Distributed Adaptive Particle Swarm Optimizer in Dynamic Environment

Xiaohui Cui and Thomas E. Potok

Applied Software Engineering Research
Computational Sciences and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6085
cuix, potokte@ornl.gov

Abstract

Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique that can be used to find an optimal, or near optimal, solution to a numerical and qualitative problem. In the PSO algorithm, the problem solution emerges from the interactions among many simple individual agents called particles. In the real world, we have to frequently deal with searching and tracking an optimal solution in a dynamical and noisy environment. This demands that the algorithm not only find the optimal solution but also track the trajectory of the non-stationary solution. The traditional PSO algorithm lacks the ability to track the changing optimal solution in a dynamic and noisy environment. In this paper, we present a distributed adaptive PSO (DAPSO) algorithm that can be used to track a non-stationary optimal solution in a dynamically changing and noisy environment.

1. Introduction

A dynamically changing environment presents a challenge to track an optimal solution. Because of the continual changes of both the external environment and parameters, the optimal solution in the environment will also change with time. This demands that the optimal algorithm for dynamic environment not only find the solution in a short time, but also track the trajectory of the optimal solution in the dynamic environment. Particle Swarm Optimization (PSO)[1] has been proven to be both effective and efficient to solve a diverse set of optimization problems.[2-6] In the past several years, PSO has been successfully applied in many research and application areas. It has been demonstrated that PSO could provide better results in a faster, cheaper way than other methods. [7]

However, the traditional PSO algorithm lacks the ability to track the non-stationary optimal solution in the dynamically changing environment.[8] The PSO algorithm does not have a mechanism to respond to the environment change. In this paper, we propose a Distributed Adaptive PSO (DAPSO) for searching and tracking the non-stationary optimum in a dynamically changing environment.

The remainder of this paper is organized as follows: in Sect 2, a brief overview of PSO and a discussion of the shortcomings of PSO in a dynamic environment are presented. Various modified PSO approaches for dynamic environment are introduced in Sect 3. In Sect 4, the DAPSO approach is described in detail. Experiment setup and results in comparisons the performance of DAPSO and other modified PSO algorithms in the dynamical environment are presented in Sect 5. Discussion and Conclusion are in Sect 6.

2. Particle Swarm Optimization Algorithm

PSO was originally developed by Eberhart and Kennedy in 1995,[1] inspired by the social behavior of the bird flock. In the PSO algorithm, birds in a flock are symbolically represented as particles. These particles can be considered as simple agents "flying" through a problem space. A problem space in PSO may have as many dimensions as needed to model the problem space. A particle's location in the multi-dimensional problem space represents one solution for the problem. When a particle moves to a new location, a different problem solution is generated. This solution is evaluated by a fitness function that provides a quantitative value of the solution's utility.

The velocity and direction of each particle moving along each dimension of the problem space are altered at each generation of movement. It is the particle's personal experience combined with its neighbors' experience that influences the movement of each particle through a problem space. For every generation, the particle's new location is computed by adding the particle's current velocity V-vector to its location X-vector. Mathematically, given a multi-dimensional problem space, the ***i***th particle changes its velocity and location according to the following equations[7, 9]:

$$v_{id} = w*(v_{id} + c_1 * rand_1 * (p_{id} - x_{id}) + c_2 * rand_2 * (p_{gd} - x_{id})) \quad (1a)$$

$$x_{id} = x_{id} + v_{id} \quad (1b)$$

where, p_{id} is the location of the particle where it experiences the best fitness value; p_{gd} is the location of the particle experienced the highest best fitness value in the whole population; $\boldsymbol{x}_{id}$ is the particle current location; c_1 and c_2 are two positive acceleration constants; *d* is the number of dimensions of the problem space; $rand_1$, $rand_2$ are random values in the range of (0,1). ***w*** is called the constriction coefficient[7] and it is computed according to Eq.(2a):

$$w = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \quad (2a)$$

$$\varphi = c_1 + c_2, \varphi > 4 \quad (2b)$$

Eq.(1a) requires each particle to record its current coordinate $\boldsymbol{x}_{id}$, its velocity $\boldsymbol{V}_{id}$ that indicates the speed of its movement along the dimensions in a problem space, its personal best fitness value location vector P_{id} and the whole population's best fitness value location vector P_{gd}. The best fitness values are updated at each generation based on Eq.(3), where the symbol *f* denotes the fitness function; $P_i\ (t)$ denotes the best fitness values and the coordination where the value calculated; and *t* denotes the generation step.

$$P_i(t+1) = \begin{cases} P_i(t) & f(X_i(t+1)) \le f(X_i(t)) \\ X_i(t+1) & f(X_i(t+1)) > f(X_i(t)) \end{cases} \quad (3)$$

The P_{id} and P_{gd} and their coordinate fitness values $f(P_{id})$ and $f(P_{gd})$ can be considered as each individual particle's experience or knowledge and Equation 3 is the particle's knowledge updating mechanism. In PSO, particles' knowledge will not be updated until the particle encounters a new vector location with a higher fitness value than the value currently stored in its memory. However, in a dynamic environment, the fitness value of each point in the problem space may change over time. The location vector with the highest fitness value ever found by a specific particle may not have the highest fitness value after several generations. It requires the particle to renew its memory whenever the real environment status does not match the particle's memorized knowledge. However, the traditional PSO lacks an updating mechanism to monitor the change of the environment and renew the particles' memory when the environment changed. As a result, the particle continually uses the outdated experience/knowledge to direct its search, which inhibits the particle from following the moving path of the current optimal solution and eventually, results the particle to be easily trapped in the region of the former optimal solution.

3. Related Work

To stop particles using the outdated knowledge in a dynamic environment, Carlisle [8] and Eberhart [10] proposed to periodically reset all particles' memory and replace the particle's best fitness value and location vector with its current location vector and fitness value to force the particle to "forget" its former experience. One major disadvantage of this reset mechanism is the difficulty of determining the reset frequency. Without prior-knowledge about the environment changing frequency, the particle's memory reset frequency needs to be a high value to capture the changing step of the environment.

However, high resetting frequency reduces the efficiency of the convergence of the PSO. The essence of the PSO algorithm lies in each particle's learning from both its past search experience and its neighbor's past search experience and utilizing this knowledge to guide its next moving velocity. Periodic resetting may cause all particles to lose their knowledge gathered during the search and restart learning. This decreases the search efficiency of the swarm. Especially during the initial period of searching, frequently resetting the personal best vector may cause particles unable to quickly converge on the vicinity of the optimal solution. Following each reset, the optimization algorithm needs extra time to re-evaluate each particle's current fitness value.

In 2002, Carlisle [11] proposed an adaptive PSO (APSO) for monitoring and reacting the changing of the dynamic environment. Carlisle introduced a new notion, "sentry", in his APSO algorithm. The "sentry" is one or many special designed particles that are deployed in the problem space to monitor the environment changes. When the "sentry" detects a change in the environment, it informs all others and forces other particles to reset their memory.

However, the "sentry" can only detect the local changes where the "sentry" point resided. Some complex environments only exhibit local changes, which may not be detected by the "sentry". In most real world applications, the fitness value is not stable because of the environment noise interference, the “sentry” may be constantly triggered by the environment noise and requesting all other particles reset their memory. In addition, this algorithm alters the classical PSO’s decentralized processing model into an essentially centralized control model. All other particles have to depend on one or a limited number of sentries for detecting and reacting to the change of the environment, which reduces the robustness of the whole system. Designing a particle that is capable of working as a sentry to monitor the environment will also increase the complexity of the entire system and make it hard to implement the modified PSO in a distributed environment.

Inspired from the multi-population Evolutionary Algorithm approaches, Blackwell[12] proposed a multi-swarm PSO that maintains multiple particle groups that mutually repels each other to prevent all particles from converging at the same optimum. The multiple particle group approach is adaptive in a highly complex multimodal dynamic environment where multiple peeks exist. However, this adaptive is generated by largely increasing the particle number in the algorithm, which eventually increases the computational complex of the algorithm. It is necessary to find a new method for particles to renew their memory without any centralized control and to maintain simplicity of each particle.

4. Distributed Adaptive PSO Approach

In this research, we propose a new modified PSO, the distributed adaptive PSO approach (DAPSO). In DAPSO, there is no specially designed particle to monitor the change of the environment and there is no additional fitness evaluation computing to enable the particle to adapt to the changed environment. Like the traditional PSO, each particle uses the Eq (1) to determine its next velocity. The only difference is each particle will compare the fitness value of its current location with that of its previous location. If the current fitness value doesn’t have any improvement compare to the previous value, the particle will use Eq.(4) for the fitness value update. Eq.(4) is slightly different compare to the traditional fitness value update function in Eq.(3).

$$P_i(t+1) = \begin{cases} P_i(t)*T & f(X_i(t+1)) \le P_i(t)*T \\ X_i(t+1) & f(X_i(t+1)) > P_i(t)*T \end{cases} \tag{4}$$

In Eq.(4), a new notion, the evaporation constant ***T***, is introduced. ***T*** has a value between 0 and 1. The personal fitness value that is stored in each particle’s memory and the global fitness value of the particle swarm will gradually evaporate (decrease) at the rate of the evaporation constant ***T*** over time.

Once the particle continuously fails for improving its current fitness value by using its previous searching experience, the particle’s personal best fitness value as well as the global best fitness value will gradually decrease. Eventually, the personal and global best fitness value will be lower than the fitness value of the particle’s current location and the best fitness value will be replaced by the particle’s current fitness value. Although all particles have the same evaporation constant ***T***, each particle’s updating frequency may not be the same. The updating frequency depends on the particle’s previous personal best fitness value ***f(P)*** and the current fitness value ***f(X)*** that the particle acquired. The particle will update its best fitness value more frequently by using the current fitness value when the ***f(P)*** is lower and the ***f(X)*** is higher. However, when the ***f(P)*** is higher and the ***f(X)*** is lower in a changing environment, it indicates the particle’s current location is far away from the current optimal solution compared to the distance between the optimal solution and the best fitness value’s position stored in the particle’s memory. Usually the new environment (after change) is closely related to the previous environment from which it evolutes. It would be beneficial to use the knowledge/experience about the previous search space to help searching for the new optimal. In this situation, the particle will keep the best fitness value in its memory until the best fitness value becomes obsolete. The fitness value update equation enables each particle to self-adapt to the changing environment. This mechanism is similar to the human society’s knowledge/experience learning and updating. Human obtains knowledge through personal experience and social experience. When human can not improve his/her problem solving capability by following the experience and knowledge he or she early acquired, he or she may gradually reduce the weight of the knowledge in his/her problem solving practice and gradually alter the knowledge with the new knowledge or experience obtained from practice.

5. Experimental Implementation

5.1. Environmental Simulation

The simulated dynamic environment can be constructed by using the Test Function Generator, DF1, proposed by Morrison and De Jong.[13] The generator is capable of generating a given number of peaks (optimal solutions) in a given number of dimensions. For a two dimensional problem, the static evaluation function in DF1 is defined as following:

$$f(X,Y) = MAX[H_i - R_i * \sqrt{(X-x_i)^2 + (Y-y_i)^2}] \quad (i=1,...N) \quad (5)$$

where ***N*** denotes the number of peaks in the environment. The (***x_i*, *y_i***) represents each cone's location. R_i and H_i represent the cone's height and slope.

Different movement functions generate different types of dynamic changing environments. In this research, the environment changing rate is controlled through the following logic function [13] :

$$Y_i = A * Y_{i-1} * (1 - Y_{i-1}) \quad (6)$$

where ***A*** is a constant and $\boldsymbol{Y_i}$ is the value at the iteration. The ***Y*** value produced on each iteration will be used to control the changing step sizes of the dynamic environment. In this research, the dynamic environment is simulated by the movement of the cone's location ($\boldsymbol{x_i}$, $\boldsymbol{y_i}$). The ***Y*** value represents the cone location's moving velocity.

In the real applications, the evaluated fitness value can not always be calculated in precision. Most of the time, the fitness value will be polluted by some degree of noise. To simulate this kind of noise pollution in the fitness evaluation of the dynamic environment, we generate the noise polluted fitness value with the following approach. At each iteration, the fitness value ***f(x)*** can only be obtained in the form of $\boldsymbol{f^n(x)}$, where $\boldsymbol{f^n(x)}$ is the approximation of ***f(x)*** and contains a small amount of noise ***n***. The function can be represented as[14]:

$$f^n(x) = f(x) * (1+\eta), \ \eta \sim N(0, \sigma^2) \quad (7)$$

where η is a Gaussian distributed random variable with zero mean and variance σ^2. Therefore, at each time, the particle will get a $\boldsymbol{f^n(x)}$ evaluation value instead of ***f(x)***.

5.2. Experiment Setup

To evaluate the efficiency of the DAPSO algorithm in tracking the movement of the optimum in a dynamic environment, the performance of the DAPSO algorithm and the three different modified PSO algorithms are compared over the dynamic environment generated by using DF1. These four PSO algorithms include DAPSO, standard PSO, APSO[11] and PSO with constant memory reset. [10] These PSO models share the same standard PSO configuration and are tested in the same dynamical environment. Twenty particles are randomly distributed in a two dimensional environment with ±100 width in each dimension. All of these four PSO algorithms use Eq.(1) as their velocity update mechanism. In Eq.(1), $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ are set to 2.05 and $\boldsymbol{V_{max}}$ is set to 0.03*dimension size. [10] The value of ***w*** is set to 0.72. [7] The algorithms are implemented with Matlab 6.5® and run on a 3.0G HZ CPU and 2.0G memory Windows XP platform.

The first implementation is the proposed DAPSO algorithm. In this implementation, particles use Eq.(4) to update their best fitness value. The evaporation constant $\boldsymbol{T_p}$ for the personal best fitness value and the constant $\boldsymbol{T_g}$ for the global best fitness value are set as ***exp(-0.85)*** and ***exp(-3/particles number).*** The detail of discovering the values of $\boldsymbol{T_p}$ and $\boldsymbol{T_g}$ is discussed in. [15]

The second implementation is the PSO with constant memory resetting model. In this modified PSO algorithm, all particles will automatically reset the personal best fitness value and location vector and replace them with the particle's current fitness value and location vector in a pre-set frequency. In this experiment, we set the reset frequency as 15 iterations.

In the implementation of the APSO algorithm, at every iteration, a randomly chosen particle is automatically elected as the "sentry". This "sentry" particle will not update its moving speed and location as other particles. Instead, it will re-evaluate the fitness value of its previous location. If the value changed, the "sentry" will alarm all other 19 particles and force each particle to replace the personal best fitness value and location vector with the particle's current fitness value and location vector. As we discussed in the previous session, the sentry may be triggered by the noise polluted fitness value and it will cause all particles to frequently reset the memory even though the environment doesn't change. In this implementation, a noise threshold is introduced. The sentry will alarm other particles to reset the memory only when the difference between the current fitness value and the previous fitness value is larger than the noise threshold. Although it takes extra time for the sentry to monitor the change of the dynamic environment, the reset memory frequency in APSO more accurately indicates the change of the dynamic environment than the PSO with constantly reset memory model.

The last implementation in the experiment is the standard PSO algorithm, which is used as the reference for the three modified PSO implementation. The difference of the standard PSO from the traditional PSO is the standard PSO uses different parameters to make particles that can quickly converge to the optimal solution.

5.3 Measurement

The ability for the algorithm to track optimum in the dynamic environment is measured by the distance between the particle with the best fitness value and the cone with the highest peak at each iteration. The distance value shows the tracking ability of algorithm in the entire searching procedure. If the algorithm can keep at least one particle located in a short distance from the optimal solution at anytime, regardless of the solution's movement, this distance value will be kept in low value in the whole searching period. In this research, the iterations for searching optimum are set as 200. Each algorithm implementation will run 100 times, and the distance value of every iteration is the average value over 100 runs.

6. Experiment Results

The performances of the four algorithms in tracking non-stationary optimal solution are shown in Fig 1. Figure 1 displays the shortest distance values between the optimal solution and the particles controlled by each algorithm from iteration 10 to iteration 200. The smaller the distance value, the better the algorithm's solution in the dynamic environment. At the initial stage, particles are randomly deployed in the environment and the distance between the particle and the optimal solution are usually high. For easy displaying the change pattern of the distance value in the high iteration stage, the distance value for iteration 1 to iteration 10 are not displayed in Fig 1.

As shown in Fig 1, compared to the other three algorithms, DAPSO performs efficiently in the dynamically changing environment. Although the optimal solution is continually changing in the entire searching period, the DAPSO algorithm can maintain the lowest distance (below 0.5) between the best fitness value particle and the optimal solution.

The APSO algorithm is the second best efficient dynamic optimization tracking algorithm in the experiment. The APSO algorithm implementation can maintain the shortest distance from the optimal solution between 0.5 and 1.

The PSO with constant memory reset model implementation generates a saw shape curve of the distance between the best solution of algorithm and the optimal solution. As we indicated in Sect 2, frequently resetting the particle's memory will result in losing the tracking of the optimal solution and it cost the algorithm several additional iterations to re-discover the new optimum in the environment. On the other hand, low frequency resetting may cause the algorithm to be trapped in an outdated optimum location until the memory resetting start.

The distance values for the standard PSO model are too high to be displayed in the same figure with the other three modified PSO distance values. In Fig 1, we use the distance's log value to represent the changing pattern of shortest distance from particles to the optimal solution. As shown in Fig 1, the standard PSO algorithm failed to track the movement of the optimal solution in the dynamically changing environment.

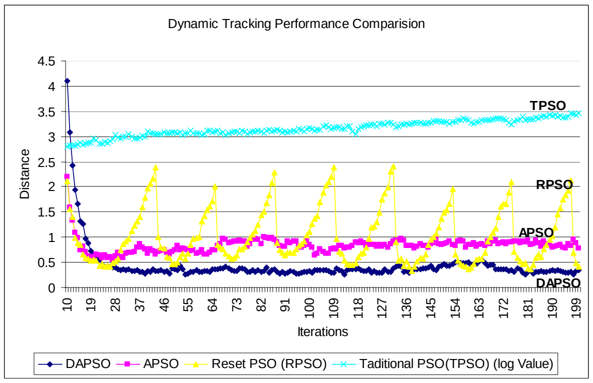

Fig 1, Performance comparison of DAPSO, PSO with constant memory reset, APSO and traditional PSO.

The quantitative comparison of the tracking performances between different PSO models is shown in Table 1. The summary of the distance values of each PSO model during the whole searching procedure are listed in the table. DAPSO has the lowest distance summary value that indicates the DAPSO can maintain the shortest distance between the particle and the optimal solution. Therefore, the DAPSO has the best performance in continuously tracking the movement of the optimal solution in a dynamically changing environment.

Table 1. The Summary of the Distance Value between the Optimal Solution and the Particle that has the Highest Fitness Evaluation Value of all Iterations

Algorithms	Summary of distance
DAPSO	90.99
APSO	217.69
Traditional PSO	8818
PSO with constantly memory reset	284.48

7. Discussion and Conclusion

Most papers reporting applications of the optimization algorithms only discuss the scenario in the static environment. The performance evaluation of various approaches is mainly based on how fast an approach can find the optimal point in the benchmark problems. It has been proven that PSO is very effective in applications with static environment. However, the real world is rarely static, and a frequently changing solution space may cause the optimal solutions changing over time. The optimal solution found at time $\boldsymbol{T_1}$ may no longer valid at time $\boldsymbol{T_2}$. When the problem space is dynamically changing, the goal of optimization is not only to acquire the optimal solution but also to track the solution's trajectory in the whole searching procedure as closely as possible. The traditional PSO has difficulty in tracking the non-stationary solutions. The reason is that PSO lacks a mechanism to update each particle's knowledge obtained from the environment. That will induce a bias toward searching the region that once held the optimum; however, this region may not contain the most recent goal.

In this paper, we present a new approach, DAPSO, a modified PSO, for tracking the optimization solution in a dynamically changing environment. Unlike other adaptive PSO algorithms for the dynamical environment, which needs one or more special designed "sentry" particles to detect the change of the environment and to control other particles' action, each particle in DAPSO individually updates its knowledge based on the local environment status that the particle perceived. Furthermore, all particles in the DAPSO system are homogenous.

In DAPSO, each particle adaptively updates its memory based on the performance results stored in its memory and the local environment. If its performance doesn't improve when the particle uses the previous searching knowledge stored in its memory, the particle will gradually reduce the impact of the stored knowledge on its decision for the next searching direction and speed. If the particle finds the newest and better fitness value, the value and location vector will be used to replace the outdated knowledge that is stored in the particle's memory. The updating frequency of the particle's memory is determined by the individual particle's knowledge evaporation rate and the current fitness value that the particle perceived from the point it located. Each particle in the system may update its memory in different generations. Thus, each particle does not need to know the results from other particles of the population until a global best is found. DAPSO can be easily implemented in a distributed computing environment because the centralized control is not required in DAPSO.

Our simulation experiment results indicate that DAPSO can more efficiently track the movement of an optimal solution in a dynamically changing environment, than other modified PSO models. Because each particle updates its memory only based on its perception and its knowledge evaporation rate, this DAPSO algorithm can avoid losing tracking the optimal solution as happened in other modified PSO approaches that are based on resetting the memory periodically.

Acknowledgments

Prepared by Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, Tennessee 37831-6285, managed by UT-Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR22725.

References

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

[2] Cruz, Jr., G. Chen, D. Li, and X. Wang, "Particle Swarm Optimization for resource allocation in UAV cooperative control," *AIAA Guidance, Navigation, and Control Conference*, pp. 2549-2559, Providence, RI, United States, August, 2004.

[3] X. Cui and T. E. Potok, "Document Clustering Analysis Based on Hybrid PSO+K-means Algorithm," *Journal of Computer Sciences*, Special Issue, pp. 27-33, June, 2005.

[4] J. Y. Jeon and M. Okuma, "Acoustic radiation optimization using the particle swarm optimization algorithm," *JSME International Journal, Series C: Mechanical Systems, Machine Elements and Manufacturing*, vol. 47, pp. 560-567, 2004.

[5] B. A. Kadrovach and G. B. Lamont, "A particle swarm model for swarm-based networked sensor systems," *Proceedings of the 2002 ACM Symposium on Applied Computing*, pp. 918-924 , Madrid, Spain, March 2002.

[6] D. W. van der Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization," *Proceedings of the 2003 Congress on Evolutionary Computation*, Canberra, pp. 215-220, ACT, Australia, December 2003.

[7] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 58-73, 2002.

[8] A. Carlisle and G. Dozier, "Adapting particle swarm optimization to dynamic environments," *Proceedings of the International Conference on Artificial Intelligence*, pp. 429-433, Las Vegas, NV, USA, 2000.

[9] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization," *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 1951-1957, Washington, DC, USA, 1999.

[10] R. C. Eberhart and S. Yuhui, "Tracking and optimizing dynamic systems with particle swarms," *Proceedings of the 2001 Congress on Evolutionary Computation*, pp94-100, Seoul, South Korea, 2001.

[11] A. Carlisle and G. Dozler, "Tracking changing extrema with adaptive particle swarm optimizer," *Proceedings of the 2002 Soft Computing, Multimedia Biomedicine, Image Processing and Financial Engineering*, pp. 265-270, Orlando, FL, USA, 2002.

[12] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," *Proceedings of the 2004 Applications of Evolutionary Computing Workshops*, pp. 489-500, Coimbra, Portugal, 2004.

[13] R. W. Morrison and K. A. De Jong, "A test problem generator for non-stationary environments," *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 2047-2053, Washington, DC, USA, 1999.

[14] K. E. Parsopoulos and M. N. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Computing*, vol. 1, pp. 235-306, 2002.

[15] X. Cui, C. T. Hardin, R. K. Ragade, T. E. Potok, and A. S. Elmaghraby, "Tracking non-stationary optimal solution by particle swarm optimizer," *Proceedings. 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/ Distributed Computing*, pp. 23-25, Towson, MD, USA, May 2005.