International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 24 (2017) pp. 13981-13989

Bio-inspired Swarm Intelligence Algorithms –A Systematic Analysis

Manu Rajan Nair,
Research Scholar, Department of Computer Applications,
Bharathiar University, Coimbatore-641046, Tamilnadu, India.

Orcid : 0000-0001-5823-2478

Dr.T.Amudha,
Assistant Professor, Department Of Computer Applications,
Bharathiar University, Coimbatore-641046, Tamilnadu, India.

Orcid :0000-0002-0089-5831

Abstract

Objectives: The paper reviews and compares relative strengths and weaknesses of four algorithms in Swarm intelligence namely Particle swarm optimization, Ant colony optimization, Bacterial foraging optimization and Firefly.

Methods and Statistical analysis: Swarm intelligence inspired by nature constitutes one of newest methods used for optimizing solutions for computing problems. The meta-heuristics extracted from biological phenomenon specifically behavior of colonial organisms has led to several interesting algorithms. These algorithms are analyzed by dividing each of them into their common denominator constituents, namely Environment, Components, Start and End Configurations and Iteration engine. Further analysis is based on resemblances and differentiation among these elements.

Findings: It was found that the environment component itself can be sub-typed based on movement freedom, presence of solutions in environment. The component element can be sub-typed based on solution capacity and memory capacity. The start and end configuration can be sub-typed based on the usage of random initialization functions and generation count limit. The iteration engine shows greatest variation and be sub-typed based on use of various functions such as culling, cloning and randomization. Thereby this paper seeks to enhance the readers understanding of these algorithms in particular and swarm intelligence algorithms in general.

Application and Improvements: The common denominator constituent elements can be used to suggest subtypes for further detailed classification of the algorithms.

Keywords: Swarm intelligence, Bio-inspired techniques, Algorithm analysis, Algorithm behaviour comparison.

INTRODUCTION

Normal polynomial computational problems can be solved by applying reasonable amount of resources such as time and space. An increase in problem size leads to polynomial increase in amount of resources needed, such problems are called solvable problems. There is another class of problems that can be solved using same approach called hard problems. Polynomial problems can be solved faster by increasing the speed processing entity. This approach has led to development of faster physical computational elements such as high speed processors and memory. Unfortunately this brute force approach of solving problems is ineffectual in the case of hard problems due to the exponential nature of resource requirement. Polynomial programming requires a single processing entity to solve a single problem, whereas actual limitations of hardware limit their usability. But to solve hard problems is to use multiple processing entities or swarms. Parallel processing provides mathematical solution to such problems.

Researchers have been looking to optimize parallel processing by taking inspiration from nature. This has led to development of nature inspired algorithms [1, 2, 3]. Nature is observed to solve fundamentally complex problems by relatively simple solutions. The algorithmic interpretations of these solutions can be used to solve computational problems. Nature consists of entire set of all, describable phenomenon, in the universe. They include physical, chemical, and biological phenomenon. A special subcategory of nature inspired algorithm takes inspiration from biological phenomenon and bio inspired algorithm. BIA (Bio Inspired Algorithm) [3, 4, 5] is often the survivor of Darwinian evolution pressure, the fittest and most suitable solution to given problem. This means that such solutions are efficient, elegant, using component, compartmentalized, discrete, survivable, and adaptable by design.

One of most important biological phenomenon is collective behavior shown by simple components leading to emergent behavior. For example, ants in ant colony are simple components which work together behaving like single group for advancement of the colony as a whole. This social behavior is the model for developing a subset of BIA, Swarm intelligence algorithms [5, 6, 7, 8].The defining characteristics of SIA (Swarm Intelligence Algorithm) are simple components that can act independently, share information

among components, taking decentralized collective decisions, to further common interest and achieve a particular goal. SIA acts within a predefined solution space which is assumed to contain the best possible solution.

The components start with initial random position, symmetric or asymmetric, within the search space. The components then iterate, searching for the best solution called the Global optima [6, 7, 9]. In between, components may exchange information, or change configuration depending on the algorithm used. Configuration changes may use cloning, reduction, expansion, culling, switching or randomizing methods to change both local as well as global topology of the system. Nested iterations are often used wherein a fixed number of local loops updates by the components, followed by a global loop update by the system. The iterations end upon attaining a particular maturity value or termination condition. In addition to above common characteristics shown by most SIA, individual SIA have additional defining peculiarities.

REVIEW OF SIAs

The paper seeks to introduce the various Bio inspired Swarm intelligence Algorithms by pointing out the similarities and differences between them, thereby creating a better understanding about these algorithms. Though the algorithms under consideration may differ in sources of inspiration, framework and implementation, they still have several similarities [5, 7, 10, 11].All SIA under discussion share the following elements: (i) A Dimensioned environment or Search space, (ii) Mobile components that move within (i), (iii) Global Start with end configuration and (iv) Iteration engine that controls both (i) and (ii). It must be noted that the actual solution to the problem being solved depends on the SIA, it may be a point or plane in (i), a grouping of (ii) or a global configuration involving both (i) and (ii). It may also be noted that the application areas of the SIA may differ in terms of dimensionality and complexity. The following SIAs will be discussed based on their elements namely Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Bacterial Foraging Algorithm (BFOA) and Firefly Algorithm (FA).

Particle Swarm Optimization

By observing the flocking behaviour of birds, J. Kennedy and R. Ederhart proposed PSO algorithm [1, 5, 10, 28, 30] in 1995. Flocks of birds fly around in large swarms looking for food, whenever a bird locates a food source, all other birds in the flock move towards the source. The individual birds in the flock communicate with other birds thereby allowing all the flock to converge on the food source. The larger the number of birds in a flock, the better is the chance that they can locate good food sources in an area in a shorter time, thereby increasing the success rate of the flock as a whole. The PSO algorithm mimics this behaviour, in which individual components of a group work together to optimize the problem solving efficiency of the whole.

The algorithm has the constituent elements as mentioned in the following section.

Elements of PSO

1) Environment

   It is an N dimensional space with unrestricted movement allowed. It contains the solution space or search space which contains a set of all possible solutions.

2) Components or Particles

   They are particles that move through environment, each are having a position X, a velocity V and memory for storing best position of that particle. It can be vector in N-dimensional Cartesian space.

3) Start Configuration

   Initially components are given random positions and velocities. A fitness value function is used to find the best solution for that particle, often denoted as pbest [5, 8, 11].

4) End Configuration

   The stopping criteria are fulfilled due to following conditions; first if it exceeds maximum number of iterations and second if global fitness value reaches limiting conditions.

5) Iteration Engine

   After initialization, the system enters iteration phase where positions and velocities of individual particles are updated. Within iteration, the velocity of each particle is updated depending on its previous velocity, current position, best position in the current context, and optionally random or predetermined weights. The weights may be social factors so as to control expansion or reduction functions. This controls, how wide the particles move with respect to the local or global environment. Position of particle is updated by adding its updated velocity to its current position, causing a translation to new position.

A fitness evaluation is conducted by each particle updating its best solution, using a fitness function with respect to global best historical solution denoted as gbest. The iteration engine evaluates two sets of positional values, stored in its memory using fitness functions. The first set contains global best position for the system. The second set is unique to each particle, containing that particles history of best positions. The former is denoted as gbest or global best, the latter is denoted as pbest or particle best. A common mechanism within iteration is to determine best solution, both global and current

particle, is better or worse, than newly computed one replacing the current with new value if better or else, current values are maintained.

For PSO system [4, 5, 8, 12] consisting of N particles, iterating over T iterations, each particle can be expressed as follows, where x is position, v is velocity, b is best position, i= 0 to N-1 and t= 0 to T-1

$$P_i(t) = \{x_i(t), v_i(t), b_i(t)\} \quad (1)$$

The swarm can be represented as follows where G is global best of all b_i (t)

$$S(t) = \{G(t), P_i(t)\} \quad (2)$$

Velocity updating equation is given by

$$v_i(t) = v_i(t-1) + c_1 r_1(b_i(t-1) - x_i(t-1)) + c_2 r_2(G(t-1) - x_i(t-1)) \quad (3)$$

Position updating equation

$$x_i(t) = x_i(t-1) + v_i(t) \quad (4)$$

Memory updating equation

$$v_i(t) = Best(b_i(t), x_i(t)) \quad (5)$$

After initialization using Eq. (1) by setting random values, Eq. (5) is run to set $b_i(t)$ for each particle. Then the iteration engine recursively executes steps in eqs. (3), (4) and (5) for each particle .It then runs Eq. (2). The recursion is continued until one of end configurations are encountered.

Applications of PSO

PSO is used in Power System Optimization problems [1, 2, 31], Edge detection in noisy images [2] ,finding optimal machining parameter [28], assembly line balancing in production and operations management [2, 4, 33],various scheduling problems [28] ,vehicle routing problems ,prediction of tool life in ANN [4, 7] ,multi objective dynamic , constrained and combinatorial optimization problems [7] ,QoS in adhoc multicast ,anomaly detection ,colour image segmentation ,sequential ordering problem , constrained portfolio optimization problem [11] ,selective particle regeneration for data clustering [11], Extracting rules from fuzzy neural network [29] , machine fault detection [4] ,Unit commitment computation [12] ,Signature verification [12],Multimodal biomedical image registration and the iterated Prisoners Dilemma, classification of instances databases [14], feature selection, and web service course composition.

Ant Colony Optimization

Ants and ant colonies are one of nature's typical examples of a swarm based problem solver. Marco Dorigo in 1992 proposed the ACO algorithm [5, 8, 15, 16, 17] drawing inspiration from this. Ants start from their nest and randomly search for food sources leaving pheromones in its path; on locating one it lays a pheromone trail back to its nest. Any other free ant that happens upon this pheromone trail picks it up and follows it to the food source while adding pheromones to the path. As more ants use the trail to the food source, the trail represents the best solution for reaching the food source from the nest. ACO algorithm consists of independent components that try to find the best path to a target from a starting point, both points existing in the same environment. The pheromone trail laid by each component represents that component's solution or path for the problem. A greater number of components using the same trail signify the higher quality of that particular path or solution.

Elements of ACO

1) Environment

   It is a Construction graph with nodes representing domain points and edges representing relations between nodes. Movement is restricted to only between connected nodes.

2) Components or Ants

   Ants move through environment, each keeping track of path traversed from its starting point called, Historic path memory (HPM) [8, 11, 14, 15]. Ants deposit pheromones at edges, quantity of pheromone being determined heuristically. Pheromones are deterministically degrading weighted values. The degradation rate is exponential. Ants traverse the environment choosing paths based on existing pheromone levels, exploring all possible nodes. The list of nodes and order of traversal is stored in ant's HPM.

3) Start Configuration

   All components are distributed randomly among nodes, depending on the problem domain under consideration. Some nodes may be specifically designated, for example, start node and end node. All edges in the environment are given an initial equal pheromone level.

4) End Configuration

   Two possible end configurations can exist; (1) Convergence and (2) Termination condition including maximum iteration.

5) Iteration Engine

   After initialization, each ant traverses the environment, moving to next node based on state transition rule. Once the complete traversal is done by all ants, the pheromone updating initiates, updating all

the pheromone quantity on all edges based on fitness function. These steps are repeated until end configuration is encountered. For each ant the path traversed by it is stored in its HPM. At the end of iteration, it uses this information to update pheromone levels of all edges in the environment. In the next iteration, the ant uses transition probability to determine the traversal order that it undertakes. Transition probability [5, 8, 15, 16] depends on pheromone levels of edges and inverse of directed weight between nodes. At the end, the iteration convergence check is made based on number of ants and the path stored in each ants HPM. If a predetermined significant percentage of ants have similar HPM, then convergence check is passed and algorithm succeeds.

For n nodes and m ants, The Transition Probability Equation from node r to node s for the k^{th} ant

$$P_k(r,s) = \begin{cases} \frac{[\tau(r,s).[\eta(r,s)]^{\beta}}{\sum_{u \in J_k(r)}[\tau(r,u).[\eta(r,u)]^{\beta}}, if\ s,u \in J_k(r) \\ 0, \end{cases} \quad (6)$$

where r is the current node, s is the next node ,τ (r,s) is the pheromone level between node r and node s, η (r,s) = δ (r,s)-1 is the inverse of the distance δ (r,s) between node r and s , J_k (r) is the set of nodes that remain to be visited by the k^{th} ant positioned on node r, and β is a parameter determining the relative importance of pheromone level versus distance. u is total number of remaining nodes to be visited.

Updating pheromone level between nodes uses the following equations given below

$$\tau(r,s) \leftarrow (1-\alpha).\tau(r,s) + \sum_{k=1}^{m} \Delta\tau_k\ (r,s) \quad (7)$$

$$\Delta\tau_k(r,s) = \begin{cases} \frac{1}{L_k}, \text{if}(r,s) \in \text{route done by ant k}, \\ 0, \text{otherwise}, \end{cases} \quad (8)$$

Where $\Delta\tau_k$ (r, s) is the pheromone level laid down between nodes r and s by the k^{th}ant, m is the number of ants, L_k is the length of the route visited by k^{th} ant, and $0<\alpha<1$ is a pheromone decay parameter.

Applications of ACO

ACO is used to solve TSP Problem, Quadratic Assignment Problem (QAP) [2], Job-Shop Scheduling Problem [2], Dynamic Problem of data networking and routing [2, 3], a shortest path problem where properties of system such as node availability vary over time, continuous optimization and parallel processing implementation vehicle routing problem [7], graph coloring and set covering, agent based dynamic scheduling [32], digital image processing [8], classification problem in data mining and protein folding problem [12].

Bacterial Foraging Optimization Algorithm

Bacterial Foraging Algorithm [1, 2, 6, 11, 19, 20, 21, 22, 27] was developed by Kevin.M.Passino in 2002, based on foraging behaviour of microscopic single cell organisms like bacteria. Bacteria are simple organisms which uses a limited set of actions in order to survive. These actions are called chemo taxis. It consists mainly of tumbling and running. Running describes movement of bacteria in single direction and tumbling describes change of direction. The survival of bacteria involves foraging for food, avoiding unfavourable environments and maximizing chances for finding better sources of food. The environment of bacteria is chemical in nature, made up of mixed nutrients and noxious or harmful substances. These chemicals exist as gradient, both positive and negative. The bacteria seek positive gradient for nutrients and negative gradient for noxious substances. It swims up the positive gradient for nutrients and swims down, negative gradient for noxious substances. In neutral media, it randomly searches for gradient.

A typical example of bacteria is Escheria Coli bacteria with diameter 1μm and length of 2 μm. The primary locomotion method or organ used is flagella. It has six rigid flagella which operates like propellers of ship of which operates at 200 rpm pushing the bacteria at up to 20 μm per second. Another feature of bacteria is the natural selection based evolution where only the bacteria that successfully survive are able to reproduce. This cloning process allows successful bacteria to make a copy of it, which occupies same location as parent. The success of bacteria is determined by its ability to locate best sources of food and avoidance of harmful materials. The unsuccessful bacteria are the ones that are unable to collect enough food to reproduce or even to survive, since movement of bacteria involves expending energy that it collects from food. All actions, taken by a bacteria can be seen as a practical expression of cost minimization or effect maximization function. So it can be summarized that at the end of given period of time within a limited environment with a fixed no of randomly distributed bacteria, only successful ones and the progenies will remain. Majority of which will be located in areas in the environment with highest gradient level, for nutrients and lowest for noxious substances. This behaviour has inspired the development of Bacterial Foraging Optimization algorithm where by the chemo taxis, survival and dispersal behaviour is simulated by artificial bacterium.

The algorithm consists of following three actions namely Chemo taxis, Reproduction-Elimination and Dispersal. Chemo taxis involves two types of steps which are running-swimming and tumbling. Running or swimming involves bacterium moving in a single direction continuously for a fixed amount of time or distance called run length or step size. Tumbling involves change of direction. The direction may be random but definitely different from earlier direction and backtracking is avoided. Reproduction is a cloning function that creates an exact copy of the bacterium in place, called

child. Other than designation there is no discernable difference between parent and child. Elimination consists of Culling function that reduces or eliminates a part of a given population of bacteria based on preset probability. Dispersion consists of randomization function that spreads the remaining bacteria around the environment at random locations.

Elements of BFOA

1) Environment

It is an N dimensional unrestricted space, data points are expressed as gradient, both positive and negative. Environment [16, 19, 27] may be continuous but also episodic. Environment may contain only food sources or may contain both food sources and noxious substances.

2) Components or Bacteria

The components may be randomly distributed or may start at extreme points of environment. Each bacterium is capable of chemo taxis, and reproduction. In the chemo taxis mode it can run or tumble based on three conditions [20, 23]; (1) in neutral medium, and it behaves randomly by using running and tumbling. This is done to find favourable gradient, (2) On encountering positive gradient, bacteria swims up that it executes a fixed number of run actions in direction of increasing gradient (food seeking behaviour) and (3) On negative gradient, it swims down a negative gradient. This is done to avoid unfavourable gradient (avoidance behaviour).

The component uses a fitness function or cost function to determine, the direction and mode of movement in the next step in chemo taxis mode. Run length or step size determines duration for this mode. A memory is used to store previous result of this function which if worse than present one then it is replaced. This serves as an input for determining if next chemo tactic step is either run or tumble.

θ^i (j, k, l) represents position of i^{th} bacteria at chemo tactic step j, reproduction step k, dispersal or elimination step l. J (i, j, k, l) represents fitness function for i^{th} bacteria at each step. J_{last} represents previous best J value stored in memory of component or bacteria in memory of component or bacteria. The result of J (i, j, k, l) may be a run vector in same direction as previous movement of component or tumble vector which may be random direction.

3) Start Configuration

Initially the components or Bacteria are randomly distributed in environment.

4) End Configuration

There are three types of end configurations possible such as (1) Maximum iterations, (2) Maximum number of generations or reproductive steps and (3) Generation exhaustion where no new generations are produced.

5) Iteration Engine

Considering maximum values for number of generations, K, Maximum number of Bacteria, S, where i = 1, 2……S, the maximum number of iterations is determined by limit values of j and l. The constant values, probability of elimination p_{ed} and C (i) denotes run length unit.

Iteration engine has the following nested loops [19, 20];

(1) Chemo taxis loop: In the first phase of iteration, the chemo taxis, whether either run or tumble to be taken by each bacteria is determine.

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + c(i)\varphi(j) \qquad (9)$$

Where ϕ (j) determines whether action to be taken is run or tumble.

Condition (1)

If J (i, j+1, k, l) is better than J_{last}, where J_{last} = J (i, j, k, l), then run is executed with the same direction as the previous ie, θ^i (j, k, l). Here ϕ (j) is dependent on the magnitude of food source in the immediate environment of the bacteria i. By comparing with previous value the iteration engine can determine whether the bacteria are moving to favourable gradient.

Condition (2)

If J (i, j+1, k, l) is worse than J_{last} , then tumble is executed. Tumble consists of random vector in the range [-1, 1]. This is done when the iteration engine encounters a bacteria moving through an unfavourable gradient ,then direction of bacteria is changed and a run of magnitude C(i) ,is executed in that new direction.

At the end of chemo taxis step, all the bacteria are translated to new position. The iteration engine moves all bacteria in a favourable gradient, up the gradient and all the bacteria in an unfavourable gradient are given a new direction. All the bacteria store their best fitness value as J_{last} .

(2) Reproduction loop: The iteration engine first determines a health value for each bacterium and orders them in terms of decreasing health value.

$$J^i_{health} = \sum_{j=1}^{j_{max}}(i,j,k,l) \qquad (10)$$

The bacteria with least health value are culled and remaining bacteria are cloned. The cloned bacteria share same position in the environment as their parents.

(3) Elimination and dispersal loop: For each bacterium, second round of culling and randomization is done by iteration based on predetermined probability value. Another check is made to ensure that number of bacteria remain constant. This is done by randomizing position of live bacteria whenever a bacteria is eliminated. These three nested loops are run until end configurations are encountered.

Applications of BFOA

BFOA is used in harmonic estimation problem in power systems [2], inverse airfoil design, optimal power system stabilizers design [2],tuning the PID controller of AVR [5] ,an optimal power flow solution [11] ,machine learning, job shop scheduling benchmark problems ,parameters of membership functions and the weights of rules of a fuzzy rule set [16] ,transmission loss reduction ,implemented as the parameter estimation of non linear system model(NSM) for heavy oil thermal cracking [20],evaluation of independent components to work with mixed signals[23] , solve constrained economic load dispatch problems ,null steering of linear antenna arrays by controlling the element amplitudes and in multi objective optimization [24].

Firefly Algorithm

Firefly algorithm [1, 3, 7, 25, 26, 27] was proposed in 2008. It was inspired by intra population, species specific, communication strategy of fireflies, particularly, reproductive behaviour between them. Fireflies have unique adaptation, an organ for producing light by a process bioluminescence. This is an evolution of communication method using biochemical like pheromones seen in other insects. The primary purpose of bioluminescence as used by fireflies is to signal reproductive fitness of that individual. It is in nature that fireflies that are able to produce more intense light; are able to attract more mates, increasing reproductive success of that individual. This positive genetic feedback system has led to evolution of fireflies with light producing ability of highest order. It is observed that even though the primary criteria in attraction between fireflies are light intensity, there is second contradicting parameter is the distance between light producer and observer as the intensity by physical laws decreases with distance. This means that a distant brighter source may be at disadvantage compared to a dimmer but closer source. The brighter the light emitted by a firefly as observed by other fireflies, the more likely they are to move towards that firefly. This leads to convergence of various individuals of swarm into areas with higher emission intensity. This property is utilized by firefly algorithm in order to solve continuous optimization problems. The meta-heuristics of firefly algorithm require certain assumptions (i) Gender of fireflies is irrelevant, (ii) Light intensity varies inversely proportional to distance and directly proportional to attractiveness,(iii) Attraction is always positive gradient. Higher intensity is more attractive than lower intensity and (iv) the light intensity of fireflies is modified by landscape of fitness function.

In the simplest expression of firefly algorithm when applied to maximum optimization problems, the attractiveness is determined as being dependant on the brightness of firefly at particular location. The primary construct in determining attractiveness is an objective function, which is directly proportional to brightness of firefly at that location. But attractiveness is considered a relative value, based on an ordered pair set over all fireflies in a population.

For two fireflies i and j, both belonging to same population P, i, j ϵ P, $x_i$and x_j are location of i and j. r_{ij} is the Cartesian distance [2, 8, 11, 25, 26] between x_i and x_j. For any Cartesian d dimensions,

$$r_{ij} = \left|\left|x_i - x_j\right|\right| = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2} \quad (11)$$

The attractiveness function [25, 26] for a firefly is based on square law and absorption co-efficient of the environment. As the attractiveness depends on the observed brightness of that firefly at that location from other fireflies. For an ordered set, the (i, j), attractiveness function β (r_{ij}) can be approximated as

$$\beta(r_{ij}) = \beta_0 e^{-\gamma r_{ij}^2} \quad (12)$$

Where r is the absorption co-efficient, r_{ij} is Cartesian distance between $x_i$and $x_j$which are locations of fireflies i and j, β_0 is the attractiveness when r_{ij} =0.

The movement of firefly i towards another firefly j with higher intensity β is determined by

$$x_i = x_i + \beta(r_{ij}).(x_i - x_j) + \alpha\epsilon_i \quad (13)$$

Where $\alpha\epsilon_i$ denotes random walk with randomization parameter α and random number ϵ_i.

Elements of FA

1) Environment

It allows N dimensional unrestricted movement. It has universal properties such as absorption co-efficient; the topology is modified by an objective function.

2) Components or Fireflies

It is the solution carrying element of firefly algorithm. It has property, light intensity. It is capable of movement across environment towards other fireflies based on attractiveness. Initial distribution is based on

solution space under consideration and relative fitness of these solutions. Fireflies whose solutions display a greater variance will be separated by greater distance. Initial population also reflects set of all possible candidate solutions, all being separated by distance proportional to relative fitness. Each component or firefly has Cartesian distance and light intensity and has no memory.

3) Start Configuration

The population is made up of components randomly distributed. Distribution of fireflies may be relative to fitness.

4) End Configuration

It is based on maximum generations or iterations.

5) Iteration Engine

Iteration Engine in Firefly Algorithm passes through the population iterating over an ordered pair set consisting every combination of individuals calculating their relative attractiveness with respect to each other. Each member of the ordered set is (i, j) pair. i is an observer and j is observed target. At each iteration relative attractiveness of j is observed by i at their current position x_i and x_j is evaluated; if the target is observed to have higher attraction than the observer, then i move towards j relative to quantity based on their attractiveness and random walk variable. At the end of the iteration, the fireflies are arranged in order of fitness. The best solution is determined for that generation. Iteration continues to maximum number of generations in the end configuration. The best solution at end of all generation is used to determine locality of the final best solution.

Applications of FA

FA is used in Function optimization [25], Parameter estimation [25], combinatorial estimation [26]; Least squares support vector machine and Geo technical engineering problems [26].

CONCLUSION AND FUTURE WORK

Bio-inspired algorithms have revolutionized computing by providing new set of solutions inspired by nature. Distributed computing [2] and agent computing [8], that take advantage of SI based solutions are also forefront of modern computing solutions. Evolution in nature uses swarm intelligence as one of its primary problem solver, a paradigm, which was the result of millions of years of Darwinian evolutionary pressure [9, 13, 18]. The survival has driven each species to discover and evolve unique to optimize efficiency. Swarms made up of relatively simple individuals acting together to a common goal, particularly social organisms, which represent one of the biggest success stories of evolution. Inspired by this , several meta-heuristics have been suggested such as Particle Swarm Optimization inspired by flocking behaviour of birds, Ant Colony Optimization inspired by foraging behaviour of ants, Bacterial foraging Optimization algorithm inspired by foraging and survival behaviour of bacteria and Firefly algorithm based on mating behaviour of fireflies.

This paper makes an in depth study of these meta-heuristics by breaking down these algorithms based on four common components. These components though shared by all algorithms are implemented differently for each. This method can be used to further classify Swarm Intelligence algorithms based on the various subtypes as implementation of these elements. The environment component itself can be sub-typed based on movement freedom, presence of solutions in environment. The component element can be sub-typed based on solution capacity and memory capacity. The start and end configuration can be sub-typed based on the usage of random initialization functions and generation count limit. The iteration engine shows greatest variation and be sub-typed based on use of various functions such as culling, cloning and randomization. All these classification techniques can be extended to other SI algorithms, thereby increasing our understanding of biologically inspiration of SI algorithms.

REFERENCES

[1] Bonabeau E, Dorigo M and Theraulaz, G.Swarm Intelligence, Oxford University Press.1999.

[2] Selvaraj.C, Kumar.S, Karnan M. A Survey on Applications of Bio Inspired Algorithms. IJCSIT, International Journal of Computer Science and Information Technologies, 2014.5(1) pp.1-5.

[3] Kennedy J, EberhartR.SwarmIntelligence, Morgan Kaufmann, San Francisco.2001.

[4] Fister I, Yang X.S, Brest J, Fister D. A Brief Review of Nature Inspired Algorithms for Optimization. 2013 July, 80(3).

[5] Chu S.C, Huang H.C, Roddick.J.F and Pan.J.S.Overview Of Algorithms for Swarm Intelligence.ICCCI 2011, Part I, LNCS 6922, pp, Springer-Verlag Berlin Heidelberg 2011,28-41.

[6] Agarwal.P, Mehta.S.Nature inspired algorithms: state of art, problem and prospects. International Journal of Computer Applications.2014 August. (0975-8887), Volume 100-No.1.

[7] Zhang.Y, Agarwal.P, Bhatnagar.V, Balochian.S, andYan.J.SwarmIntelligence and its Applications, The Scientific World Journal.2013.

[8] BinithaS, SathyaS.S. A Survey of Bioinspired Optimization Algorithms.IJSCE, ISSN: Volume-2, 2012 May, Issue-2, 2231-2307.

[9] Ridge.E, Curry.E, A roadmap of Nature Inspired system Research and Development, Journal.

[10] Kennedy.J,Eberhart.R,Particle Swarm Optimization.Proceedingsof

[11] IEEE International Conference on Neural Networks.1995. pp 1942-1948.

[12] Sureja.N.New inspirations in nature: A survey International Journal of Computer Applications and Information Technology .2012 November Vol1, Issue 111, (ISSN: 2278-7720).

[13] Yang.X.S .Nature Inspired Metaheuristic Algorithms, Luniver Press.2008.

[14] Blas N.G, DeMingo L.F, Peneula J.C. Bioinspired Optimization Strategies: A Survey. Natural Computing Group, Spain.

[15] YangX.S. Cue Z., Xiao R., GandomiA.H et al, Swarm Intelligence and Bio inspired Computation, Theory and Applications, Elsevier, Waltham, Mass, USA.2013.

[16] Dorigo.M,ManiezzoV,and,ColorniA.Ant System :Optimization by a colony of cooperating agents.IEEE Transactions on Systems ,Man ,Cybernetics1996, Part B Algorithms.Foundations And Applications (SAGA 109) ,Vol 5792 of lecture notes of Computer Sciences, Springer,Oct,2009.

[17] Mahale R.A, Prof ChavanS.D, A Survey: Evolutionary and Swarm Based Bioinspired Optimization Algorithms.International Journal of Scientific and Research Publications, Volume 2, Issue12.December 2012.

[18] Binu A, Nandhakumar N.K .A survey on Bioinspired methods for resource discovery.IJARCCE, Vol2, Issue 5.2013 May.

[19] Back.T, Evolutionary Algorithms in theory and Practice, Oxford University Press.1996.

[20] Chen.H,Zhu.Y and Hu.k.Cooperative Bacterial Foraging Optimization .Hindawi Publishing Corporation ,Discrete Dynamics in Nature and Society,2009Article ID 815247,17 pages.

[21] Liu.Y and Passino K.M.Biomimicry of Social Foraging Bacteria for Distributed Optimization: Models, Principles and Emergent Behaviors. Journal of Optimization Theory and Applications, 2002 December Vo115, No 3, pp 608-628.

[22] TangW.J,Wu Q.H,Senior Member ,IEEE and J.R .Saunders, Bacterial Foraging Algorithm For Dynamic Environments,IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel,Vancover,BC,Canada,July,16-21.

[23] Zareh.S, Seyedjavadi H.H, Erfani H, Grid Scheduling using Cooperative BFO Algorithm, American Journal of Scientific Research ISSN 1450-223X Issue62,2012, pp.78-87.

[24] Thomas R.M .Survey of Bacterial Foraging Optimization Algorithm"IJISME, ISSN: 2319-6386, Volume-1 Issue-4 2013 March.

[25] K.M.Passino.Biomimicry of bacterial foraging for distributed optimization and control, IEEEControl Syst.Mag, Vol22, no3 pp52-672002 June.

[26] Fister.I, YangX.S, Brest.J.A Comprehensive review of firefly algorithms.Swarm and Evolutionary Computation.2013.

[27] Yang.X.S and He.X, Firefly Algorithm: Recent Advances and Applications, IntJ.Bio-inspired Computation Vol 2, No 2, pp52-67.2010.

[28] Kanaka Vardhini.K, Sitamahalakshmi.T, "A Review on Nature-based Swarm Intelligence Optimization Techniques and its Current Research Directions", Indian Journal of Science and Technology, 2016 Mar, 9(10), Doi no: 10.17485/ijst/2016/v9i10/81634.

[29] Laouratou .D, Aisha-Hassan A. Hashim, Rashidah.F.O, Shayla I, Abdullah A.Z,"Two Objectives Big Data task Scheduling using Swarm Intelligence in Cloud Computing",Indian Journal of Science and Technology,2016 July, 9(28), Doi no:10.17485/ijst/2016/v9i28/96635.

[30] Revathi.S, Malathi.A, "Multi-Tier Framework Using Sugeno Fuzzy Inference System with Swarm Intelligence Techniques for Intrusion Detection", Indian Journal of Science and Technology, 2014 Jan, 7(9), Doi no:10.17485/ijst/2014/v7i9/47317.

[31] Venkat .R, Natarajan.A.M,"Comparison of Genetic Algorithm with Particle Swarm Optimisation, Ant Colony Optimisation and Tabu Search based on University Course Scheduling System",Indian Journal of Science and Technology,2016 June, 9(21), Doi no: 10.17485/ijst/2016/v9i21/85379.

[32] Reza.E, Fazlollah.R,"Unit Commitment in Power System by Combination of Dynamic Programming (DP), Genetic Algorithm (GA) and Particle Swarm Optimization (PSO)",Indian Journal of Science and Technology,2015 Jan, 8(2), Doi no:10.17485/ijst/2015/v8i2/57782.

[33] Ashwin Kumar S.V, RahuL .R, Dheepan.P, SendhilKumar.K.S, "An Optimal Ant Colony Algorithm for Efficient VM Placement", Indian Journal

of Science and Technology, 2015 Jan, 8(S2), Doi no: 10.17485/ijst/2015/v8iS2/60286.

[34] Jung Man Hong, Geun-Cheol Lee, Kilhwan Kim, Seong-Hoon Choi ,"An Ant Colony Algorithm for Assembly Line Balancing Problems with RelaxableConstraints",Indian

[35] Journal of Science and Technology, 2015 Apr, 8(S8), Doi no:10.17485/ijst/2015/v8iS8/70479.