Biologically Inspired Computing Technique for Optimizing Discontinuous Mathematical Functions

Sanjay Agrawal[1*], Pooja Tiwari[2]

[1*]School of Engineering & Technology, IGNOU, New Delhi-110068; sanjay.agrawal@ignou.ac.in

[2]Gyan Vihar University, Jaipur; pooja.tiwari03@gmail.com

Abstract

This paper explores the immense capabilities of the recently proposed biologically inspired soft computing technique named Particle Swarm Optimization for the optimization of non-convex, nonlinear and discontinuous mathematical functions; which primarily focus upon the attainment of the global optimum, despite of the existence of local multi-optimums in the vicinity. This technique uses an innovative distributed intelligent paradigm for solving optimization problems that originally took its inspiration from the biological examples like bird flocking and fish schooling. This is also a population-based optimization tool, which could be implemented and applied easily to solve various function optimization problems. But unlike Genetic Algorithm it uses only primitive mathematical operator and does not uses cross-over, mutation and reproduction. Potential of this technique is illustrated by implementing it on the well known bench mark mathematical problems which includes Rastrigins, Ackley, Alpine, and Schaffer's F6 functions.

Keywords: biological, genetic algorithm, PSO

1. Introduction

In few last years, it has been observed that optimization algorithms is not only very attractive by the research group but also by the person from industrial side. In the field of evolutionary computation (EC), inspiration for optimization algorithms basically comes in Darwin's ideas of evolution and survival of the fittest. This type of technique is usefull for an evolutionary proces. This is required where the purpse is to find out the answer solutions with help of of crossover, mutation, and selection which is dependenet on their quality wrt to the optimization problem. Evolutionary technique are very useful to solve the problen which is related to organization where industrial products come in picture, this is due that they have idea to solve such type of problem with non-linear limitation or restriction, different type of aims, and dynamic components-charaterstics that frequently comes in real world. This paper introduces such an algorithm called PSO and demonstrates its potential on real-world mathematical problems.

Swarm Intelligence (SI) is a new idea to produce distributed intelligent model for solving optimization problems that starts its a good idea from the biological examples by swarming, flocking and herding phenomena in vertebrates (Eberhart & Kennedy, 1995; Kennedy & Eberhart, 1995; Parsopoulos & Vrahatis, 2001). Particle swarm optimization has two methodologies. It may be more obvious are its ties to artificial life in general, and to bird flocking, fish schooling, and swarming theory in particular (Parsopoulos & Vrahatis, 2002; Eberhart & Hu, 1999).

It is also related, however, to evolutionary computation, and has ties to both genetic technique and evolutionary programming (Shi & Eberhart, 1998; Lis & Eiben, 1996). The initial ideas of James Kennedy (a social psychologist) and Russel Eberhart (an engineer and computer scientist) were essentially aimed at producing computational intelligence by exploiting simple analogues of social interaction (Zitzler, Deb, & Thiele, 1999)., rather than purely individual cognitive abilities. Their simulations were influenced by Heppner's works, which involve analogues of bird flocks searching for corn.

Swarm system is a rich source of novel computational methods that can solve difficult problems efficiently and reliably. When swarms solve problems in nature, their abilities are usually attributed to swarm intelligence; perhaps the best-known examples are colonies of social insects such as termites, bees and ants. One of the best-developed techniques of this type is Particle Swarm Optimization.

For applying PSO successfully, one of the key issues is finding how to map the problem solution into the PSO particle, which directly affects its feasibility and performance.

* *Address for correspondence:*

Sanjay Agrawal

School of Engineering & Technology, IGNOU, New Delhi-110068

sanjay.agrawal@ignou.ac.in

2. Particle Swarm Optimization

2.1 Swarms and Particles

This particular term i.e. swarm is used in accordance with a paper by Millonas. He articulated five basic principles of swarm intelligence:

a) Proximity principle
b) Quality principle
c) Diverse response
d) Principle of stability
e) Principle of adaptability

The principles four and five are the opposite sides of the same coin. The particle swarm optimization concept and paradigm presented in this paper seem to adhere to all five principles.

2.2 Overview of PSO Technique

The algorithm of PSO is initialized with a population of random solutions, called 'particles'. Each particle in PSO flies through the search space with a velocity that is dynamically adjusted according to its own and its companion's historical behaviors. The particles have a tendency to fly toward better search areas over the course of a search process. During flight, each particle keeps track of its coordinates in the problem space, which are associated with the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called 'pbest'. Another 'best' value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called 'lbest'. When a particle takes all the population as its topological neighbors, the best value is a global best and is called 'gbest'. The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its pbest and lbest locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward pbes*t* and lbest locations.

Each particle tries to modify its position using the concept of velocity. The velocity of each agent can be updated by the following equation:

$$v_i^{k+1} = \omega v_i^k + \psi_1 rand_1 \times (pbest_i - s_i^k) + \psi_2 rand_2 \times (gbest_i - s_i^k) \quad (1)$$

where v_i^{k+1} is velocity of agent *i* at iteration *k, ω* is weighting function, ψ1 and ψ2 are weighting factors, $rand_1$ and $rand_2$ are random numbers between 0 and 1, s_i^k is current position of agent *i* at iteration *k,* $pbest_i$ is the *pbest* of agent *i*, and *gbest* is the best value so far in the group among the *pbests* of all agents.

The following weighting function is usually used:

$$\omega = \omega_{\max} - ((\omega_{\max} - \omega_{\min}) / (iter_{\max})) \times iter \quad (2)$$

Where, $\omega_{\max}$ is the initial weight, $\omega_{\min}$ is the final weight, $iter_{max}$ is the maximum iteration number, and *iter* is the current iteration number. Using the previous equations, a certain velocity, which gradually brings the agents close to *pbest* and *gbest*, can be calculated. The current position (search point in the solution space) can be modified by the following equation:

$$s_i^{k+1} = s_i^k + v_i^{k+1} \quad (3)$$

2.3 Algorithm for Particle Swarm Optimization

1. Initialize the size of the particle swarm *n*, and other parameters.
2. Initialize the positions x_i and the velocities v_i for all the particles randomly.
3. While (the end criterion is not met) do:
4. Calculate the fitness value of each particle;
5. Update pBest, (If fitness value is better than the best fitness value (pBest) in past, set current value as the new pBest).
6. Update gBest, by choosing the particle with the best fitness value of all the particles as the gBest
7. For i=1 to n;
8. Calculate particle velocity according to equation (1).
9. Update particle position according to equation (3).
10. Next i.
11. End While.

Flowchart for PSO algorithm is shown in figure 1

2.4 Parameters of PSO

The role of inertia weight ω, in Eq. (2), is considered critical for the convergence behavior of PSO. The inertia weight is employed to control the impact of the previous history of velocities on the current one. Accordingly, the parameter ω regulates the tradeoff between the global (wide-ranging) and local (nearby) exploration abilities of the swarm. A suitable value for the inertia weight ω usually provides balance between global and local exploration abilities and consequently results in a reduction of the number of iterations required to locate the optimum solution. Initially, the inertia weight is set as a constant. However,

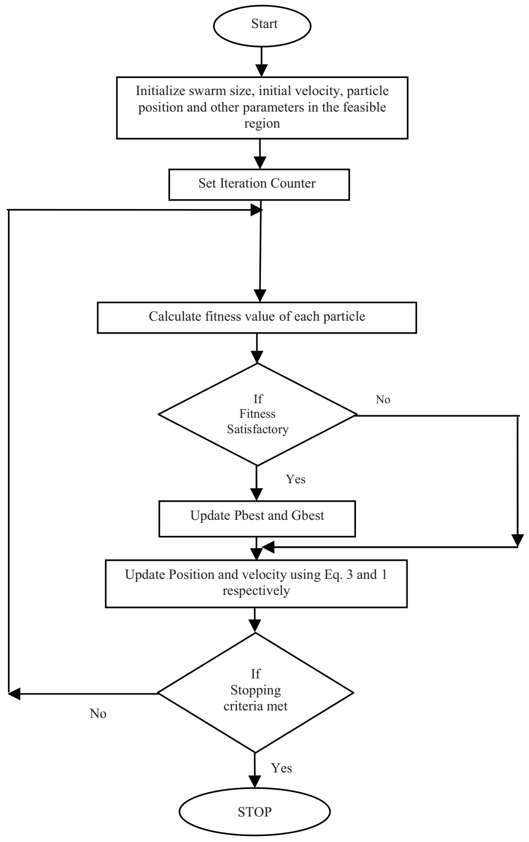
Start
Initialize swarm size, initial velocity, particle position and other parameters in the feasible region
Set Iteration Counter
Calculate fitness value of each particle
If Fitness Satisfactory
No
Yes
Update Pbest and Gbest
Update Position and velocity using Eq. 3 and 1 respectively
If Stopping criteria met
No
Yes
STOP

Figure 1. Flow chart for proposed PSO method

some experiment results indicates that it is better to initially set the inertia to a large value, in order to promote global exploration of the search space, and gradually decrease it to get more refined solutions.

The parameters ψ_1 and ψ_2, in Eq. (2), are not critical for the convergence of PSO. However, proper fine-tuning may result in faster convergence and alleviation of local minima. As default values, usually, $\psi_1=\psi_2=2$ is used, but some experiment results indicate that $\psi_1=\psi_2=1.49$ might provide even better results. Recent work reports that it might be even better to choose a larger cognitive parameter, ψ_1 than a social parameter, ψ_2 but with $\psi_1+\psi_2\leq 4$.

2.5 Boundary Conditions

Often in engineering applications it is desirable to limit the search to what is physically possible. Experience has shown that, constriction factors, and inertial weights do not always confine the particles within the solution space. To address this problem, the authors have imposed three different boundary conditions as shown in figure 2.

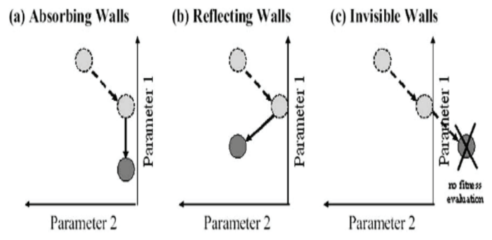

Figure 2. Boundary conditions for swarms

1) Absorbing Walls: When a particle hits the boundary of the solution space in one of the dimensions, the velocity in that dimension is zeroed, and the particle will eventually be pulled back toward the allowed solution space. In this sense the boundary "walls" absorb the energy of particles trying to escape the solution space.
2) Reflecting Walls: When a particle hits the boundary in one of the dimensions, the sign of the velocity in that dimension is changed and the particle is reflected back toward the solution space.
3) Invisible Walls: The particles are allowed to fly without any physical restriction. However, particles that roam outside the allowed solution space are not evaluated for fitness.

For nearly all-engineering applications, the computationally expensive portion of the algorithm is the fitness evaluation. The motivation behind this technique is to save computation time by only evaluating what is in the allowed solution space, while not interfering with the natural motion of the swarm. The results show that the "invisible walls" technique proves slightly better than other techniques.

2.6 End Criteria

There are several methods to determine the termination criteria. Various methods have been used by the various researchers; most common methods are listed below.

a) Maximum Number of Iteration: With this termination condition, the PSO ends when the process has been repeated a user-defined number of times. Although, the best result is not guaranteed, but this method is used most of the times. The reason is its simplicity and generality.

b) Number of Iterations without Improvement: In this case the optimization process is terminated after some fixed number of iterations if any improvement is not obtained. The number decided should be such that case of particles being trapped in local maxima (or minima) is eliminated.

c) *Minimum Objective Function Error:* The optimization process is terminated if the error between the obtained objective function value and the best fitness value (target value) is less than a prefixed threshold. At any time if a solution is found that is greater than or equal to the target fitness value, then the PSO is stopped at that point. This is useful when one has a very specific engineering goal for the value of the fitness function, and is not necessarily concerned with finding the "best" solution. In some cases if a solution is found to be better than the target fitness, then the solution is good enough and there is no reason to continue to run.

3. Simulation Results

We need to build fitness function, which uses the program being evaluated as the function in a PSO, and evaluates the performance of the resulting PSO on a training set of problems taken from the given class.

3.1 BenchMark Functions

This technique is tested on functions with many local minima like Ackley function, Rastrigin's function, Alpine function and Schaffer's F6 function.

3.1.1 Ackley Function F1 (Minimization)

$$20+e-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\sqrt{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$$

where, $-10 \le x_i \le 10$

Figure 3 and 4 shows the Ackely function and PSO convergence during iterative runs.

3.1.2 Alpine Function

$$0.1\sum_{i=1}^{n}x_i + abs\left(\sum_{i=1}^{n}x_i \sin x_i\right) \text{ where, } -10 \le x_i \le 10$$

Figure 5 and 6 shows alpine function and its convergence.

3.1.3 Schaffer Function F6 (Minimization)

$$0.5+\frac{\sin^2\left(\sqrt{x_1^2+x_2^2}\right)-0.5}{\left(1.0+0.001\left(x_1^2+x_2^2\right)\right)^2} \text{ where, } -100 \le x_i \le 100$$

Figure 7 and 8 shows Schaffer function and its convergence.

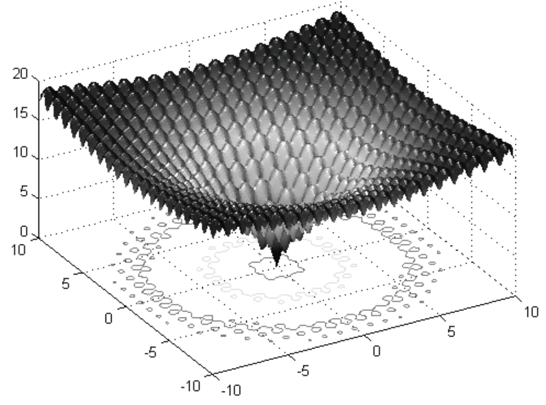

Figure 3. Ackley Function plot in 3-dimensions

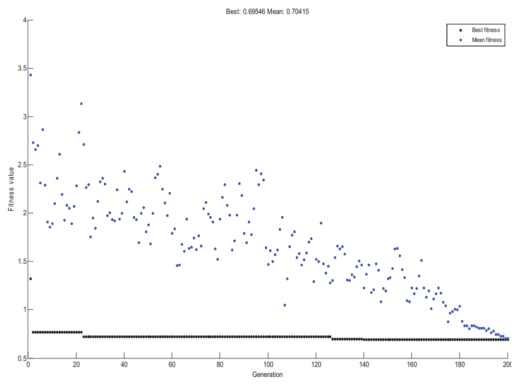

Figure 4. Swarm Convergence during iterative run for Ackley

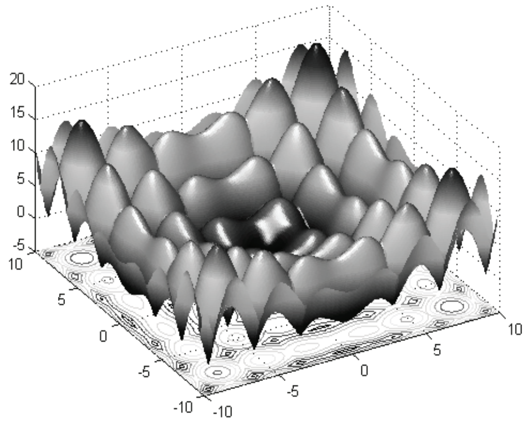

Figure 5. Alpine function plot in 3-dimensions

3.1.4 Rastrigin function F1 (Minimization)

$$\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)+10\right) \text{ where, } -5.15 \le x_i \le 5.12$$

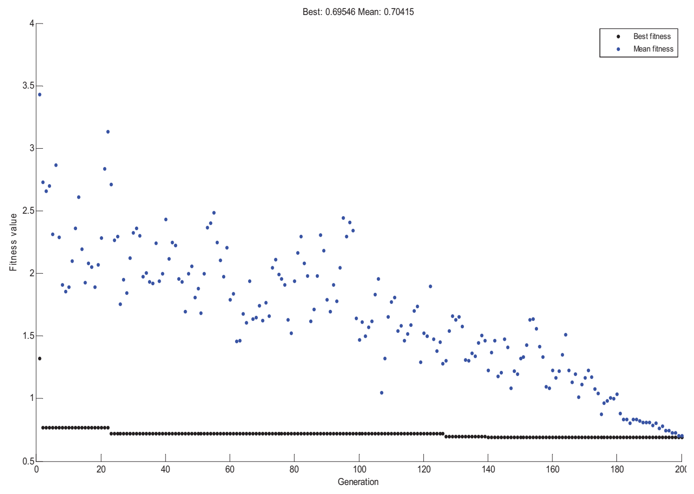

Figure 6. Swarm Convergence during iterative run for Alpine.

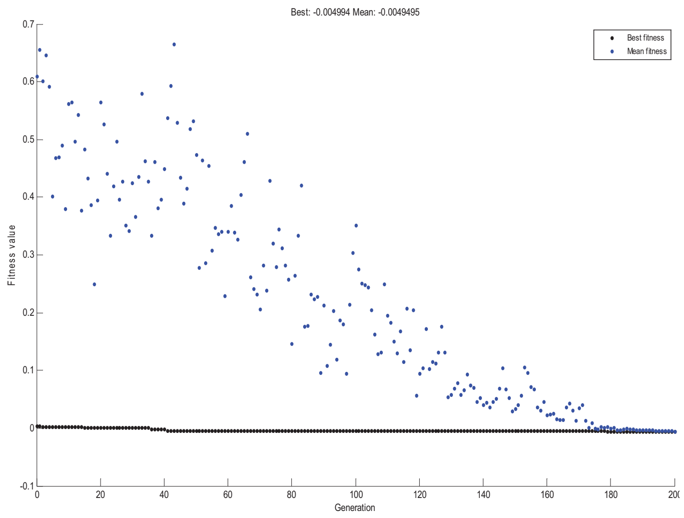

Figure 8. Swarm Convergence during iterative run for Schaffer F6.

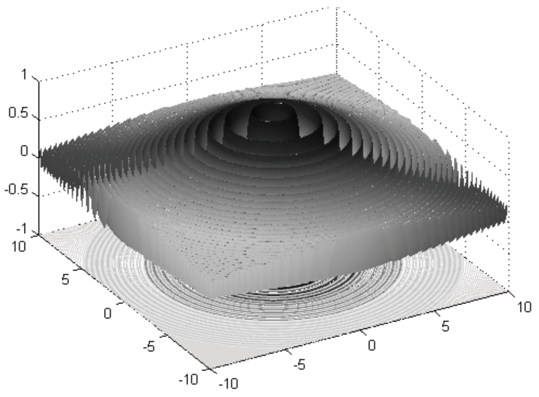

Figure 7. Schaffer Function F6 in 3-dimension.

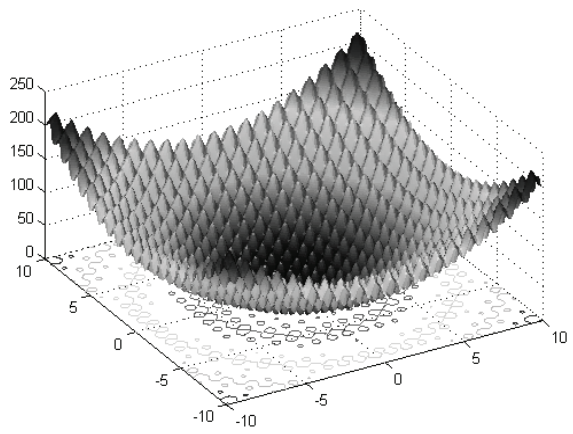

Figure 9. Rastrigin function plot in 3-dimensions.

Figure 9 and 10 shows Rastrigins function and its convergence using PSO minimization Technique.

Numerical simulations show that the proposed algorithm is very effective to deal with the multi objective optimization problems.

It is not the case in the proposed algorithm, only a small population size is needed to obtain the desired results. In addition, the proposed algorithm can be understood and performed easily because there is no operation such as ‘crossover” and “mutation” used in other evolutionary algorithms solving multi-objection problems.

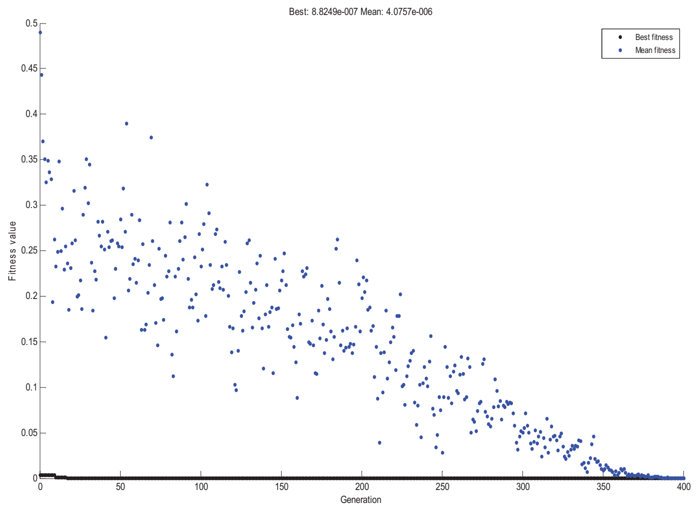

Figure 10. Swarm Convergence during iterative run for Rastrigin.

References

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium *on Micro Machine and Human Science*, 39–43.

Eberhart, R. C., & Hu, X. (1999). Human tremor analysis using particle swarm optimization. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999)*. 1927–1930.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *IEEE International Conference on Neural Networks (Perth, Australia)*, *4*, 1942–1948.

Lis, J., & Eiben, A. E. (1996). A multi-sexual genetic algorithm for multiobjective optimization. *Proceedings of the 1996 International Conference on Evolutionary Computation*, 59–64.

Parsopoulos, K. E., & Vrahatis, M. N. (2001). Particle Swam Optimizer in Noisy and Continuously Changing Environments. In Hamza, M.H. (Ed.). *Artificial intelligence and soft computing* (pp. 289–294). IASTED/ACTA Press.

Parsopoulos, K. E., & Vrahatis, M. N. (2002). Particle swarm optimization method for ‘constrained optimization problems. *Proceedings of the Euro-International Symposium on Computational Intelligence*, 214–220.

Shi, Y., & Eberhart, R. (1998). A Modified Particle Swarm Optimizer. *IEEE International Conference on Evolutionary Computation*. 69–73.

Zitzler, E., Deb, K., & Thiele, L. (1999). *Comparison of Multi Objective Evolutionary Algorithms: Empirical Results* (Tech. Rep. 70). Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland.