Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems

Xiaohui Hu [1, 2] and Russell C. Eberhart [2]

[1] Department of Biomedical Engineering
Purdue University, West Lafayette, Indiana, USA
hux@ecn.purdue.edu

[2] Department of Electrical and Computer Engineering
Purdue School of Engineering and Technology, Indianapolis, Indiana, USA
reberhar@iupui.edu

Abstract- This paper introduces an adaptive PSO, which automatically tracks various changes in a dynamic system. Different environment detection and response techniques are tested on the parabolic and Rosenbrock benchmark functions, and re-randomization is introduced to respond to the dynamic changes. Performance on the benchmark functions with various severities is analyzed.

I. INTRODUCTION

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Kennedy and Eberhart [1, 2]. It exhibits common evolutionary computation attributes: 1. It is initialized with a population of random solutions. 2. It searches for optima by updating generations. 3. Updating is based on previous generations. In PSO, the potential solutions, called particles, are then "flown" through the problem space by following the current optimum particles.

The updates of the particles are accomplished according to the following equations. Equation 1 calculates a new velocity for each particle (potential solution) based on its previous velocity (V_{id}), the particle's location at which the best fitness so far has been achieved (p_{id}, or *pBest*), and the population global (or local neighborhood, in the neighborhood version of the algorithm) location (p_{gd}, or *gBest*) at which the best fitness so far has been achieved. Equation 2 updates each particle's position in solution hyperspace. The two random numbers are independently generated. The use of the inertia weight w has provided improved performance in a number of applications [3].

$$V_{id} = w \times V_{id} + c_1 \times rand() \times (p_{id} - x_{id}) + c_2 \times Rand() \times (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + V_{id} \quad (2)$$

A dynamic system means that the system changes state in a repeated or non-repeated manner. The changes may occur frequently or perhaps even almost continuously. There are several ways in which systems can change over time. In our previous work [4, 5], we defined three kinds of dynamic systems. First, the location in problem space of the optimum value can change. Second, the location can remain constant, but the optimum value can vary. Third, both the location and value of the optimum can vary. Furthermore, in a multidimensional system, these variations can occur on one or more dimensions, either independently or simultaneously. Some preliminary analysis using the particle swarm on the parabolic benchmark function has been reported [4, 5].

In this paper, we focus on the locations varying in the problem space where the optimum value occurs. We vary the location simultaneously and equally on each dimension. This choice was made primarily to facilitate comparison with prior work.

II. BACKGROUND

Since 1995, particle swarm optimization has been proven to be very effective for static problems. Much work has been done in this area [6, 7]. However, tracking dynamic systems using PSO is still a new area. Other dynamic tracking algorithms have been used in this area for evolutionary programming [8] and evolution strategies [9]. Eberhart repeated the dynamic tracking procedures with PSO and demonstrated successful tracking of a 10-dimensional parabolic function with a severity of up to 1.0 [4]. Carlisle [10] also tried to use PSO to track dynamic enviroments with continuous changes.

However, there are problems with some of the algorithms. Most of the algorithms mentioned above do not detect the environment changes automatically; they need some outside information about the changes. These requirements are usually not available in real world problems. So some kind of automatic environment sensing technique should be employed to solve the problem.

In previous work, we introduced a "changed-*gBest*-value" method to detect the environmental changes [5]. In each generation, the global optimum location is re-evaluated. If there is a change, it means the evaluated function has changed. The method is based on the assumption that if the optimum location of a dynamic system changes, the optimum

0-7803-7282-4/02/$10.00 ©2002 IEEE

value of the current location also changes. Carlisle [10] also proposed a similar algorithm by monitoring randomly picked positions in the search space. It makes the same assumptions discussed above.

This might not be true in some systems. So a new "fixed-*gBest*-value" method is introduced to solve the problem. If the PSO cannot follow the dynamic changes, the *gBest* value will be trapped into the previous optimum value and never changed. So we monitor the *gBest* value, and if it has not changed for certain number of iterations, there is a possible optimum change. In reality, we monitor the *gBest* value and second-best *gBest* value to increase the accuracy and prevent false alarms. This fixed-duration number should not be too small or too large. If it is too small, there can be many false alarms, which means it found environment changes that do not exist. If it is too large, PSO will delay the response to dynamic changes.

After the detection of environment changes, there must be a strategy to effectively respond to a wide variety of changes. In previous work, resetting the particle's memory is used to respond to the changes [4]. However, there might be some problems. For example, if the whole population has already converged to a small area, it might not be easy to jump out to follow the changes. So, we are investigating the re-randomization of a portion of the population when a change is detected. We tested different re-randomization methods.

In the PSO algorithm, there is already a mechanism to adapt to environmental changes. The swarm searches for optima in the solution space and typically shrinks to a smaller search area step by step. If the dynamic changes occur in this search area, the PSO will automatically find the new optimum without any modification.

III. EXPERIMENTAL DESIGN

For the inertia weight, we used values we used previously [4, 5]. The inertia weight was [0.5 + (Rnd/2.0)]. The population size used in PSO is often between 10 and 40. Carlisle *et al.* compared different population sizes, and suggested 30 is a proper choice [11]. Here we use 30 as the population size.

The dynamic range for the parabolic function was set to [-50, 50] on each dimension. A smaller range of [-10, 10] is often used for the parabolic (or spherical) function; see Eberhart and Shi [12] and Kennedy, Eberhart and Shi [7]. We set the maximum velocity *Vmax* equal to the dynamic range on each dimension.

We tested various detection and response methods. There are two detection methods: the changed-*gBest*-value method, and the fixed-*gBest*-value method. For the fixed-*gBest*-value method, we chose 20 as the fixed-duration number. The methods are illustrated in Table 1.

Table 1: Detection and response methods.

Detection methods	A	Detect the changes of *gBest* value by re-evaluating fitness
	B	Monitor the *gBest* and second *gBest* value to check if they have no changes for 20 iterations
Response Methods	0	Do nothing
	1	Re-randomize 10% of particles and reset other particles
	2	Re-randomize 10% of particles
	3	Re-randomize *gBest* and reset other particles
	4	Re-randomize *gBest*
	5	Reset all particles
	6	Re-randomize 50% of particles and reset other particles
	7	Re-randomize 50% of particles
	8	Re-randomize all particles

* The particles re-randomized are randomly picked.

The parabolic function was used as the test case. The reason is that it is easy to control the dynamic changes. It is defined in Table 2. There is a dynamic parameter *offset* in the function that changes in various ways.

Table 2: Benchmark function.

Function	Equation	Parameters
Parabolic (De Jong F1)	$f(\vec{x}) = \sum_{i=1}^{n} (x_i - offset)^2$	Dimensions: 10 Range: [-50, 50] Error level: 0.0001

During testing, PSO finds the optimum first and records the number of iterations needed to reach the required error level, then dynamic changes to the function are applied, and PSO continues to find the new optima and records the number of iterations needed to re-reach the required error level. PSO is repeated for 100 runs for each combination of the detection methods and response methods.

IV. RESULTS

A. Comparison of the detection techniques

We compared two environment detection techinques. Both of them successfully tracked the dynamic changes. The following table and graph show the results, each column shows a different method; the codes representing the different methods are from Table 1. Each double row represents a different severity (*offset* in the equation). The double rows are divided into two rows. The upper row shows the average number of iterations (100 runs) that PSO used the find the optimum, while the lower row shows the average iterations PSO used after the dynamic changes.

Following Table 3 the results are shown. Figures 1, 2, and 3 compare the two different detection methods side by side. Each figure shows two different detecion methods with the same response methods. The performance is showed by average iterations required by the methods used to track the changes.

Table 3: Parabolic function tracking results.

	A1	B1	A6	B6	A8	B8
10	222.57	220.06	220.46	222.23	221.36	222.19
	221.45	241.61	213.97	233.60	217.78	238.31
1	220.25	221.60	221.25	222.74	221.85	218.26
	186.74	214.12	190.01	209.27	219.48	240.74
0.1	219.17	222.36	223.55	221.76	219.76	219.77
	115.18	133.20	144.20	160.75	221.41	241.65
0.01	219.71	221.60	221.37	222.85	221.68	221.53
	54.64	72.92	73.02	90.74	221.80	243.31
0.001	222.59	221.70	224.06	220.67	222.74	222.48
	15.98	32.41	26.23	41.63	221.26	240.67
0.0001	222.15	223.06	224.41	221.02	223.03	222.45
	1.62	1.61	2.87	1.07	166.51	2.31
0.00001	220.36	221.40	221.79	223.96	222.59	221.58
	0.37	0.08	1.47	0.06	141.87	0.07

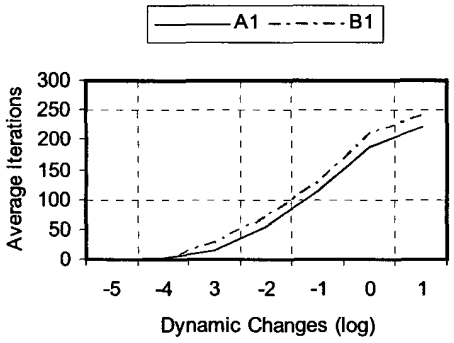

Figure 1: Comparison of two detection methods.

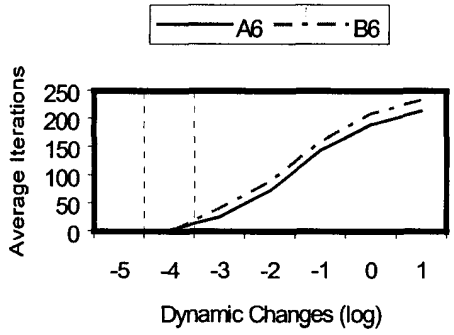

Figure 2: Comparison of two detection methods.

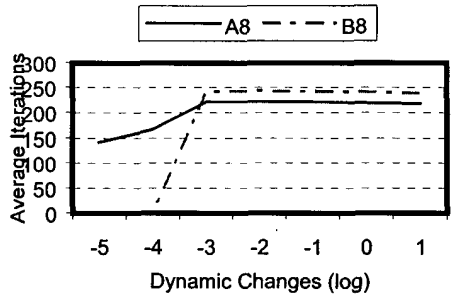

Figure 3: Comparison of the two different detection methods.

The above results show that both detection methods successfully detect the dynamic changes. The average iterations needed to follow the optimum in method A is faster than method B. However, as we stated, method B can be used in cases where method A is not acceptable.

B. Comparison of the response methods.

Table 4: Mean iterations needed to follow the optimum (100 runs, parabolic).

	A0	A1	A3	A5	A6	A8
10	221.7	222.57	221.52	220.59	220.46	221.36
	NA	221.45	223.65	310.17	213.97	217.78
1	222.64	220.25	222.52	220.17	221.25	221.85
	NA	186.74	191.32	196.22	190.01	219.48
0.1	221.04	219.17	221.19	223.35	223.55	219.76
	NA	115.18	113.81	108.82	144.2	221.41
0.01	221.86	219.71	224.67	220.55	221.37	221.68
	NA	54.64	52.72	52.4	73.02	221.8
0.001	224.46	222.59	220.39	221.48	224.06	222.74
	NA	15.98	16.55	14.86	26.23	221.26
0.0001	221.49	222.15	224.37	221.68	224.41	223.03
	1.32	1.62	2.44	0.77	2.87	166.51
0.00001	222.79	220.36	222.51	222.76	221.79	222.59
	0.06	0.37	1.78	0.08	1.47	141.87

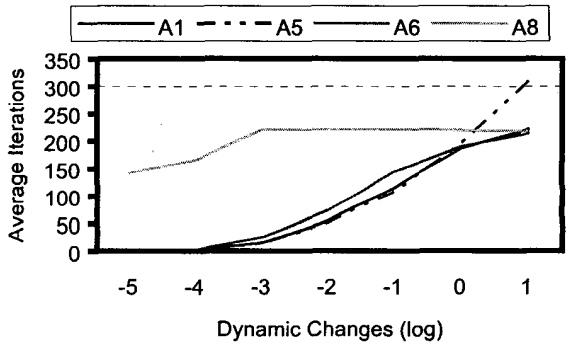

Figure 4: Comparison of response methods.

The above results show the performance of the different re-randomization rates. Total re-randomization is not acceptable since it starts a new search every time regardless of the dynamic changes. Lower re-randomization rates are better than higher re-randomzation rates when the changes are small. When changes are large, there is no big difference. If there is

no re-randomization, the performance is bad when there are big dynamic changes.

V. CONCLUSIONS

The modified particle swarm optimization can automatically track dynamically varying functions. Two new features are added to PSO: environment detection and response.

We introduced two enviroment detection methods: changed-*gBest*-value method and fixed-*gBest*-value method. Both of them can successfully detect the various dynamic changes. Each method has its own advantages: the changed-*gBest*-value method is fast but requires extra time to re-evaluate the fitness; the fixed-*gBest*-value method is slow but can be used in any situation.

Attention must be paid to a parameter in the fixed-*gBest*-value method. That is number of iterations when the *gBest* is fixed. The system performance is sensitive to the number of iterations. If it is too big, the PSO has slow response to changes, if it is too small there might be many false alarms. And it is highly problem dependent. For slow-convergence problems, the number should be big enough to avoid false alarms; for fast changing environments, the number should be small enough the follow the changes.

After the detection of environment changes, there must be a strategy to effectively respond to a wide variety of changes. In the case of particle swarm, we are investigating the re-randomization of a portion of the population when a change is detected. We tested different re-randomization methods. A 10 percent re-randomization rate is a good choice for most cases up to now. An alternative method is to re-randomize the *gBest* particle.

Here only a simple parabolic function test case is presented. For more complicated problems, further investigation is needed to the test the different detection and response techniques.

VI. DISCUSSION

As a evolutionary algorithm, particle swarm optimization has been successfully applied to many applications. Currently the information processing of PSO is focused on two levels of intelligence. In the velocity update formula, one is the individual cognition component and the other is social communication component. The individual cognition component represents the search ability of the particle itself. (c_1, P_{id}, V_{id}, V_{max}, w). The social communication component represents the influence from the social environemnt (c_2, P_{gd}).

However, for some more complicated applications, for example, dynamic systems, higher level intelligence should be integrated into the algorithm. This kind of intelligence either cannot be furnished by particles themselves or is not easy to accomplish without losing performance. Environment detection and response are new intelligence methods which cannot be achieved by individual particles. This is population-level intelligence; in this level, we can integrate more human knowledge and intelligence to the PSO to accommodate complex systems.

Another problem introduced is the birth and death of the particles. Unlike other evolutionary algorithm, PSO began as a social simulation. All the particles are independent and no crossover or mutation is used, so each particle will survive to the end of the search. However, in the adaptive PSO, we introduced re-randomization rate. It can be viewed as the death of old particles and birth of new particles. Particles now have life cycles. Regarding this, further investigation is needed.

REFERENCES

[1] Eberhart, R. C. and Kennedy, J. A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science.* Nagoya, Japan, pp. 39-43, 1995.

[2] Kennedy, J. and Eberhart, R. C. Particle swarm optimization. *Proc. IEEE Int'l Conf. on Neural Networks*, Piscataway, NJ: IEEE Press, IV pp.1942-1948, 1995.

[3] Shi, Y. and Eberhart, R. C. A modified particle swarm optimizer. *Proceedings of the IEEE International Conference on Evolutionary Computation.* Piscataway, NJ: IEEE Press, pp. 69-73, 1998.

[4] Eberhart, R. C. and Shi, Y. Tracking and optimizing dynamic systems with particle swarms. *Proc. Congress on Evolutionary Computation 2001*, Piscataway, NJ: IEEE Press, pp. 94-97, 2001.

[5] Hu, X. and Eberhart, R. C. Tracking dynamic systems with PSO: where's the cheese? *Proceedings of the workshop on particle swarm optimization.* Purdue School of Engineering and Technology, Indianapolis, IN, 2001.

[6] Eberhart, R. C. and Shi, Y. Particle swarm optimization: developments, applications and resources. *Proc. Congress on Evolutionary Computation 2001.* Piscataway, NJ: IEEE Press, pp.81-86, 2001.

[7] Kennedy, J., Eberhart, R. C., and Shi, Y., *Swarm Intelligence.* San Francisco CA: Morgan Kaufmann Publishers, 2001.

[8] Angeline, P. J. Tracking extrema in dynamic environments. *Proc. Evolutionary Programming VI.* Indianapolis, IN. pp. 335-345, 1998.

[9] Back, T. On the behavior of evolutionary algorithms in dynamic environments. *Proc. Int. Conf. on Evolutionary Computation.* Piscataway, NJ: IEEE Press, pp. 446-451, 1998.

[10] Carlisle, A. and Dozier, G. Adapting particle swarm optimization to dynamic environments. *Proceedings of International Conference on Artificial Intelligence.* Las Vegas, Nevada, USA. pp. 429-434, 2000.

[11] Carlisle, A. and Dozier, G. An off-the-shelf PSO. *Proceedings of the workshop on particle swarm optimization.* Purdue School of Engineering and Technology, Indianapolis, IN, 2001.

[12] Eberhart, R. C. and Shi, Y. Comparing inertia weights and constriction factors in particle swarm optimization. *Proc. of Conference on Evolutionary Computation 2000,* Piscataway, NJ: IEEE Press, pp. 84-88, 2000.