ELEKTROTEHNIŠKI VESTNIK 79(1-2): 68-72, 2012
ENGLISH EDITION

A Comparative Study of Fuzzy–PSO and Chaos–PSO

O. Tolga Altinoz[1], S. Gökhun Tanyer[2], A. Egemen Yilmaz[3]

[1,3]Ankara University, Electronics Engineering Department, Tandıgan Ankara,Turkey
[2]TUBITAK, Kavaklıdere, Ankara, Turkey
E-mail: taltinoz@ankara.edu.tr, gokhun.tanyer@gmail.com, asim.egemen.yilmaz@eng.ankara.edu.tr

Abstract. Two popular particle swarm optimization (PSO) formulations; fuzzy–PSO (FPSO) and chaos–PSO (CPSO) have previously been studied in the literature. The charisma factor in FPSO gives the ability to track the particles which are closest to the optimum. CPSO has been aimed to search the area by using the chaotic maps. These two different algorithms are shown to demonstrate sufficient performance independently. Unfortunately, comparisons of their performances are currently unavailable. In this study, both FPSO and CPSO algorithms are examined. The underlying mechanisms and motivations for these methods are discussed. The performances for FPSO and CPSO are compared. The results are presented using benchmark functions.

Keywords: particle swarm optimization, logistic map, chaos, fuzzy logic.

1 INTRODUCTION

Inspired from the social and cognitive behaviors of animals living as swarms, particle swarm optimization (PSO) provides a simple but very powerful tool for researchers who are dealing with collective intelligence. This optimization algorithm is shown to demonstrate an outstanding performance for complex problems. The algorithm depends on modeling the very basic random behavior (exploration capability) of the individuals in addition to their tendency to revisit positions of good memories (cognitive behavior) and the tendency to keep an eye on and follow the majority of the swarm members (social behavior) as illustrated in Fig. 1.

The balance among these three major approaches is the key to success of the algorithm. But still, occasionally this algorithm is observed to have problems such as getting stuck at local optima and stagnation at the application of the PSO algorithms (ex. Controller parameter tuning mechanical and antenna design). Thus, in order to avoid these problems, various improvements have been proposed. The hybrid particle swarm optimization algorithms have recently gained attention due to this particular reasoning.

2 CONVENTIONAL PARTICLE SWARM OPTIMIZATION (PSO)

Swarm intelligence constitutes a very significant portion of the literature regarding nature inspired methods. The term "swarm intelligence" has been a major multidisciplinary attraction center for researchers dealing especially with complicated inverse (e.g. design and synthesis) problems since its introduction by Beni and Wang [1] with the context of cellular robotic systems. Typically, a swarm intelligence system consists of a population with members having some characteristic behavior and local interaction with each other. In this system, the individual members have freedom to a certain extent to interact with each other. These interactions yield a global behavior even though there is no dictating centralized mechanism. This global behaviour is much more organized and directive than a stand-alone individual.

Today, two main algorithms dominate the "swarm intelligence" approach. These are the Ant Colony Optimization (ACO) which is proposed in 1992 by Dorigo [2], later formalized by Dorigo, Di-Caro and Gambardella [3], and the Particle Swarm Optimization (PSO) which was proposed by Kennedy and Eberhart [4]. Both algorithms were developed by observing the behaviour of animals living as swarms/colonies and getting inspired by them, and for more than a decade, they proved to be successful in solving various complex problems. Originally, ACO was designed for combinatorial optimization problems; whereas PSO was designed for continuous ones. Later, successful versions of continuous ACO [5] and discrete PSO [6] have also been developed.

Very similar to the genetic algorithm (GA), PSO algorithm is initialized with a population of random solutions. However, unlike GA each potential solution is also assigned a randomized velocity and the potential solutions. These solutions are called particles, and are “flown” through the problem space. When compared with the GA, PSO has a simpler concept, easier implementation and faster convergence. PSO has gained

Received March 31, 2012
Accepted April 4, 2012

much attention on wide applications in various fields, especially during the last decade.

Existing innovations on the PSO algorithm are accomplished by the variation of PSO mechanism itself, both in mathematics and topology and also by taking advantage of other optimization techniques, such as GA [7]. Moreover, the innovations on the PSO algorithm can be categorized into five groups; exploration behaviour adjustment, searching area adjustment, parameter adaptation, neighbourhood topologies and hybrid methods. The behavior of the swarm is based on local and global exploration of particles. The exploration of the swarm can be controlled via algorithm parameters. Thus, various topologies result from the improvement on the PSO algorithm. Hybrid methods utilize the combinations of different algorithms such as the PSO-GA structure.

2.1 Fundamentals of Particle Swarm Optimization

The PSO algorithm depends on motions of particles (swarm members) searching for the global best in N-dimensional continuous space. The position of each particle is simply a solution candidate, and at each time step the fitness of this candidate is re-evaluated. In addition to its exploration capability (tendency for random search throughout the domain), each particle has a cognitive behavior (remembering its own good memories and having the tendency to return there); as well as a social behavior (observing the rest of the swarm and having the tendency to go where most other particles go).

The original PSO formulation of Kennedy and Eberhart [4] depends on the update of the position x_i and the velocity v_i of the i'th particle (swarm member) and is as follows.

$$v_{id} = v_{id} + c_1 * rand() * (p_{id} - x_{id}) + c_2 * rand() * (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id}.\Delta t \quad (2)$$

where c_1 and c_2 are measures indicating the tendencies of approaching to *pbest* and *gbest* which are the best positions achieved personally by the i'th particle and the whole swarm, respectively. In other words, c_1 and c_2 are the measures of the cognitive and the social behaviors (called cognitive and social parameters respectively), respectively. In earlier PSO studies, the values of these parameters were set to 2.0 whereas for recent PSO studies 1.494 is observed to be the most preferred value for c_1 and c_2. *rand()* is a random number between 0.0 and 1.0, and the time step size Δt is usually selected to be unity for simplicity.

2.2 PSO Algorithm

```
Initialize random velocity and position
Do
  For i = 1 to swarm size
      Calculate fitness function (fit) which is the
minimization function
      If fit < best pattern
          ith particle best position = ith particle position
          best pattern = fit
      end if
  end for
  find min best pattern and corresponding particle
  For i = 1 to swarm size
      Update velocity
      Update position
  end for
  Check the limits of the maximum velocity
while break if maximum iterations or minimum error
```

The methodology of the PSO algorithm can be divided into two different categories based on the information sharing methods which are called "*gbest*" and "*lbest*" methods.

"gbest" Method:
Each particle is moved to the optimal value. Thus, the information related to positioning must be shared between the particles. In "*gbest*" (global best) method, the information related to each member of the swarm is shared with all swarm numbers.

"lbest" Method:
In "*lbest*" method (local best), each of the particle positions are shared only with the neighbourhood particles. Thus, a topology for neighbourhood should be defined.

3 FUZZY – PARTICLE SWARM OPTIMIZATION

In conventional PSO algorithm, the influence of the best particle of the swarm is applied as the third additive parameter in the formulation in (1). The best particle in the swarm is applied to change the trajectory of each particle as given in Fig. 1. In this manner, the particle moves to the direction of the best particle position and best swarm position. More than one best solution can be found if more than one particle is utilized. Abdelbar et al. [8] used a group of best particles (instead of a single best) in the PSO formulation. Each group member is assigned with the multiplier which is called the charisma factor. The influence of each best particle to others is calculated using this charisma value. The sum of these influences is applied to the original formulation. The FPSO formulation is given in (3).

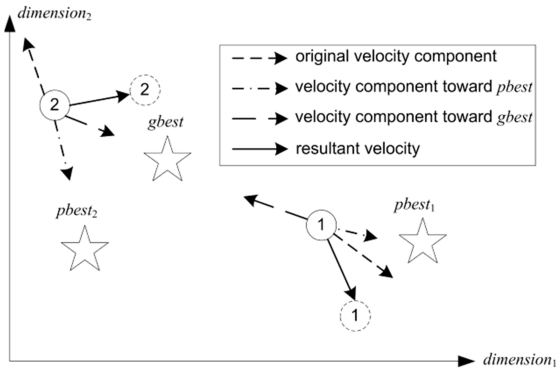

Figure 1. Change of the particle position.

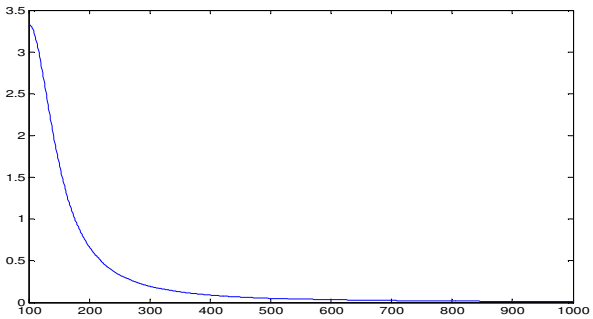

Figure 2. Example of the Cauchy distribution.

$$v_{id} = v_{id} + c_1 * rand() * (p_{id} - x_{id}) + \sum_{h=1}^{k} c_2 * rand() * \Psi(p_{hd})(p_{hd} - x_{id}) \quad (3)$$

where $\Psi(p_{hd})$ is defined as charisma function. This function is a membership function which promotes the name; Fuzzy PSO. Abdelbar et al. [8] used the Cauchy distribution as the charisma function since the growth of the corresponding values are further away from the centre which means the Cauchy has a "long tail". In summary, the effects of the particles with longer distance from the global best position have smaller effects. In (4), the charisma function is defined, and in Fig. 2 the behaviour of this function is presented.

$$\Psi(p_h) = \frac{1}{1 + \left(\ell \frac{f(p_h) - f(p_g)}{f(p_g)} \right)^2} c \quad (4)$$

where $f(.)$ is the fitness function. Abdelbar et al. [8] applied the parameter values as $c_1 = 0.5$, $c_2 = 0.6$, $k = 3$ and $l = 1.6$. In this study, the same values are used for the comparisons. Abdelbar et al. [8] also proposed the usage of other distribution functions instead of the Cauchy distribution. In this study however, due to its proven performance, the Cauchy distribution is preferred.

3.1 Fuzzy–PSO Algorithm

```
Initialize random velocity and position
Do
  For i = 1 to swarm size
    Calculate fitness function (fit) which is the
    function to be minimized
    If fit < best pattern
      ith particle best position = ith particle position
      best pattern = fit
    end if
  end for
  find the first k min best pattern and corresponding
  particle
  For i = 1 to swarm size
       Find charisma values for k particle
       Update velocity
       Update position
  end for
  Checking the limits of the maximum velocity
```

4 CHAOS – PARTICLE SWARM OPTIMIZATION

Hybrid particle swarm optimization algorithms have gained attention because of the aforementioned drawbacks of the conventional PSO algorithm. As a novel improvement, chaos based approaches are also being applied to the PSO recently via the chaotic maps.

The chaotic map can be helpful to escape from a local minimum [9], and it can also improve the global/local searching capabilities. Chaos is applied to PSO in various ways, and can be generally categorized into two methods. In the first method (FM), the chaos map is used against the random number generator. Chaos is used; to control the values of the parameters in velocity update formulation, to determine the inertia weight coefficient *w*, and to generate the *c*rand()* coefficients. In the second method (SM), chaos is used to interact with the PSO algorithm in search of the solution space. Results in the literature demonstrate that the second method is better [10].

4.1 Chaotic Maps

The discrete-time dynamical system in the iteration form in (5) is called mapping or simply a map.

$$x_{i+1} = F(x_i, P) \quad (5)$$

where P is the control parameter, x is a vector and F is a nonlinear transformation [11]. The chaotic map is applied for each dimension of the problem space and for each particle when SM is preferred. Thus, a time requirement for SM depends on the dimension and the population of the PSO. One-dimensional maps include chaotic behaviour. These maps have been used in many applications due to their simplicities. The Logistic Map which is a model of population biology, is frequently

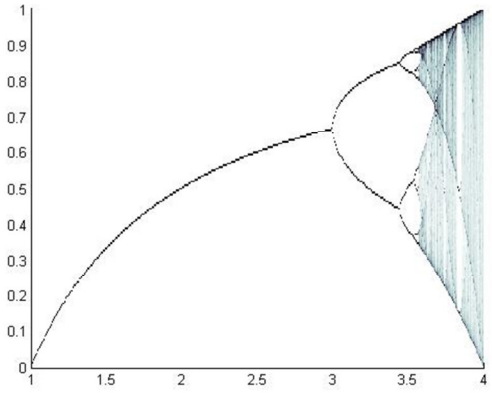

Figure 3. Bifurcation diagram of the Logistic Map.

used with PSO [12-14]. This map is a unimodal map of finite interval and single maximum as given below.

$$x_{i+1} = \mu\, x_i(1 - x_i) \tag{6}$$

where μ is set to 4 for ergodicity. The bifurcation diagram of the Logistic Map is illustrated in Fig. 3.

4.2 Variable Mapping

For chaotic search, the initial value of the chaotic map must be determined. The easiest way is to determine the initial value as a random number in [0,1]. The most common way to find the initial value is to define the carrier method by defining a decision variable from PSO variables, and to map this variable into the chaotic domain by using the carrier equation as defined in (7) [10].

$$cx_i = \frac{x - X_{\min}}{X_{\max} - X_{\min}} \tag{7}$$

where cx_i is decision variable which is the initial value of the chaotic map, x is the position of the particle and X_{min} and X_{max} are the boundaries of the position.

4.3 Inverse Variable Mapping

After determination of the chaotic variable from chaotic map with the initial value from variable mapping, the chaotic variable is converted into the particle velocity using (8).

$$x = X_{\min} + cx_i(X_{\max} - X_{\min}) \tag{8}$$

4.4 Search Space

If the search space extends to a wide area the search operation cannot be completed in a reasonable amount of time [13]. Thus, in order to obtain high performance the chaotic search is run in a small range. This search area is changed in the current optimal solution neighbourhood [15] as given in (9) and (10).

$$X_{\min} = \max\left[X_{\min},\ x_g - r(X_{\max} - X_{\min})\right] \tag{9}$$

$$X_{\max} = \min\left[X_{\max},\ x_g + r(X_{\max} + X_{\min})\right] \tag{10}$$

where x_g is the global best position and r is the variable defined in [0,1]. The search efficiency is expected to increase if the range of the chaotic search is decreased.

4.5 Chaos–PSO Algorithm

```
Initialize random velocity and position
Do
  For i = 1 to swarm size
     Calculate fitness function (fit) which is the function to
     be minimized
     If fit < best pattern
        ith particle best position = ith particle position
        best pattern = fit
      end if
  end for
  find min best pattern and corresponding particle
  For i = 1 to swarm size
    Update velocity
    Update position
  end for
  Checking the limits of the maximum velocity
  For k = 0 to any number
     Execute variable mapping that maps the positions
     into chaos variables
     Use Logistic map and find new chaos variable
     Execute Inverse Variable Mapping that convert
     chaos variable into positions
    If new position < old position
          Position = New Position
          Break
    End if
  end for
  end for
while break if maximum iterations or minimum error
  end for
```

5 SIMULATION AND RESULTS

In this study, two different PSO formulations are compared based on the benchmark functions given in Table 1. Two different types of benchmark functions are used in this study; unimodal and multimodal. A unimodal function has only one optimum whereas a multimodal one has many local optima. As the dimension of the unknown function increases, the performance of the algorithm decreases. Hence, in this study, higher dimensional benchmark functions are used. After the simulations are executed, the results presented in Table 2 are obtained.

Results illustrate that for unimodal functions the FPSO outperforms the CPSO. However, for the multimodal function CPSO proves itself to be much better, especially for its success in avoiding local minima.

Table 1: The three benchmark functions used in our experimental studies where n is the dimension of the function, S is the feasible search space, and $f_{\min}$ is the minimum value of the function.

Test Function	n	S	$f_{\min}$
$f_1(x)=\sum_{i=1}^{n} x_i^{\,2}$	30	$[-100,100]^n$	0
$f_2(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	30	$[-100,100]^n$	0
$f_3(x)=\frac{1}{4000}\sum_{i=1}^{n}(x_i-100)^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i-100}{\sqrt{i}}\right)+1$	30	$[-600,600]^n$	0

Table 2: Comparison of our proposal with the inertial weight PSO formulation (Swarm Size = 25, Number of Iterations = 100, Number of Independent PSO Executions = 100).

	Fuzzy PSO		
Function	**Min.**	**Mean**	**Standard Deviation**
f_1	0	8.6816e-007	8.6816e-006
f_2	0	3.8593e-023	3.8593e-022
f_3	1.0002	1.0002	4.4633e-016
	Chaos PSO		
f_1	4.6697e-056	1.0951e-010	1.0886e-009
f_2	0	5.1018e-021	5.1006e-020
f_3	0	9.0965e-005	5.6348e-004

6 CONCLUSION

In this study, two interesting PSO formulations; FPSO and CPSO are compared via benchmark functions. In FPSO, the parameter called charisma is defined and used as the weighting coefficient of the best position group. Instead of one particular leader particle, several particles are allowed to influence the other particles. In CPSO, the search of the global optimum point is accomplished by using the chaotic search approach. This method is used to ensure that there is no other best solution in the vicinity of the search area.

It is observed that the CPSO has a better performance than the FPSO for multimodal benchmark functions. It is shown that the CPSO is a better candidate for real time applications. However, this method has computational complexity and has the disadvantage of working slower than FPSO.

The definition of the Chaotic Fuzzy PSO algorithm and the comparisons with the PSO formulations available in the literature are planned as future studies.

REFERENCES

[1] G. Beni, J. Wang, "Swarm intelligence in cellular robotic systems," *NATO Advanced Workshop on Robots and Biological Systems*, Tuscany, Italy, 1989.

[2] M. Dorigo, "Optimization, learning and natural algorithms," (in Italian), Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

[3] M. Dorigo, G. DiCaro and L.M. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, 5, pp. 137-172, 1999.

[4] J. Kennedy, R.C. Eberhart, "Particle swarm op-timization," Proc. *IEEE Int. Conf. on Neural Networks*, pp. 1942-1948, 1995.

[5] J. Dréo, P. Siarry, "A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions," *Lecture Notes in Computer Science*, 2463, pp. 216-221, 2002.

[6] J. Kennedy, R.C. Eberhart, "A discrete binary version of the particle swarm algorithm," *Proc. Conference on Systems, Man, and Cybernetics*, pp. 4104-4109, 1997.

[7] X.H. Shi, Y.C. Lianga, H.P. Lee, C. Lu, and L.M. Wang, "An improved GA and a novel PSO-GA-based hybrid algorithm," *Information Processing Letters*, 93(5), pp. 255–261, 2005.

[8] A.M. Abdelbar, S. Abdelbar and D.C. Wunsch, "Fuzzy PSO: A Generalization of Particle Swarm Optimization," *Proc. International Joint Conference on Neural Networks*, pp. 1086-1091 Montrael, Canada, 2005.

[9] L.S. Coelho, V.C. Mariani, "A novel chaotic particle swarm optimization approach using Henon map and implicit filtering local search for economic load dispatch," *Chaos, Solitons and Fractals*, 39, pp. 510-518, 2009.

[10] T.Xiang, X. Liao and K.W. Wang, "An Improved Particle Swarm Optimization Combined with Piecewise Linear Chaotic Map," *Applied Mathematics and Computation*, 190(2), pp. 1637-1645, 2007.

[11] J.M.T. Thompson, H.B. Stewart, Nonlinear Dynamics and Chaos, John Wiley & Sons 2nd Edition, 2002.

[12] B. Liu, L. Wang, Y.H. Jin and C.X. Huang, "Designing Neural Networks Using Hybrid Particle Swarm Optimization," *Lecture Notes in Computer Science*, pp. 3496, 391-397, 2005.

[13] H.J. Meng, P. Zheng, R.Y. Wu, X.J. Hao and Z. Xie, "A hybrid particle swarm algorithm with em-bedded chaotic search," *Proc. IEEE Conference on Cybernetics and Intelligence Systems*, pp. 367-371, 2004.

[14] A.L.Chen, Z.M. Wu and G.K. Yang, "LS-SVM based on chaotic particle swarm optimization with simulated annealing," *Lecture Notes in Computer Science*, 3959, pp. 99-107, 2006.

[15] X.Y. Gao, L.Q. Sun and D.S. Sun, "An enhanced particle swarm optimization algorithm," *Information Technology Journal*, 8, pp. 1263-1268, 2009.

Okkes Tolga Altinoz received his B.Sc. and M.Sc degrees in Electrical and Electronics Engineering from Başkent University in 2003 and 2009, respectively. His research interests include optimization algorithms, soft computing methods, control systems and dynamics.

Süleyman Gökhun Tanyer received the B. Sc. and the M. Sc. in electrical engineering from Middle East Technical University and Bilkent University, Ankara, Turkey in 1988 and 1990 respectively, and the Ph. D. in electrical engineering from Washington State University, Pullman, WA, United States in 1994. His research interests include electromagnetic wave propagation and scattering, radar signal processing and statistical signal processing.

Asim Egemen Yilmaz received his B.Sc. degrees in Electrical-Electronics Engineering and Mathematics from the Middle East Technical University in 1997. He received his M.Sc. and Ph.D. degrees in Electrical-Electronics Engineering from the same university in 2000 and 2007, respectively. His research interests include computational electromagnetics, nature-inspired optimization algorithms, knowledge-based systems, software development processes and methodologies.