equation:

$$R_i = \frac{|f_L - f_i|}{f_L(t - k\tau)} \tag{5}$$

where f_i is the fitness of the current considered individual, f_L is the fitness value of $\boldsymbol{X}_L^t$ in Equation (4), t is the current iteration, τ is the environmental change cycle and $k = \lceil t/\tau \rceil - 1$. It can be observed in Equation (5), at the beginning of the environmental change cycle, *Follower* individuals perform a wider search around the *Leader*, thus enhancing the exploration ability of the *Follower*. When arriving at the end of the environmental change cycle, most individuals are gathering around the *Leader*, the difference between the parameters of the f_L and f_i decreases, and the perturbation on the $\boldsymbol{X}_i^t$ decreases, too. Thus, as the search process continues, the search scope is adaptively reduced.

The *Scouter* individuals use Equations (1) and (2) to update, where $\boldsymbol{P}_g$ is the fittest individual among the *Scouter* individuals.

The pseudo code of the ANSPSO algorithm is illustrated in Algorithm 1.

Algorithm 1 The pseudo code for ANSPSO

```
1: Setting parameters:
2:       $T_{\max}$ – the maximum iterations for the ANSPSO;
3:       $N$ – the population size;
4:       $R_L$,$R_F$ – the rates for Leader and Follower;
5: Initialize: Randomly generate an initial population $\boldsymbol{P(0)}$ with size $N$.
6: while $t < T_{\max}$ do
7:    Evaluate $\boldsymbol{P(t)}$;
8:    Sort $\boldsymbol{P(t)}$ in descending order;
9:    Select $R_L * N$ individuals in the formerly ranked individuals as Leader, the next
      $R_F * N$ as Follower individuals, the remainder are marked as Scouter individual;
10:   Update Leader individuals according to Equation (3);
11:   Update Follower individuals according to Equation (4);
12:   Update Scouter individuals according to Equations (1) and (2).
13:   $\boldsymbol{P}(t+1) \leftarrow \boldsymbol{P}(t)$;
14: end while
```

4. **Experimental Study.** To provide experimental evidence to study how the proposed algorithm improves the ability to track changing optimum in a dynamic environment than the other schemes, a commonly used dynamic test problem, Moving Peaks Benchmark (MPB) [18], is adopted in this study. The ANSPSO is compared with random immigration [19] PSO, Charged PSO (CPSO) [20] and standard PSO (SPSO). In this section, we firstly give a brief description of moving peaks benchmark, and then introduce experimental setting used in our study.

4.1. **Moving peaks benchmark.** The MPB is used by many researchers to test their dynamic optimization algorithm's performance. The MPB consists of a set of $\boldsymbol{m}$ peaks, each of which is defined by specific location ($\boldsymbol{X_i}$), heights (H_i), and widths (W_i). The peaks are distributed randomly in a search space. An MPB for n dimensions and m peaks has the following form:

$$f(\boldsymbol{x}, t) = \max_{i=1,\ldots,m} \frac{H_i(t)}{1 + W_i(t) \sum_{j=1}^{n} (x_j(t) - X_{ij}(t))^2} \tag{6}$$

Then, at a certain iteration step, the height, width and location of peaks are changed by adding a random Gaussian variable (σ) multiplied by a specific severity factor, $h_{severity}$ and $w_{sererity}$, respectively. A change of a single peak can be described as follows:

$$H_i(t) = H_i(t-1) + h_{severity} \cdot \sigma \tag{7}$$

$$W_i(t) = W_i(t-1) + w_{severity} \cdot \sigma \tag{8}$$

$$\boldsymbol{X}_i(t) = \boldsymbol{X}_i(t-1) + \boldsymbol{v}_i(t) \tag{9}$$

where the shift vector $\boldsymbol{v}_i$ is linear combination of random vector $\boldsymbol{r}$ and the previous shift vector $\boldsymbol{v}_i(t-1)$ and is normalized to movement length s, which determines the change severity of the environment. The movement of a single peak can be described as follows:

$$\boldsymbol{v}_i(t) = \frac{s}{|\boldsymbol{r} + \boldsymbol{v}_i(t-1)|}((1-\lambda)\boldsymbol{r} + \lambda \boldsymbol{v}_i(t-1)) \tag{10}$$

The random vector $\boldsymbol{r}$ is created by drawing a random number for each dimension and its length is normalized to the movement severity s. The parameter λ allows to control the trends, a random direction for $\lambda = 0$ or a direction depending on the previous director for $\lambda > 0$.

Table 1. Default settings of moving peaks benchmark

Parameter	Value
Number of peaks m	10
Frequency of change	5000
Search space range	$\in [0, 100]$
Peak heights	$\in [30, 70]$
Peak widths	$\in [1, 12]$
Height severity	7.0
Width severity	1.0
Peak shape	cone
Shift length s	$\in \{0.5, 1.0, 2.0, 3.0, 4.0, 5.0\}$
Number of dimension D	5
Correlation coefficient	0

4.2. **Experimental setup.** In our experiments, each algorithm is carried out with a population of 50 individuals. The random immigrants algorithms replace 10% worst individuals of the population with randomly generated individuals at each generation, i.e., 5 worst individuals will be replaced by the randomly generated particles. In CPSO, the parameters are set as follows: $Q = 0.15$, $p_{core} = 2.187$, $p_{repel} = 31.5$ (cf. [21]). The acceleration coefficient c_1 and c_2 are set to 2.05. In ANSPSO, the ratio among *Leader*, *Follower* and *Scouter* is set to 0.05 : 0.5 : 0.45 according to preliminary experiments.

Unless stated otherwise, the parameters of MPB have been set as follows: the search dimensionality D is set to 5, the search space is $X^D = [0, 100]^D$, there are $p = 10$ peaks, the peaks heights vary randomly within $[30, 70]$, and the peaks widths randomly draw from $[1, 12]$. These MPB parameter settings are summarized in Table 1. we also measure the performance of the algorithms under different change severity. In this paper, the performance of the algorithms was measured by the offline error, which is calculated as the average of the difference between the current best individual and the real optimum in current environment, for every function evaluation and for all environmental changes [22]:

$$e_{\mathit{off}} = \frac{1}{N_c N_e} \sum_{k=1}^{N_c} \sum_{j=1}^{N_e} \left(f_k^* - f_{kj}\right) \tag{11}$$

where N_c is the total number of environmental changes in a run, N_e is the total number of evaluations allowed before the next change, f_k^* is the theoretical optimum of the kth change, and f_{kj} is the best evaluation found by the algorithm since last change up to the jth evaluation of the kth environmental change. We also use the offline performance [9] to compare the performance in more detail of the selected algorithms. The offline performance can be described as follows:

$$F_p = \frac{1}{T} \sum_{t=1}^{T} f_t^* \tag{12}$$

where f_t^* is the best solution at time t, and T is the maximum evaluations allowed in a run.

5. **Results Analysis.** For each test case, we run each algorithm 30 times with different random seed, and each run consists of 30 environmental changes, i.e., there are 30×5000 evaluations in each run. Table 2 shows the means and variances of offline error obtained by the four algorithms. We also plot the offline performance of the algorithms as shown in Figure 1.

TABLE 2. Offline error and standard deviation for $s = 1.0$

Algorithm	Offline error	Standard deviation
ANSPSO	10.739	2.012
CPSO	13.978	4.654
RIPSO	12.616	2.575
SPSO	14.564	5.068

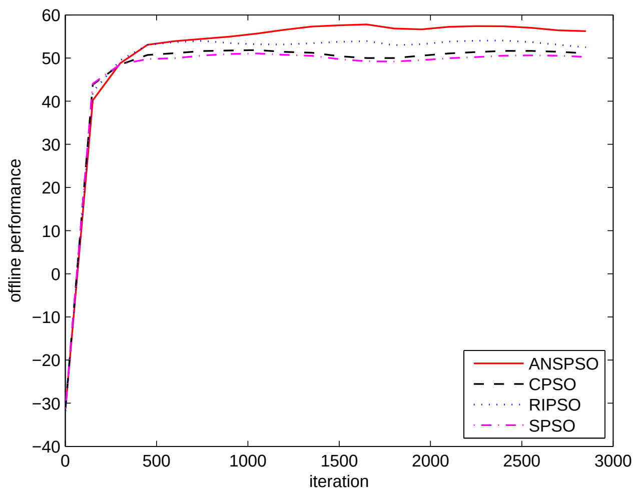

FIGURE 1. The offline performance of the algorithms with $s = 1.0$

5.1. **Performance analysis with shift length s = 1.0.** From Table 2 we can see that the proposed ANSPSO algorithm performed the best, and the SPSO performs the worst. The results show, although PSO has achieved a great success in static optimization problems, it cannot effectively track the changing optimum in the dynamic environment due to the convergence of the algorithm. As the parameters selected in our study, the performance of the CPSO is worse than RIPSO, which means that even introducing a certain number of randomly generated individuals in canonical PSO, it can effectively increase the population diversity, and thus enhance the ability to track the moving optima in the dynamic environment.

From Figure 1 we can draw the similar conclusion as Table 2 shows. Before the first environmental change, all the four algorithms evolve toward the optima and we cannot distinguish clearly from each other. After the first change, from Figure 1 we can see, with the evolution going on, the difference in the performance of the selected algorithms gradually shows up. Before the third environmental change, the proposed algorithm performs worse than the other algorithms. The superiority of the multi-role evolution does not emerge. Because at this time, the shift of the optima is little and the individual number which conducts the exploring work in ANSPSO is less than the other algorithms. While for canonical PSO, the global random search helps the algorithm to find the "static" optima. About 5 environmental changes later, the optima move away from the original position further and further. The canonical PSO loses the ability to track change optima because of the convergence, while the superiority strategy introduced in our algorithm shows up. The offline performance of the ANSPSO improves rapidly. Starting from the fifth change until the end, the ANSPSO performs significantly better than the other algorithms. The performance of CPSO and SPSO decreased slowly, and the performance of RIPSO maintains a steady level, which indicates the effectiveness of the immigration of new randomly generated individuals in a dynamic environment. Because we adopt multi-role evolution, the population uses particles with a different role to act exploring and exploiting works. Thus keep a good balance between the exploitation ability and exploration ability and maintain a high diversity, and it can effectively track the changing optima.

5.2. **Performance with different shift length.** From Table 3 we can see that the ANSPSO seems to be superior to the rest of the algorithms tested. Table 3 indicates that the performance of ANSPSO is significantly better than the performance of CPSO, RIPSO and SPSO on the MPB. With a small change severity $s = 0.5$, it is less hard for an algorithm to track the change of optimum, the ANSPSO achieves a low offline error which is less about 30% than that of the worst one, SPSO. As we have analyzed before, the RIPSO ranks the second, which confirms the simple introduction of new random generated individual strategy works well in a dynamic environment. With the shift severity gradually increasing, all of the algorithms performance decrease. The most rapid decline is CPSO and SPSO, with the shift severity varying from 0.5 to 5.0, the offline error increases from 13.452 to 22.598 and 14.354 to 22.228 respectively. The least decline of performance is the proposed algorithm, and then the second is the RIPSO. The offline error of the ANSPSO increases only about 42% with the shift severity change from 0.5 to 5.0.

Table 3. Offline error under different shift length(s)

Algorithms	Shift length					
	0.5	1.0	2.0	3.0	4.0	5.0
ANSPSO	10.024 ±2.468	10.739 ±2.012	12.852 ±1.413	13.812 ±1.530	14.025 ±1.302	14.192 ±0.895
CPSO	13.452 ±5.713	13.978 ±4.654	17.889 ±5.112	20.457 ±4.869	21.981 ±5.104	22.598 ±5.044
RIPSO	12.449 ±3.147	12.616 ±2.575	14.330 ±2.324	16.125 ±1.634	16.947 ±2.246	17.153 ±1.946
SPSO	14.354 ±5.984	14.564 ±5.068	18.312 ±5.213	20.443 ±4.725	21.565 ±4.652	22.228 ±4.464

From Table 3 we can see another interesting result. With the shift severity increase, the performance of all algorithms declined, although the ANSPSO offline performance also increased from 10.024 to 14.192, but the standard deviation of the ANSPSO decreased; this gives us a hint that the more severity of the environmental change, the more steady performance ANSPSO will achieve. The reason is that we have introduced multi-role evolution strategy in ANSPSO. In each iteration, there are three parts individuals with different roles to conduct exploring and exploiting works. The *Leader* individuals perform exploiting work around themselves in a certain range, while the *Follower* individuals perform exploring work around the *Leader* individuals in a wider range. The *Scouter* individuals perform exploring work randomly. In ANSPSO, as the parameters we selected in this study show, most individuals are assigned to the last two types of roles, thus ensuring the population diversity maintains a higher level in all severity of environmental change. The more severity of environmental change, the more superiority of the multi-role evolution appears.

6. **Conclusion and Future Work.** In a dynamic environment, the ability to continuously track the moving optimum over time is a key fact to measure the performance of an optimization algorithm. In this study, we extend PSO by introducing multi-role evolution strategy to tackle dynamic environments. Experiments on moving peaks benchmark function show the effectiveness of ANSPSO to capture the moving optima in a dynamic environment.

Future work will test other variants of the problem used (for example, varying the number of dimensions), and also use other benchmark functions. It would also be very interesting to see the performance of the proposed technique under different change period.

Acknowledgment. This work is supported by the Educational Commission of Hubei Province of China (No. D20142703). We also gratefully acknowledge the anonymous reviewers for their insight comments and useful suggestions.

REFERENCES

[1] T. T. Nguyen, S. Yang and J. Branke, Evolutionary dynamic optimization: A survey of the state of the art, *Swarm and Evolutionary Computation*, vol.6, pp.1-24, 2012.

[2] S. Hui and P. N. Suganthan, Ensemble differential evolution with dynamic subpopulations and adaptive clearing for solving dynamic optimization problems, *IEEE Congress on Evolutionary Computation*, pp.1-8, 2012.

[3] J. Karimi, H. Nobahari and S. Pourtakdoust, A new hybrid approach for dynamic continuous optimization problems, *Applied Soft Computing*, vol.12, no.3, pp.1158-1167, 2012.

[4] J. Branke, *Evolutionary Optimization in Dynamic Environments*, Kluwer Academic Publishers, Dordrecht, 2001.

[5] H. Cheng, Dynamic genetic algorithms with hyper-mutation schemes for dynamic shortest path routing problem in mobile ad hoc networks, *International Journal of Adaptive, Resilient and Autonomic Systems*, vol.3, no.1, pp.87-98, 2012.

[6] R. Tinós and S. Yang, A self-organizing random immigrants genetic algorithm for dynamic optimization problems, *Genetic Programming and Evolvable Machines*, vol.8, no.3, pp.255-286, 2007.

[7] S. Yang, Genetic algorithms with memory- and elitism-based immigrants in dynamic environments, *Evolutionary Computation*, vol.16, no.3, pp.385-416, 2008.

[8] H. Richter and S. Yang, Learning in abstract memory schemes for dynamic optimization, *The 4th International Conference on Natural Computation*, vol.13, no.12, pp.86-91, 2008.

[9] J. Branke, T. Kaußler, C. Schmidt and H. Schmeck, A multi-population approach to dynamic optimization problems, in *Adaptive Computing in Design and Manufacturing*, Springer, pp.299-308, 2000.

[10] J. Yao, N. Kharma and P. Grogono, BMPGA: A bi-objective multi-population genetic algorithm for multi-modal function optimization, *IEEE Congress on Evolutionary Computation*, vol.14, no.1, pp.80-102, 2010.

[11] J. Kennedy and R. Eberhart, Particle swarm optimization, *Proc. of the International Conference on Neural Networks*, vol.4, pp.1942-1948, 1995.

[12] W.-N. Chen, J. Zhang, H. S. H. Chung, W.-L. Zhong and W.-G. Wu, A novel set-based particle swarm optimization method for discrete optimization problems, *IEEE Trans. Evolutionary Computation*, vol.14, no.2, pp.278-300, 2010.

[13] C. Li and S. Yang, A clustering particle swarm optimizer for dynamic optimization, *IEEE Congress on Evolutionary Computation*, no.2, pp.439-446, 2009.

[14] J. Kennedy, R. C. Eberhart and Y. Shi, *Swarm Intelligence*, The Morgan Kaufmann Series in Evolutionary Computation, Elsevier Science Limited, 2001.

[15] R. Poli, J. Kennedy and T. Blackwell, Particle swarm optimization – An overview, *Swarm Intelligence*, vol.1, no.1, pp.33-57, 2007.

[16] S. He, Q. Wu and J. Saunders, Group search optimizer: An optimization algorithm inspired by animal searching behavior, *IEEE Trans. Evolutionary Computation*, vol.13, no.5, pp.973-990, 2009.

[17] D. Karaboga, *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Erciyes University, Tech. Rep. TR06, 2005.

[18] Branke, *The Moving Peaks Benchmark Website*, http://www.aifb.uni-karlsruhe.de/jbr/movpeaks.

[19] J. Grefenstette, Genetic algorithms for changing environments, in *Parallel Problem Solving from Nature 2*, Elsevier, pp.137-144, 1992.

[20] T. Blackwell and P. J. Bentley, Dynamic search with charged swarms, *Proc. of the Genetic and Evolutionary Computation Conference*, San Francisco, CA, Morgan Kaufmann Publishers Inc., pp.19-26, 2002.

[21] T. Blackwell and J. Branke, Multiswarms, exclusion, and anti-convergence in dynamic environments, *IEEE Trans. Evolutionary Computation*, vol.10, no.4, pp.459-472, 2006.

[22] I. G. del Amo, D. A. Pelta, J. R. González and A. D. Masegosa, An algorithm comparison for dynamic optimization problems, *Applied Soft Computing*, vol.12, no.10, pp.3176-3192, 2012.