A Bi-level Objective Particle Swarm Optimization Algorithm Based on Improved Strategies of Inertia Weight

Defa Hu

School of Computer and Information Engineering,
Hunan University of Commerce, Changsha 410205, Hunan, China

Zhuang Wu*

Information College, Capital University of Economics and Business Beijing 100070, China

**Corresponding author: wuz9080@163.com*

Abstract
In accordance with the features of bi-level programming model and based on the concept of the improved strategies of inertia weight, this paper proposes a Bi-level Objective Particle Swarm Optimization (BLOPSO) to solve a kind of optimization problems. In the BLOPSO algorithm, a lower-level multi-objective optimization is embedded in the upper-level objective optimization process to seek its feasible searching space, adjusts the interactive iteration between the two particle swarm optimization algorithms in the upper and lower levels through inertia weight, obtains ideal approximate global optimal solution and reflects the decision-making process of the bi-level programming problems. BLOPSO algorithm doesn't rely on concrete bi-level programming model and it can seek special bi-level programming problems and provides a general framework to seek complicated bi-level programming problems, besides, it has extensive universality and suitability and it can obtain high-quality global optimal solution.
Key words: BI-LEVEL, PARTICLE SWARM OPTIMIZATION, IMPROVED STRATEGIES OF INERTIA WEIGHT

1. Introduction

Bi-level programming optimization problem is a vigorous and challenging field which people have been investigating and exploring for a long time. Numerous actual optimization problems are difficult to solved and the traditional optimization methods such as Newton's method, conjugate gradient method, pattern search method and simplex algorithm have failed to meet people's requirements, therefore, to design highly-efficient optimization algorithms have become the research objective for many science researchers [1]. In the year of 1995, James Kennedy, a US social psychologist and Russell Eberhart, an electrical engineer had come up with Particle Swarm Optimization (PSO) together, which is a

global random search algorithm based on swarm intelligence, which was inspired by the research achievements of artificial life and simulating the migration and cluster behaviors of birds flock's foraging[2]. As the other evolution algorithms, this algorithm is also based on the concepts of "swarm" and "evolution" and it realizes the search of the optimal solution in the complicated space through the collaboration and competition among the individuals, in the meanwhile, different from the other evolution algorithms which conduct evolution operator operations like crossover, mutation and selection on the individuals, PSO sees the individuals in the swarm as the particles with no mass and volume in n-dimensional space where every particle moves in the solution space at a certain rate, aggregates towards its previous optimal position and the previous optimal position in the neighborhood and realizes the evolution of the candidate solutions[3].

In 1995, IEEE international neural network academic conference had released a paper named "Particle Swarm Optimization", suggesting the birth of PSO algorithm[4]. PSO algorithm is easy to understand with excellent biological and social backgrounds and to realize with few parameters and it also has strong global search capacity on non-linear and multi-modal problems, therefore, it has drawn extensive attention in the scientific research and engineering practice[5]. To analyze and improve PSO not only has theoretical significance, but also has certain practical application value. Only a dozen years passed between the coming-into-being and the further development of PSO algorithm, therefore, it still has a weak theoretical foundation and it also has the defects of slow convergence rate and prematurity. Whilst resolving the single- and multi-objective optimization problems, most of the researchers have been focusing on how to accelerate the convergence rate of PSO and avoid premature convergence[6].

This paper firstly introduces the principle of Particle Swarm Optimization algorithm as well as global and local models, elaborates the procedures of PSO and proposes inertia weight adjusting PSO in order to avoid the problems that PSO is slow in search rate and easy to be trapped in local extremum in solving complicated functions. Then through experimental simulation and analysis, it applies this algorithm to the upper and lower level solution procedures of the bi-level programming problems and raises a universal BLOPSO algorithm to solve bi-level programming model problems. As a general algorithm to resolve bi-level programming models, this algorithm of the paper doesn't need any assumption terms or any conversion processing of the objective function and constraints and it doesn't depend on concrete bi-level programming model.

2. Origin of Particle Swarm Optimization

Ever since the 1930s, the development of social psychology has revealed that we are the followers of fish and bird flock's aggregation behavior. In people's continuous interaction and due to the interactive influence and simulation, they always become more and more similar, thus forming code the civilization. Human's natural behaviors are not similar with fish or birds, however, humans have a very similar thinking track in the high-dimensional cognitive space. The social phenomenon behind the thinking is far more complicated than the beautiful motions in the clustering of fish and birds: firstly, thinking occurs in the faith space and its dimension is more than 3, besides, when two thoughts converges to the same point, we call it consistency but not conflict. With the research human beings make on life heuristic computation, the self-organizing behaviors of some social animals (i.e. ant colony, bee colony and bird flock) have aroused extensive attention of the scientists[7]. These social animals have formed a common feature in the long-lasting evolution process: their individuals have simple behaviors, however, when they work together, they can fulfill very complicated behaviors. Based on this, people have designed a great number of optimization algorithms such as ant colony algorithm, particle swarm optimization, shuffled frog leaping algorithm and artificial fish swarm algorithm and have applied these algorithms in many fields successfully. The concept of particle swarm optimization comes from the simulation of the foraging behavior of birds. Although the birds might change their flying direction and clustering behavior in the foraging, certain regularity exists as a whole. Research has proved that there is an information sharing mechanism in the birds. PSO algorithm first randomly initializes particle swarm in the given solution space and the number of the variables of the problems to be optimized determine the dimensions of the solution space. Every particle has the initial position and speed and optimizes through iterations. In every iteration, every particle updates its own position and flying speed in the solution space by tracking two extremums. The first extremum is the optimal solution particle a signal particle can find in the

iteration, which is called the individual extremum. In the simulation, every individual follows three principles: to avoid clashes with the neighborhood individuals, to fly towards the center of the bird flock and the entire population flies towards the goal. In 1990, Frank Heppner, a biologist had put forth bird model. Its difference is that the birds are attracted to the habitat. In the simulation, every bird has no specific flying goal at the beginning, it determines its flying direction and speed with simple rules (every bird tries to stay in the bird flock and not to clash each other). When one bird has flown to the habitat, the birds surrounding this bird will follow the same direction. In this way, all birds will go to the habitat. Fig.1 is the Cluster Model of Bird Flock of PSO[8].

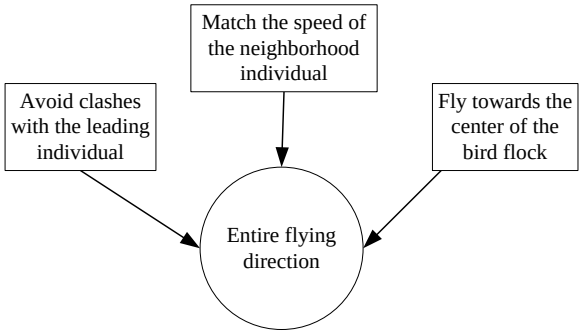

Figure 1. Cluster model of bird flock

It can be seen from the Fig.1 that PSO algorithm realizes the search of optimal solution in the complicated space through the collaboration and competition among the individuals, meanwhile, unlike other evolution algorithms which perform operator operations such as crossover, mutation and selection on the individuals, PSO treats the individuals of the swarm as the particles with no mass and volume in the *n*–dimensional search space, where every particle moves in the solution space at a certain speed and clusters towards the previous optimal solution and the previous optimal solution in the neighborhood so as to realize the evolution of candidate solutions. PSO algorithm is easy to understand with its excellent biological and social backgrounds and to realize with few parameters and it has strong global searching capacity on the non-linear and multi-modal problems[9].

3. The Theory of Particle Swarm Algorithm

PSO algorithm initializes a group of random particles (random solutions) and its final optimal solution is sought through several iterations[10]. In every iteration, every particle updates itself through two factors (obtains new positions through new speed): one is the particle searches its own optimal solution, which is called "self-awareness", which is usually significantly related to the local searching performance of the algorithm and the other is what is called "swarm intelligence", which is the optimal solution sought by the entire swarm and which can lead the entire swarm to cluster towards the global optimal solution in the speed update so as to obtain the optimal solution under the cooperation of the individuals and the swarm[11,12]. The motion trail of the particles in the optimization process is shown in Fig.2.

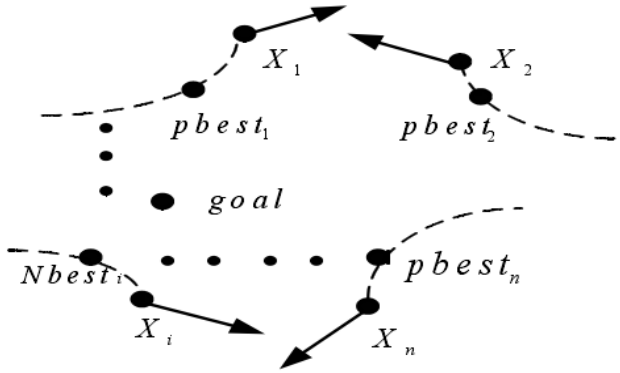

Figure 2. Optimization search diagram of particle swarm optimization

Assume that in a D–dimensional goal search space, N particles have formed a swarm and the ith particle is a D–dimensional vector.

$X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$, $i = 1,2,\cdots,N$ 。

The "flying" speed of the ith particle is also a D–dimensional vector, which is recorded as:

$V_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$, $i = 1,2,\cdots 3$ 。

The optimal position the ith particle has searched is named as the individual extremum, which is recorded as:

$p_{best} = (p_{i1}, p_{i2}, \cdots, p_{iD})$, $i = 1,2,\cdots,N$ 。

The optimal position the entire particle swarm has searched is the global extremum, which is recorded as:

$g_{best} = (p_{g1}, p_{g2}, \cdots, p_{gD})$

When finding these two optimals, the particle updates its speed and position according to the following formulas, namely Formula (1) and (2).

$$v_{id} = w * v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (1)$$

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

In here: c_1 and c_2 are learning factors, also named acceleration constants and r_1 and r_2 are the uniform random numbers with [0,1].

4. Procedures of PSO in Solving Bi-level Programming Model

The biggest challenge to solve bi-level programming problems lies in the fact that a solution is the feasible solution of the entire optimization problem only when it is the optimal solution of the lower-level optimization problem. In a way, this requirement doesn't allow any approximate methods to be used to solve the lower-level optimization problems [13]. The realization of upper-level objectives is subject to the accomplishment of the decision objectives in the lower level. Therefore, while selecting the strategies to optimize the accomplishment of the upper-level objectives, the adverse effects on the upper-level must be taken into consideration by the strategies that might be adopted in the lower-level. Generally, the admissible strategy sets of the upper-and lower-level decision-makers are inseparable and they usually form a correlated whole [14,15]. The basic steps of BLOPSO algorithm are classified as follows[16]:

Step1: Initialize the *D*-dimensional variable of every particle i. Randomly initialize the position variable X_i and the speed V_i of the particle i which obeys uniform distribution, set the maximum iteration $T_{\max}$, initialize the global and local learning items of particle i as the number of the particle in the swarm and randomly generate the initial solution which meets the lower-level programming of the bi-level model.

Step2: Set the individual extreme $pBest$ of every particle in the swarm and the global extreme $gBest$ of the entire particle swarm.

Step3: Implement Step 3.1-3.5 on every particle in the swarm.

Step3.1: According to Formula (1) and (2), update the position variable x_i and speed variable v_i of the individual i in the swarm, evaluate the particle fitness and update the previous optimal solution of the particle and the previous optimal solution of the swarm according to the fitness value.

Step3.2: Substitute the solution of the upper-level bi-level programming model, namely the position variable X_i of particle i into the lower-level model and seek the optimal solution Y_i of the lower-level model through inertia weight adjusting PSO.

Step3.3: Substitute the value (X_i, Y_i) of the position variable and the optimal solution of particle i into the upper-level programming model and calculate the fitness value $F(X_i, Y_i)$ of particle i where $i \in [1, m]$ and m is the number of particles according to the fitness function $F(x, y)$.

Step3.4: Calculate the threshold value of the cluster degree and the average cluster distance among the particles and judge whether it is trapped in local convergence with the local convergence of the algorithm. If the fitness value $F(X_i, Y_i)$ of the ith particle is superior to the current fitness value, update its position X_i as $pBest$ and the corresponding optimal solution of the lower-level model as Y_i.

Step3.5: Use the improved inertia weight adjusting strategies to recombine the global and local learning combined variables. If the fitness value $F(X_i, Y_i)$ of the ith particle is superior to the current global fitness value, update the position X_i of this particle as $gBest$ and the corresponding optimal solution in the lower-level programming as Y_i, otherwise, keep the current learning item unchanged (namely keep the combined variable value).

Step4: When the maximum iterations meet the termination condition of the algorithm, turn to Step 6 and output the optimal solution, otherwise, turn to Step 5.

Step5: Use the inertia weight adjusting PSO to seek the corresponding $ygBest$ in the lower-level programming model to the $gBest$ in the upper-level. Turn to Step 3 directly and repeat the steps.

Step6: Update the previous optimal solution of the particle according to the fitness value, seek the optimal solutions $gBest$ and $ygBest$ and to the bi-level programming problems and output the optimal solution. Calculate the corresponding fitness function value to the upper- and lower-level optimal solutions of the bi-level programming model.

The flowchart of BLOPSO algorithm is indicated in Figure 3.

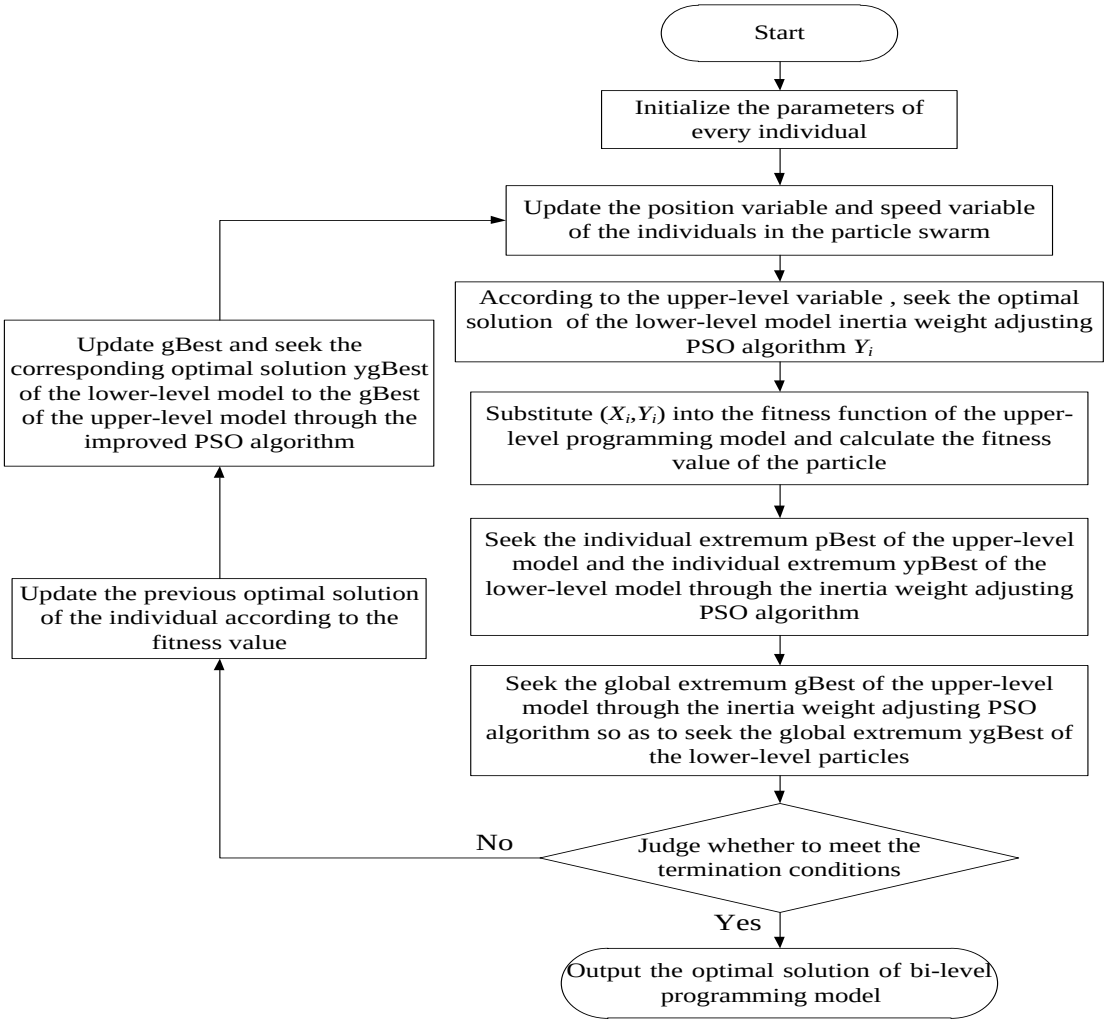

Figure 3. Flowchart of BLOPSO algorithm based on Bi-level programming

5. Example Verification of BLOPSO Algorithm

The following four test functions are selected to verify the effectiveness of BLOPSO. Their parameters are set as follows: particles=60, dimensions=20, learning factors c_1 and c_2=2, the maximum inertia weight=0.9, the minimum inertia weight=0.3, the maximum iterations=500 and the convergence accuracy=10^{-10}. The verification tests of this algorithm are realized in Windows7 platform and in MATLAB R2012a.

(1) Function F1:

$$\begin{aligned}&\min_x F(x,y)=3x_1+2x_2-2y_1-3y_2-50\\&s.t. x_1+x_2+y_1-2y_2\le 45\\&0\le x_1\le 45,\ 0\le x_2\le 45\\&\min_y f(x,y)=(y_1-x_1+30)^2+(y_2-x_2+30)^2\\&s.t. 2y_1-x_1+10\le 0,\ \ 2y_2-x_2+10\le 0\\&-10\le y_1\le 30,\ -10\le y_2\le 30\end{aligned}\qquad (3)$$

(2) Function F2:

$$\begin{aligned}&\min_x F(x,y)=(x_1-20)^2+(x_2-30)^2-15y_1+15y_2\\&s.t. -x_1-x_2+30\le 0, x_1+x_2-20\le 0, x_2\le 10\\&\min_y f(x,y)=(x_1-y_1)^2+(x_2-y_2)^2\\&s.t. 0\le y_1\le 15,\ 0\le y_2\le 15\end{aligned}\qquad (4)$$

(3) Function F3:

$$\begin{aligned}&\min_x F(x,y)=x^2+(y-10)^2\\&s.t. x+3y-10\le 0, x\ge 0\\&\min_y f(x,y)=x^3-3y^3+x-3y-x^2\\&s.t. -x+3y-2\le 0, y\ge 0\end{aligned}\qquad (5)$$

(4) Function F4:

$$\begin{aligned}&\min_x F(x,y)=(x-6)^2+(3y+1)^2\\&s.t.\ x\ge 0\\&\min_y f(x,y)=(x-1)^2-2xy+x^3\\&s.t. -2x+y+3\le 0, x-y-4\le 0\\&x+y-8\le 0, y\ge 0\end{aligned}\qquad (6)$$

We have operated BLOPSO algorithm on each of the four test functions 20 times and obtained the approximate global solutions. We

verify this algorithm by adopting the following indexes: the optimal solutions, the worst solutions and the average solutions of the bi-level programming problems, the standard deviation of the optimal solutions in the 20 experiments and the objective value function of the corresponding lower-level programming problems to the optimal and the worst solutions to the objective functions in the given upper-level programming problems. See the experimental results in Table 1.

Table 1. Test Results of BLOPSO Algorithm

Function	Upper-level Objective Function Value				Lower-level Objective Function Value	
	Average Solution	Standard Deviation	Optimal Solution	Worst Solution	Optimal Solution	Worst Solution
F1	-38	0	-38	-38	520	520
F2	122.0135	1.0306	121.0213	125.3274	122.7282	121.3364
F3	64.5816	0.1365	64.2169	64.9543	-27.3159	-26.8328
F4	5.3062	0.0728	5.1546	5.6137	75.4271	74.8173

BLOPSO algorithm has good robustness, can find many feasible solution, these feasible solution has the same upper programming objective function values, but lower level programming objective function value is different. The data in Table 1 have demonstrated that for the above four functions, the standard deviations of the optimal value in 20 experiments are small with one function to be 0, suggesting that the result is stable and that BLOPSO algorithm can find many feasible solutions, which have the same upper-level programming objective function value but different lower-level programming objective function value. If the actual mathematical model is a bi-level programming problem, the solution is the optimal solution of the bi-level programming from the point of view of the model. If attach certain conditions, even optimal solution cannot meet, should also consider first in the feasible region for the meet the conditions of the suboptimal solution and its hierarchical structure reflected in the upper right greater than that of the lower layer, the lower layer obey the upper. The reason is that the function value of the upper-level model depends on the decision variables of the lower-level model and that it is not directly related to the decision variables of the upper-level programming, which will only affect the lower-level programming objective function value. These feasible solutions have the same lower-level decision variable value.

6. Conclusion

PSO algorithm is the first choice to solve bi-level programming problems because it is simple to understand and easy to realize and it has strong global search capacity and few adjustable parameters. This paper problems a Bi-level Objective Particle Swarm Optimization (BLOPSO) algorithm to solve this kind of problems according to the characteristics of bi-level optimization and it verifies that BLOPSO algorithm has excellent robustness and it can find satisfactory approximate global optimal solution through examples.

Acknowledgements

This work was supported by the Beijing Philosophical Social Science Project (No.14SHB015), the Beijing Municipal Education Commission Foundation of China (No. SM201410038013) and the National Natural Science Foundation of China (Grant No: 61202464).

References

1. Reza Firsandaya Malik, Tharek Abdul Rahman, Razali Ngah, et al.(2012) The New Multipoint Relays Selection in OLSR using Particle Swarm Optimization. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 10(2), p.p.343-352.
2. H. Omranpour, Indonesia, M. Ebadzadeh, S. Shiry, S. Barzegar (2012) Dynamic Particle Swarm Optimization for Multimodal Function. *IAES International Journal of Artificial Intelligence*, 1(1), 1-10.
3. Shih-Ying Lin, Shi-Jinn Horng, Tzong-Wann Kao, et al.(2012) Solving The Bi-objective Personnel Assignment Problem using Particle Swarm Optimization. *Applied Soft Computing*, p.p.12(9), 2840-2845.
4. Mengqi Hu, Jeffery D. Weir, Teresa Wu (2014) An Augmented Multi-objective Particle Swarm Optimizer for Building Cluster Operation Decisions. *Applied Soft Computing*, 25(12), p.p.347-359.

5. J. Behnamian, S.M.T. Fatemi Ghomi (2014) Multi-objective Fuzzy Multiprocessor Flowshop Scheduling. *Applied Soft Computing*, 21(8), p.p.139-148.
6. Amirhossain Chambari, Seyed Habib A. Rahmati, Amir Abbas Najafi, et al. (012) A Bi-objective Model to Optimize Reliability and Cost of System with A Choice of Redundancy Strategies. *Computers & Industrial Engineering*, 63(1), p.p.109-119.
7. Jinxing Che (2014) A Novel Hybrid Model for Bi-Objective Short-Term Electric Load Forecasting. *International Journal of Electrical Power & Energy Systems*, 61(10), p.p.259-266.
8. Ma Luz López García, Ricardo García-Ródenas, Antonia González Gómez (2014) Hybrid Meta-heuristic Optimization Algorithms for Time-domain-constrained Data Clustering. *Applied Soft Computing*, 23(10), p.p.319-332.
9. Ashish Kumar Bhandari, Vineet Kumar Singh, Anil Kumar, et al. (2014) Cuckoo Search Algorithm and Wind Driven Optimization Based Study of Satellite Image Segmentation for Multilevel Thresholding Using Kapur's Entropy. *Expert Systems with Applications*, 41(7), p.p.3538-3560.
10. Laura Emmanuella A. dos S. Santana, Anne M. de Paula Canuto (2014) Filter-based Optimization Techniques for Selection of Feature Subsets in Ensemble Systems. *Expert Systems with Applications*, 41(4), p.p.1622-1631.
11. Mohammad Reza Bonyadi, Xiang Li, Zbigniew Michalewicz (2014) A Hybrid Particle Swarm with A Time-adaptive Topology for Constrained Optimization. *Swarm and Evolutionary Computation*, 18(10), p.p.22-37.
12. Saber M. Elsayed, Ruhul A. Sarker, Efrén Mezura-Montes (2014) Self-adaptive Mix of Particle Swarm Methodologies for Constrained Optimization. *Information Sciences*, 277(1), p.p.216-233.
13. Sarthak Chatterjee, Debdipta Goswami, Sudipto Mukherjee, et al. (2014) Behavioral Analysis of the Leader Particle During Stagnation in A Particle Swarm Optimization Algorithm. *Information Sciences*, 279(20), p.p.18-36.
14. Zahra Beheshti, Siti Mariyam Hj. Shamsuddin (2014) CAPSO: Centripetal Accelerated Particle Swarm Optimization. *Information Sciences*, 258(10), p.p.54-79.
15. Ketan Tamboli, Sunny Patel, P.M. George (2014) Optimal Design of a Heavy Duty Helical Gear Pair Using Particle Swarm Optimization Technique. *Procedia Technology*, vol. 14, p.p.513-519.
16. Amer Fahmy, Tarek M. Hassan, Hesham Bassioni (2014) Improving RCPSP Solutions Quality with Stacking Justification - Application with Particle Swarm Optimization. *Expert Systems with Applications*, 41(13), p.p.5870-5881.