Tracking non-Stationary Optimal Solution by Particle Swarm Optimizer

X. Cui[1], C. T. Hardin[2], R. K. Ragade[2], T. E. Potok[1] and A. S. Elmaghraby[2]

[1]Applied Software Engineering Research
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6085
[2]Computer Engineering and Computer Science Department
University of Louisville
Louisville, KY 40292
cuix, potokte@ornl.gov
cthard01,rkraga01,adel@Louisville.edu

Abstract

In the real world, we have to frequently deal with searching for and tracking an optimal solution in a dynamic environment. This demands that the algorithm not only find the optimal solution but also track the trajectory of the solution in a dynamic environment. Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique, which can find an optimal, or near optimal, solution to a numerical and qualitative problem. However, the traditional PSO algorithm lacks the ability to track the optimal solution in a dynamic environment. In this paper, we present a modified PSO algorithm that can be used for tracking a non-stationary optimal solution in a dynamically changing environment.

1. Introduction

A dynamically changing solution space presents a challenge in tracking an optimal solution. Because of the continual changing of both the external environment and parameters, the optimum solution in the environment will also change with time. This demands that the optimal algorithm not only can find the solution in short time but also track the trajectory of the optimal solution in a dynamic environment. Particle Swarm Optimization (PSO) [8] has been proven to be both effective and quick to solve a diverse set of optimization problems [9]. In the past several years, PSO has been successfully applied in many research and application areas [4, 7, 11]. It has been demonstrated that using PSO got better results in a faster, cheaper way than using other methods [6, 12]. However, the traditional PSO algorithm lacks the ability to track the optimal solution in a dynamic environment. The PSO algorithm does not have the mechanism to respond to the environment change. In this paper, we propose a Tracking Dynamical PSO (TDPSO), which is a modified PSO that can be used for searching and tracking the non-stationary optimal solution in a dynamically changing environment.

The remainder of this paper is organized as follows: In section 2, a brief overview of PSO is presented. A discussion of the shortcomings of the original PSO in a dynamic environment is presented in section 3. Various possible modified PSO approaches are also presented in Section 3. In section 4, the structure and algorithms used in our research are described in detail. The setup and results of the experiments for comparing the performance of TDPSO and PSO in dynamical environment are illustrated in section 5. Conclusion and future work are in section 6.

2. Background

2.1 PSO algorithm

PSO was originally developed by Eberhart and Kennedy in 1995 [8], inspired by the social behavior of the bird flock. In the PSO algorithm, the 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 will be altered at each generation of movement. It is this 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 [6, 8]:

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

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

where, w is the inertia weight; p_{id} is the location of the particle that experiences the best fitness value; p_{gd} is the location of the particles in the population experienced the highest best fitness value; c_1 and c_2 are acceleration constants; d is the dimension of the problem space; $rand_1$, $rand_2$ are random values in the range of (0,1).

Equation 1a requires each particle to record its current coordinate X_{id}, its velocity V_{id} that indicates the speed of its movement along the dimensions in a problem space, the coordinates P_{id} *and* P_{gd} where the best fitness values were computed. The best fitness values are updated at each generation based on equation 2, 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.

$$P_i(t+1) = \begin{cases} P_i(t) & f(X_i(t+1)) \le f(X_i(t)) \\ X_i(t+1) & f(X_i(t+1)) > f(X_i(t)) \end{cases} \quad (2)$$

Each particle's structure in a PSO algorithm is relatively simple and only has limited amount of memory. It merely stores the personal best fitness value location vector P_{id} , the global best fitness value location vector P_{gd} and the fitness values $f(P_{id})$ and $f(P_{gd})$. We consider these stored values as particle's experience or knowledge. Equation 2 is the particle's knowledge updating mechanism. In PSO, the knowledge will not be updated until this particle encounters a new vector location with a higher fitness value than the value currently stored in the particle's memory.

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 renew the particles' memory when the environment changed. That causes the particle to continue using the obsolete knowledge to direct its search, which inhibits the particle from following the path of the current optimal solution and results the particle to be easily trapped in the region of the former optimal solution.

2.2. Related work on PSO in the dynamical environment

To solve the memory renewal problem in a dynamic environment, Carlisle proposed the adaptive particle swarm optimization (APSO) [3]. By periodically resetting all particles' memory and replacing their best fitness value and location vector with the particles' current location vector and fitness value, APSO forces the particles to "forget" their former experience. Eberhart and his colleagues published the same idea in [9]. The major disadvantage of APSO is the difficulties in determining the reset frequency. Without prior-knowledge about the environment changing frequency, the particle's memory reset frequency needs to be set to a high value for capturing the changing step of the environment.

However, the high reset frequency will reduce the efficiency of the convergence of 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 will cause all particles to lose their knowledge and restart learning. This decreases the search efficiency of the swarm. Especially during the initial period of searching, frequently resetting the personal best vector will render particles unable to quickly converge on the optimal solution.

Another approach was also proposed by Carlisle in 2001 [5]. Carlisle introduced a new notion, "sentry" in his APSO algorithm. The "sentry" is one or many special designed particles, which are deployed in the problem space and used for monitoring the environment changing. When the "sentry" detects a change in the environment, it will inform all others and force other particles to reset their memory. However, designing a particle as a sentry to monitor the environment will increase the complexity of the whole system. In addition, this algorithm changes the classical PSO's ideally decentralized processing model as an essentially centralized control model and reduces the robustness of the modified PSO.

3. Tracking Dynamic PSO Approach

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. In this research, we propose a new modified PSO, the tracking dynamic PSO approach (TDPSO) to satisfy these requirements. In TDPSO, there is no specially designed particle to monitor the environment. Same as the traditional PSO, each particle uses the equation 1 to determine its next velocity. The only modification is the particle's best fitness value update mechanism. Instead of using equation 2 to update the fitness value, we use equation 3 for the fitness value update.

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

In equation 3, a new notion, the evaporation constant ***T***, is introduced. ***T*** has a value between 0 and 1. The personal fitness value and global fitness value that stored in each particle's memory will gradually evaporate (decrease) at the rate of the evaporation constant over time. After the value of fitness evaporates for a period, the fitness value, X-fitness, of the current location may be higher than the evaporated fitness values and will replace the old fitness value. Although all particles have the same evaporation constant ***T***, each particle's update frequency may not same. Depending on the particle's current stored best fitness value ***f(P)*** and the current fitness value ***f(X)*** the particle acquired, the particle will more frequently update the best fitness value by using its 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 may indicate the particle's current location is farther 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. In this situation, it would be better to keep the best fitness value in the particle's memory until after several generations passed and the best fitness value has become too obsolete. The fitness value update equation enables each particle to self-adapt to the changing environment.

4. Experimental Implementations

4.1. Dynamical environment simulation

The simulated dynamic environment can be constructed by starting with a simple parabolic function [1] as described in equation 4. This equation has been used to simulate the dynamic environment in [1, 3, 5].

$$f(x) = \sum_{i=1}^{n} x_i^2 \tag{4}$$

In the equation 4, ***n*** is the dimension number of the problem space and ***f*** can be considered as the fitness value evaluation function. The optimal point (solution) is a vector **(0, …., 0)**. Based on this parabolic function, an environment with a dynamically changed optimal solution can be generated by adding an offset on $\boldsymbol{x_i}$ in each dimension. Equation 5 represents this dynamic environment, where $\boldsymbol{s_i}$ is the offset in dimension ***i***.

$$f(x) = \sum_{i=1}^{n} (x_i + s_i)^2 \tag{5}$$

Different offset movement functions generate different types of dynamic environments. In this research, to implement a randomly moving optimal solution, the update function of $\boldsymbol{s_i}$ is described in equation 6:

$$s_i(t+1) = s_i(t) + v_k * rand(0,1) \tag{6}$$

where $\boldsymbol{v_k}$ is a constant represent the speed of the offset and ***rand(0,1)*** is a Gaussian random function.

Increasing dimensionality of the solution space would increase the systematic complexity level. To simplify the simulation and improve the experimental speed, we choose the three dimensions parabolic function as the base environment function in our experiments. The three dimension function has a minimum point at (0,0,0) as the optimal solution. The moving speed $\boldsymbol{v_k}$ of the offset are set to ***0.02*MAX, 0.05*MAX,*** and ***0.1*MAX***, respectively in the simulation. The "***MAX***" indicates the maximum velocity of the particles in the simulation.

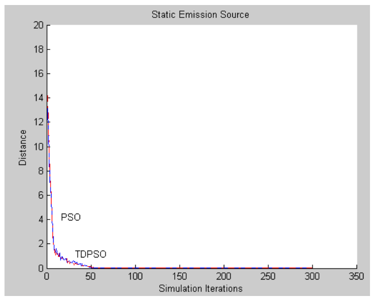
4.2. Experiment Setup and Results

To evaluate the efficiency of the TDPSO in the dynamic environment and to determine the best evaporation constant, two experiments are performed. The first experiment is to compare the dynamic tracking ability of the TDPSO with PSO. The second is to investigate the influence of the evaporation constant on the performance of TDPSO.

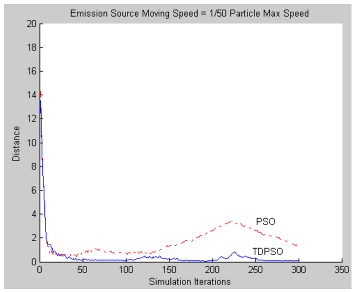
In the first experiment, TDPSO and PSO have the same configuration and are tested in the same dynamic environment. In the TDPSO algorithm, the particles use equation 3 to update their best fitness value and the evaporation constant ***T*** is set as ***exp(-1)*** according to [2]. Twenty particles are randomly distributed in an environment with ±20 wide in each dimension. According to Engerlbrecht's work, in equition 1a and 1b, c_1 and c_2 are set to 1.49 and V_{max} is set to 0.5 [10]. The ***w*** value is set to 0.72 [4, 12]. TDPSO and PSO are applied on both the static environment and a dynamic environment with a randomly moving optimal solution. For evaluating the performance of the tracking ability of each algorithm in the dynamical environment, instead of using the distance between the optimal solution and the particle that has the highest fitness evaluation value, we choose the sum distance d_{sum} between all particles and optimal solution point as the algorithm evaluation value. The summed value demonstrates the tracking ability of algorithm in the entire searching procedure. If this value is small, the particles can keep themselves in a short distance from the goal at anytime, regardless of the goal's movement.

The results of the first experiment are illustrated in Figure 1, which shows the performance results of searching and tracking of TDPSO and PSO in both the static and dynamic environments. The distance summation from each particle to the optimal solution at each generation is used to evaluate the performance of the particle system. The smaller this value, the closer all particles to the optimal solution are.

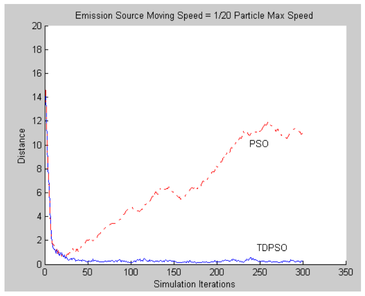
As shown in the figure 1 (a), both TDPSO and PSO perform efficiently in the static environment. It takes nearly fifty generations for all particles to converge in the region of the goal. However, in the dynamic environment, the traditional PSO fails to track the randomly moving optimal solution. As shown in figure 1 (b) (c) (d), the traditional PSO cannot track the movement of the goal and all particles are trapped at the 40th generation. However, TDPSO can quickly converge on the optimal solution and maintain the shortest distance from the optimal solution.



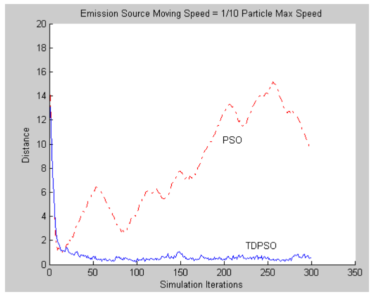
(a)



(b)



(c)



(d)

Figure 1 Tracking performance comparison (a) Static environment; (b) Offset moving speed = 0.02 * max; (c) Offset moving speed = 0.05 * max; (d) Offset moving speed = 0.1 * max

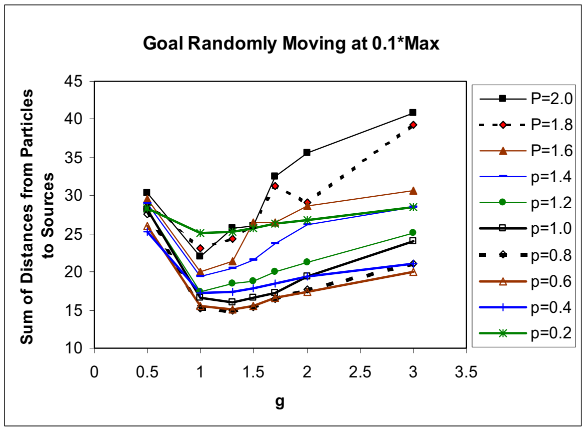
In the second experiment, the impacts of different evaporation constants of the personal fitness value and the global fitness value on the TDPSO algorithm's performance are investigated. Different pairs of evaporation constants for personal fitness value and global fitness value are tested in the simulations. The sum distance between all particles and the optimal solution are recorded at each generation.

To easily draw a 2D graph to display the performance comparing of the algorithm on different evaporation constants, we use the summation value of the distances of the whole generation during the simulation as the evaluation for the performance. To reduce the impact of the initial position of the particles on the summary value, the summed value only sums the distance data between the 30th generation and the 200th generation. Figure 2 illustrates the d_{sum} values of different pairs of evaporation constants when the optimal goal moves randomly in the environment. From the graphs, we can conclude that: the optimal value of the personal fitness value evaporation constant Tp ranges between ***exp(-0.6)*** and ***exp(-0.8)***. The optimal value of personal fitness value evaporation constant Tg ranges between ***exp(-1)*** and ***exp(-1.3)***.

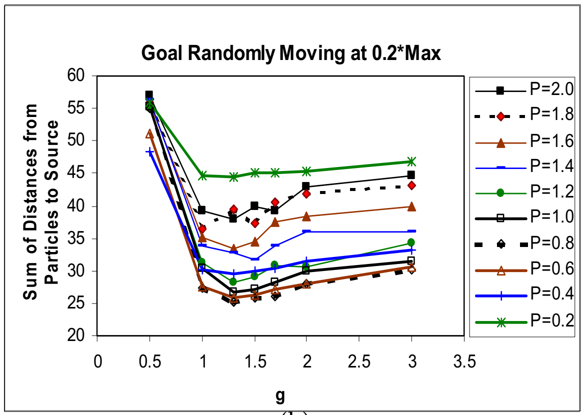
5. Discussion and Conclusion

Most papers reporting applications of optimization algorithms only discuss the scenario of a static environment [7, 10, 11, 12]. The performance evaluation of various approaches is mainly based on how fast an approach can find the optimal point in the benchmark problems. PSO has proven to be very effective in applications with static environment. However, in the real world, a frequently changing solution space causes the optimal solutions to change over time. The optimal solution found at time T_1 may no longer valid at time 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 their progression through the solution space as closely as possible. The traditional PSO has a difficulty to track the 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 more recent goal.

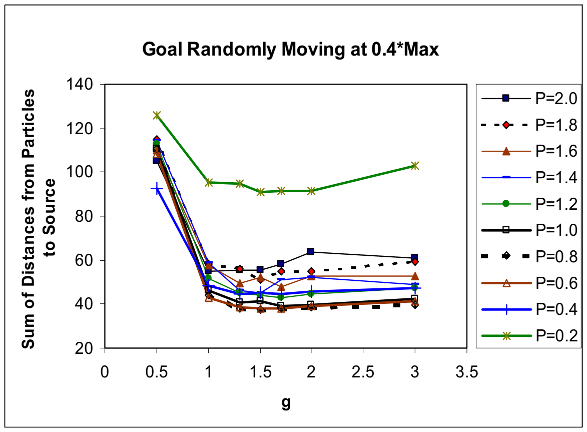
In this paper, we present a new approach, TDPSO, a modified PSO, for tracking the optimization solution in a dynamically changing environment. Unlike other modified PSO for the dynamical environment, which needs one or more sentry particles to control other particles' action, each particle in TDPSO individually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



(a)



(b)



(c)

Figure 2 Tracking performance comparisons of different pair of evaporation constant. (a) Offset moving speed = 0.1 * max; (b) Offset moving speed = 0.2 * max; (c) Offset moving speed = 0.4 * max

Each particle's memory has an evaporation constant to control the rate of the knowledge evaporation (decay). The updating frequency of the particle's memory is determined by the knowledge evaporation rate and the current fitness value the particle perceived from the point it located. Each particle in the system may update its memory in different generations.

The simulation experiments indicate that TDPSO can efficiently track the movement of an optimal solution in a dynamically changing environment, while the traditional PSO failed. Because each particle updates its memory only based on its perception and the knowledge evaporation rate, the system can avoid losing tracking of the optimal solution as happened in other modified PSO approaches that are based on resetting the memory periodically.

We also illustrated the impact of different evaporation constant parameters on the performance of tracking the dynamically changed optimal solution in TDPSO. In the simulation where the optimal solution moves at varying velocities, TDPSO would have a better performance if the evaporation constant of the personal best value and global best value are set in the range of ***(exp(-0.6), exp(-0.8))*** and ***(exp(-1), exp(-1.3),*** separately. An alternative solution to the fixed evaporation rate is to use reinforce learning. If each particle can implement the reinforcement mechanism for dynamically adjusting the best fitness value's evaporation rate, the entire system may have better performance than setting a fixed evaporation rate for all particles. However, that will increase the design complexity of each particle.

6. References

[1] Angeline, P. (1997). Tracking Extrema in Dynamic Environments, Proceedings of the Sixth Annual Conference on Evolutionary Programming VI, Indianapolis, IN, USA. pp. 335-345

[2] Benjafield, J. (1992). Cognition, Prentice Hall, 1992, Englewood Cliffs, New Jersey, USA

[3] Carlisle, A., and Dozier, G., (2000). Adapting particle swarm optimization to dynamic environments. In Int. Conf. on Artificial Intelligence, Monte Carlo Motel, Las Vegas, NV, USA

[4] Carlisle, A. and Dozier, G., (2001a). An Off-The-Shelf PSO, Proceedings of the 2001 Workshop on Particle Swarm Optimization, Indianapolis, IN, USA

[5] Carlisle, A., and Dozier, G. (2001b). Tracking changing extrema with particle swarm optimizer. Tech. Rep. CSSE01-08, Auburn University, Auburn, AL, USA

[6] Clerc, M., (1999). The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. Proceedings, ICEC, Washington, DC, USA

[7] Cui X., Hardin T., Ragade R. K., and Elmaghraby A. S., A Swarm Approach for Emission Sources Localization, The 16th IEEE International Conference on Tools with Artificial Intelligence, 2004, Boca Raton, Florida, USA

[8] Eberhart, R. C., and Kennedy, J., (1995). A new optimizer using particle swarm theory. In Sixth International Symposium on Micro Machine and Human Science, agoya, Japan, IEEE Press

[9] Eberhart R.C. and Shi Y., (2001a). Tracking and optimizing dynamic systems with particle swarms. Proc CEC 2001. Piscataway, NJ, USA

[10] Engelbrecht A. P., (2002). Computational Intelligence, John Wiley & Sons Ltd, 2002, Chichester, West Sussex, England

[11] Hardin T., Cui X., Ragade R. K., Graham J. H., and Elmaghraby A. S., (2004). A Modified Particle Swarm Algorithm for Robotic Mapping of Hazardous Environments, The 2004 World Automation Congress, SEVILLE, Spain

[12] Shi, Y. H., Eberhart, R. C., (1998). Parameter Selection in Particle Swarm Optimization, The 7th Annual Conference on Evolutionary Programming, San Diego, USA