IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727
PP 12-19
www.iosrjournals.org

A Review on Bio-Inspired Computing Algorithms and Application

Ms. Neeta Nemade[1], Mr. Dhiraj Rane[2]
[1](neetumn111@yahoo.co.in , *MCA, IGNOU, Nagpur, India)*
[2]*(dhirajrane2302@gmail.com, Department of CS, GHRIIT, Nagpur, India)*

Abstract: *Computer science and biology have enjoyed a long and fruitful relationship for decades. Biologists rely on computational methods to analyze and integrate large data sets, while several computational methods were inspired by the high level design principles of biological systems. Recently, these two directions have been converging. In this review, we argue that thinking computationally about biological processes may lead to more accurate models, which in turn can be used to improve the design of algorithms. We discuss the similar mechanisms and requirements shared by computational and biological processes and then present several recent studies that apply this joint analysis strategy to problems related to coordination, network analysis, and tracking and vision. We also discuss additional biological processes that can be studied in a similar manner and link them to potential computational problems. With the rapid accumulation of data detailing the inner workings of biological systems, we expect this direction of coupling biological and computational studies to greatly expand in the future.*

Keywords *– AI, Swarm Intelligence, Cock search, Ant Colony Optimization, DNA Computing*

I. INTRODUCTION

Bio-inspired computing, short for biologically inspired computing, is a field of study that loosely knits together subfields related to the topics of connectionism, social behaviour and emergence. It is often closely related to the field of artificial intelligence, as many of its pursuits can be linked to machine learning. It relies heavily on the fields of biology, computer science and mathematics. Briefly put, it is the use of computers to model the living phenomena, and simultaneously the study of life to improve the usage of computers. Biologically inspired computing is a major subset of natural computation.

Some areas of study encompassed under the canon of biologically inspired computing, and their biological counterparts:

- genetic algorithms ↔ evolution
- biodegradability prediction ↔ biodegradation
- cellular automata ↔ life
- emergent systems ↔ ants, termites, bees, wasps
- neural networks ↔ the brain
- artificial life ↔ life
- artificial immune systems ↔ immune system
- rendering (computer graphics) ↔ patterning and rendering of animal skins, bird feathers, mollusk shells and bacterial colonies
- Lindenmayer systems ↔ plant structures
- communication networks and protocols ↔ epidemiology and the spread of disease
- membrane computers ↔ intra-membrane molecular processes in the living cell
- excitable media ↔ forest fires, "the wave", heart conditions, axons, etc.
- sensor networks ↔ sensory organs

Computer science and biology have enjoyed a long and fruitful relationship for decades. Biologists rely on computational methods to analyze and integrate large data sets, while several computational methods were inspired by the high level design principles of biological systems. Recently, these two directions have been converging. In this paper, we are checking the computationally about biological processes may lead to more accurate models, which in turn can be used to improve the design of algorithms. We discuss the similar mechanisms and requirements shared by computational and biological processes and then present several recent studies that apply this joint analysis strategy to problems related to coordination, network analysis, and tracking and vision and various other computational methodologies. We also discuss additional biological processes that can be studied in a similar manner and link them to potential computational problems. With the rapid

accumulation of data detailing the inner workings of biological systems, we expect this direction of coupling biological and computational studies to greatly expand in the future.

Biologists have been increasingly relying on sophisticated computational methods, especially over the last two decades as molecular data have rapidly accumulated. Computational tools for searching large databases, including BLAST (Altschul et al, 1990), are now routinely used by experimentalists. Genome sequencing and assembly rely heavily on algorithms to speed up data accumulation and analysis (Gusfield, 1997; Trapnell and Salzberg, 2009; Schatzet al, 2010). Computational methods have also been developed for integrating various types of functional genomics data and using them to create models of regulatory networks and other interactions in the cell (Alon, 2006; Huttenhower et al, 2009; Myers et al, 2009). Indeed, several computational biology departments have been established over the last few years that are focused on developing additional computational methods to aid in solving life science's greatest mysteries. Computer scientists have also relied on biological systems for inspiration, especially when developing optimization methods (Table I). Early work on artificial intelligence in the 1960s leveraged ideas related to the activity of neurons in the brain to develop a class of computational methods known as neural networks (Hebb, 1949; Hopfield, 1982; Bishop, 1996), which have been used in many machine learning applications ranging from image segmentation to computing missile trajectories (Bishop, 1996). Other optimization techniques, such as genetic algorithms (Goldberg, 1989), were inspired by common operations in DNA sequence evolution and have been widely applied over the last 20 years. Social insects (Dorigo et al, 2006) and particle swarms (Kennedy and Eberhart, 2002) have also motivated the study of how selforganization emerges from local interactions (Abelsonet al, 2000; Committee on Frontiers at the Interface of Computing, Biology, and National Research Council, 2005). These ideas have been applied extensively to multi-agent system optimization (Deneubourg et al, 1990; Ferber, 1999). A number of additional methods, including non-negative matrix factorization (Lee and Seung, 1999) and population protocols (Aspnes and Ruppert, 2009; Chazelle, 2009) have also capitalized on biological insights to derive new computing paradigms. While all of these methods have led to successful applications, they only relied on a high-level (and sometimes flawed) understanding of the biological processes they were based on, and thus they usually did not directly lead to new biological insights. Similarly, though novel computational methods have been developed to help researchers learn new biology, the application of these methods to the biological problem (i.e., the biological system itself) rarely fed back to help computer scientists design better algorithms. Thus, the two directions— relying on biological ideas to develop computational methods and using computational methods to study biology—remained largely separated (Figure 1)

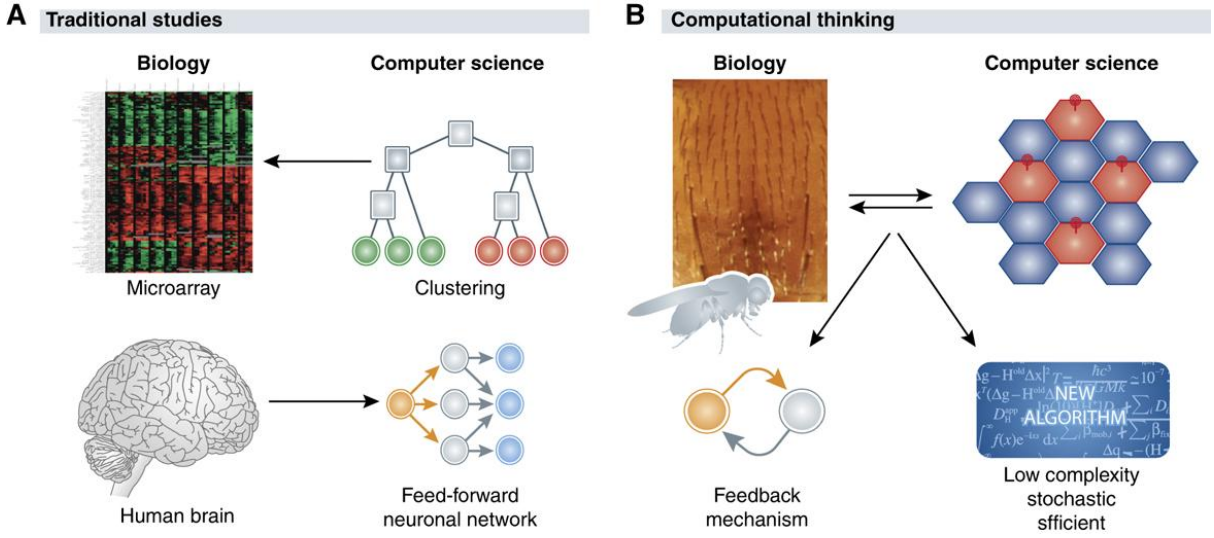

Figure 1 Traditional studies versus computational thinking. (A) Traditionally, biologists leveraged computing power to analyze and process data (e.g., hierarchically clustering gene expression microarrays to predict protein function), and computer scientists used high-level design principles of biological systems to motivate new computational algorithms (e.g., neural networks). Rarely these two directions were coupled and mutually beneficial. (B) By thinking computationally about how biological systems process information (Nurse, 2008; Hogeweg, 2011), we can develop improved models and algorithms and provide a more coherent explanation of how and why the system operates as it does.

I.1. BIO-INSPIRED COMPUTING AND AI

The way in which bio-inspired computing differs from the traditional artificial intelligence (AI) is in how it takes a more evolutionary approach to learning, as opposed to the what could be described as 'creationist' methods used in traditional AI. In traditional AI, intelligence is often programmed from above: the programmer is the creator, and makes something and imbues it with its intelligence. Bio-inspired computing, on the other hand, takes a more bottom-up, decentralised approach; bio-inspired techniques often involve the method of specifying a set of simple rules, a set of simple organisms which adhere to those rules, and a method of iteratively applying those rules. For example, training a virtual insect to navigate in an unknown terrain for finding food includes six simple rules. The insect is trained to

- turn right for target-and-obstacle left;
- turn left for target-and-obstacle right;
- turn left for target-left-obstacle-right;
- turn right for target-right-obstacle-left,
- turn left for target-left without obstacle and
- turn right for target right without obstacle.

The virtual insect controlled by the trained spiking neural network can find food after training in any unknown terrain.[1] After several generations of rule application it is usually the case that some forms of complex behavior arise. Complexity gets built upon complexity until the end result is something markedly complex, and quite often completely counterintuitive from what the original rules would be expected to produce (see complex systems). For this reason, in neural network models, it is necessary to accurately model an in vivo network, by live collection of "noise" coefficients that can be used to refine statistical inference and extrapolation as system complexity increases. [2]

Natural evolution is a good analogy to this method–the rules of evolution (selection, recombination/reproduction, mutation and more recently transposition) are in principle simple rules, yet over millions of years have produced remarkably complex organisms. A similar technique is used in genetic algorithms

II. STRATEGIES EVOLVE BY NATURE

II.1. Shared Principal of Computation

There are many parallel requirements of computational and biological systems, which suggest that one can learn from the other (Figure 2). First, like virtually all large-scale computing platforms, biological systems are mostly distributed consisting of molecules, cells, or organisms that interact, coordinate, and make decisions without central control (Seeley, 2002; Babaogluet al, 2006; Figure 2A). Second, biological processes need to be able to successfully handle failures and attacks to thrive (Jeonget al, 2000; Kitano, 2004). Robustness is also a key property algorithm engineers covet when designing computing systems that persist in noisy environments. Third, networks serve as an important medium through which interactions occur and information propagates (Alon, 2006; Figure 2B). In both settings, the structure of these networks is often directly linked to the system's function. Fourth, biological systems are often modular; that is, they reuse certain components in multiple, and sometimes very different, applications. This is a key design principle in many programming languages and in large complex networks (Kashtan and Alon, 2005; Fortunato, 2010; Figure 2C). Fifth, biological processes are often stochastic (Kaernet al, 2005) resembling randomized algorithms whose power has been well documented for the design of approximation algorithms and whose use ranges from single process systems to large distributed systems (Motwani and Raghavan, 1996; Vazirani, 2004; Figure 2D)

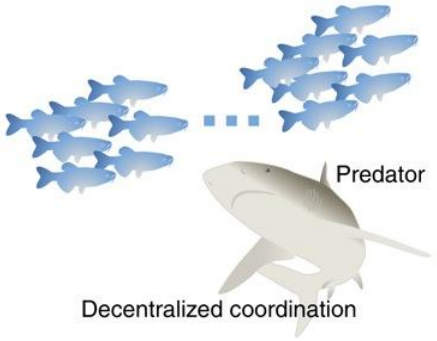

Figure 2(A)

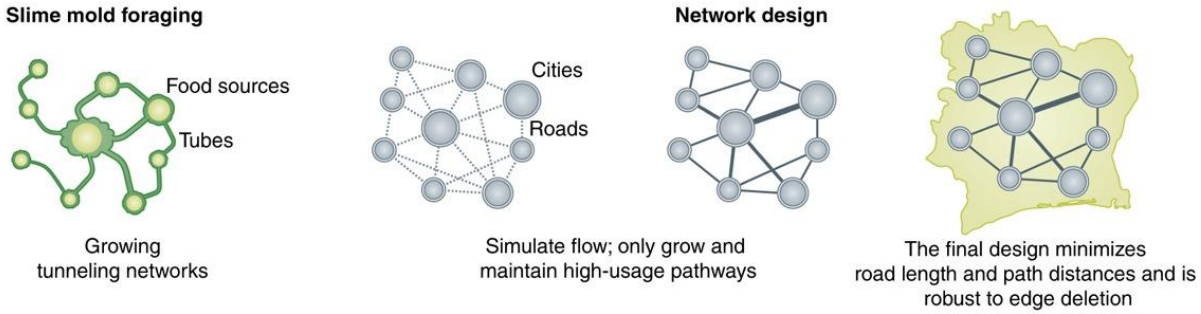

Figure 2(B)

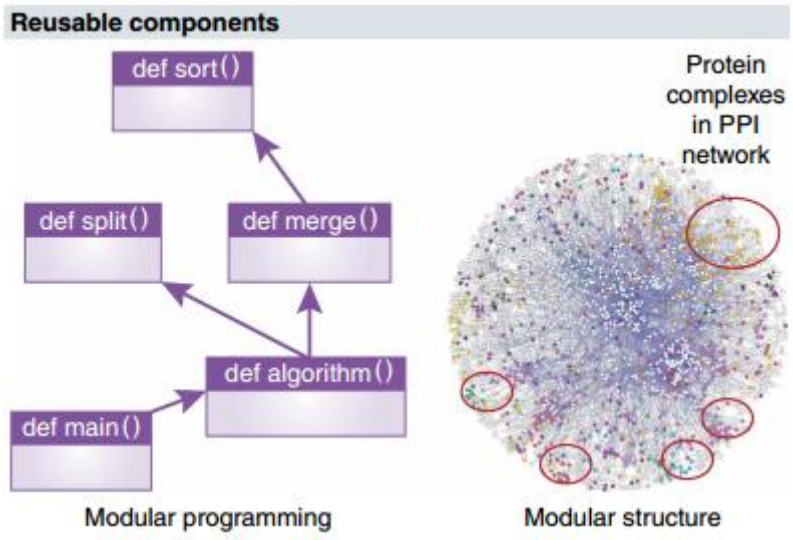

Figure 2©

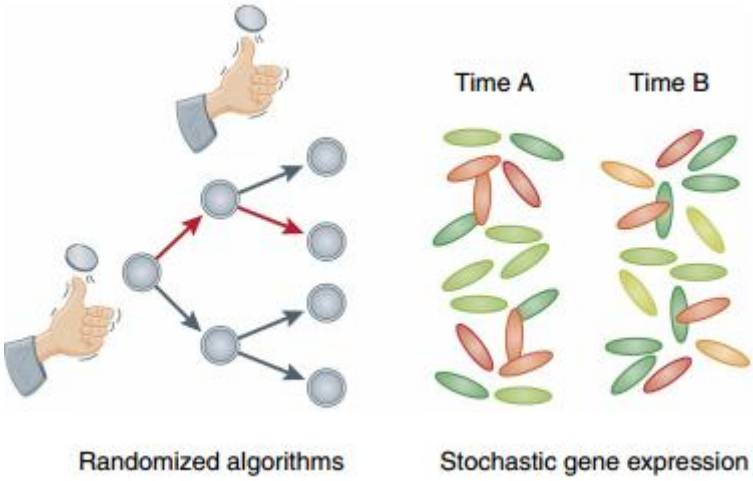

Figure 2(D)

II.2. **Coordination**

Coordination is a major challenge for both computational and biological processes. At the molecular level, coordination is required to activate sets of genes that together respond to external conditions (Kauret al, 2010). At the cellular level, cells coordinate to determine cell fate during development (Afek et al, 2011) and to synchronize heart beats (Mirollo and Strogatz, 1990). Organisms, including the octopus, need to coordinate the activation of limbs (Sumbreet al, 2001), while shoals of fish coordinate to avoid predators (Wardet al, 2011). In computational systems, coordination is required for virtually all large-scale computing systems. Examples include search engines that coordinate thousands of back-end servers to quickly respond to user queries (Brin and Page, 1998), sensor networks that aggregate data when monitoring environments (Mainwaringet al, 2002; Werner-Allen et al, 2006), and mobile networks that synchronize data and schedules across multiple devices (Bernardet al, 2004). Below we discuss a few examples in detail.(Figure 3)

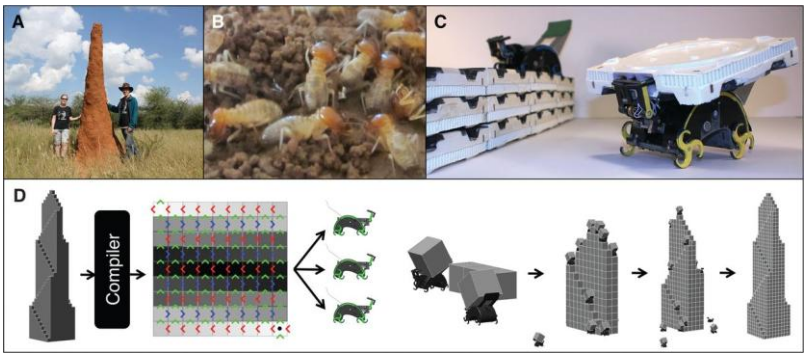

Figure 3: Coordination System

III. ALGORITHM BASED ON BIO INSPIRED STRATEGIES

The ideas from nature and biological activities have motivated the development of many sophisticated algorithms for problem-solving. These algorithms are broadly classified as evolutionary computation and swarm intelligence (SI) algorithms. Evolutionary computation is a term used to describe algorithms which were inspired by 'survival of the fittest' or 'natural selection' principles3; whereas 'swarm intelligence' is a term used to describe the algorithms and distributed problems-solvers which were inspired by the cooperative group intelligence of swarm or collective behaviour of insect colonies and other animal societies

III.1. Evolutionary Algorithm

Evolutionary algorithms (EAs) are computational methods inspired by the process and mechanisms of biological evolution. According to Darwin's natural selection theory of evolution, in nature the competition among individuals for scarce resources results in the fittest individuals dominating over the weaker ones (i.e. survival of the fittest). The process of evolution by means of natural selection helps to account for the variety of life and its suitability for the environment. The mechanisms of evolution describe how evolution actually takes place through the modification and propagation of genetic material (proteins). EAs share properties of adaptation through an iterative process that accumulates and amplifies beneficial variation through a trial and error process. Candidate solutions represent members of a virtual population striving to survive in an environment defined by a problem-specific objective function. In each case, the evolutionary process refines the adaptive fit of the population of candidate solutions in the environment, typically using surrogates for the mechanisms of evolution such as genetic recombination and mutation

- **Genetic algorithms**

In the field of artificial intelligence, a genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a metaheuristic) is routinely used to generate useful solutions to optimization and search problems.[1] Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

The Canonical GA (pseudo code):

choose initial population
evaluate each individual's fitness
determine population's average fitness
repeat
    select best-ranking individuals to reproduce
    mate pairs at random
    apply crossover operator
    apply mutation operator
    evaluate each individual's fitness
    determine population's average fitness
until terminating condition (e.g. until at least one individual has
    the desired fitness or enough generations have passed)

- **Differential evolution**

DE is a modern optimization technique in the family of EAs introduced by Storn and Price[x]. It was proposed as a variant of EAs to achieve the goals of robustness in optimization and faster convergence to a

given problem. DE algorithm differs from other EAs in the mutation and recombination phases. Unlike GAs, where perturbation occurs in accordance with a random quantity, DE uses weighted differences between solution vectors to perturb the population.

A typical DE works as follows: after random initialization of the population (where the members are randomly generated to cover the entire search space uniformly), the objective functions are evaluated and the following steps are repeated until a termination condition is satisfied. At each generation, two operators, namely mutation and crossover are applied on each individual to produce a new population. In DE, mutation is the main operator, where each individual is updated using a weighted difference of a number of selected parent solutions; and crossover acts as background operator where crossover is performed on each of the decision variables with small probability. The offspring replaces the parentonly if it improves the fitness value; otherwise the parent is carried over to the new population.

III.2. **Swarm intelligence**

Swarm intelligence (SI) is artificial intelligence based on the collective behavior of decentralized, self-organized systems. The expression was introduced by Gerardo Beniand Jing Wang in 1989, in the context of cellular robotic systems. SI systems are typically made up of a population of simple agents interacting locally with one another and with their environment. The agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local interactions between such agents lead to the emergence of complex global behavior. Natural examples of SI include ant colonies, bird flocking, animal herding, bacterial growth, and fish schooling. The application of swarm principles to robots is called swarm robotics, while 'swarm intelligence' refers to the more general set of algorithms.

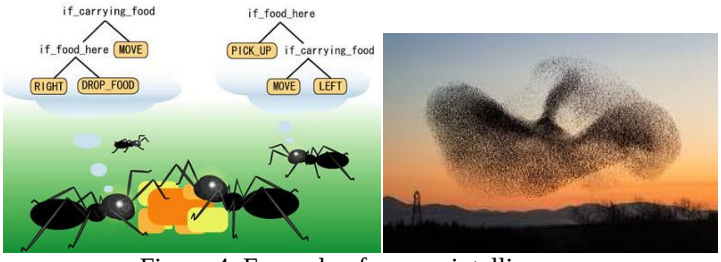

Figure 4: Example of swarm intelligence

Swarm intelligence is the emergent collective intelligence of groups of simple autonomous agents. Here, an autonomous agent is a subsystem that interacts with its environment, which probably consists of other agents, but acts relatively independently from all other agents. The autonomous agent does not follow commands from a leader, or some global plan . For example, for a bird to participate in a flock, it only adjusts its movements to coordinate with the movements of its fock mates, typically its neighbors that are close to it in the fock. A bird in a flock simply tries to stay close to its neighbors, but avoid collisions with them. Each bird does not take commands from any leader bird since there is no lead bird. Any bird can be in the front, center and back of the swarm. Swarm behavior helps birds take advantage of several things including protection from predators (especially for birds in the middle of the flock), and searching for food (essentially each bird is exploiting the eyes of every other bird).

IV. IMPLEMENTATION OF BIOCOMPUTING AND APPLICATIONS

Some of the techniques base on biocomputing has discussed below:

IV1. **Ant Colony Optimization**

Ant colony optimization (ACO) is a population-based metaheuristic that can be used to find approximate solutions to difficult optimization problems. In ACO, a set of software agents called *artificial ants* search for good solutions to a given optimization problem. To apply ACO, the optimization problem is transformed into the problem of finding the best path on a weighted graph. The artificial ants (hereafter ants) incrementally build solutions by moving on the graph. The solution construction process is stochastic and is biased by a *pheromone model*, that is, a set of parameters associated with graph components (either nodes or edges) whose values are modified at runtime by the ants.

IV2. **Particle swarm optimization**

In computer science, particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles

around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position but, is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

PSO is originally attributed to Kennedy, Eberhart and Shi[1][2] and was first intended for simulating social behaviour,[3] as a stylized representation of the movement of organisms in a bird flock or fish school. The algorithm was simplified and it was observed to be performing optimization. The book by Kennedy and Eberhart[4] describes many philosophical aspects of PSO and swarm intelligence. An extensive survey of PSO applications is made by Poli.[5][6] Recently, a comprehensive review on theoretical and experimental works on PSO has been published by Bonyadi and Michalewicz.[7]

PSO is a metaheuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics such as PSO do not guarantee an optimal solution is ever found. More specifically, PSO does not use the gradient of the problem being optimized, which means PSO does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods.

IV3. **Stochastic Diffusion search**

Stochastic diffusion search (SDS) was first described in 1989 as a population-based, pattern-matching algorithm [Bishop, 1989]. It belongs to a family of swarm intelligence and naturally inspired search and optimisation algorithms which includes ant colony optimization, particle swarm optimization and genetic algorithms. Unlike stigmergetic communication employed in ant colony optimization, which is based on modification of the physical properties of a simulated environment, SDS uses a form of direct (one-to-one) communication between the agents similar to the tandem calling mechanism employed by one species of ants, Leptothorax acervorum.

In SDS agents perform cheap, partial evaluations of a hypothesis (a candidate solution to the search problem). They then share information about hypotheses (diffusion of information) through direct one-to-one communication. As a result of the diffusion mechanism, high-quality solutions can be identified from clusters of agents with the same hypothesis. The operation of SDS is most easily understood by means of a simple analogy - The Restaurant Game.

IV4. **Cuckoo Search Algorithm**

Cuckoo Search Algorithm (CSA) [9] is based on the obligate brood parasitic behavior of some cuckoo species in combination with the Levy flight behavior of some birds and fruit flies.CSA is a new meta-heuristic approach that models the natural behavior of cuckoos. To describe the new CSA simplicity, the algorithm's idealized rules are summed up follows [9]:

- Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest. The best nest with high quality eggs (solutions) will be carried out over to the next generations.
- The number of available host nests is fixed, say n, and the host can discover an alien egg by a probability Pa [0, 1].
- The host bird can either throw the egg away or abandon the nest in order to build a completely new nest in a new location.

IV5. **Artificial Bee Colony Algorithm**

Artificial Bee Colony Optimization was proposed by Karaboga in 2005.In the artificial bee colony (ABC) algorithm [7], the colony of artificial bees comprises three groups of bees: employed bees, onlookers and scouts. A bee waiting on the dance area for making decision to choose a food source is called an onlooker and a bee going to the food source visited by itself previously is named an employed bee. A bee carrying out random search is called a scout. In the ABC algorithm, first half of the colony consists of employed artificial bees and the second half constitutes the onlookers. For every food source, there is only one employed bee. In other words, the number of employed bees is equal to the number of food sources around the hive. The employed bee whose food source is exhausted by the employed and onlooker bees becomes a scout. There are four phases[8] in ABC algorithm initialization phase, employed bees phase, onlooker bees' phase and scout bees phase. In initialization phase, from the search space the individuals are randomly selected. Mutation phase is added after the employed bee phase. The local search is done by employed bee phase. The local best position can be changed through mutation, and the algorithm may not be trapped into local optima. By sharing the information individuals can make use of others' advantage. For solving the job scheduling problem with the criterion to decrease the maximum completion time, crossover operator after the employed bee phase and mutation operator after onlooker bee phase of ABC algorithm are added.

V. CONCLUSION

We have studied the basic of biocomputing and the associated evolve algorithms, practical implementations and their results and their application areas. It is found that biocomputing providing the very natural way to solve the problem as the human and lots of the animals in the nature. The biggest problems here are to implements and form the computational methodology by the machine. There will be a promising domain for the computer science development via this bridge opening the new techniques for better problem solving. In future we will focus on the practical implementation of these algorithms.

REFERENCES

[1]. Xu Z, Ziye X, Craig H, Silvia F; Xu; Henriquez; Ferrari (Dec 2013). "Spike-based indirect training of a spiking neural network-controlled virtual insect". Decision and Control (CDC), IEEE: 6798–6805. doi:10.1109/CDC.2013.6760966. ISBN 978-1-4673-5717-3.

[2]. http://www.duke.edu/~{}jme17/Joshua_E. _Mendoza-Elias/Research_Interests.html# Neuroscience_-_Neural_Plasticity_in_brain

[3]. Saket Navlakha and Ziv Bar-Joseph, Algorithms in nature: the convergence of systems biology and computational thinking, Molecular Systems Biology 7; Article number 546; doi:10.1038/msb.2011.78

[4]. Thamarai Selvi Somasundaram and KannanGovindarajan, CLOUDRB: A framework for scheduling and managingHighPerformance Computing(HPC) applications in science cloud, Future Generation Computer Systems, 34 (2014) 47–65.

[5]. M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson,A. Rabkin, I. Stoica and M. Zaharia. Above the Clouds: A Berkeley View of Cloud computing. Technical Report No. UCB/EECS-2009-28, University of

[6]. California at Berkley, USA, Feb. 10, 2009.

[7]. Gunho Lee, Resource Allocation and Scheduling inHeterogeneous Cloud Environments, A Thesis submitted to University of California at Berkley, 2012.

[8]. Abirami S.P. and ShaliniRamanathan. Linear Scheduling Strategy for Resource Allocation in Cloud Environment,International Journal on Cloud Computing: Services and Architecture(IJCCSA),Vol.2, No.1,February 2012.

[9]. Kousalya.K and Balasubramanie. P, An Enhanced Ant Algorithm for Grid Scheduling Problem, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.4, April 2008.

[10]. StefkaFidanova and MariyaDurchova, Ant Algorithm for Grid Scheduling Problem, IPP – BAS, Acad. G. Bonchev, bl.25A, 1113 Sofia, Bulgaria.

[11]. Dervis Karaboga and BahriyeBasturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm,J Glob Optim (2007) 39:459–471

[12]. Manish Gupta and Govind Sharma, AnEfficient Modified Artificial Bee Colony Algorithm for Job SchedulingProblem, International Journal of Soft Computing and Engineering, Vol.1, Issue 6, January, 2012.

[13]. X.-S. Yang and S. Deb, Cuckoo search via Levy flights, in Proc. World Congress on Nature & Biologically Inspired Computing, 2009, pp. 210-214.

[14]. HesamIzakian,BehrouzTorkLadani,Ajith Abrahamand Vaclav Snasel, A discrete particle swarm optimization approach for grid job scheduling,International Journal of Innovative Computing, Information and Control, Volume 6, Number 9, September 2010 pp. 109-0370 .

[15]. J. Kennedy and R. C. Eberhart, Particle swarm optimization, Proc. of the IEEE international Conference on Neural Networks, pp.1942-1948, 1995.

[16]. A. Salman, I. Ahmad and S. Al-Madani, Particle swarm optimization for task assignmentproblem, Microprocessors and Microsystems, vol.26, no.8, pp.363-371, 2002