Animal Food Chain Based Particle Swarm Optimization

Ayca Altay and Gulgun Kayakutlu

Abstract— Particle Swarm Optimization has been an appealing research area for researchers for over 15 years. During these years, a variety of algorithms have been developed around the particle swarm concept. One of these variations is Predator-Prey Particle Swarm Optimization algorithm which is also based on natural swarms which have a hierarchical relationship. Hunting search has also become a new meta-heuristic originating from hunting structure of species. In this paper, a new Particle Swarm Optimization Algorithm has been developed. Animal Food Chain Based Particle Swarm Optimization Algorithm simulates the animal food chain structure in three swarms: omnivores, carnivores and herbivores, in order to balance exploration and exploitation.

Index Terms— Animal Food Chain Based Particle Swarm, Hunting Search, Particle Swarm Optimization, Predator Prey Particle Swarm Optimization,

I. Introduction

Particle Swarm Optimization (PSO) is a population based optimization technique invented by Kennedy and Eberhart in 1995 influenced by the social behavior of fish schooling and bird flocking [1]. It simulates the "collective behavior" of the animals which socio-cognitively share information among the swarms [2].

Animals in nature, urge into swarms for different objectives: finding food, escaping predators, etc… These swarms, also called boids (a special name for bird swarms), have three vital principles for their collaborative movements: "collision avoidance – separation", "velocity matching – alignment" and "flock centering – cohesion" [3]. Collision avoidance refers to not crashing with nearby elements of the swarm, whereas velocity matching refers to adapt velocities according to the velocities of neighboring elements. The flock centering is a result of velocity matching. Since each flock-mate tends to adapt his velocity according to his neighbors, the flock tends to stay close to each other. Generalizing it to the whole swarm, they tend to stay close to a neighborhood center (Veenhuis and Köppen, 2006). Apart from these three principles, environmental principles such as obstacle avoidance and following a desired path are also valid [4-5].

Imitating the aforementioned principles, the Particle Swarm Optimization field has gained interest because of its applicability, simplicity and efficiency. The algorithm is proven to be robust and effective in various types of problems from single objective to combinatorial [6]. However, majority of algorithms developed are focused on exploring. This study aims to develop an algorithm that is required to balance the exploitation and exploration functions. This paper is so organized that, the next part is reserved for summarizing the basic PSO algorithm in detail. In the third section, the most recent approaches for PSO are presented: Predator-Prey and the Hunting Search. In the fifth part, Animal Food Chain Based PSO, our development study will be presented. The final section will include the conclusions.

.

Ayca Altay is with the Istanbul Technical University Industrial Engineering Department, Macka, İstanbul, 34367 Turkey (e-mail: altaya@ itu.edu.tr).

Gulgun is with the Istanbul Technical University Industrial Engineering Department, Macka, İstanbul, 34367 Turkey (corresponding author; phone : +905322127734, e-mail: kayakutlu@itu.edu.tr).

II. Particle Swarm Optimization

A. Steps of the Algorithm

In the PSO algorithm, each solution is presented with a particle, that is, an element of the swarm. The swarm consists of a number of particles, in other words, a population of solutions [6]. The particles move through the search space at a random degree of freedom limited by the parameters of each other [7]. The moving particles have two properties, position and velocity, which are updated at every iteration of the algorithm [8]. Each particle is intelligent in a way that it keeps the memory of best position of himself and the neighborhood, which introduces the neighborhood concept. In the ultimate condition, generally the neighborhood denotes the whole swarm (Léon-Javier et. al., 2009).

Let p_i be the position of the ith particle in the swarm which consists of N particles and let each particle have n dimensions defined over a maximization objective function f. The steps of the algorithm is given below [9]

Step 1. Initiating particle velocity and position of each p_i such that

$$x_{i,j} = x_{min} + r(x_{max} - x_{min}), \qquad i = 1, \dots, N \qquad j = 1, \dots, n$$

$$v_{i,j} = \alpha \frac{x_{min} + r(x_{max} - x_{min})}{\Delta t} \qquad i = 1, \dots, N \qquad j = 1, \dots, n$$

where x denotes the position, v denotes the velocity and α is constant in the range [0,1].

Step 2. The objective value of each particle is calculated as $f(x_i)$

Step 3. The best position for each particle and the global best position for the swarm is updated. For a problem

If $f(x_i) < f(x_i^{pb})$ then $x_i^{pb} \leftarrow x_i$

If $f(x_i) < f(x_i^{sb})$ then $x_i^{sb} \leftarrow x_i$

where *pb* denotes the particle best and *sb* denotes the swarm best.

Step 4. Particle velocity and particle position are updated, that is, the new velocities and positions are calculated for each particle.

$$v_{i,j} \leftarrow wv_{i,j} + c_1 r_1 \left(\frac{x_{i,j}^{pb} - x_{i,j}}{\Delta t}\right) + c_2 r_2 \left(\frac{x_{i,j}^{sb} - x_{i,j}}{\Delta t}\right) \qquad i = 1, \dots, N \qquad j = 1, \dots, n$$

$$x_{i,j} \leftarrow x_{i,j} + v_{i,j}\Delta t \qquad i = 1, \dots, N \qquad j = 1, \dots, n$$

where w is the inertia rate between [0,1], r_1 and r_2 are random numbers between [0,1]. In the velocity update formula, $v_{i,j}$ is the inertia term where the particle attempts to save its own velocity, $c_1 q \left(\frac{x_{i,j}^{pb} - x_{i,j}}{\Delta t}\right)$ is the cognitive term where the particle attempts to reach at least its best position and $c_2 r \left(\frac{x_{i,j}^{sb} - x_{i,j}}{\Delta t}\right)$ is the social term where the particle attempts to keep up with the best position of the swarm.

Step 5. Step 2 is returned until a termination criterion is satisfied. Various termination criteria include iteration numbers, convergence of the result, convergence of error in results, etc…

B. Parameters of the Algorithm

Particle Swarm Optimization algorithm provides a number of parameters to be tuned. The effects of these parameters are presented below:

- *The Inertia Coefficient – w*: The inertia term is generated from the Newton's law that all particles tend to save their velocities unless an outside force is applied. If $w = 0$, then the velocity update formula is called selfless [6]. It is also necessary for monitoring the exploration degree of the algorithm. Assigning a high value for w would yield a higher speed, thus more exploration than exploitation. Assigning a lower value, would yield a better exploitation [10]. In literature, different adjustments for inertia have been applied: random assignment, increasing, decreasing, fuzzy, etc… [6]
- *The Cognitive Coefficient – c_1:* The cognitive term of the formula refers to the ambition of the particle for developing its own objective value. This coefficient defines the degree of self confidence of the particle. The higher the cognitive coefficient is, the more the particle tends to circle around its personal best [6]. If c_1 is ignored, the formula becomes cognitively memoryless [1].
- *The Social Coefficient – c_2*: The social term of the formula refers to the ambition of the particle for reaching to the best position of the swarm. This coefficient defines the degree of trust in the swarm. The higher the social coefficient is, the more the particle tries to be the best of the swarm [6]. If c_2 is ignored, the formula is socially memoryless [11].

  The cognitive and social coefficients are the acceleration coefficients of the algorithm. They are usually compared with each other in order to conduct a useful comment on the behavior of the particle. If c_1 is highly larger than c_2, than the particle has more trust in itself rather than the swarm. Kennedy proposes that the best values of the algorithm are $c_1 = c_2 = 2$, yet, it is proven to be problem dependant [11].
- *Swarm Size*: Engelbrecht [6] offers that the excessive number of particles would slow down the algorithm whereas too few number of particles are insufficient for covering up the search space. Moreover, Engelbrecht [6] states that comparing former studies, the optimal swarm size changes from 10 to 30.
- *Neighborhood Size*: The neighborhood size determines the extent of the social term of the velocity update formula. If the neighborhood is too small, then the particle would be relying on its cognitive information [6].
- *Maximum Velocity - v_{max}*: Though it is not directly defined in the algorithm itself, maximum velocity is one of the constraints. If maximum velocity is too high with respect to the search space, then the particles exceed the boundaries. If it is too low, then the particles will not have a robust information sharing.

C. Comparison with Other Algorithms

PSO algorithm is generally compared to Genetic Algorithms based on the analogies and Ant Colony Optimization because of sharing a similar origin.

Both PSO and Genetic Algorithms (GAs) are population based and allow information sharing in the operations. Yet, in many studies PSO has been found to be more efficient since it provides a more rapid convergence [2, 12, 13]in certain cases, it is also mentioned that GAs are more rapid whilst PSO provides better results [14]. In overall, studies show that PSO is more reliable than GAs.

The Ant Colony Optimization (ACO) is also a population based stochastic metaheuristic that simulates the nature. It imitates the foregoing behavior of ants and commonly used in shortest route problems [15-17]. In several studies, PSO is found to outrun ACO in efficiency whereas hybrid algorithms are found to be most efficient [18].

D. Hybridizations of the Algorithm

Not surprisingly, PSO algorithm is most hybridized with GAs in order to capture the advantages of both algorithms [19]. For GA hybridizations, largely various combinations of selection, crossover and mutation operators are embedded in PSO algorithm [20-21]. Furthermore, Valdez et al. [22] leads to a 3-method-hybrid methodology by combining PSO and GA using Fuzzy Logic. All studies prove that hybrid

methods produce more reliable or faster results than the simple PSO algorithm.
As for ACO, various hybrid methods exist in literature. Shelokar et. al. [23] initializes two algorithms at the first time and combines them during the iterations. Holden and Freitas [24] provide a new hybrid algorithm for dealing with continuous and nominal data.
Other hybridizations involve Local Search[25-26], GRASP [27], Differential Evolutions [19, 28] and Simulated Annealing [29].

III. THE RECENT PSO APPROACHES

A. *Predator Pray PSO*

As the PSO algorithm has evolved into more complicated algorithms in order to provide efficient results and effective iterations, the algorithms remained devoted to the motion of nature. One of these algorithms is Predator-Prey PSO algorithms which is a competitive PSO approach [6].

Three problems faced with the classical PSO are exploration overwhelming the exploitation, being blocked by the local optima and the early convergence. The hunting scheme of nature is simulated by the introduction of a second swarm [30] in order to overcome the three difficulties. If a prey swarm meets a predator swarm, they diffuse just to regroup again after the predator is gone. Diffusion provides a better exploration whereas regrouping provides a better exploitation. The steps of the Predator-Prey PSO algorithm are as follows:

Step 1. Initiating particle velocity and position of each p_i such that

$$x_{i,j} = x_{min} + r(x_{max} - x_{min}), \quad i = 1, \dots, N \quad j = 1, \dots, n$$

$$v_{i,j} = \alpha \frac{x_{min} + r(x_{max} - x_{min})}{\Delta t} \quad i = 1, \dots, N \quad j = 1, \dots, n$$

where *x* denotes the position, *v* denotes the velocity and *α* is constant in the range [0,1].

Step 2. Particles are divided into two sub-swarms randomly, namely the predator swarm and the prey swarm.

Step 3. The objective value of each particle of each sub-swarm is calculated as $f(x_i)$

Step 3. The best position for each particle and the global best position for each swarm is updated. For a problem

If $f(x_i) < f(x_i^{pb})$ then $x_i^{pb} \leftarrow x_i$

If $f(x_i) < f(x_i^{sb})$ then $x_i^{sb} \leftarrow x_i$

where *pb* denotes the particle best and *sb* denotes the swarm best.

Step 4. Particle velocity and particle position are updated, that is, the new velocities and positions are calculated for each particle in each swarm.

For predator swarm, the velocity update formula is:

$$v_{i,j} \leftarrow \frac{r(x_{i,j}^{sb} - x_{i,j})}{\Delta t}$$

where r is uniformly distributed between 0 and maximum velocity v_{max}, $x_{i,j}^{sb}$ is the swarm best of the prey swarm. It must be noticed that the predator swarm do not use best position of its own swarm but the best position prey swarm, since the predator is attracted by the prey.

For prey swarm, the velocity update formula is

If $rn \le p_f$ then

$$v_{i,j} \leftarrow wv_{i,j} + c_1 r_1 \left(\frac{x_{i,j}^{pb} - x_{i,j}}{\Delta t}\right) + c_2 r_2 \left(\frac{x_{i,j}^{sb} - x_{i,j}}{\Delta t}\right) + c_3 r_3 \frac{D(d)}{\Delta t} \quad i = 1, \dots, N \quad j = 1, \dots, n$$

Else

$$v_{i,j} \leftarrow wv_{i,j} + c_1 r_1 \left(\frac{x_{i,j}^{pb} - x_{i,j}}{\Delta t}\right) + c_2 r_2 \left(\frac{x_{i,j}^{sb} - x_{i,j}}{\Delta t}\right)$$

$$x_{i,j} \leftarrow x_{i,j} + v_{i,j}\Delta t \qquad i = 1, \dots, N \qquad j = 1, \dots, n$$

where *w* is the inertia rate between [0,1], r_1 , r_2 , r_3 and *rn* are random numbers between [0,1]. p_f is the fear probability of the prey particle from the predator particle and *D*(*d*) is a measure of the effect that the predator has on the prey and it is formulated as

$$D(d) = ae^{-bd}$$

where *d* is the Euclidean distance between the prey particle and the nearest predator particle. *a* and *b* are positive constants that define the effect of distance to velocity.

Step 5. Step 2 is returned until a termination criterion is satisfied.

The Predator-Prey PSO algorithm provides additional parameters to be tuned.

- *Fear probability* – p_f: If the fear probability is assigned 0 for all prey particles, then the particles treat as an ordinary swarm given in Part 2., The fear probability is decreased over iterations [6].
- *Prey coefficient* – c_3: If the prey coefficient is assigned much greater than the cognitive coefficient – c_1 and the social coefficient – c_2, the prey group is expected to diverge and not to regroup which results in random search for the prey particles.
- *Distance coefficients – a and b:* The coefficient *a* has the same effect as the fear probability and should be decreased over time. On the other hand, *b* has the counter effect of *a* and should be increased over time.

B. *The Hunting Search*

In the Hunting Search(HS), the hunting scheme of animals (e.g. lions) are simulated with minor deficiencies. The particles have become hunters and the hunters are after a prey which is the optimum solution. In nature, predators can see or smell the preys but in the HS, a blind search process is run. Additionally, in nature, preys are dynamic (as in Predator-Prey PSO) whereas in the HS, the prey (the optimal solution) is static. The algorithm is based on approaching the hunter leader which is in the best position and reorganizing if the hunters are close enough but still cannot find the prey [31].

The steps of the algorithm are as follows.
Step 1. Initializing the hunting group randomly (as in PSO).

$$x_{i,j} = x_{min} + r(x_{max} - x_{min})$$

Step 2. Calculating the objective values of each hunter by $f(x_i)$. Assign the hunter with the best objective value as the hunter leader.
Step 3. Move hunters closer to the hunter leader.

$$x_{i,j} \leftarrow x_{i,j} + r \cdot MML \cdot \left(x_{i,j}^{L} - x_{i,j}\right)$$

where r is a random number between [0,1], *MML* is the maximum movement towards leader and $x_{i,j}^{L}$ is the position of the leader.
If the movement is successful, the hunters stay in the new position. If not, the hunter stays in his previous position. This enables weak hunters to search for other solutions and avoids premature convergence.
Step 4. After approaching to the hunter leader, hunters make a position correction, that is, they search their environments and correct their position according to the new information. There are two ways for position correction: real value correction and digital value correction.

a. Real Value Correction: Real value correction uses a probability named Hunting Group Consideration Rate (*HGCR*) according to the formula below:

$$x_{i,j} \leftarrow \begin{cases} x_{i,j} \; where \; i \in \{1,2,\ldots,HGS\} & rn < HGCR \\ x_{i,j} \mp RA & rn > HGCR \end{cases}$$

where *HGS* is the number of hunters and *RA* is the distance radius.

b. Digital Value Correction: The same method as real value correction but digits instead of values. The value 15.8645 is a value which has 6 digits. Digit value correction involves selecting a random digit and applying value correction over that digit.

Step 5. If at any part of the process the hunters get stuck with a local optimum, they reorganize themselves. This situation is defined two-folds: (a) the difference of objective values between the best and the worst hunter is constant and close to 0 (b) a predetermined number of iteration is reached. In this case, the leader saves its position. The other hunters are distributed among the search space as follows:

$$x_{i,j} = x_{min} + r(x_{max} - x_{min}) + ae^{-bEN}$$

where *EN* is the number of past reorganizations and *a* and *b* are positive constants.
Step 6. Repeat steps 2-6 until a termination criterion is satisfied.
The new parameters of this algorithm are:

- *Maximum movement towards leader – MML*: It is case dependent. For a small number of iterations, the number is assigned larger and for a large number of iterations, the number is assigned smaller.
- *Hunting group consideration rate* –It is generally assigned between 01. And 0.4 depending on the problem
- *Distance radius – RA*: It is an arbitrary radius for continuous variables and kept constant or reduced during iterations (Oftadeh et. al., 2010).
- *Hunting group size – HGS*: The number of the hunters is defined by *HGS*. There is no predetermined group size defined for problems. Yet, it can be commented that in PSO, the optimal swarm size is between 10 and 30 and this number can be a reference to HS.

IV. ANIMAL FOOD CHAIN BASED PSO

A. The Algorithm

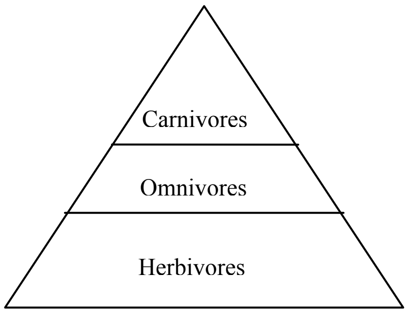

Fig. 1. Animal Food Pyramid

In the food chain, animals form three groups: herbivores, omnivores and carnivores. Herbivores are animals that eat plants, carnivores are animals that eat other animals and finally omnivores are animals that eat both animals and plants. In nature, herbivores are animals that are below in the food pyramid. Omnivores are in the middle in the food pyramid and feed on both plants and specific herbivores. Lastly, carnivores are at the top of the food pyramid and feed on specific herbivores and omnivores. This makes herbivores the ultimate preys, the carnivores the ultimate predators and omnivores both predators and preys.
In nature, according to the transformation of energy, the number of herbivores is greater than omnivores and the number of omnivores is greater than the number of carnivores. In wild environments, the herbivore-omnivore-carnivore ratio can be 10:3:1 whereas in calm environments the ratio can be 40:10:1.
Omnivores are the slowest of the food chain whereas carnivores are the fastest.
The Animal Food Chain Based PSO is based on this food flow in nature. The steps of the algorithm are given below:
Step 1. Define the herbivore-omnivore-carnivore ratio according to the environment
IF the environment is harsh – wild → 10:3:1
IF the environment is average → 25:6:1
IF the environment is calm → 40:10:1
Step 2. According to the environment initialize swarms.
n_h: the number of herbivores
n_o: the number of omnivores
n_c: the number of carnivores
Step 3. Calculate the objective function for all swarm particles
Step 4. Find the best position for each swarm

Step 5. Apply velocity updates for each swarm using the formula given below:

- Herbivores (Ultimate preys):

$$v_{ij} \leftarrow \omega v_{ij} + c_1 r_{1j}(t)(y_{ij} - x_{ij}) + c_2 r_{2j}(t)(\hat{y}_i - x_{ij}) + p_{fho} c_3 r_{3j} D(d_o) + p_{fhc} c_4 r_{4j} D(d_c)$$

where
$v_{ij}(t)$: velocity of j^{th} particle of i^{th} swarm
ω: the inertia coefficient
c_1 and c_2: acceleration coefficients
$r_{1j}(t)$, r_{2j}(t), $r_{3j}(t)$ and $r_{4j}(t)$: random numbers for the j^{th} particle in the interval [0,1]
$y_{ij}(t)$: personal best for the j^{th} particle of the i^{th} swarm
$x_{ij}(t)$: the position of the j^{th} particle of the i^{th} swarm
$\hat{y}_i(t)$: best position of the i^{th} swarm
p_{fho}: fear factor or probability of herbivores from omnivores (in the interval [0,1])
p_{fhc}: fear factor or probability of herbivores from carnivores (in the interval [0,1])
$D(d_o)$: distance based coefficient of herbivores from omnivores
$D(d_c)$: distance based coefficient of herbivores from carnivores

- Carnivores (Ultimate predators):

$$v_{ij} \leftarrow r \cdot (\hat{y}_i - x_{ij})$$

where
$v_{ij}(t)$: velocity of j^{th} particle of i^{th} swarm
r: random number in the interval [0,1]
$\hat{y}_i(t)$: best position of the i^{th} swarm
$x_{ij}(t)$: the position of the j^{th} particle of the i^{th} swarm

- Omnivores (Both predators and preys):

$$v_{ij} \leftarrow (1 - p_p) \cdot \left(\omega v_{ij} + c_1 r_{1j}(y_{ij} - x_{ij}) + c_2 r_{2j}(\hat{y}_i - x_{ij}) + p_{foc} c_3 r_{3j} D(d_o)\right) + p_p \cdot \left(r \cdot (\hat{y}_i - x_{ij})\right)$$

where
p_p: the probability of omnivores being a predator
p_{foc}: fear factor or probability of omnivores from carnivores (in the interval [0,1])

Step 6. Update positions using the formula

$$x_{ij} \leftarrow x_{ij} + v_{ij}$$

Step 7. Repeat steps 3-7 until convergence is caught.
The new parameters are:

Fright factor of herbivores to omnivores - p_{fho}:
Fright factor of preys to predators has been defined before. Yet, this factor (i) has not been formulized and (ii) always been used with a threshold distance, that is, if a predator is nearer than a threshold distance then the prey fears at a full level, no matter how close the predator is. The new approach to this parameter involves a distance based formula which again uses a threshold. Contrarily, this threshold denotes the minimum distance that the prey should start to fear (d_{fho}^{min}) from its predator. The fear probability is inversely proportional with the distance.

$$p_{fho} = 1 - \frac{d_{fho}}{d_{fho}^{min}}$$

Fright factor of herbivores to carnivores $-$ p_{fhc}:
This factor is fear factor or probability of herbivores from carnivores. It has the same characteristics with p_{fho}. It is calculated as

$$p_{fhc} = 1 - \frac{d_{fhc}}{d_{fhc}^{min}}$$

Since carnivores move faster than omnivores,
$d_{fho}^{min} < d_{fhc}^{min}$.

Fright factor of omnivores to carnivores $-$ p_{foc}:
This factor is fear factor or probability of omnivores from carnivores. It has the same characteristics with p_{fho}. It is calculated as

$$p_{foc} = 1 - \frac{d_{foc}}{d_{foc}^{min}}$$

Since omnivores move faster than herbivores,
$d_{foc}^{min} < d_{fhc}^{min}$.
All d_{foc}^{min}, d_{fhc}^{min} and d_{fho}^{min} are new parameters to be optimized.

Predator Probability $-$ p_p:
This factor is the probability of omnivores being a predator which is formulated as

$$p_p = \frac{d_o}{d_c + d_o}$$

where
d_o: distance to the nearest omnivore
d_c: distance to the nearest carnivore

Environmental Factor: The initial number and rate of swarms are defined according to the environment being wild, average or calm. This is also a parameter to be tuned.

V. Conclusion

The most recent PSO algorithms enhance the swarm simulations as collaborative activities are not anymore limited to the positioning. The hunting based approach takes the difference of power among the particles. But the power of particles differ by the environmental influences.

The Animal Food Chain Based PSO originates from Predator Pray, therefore has more than one swarm. However in the Predator-Prey PSO, both swarms have an equal number of particles which is improved to differentiate to the natural reality. The Animal Food Chain Based PSO introduces environment factor. Based on this parameter, the swarm size, thus exploration-exploitation balance changes. In the Hunting Search, only one swarm blindly seeks a static prey. In the Animal Food Chain Based PSO, all preys and predators are dynamic and leave traces for the predator. Moreover, considering the diffusion and regrouping of prays the Animal Food Chain Based PSO introduces a new swarm which is both predator and prey. Even when the ultimate preys swarm diffuse too much and cannot regroup, the balancing swarm will take this characteristic and avoid the early convergence.

References

[1] T. Weise, Global Optimization Algorithms – Theory and Applications, 2009.

[2] R. Hassan, B. Cohanim, B., O. de Weck and G. Venter, "A Comparison of Particle Swarm Optimization and The Genetic Algorithm", 46th AIAA/ASME/ASCE/AHS/ASC Structures,

Structural Dynamics, and Materials Conference (2005), Austin, Texas, USA, April 18-21, 2005, pp. 1-13.

[3] A. Dutot, D. Olivier and G. Savin, "Collaboration and Competition in Boids Simulation with Predation", Emergent Properties in Natural and Artificial Complex Systems, EPNACS 2010, Lisbon, Potugal, September 15-16, 2010, pp. 1-3.

[4] C. Veenhuis and M. Köppen, "Data Swarm Clustering", *Swarm Intelligence in Data Mining,* 2006, pp. 221-241.

[5] CW Reynolds, "Flocks, Herds, Schools: A Distributed Behavioral Model", *Computer Graphics*, 1987, vol 21, pp. 25-33.

[6] A.P. Engelbrecht, Fundamentals of Computational Swarm Intelligence, Wiley Interscience, 2005, New York.

[7] G. Ciuprinu, D. Ioan and I. Monteanu, "Use of Intelligent-Particle Swarm Optimization in Electromagnetics", *IEEE Transactions on Magnetics*, Vol. 38, No. 2, March 2002, pp. 1037-1040.

[8] E.G. Castro and MSG Tsuzuki, "Simulation Optimization Using Swarm Intelligence as Tool for Cooperation Strategy Design in 3D Predator-Prey Game", Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, *edited by: Felix T.S. Chan and Manoj Kumar Tiwari, ISBN: 978-3-902613-09-7*, 2007, InTech.

[9] A. Léon-Javier, N. Cruz-Cortez, M.A. Moreno-Armendariz and S. Orantes-Jiménez, "Finding Minimal Addition Chains with a Particle Swarm Optimization Algorithm", *Lecture Notes in Computer Science, 2009, Volume 5845, MICAI 2009: Advances in Artificial Intelligence*, 2009, pp. 680-691.

[10] C.D. Liou and Y.C. Hsieh, "A PSO-Based Algorithm for Two-Machine No-Wait Flowshop Scheduling with Setup Time*", Proceedings of the International Conference on Business and Information*, 2009, Kuala Lumpur, Malaysia, July 6-8.

[11] W. Zhang, L. Hua, Z. Zhang and H. Wang, "The Selection of Acceleration Factors for Improving Stability of Particle Swarm Optimization", Fourth International Conference on Natural Computation, August 2008, Jinan, China 25-27, pp 376-380.

[12] F. Djeffal, N. Lakhdar, M. Meguellati and A. Benhaya, "Particle Swarm Optimization Versus Genetic Algorithms to Study The Electron Mobility in Wurtzite GaN-Based Devices", *Solid-State Electronics*, Vol. 53, 2009, pp. 988–992.

[13] H.R.Golmakani and M. Fazel, "Constrained Portfolio Selection using Particle Swarm Optimization", *Expert Systems with Applications*, Vol.38, 2011, pp. 8327–8335.

[14] S. Panda and N.P. Padhy, "Comparison of Particle Swarm Optimization and Genetic Algorithm for FACTS-Based Controller Design", *Applied Soft Computing*, Vol. 8, 2008, pp. 1418–1427.

[15] A. Chebouba, F. Yalaoui, A. Smati, L. Amadeo, K. Younsi and A. Tairi, "Optimization of Natural Gas Pipeline Transportation Using Ant Colony Optimization", *Computers and Operations Research*, Vol. 36 Issue 6, 2009, pp. 1916-1923.

[16] V. Arora, F.T.S. Chan and M.K. Tiwari, 2009. "An Integrated Approach For Logistic And Vendor Managed Inventory In Supply Chain", *Expert Systems with Applications*, Vol.37 Issue 1, 2010, pp. 39-44.

[17] A. Uğur and D. Aydın, "An Interactive Simulation And Analysis Software For Solving TSP Using AntColony Optimization Algorithms", *Advances in Engineering Software*, Vol. 40 Issue 5, 2009, pp. 341-349.

[18] Y. Marinakis, M. Marinaki, M. Doumpos and C. Zopounidis, "Ant Colony and Particle Swarm Optimization for Financial Classification Problems", *Expert Systems with Applications*, Vol. 36 Issue 7, 2009, 10604-10611.

[19] R. Thangaraj , M. Pan, A. Abraham and P. Bouvry, "Particle Swarm Optimization: Hybridization Perspectives and Experimental Illustrations", *Applied Mathematics and Computation*, Vol. 217, 2011, pp. 5208–5226.

[20] Z. Lian, X.Gu and B. Jiao, "A Similar Particle Swarm Optimization Algorithm for Permutation Flowshop Scheduling to Minimize Makespan", *Applied Mathematics and Computation*, Vol. 175, 2006, pp. 773–785.

[21] R.J. Kuo and Y.S. Han, "A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Solving Bi-level Linear Programming Problem- A Case Study on Supply Chain Model", *Applied Mathematical Modelling*, DOI: 10.1016/j.apm.2011.02.008.

[22] F.Valdez, P. Melin and O. Castillo, "An Improved Evolutionary Method with Fuzzy Logic for Combining Particle Swarm Optimization and Genetic Algorithms", Applied Soft Computing, Vol. 11, 2011, pp. 2625–2632.

[23] P.S. Shelokar, P. Siarry, V.K. Jayaraman and B.D. Kulkarni, "Particle Swarm and Ant Colony Algorithms Hybridized for Improved Continuous Optimization", *Applied Mathematics and Computation*, Vol. 188, 2007, pp. 129–142.

[24] N. Holden and A.A. Freitas, "A Hybrid PSO/ACO Algorithm for Discovering Classification Rules in Data Mining", *Journal of Artificial Evolution and Applications*, 2008, pp. 1–11, doi:10.1155/2008/316145.

[25] C.H. Liao, C.T: Tseng and P. Luarn, "A Discrete Version of the Particle Swarm Optimization for Flowshop Scheduling Problems", *Computers and Operations Research*, Vol. 34 Issue 10, 2005, pp. 3099-3111.

[26] G. Haibing, Z. Chi and G. Liang, G., "Particle Swarm Optimization Based Algorithm for Economic Load Dispatch" *International Symposium on Intelligence Computation & Applications (ISICA 2005)*, Wuhan, China, April, 4-6, 2006, pp. 594-599.

[27] Y. Marinakis and M. Marinaki, "A Hybrid Multi-Swarm Particle Swarm Optimization Algorithm for the Probabilistic Traveling Salesman Problem", *Computers & Operations Research*, Vol. 37 Issue 3, 2010, pp 432-442.

[28] M. Pant, R. Thangaraj nad A. Abraham, "Particle Swarm Optimization Using Adaptive Mutation", *Proceedings of 19th International Conference on Database and Expert Systems Application*, Italy, 2008, pp. 519–523.

[29] Y. Da and G. Xiurun, "An Improved PSO-Based ANN with Simulated Annealing Technique", *Neurocomputing*, Vol. 63, 2005, pp.527–533.

[30] Z. Xian-cheng, "Color Image Filter Based on Predator-prey Particle Swarm Optimization", *2009 International Conference on Artificial Intelligence and Computational Intelligence*, Las Vegas, Nevada, USA; July 13-17, 2006,pp.480-484.

[31] R. Oftadeh, M.J. Mahjoob and M. Shariatpanahi, "A Novel Meta-Heuristic Optimization Algorithm Ispired by Group Hunting of Animals: Hunting Search", Computers and Mathematics with Applications, Vol. 60, 2010, pp. 2087-2098.