4

Swarm Intelligence Algorithms in Bioinformatics

Swagatam Das, Ajith Abraham*, and Amit Konar

Department of Electronics and Telecommunication Engineering,
Jadavpur University, Kolkata 700032, India
*Center of Excellence for Quantifiable Quality of Service,
Norwegian University of Science and Technology, Trondheim, Norway
ajith.abraham@ieee.org

Summary. Research in bioinformatics necessitates the use of advanced computing tools for processing huge amounts of ambiguous and uncertain biological data. Swarm Intelligence (SI) has recently emerged as a family of nature inspired algorithms, especially known for their ability to produce low cost, fast and reasonably accurate solutions to complex search problems. In this chapter, we explore the role of SI algorithms in certain bioinformatics tasks like micro-array data clustering, multiple sequence alignment, protein structure prediction and molecular docking. The chapter begins with an overview of the basic concepts of bioinformatics along with their biological basis. It also gives an introduction to swarm intelligence with special emphasis on two specific SI algorithms well-known as Particle Swarm Optimization (PSO) and Ant Colony Systems (ACS). It then provides a detailed survey of the state of the art research centered around the applications of SI algorithms in bioinformatics. The chapter concludes with a discussion on how SI algorithms can be used for solving a few open ended problems in bioinformatics.

4.1 Introduction

The past few decades have seen a massive growth in biological information gathered by the related scientific communities. A deluge of such information coming in the form of genomes, protein sequences, gene expression data and so on have led to the absolute need for effective and efficient computational tools to store, analyze and interpret the multifaceted data.

The term *bioinformatics* literally means the science of informatics as applied to biological research. Informatics on the other hand is the management and analysis of data using various advanced computing techniques. Hence, in other words, bioinformatics can be described as the application of computational methods to make biological discoveries [1]. It presents a symbiosis of several different areas of science including biology, computer science, mathematics and statistics. The ultimate attempt of the field is to develop new insights into the science of life as well as creating a global perspective, from which the unifying principles of biology can be derived [2]. Three major objectives of bioinformatics can be put forward as:

- To develop algorithms and mathematical models for probing the relationships among the members of a large biological dataset.
- To analyze and interpret the heterogeneous kind of data including nucleotide and amino acid sequences, protein domains and protein structures.
- To implement tools that enable efficient storage, retrieval and management of high-volume biological databases.

Biologically inspired computing has been given importance for its immense parallelism and simplicity in computation. In recent times, quite a large number of biologically motivated algorithms have been invented, and are being used for handling many complex problems of the real world. For instance, neural computing [3] attempts to mimic the biological nervous systems of the living creatures to ensure a significant amount of parallel and distributed processing in computation. Genetic algorithms [4], [5] imitate the Darwinian evolutionary process through cross-over and mutation of biological chromosomes. They have successfully been used in many bioinformatics tasks that need intelligent search, optimization and machine learning approaches. Mitra and Hayashi [6] provides a comprehensive survey of the research in this direction.

Recently, a family of nature inspired algorithms known as Swarm Intelligence (SI) [7], [8], [9] has attracted the attention of researchers working on bioinformatics related problems all over the world. Algorithms belonging to this field are motivated by the collective behavior of a group of social insects (like bees, termites and wasps). These insects with very limited individual capability can jointly (cooperatively) perform many complex tasks necessary for their survival. For the past few years there has been a slow but steady increase of research papers reporting the success of SI based search, clustering and data mining methods applied to the field of computational biology.

This Chapter provides a detailed review of the role of SI algorithms in different aspects of bioinformatics mainly involving optimization, pattern recognition and data mining tasks. The rest of the chapter is organized as follows. Section 4.2 briefly describes the preliminary ideas of bioinformatics. In section 4.3, we have introduced the paradigm of Swarm Intelligence and outlined the technical details of two popular SI algorithms known as Particle Swarm Optimization (PSO) [10] and Ant Colony Systems (ACS) [11], [12]. We also discuss the relevance of SI in bioinformatics under this Section. Section 4.4 reviews a number of SI based methods available in the literature to address many difficult tasks in bioinformatics. A few open ended research problems as well as how these can be solved with SI algorithms have been discussed in Section 4.5. Finally the Chapter is concluded in Section 4.6.

4.2 Fundamental Concepts in Bioinformatics

In this section, we outline a few preliminary biological concepts which are essential for understanding several research problems that have been discussed in the subsequent Sections.

4.2.1 DNA

The complete set of instructions for making an organism is called its genome. It contains the master blueprint for all cellular structures and activities for the lifetime of the cell or organism. Found in every nucleus of a person's many trillions of cells, the human genome consists of tightly coiled threads of deoxyribonucleic acid or DNA and associated protein molecules, organized into structures called chromosomes. DNA plays a fundamental role in the different bio-chemical processes of living organisms in two respects:

- Firstly, it contains the information the cell requires to synthesize protein and to replicate itself. To be short, it is the storage repository for the information that is required for any cell to function [13].
- Secondly, it acts as a medium for transmitting the hereditary information (namely the synthesis plans for proteins) from generation to generation.

In humans, as in other higher organisms, a DNA molecule consists of two strands that wrap around each other to resemble a twisted ladder whose sides, made of sugar and phosphate molecules are connected by rungs of nitrogen- containing chemicals called bases. Each strand is a linear arrangement of repeating similar units called nucleotides, which are each composed of one sugar, one phosphate, and a nitrogenous base. Four different bases are present in DNA: adenine (A), thymine (T), cytosine (C), and guanine (G). The particular order of the bases arranged along the sugar-phosphate backbone is called the DNA sequence; the sequence specifies the exact genetic instructions required to create a particular organism with its own unique traits. In normal DNA, the bases form pairs: A to T and G to C. This is called complementarity. The pair of complementary strands then forms the double helix, which was first suggested by Watson and Crick in 1953. Figure 4.1 illustrates the double helix of the DNA sequence with a gene in the sequence delimited. Genes are specific sequences of bases that encode instructions on how to make proteins. We highlight them in the next subsection.

4.2.2 The Gene

Each DNA molecule contains many genes – the basic physical and functional units of heredity. A gene is a specific sequence of nucleotide bases, whose sequences carry the information required for constructing proteins, which provide the structural components of cells and tissues as well as enzymes for essential biochemical reactions. The human genome is estimated to comprise more than 30,000 genes. Human genes

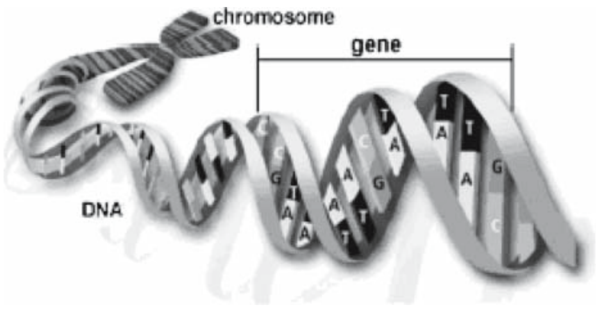


Fig. 4.1. The DNA double helix and a gene sequence

vary widely in length, often extending over thousands of bases, but only about 10% of the genome is known to include the protein- coding sequences (exons) of genes. Interspersed within many genes are intron sequences, which have no coding function.

The gene's sequence is like language that instructs cell to manufacture a particular protein. At first a gene is *transcribed* to produce messenger ribonucleic acid (m-RNA) which is next *translated* to produce proteins. It is the protein that determines the traits of an organism and is called *central dogma of life*. The m-RNA is single-stranded and has a ribose sugar molecule. There exist *promoter* and *termination* sites in a gene, responsible for initiation and termination of the transcription process. Translation consists of mapping from triplets (codons) of four bases to the twenty amino acids that serve as the building blocks of protein. There are sequences of nucleotides within the DNA that are spliced out progressively in the process of transcription and translation. Wu and Lindsay [14] provides a comprehensive survey of the research undertaken so far, in this direction.

Apart from the genes, DNA consists of three types of non coding regions:

1. Intergenic regions: Regions between genes that are ignored during the process of transcription.
2. Intragenic regions (Introns): Regions within the genes that will be spliced out after transcription, but before the RNA is used.
3. Pseudogenes: These are defunct relatives of known genes that have lost their protein-coding ability or are otherwise no longer expressed in the cell [13]. Although they may have some gene-like features (such as Promoters, CpG islands, and splice sites), they are nonetheless considered nonfunctional, due to their lack of protein-coding ability.

Figure 4.2 illustrates the different parts of a gene.

4.2.3 Proteins

An amino acid is an organic molecule that consists of an amine (NH) and a carboxylic (CO) acid group (backbone) together with a side-chain that differentiates between them. Proteins are large organic compounds made of amino acids arranged in a linear

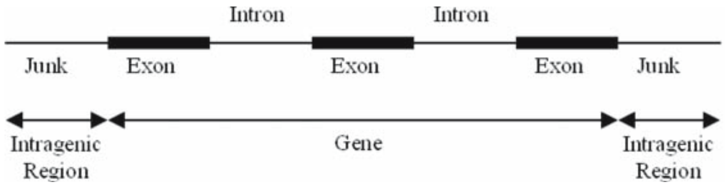


Fig. 4.2. Schematic outline of a gene

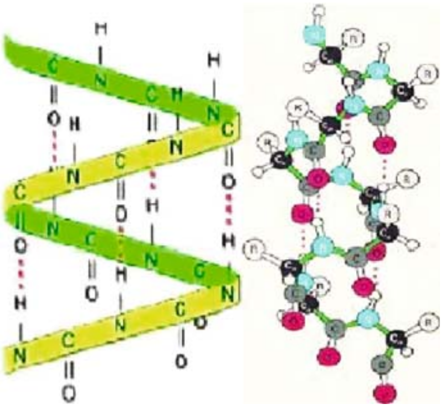


Fig. 4.3. Spiral configuration of the α helix structure. Hydrogen bonds between the CO group of one amino acids and the NH group of another amino acid holds the α helices together (adapted from [22])

chain and joined together between the carboxyl atom of one amino acid and the amine nitrogen of the other [15]. This bond is called a peptide bond. The sequence of amino acids in a protein is defined by a gene and encoded in the genetic code. Although this genetic code specifies 20 "standard" amino acids, the residues in a protein are often chemically altered in post-translational modification: either before the protein can function in the cell, or as part of control mechanisms. Proteins can also work together to achieve a particular function, and they often associate to form stable complexes.

In order to carry out their function, each protein must take a particular shape, known as its fold. When a protein is put into a solvent, within a very short time it takes a particular 3D shape. This self assembling process is called folding. More than half a century ago, Linus Pauling (Nobel prize, 1954) discovered that a major part of most proteins' folded structure consists of two regular, highly periodic arrangements of amino acids, designated "a" and "b". The key to both structures is the hydrogen bond that stabilizes the structures. The "a" structure is now called α helix (Figure 4.3). It is a spiral configuration of a polypeptide chain stabilized by hydrogen bonds between the CO group of one amino acid at position n and the NH group of the amino acid which is four residues away $(n+4)$. The "b" structure is now called β sheet (Figure 4.4). It is an essentially a flat two dimensional structure of parallel

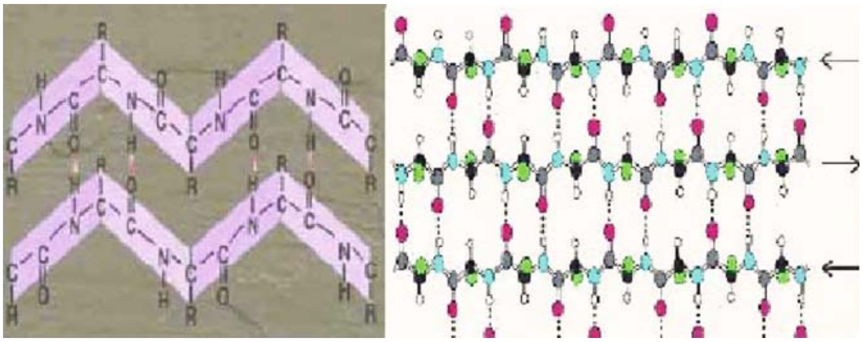


Fig. 4.4. β pleated sheets structure is stabilized by hydrogen bonds between nitrogen atoms (of the NH group of one amino acid) and oxygen atoms (of the CO group of another amino acid) of two adjacent chains (adapted from [22])

or anti-parallel β strands; each β strand consists of two polypeptide chains that are (almost) fully extended and hydrogen bonded to each other. All other local arrangements that are neither α helix nor β sheet are described as random coil: they are random in the sense that they are not periodic.

Proteins have multiple levels of structure [1]:

1. **Primary structure**: Linear structure determined solely by the number, sequence, and type of amino acid residues (R).
2. **Secondary structure**: Local structure determined by hydrogen bonding between amino acids and non-polar interactions between hydrophobic regions. These interactions produce, in general, three secondary structures: α helix (Figure 4.2), β sheet (Figure 4.3), and random coil.
3. **Tertiary structure**: It results from various interactions (mainly hydrophobic attractions, hydrogen bonding, and disulfide bonding) of the amino acids side chains (R) that pack together the elements of the secondary structure. The result is a 3D configuration of proteins.
4. **Quaternary structure**: It is characterized by the interaction of two or more individual polypeptides (often via disulfide bonds) and the result is a larger functional molecule.

4.2.4 DNA Microarray

A DNA microarray (also commonly known as DNA chip or gene array) is a collection of microscopic DNA spots attached to a solid surface, such as glass, plastic or silicon chip forming an array for the purpose of expression profiling, monitoring expression levels for thousands of genes simultaneously. Figure 4.5 illustrates a simple DNA chip [16].

Microarrays provide a powerful basis to monitor the expression of thousands of genes, in order to identify mechanisms that govern the activation of genes in an organism. Short DNA patterns (or binding sites near the genes) serve as switches

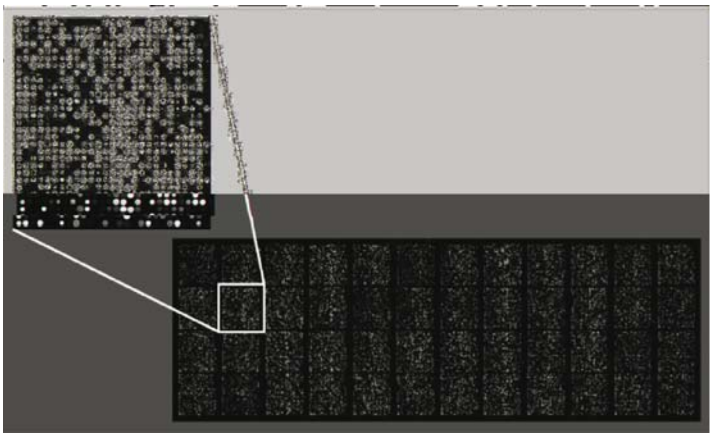


Fig. 4.5. Example of an approximately 37,500 probe spotted oligo microarray with enlarged inset to show detail (adapted from [16])

that control gene expression. Therefore, similar patterns of expression correspond to similar binding site patterns. A major cause of coexpression of genes is their sharing of the regulation mechanism (coregulation) at the sequence level. Clustering of coexpressed genes, into biologically meaningful groups, helps in inferring the biological role of an unknown gene that is coexpressed with a known gene(s). Cluster validation is essential, from both the biological and statistical perspectives, in order to biologically validate and objectively compare the results generated by different clustering algorithms.

4.3 Swarm Intelligence - an Overview and Relevance to Bioinformatics

The behavior of a single ant, bee, termite and wasp often is too simple, but their collective and social behavior is of paramount significance. A look at National Geographic TV Channel also reveals that advanced mammals including lions also enjoy social lives, perhaps for their self-existence at old age and in particular when they are wounded. The collective and social behavior of living creatures motivated researchers to undertake the study of swarm intelligence. Historically, the phrase Swarm Intelligence (SI) was coined by Beny & Wang in late 1980s [9] in the context of cellular robotics. A group of researchers in different parts of the world started working almost at the same time to study the versatile behavior of different living creatures. SI systems are typically made up of a population of simple agents (an entity capable of performing/executing certain operations) interacting locally with one another and with their environment. Although there is normally no centralized control structure dictating how individual agents should behave, local interactions between such agents often lead to the emergence of global behavior. Many biological

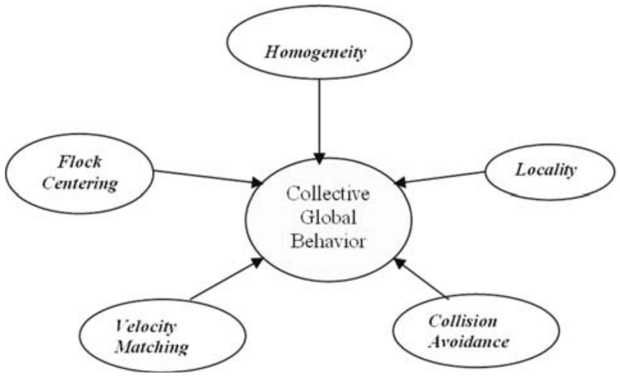


Fig. 4.6. Main traits of the collective behavior

creatures such as fish schools and bird flocks clearly display structural order, with the behavior of the organisms so integrated that even though they may change shape and direction, they appear to move as a single coherent entity [17]. The main properties of the collective behavior can be given below and is illustrated in Figure 4.6:

- **Homogeneity**: every bird in flock has the same behavioral model. The flock moves without a leader, even though temporary leaders seem to appear.
- **Locality**: its nearest flock-mates only influence the motion of each bird. Vision is considered to be the most important senses for flock organization.
- **Collision avoidance**: avoid colliding with nearby flock mates.
- **Velocity matching**: attempt to match velocity with nearby flock mates.
- **Flock centering**: attempt to stay close to nearby flock mates.

Individuals attempt to maintain a minimum distance between themselves and others at all times. This rule is given the highest priority and corresponds to a frequently observed behavior of animals in nature [18]. If individuals are not performing, an avoidance maneuver they tend to be attracted towards other individuals (to avoid being isolated) and to align themselves with neighbors [19], [20]. Couzin et al. [17] identified four collective dynamical behaviors as illustrated in Figure 4.7:

- **Swarm**: an aggregate with cohesion, but a low level of polarization (parallel alignment) among members.
- **Torus**: individuals perpetually rotate around an empty core (milling). The direction of rotation is random.
- **Dynamic parallel group**: the individuals are polarized and move as a coherent group, but individuals can move throughout the group and density and group form can fluctuate [19], [21].
- **Highly parallel group**: much more static in terms of exchange of spatial positions within the group than the dynamic parallel group and the variation in density and form is minimal.

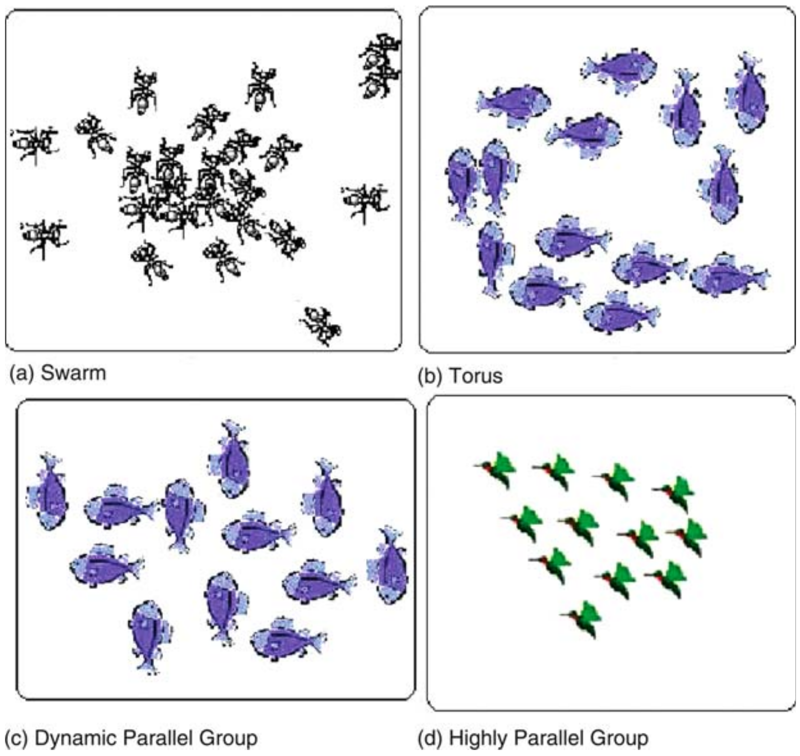


Fig. 4.7. Different models of collective behavior (adapted from [23])

A swarm can be viewed as a group of agents cooperating to achieve some purposeful behavior and achieve some goal (see Figure 4.7) [23] . This collective intelligence seems to emerge from what are often large groups.

According to Milonas, five basic principles define the SI paradigm [24]. First is the proximity principle: the swarm should be able to carry out simple space and time computations. Second is the quality principle: the swarm should be able to respond to quality factors in the environment. Third is the principle of diverse response: the swarm should not commit its activities along excessively narrow channels. Fourth is the principle of stability: the swarm should not change its mode of behavior every time the environment changes. Fifth is the principle of adaptability: the swarm must be able to change behavior mote when it is worth the computational price. Note that principles four and five are the opposite sides of the same coin.

As it appears, 'Self-organization' is one of the fundamental features of any SI system. However, it is not a simple term to define. In general, it refers to the various mechanisms by which pattern, structure and order emerge spontaneously in complex systems. Examples of such structures and patterns include the stripes of zebras, the pattern of sand ripples in a dune, the coordinated movements of flocks of birds or schools of fish, the intricate earthen nests of termites, the patterns on seashells, the whorls of our fingerprints, the colorful patterns of fish and even the spatial pattern of stars in a spiral galaxy. Bonabeau *et al.* have tried to define self-organization using the following words [7]:

Self-organization is a set of dynamical mechanisms whereby structures appear at the global level of a system from interactions of its lower-level components.

Serra and Zanarini [25] describes the concept of self-organization generally as "highly organized behavior even in the absence of a pre-ordained design". They go on to further describe examples such as the resonance phenomenon in lasers, and in cellular automata where "unexpected and complex behaviours can be considered as self-organized." Self-organization was originally introduced in the context of physics and chemistry to describe how microscopic processes give rise to macroscopic structures in out-of-equilibrium systems. Recent research, however, suggests that it provides a concise description of a wide rage of collective phenomena in animals, especially in social insects. This description does not rely on individual complexity to account for complex spatial-temporal features, which emerge at the colony level, but rather assumes that interactions among simple individuals can produce highly structured collective behaviors. There are four main features that govern the self-organization in insect colonies:

- Positive feedback (amplification)
- Negative feedback (for counter-balance and stabilization)
- Amplification of fluctuations (randomness, errors, random walks)
- Multiple interactions

At a high-level, a swarm can be viewed as a group of agents cooperating to achieve some purposeful behavior and achieve some goal. This collective intelligence seems to emerge from what are often large groups of relatively simple agents. The agents use simple local rules to govern their actions and via the interactions of the entire group, the swarm achieves its objectives. A type of self-organization emerges from the collection of actions of the group.

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 [26]. For example, for a bird to participate in a flock, it only adjusts its movements to coordinate with the movements of its flock mates, typically its neighbors that are close to it in the flock. 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 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).

Below we discuss in details two algorithms from SI domain, which have gained huge popularity in a relatively short span of time all over the world. One of these algorithms, known as Ant Colony Optimization (ACO) mimics the behavior of group of real ants in multi-agent cooperative search problems. The latter one is referred to as Particle Swarm Optimization (PSO), which draws inspiration from the behavior of particles, the boids method of Craig Reynolds and socio-cognition [27].

4.3.1 Ant Colony Systems

Insects like ants, bees, wasps and termites are quite social. They live in colonies and follow their own routine of tasks independent of each other. However, when acting as a community, these insects even with very limited individual capability can jointly (cooperatively) perform many complex tasks necessary for their survival [7]. Problems like finding and storing foods, selecting and picking up materials for future usage require a detailed planning, and are solved by insect colonies without any kind of supervisor or controller.

It is a natural observation that a group of 'almost blind' ants can figure out the shortest route between a cube of sugar and their nest without any visual information. They are capable of adapting to the changes in the environment as well [28]. It is interesting to note that ants while crawling deposit trails of a chemical substance known as pheromone to help other members of their team to follow its trace. The resulting collective behavior can be described as a loop of positive feedback, where the probability of an ant's choosing a path increases as the count of ants that already passed by that path increases [12], [28].

The basic idea of a real ant system is illustrated in Figure 4.8. In the left picture, the ants move in a straight line to the food. The middle picture illustrates the situation soon after an obstacle is inserted between the nest and the food. To avoid the obstacle, initially each ant chooses to turn left or right at random. Let us assume that ants move at the same speed depositing pheromone in the trail uniformly. However, the ants that, by chance, choose to turn left will reach the food sooner, whereas the ants that go around the obstacle turning right will follow a longer path, and so will take longer time to circumvent the obstacle. As a result, pheromone accumulates faster in the shorter path around the obstacle. Since ants prefer to follow trails with larger amounts of pheromone, eventually all the ants converge to the shorter path around the obstacle, as shown in Figure 4.8.

An artificial Ant Colony System (ACS) is an agent-based system, which simulates the natural behavior of ants and develops mechanisms of cooperation and learning. ACS was proposed by Dorigo et al. [29] as a new heuristic to solve combinatorial optimization problems. This new heuristic, called Ant Colony Optimization

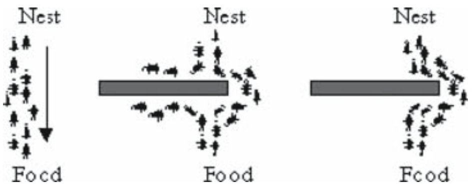


Fig. 4.8. Illustrating the behavior of real ant movements

(ACO) has been found to be both robust and versatile in handling a wide range of combinatorial optimization problems.

4.3.2 The ACO Algorithm

The main idea of ACO is to model a problem as the search for a minimum cost path in a graph. Artificial ants as if walk on this graph, looking for cheaper paths. Each ant has a rather simple behavior capable of finding relatively costlier paths. Cheaper paths are found as the emergent result of the global cooperation among ants in the colony. The behavior of artificial ants is inspired from real ants: they lay pheromone trails (obviously in a mathematical form) on the graph edges and choose their path with respect to probabilities that depend on pheromone trails. These pheromone trails progressively decrease by evaporation. In addition, artificial ants have some extra features not seen in their counterpart in real ants. In particular, they live in a discrete world (a graph) and their moves consist of transitions from nodes to nodes.

Pheromone placed on the edges acts like a distributed long term memory [29]. The memory, instead of being stored locally within individual ants, remains distributed on the edges of the graph. This indirectly provides a means of communication among the ants called *stigmergy* [30]. In most cases, pheromone trails are updated only after having constructed a complete path and not during the walk, and the amount of pheromone deposited is usually a function of the quality of the path. Finally, the probability for an artificial ant to choose an edge, not only depends on pheromones deposited on that edge in the past, but also on some problem dependent local heuristic functions.

We illustrate the use of ACO in finding the optimal tour in the classical Traveling Salesman Problem (TSP). Given a set of n cities and a set of distances between them, the problem is to determine a minimum traversal of the cities and return to the home-station at the end. It is indeed important to note that the traversal should in no way include a city more than once. Let $r(C_x, C_y)$ be a measure of cost for traversal from city C_x to C_y. Naturally, the total cost of traversing n cities indexed by i_1, i_2, i_3,..,i_n in order is given by the following expression:

$$Cost(i_1, i_2, \ldots, i_n) = \sum_{j=1}^{n-1} r(C_{i_j}, C_{i_{j+1}}) + r(C_{i_n}, C_{i_1}) \tag{4.1}$$

The ACO algorithm is employed to find an optimal order of traversal of the cities. Let τ be a mathematical entity modeling the pheromone and $\tau_{ij} = 1/r_{i,j}$ is a local heuristic. Also let $allowed_k(t)$ be the set of cities that are yet to be visited by ant k located in city i. Then according to the classical ant system [11] the probability that ant k in city i visits city j is given by:

$$\begin{aligned} p_{ij}^k(t) &= \frac{[\tau_{ij}(t)]^\alpha \bullet [\eta_{ij}]^\beta}{\sum\limits_{h \in allowed_k(t)} [\tau_{ih}(t)]^\alpha \bullet [\eta_{ih}]^\beta}, \quad if \quad j \in \quad allowed_k(t) \\ &= 0, \; otherwise \end{aligned} \tag{4.2}$$

In Equation 4.2, shorter edges with greater amount of pheromone are favored by multiplying the pheromone on edge (i, j) by the corresponding heuristic value $\eta(i, j)$. Parameters $\alpha(>0)$ and $\beta(>0)$ determine the relative importance of pheromone versus cost. Now in ant system pheromone trails are updated as follows. Let D_k be the length of the tour performed by ant k,$\Delta D_k(i, j) = 1/D_k$ if $(i, j)\varepsilon$ tour done by ant k and = 0 otherwise and finally let $\rho\varepsilon[0,1]$ be a pheromone decay parameter which takes care of the occasional evaporation of the pheromone from the visited edges. Then once all ants have built their tours, pheromone is updated on all the edges as follows:

$$\tau(i, j) = (1-\rho)\cdot\tau(i, j) + \sum_{k=1}^{m} \Delta\tau_k(i, j) \tag{4.3}$$

From Equation 4.3, we can guess that pheromone updating attempts to accumulate greater amount of pheromone to shorter tours (which corresponds to high value of the second term in (4.3), so as to compensate for any loss of pheromone due to the first term). This conceptually resembles a reinforcement-learning scheme, where better solutions receive a higher reinforcement.

The ACS differs from the classical ant system in the sense that here the pheromone trails are updated in two ways. Firstly, when ants construct a tour they locally change the amount of pheromone on the visited edges by a local updating rule. Now if we let γ to be a decay parameter and $\Delta\tau(i, j) = \tau_0$ such that τ_0 is the initial pheromone level, then the local rule may be stated as:

$$\tau(i, j) = (1-\rho)\cdot\tau(i, j) + \gamma\cdot\Delta\tau(i, j) \tag{4.4}$$

Secondly, after all the ants have built their individual tours, a global updating rule is applied to modify the pheromone level on the edges that belong to the best ant tour found so far. If $\hat{E}$ be the usual pheromone evaporation constant, D_{gb} be the length of the globally best tour from the beginning of the trial and $\Delta\tau' = 1/D_{gb}$ only when the edge (i, j) belongs to global-best-tour and zero otherwise, then we may express the global rule as:

$$\tau(i, j) = (1-k)\cdot\tau(i, j) + k\cdot\Delta\tau'(i, j) \tag{4.5}$$

The main steps of ACS algorithm are presented as Algorithm 1. The first loop (iteration) starts with m ants being placed in n cities chosen according to some initialization rule (e.g. randomly). In the embedded loop (step) each ant builds a tour (i.e., an acceptable solution to the TSP) by repeatedly applying a stochastic state transition rule. While building its tour, the ant can modify the pheromone level on the visited edges by applying the local updating rule given by (4.4). Once all the ants have terminated their tour, the pheromone trails are modified again by the global updating rule given in (4.5). Figure 4.9 illustrates the computer simulation of the ACO technique working on a 10 city TSP problem.

Algorithm 1 Ant colony system algorithm

```
Begin
  Initialize pheromone trails;
  Repeat
  Begin /* at this stage each loop is called an iteration */
     Each ant is positioned on a starting node;
     Repeat
     Begin /* at this level each loop is called a step */
        Each ant applies a state transition rule like rule (2) to incrementally build a solution
and a local pheromone-updating rule like rule (4.4);
     Until all ants have built a complete solution;
     A global pheromone-updating rule like rule (4.5) is applied.
  Until terminating condition is reached;
End
```

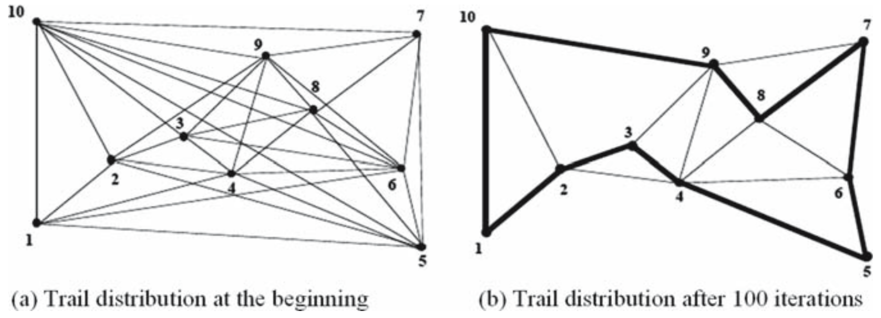

10
7
9
8
3
2
4
6
1
5
(a) Trail distribution at the beginning
10
7
9
8
3
2
4
6
1
5
(b) Trail distribution after 100 iterations

Fig. 4.9. Solving the TSP problem with ACO algorithm (adapted from [29])

4.3.3 The Particle Swarm Optimisation (PSO)

The concept of particle swarms, although initially introduced for simulating human social behaviors, has become very popular these days as an efficient search and optimization technique. The Particle Swarm Optimization (PSO) [10], as it is called now, does not require any gradient information of the function to be optimized, uses only primitive mathematical operators and is conceptually very simple. Since its advent in 1995, PSO has attracted the attention of a lot of researchers all over the world resulting into a huge number of variants of the basic algorithm as well as many parameter automation strategies. PSO [27] is in principle such a multi-agent parallel search technique. Particles are conceptual entities which fly through the multi-dimensional search space. At any particular instant, each particle has a position and a velocity. The position vector of a particle with respect to the origin of the search space represents a trial solution of the search problem. At the beginning a population of particles is initialized with random positions marked by vectors $\mathbf{X}_i$ and random velocities $\mathbf{V}_i$. The population of such particles is called a 'swarm' S. A neighborhood relation N is defined in the swarm. N determines for any two particles Z_i and Z_j whether they are neighbors or not. Thus for any particle Z, a neighborhood can be assigned as $N(Z)$, containing all the neighbors of that particle.

Different neighborhood topologies and their effect on the swarm performance have been discussed in [32]. In the basic PSO, each particle P has two state variables:

1. Its current position $\mathbf{X}_i(t)$.
2. Its current velocity $\mathbf{V}_i(t)$.

And also a small memory comprising,

1. Its previous best position $\mathbf{P}_i(t)$ i.e. personal best experience in terms of the objective function value $f(\mathbf{P}_i(t))$.
2. The best $\mathbf{P}(t)$ of all $Z\varepsilon N(Z)$: i.e. the best position found so far in the neighborhood of the particle.

The PSO scheme has the following algorithmic parameters:

1. $\mathbf{V}_{max}$ or maximum velocity which restricts $\mathbf{V}_i(t)$ within the interval $[-V_{max}, V_{max}]$
2. An inertial weight factor ω.
3. Two uniformly distributed random numbers φ_1 and φ_2 which respectively determine the influence of $\mathbf{P}(t)$ and $\mathbf{g}(t)$ on the velocity update formula.
4. Two constant multiplier terms C_1 and C_2 known as "self confidence" and "swarm confidence" respectively.

Initially the settings for $\mathbf{P}(t)$ and $\mathbf{g}(t)$ are $\mathbf{P}(0) = \mathbf{g}(0) = \mathbf{x}(0)$ for all particles. Once the particles are initialized, the iterative optimization process begins where the positions and velocities of all the particles are altered by the following recursive equations. The equations are presented for the d-th dimension of the position and velocity of the $i-th$ particle.

$$\begin{aligned} V_{id}(t+1) &= \omega V_{id}(t) + C_1\phi_1.(P_d(t) - X_{id}(t)) + C_2\phi_2.(g_d(t) - X_{id}(t)) \\ X_{id}(t+1) &= X_{id}(t) + V_{id}(t+1) \end{aligned} \tag{4.6}$$

The first term in the velocity updating formula represents the inertial velocity of the particle. The second term involving $\mathbf{P}(t)$ represents the personal experience of each particle and is referred to as "cognitive part". The last term of the same relation is interpreted as the "social term" which represents how an individual particle is influenced by the other members of its society. Typically, this process is iterated for a certain number of time steps, or until some acceptable solution has been found by the algorithm or until an upper limit of CPU usage has been reached. Once the iterations are terminated, most of the particles are expected to converge to a small radius surrounding the global optima of the search space. The velocity updating scheme is illustrated in Figure 4.10 with a humanoid particle. A pseudo code for the PSO algorithm is depicted as Algorithm 2.

The PSO algorithm can be seen as a set of vectors whose trajectories oscillate around a region defined by each individual previous best position and the best position of some other individuals [27]. There are different neighborhood topologies used to identify which particles from the swarm can influence the individuals. The most common ones are known as the *gbest* and *lbest*. In the *gbest* swarm; the trajectory of each individual (particle) is influenced by the best individual found in the entire

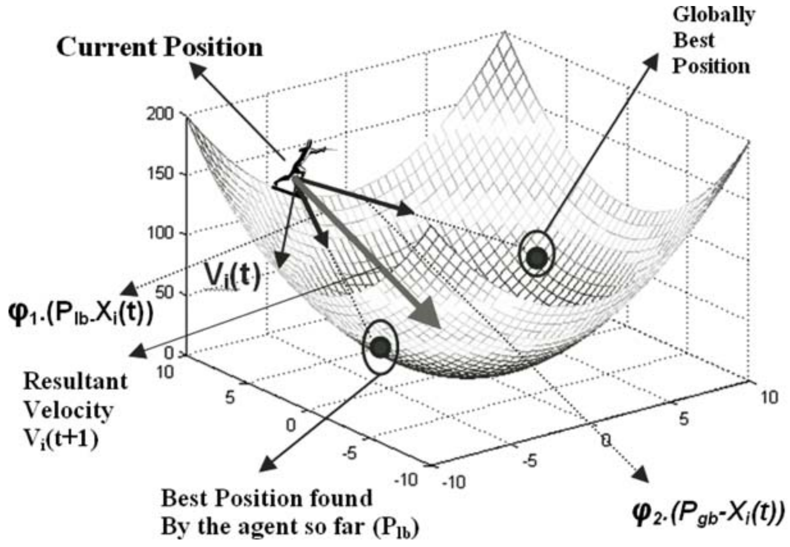

Current Position
Globally Best Position
200
150
100
50
0
$V_i(t)$
$\varphi_1.(P_{lb}-X_i(t))$
Resultant Velocity $V_i(t+1)$
10
5
0
-5
-10
-10
-5
0
5
10
Best Position found By the agent so far (P_{lb})
$\varphi_2.(P_{gb}-X_i(t))$

Fig. 4.10. Illustrating the velocity updating scheme of basic PSO

Algorithm 2 Particle swarm optimization algorithm

```
PSO Algorithm input: Randomly initialized position and velocity of the particles: $\mathbf{X}_i(0)$ and
$\mathbf{V}_i(0)$
Output: Position of the approximate global optima
Begin
  While terminating condition is not reached do
    Begin
      for $i = 1$ to number of particles
          Evaluate the fitness: $=f(\mathbf{X}_i(t))$;
          Update $\mathbf{P}(t)$ and $\mathbf{g}(t)$;
          Adapt velocity of the particle using equation 4.6;
          Update the position of the particle;
          increase;
    end while;
  end;
```

swarm. It is assumed that *gbest* swarms converge fast, as all the particles are attracted simultaneously to the best part of the search space. However, if the global optimum is not close to the best particle, it may be impossible for the swarm to explore other areas and, consequently, the swarm can be trapped in local optima [33]. In the *lbest* swarm, each individual is influenced by a smaller number of its neighbors (which are seen as adjacent members of the swarm array). Typically, *lbest* neighborhoods comprise of two neighbors: one on the right side and one on the left side (a ring lattice). This type of swarm will converge slower but can locate the global optimum with a greater chance. *lbest* swarm is able to flow around local optima, sub-swarms being

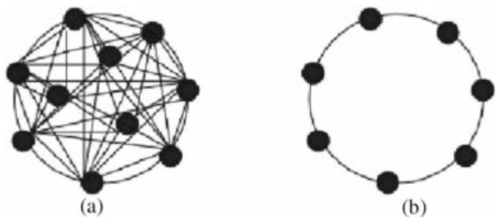


Fig. 4.11. Graphical Representation of (a) *gbest* Swarm (b) lbest Swarm (adapted from [33])

able to explore different optima [32]. A graphical representation of a *gbest* swarm and an *lbest* swarm respectively is depicted in Figure 4.11.

Watts [34] introduced the small-world network model, which allows interpolating between regular low-dimensional lattices and random networks, by introducing a certain amount of random long-range connections into an initially regular network [35]. Starting from here, several models have been developed: icing model [36], spreading of epidemics [37], evolution of random walks [38] are some of them.

4.3.4 Relevance of SI Algorithms in Bioinformatics

From the discussion of the previous subsections, we see that the SI algorithms are mainly stochastic search and optimization techniques, guided by the principles of collective behaviour and self organization of insect swarms. They are efficient, adaptive and robust search methods producing near optimal solutions and have a large amount of implicit parallelism. On the other hand, several tasks in bioinformatics involve optimization of different criteria (like energy, alignment score, overlap strength and so on); thereby making the application of SI tools more obvious and appropriate. For example, most of the ordering problems in bioinformatics, such as the sequence alignment, fragment assembly problem (FAP) and gene mapping (GM), are quite similar to the TSP (one of the most difficult ordering problems till date) with notable differences [39]. We have already discussed how TSP can be solved efficiently with the ant systems in Section 4.3.2. Thus, ACO can be tried on many of these problems and the results can be compared with the classical methods used in these contexts up to now.

The problems of bioinformatics seldom need the exact optimum solution; rather what they need are robust, fast and near optimal solutions, which SI algorithms like PSO are known to produce efficiently. Moreover, the laboratory operations on DNA inherently involve errors. These are more tolerable in executing the SI algorithms than in executing deterministic algorithms. To some extent, these errors may be regarded as contributing to population diversity, a desirable property for the convergence of the SI algorithms. The problem of integrating SI in bioinformatics, in this way, can develop a new research area.

4.4 A Review of the Present State of the Art

In this section, we provide a substantial review of the state of the art research, which focuses on the application of swarm intelligence to different bioinformatics related problems. The number of published papers reporting the applications of PSO or ACO in bioinformatics is currently smaller as compared to the huge amount of work reported for other evolutionary computing methods like GA etc in the same context. Nevertheless, we believe that in near future SI will serve as an indispensable computing methodology in the field of bioinformatics, keeping in mind the reported success of the SI algorithms over classical evolutionary algorithms in many cases [40], [41], [42]. We describe each research problem first and then illustrate how SI algorithms can be used to solve them.

4.4.1 Clustering of Gene Expression Data

Gene expression refers to a process through which the coded information of a gene is converted into structures operating in the cell. It provides the physical evidence that a gene has been "turned on" or activated. Expressed genes include those that are transcribed into m-RNA and then translated into protein and those that are transcribed into RNA but not translated into protein (e.g., transfer and ribosomal RNAs) [43].

The expression levels of thousands of genes can be measured at the same time using the modern microarray technology [44], [45]. DNA microarrays usually consist of thin glass or nylon substrates containing specific DNA gene samples spotted in an array by a robotic printing device. Researchers spread fluorescently labeled m-RNA from an experimental condition onto the DNA gene samples in the array. This m-RNA binds (hybridizes) strongly with some DNA gene samples and weakly with others, depending on the inherent double helical characteristics. A laser scans the array and sensors to detect the fluorescence levels (using red and green dyes), indicating the strength with which the sample expresses each gene. The logarithmic ratio between the two intensities of each dye is used as the gene expression data.

Proper selection, analysis and interpretation of the microarray data can lead us to the answers of many important problems in experimental biology. In the field of pattern recognition, clustering [46] refers to the process of partitioning a dataset into a finite number of groups according to some similarity measure. Currently it has become a widely used process in microarray engineering for understanding the functional relationship between groups of genes. Clustering was used, for example, to understand the functional differences in cultured primary epatocytes relative to the intact liver [47]. In another study, clustering techniques were used on gene expression data for tumor and normal colon tissue probed by oligonucleotide arrays [48].

To cluster the microarray dataset, the first thing we need a suitable similarity measure among the gene profiles. Euclidean distance serves the purpose when the objective is to partition genes displaying similar level of expression. Let $gene_i(x_{i1},$

$x_{i2},\ldots,x_{in})$ denote the expression pattern of the $i-th$ gene. Then the Euclidean distance between the $i-th$ and the $j-th$ gene is given by:

$$d_{i,j} = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2} \tag{4.7}$$

Another popular similarity measure used in this context is the Pearson Correlation Coefficient [49] given by

$$r = \frac{\sum_{k=1}^{n}((x_{ik} - \hat{x}_i)(x_{jk} - \hat{x}_j))/n}{\sigma_{x_i} * \sigma_{x_j}} \tag{4.8}$$

A number of standard clustering algorithms such as hierarchical clustering [50], [51], principle component analysis (PCA) [52] [53], genetic algorithms [54], and artificial neural networks [55] [56] [57], have been used to cluster gene expression data. However, in 2003, Xiao *et al.* [58] used a new approach based on the synergism of the PSO and the Self Organizing Maps (SOM) for clustering them. Authors achieved promising results by applying the hybrid SOM-PSO algorithm over the gene expression data of Yeast and Rat Hepatocytes. We will briefly discuss their approach in the following paragraphs.

The idea of the SOM [59] stems from the orderly mapping of information in the cerebral cortex. With SOM, high dimensional datasets are projected onto a one- or two- dimensional space. Typically, a SOM has a two dimensional lattice of neurons and each neuron represents a cluster. The learning process of SOM is unsupervised. All neurons compete for each input pattern; the neuron that is chosen for the input pattern wins it.

Xiao et al. [58] used PSO to evolve the weights for SOM. In the first stage of the hybrid SOM/PSO algorithm, SOM is used to cluster the dataset. Authors used a SOM with conscience at this step. Conscience directs each component that takes part in competitive learning toward having the same probability to win. Conscience is added to SOM by assigning each output neuron a bias. The output neuron must overcome its own bias to win. The objective is to obtain a better approx. of pattern distribution. The SOM normally runs for 100 iterations and generates a group of weights. In the second stage, PSO is initialized with the weights produced by SOM in the first stage. Then a *gbest* PSO is used to refine the clustering process. Each particle consists of a complete set of weights for SOM. The dimension of each particle is the number of input neurons of SOM times the number of output neurons of SOM. The objective of PSO is to improve the clustering result by evolving the population of particles.

4.4.2 The Molecular Docking Problem

Formally, the protein-ligand docking problem may be described as: We are given a geometric and chemical description of a protein and an arbitrary small organic molecule. We want to determine computationally whether the small molecule will

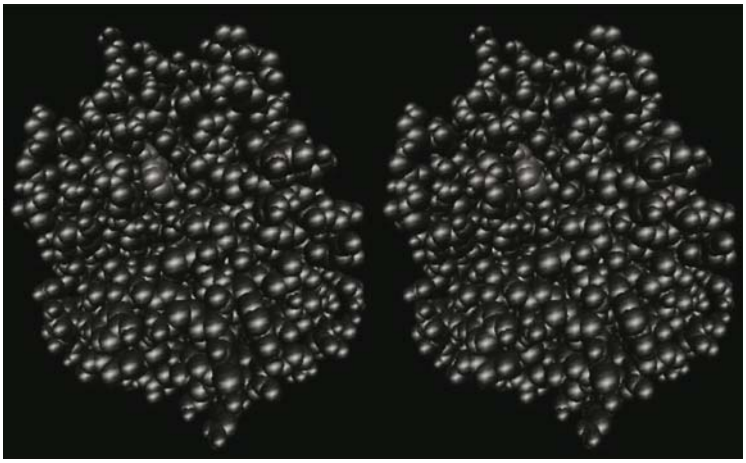


Fig. 4.12. Stereo view of benzamidine docked in the active site of trypsin

bind to the protein, and if so, we would like to estimate the geometry of the bound complex, as well as the affinity of the binding. Figure 4.12 illustrates how benzamidine, a trypsin inhibitor, docks into the active site of trypsin, a protease involved in digestion (adapted from [60]).

Liu *et al.* [61] proposed a novel PSO based docking technique, which they called SODOCK (Swarm Optimization for molecular DOCKing). After comparing with a number of state of the art docking techniques like GOLD 1.2 [62], AutoDock 3.05 [63], DOCK 4.0 [64] etc., they found promising results for SODOCK in terms of robustness, accuracy and the speed of convergence. In SODOCK, three kinds of parameters are optimized using the PSO:

- **Translation**: three parameters in this category specify the translation of the center of the ligand with respect to the center of a 3D grid box that encloses the binding site of the protein.
- **Orientation**: There are four parameters n_x,n_y,n_z and α where n_x,n_y,n_z $\varepsilon[0,1]$ specify the normal vector of the ligand whereas $\alpha\varepsilon[-\pi,\pi]$ represent the angle of self rotation along the normal vector.
- **Torsions**: These are torsion angles $tor_i\varepsilon[-\pi,\pi]$ associated with the rotating bonds, $i = 1,2,\ldots,T$.

Thus, the PSO algorithm is used to evolve a total of $N = 7+\mathrm{T}$ parameters such that the following docking energy function is minimized:

$$E_{tot} = E_{vdw} + E_{H-bond} + E_{pot} + E_{intern} \tag{4.9}$$

The first three terms in the above expression, correspond to the intermolecular energies: van der Waals force, hydrogen bonding, and electronic potential. The last term represents the internal energy of the ligand, which also consists of the three elements.

The fitness landscape of the energy function shown in 4.9 is usually riddled with multiple local minima. In order to tackle these local peaks efficiently, Liu *et al.* integrated a local search strategy (a variant of the Solis and Wet local search [63]) in SODOCK. A generation of SODOCK has four stages: update of velocity, move of particle, local search, and update of local and global best positions. The local search may be applied to the particle according to a predefined probability P_{ls}. Finally, the local and global best positions of particles are updated if their energies are improved. The particles having the smallest energy correspond to a solution to the flexible docking problem.

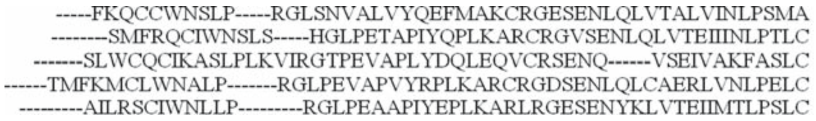
4.4.3 The Multiple Sequence Alignment Problems (MSA)

Sequence alignment refers to the process of arranging the primary sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. Given two sequences X and Y, a pair-wise alignment indicates positions of each sequence that are considered to be functionally or evolutionarily related. From a family $S = (S_0, S_1, \ldots, S_{N-1})$ of N sequences, we would like to find out the common patterns of this family. Since aligning each pair of sequences from S separately often does not reveal the common information, it is necessary to perform multiple sequence alignment (MSA). A multiple sequence alignment (MSA) is a sequence alignment of three or more biological sequences, generally protein, DNA, or RNA. In general, the input set of query sequences are assumed to have an evolutionary relationship by which they share a linkage and are descended from a common ancestor. An example of multiple alignments of five sequences is illustrated in Figure 4.13.

To evaluate the quality of an alignment, a popular choice is to use the SP (sum of pairs) score method [65]. The SP score basically sums the substitution scores of all possible pair-wise combinations of sequence characters in one column of a multiple sequence alignment. Assuming c_i representing the $i-th$ character of a given column in the sequence matrix and match (c_i, c_j) denoting the comparing score between characters c_i and c_j, the score of a column may be computed using the formula:

$$SP = (c_1, c_2, \ldots, c_N) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} match(c_i, c_j) \tag{4.10}$$

Progressive alignment is a widely used heuristic MSA method that does not guarantee any level of optimality [66]. ClustalW [67] is another widely popular program



```
    -----FKQCCWNSLP-----RGLSNVALVYQEFMAKCRGESENLQLVTALVINLPSMA
   --------SMFRQCIWNSLS-----HGLPETAPIYQPLKARCRGVSENLQLVTEIIINLPTLC
  -------SLWCQCIKASLPLKVIRGTPEVAPLYDQLEQVCRSENQ------VSEIVAKFASLC
------TMFKMCLWNALP-------RGLPEVAPVYRPLKARCRGDSENLQLCAERLVNLPELC
  ---------AILRSCIWNLLP---------RGLPEAAPIYEPLKARLRGESENYKLVTEIIMTLPSLC
```

Fig. 4.13. An example of multiple sequence alignments

that improved the algorithm presented by Feng and Doolittle [66]. The main shortcoming of ClustalW is that once a sequence has been aligned, that alignment can never be modified even if it conflicts with sequences added later.

Recently, Chen et al. [68] took a serious attempt to solve the classical MSA problem by using a partitioning approach coupled with the ACO algorithm. Authors algorithm consists of three stages. At first a genetic algorithm is employed to find out the near optimal cut-off points in the original sequences from where they must be partitioned vertically. In this way a partitioning method is continued recursively to reduce the original problem to multiple smaller MSA problems until the lengths of the subsequences are all less than an acceptable threshold. Next, an ant colony system is used to align each small subsection derived from the previous step. The ant system consists of N ants each of which represents a solution of alignment. Each ant searches for an alignment by moving on the sequences to choose the matching characters. Let the N sequences be $S = S_0, S_1, \ldots, S_{N-1}$. In that case an artificial ant starts from $S_0[0]$, the first character of S_0, and selects one character from each of the sequences of $S_1, \ldots, S_{N-1}$ matching with $S_0[0]$. From the sequence $S_i, i = 1, 2, \ldots, n_1$, the ant selects a character $S_i[j]$ by a probability determined by the matching score with $S_0[0]$, deviation of its location from $S_0[0]$ and pheromones trail on the logical edge between $S_i[j]$ and $S_0[0]$. In addition, an ant may choose to insert an empty space according to a predetermined probability. Next, the ant starts from $S_0[1]$, selects the characters of $S_1, \ldots, S_{N-1}$ matching with $S_0[1]$ to form the second path. Similarly, starting from $S_0[2], \ldots, S_0[|S_0| - 1]$, the ant can form other paths. Here $|S_0|$ indicates the number of characters in the sequence $|S_0|$. All these $|S_0|$ paths forming an alignment solution is reproduced in Figure 4.14.

To evaluate an alignment represented by a set of paths, the positions of characters not selected by the ants are calculated first by aligning them to the right and adding gaps to the left. Next their SP (sum-of-pairs) score is using relation 4.9. Finally, a solution to the MSA is obtained by concatenating the results from smaller sub-alignments. Chen *et al.* showed that the Divide-Ant-MSA algorithm outperforms the SAGA [69] a leading MSA program based on genetic algorithm (GA) in terms of both speed and accuracy especially for longer sequences.

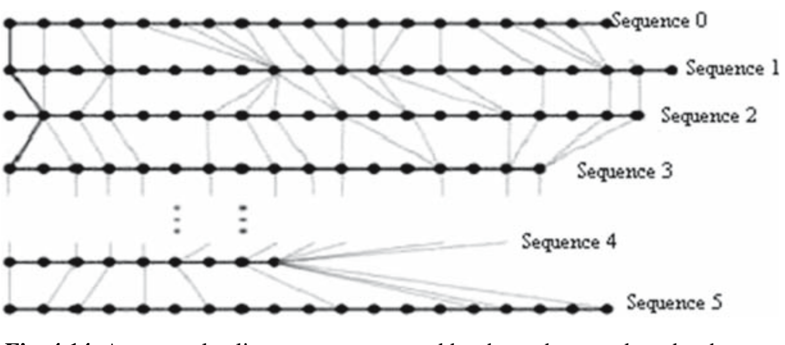


Fig. 4.14. An example alignment as presented by the paths traced out by the ants

Rasmussen and Krink in [70] focussed on a new PSO based training method for Hidden Markov models (HMMs) in order to solve the MSA problem. The authors demonstrated how the combination of PSO and evolutionary algorithms can generate better protein sequence alignments than with more traditional HMM training methods, such as Baum-Welch [71] and simulated annealing [72].

4.4.4 The Construction of Phylogenetic Trees

Every species on earth undergo slow change of their hereditary traits in course of evolution. The phylogenetic or evolutionary trees are schematic binary trees showing the evolutionary interrelationships among various species that are believed to have a common ancestor [15]. The leaves of such a tree represent the present day species while the interior nodes represent the hypothesized ancestors. Phylogenetic trees may be rooted or un-rooted (Figure 4.15). An un-rooted tree simply represents phylogenies but does not provide an evolutionary path. In case of a rooted tree, one of the internal nodes is used as an out-group, and, in essence, becomes the common ancestor of all the other external nodes. The out-group therefore enables the root of a tree to be located and the correct evolutionary pathway to be identified.

In a phylogenetic tree, the phylogenies are reconstructed based on comparisons among the present-day objects. The term object is used to denote the units for which one wants to reconstruct the phylogeny. The input data essential for constructing phylogeny are of two types [15].

- Discrete characters, such as beak shape, number of fingers, presence or absence of a molecular restriction site. Each character can have a finite number of states. The data relative to these characters are placed in an objects character matrix called character state matrix.
- Comparative numerical data, called distances between objects. The resulting matrix is called distance matrix.

Given data (character state matrix or distance matrix) for n taxa (object), the phylogenetic tree reconstruction problem is to find the particular permutation of taxa

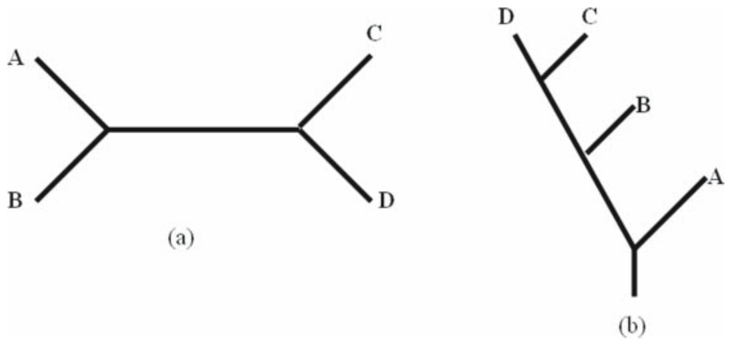


Fig. 4.15. Topologies of phylogenetic trees (a) un-rooted (b) rooted

that optimize the criteria (distance). Felenstein has shown that considering n species, it is possible to construct N number of trees where, N is given by the following equations for unrooted trees and rooted trees [73]:

$$N = \frac{(2n-3)!}{2^{n-2}(n-2)!} \tag{4.11}$$

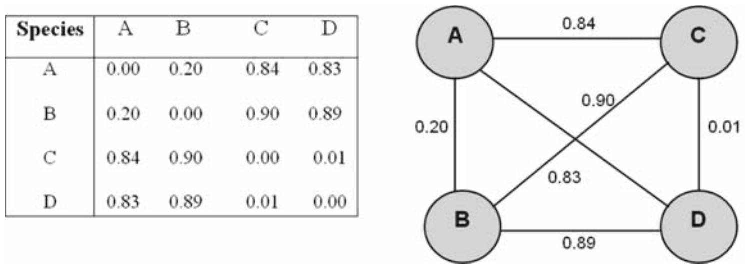
$$N = \frac{(2n-5)!}{2^{n-3}(n-3)!} \tag{4.12}$$

The problem poses severe computational challenge before us, example, if we would like to find the best tree using the method of maximum similarity for (only) 15 species, we should try 213, 458, 046, 676, 875 trees.

The phylogenetic tree construction problem bears close resemblance to a standard TSP, (Traveling Salesman Problem) described earlier in Section 4.3.2. One can simply associate one imaginary city to each taxa, and define as the distance between two cities the data obtained from the data matrix for the corresponding pair of taxas. This kind of formulation of the problem paves the path for the application of heuristic algorithms like GA [74], [75] and ACO. Perretto *et al.* [76] proposed a slightly modified artificial ant colony based algorithm for the construction of phylogenetic trees. Their approach starts with building a two-dimensional fully-connected graph using the distance matrix among the species. In this graph, nodes represent the species and edges represent the evolutionary distances between species. An example of such a graph is provided in Figure 4.16. The ants start from a randomly selected node and continue traveling across the structured graph. At each node a transition function similar in form to equation 4.2, determines its direction.

The method described in [76], differs from the classical ant systems based TSP in only one respect. In case of the former algorithm, moves are made between nodes, but here, the ant system creates an intermediary node between the two previously selected ones. This node will represent the ancestral species of the other two, and it will not be in the list of nodes (species) to be set in the tree. Using such an intermediary node, distances to the remaining nodes (species) are recomputed.

The ants initially start from a randomly selected node and continue traveling across the structured graph. At each node a transition function similar in form to



Species	A	B	C	D
A	0.00	0.20	0.84	0.83
B	0.20	0.00	0.90	0.89
C	0.84	0.90	0.00	0.01
D	0.83	0.89	0.01	0.00

Fig. 4.16. Distance matrix for four species and the corresponding two-dimensional graph

equation 4.2 determines its direction. In original ACO based algorithm for TSP, moves are made between nodes. But here, the ant system creates an intermediary node between the two previously selected ones. This node will represent the ancestral species of the other two, and it will not be in the list of nodes (species) to be set in the tree. Using such an intermediary node, distances to the remaining nodes (species) are recomputed. This procedure is repeated until all nodes belong to the list of already visited nodes, and then a path is constructed. The score of this path is given by the sum of the transition probabilities of the adjacent nodes of the path. Paths constructed by the ants are then used for updating the pheromone trail. An increment of the pheromone trail is made at all nodes belonging to at least one path, created in an execution cycle. This key point helps to avoid trapping in a local maximum. In this way, following an algorithm very close in spirit to the ant colony algorithm for solving the TSP, the phylogenetic trees may be reconstructed efficiently.

Ando and Iba [77] proposed an ant algorithm for the construction of evolutionary trees from a given DNA sequence. Authors algorithm searches for a tree structure that minimizes the score for a given set of DNA sequences. It uses the mutual distance matrix of the leaves as the input. While the ACO for TSP visits the respective cities once to construct a round trip, ants in tree constructing algorithm visit leaves and vertices of the tree to construct a suffix representation of the bifurcating tree. The algorithm is shown to compete with conventional methods of the exhaustive search or the sequential insertion method, taken by the most popular methods.

4.4.5 The RNA Secondary Structure Prediction

Ribonucleic acid (RNA) is a nucleic acid polymer (like DNA) consisting of nucleotide monomers. Unlike deoxyribonucleic acid (DNA), which contains deoxyribose and thymine, RNA nucleotides contain ribose rings and uracil. As pointed out in Section 4.2.2, RNA serves as the template for translation of genes into proteins, transferring amino acids to the ribosome to form proteins, and also translating the transcript into proteins.

Like protein secondary structure (discussed in Section 4.2.3), RNA secondary structure may be conveniently viewed as an intermediate step in the formation of a three dimensional structure [13]. RNA secondary structure is composed primarily of double-stranded RNA regions formed by folding the single-stranded molecule on itself. To produce such double-stranded regions, a downstream sequence of the bases in RNA must be complementary to another upstream sequence so that Watson-Crick base pairing can occur between the complementary nucleotides G-C and A-U (analogous to the G-C and A-T base pairs in DNA). Among the several recognizable “domains” of secondary structure three well known ones are hairpin loops, bulges and internal loops. Figure 4.17 shows the folding of a single stranded RNA molecule into a hairpin structure.

Secondary structure of RNA molecules can be predicted computationally by calculating the minimum free energy (MFE) structure for all different combinations of hydrogen bonds and domains. Neethling and Engelbrecht [78] attempted to optimize the structure of RNA molecules using a modified PSO algorithm. The SetPSO, which

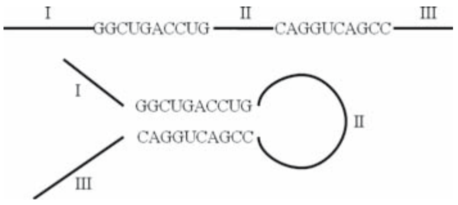


Fig. 4.17. Illustrating the formation of a hair-pin RNA secondary structure

the authors proposed for this purpose, can operate on mathematical sets in order to solve set-based combinatorial optimization problems in discrete search spaces. In SetPSO, the position of each particle denotes a set and this necessitates the redefinition of the addition and subtraction operators suitably. The addition of the position vectors of two particles here essentially means the union of the two sets, which they represent (i.e. $A+B$ now represents the set of all elements which belong to both A and B). On the other hand the subtraction operation is basically the set-theoretic difference between two sets A and B (i.e. $A-B$ denotes a set of all elements which belong to A but not to B).

In the course of folding back of RNA, the process of binding of the adjacent complementary bases is known as stacking. A stack or stem representing a valid RNA secondary structure should satisfy a few constraints like each base can pair with only one canonical base, no pseudo knots should be allowed etc. The collection of all feasible stems (i.e. those obeying the constraints) forms a universal set U. Each particle of the SetPSO is then initialized as a randomly chosen subset of U. Positions and velocities of these particles are updated using the modified addition and subtraction operators with a view to minimizing the thermodynamic free energy function defined for the RNA structure [79]. Although the SetPSO based algorithm yielded near-optimal configuration for RNA molecules in a number of benchmarks, further research is necessary to select a more robust energy function, which can eliminate the formation of pseudo-knots in the predicted structure.

4.4.6 Protein Secondary Structure Prediction

Protein secondary structures have already been introduced in Section 4.2.3. Protein structures are primarily determined by techniques such as NMRI (nuclear-magnetic resonance imaging) and X-ray crystallography, which are expensive in terms of equipment-cost, computation and time. In addition, they require isolation, purification and crystallization of the target protein. Computational approaches to protein structure prediction are therefore very attractive and cost effective. Since the processes involved in the folding of proteins are very complex and only partially understood simplified models like Dill's Hydrophobic-Polar (HP) have become one of the major tools for studying proteins [80]. The HP model is based on the observation that hydrophobic interaction is the driving force for protein folding and the hydrophobicity of amino acids is the main force for development of a native conformation of small globular proteins [80], [81]. In the HP model, each amino acid can be either of

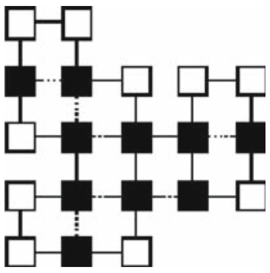


Fig. 4.18. A sample protein conformation in the 2D HP model. The underlying protein sequence (Sequence 1 from Table 1) is HPHPPHHPHPPHPHHPPHPH; black squares represent hydrophobic amino-acids, while white squares symbolize polar amino-acids

two types: *H* (hydrophobic, i.e., non-polar) or *P* (hydrophilic, i.e., polar). For simplicity, we denote *H* by "1" (black) and *P* by "0" (white). The protein conformations of this sequence are restricted to self-avoiding paths on a lattice. An example for a protein conformation under a 2D HP lattice model is illustrated in Figure 4.18.

But, finding the optimal folds even in case of the simplest two-dimensional HP model is computationally hard and knows no polynomial time solution [82]. Shmygelska and Hoos proposed a modified version of the ACO for solving this NP-hard problem [83], [84]. The ants, in their method, first randomly select a starting point within the given protein sequence. From this starting point, the given protein sequence is folded in both directions, adding one amino-acid symbol at a time. In this way, the tours of these ants construct candidate conformation for a given HP protein sequence, apply local search to achieve further improvements, and finally update the pheromone trails based on the quality of the solutions found. The ant system incorporates a local search element as a means of by-passing local minima and preventing the algorithm from premature convergence.

Chu *et al.* [85] extended the 2-D solutions of the HP protein folding problems to the 3-D case by using a parallel ant colony system. They proposed a Multi Ant Colony Optimization (MACOS) algorithm for optimizing the 3-D HP lattice configuration. The MACOS utilizes multiple colonies of artificial ants. It employs separate pheromone matrices for each colony and allows limited cooperation between different colonies.

4.4.7 Fragment Assembly Problem (FAP)

The fragment assembly problem (FAP) deals with the sequencing of DNA. Currently strands of DNA, longer than approximately 500 base pairs, cannot be sequenced very accurately. As a consequence, in order to sequence larger strands of DNA, they are first broken into smaller pieces. The FAP is then to reconstruct the original molecule's sequence from the smaller fragment sequences. FAP is basically a permutation problem, similar in spirit to the TSP, but with some important differences (circular tours, noise, and special relationships between entities) [15]. It is also NP-complete in nature. Meksangsouy and Chaiyaratana [86] attempted to solve the DNA fragment reordering problem with the ant colony systems. The authors investigated two types of assembly problems: single-contig and multiple-contig problems. The simulation results indicate that in single-contig problems, the ant colony system algorithm outperforms the nearest neighbor heuristic algorithm when multiple-contig problems are considered.

4.5 A Few Open-ended Problems and Future Research Directions

In the last section, we discussed the research works already undertaken for making an efficient use of SI tools in bioinformatics. The papers published in this context, may be small in volume, but are of immense significance to the researchers of tomorrow. We note that the SI algorithms are yet to be applied to a huge lot of NP-hard problems from computational biology, for which no universally acceptable solution is known till date. In the present Section, we address a few research problems of this kind and provide hints on their possible solution through the application of SI algorithms.

4.5.1 Identifying Gene Regulatory Networks

A Gene Regulatory Network (GRN) may be defined as a collection of genes in a cell that interact with one another, governing the rate of transcription [87]. Inferring the network from gene expression data obtained through DNA microarray constitutes one of the most challenging problems in the field of bioinformatics. Genes can be viewed as nodes in a complex network, with input being proteins such as transcription factors, and outputs being the level of gene expression. The node itself can also be viewed as a function which can be obtained by combining basic functions upon the inputs (in the Boolean network described below these are Boolean functions or gates computed using the basic AND, OR and NOT gates in electronics). These functions have been interpreted as performing a kind of information processing within the cell which determines cellular behavior.

PSO can be utilized very effectively to solve the GRN identification problem. Each particle may represent the real valued expression levels of all the genes. Each gene has a specific expression level for another gene; thus a total of N genes correspond to N^2 expression levels. Fitness of the particles may be computed from the absolute error with generated expression pattern (sum of all expressions) from the target expression pattern. Investigations of the same problem with evolutionary algorithms can be found in [88], [89], [90], [91].

4.5.2 Protein Tertiary Structure Prediction and Folding

Once a protein sequence has been determined, deducing its unique 3-D native structure is a daunting task. Experimental methods to determine detailed protein structure, such as x-ray diffraction studies and nuclear magnetic resonance (NMR) analysis, are highly labor intensive. Since it was discovered that proteins are capable of folding into their unique functional 3D structures without any additional genetic mechanisms, over 25 years of effort has been expended into the prediction of 3D structure from sequence. Despite the large amount of effort expended, the protein folding or protein structure prediction problem, as it has come to be known, remains largely unsolved [92].

Since PSO is known as a fast and accurate global optimization method, it may be integrated in the *ab initio* approach to protein tertiary structure prediction [93], [94], [95]. The *ab initio* approach is a mixture of science and engineering. The science is in understanding how the three-dimensional structure of a protein is attained. The engineering portion is in finding the 3-Dstructure from a given the sequence. The *ab initio* folding process can be broken down into two components: devising a scoring function that can distinguish between correct/good (native or native like) structures from incorrect (non-native) ones, and a search method to explore the conformational space. The PSO may be used in the searching phase in order to enhance the performance of the process as a whole.

4.5.3 Characterization of Metabolic Pathways between Different Genomes

In biochemistry, a metabolic pathway is a series of chemical reactions occurring within a cell, catalyzed by enzymes, resulting in either the formation of a metabolic product to be used or stored by the cell, or the initiation of another metabolic pathway (then called a flux generating step). Many pathways are elaborate, and involve a step by step modification of the initial substance to shape it into the product with the exact chemical structure desired [96].

The goal of characterizing the metabolic pathways is to estimate the "best" set of parameter values, which minimizes the error between the process data and the model metabolic network response. This parameter estimation problem can be formulated as a non-convex, nonlinear optimization problem and can therefore be solved using global optimization techniques. This feature makes the problem ideal for the application of algorithms like PSO.

4.5.4 Characterization of Metabolic Pathways between Different Genomes

One of the most promising applications of bioinformatics appears in computer-aided molecular design (CAMD). In pharmaceutical development, this effort is focused on modeling the drugs and the biological receptors that the drugs bind to so that better binding, and therefore, more potent or precise drugs can be developed [97], [98]. SI algorithms like PSO may find important applications for the design of a ligand molecule, which can bind to the active site of a target protein.

Unlike the GA based methods [99], [100], PSO or ACO have not been applied to the molecular design problem till date. The formulation of the drug design problem with PSO requires the control of a fitness function. The fitness function must be capable of determining which of two arbitrary molecules is better for a specific task. The algorithm may begin by generating a population of particles each representing one randomly oriented molecule. The individual molecules in a population are then evolved towards greater fitness by using the velocity updating schemes of the PSO or its variants. However, finding a suitable representation of the particles (thereby enabling them to contain information about the each random molecule) constitutes a major research issue in this direction.

4.6 Conclusions

With an explosive growth of the annotated genomic sequences in available form, bioinformatics has emerged as a challenging and fascinating field of science. It presents the perfect harmony of statistics, biology and computational intelligence methods for analyzing and processing biological information in the form of gene, DNA, RNA and proteins. SI algorithms on the other hand, have recently gained wide popularity among the researchers, for their amazing ability in finding near optimal solutions to a number of NP hard, real world search problems. A survey of the bioinformatics literature reveals that the field has a plethora of problems that need fast and robust search mechanisms. Problems belonging to this category include (but are not limited to) the multiple sequence alignment (MSA), protein secondary and tertiary structure prediction, protein ligand docking, promoter identification and the reconstruction of evolutionary trees. Classical deterministic search algorithms and the derivative based optimization techniques are of no use for them as the search space may be enormously large and discontinuous at several points. SI presents a collection of multi-agent parallel search techniques which can be very effective for solving bioinformatics related tasks of this kind. We fervently hope that the SI community will make significant contribution to the emerging research area of modern computational biology in near future.

This article surveyed several important applications of SI tools in bioinformatics. We also illustrated a few open-ended research problems of computational biology, where the SI algorithms like PSO and ACO may find very good use. Even though the current approaches in biological computing with SI algorithms are very helpful in identifying patterns and functions of proteins, genes etc., the final results are far from being perfect.

There are a few general issues which should be addressed by the researchers in future in order to exploit the SI algorithms to their full potential in bioinformatics: firstly, the basic velocity updating scheme in PSO or the pheromone trail updating mechanism in ACO are common to all applications; research should now focus on the design of problem specific operators to get better results. Secondly, the parameters of the ACO or PSO require extensive experimentation so that the appropriate range of values can be identified for different bioinformatics tasks. Finally, algorithms like PSO or ACO and their variants involve a large degree of randomness and different runs of the same program may yield different results; so it is necessary to incorporate problem specific domain knowledge in the SI tools to reduce randomness and computational time and current research should progress in this direction also.

References

1. Baldi P and Brunak S (1998) Bioinformatics: The Machine Learning Approach, MIT Press, Cambridge, MA.
2. Altman RB, Valencia A, Miyano S and Ranganathan, S (2001) Challenges for intelligent systems in biology, IEEE Intelligent Systems, vol. 16, no. 6, pp. 14–20.
3. Haykin S. (1999) Neural Networks: A Comprehensive Foundation, Prentice Hall.
4. Holland JH (1975) Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor.
5. Goldberg DE (1975) Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA.
6. Mitra S and Hayashi Y (2006) Bioinformatics with soft computing, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol. 36, pp. 616–635.
7. Bonabeau E, Dorigo M, Theraulaz G (2001) Swarm intelligence: From natural to artificial systems. Journal of Artificial Societies and Social Simulation, 4(1).
8. Engelbrecht AP (2005) Fundamentals of Computational Swarm Intelligence. Wiley.
9. Beni G and Wang U (1989) Swarm intelligence in cellular robotic systems. In NATO Advanced Workshop on Robots and Biological Systems, Il Ciocco, Tuscany, Italy.
10. Kennedy J, Eberhart R (1995) Particle swarm optimization, In Proceedings of IEEE International conference on Neural Networks. 1942–1948.
11. Dorigo M (1992) Optimization, learning, and natural algorithms, Ph.D. dissertation (in Italian), Dipartimento di Elettronica, Politecnico di Milano, Milano, Italy.
12. Dorigo M, Di Caro G and Gambardella L (1999) Ant colony optimization: A new meta-heuristic. In PJ Angeline, Z Michalewicz, M Schoenauer, X Yao, and A Zalzala, (eds), Proceedings of the Congress on Evolutionary Computation, IEEE Press, Vol. 2, pp. 1470–1477.
13. Lewin B (1995) Genes VII. Oxford University Press, New York, NY.
14. Wu AS and Lindsay RK (1996) A Survey of Intron Research in Genetics, In Proc. 4th Conf. of on Parallel Problem Solving from Nature, pp. 101–110.
15. Setubal J and Meidanis J (1999) Introduction to Computational Molecular Biology, International Thomson Publishing, 20 park plaza, Boston, MA 02116.
16. http://en.wikipedia.org/wiki/DNA_microarray
17. Couzin ID, Krause J, James R, Ruxton GD, Franks NR (2002) Collective Memory and Spatial Sorting in Animal Groups, Journal of Theoretical Biology, 218, pp. 1–11.
18. Krause J and Ruxton GD (2002) Living in Groups. Oxford: Oxford University Press.
19. Partridge BL, Pitcher TJ (1980) The sensory basis of fish schools: relative role of lateral line and vision. Journal of Comparative Physiology, 135, pp. 315–325.
20. Partridge BL (1982) The structure and function of fish schools. Science American, 245, pp. 90–99.
21. Major PF, Dill LM (1978) The three-dimensional structure of airborne bird flocks. Behavioral Ecology and Sociobiology, 4, pp. 111–122.
22. Branden CI and Tooze J (1999) Introduction to Protein Structure: 2nd edition. Garland Publishing, New York, 2nd edition.
23. Grosan C, Abraham A and Monica C (2006) Swarm Intelligence in Data Mining, in Swarm Intelligence in Data Mining, Abraham A, Grosan C and Ramos V (Eds), Springer, pp. 1–16.
24. Milonas MM (1994) Swarms, phase transitions, and collective intelligence, In Langton CG Ed., Artificial Life III, Addison Wesley, Reading, MA.

25. Serra R and Zanarini G (1990) Complex Systems and Cognitive Processes. New York, NY: Springer-Verlag.
26. Flake G (1999) The Computational Beauty of Nature. Cambridge, MA: MIT Press.
27. Kennedy J, Eberhart R and Shi Y (2001) Swarm Intelligence, Morgan Kaufmann Academic Press.
28. Dorigo M and Gambardella LM (1996) A Study of Some Properties of Ant Q, in Proc. PPSN IV - 4th Int. Conf. Parallel Problem Solving From Nature, Berlin, Germany: Springer-Verlag, pp. 656–665.
29. Dorigo M and Gambardella LM (1997) Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput., vol. 1, pp. 53–66.
30. Deneubourg JL (1997) Application de I'ordre par fluctuations? la descriptio de certaines? tapes de la construction dun id chez les termites, Insect Sociaux, vol. 24, pp. 117–130.
31. Dorigo M, Maniezzo V and Colorni A (1996) The ant system: Optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B, vol. 26.
32. Kennedy J (1999) Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance, Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, IEEE Press, pp. 1931–1938.
33. Kennedy J and Mendes R (2002) Population structure and particle swarm performance. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), IEEE Press, pp. 1671–1676.
34. Watts DJ and Strogatz SH (1998) Collective dynamics of small-world networks. Nature, 393, 440–442.
35. Dall'Asta L, Baronchelli A, Barrat A and Loreto V (2006) Agreement dynamics on small-world networks. Europhysics Letters.
36. Barrat A and Weight M (2000) On the properties of small-world network models. The European Physical Journal, 13, pp. 547–560.
37. Moore C and Newman MEJ (2000) Epidemics and percolation in small-world networks. Physics. Review. E 61, 5678–5682.
38. Jasch F and Blumen A (2001) Trapping of random walks on small-world networks. Physical Review E 64, 066104.
39. Chen J, Antipov E, Lemieux B, Cedeno W, and Wood DH (1999) DNA computing implementing genetic algorithms, Evolution as Computation, Springer Verlag, New York, pp. 39–49.
40. Vesterstrom J and Thomsen R (2004) A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, In Proceedings of the IEEE Congress on Evolutionary Computation (CEC 04), IEEE Press, pp. 1980–1987.
41. Das S, Konar A, Chakraborti UK (2005) A New Evolutionary Algorithm Applied to the Design of Two-dimensional IIR Filters in ACM-SIGEVO Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2005), Washington DC.
42. Hassan R, Cohanim B and de Weck O (2005) Comparison of Particle Swarm Optimization and the Genetic Algorithm, AIAA-2005-1897, 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference.
43. Luscombe NM, Greenbaum D and Gerstein M (2001) What is Bioinformatics? A Proposed Definition and Overview of the Field, Yearbook of Medical Informatics, pp. 83–100.
44. Quackenbush J (2001) Computational analysis of microarray data, National Review of Genetics, vol. 2, pp. 418–427.

45. Special Issue on Bioinformatics, IEEE Computer, vol. 35, July 2002.
46. Jain AK, Murty MN and Flynn, PJ (1999) Data clustering: a review, ACM Computing Surveys, vol. 31, no. 3, pp. 264–323.
47. Baker TK, Carfagna MA, Gao H, Dow ER, Li O, Searfoss GH, and Ryan TP (2001) Temporal Gene Expression Analysis of Monolayer Cultured Rat Hepatocytes, Chem. Res. Toxicol., Vol. 14, No. 9.
48. Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D and Levine AJ (1999) Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays, Proc. Natl. Acad. Sci. USA, Cell Biology, Vol. 96, pp. 6745–6750.
49. Maurice G and Kendall M (1961) The Advanced Theory of Statistics, Vol. 2, Charles Griffin and Company Limited.
50. Wen X, Fuhrman S, Michaels GS, Carr DB, Smith S, Barker JL, and Somogyi R (1998) Large-scale temporal gene expression mapping of central nervous system development, Proc. Natl. Acad. Sci. USA, Neurobiology, Vol. 95, pp. 334–339.
51. Spellman EM, Brown PL, Brown D (1998) Cluster Analysis and Display of Genome-wide expression patterns, Proc. Natl. Acad. Sci. USA 95: 14863–14868.
52. Yeung KY, Ruzzo WL (2001) Principal Component Analysis for Clustering Gene Expression Data, Bioinformatics, 17, pp. 763–774.
53. Raychaudhuri S, Stuart JM and Altman RB (2000) Principal Components Analysis to Summarize Microarray Experiments: Application to Sporulation Time Series, Pacific Symposium on Biocomputing 2000, Honolulu, Hawaii, pp. 452–463.
54. Li L, Weinberg CR, Darden TA and Pedersen LG (2001) Gene Selection for Sample Classification Based on Gene Expression Data: Study of Sensitivity to Choice of Parameters of the GA/KNN Method, Bioinformatics, 17, pp. 1131–1142.
55. Herrero J, Valencia A and Dopazo J (2001) A hierarchical unsupervised growing neural network for clustering gene expression patterns, Bioinformatics, 17, pp. 126–136.
56. Tamayo P, Slonim D, Mesirov J, Zhu Q, Kitareewan S, Dmitrovsky E, Lander ES and Golub TR (1999) Interpreting patterns of gene expression with self organizing maps: Methods and applications to hematopoietic differentiation. PNAS, 96, pp. 2907–2912.
57. Toronen P, Kolehmainen M, Wong G, Castren E (1999) Analysis of Gene Expression Data Using Self-organizing Maps, FEBS letters 451, pp. 142–146.
58. Xiao X, Dow ER, Eberhart RC, Miled ZB and Oppelt RJ (2003) Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization, Proc of the 17th International Symposium on Parallel and Distributed Processing (PDPS '03), IEEE Computer Society, Washington DC.
59. Kohonen T (1995) Self-organizing Maps, 2nd ed., Springer-Verlag, Berlin.
60. http://cnx.org/content/m11456/latest/
61. Liu BF, Chen HM, Huang HL, Hwang SF and Ho SY (2005) Flexible protein-ligand docking using particle swarm optimization, in Proc. of Congress on Evolutionary Computation (CEC 2005), IEEE Press, Washinton DC.
62. Jones G, Willett P, Glen RC, Leach AR and Taylor R (1997) Development and validation of a genetic algorithm for flexible docking. Journal of Molecular Biology, 267(3): pp. 727–748.
63. Morris GM, Goodsell DS, Halliday RS, Huey R, Hart WE, Belew RK and Olson AJ (1998) Automated docking using a lamarckian genetic algorithm and an empirical binding free energy function. Journal of Computational Chemistry, 19(14): pp. 1639–1662.

64. Ewing TJA, Makino S, Skillman AG and Kuntz ID (2001) Dock 4.0: Search strategies for automated molecular docking of flexible molecule databases. Journal of Computer-Aided Molecular Design, 15(5): pp. 411–428.
65. Lipman DJ, Altschul SF and Kececioglu JD (1989). A tool for multiple sequence alignment. Proc. Natl. Acad. Sci. USA, 86: pp. 4412–4415.
66. Feng DF, Doolittle RF (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J. Mol. Evol. 25, pp. 351–360.
67. Thompson JD, Higgins DG and Gibson TJ (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. Nucleic Acids Research, vol. 22, No. 22, pp. 4673–4680.
68. Chen Y, Pan Y, Chen L, Chen J (2006) Partitioned optimization algorithms for multiple sequence alignment, Proc. of the 20th International Conference on Advanced Information Networking and Applications - (AINA'06), IEEE Computer Society Press, Washington DC., Volume 02, pp. 618–622.
69. Notredame C and Higgins DG, SAGA: sequence alignment by genetic algorithm, Nucleic Acids Research, vol. 24, no. 8, pp. 1515–1524.
70. Rasmussen TK and Krink T (2003) Improved hidden Markov model training for multiple sequence alignment by a particle swarm optimization-evolutionary algorithm hybrid, BioSystems 72 (2003).
71. Stolcke A and Omohundro S (1993) Hidden Markov Model induction by Bayesian model merging. In NIPS 5, pp. 11–18.
72. Hamam Y and Al-Ani T (1996). Simulated annealing approach for Hidden Markov Models. 4th WG-7.6 Working Conference on Optimization-Based Computer-Aided Modeling and Design, ESIEE, France.
73. Felsenstein J (1973). Maximum likelihood estimation of evolutionary trees from continuous characters.Am. J. Hum. Gen. 25: 471–492.
74. Lewis PO (1998), A genetic algorithm for maximum likelihood phylogeny inference using nucleotide sequence data, Molecular Biology and Evolution, vol. 15, no. 3, pp. 277–283.
75. Lemmon AR and Milinkovitch MC (2002) The metapopulation genetic algorithm: An efficient solution for the problem of large phylogeny estimation, Proc. Natl Acad Sci U S A., vol. 99, no. 16, pp. 10516–10521.
76. Perretto M and Lopes HS (2005) Reconstruction of phylogenetic trees using the ant colony optimization paradigm, Genetic and Molecular Research 4 (3), pp. 581–589.
77. Ando S and Iba H (2002) Ant algorithm for construction of evolutionary tree, in Proc. of Congress on Evolutionary Computation (CEC 2002), IEEE Press, USA.
78. Neethling M and Engelbrecht AP (2006) Determining RNA Secondary Structure using Set-based Particle Swarm Optimization, in Proc. of Congress on Evolutionary Computation (CEC 2006), IEEE Press, USA.
79. Hofacker IL (2003) Vienna rna secondary structure server, Nucleic Acids Research, vol. 31:13, pp. 3429–3431.
80. Lau KF and Dill KA (1989) A lattice statistical mechanics model of the conformation and sequence space of proteins. Macromolecules 22, pp. 3986–3997.
81. Richards FM (1977) Areas, volumes, packing, and protein structures. Annu. Rev. Biophys. Bioeng. 6, pp. 151–176.
82. Krasnogor N, Hart WE, Smith J and Pelta DA (1999) Protein structure prediction with evolutionary algorithms. Proceedings of the Genetic & Evolutionary Computing Conf (GECCO 1999).

83. Shmygelska A, Hoos HH (2003) An Improved Ant Colony Optimization Algorithm for the 2D HP Protein Folding Problem. Canadian Conference on AI 2003: 400–417.
84. Shmygelska A, Hoos HH (2005) An ant colony optimization algorithm for the 2D and 3D hydrophobic polar protein folding problem. BMC Bioinformatics 6:30.
85. Chu D, Till M and Zomaya A (2005) Parallel Ant Colony Optimization for 3D Protein Structure Prediction using the HP Lattice Model, Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), IEEE Computer Society Press.
86. Meksangsouy P and Chaiyaratana N (2003) DNA fragment assembly using an ant colony system algorithm, in Proc. of Congress on Evolutionary Computation (CEC 2006), IEEE Press, USA.
87. Ando S and Iba H (2001) Inference of gene regulatory model by genetic algorithms, Proc. Congress on Evolutionary Computation (CEC 2001), vol. 1, pp. 712–719.
88. Behera N and Nanjundiah V (1997) Trans-gene regulation in adaptive evolution: a genetic algor-ithm model, Journal of Theoretical Biology, vol. 188, pp. 153–162.
89. Ando S and Iba H (2000) Quantitative Modeling of Gene Regulatory Network - Identifying the Network by Means of Genetic Algorithms, The Eleventh Genome Informatics Workshop, 2000.
90. Ando S and Iba H (2001) The Matrix Modeling of Gene Regulatory Networks - Reverse Engineering by Genetic Algorithms, Proc. Atlantic Symposium on Computational Biology and Genome Information Systems and Technology.
91. Tominaga D, Okamoto M, Maki Y, Watanabe S and Eguchi Y (1999) Nonlinear Numerical optimization technique based on a genetic algorithm for inverse Problems: Towards the inference of genetic networks, Computer Science and Biology (Proc. German Conf. on Bioinformatics), pp. 127–140.
92. Branden CI and Tooze J (1999) Introduction to Protein Structure: 2nd edition. Garland Publishing, New York, 2nd edition.
93. Liu Y and Beveridge DL (2002) Exploratory studies of ab initio protein structure prediction: multiple copy simulated annealing, amber energy functions, and a generalized born/solvent accessibility solvation model. Proteins, 46.
94. Unger R and Moult J (1993) A genetic algorithm for 3d protein folding simulations. In 5th Proc. Intl. Conf. on Genetic Algorithms, pp. 581–588.
95. Pokarowski P, Kolinski A and Skolnick J (2003) A minimal physically realistic protein-like lattice model: Designing an energy landscape that ensures all-or-none folding to a unique native state. Biophysics Journal, 84: pp. 1518–26.
96. Kitagawa U and Iba H (2002) Identifying Metabolic Pathways and Gene Regulation Networks with Evolutionary Algorithms, in Evolutionary Computation in Bioinformatics, Fogel GB and Corne DW (Eds.) Morgan Kaufmann.
97. Shayne CG (2005), Drug Discovery Handbook, Wiley-Interscience.
98. Madsen U. (2002), Textbook of Drug Design and Discovery, CRC Press, USA.
99. Venkatasubramanian V, Chan K and Caruthers JM (1995). Evolutionary Design of Molecules with Desired Properties Using the Genetic Algorithm, J. Chem. Inf. Comp. Sci., 35, pp. 188–195.
100. Glen RC and Payne AWR (1995) A Genetic Algorithm for the Automated Generation of Molecule within Constraints. J. Computer-Aided Molecular Design, 9, pp. 181–202.