International Journal of Software Engineering and Its Applications
Vol. 9, No. 4 (2015), pp. 91-104
http://dx.doi.org/10.14257/ijseia.2015.9.4.11

An Exhaustive Survey on Nature Inspired Optimization Algorithms

Manish Dixit[1]*, Nikita Upadhyay[2] and Sanjay Silakari[3]

[1, 2]*Madhav Institute of Technology & Science, Gwalior, India*
[3]*University Institute Of Technology, RGPV,Bhopal,India*
[1]*dixitmits@gmail.com*, [3]*ssilakari@yahoo.com*

Abstract

Human being are greatly inspired by nature. Nature has the ability to solve very complex problems in its own distinctive way. The problems around us are becoming more and more complex in the real time and at the same instance our mother nature is guiding us to solve these natural problems. Nature gives some of the logical and effective ways to find solution to these problems. Nature acts as an optimizer for solving the complex problems. In this paper, the algorithms which are discussed imitate the processes running in nature. And due to this these process are named as "Nature Inspired Algorithms". The algorithms inspired from human body and its working and the algorithms inspired from the working of groups of social agents like ants, bees, and insects are the two classes of solving such Problems. This emerging new era is highly unexplored young for the research. This paper proposes the high scope for the development of new, better and efficient techniques and application in this area.

Keywords: *Nature inspired algorithms; Evolutionary algorithms; Swarm intelligence, Genetic Algorithm*

I. Introduction

Nature inspired computing is the computing which has its foundation in the biological components of the nature *i.e.*, humans and animals. Nature has four powerful features which are basic building blocks are self optimization, self healing, self learning and self processing. Nature as Self optimizer is that it can automatically manage its resources in an efficient manner to meet enterprise need. Nature as self healer is as the components of nature on seeing any problem finds a solution and come out of it. Self learning and self processing are two related terms. They go hand in hand and moved together. Nature and its components self processes the changing conditions in the environment learn from the past and present conditions to evolve in the changed environment in natural evolution. As the individuals of nature have the capability to evolve according to the changing environment so in present scenario it is indeed required that computers and their intelligence to learn and involve as per changing conditions and solve highly complex problems as nature does. To fulfill this desire, we want our algorithms to adopt the techniques and features from nature and become more effective and efficient too.

ISSN: 1738-9984 IJSEIA

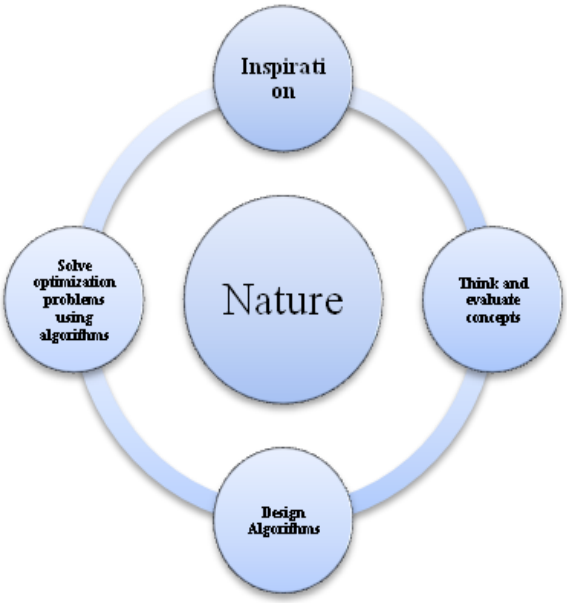


Figure 1. Flow Cycle of Whole Process

This paper is divided into four sections; the first section explains evolutionary algorithms which are further divided into three subsections namely genetic algorithms, genetic programming and evolutionary strategies. The second section describes swarm intelligence based algorithms which are further classified into three categories – particle swarm optimization, ant colony optimization and artificial bee colony optimization. The third section shows various merits and demerits of nature inspired algorithms and concluding remarks in the last section.

2. Evolutionary Algorithms

In the Origin of Species, Charles Darwin stated the theory of Natural Evolution. Over many stages of life, biological organisms develop according to the principles of natural selection like "Survival Of The Fittest" to attain some astounding form of accomplishment. The best example of natural evolution can be seen in generation of human beings. So if it works so admirably in nature, it should be interesting to imitate natural evolution and try to procure a technique which may solve existing search and optimization problems.

In Computer science, evolutionary computation [1, 14] had its foundation in natural evolution. Evolution computing is the common term for a domain of problem solving techniques based on principles of biological evolution. Evolutionary algorithms are well known and successful algorithms among nature inspired algorithms.
EAs can be subdivided as follows:

- Genetic algorithms
- Genetic programming
- Evolutionary strategies

A. Genetic Algorithms

Genetic algorithm is developed by J. Holland, K. DeJong, D. Goldberg in 1970's [2][4]. Genetic algorithm mainly lies on the idea of natural evolution. Genetic algorithms are meta heuristic algorithms which mean that it generates useful solutions to optimization

and search problems. It obeys the postulate of Charles Darwin's theory of Survival Of The Fittest. Due to its great performance in optimization, GA is considered to be a great function optimizer. In GA, firstly initial set of population is created from a random assemblage of solutions. Each solution is depicted through a chromosome. Solutions are not exactly the answers. Solutions can be visualized as an estimate possible path which will lead the system to the answer of the problem in more effective and fast manner as compared to other paths. Than each chromosome is evaluated. A value for fitness is decided on the basis of how close the solution actually is to solving the problem.

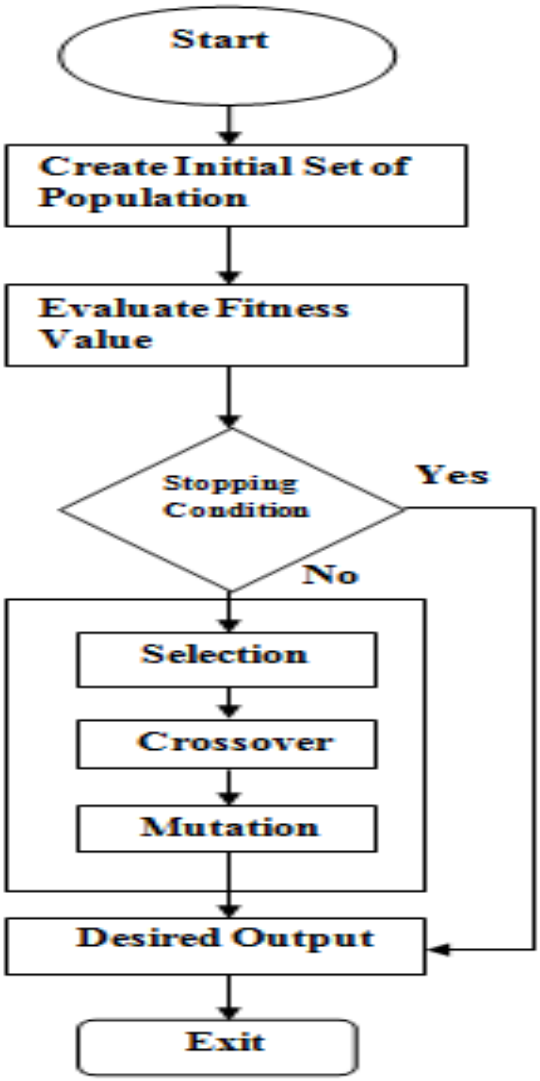


Figure 2. Flow Chart of Genetic Algorithm

GAs work with three operators:

a. Selection: Picking up of two chromosomes with higher fitness value from population for crossing.

b. Crossover: producing a child(a new better solution) by taking two parent solutions. Crossover is a recombination operator that proceeds in three steps:

- The reproduction operator chooses at random a pair of two individual strings for the mating.

- A cross site is then decided at random along the string length.
- Finally, the selected position values are swapped between the two strings following the cross site.

c. Mutation: For prevention of the algorithm to be trapped in a local minimum. Its use is to change the value of a random position in a string. Lastly the new generation which is evolved if contains a solution that produces an output to the problem that is very close to the desired result would be actual solution. If the problem does not reach to its solution *i.e.*, after whole procedure the solution is not obtained, then the new generation goes through the same procedure as their parents did. This will continue until a solution is achieved.

B. Genetic Programming

Genetic Programming was proposed by John Koza in 1992 [3]. Genetic programming is a representation of programming which adopts the concepts of natural evolution to solve a complicated problem. Genetic programming can be seen as a new version of the genetic algorithm. Genetic Programming is an automatic process for fabricating a working computer program from a high level complex problem statement. GP reach to this point of automatic programming by genetically procreating a population of computer programs using the postulates of Darwinian natural selection and nature inspired operations.

GP works in this way:

1. First arbitrarily create an initial set of population of independent computer programs made of functions and terminals.

2. Execute each program in the population and calculate fitness value for each program using a criterion decided in starting.

3. Select one or two independent program(s) from the population with using a probabilistic approach for fitness to take part in the genetic operations.

4. Create new independent program(s) for the population by applying the following genetic operations:

- **Reproduction:** Replicate the selected independent programs to the new population.
- **Crossover:** Construct new offspring program(s) for the new population by recombining arbitrarily chosen parts from the selected programs.
- **Mutation:** Construct a single new offspring program for the new population by arbitrarily mutating an arbitrarily chosen part of one selected program.

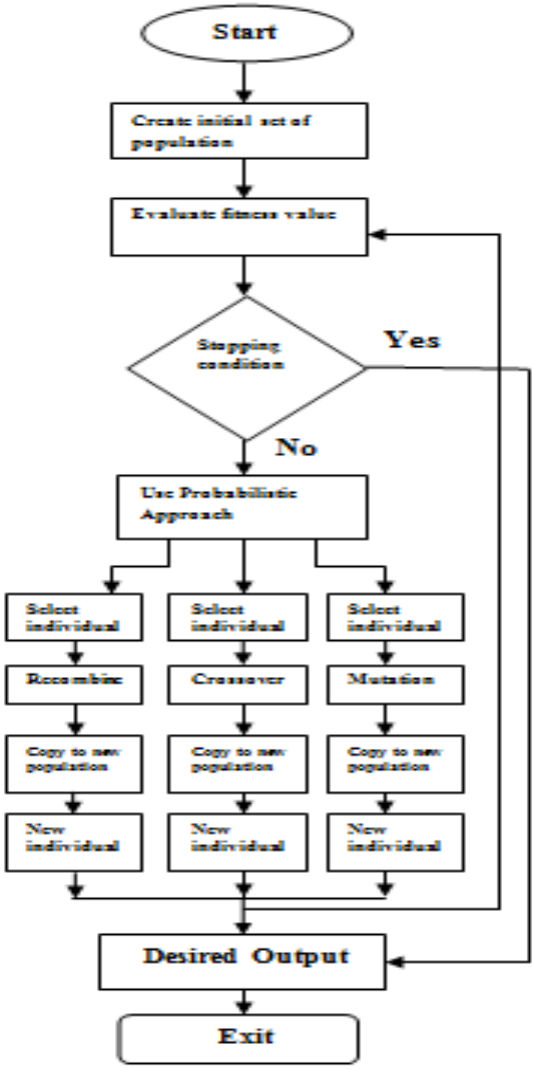


Figure 3. Flow Chart of Genetic Programming

5. After the stopping criterion is fulfilled, the single best program in the population created during the process is gathered and classified as the result of execution. By taking the union if we get the desired result, which is a solution (or approximate solution) to the problem. GP is a magnificent problem solver, a superb function approximator and an effective tool for creating functions to solve particular tasks.

C. Evolutionary Strategies

Evolution Strategies [5] was designed to solve technical optimization problem. Students of University of Berlin, Rechenberg, Schwefel and Bienert did experiments to randomly change the criterion which are defining the shape of the bodies by taking the idea from mutation. In this whole process, evolutionary strategy was developed.

Evolution strategies [25, 26], sometimes also referred to as evolutionary programming are search models inspired by the laws of natural evolution. They belong to the domain of evolutionary algorithms that conquer optimization problems by administering a iterative procedure with minute variations followed by selection. In each iteration (generation), a new offspring (solutions) are evolved from their parents (solutions already seen), their

fitness is calculated, and the better offspring are chosen to become the parents for the next generation selection and sampling in ES schemes can be characterized as:

(1+1)-ES: In this process, an independent solution is selected and further mutated and a comparison with the previous solution is done. The best between the two is chosen as the parent for next generation.

(μ+λ)-ES: In this process, μ parents are chosen from the generation and by recombining them, mutating them or both, λ offsprings are generated. Then μ parents and λ offsprings are combined to form set of (μ+λ) solutions. The best among all of them, μ solutions are kept as parents for next generation.

(μ,λ)-ES: In this process, μ parents are selected from the generation to produce λ offprings with number of offsprings greater than the number of parents. From λ offsprings only μ best solutions are considered to take part as parent in next generation. Previous participated parents are discarded fully.ES works in this way.

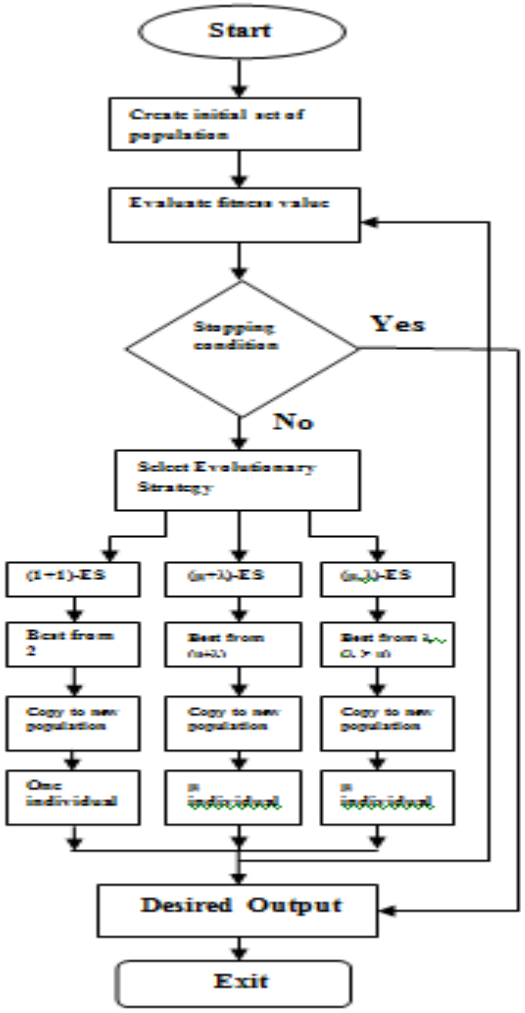


Figure 4. Flow Chart of Evolutionary Strategies

1. Initialize **μ**(parent) independent solutions to create the starting parent population.

2. Iteratively repeat process until terminating criteria is satisfied (here the maximum number of generations).

- Choose and recombine, mutate or both parents from the parent population to generate**λ**(child) offsprings.
- Choose the best **μ**(parent) individuals from the population of children and parents based on the values of the fitness function.
- Use the selected **μ**(parent) children as parents for the next generation.

3. Swarm Intelligence

Gerardo Beni and Jing Wang introduced it in 1989 in the context of cellular robotic systems. Swarm intelligence can be described as the collaborative conduct of a group of animals, especially insects such as ants, bees and termites, that are each following very basic rules but when seen in the field of computer science, swarm intelligence is a simulated way to problem solving using algorithms formed on the concept of self managed collective behavior of social insects. Bonabeau *et al.*, 1999 [9] gave a simple definition for better understanding of swarm intelligence *i.e.*, "the popular way of simple and common intelligence of social agents". Swarm intelligence is a emerging new domain that visualize intelligence as a method of communication between independent agents.

To mimic the performance evolved from a swarm several general postulates for swarm intelligence were defined:

1. Proximity Principle: The set of solutions should be able to execute space and time effective computations.

2. Quality Principle: The set of solutions should be able to interact with the quality features of its environment.

3. Diverse Response Principle: The set of solution should not commit its activity along excessively narrow channels.

4. Stability Principle: The set of solutions should not change their actions according to changing conditions.

5. Adaptability Principle: The set of solution should be able to change their actions when effective space and time computational price is needed.

A. Particle Swarm Optimization

PSO [10, 30] is inspired by nature and a computational search for optimization developed in 1995 by Eberhart and Kennedy based on the behaving aspects of birds flocking or fish schooling. PSO adopts the nature's general behavior, rapidly changing movements and interaction among social agents such as birds or fishes. PSO unites the personal experiences and experiences which are learned when working in group. This algorithm uses a number of agents (particles) that constitute a swarm moving around in the search looking for the best solution. Each particle in the search space alters its flying according to its own personal flying experiences as well as the flying experiences of other particles in the group. Each particle keeps an eye of its own coordinates in the solution space which are correlated with its best fitness value, attained by it so far. This value is called personal best, pbest. Another best fitness value that is considered by the PSO is the best fitness value obtained so far by any particle in the close range of that particle. This value is called global best or gbest. The main idea of PSO lies in speeding each particle toward its pbest and the gbest locations with a arbitrarily weighted speeding factor at each time step. Every particle seeks to alter its coordinates based on the some equation [32]

and parameters: The current position, current velocities, distance between the current position and pbest, the distance between the current position and gbest. The alteration of the particle's velocity and position can be mathematically described using following equations:

$$\mathbf{Vel_i^{k+1} = Vel_i^k + cw_1\, rand_1(\ldots) \times (pbest_i - sp_i^k) + cw_2\, rand_2(\ldots) \times (gbest - sp_i^k)} \quad \ldots\ldots (1)$$

$$\mathbf{x_i^{k+1} = x_i^k + Vel_i^{k+1}} \quad \ldots\ldots (2)$$

Where,

Vel_i^k: Velocity of agent i at iteration k,

cw_j: weighting factor for agent j,

x_i^k : particle position of agent i at iteration k

rand: uniformly distributed random number between range (0,1)

sp_i^k: current position of agent i at iteration k,

$pbest_i$: pbest of agent i,

gbest: gbest of the group.

PSO works in this way:

1. Initialize particles.

2. For each particle calculate fitness value.

3. If the fitness value is better than its personal best (pbest) then assign this value as its pbest.

4. Choose the particle with the best fitness value of all as gbest.

5. For each particle calculate particle velocity & position according to equation 1 & 2.

6. Repeat steps 2 to 5 until stopping criteria are met.

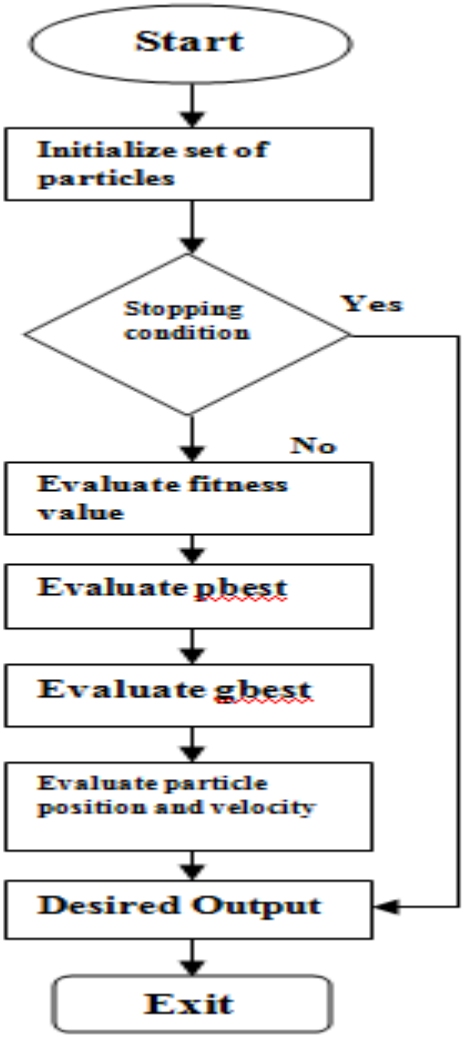


Figure 5. Flow Chart of PSO

B. Ant Colony Optimization

For finding food, ants start out from their colony and move randomly in all directions. Once a ant find food, it returns to colony and leave a trail of chemical substances called pheromone along the path. Other ants of the swarm can sense pheromone trails and move on the same path. The interesting point is that how often the path visit by ants is determined by the concentration of pheromone along the path. Since pheromone will naturally evaporate over time, the length of the path is also a factor. Therefore under these conditions, a shorter and best path will be chosen because ants moving on that path keep adding pheromone to it which makes the concentration strong enough against evaporation. As a result, the shortest and best path from colony to food emerges. This type of interaction between social agents is known as stigmergy. Stigmergy is a mechanism of indirect coordination between agents or actions. The law of stigmergy says that the indications left in the environment of current action simulate further the next subsequent actions [30]. This whole process of simulation in last creates a systematic way of performing some activity.

Ant colony optimization algorithm takes its inspiration from the real world of ant colonies to solve optimization problems. It was first introduced by Marco Dorigo in 1992 [11][29], originally was applied to travelling salesman problem and then applied later to various hard optimization problems. The main motive behind this algorithm is to solve

problem of searching an optimal path using weighted graph technique, also known as construction graph. Artificial ants are used to search best paths in the graph. Real ants deposit pheromone trail on their forward and return journey while artificial ants deposit pheromone trail only after solution has been constructed.
ACO algorithm works in this way:

1. Initialization: Initialize ACO parameters (*e.g.*, no. of artificial ants) and pheromone trails.

2. Construct Ant Solutions: A set of some artificial ants construct solutions from elements of a finite set of available solution components.

3. Daemon Actions: When solution set has been created, before increasing or decreasing pheromone values, some particular actions are to be taken according to the problem given. These actions are known as Daemon Actions. These actions vary problem to problem. The Daemon Actions are optional.

4. Update Pheromone: Update pheromone is the mechanism of increasing or decreasing the pheromone values along the path. The pheromone values are increased for good solutions and decreased for bad solutions. There is another procedure which is executed inside update pheromone action which is evaporation. The pheromone values are evaporated so that bad solutions which were learned earlier can be forgotten.

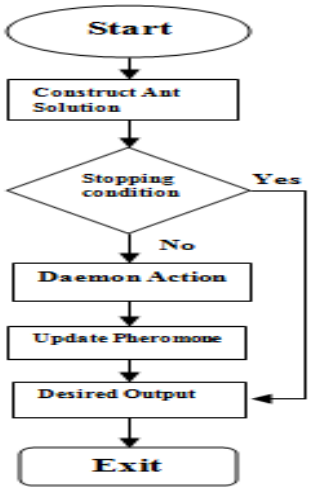


Figure 6. Flow Chart of ACO

C. Artificial Bee Colony Optimization

As the ants search their food using pheromones, similar food collecting behaviors are found in honey bees. Instead of pheromones, the algorithm relies on the foraging behavior of honey bees. The first step in this process is to sending bees in different directions to search for good quality food. When the bees have located the desired food source they return to their colony and tell other bees about the food source using a technique known as waggle dance. This dance tells other bees about three information *i.e.*, distance of food source from colony, the direction in which all other bees have to go and last the quality of food source. The bees are attracted to that bee only which has brought information about the best quality food source. This process gives the best food source.

Artificial bee colony (ABC), introduced by Karaboga in 2005 [12, 13] is the most popular algorithm based on bee colony optimization. In ABC the BCO's foraging

behavior is imitated. The Artificial Bee Colony optimization differs from a real Bee Colony Optimization because in ABC we use only scout bees and onlooker bees in equal quantity as initial set of solution (population). ABC based on approach of examining the behaviors of real bees on finding food and share this information of food sources to the bees in the area. This algorithm solves multidimensional and multimodal optimization problems.

ABC contains The Employed bees which are selected to search for food source, Onlooker bee that follows the paths provided by best employed bee and The Scout bee to found the lowest quality of food source and is now responsible for finding new food, the new food sources.

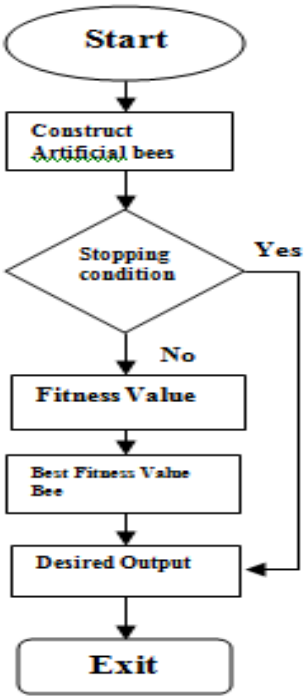


Figure 7. Flow Chart of ABC

ABC works in this way:

1. Initialization: Create an initial set of solutions in which scout bess will search for optimal slution.

2. Fitness Evaluation: Scout bees which are chosen to search the set for optimal solution, visit sites, then the fitness value is evaluated based on some criterion for those visited sites.

3. Evaluating Best Value: The bees which have higher fitness value are chose as "selected bees" and the sites which they visit are chosen for neighborhood search.

4. Iteration: If the optimal solution is not found, the remaining scout bees are sent to search for another solutions near to those sites which are chosen as best sites.

4. Conclusion

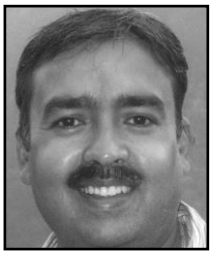
The nature inspired algorithm are flexible and work in changing environment to organize and grow accordingly. These algorithm quality perform in highly complex problems and can even show very good results when problem is not properly defined. These tends to find the best available solution in every changed environment and very good decision maker. Scalability is not a challenge. But in another side these Nature inspired systems are very hard to design as algorithms are inspired from nature and understanding nature fully is complex. Nature inspired systems do not adapt to real world system fully in terms of scalability and performance. Systems can work well in some domain but not in other. As systems are nature inspired, then not having proper knowledge of nature can affect the design of algorithm. So for effective results no ambiguity in data is required. These algorithms have the ability to self learn, self train, self organize and self grow. They can find best optimal solutions to complex problems using simple conditions and rules of nature. The scope in this field is very vast. This field is largely unexplored and therefore no limit on development. This paper gives a complete insight on various nature inspired algorithms based on evolutionary computation and swarm intelligence.

References

[1] Back, T. 1996: Evolutionary algorithms in theory and practice, Oxford University Press.
[2] J.H. Holland, Genetic algorithms and the optimal allocation of trials, SIAM J. Comput. 2 (2) (1973) 88–105.
[3] Koza, John R. 1992. Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA: The MIT Press.
[4] R.Shivakumar and Dr R.Lakshmipathi, "Implementation Of an innovative Bio Inspired GA and PS Algorithm for Controller design considering steam GT Dynamics" , IJCSI International Journal of Computer Science Issues, , Vol. 7, Issue 1, No. 3, January 2010.
[5] Beyer, H.G. and Schwefel, H.P. 2002: Evolution strategies. Natural Computing 1,3–52.
[6] Bremermann, H.J. " The evolution of intelligence. The nervous system as a model of its environment", Technical report, no.1, contract no. 477(17), Dept. Mathematics, Univ. Washington, Seattle, July, 1958.
[7] D. E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.
[8] Upeka Premaratne , Jagath Samarabandu, and Tarlochan Sidhu, ―A New Biologically Inspired Optimization Algorithm‖,Fourth International Conference on Industrial and Information Systems, ICIIS 2009,28-31 December 2009, Sri Lanka.
[9] Bonabeau, E., Dorigo, M. and Theraulaz, G.1999: Swarm intelligence. Oxford University Press.
[10] Kennedy, J.; Eberhart, R. (1995). "Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks. **IV**. pp. 1942–1948.
[11] Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics – Part B, 26, 29–41.
[12] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization 39 (2007) 459–471.
[13] Dervis Karaboga, Bahriye Akay. "A comparative study Of Artificial Bee Colony algorithm", Applied Mathematics and Computation 214 (2009) 108–132.
[14] J.E. Smith , Agoston E. Eiben, "Introduction to Evolutionary computation" Alex M. Andrew, Robotica, Vol. 22, 2004.
[15] David B. Fogel, "Evolutionary Computing", IEEE Press 2002.
[16] Darrel Whitely, A Genetic Algorithm Tutorial, Co mputer Science Department, Colorado University.
[17] Bill P. Buckles and Frederick E.Petry, "Genetic Algorithms" IEEE Press 1992.
[18] Back, T., Schwefel, H.P. "Evolutionary computation: an overview", Evolutionary Computation, 1996, Proceedings of IEEE International Conference on, pp 20-29, 1996.
[19] Whitley, D., "A Genetic Algorithm Tutorial", Computer Science Department, Colorado State University, whiteky@cs.colostate.edu.
[20] Abbott, R. (2005), "Challenges for Bio-inspired, Computing", The Proceedings of The BioGEC Workshop, GECCO, New York: ACM, pp. 12-22
[21] J.E. Smith , Agoston E. Eiben, "Introduction to Evolutionary computation" Alex M. Andrew, Robotica, Vol. 22, 2004
[22] GENETIC ALGORITHM, http://en.wikipedia.org/wiki/Genetic_algorithm [Online].2015.

[23] GENETIC ALGORITHMS http://www.doc.ic.ac.uk/~nd/surprise_96/ journal/vol1/hmw/article1.html [Online].2015.
[24] GENETIC PROGRAMMING, http://en.wikipedia.org/wiki/Genetic_programming [Online] 2015.
[25] Kaisa Miettinen, P. Neittaanmaki, Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, GE ,John Wiley & Sons, Inc. New York, NY, USA© 1999, ISBN:0471999024.
[26] Hans-Georg Beyer, Hans-Paul Schwefel, Evolution Strategies A Comprehensive Introduction, Natural Computing 1:3-52,2002 © 2002 Kluwewr Academic Publishers.
[27] Daan Wierstra, Tom Schaul, Jan Peters, Juergen Schmidhuber, Natural Evolution Strategies, http://www.idsia.ch/~daan/papers/cec08.pdf
[28] SWARMINTELLIGENCE. http://en.wikipedia.org/wiki/Swarm_intelligence. [Online] [2015].
[29] ANTCOLONY OPTIMIZATION. http://en.wikipedia.org/wiki/Ant_colony_optimization_ algorithms.[Online][2015].
[30] PARTICLE SWARM OPTIMIZATION. http://en.wikipedia.org/wiki/Particle_swarm_ optimization.[Online][2015].
[31] STIGMERGY http://en.wikipedia.org/wiki/Stigmergy [Online][2015]
[32] PARTICLE SWARM OTIMIZATION http://www23.homepage.villanova.edu/ varadarajan.komanduri/PSO_meander-line.ppt[Online][2015]
[33] Rashmi A. Mahale, Prof.S.D.Chavan, A Survey : Evolutionary and Swarm Based Bio-Inspired Optimization Algorithm, IJSRP, Vol-2, Issue-12, December 2012, 1-6.
[34] Binitha S, S Siva Sathya, A Survey of Bio inspired Optimization Algorithms, IJSCE, Vol-2, Issue-2, May 2012, 137-151
[35] Mehrdad Dianati, Insop Song, and Mark Treiber, An Introduction to Genetic Algorithms and Evolution Strategies, 200 Univ. Ave. West, University of Waterloo, Ontario, N2L 3G1, Canada

Authors



Manish Dixit received the B.E. in Computer Technology from Barkatullah University, Bhopal and M.E. in Communication Control and Networking from MITS, Gwalior In 1993 and 2006 respectively. He is pursuing his PhD from Rajiv Gandhi Technical University, Bhopal. He is currently working as Reader in the Department of Computer Science and Information Technology, MITS, Gwalior, India .He has presented various research papers in National and International Conferences and Journals. He is a Member of CSI, IETE, IEEE.

Sanjay Silakari is the Dean in faculty of CS/IT and the chairman of board of studies, CSE, RGPV, Bhopal. He is also a professor and Head of Department of CSE in University Institute of Technology of RGPV. Dr. Silakari has more than two decades of teaching and administrative experience and has guided several students for their doctoral and master studies. He has several research publications to his credit in different reputed national and internal conferences and journals. He is a life member of ISTE, CSI, IAENG and a member of IEEE and ACM.