Evolutionary Computation in Bioinformatics: A Review

Sankar K. Pal, *Fellow, IEEE*, Sanghamitra Bandyopadhyay, *Senior Member, IEEE*, and Shubhra Sankar Ray

***Abstract*—This paper provides an overview of the application of evolutionary algorithms in certain bioinformatics tasks. Different tasks such as gene sequence analysis, gene mapping, deoxyribonucleic acid (DNA) fragment assembly, gene finding, microarray analysis, gene regulatory network analysis, phylogenetic trees, structure prediction and analysis of DNA, ribonucleic acid and protein, and molecular docking with ligand design are, first of all, described along with their basic features. The relevance of using evolutionary algorithms to these problems is then mentioned. These are followed by different approaches, along with their merits, for addressing some of the aforesaid tasks. Finally, some limitations of the current research activity are provided. An extensive bibliography is included.**

***Index Terms*—Biocomputing, data mining, evolutionary algorithm, molecular biology, soft computing.**

I. Introduction

OVER the past few decades, major advances in the field of molecular biology, coupled with advances in genomic technologies, have led to an explosive growth in the biological information generated by the scientific community. This deluge of genomic information has, in turn, led to an absolute requirement for computerized databases to store, organize, and index the data, and for specialized tools to view and analyze the data.

Bioinformatics can be viewed as *the use of computational methods to make biological discoveries* [1]. It is an interdisciplinary field involving biology, computer science, mathematics, and statistics to analyze biological sequence data, genome content and arrangement, and to predict the function and structure of macromolecules. The ultimate goal of the field is to enable the discovery of new biological insights as well as to create a global perspective from which unifying principles in biology can be derived [2]. There are three important subdisciplines within bioinformatics.

1) Development of new algorithms and models to assess different relationships among the members of a large biological data set in a way that allows researchers to access existing information, and to submit new information as they are produced.
2) Analysis and interpretation of various types of data including nucleotide and amino acid sequences, protein domains; and protein structures.
3) Development and implementation of tools that enable efficient access and management of different types of information.

Recently, evolutionary algorithms (EAs), a class of randomized search and optimization techniques guided by the principles of evolution and natural genetics, have been gaining the attention of researchers for solving bioinformatics problems. Genetic algorithms (GAs) [3]–[9] evolutionary strategies (ES), and genetic programming (GP) are the major components of EAs. Of these, GAs are the most widely used. GAs are efficient, adaptive, and robust search processes, producing near optimal solutions, and have a large amount of implicit parallelism. Data analysis tools used earlier in bioinformatics were mainly based on statistical techniques such as regression and estimation. The role of GAs in bioinformatics gained significance with the need to handle large data sets in biology in a robust and computationally efficient manner.

This paper provides a survey of the various evolutionary-algorithm-based techniques that have been developed over the past few years for different bioinformatics tasks. First, we describe the basic concepts of bioinformatics along with their biological basis. Methodology for applying GAs to bioinformatics tasks is also mentioned in Section II. In Section III, various bioinformatics tasks and different evolutionary algorithms based methods available to address the bioinformatics tasks are explained. Finally, conclusions and some future research directions are presented in Section IV.

II. Basic Concepts in Bioinformatics and Relevance of Evolutionary Algorithms

First, we introduce the basic biological concepts required to understand the various problems in bioinformatics, and then we describe the relevance of EAs in bioinformatics with particular emphasis on their application of GAs.

A. Basic Units of Cell Biology and Bioinformatics Tasks

Deoxyribonucleic acid (DNA) and proteins are biological macromolecules built as long linear chains of chemical components. A DNA strand consists of a large sequence of nucleotides, or bases. For example there are more than three billion bases in human DNA sequences. DNA plays a fundamental role in different biochemical processes of living organisms in two respects. First, it contains the templates for the synthesis of proteins, which are essential molecules for any organism [10]. The second role in which DNA is essential to life is as a medium to transmit hereditary information (namely, the building plans for

Manuscript received May 7, 2004; revised January 7, 2005. This paper was supported by the Council of Scientific and Industrial Research, New Delhi, India, under the project "Knowledge Based Connectionist Data Mining System: Design and Application" under Grant 22(0346)/02/EMR-II. This paper was recommended by Associate Editor M. Last.

The authors are with the Machine Intelligence Unit, Indian Statistical Institute, Kolkata 700108, India (e-mail: sankar@isical.ac.in; sanghami@isical.ac.in; shubhra_r@isical.ac.in).

Digital Object Identifier 10.1109/TSMCC.2005.855515

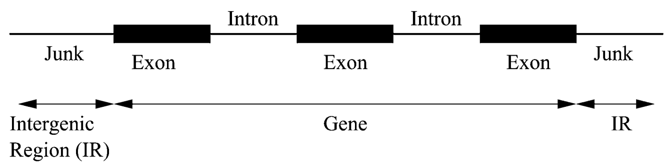

Fig. 1. Various parts of DNA.

proteins) from generation to generation. Proteins are responsible for structural behavior.

The units of DNA are called nucleotides. One nucleotide consists of one nitrogen base, one sugar molecule (deoxyribose), and one phosphate. Four nitrogen bases are denoted by one of the letters A (adenine), C (cytosine), G (guanine), and T (thymine). A linear chain of DNA is paired to a complementary strand. The complementary property stems from the ability of the nucleotides to establish specific pairs (A-T and G-C). The pair of complementary strands then forms the double helix that was first suggested by Watson and Crick in 1953. Each strand, therefore, carries all the information, and the biochemical machinery guarantees that the information can be copied over and over again, even when the "original" molecule has long since vanished.

A gene is primarily made up of a sequence of triplets of the nucleotides (exons). Introns (noncoding sequence) may also be present within the gene. Not all portions of the DNA sequences are coding. A coding zone indicates that it is a template for a protein. As an example, for the human genome, only 3%–5% of the sequence are coding; i.e., they constitute the gene. The promoter is a region before each gene in the DNA that serves as an indication to the cellular mechanism that a gene is ahead. For example, the codon AUG is a protein which codes for methionine and signals the start of a gene. Promoters are key regulatory sequences that are necessary for the initiation of transcription. Transcription is process in which ribonucleic acid (RNA) is formed from a gene, and through translation, aminoacids are formed from RNA. There are sequences of nucleotides within the DNA that are spliced out progressively in the process of transcription and translation. A comprehensive survey of the research done in this field is given in [11]. In brief, the DNA consists of three types of noncoding sequences (see Fig. 1) as follows:

1) Intergenic regions: Regions between genes that are ignored during the process of transcription.
2) Intragenic regions (or Introns): Regions within the genes that are spliced out from the transcribed RNA to yield the building blocks of the genes, referred to as Exons.
3) Pseudogenes: Genes that are transcribed into the RNA and stay there, without being translated, due to the action of a nucleotide sequence.

Proteins are polypeptides, formed within cells as a linear chain of amino acids [10]. Amino acid molecules bond with each other by eleminating water molecules and forming peptides. 20 different amino acids (or "residues") are available, which are denoted by 20 different letters of the alphabet. Each of the 20 amino acids is coded by one or more triplets (or codons) of the nucleotides making up the DNA. Based on the genetic code, the linear string of DNA is translated into a linear string of

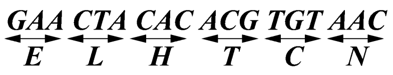

Fig. 2. Coding of amino acid sequence from DNA sequence.

amino acids; i.e., a protein via mRNA (messenger RNA) [10]. For example, the DNA sequence GAACTACACACGTGTAAC codes for the amino acid sequence ELHTCN (shown in Fig. 2).

Three-dimensional (3-D) molecular structure is one of the foundations of structure-based drug design. Often, data are available for the shape of a protein and a drug separately, but not for the two together. Docking is the process by which two molecules fit together in 3-D space. Ligands are small molecules such as a candidate drug and are used for docking to their macromolecular targets (usually proteins, sometimes DNA).

Different biological problems considered within the scope of bioinformatics involve the study of genes, proteins, nucleic acid structure prediction, and molecular design with docking. A broad classification of the various bioinformatics tasks is given as follows.

1) alignment and comparison of DNA, RNA, and protein sequences;
2) gene mapping on chromosomes;
3) gene finding and promoter identification from DNA sequences;
4) interpretation of gene expression and microarray data;
5) gene regulatory network identification;
6) construction of phylogenetic trees for studying evolutionary relationship;
7) DNA structure prediction;
8) RNA structure prediction;
9) protein structure prediction and classification;
10) molecular design and molecular docking.

Descriptions of these tasks and their implementation in evolutionary computing (or genetic algorithmic) framework are provided in Section III. Before that, the relevance of GAs in bioinformatics is explained.

B. Relevance of Genetic Algorithms in Bioinformatics

Genetic algorithms [3]–[6], a biologically inspired technology, are randomized search and optimization techniques guided by the principles of evolution and natural genetics. They are efficient adaptive, and robust search processes, producing near optimal solutions, and have a large degree of implicit parallelism. Therefore, the application of GAs for solving certain problems of bioinformatics, which need optimization of computation requirements, and robust, fast and close approximate solutions, appears to be appropriate and natural [4]. Moreover, the errors generated in experiments with bioinformatics data can be handled with the robust characteristics of GAs. To some extent, such errors may be regarded as contributing to genetic diversity, a desirable property. The problem of integrating GAs and bioinformatics constitutes a new research area.

GAs are executed iteratively on a set of coded solutions, called population, with three basic operators: selection/reproduction, crossover, and mutation. They use only the payoff (objective function) information and probabilistic transition rules for moving to the next iteration. They are different from most of the normal optimization and search procedures in four ways:

1) GAs work with the coding of the parameter set, not with the parameters themselves.
2) GAs work simultaneously with multiple points, and not a single point.
3) GAs search via sampling (a blind search) using only the payoff information.
4) GAs search using stochastic operators, not deterministic rules.

A GA typically consists of the following components:

1) a population of binary strings or coded possible solutions (biol)ogically referred to as chromosomes);
2) a mechanism to encode a possible solution (mostly as a binary string);
3) objective function and associated fitness evaluation techniques;
4) selection/reproduction procedure;
5) genetic operators (crossover and mutation);
6) probabilities to perform genetic operations.

Of all the evolutionarily inspired approaches, GAs seem particularly suited to implementation using DNA, protein, and other bioinformatics tasks [12]. This is because GAs are generally based on manipulating populations of bitstrings using both crossover and pointwise mutation.

The main advantages using GAs are as follows.

1) Several tasks in bioinformatics involve optimization of different criteria (such as energy, alignment score, and overlap strength), thereby making the application of GAs more natural and appropriate.
2) Problems of bioinformatics seldom need the exact optimum solution; rather, they require robust, fast, and close approximate solutions, which GAs are known to provide efficiently.
3) GAs can process, in parallel, populations billions times larger than is usual for conventional computation. The usual expectation is that larger populations can sustain larger ranges of genetic variation, and thus can generate high-fitness individuals in fewer generations.
4) Laboratory operations on DNA inherently involve errors. These are more tolerable in executing evolutionary algorithms than in executing deterministic algorithms. (To some extent, errors may be regarded as contributing to genetic diversity—a desirable property.)

C. Example

Let us now discuss with an example the relevance of GAs in bioinformatics. Most of the ordering problems in bioinformatics, such as sequence alignment problem, fragment assembly problem (FAP), and gene maping (GM), are quite similar to traveling salesman problem (TSP best-known NP-hard ordering problem) with notable differences. The TSP can be formally defined as follows: Let $1, 2, \ldots, n$ be the labels of the n cities and $C = [c_{i,j}]$ be an $n \times n$ cost matrix where $c_{i,j}$ denotes the cost of traveling from city i to city j. The TSP is the problem of finding the shortest closed route among n cities, having as input the complete distance matrix among all cities. A symmetric TSP (STSP) instance is any instance of the TSP such that $c_{i,j} = c_{j,i}$ for all cities i, j. An asymmetric TSP (ATSP) instance is any instance of the TSP that has at least one pair of cities such that $c_{i,j} \neq c_{j,i}$. The ATSP is a special case of the problem on which we restrict the input to asymmetric instances. The total cost A of a TSP tour is given by

$$A(n) = \sum_{i=1}^{n-1} c_{i,i+1} + c_{n,1}. \quad (1)$$

The objective is to find a permutation of the n cities which has minimum cost.

```
  ACTGGC
    TGGCTTACT
TCACTC
        TTACTAAG
----------------
TCACTGGCTTACTAAG  ⟶ CONSENSUS
                    SEQUENCE
```

Fig. 3. Alignment of DNA fragments.

The FAP deals with the sequencing of DNA. Currently, strands of DNA longer than approximately 500 base pairs cannot routinely be sequenced accurately. Consequently, for sequencing larger strands of DNA, they are first broken into smaller pieces. In the shotgun sequencing method (to which this work applies), DNA is first replicated many times, and then individual strands of the double helix are broken randomly into smaller fragments. The assembly of DNA fragments into a consensus sequence corresponding to the parent sequence constitutes the "fragment assembly problem" [10]. It is a permutation problem, similar to the TSP, but with some important differences (circular tours, noise, and special relationships between entities) [10]. It is NP-complete in nature.

Note that the fragmenting process does not retain either the ordering of the fragments on the parent strand of DNA or the strand of the double helix from which a particular fragment came. The only information available in assembly stage is the base pair sequence for each fragment. Thus, the ordering of the fragments must rely primarily on the similarity of fragments and how they overlap. An important aspect of the general sequencing problem is the precise determination of the relationship and orientation of the fragment. Once the fragments have been ordered, the final consensus sequence is generated from the ordering. Basic steps with four fragments are shown below as an example in Fig. 3. Here, the fragments are aligned in a fashion so that in each column all the bases are the same. As an example, the base in the sixth column is selected, after voting, as G to make the consensus sequence TCACTGGCTTACTAAG.

Formulation of the FAP as a TSP using GA: although the endpoints of the tour of TSP are irrelevant since its solution is a circular tour of the cities, in the case of FAP, the endpoints are relevant as they represent fragments on opposite ends of the parent sequence. Moreover, the cities in the TSP are not assumed to have any relationship other than the distances, and the ordering is the final solution to the problem. In FAP, the ordering referred to as "beads on a string," is only an intermediate step; the layout process uses the overlap data to position the bases within the fragments relative to each other. Here, GAs can be applied. A way of using it in FAP is explained as follows.

Step 1) Let $1, 2, \ldots, j, \ldots, n$ represent the indices of n fragments in the spectrum of fragments. Pairwise relationship (similarity) of a fragment with all other fragments (oligonucleotides) is calculated and kept in an $n \times n$ matrix. Dynamic programming gives best alignment between two sequences (fragments). In this method, each possible orientation is tried for the two fragments, and the overlap, orientation, and alignment are chosen to maximize the similarity between fragments.

Step 2) All the indices of fragments are then ordered randomly with no repetition. Let $f_1, f_2, \ldots, f_i, \ldots, f_n$ be such an ordering of a sequence of n fragments, where $f_i = j$ means that fragment j (in the fragment set) appears in position i of the ordering. The fitness function of this ordered sequence can be computed using

$$F = \sum_{i=1}^{n-1} W_{f_i, f_{i+1}} \tag{2}$$

where $W_{i,j}$ is the pairwise overlap strength (similarity) of fragments i and j in the ordered sequence, as obtained in the $n \times n$ matrix.

Such an ordered sequence provides a genetic representation of an individual chromosome in GA.

Step 3) In this way, P ordered sequences are generated, where P is the size of the population of GA.

Step 4) GA is applied with this population and the following operations.

Selection: Fitness of each sequence is evaluated as in (2), and sequences with higher fitness are selected with roulette wheel.

Crossover: Crossover is performed between two randomly selected sequences for a given crossover rate.

Mutation: For a given mutation rate, only that mutation operator can be applied for which there will be no repetition of fragment indexes in the sequence.

Elitist model: A new population is created at each generation of GA. The sequence with highest fitness from the previous generation replaces randomly a sequence from this new generation, provided the fitness of the fittest sequence in the previous generation is higher than the best fitness in this current generation.

Step 5) The best sequence of indices with maximum F value is obtained from the GA. From this sequence of indices, the corresponding sequence of fragments is obtained using the overlapping information in the $n \times n$ matrix of Step 1).

Step 6) This alignment of fragments is examined to determine the places where insertion or deletion error likely occurred, and gaps or bases are then inserted or deleted into the fragments to obtain their best possible alignment. The resulting sequence is called consensus sequence.

Note: The neighboring fragments in the resulting sequence are assumed to be maximally overlapped—thereby ensuring inclusion in the resulting sequence as many fragments as possible. The fitness function GA evaluating an individual selects the best substring of oligonucleotides, or the chromosome; i.e., the one composed of the most fragments, provided its length is equal to the given length of the reference DNA sequence.

Different GA operators for the assembly of DNA sequence fragments associated with the Human Genome project was studied in [13]. The sorted order representation and the permutation representation are compared on problems ranging from 2–34 K base pairs (KB). It is found that edge-recombination crossover used in conjunction with several specialized operators performs the best. Other relevant investigations for solving FAP using GAs are available in [14] and [15].

III. Bioinformatics Tasks and Application of EAs

We now describe the different problems and associated tasks involved in bioinformatics, their requirements, and the ways in which computational models can be formulated to solve them. The classified tasks (as mentioned in Section II-A) are first explained in this section, followed by a description of how GAs and other evolutionary techniques are applied in solving them.

A. Alignment and Comparison of DNA, RNA, and Protein Sequences

An alignment is a mutual placement of two or more sequences which exhibit where the sequences are similar, and where they differ. These include alignment and prediction of DNA, RNA, protein sequences, and fragment assembly of DNA. An optimal alignment is the one that exhibits the most correspondences and the fewest differences. It is the alignment with the highest score, but which may or may not be biologically meaningful. Basically, there are two types of alignment methods: global alignment and local alignment. Global alignment [16] maximizes the number of matches between the sequences along the entire length of the sequence. Local alignment [17] gives a highest scoring to local match between two sequences. Global alignment includes all the characters in both sequences from one end to the other, and is excellent for sequences that are known to be very similar. If the sequences being compared are not similar over their entire lengths, but have short stretches within them that have high levels of similarity, a global alignment may miss the alignment of these important regions, and local alignment is then used to find these internal regions of high similarity. Pairwise comparison and alignment of protein or nucleic acid sequences is the foundation upon which most other bioinformatics tools are built. Dynamic programming (DP) is an algorithm that allows for efficient and complete comparison of two (or more) biological sequences, and the technique is known as the Smith–Waterman algorithm [17]. It refers to a programmatic technique or algorithm which, when implemented correctly, effectively makes all possible pairwise comparisons between the characters (nucleotide or amino acid residues) in two biological sequences. Spaces may need to be inserted within the sequences for alignment. Consecutive spaces are defined as a gap. The final result is a mathematically, but not necessarily biologically, optimal alignment of the two sequences. A similarity score is also generated to describe how similar the two sequences are, given the specific parameters used.

A multiple alignment arranges a set of sequences in a manner that positions thought to be homologous are placed in a common column. There are different conventions regarding the scoring of a multiple alignment. In one approach, the scores of all the induced pairwise alignments contained in a multiple alignment are simply added. For a linear gap penalty, this amounts to scoring each column of the alignment by the sum of pair (SP-) scores in this column [10]. Although it would be biologically meaningful, the distinctions between global, local, and other forms of alignment are rarely made in a multiple alignment. A full set of optimal pairwise alignments among a given set of sequences will generally overdetermine the multiple alignment. If one wishes to assemble a multiple alignment from pairwise alignments, one has to avoid "closing loops," i.e., one can put together pairwise alignments as long as no new pairwise alignment is included to a set of sequences which is already part of the multiple alignment.

Methods: GAs are used to solve the problem of multiple sequence alignment. Before we describe them, it may be mentioned that other optimization methods, such as simulated annealing [18] and Gibbs sampling [19], are also used in this regard. Simulated annealing can sometimes be very slow, although it works well as an alignment improver. Gibbs sampling is good in finding local multiple alignment block with no gaps, but is not suitable in gapped situations.

It was first described in Sequence Alignment by Genetic Algorithm (SAGA) [20] how to use GA to deal with sequence alignments in a general manner (without DP), shortly before a similar work by Zhang *et al.* [21]. The population is made of alignments, and the mutations are processing programs that shuffle the gaps using complex methods. In SAGA, each individual (chromosome) is a multiple alignment of sequences. The population size is 100 and there is no identical individual in it. To create one of these alignments, a random offset is chosen for all the sequences (the typical range is from 0–50 for sequences 200 residues long) and each sequence is moved to the right, according to its offset. The sequences are then padded with null signs in order to have the same length. The fitness of each individual (alignment) is computed as the score of the corresponding alignment. All the individuals are ranked according to their fitness, and the weakest are replaced by new children. Only a portion (e.g., 50%) of the population are replaced during each generation. Two types of crossover, two types of gap insertion mutation, 16 types of block shuffling mutation, one block searching mutation, and two local optimal rearrangemet mutation operators are used in SAGA. During initialization of the program, all the operators have the same probability of being used, equal to $1/22$. An automatic procedure (dynamic schedules, proposed by Davis [22]) for selecting operator has been implemented in SAGA. In this model, an operator has a probability of being used that is a function of the efficiency it has recently (e.g., ten last generations) displayed at improving alignments. The credit an operator receives when performing an improvement is also shared with the operators that came before, and may have played a role in this improvement. Thus, each time a new individual is generated, if it yields some improvement on its parents, the operator that is directly responsible for its creation gets the largest part of the credit (e.g., 50%). Then the operator(s) responsible for the creation of the parents also get their share of the remaining credit (50% of the remaining credit; i.e., 25% of the original credit), and so on. This report of the credit goes on for some specified number of generations (e.g., 4). After a given number of generations (e.g., 10) these results are summarized for each of the operators. The credit of an operator is equal to its total credit divided by the number of children it generated. This value is taken as usage probability and will remain unchanged until the next assessment, ten generations later. To avoid the early loss of some operators that may become useful later on, all the operators are assigned a minimum probability of being used (the same for all them, typically equal to half their original probability, i.e., $1/44$). The automatically assigned probabilities of usage at different stages in the alignment give a direct measure of usefulness or redundancy for a new operator. SAGA is stopped when the search has been unable to improve for some specified number of generations (typically 100). This condition is the most widely used when working on a population with no duplicates.

Other approaches [23]–[25] are similar to SAGA where, a population of multiple alignment evolves by selection, combination, and mutation. The main difference between SAGA and recent algorithms has been the design of better mutation operators. A simple GA, applied in a straightforward fashion to the alignment problem, was not very successful [20]. The main devices which allow GAs to efficiently reach very high quality solutions are the use of: 1) a large number of mutation and crossover operators, and 2) their automatic scheduling. The GA based methods are not very efficient at handling all types of situations. So it is necessary to invent some new operators designed specifically for the problem, and to slot them into the existing scheme. Most of the investigations using GAs for sequence alignment are on different data sets and results are compared with that of CLUSTAL W [26], so a clear comparison between the GA based methods is not possible. A hybrid approach [27], [28], uses the searching ability of GAs for finding match blocks, and dynamic programming for producing close to optimum alignment of the match blocks. This method is faster and produces better results than pure GA and DP based approaches. Here, the population size is determined as $Q = mn/100$, where m is the average sequence length and n is the number of sequences.

In [29], it was pointed out that the combination of high-performance crossover and mutation operators does not always lead to a high performance GA for sequencing because of the negative combination effect of those two operators. A high-performance GA can be constructed by utilizing the positive combination effect of crossover and mutation.

Other relevant investigations for solving multiple sequence alignment using GAs are available in [30]–[34].

B. Gene Mapping on Chromosomes

Gene mapping is defined as the determination of relative positions of genes on a chromosome, and the distance between them. A gene map helps molecular biologists to explore a genome. A primary goal of the Human Genome Project is to make a series of descriptive diagram maps of each human chromosome at increasingly finer resolutions. Two types of gene maps, *viz.*, cytogenetic map and linkage map are generally used. A cytogenetic map, also known as a physical map, offers a physical picture of the chromosome. In a cytogenetic map, the chromosomes are divided into smaller fragments that can be propagated and characterized, and then the fragments are ordered (mapped) to correspond to their respective locations on the chromosomes. A genetic linkage map shows the relative locations (order) of specific DNA markers along the chromosome.

Since EAs have been used for determining the genetic linkage map, it is described here briefly. The genetic markers in a linkage map are generally small, but precisely defined sequences and can be expressed as DNA regions (genes) or DNA segments that have no known coding function but whose inheritance pattern can be followed. DNA sequence differences are especially useful markers because they are plentiful and easy to characterize precisely [10]. A linkage map is constructed by the following:

1) producing successive generations (chromosomes) of certain organisms through crossover (recombination), and
2) analyzing the observed segregation percentages of certain characteristics in each chromosomal data to find the actual gene order.

A linkage map shows the order and relative distance between genes, but has two drawbacks [10]. First, it does not tell the actual distance of genes, and second, if genes are very close, one can not resolve their order, because the probability of separation is so small that the observed recombinant frequencies are all zero. The closer two genes are, the lower the probability that they will be separated during the DNA repair or replication process, and hence the probability is greater that they will be inherited together. For example, suppose a certain stretch of DNA has been completely sequenced, giving us a sequence S. If we know which chromosome S came from, and if we have a physical map of this chromosome, we could try to find one of the map's markers in S. If the process succeeds, we can locate the position of S in the chromosome. The best criterion to quantify how well a map explains the data set is the multipoint maximum likelihood (exploiting the data on all markers simultaneously) of the map. Given a probabilistic model of recombination for a given family structure, a genetic map of a linkage group, and the set of available observations on markers of the linkage group, we can define the probability that the observations may have occurred given the map. This is termed the likelihood of the map. The likelihood is only meaningful when compared to the likelihood of other maps.

The problem of finding a maximum likelihood genetic map can be described as a double optimization problem. For a given gene order, there is the problem of finding recombination probabilities (crossover probabilities) that yield a maximum multipoint likelihood; then, one must find an order that maximizes this maximum likelihood. The first problem is solved by using the expectation maximization (EM) algorithm. The second problem is more difficult, because the number of possible orders to consider for N markers is $N!/2$. This type of combinatorial problem can be handled efficiently by evolutionary algorithms. The problem of finding an order of genes that maximizes the maximum multipoint likelihood is equivalent to the symmetric TSP. One can simply associate one imaginary city to each marker, and define as the distance between two cities the inverse of the elementary contribution to the log-likelihood defined by the corresponding pair of markers.

Methods: The method of genetic mapping described in [35] is embodied in a hybrid framework that relies on the statistical optimization algorithms (e.g., expectation maximization) to handle the continuous variables (recombination probabilities), while GAs handle the ordering problem of genes. The efficiency of the approach lies critically in the introduction of greedy local search in the fitness evaluation of the GA, using a neighborhood structure inspired by the TSP. A population size ranging from 25–250 has been used for number of markers between 10–29.

In gene mapping problem, Gunnels *et al.* [36] compared GAs with simulated annealing (SA), and found that the GA-based method always converges to a good solution faster since its population-based nature allows it to take advantage of the extra information to construct good local maps that can then be used to construct good global maps.

In canonical GAs with the fixed map it is difficult to design the map without *a priori* knowledge of the solution space. This is overcome in [37], where GAs using a coevolutionary approach are utilized for exploring not only within a part of the solution space defined by the genotype-phenotype map, but also with the map itself. Here, the genotype-phenotype map is improved adaptively during the searching process for solution candidates. The algorithm is applied to three-bit deceptive problems as a kind of typical combinatorial optimization problem. The difficulty with canonical GAs can be controlled by the genotype-phenotype map, and the output shows fairly good performance.

Relevant investigation for gene mapping using GAs is also available in [38].

C. Gene Finding and Promoter Identification From DNA Sequences

Automatic identification of the genes from the large DNA sequences is an important problem in bioinformatics [39]. A cell mechanism recognizes the beginning of a gene or gene cluster with the help of a promoter and is necessary for the initiation of transcription. The promoter is a region before each gene in the DNA that serves as an indication to the cellular mechanism that a gene is ahead. For example, the codon AUG (which codes for methionine) also signals the start of a gene. Recognition of regulatory sites in DNA fragments has become particularly popular because of the increasing number of completely sequenced genomes and mass application of DNA chips. Experimental analyses have identified fewer than 10% of the potential promoter regions, assuming that there are at least 30 000 promoters in the human genome, one for each gene.

Methods: Using GA, Kel *et al.* [40] designed sets of appropriate oligonucleotide probes capable of identifying new genes belonging to a defined gene family within a cDNA or genomic library. One of the major advantages of this approach is the low homology requirement to identify functional families of sequences with little homology.

Levitsky *et al.* [41] described a method for recognizing promoter regions of eukaryotic genes with an application on Drosophila melanogaster. Its novelty lies in realizing the GA to search for an optimal partition of a promoter region into local nonoverlapping fragments, and selection of the most significant dinucleotide frequencies for the fragments.

The method of prediction of eukaryotic Pol II promoters from DNA sequence [42] takes advantage of a combination of elements similar to neural networks and GAs to recognize a set of discrete subpatterns with variable separation as one pattern: a promoter. The neural networks use, as input, a small window of DNA sequence, as well as the output of other neural networks. Through the use of GAs, the weights in the neural networks are optimized to discriminate maximally between promoters and nonpromoters.

D. *Interpretation of Gene Expression and Microarray Data*

Gene expression is the process by which a gene's coded information is converted into the structures present and operating in the cell. Expressed genes include those that are transcribed into mRNA and then translated into protein, and those that are transcribed into RNA but not translated into protein (e.g., transfer and ribosomal RNAs). Not all genes are expressed, and gene expression involves the study of the expression level of genes in the cells under different conditions. Conventional wisdom is that gene products which interact with each other are more likely to have similar expression profiles than if they do not [43].

Microarray technology [44] allows expression levels of thousands of genes to be measured at the same time. A microarray is typically a glass (or some other material) slide, on to which DNA molecules are attached at fixed locations (spots). There may be tens of thousands of spots on an array, each containing a huge number of identical DNA molecules (or fragments of identical molecules), of lengths from twenty to hundreds of nucleotides. Each of these molecules ideally should identify one gene or one exon in the genome. The spots are either printed on the microarrays by a robot, or synthesized by photolithography (as in computer chip productions), or by ink-jet printing.

Many unanswered and important questions could potentially be answered by correctly selecting, assembling, analyzing, and interpreting microarray data. Clustering is commonly used in microarray experiments to identify groups of genes that share similar expressions. Genes that are similarly expressed are often coregulated and are involved in the same cellular processes. Therefore, clustering suggests functional relationships between groups of genes. It may also help in identifying promoter sequence elements that are shared among genes. In addition, clustering can be used to analyze the effects of specific changes in experimental conditions, and may reveal the full cellular responses triggered by those conditions.

A good solution of the gene ordering problem (i.e., finding optimal order of DNA microarray data) will have similar genes grouped together, in clusters. A notion of distance must thus be defined in order to measure similarity among genes. A simple measure is the Euclidean distance (other options are possible using Pearson correlation, absolute correlation, Spearman rank correlation, etc.). One can thus construct a matrix of intergene distances. Using this matrix one can calculate the total distance between adjacent genes and find that permutation of genes for which the total distance is minimized [similar to what is done in the TSP using GA (Section II-B)].

Methods: Finding the optimal order of microarray data is known to be NP complete. Tsai *et al.* [45] formulated this as the traveling salesman problem and the applied family competition GA (FCGA), to solve it. The edge assembly crossover (EAX) is combined with the family competition concept and neighbor join mutation (NJ). In [46], a modified EAX and NJ are used in EA for efficiently optimizing the clustering and ordering of genes, ranging in size from 147 to 6221. Chromosomes in EAs are represented as a permutation of genes. The size of the population is assumed to equal to the number of genes in problems that involved fewer than 1000 genes, and half of the number of gens in larger problems. Fitness of chromosomes are evaluated from (1) and distance matrix is formed using pearson correlation. Crossover and mutation rates are set to one. Microarray data analysis is a competitive field, and no decisive measure of the performance of methods is available, so methods using EAs for microarray are compared in the TSP framework [46].

Garibay *et al.* [47] introduced a proportional GA (PGA) that relies on the existence or nonexistence of genes to determine the information that is expressed. The information represented by a PGA individual depends only on what is present in the individual, and not on the order in which it is present. As a result, the order of the encoded information is free to evolve in response to factors other than the value of the solution.

E. *Gene Regulatory Network Identification*

Inferring a gene regulatory network from gene expression data obtained by DNA microarray is considered one of the most challenging problems in the field of bioinfomatics [48]. An important and interesting question in biology, regarding the variation of gene expression levels, is how genes are regulated. Since almost all cells in a particular organism have an identical genome, differences in gene expression, and not the genome content, are responsible for cell differentiation during the life of the organism.

For gene regulation, an important role is played by a type of proteins called transcription factors [10]. The transcription factors bind to specific parts of the DNA, called transcription factor binding sites (i.e., specific, relatively short combinations of A, T, C or G), which are located in promoter regions. Specific promoters are associated with particular genes and are generally not too far from the respective genes, although some regulatory effects can be located as far as 30 000 bases away, which makes the definition of the promoter difficult.

Transcription factors control gene expression by binding to the gene's promoter and either activating (switching on) the gene or repressing it (switching it off). Transcription factors are gene products themselves, and therefore, in turn, can be controlled by other transcription factors. Transcription factors can control many genes, and some (probably most) genes are controlled by combinations of transcription factors. Feedback loops are possible. Therefore, we can talk about gene regulation networks. Microarrays and computational methods are playing a major role in attempts to reverse engineer gene networks from various observations.

Methods: In gene network inference problem the objective is to predict a regulating network structure of the interacting genes from the observed data; i.e., expression pattern. The gene expressions are regulated in discrete state transitions such that the expression levels of all genes are updated simultaneously. In [49], each real valued chromosomes (in GAs) represents the expression level of all the genes. Each gene has a specific expression level for another gene; so, for N genes there are N^2 expression levels. Fitness of the chromosomes are evaluated by absolute error with generated expression pattern (The sum of all expressions) from the target expression pattern. A population size of 2500, 5000, and 7000 are taken for 5, 7, and 10 genes, respectively. The GA run for 150 generations with a crossover and mutation rate of 0.99 and 0.01, respectively. Relevant investigations using GAs are also available in [50]–[53].

F. Construction of Phylogenetic Trees for Studying Evolutionary Relationship

All species on earth undergo a slow transformation process called evolution. To explain the evolutionary history of today's species and how species relate to one another in terms of common ancestors, trees are constructed whose leaves represent the present day species, and interior nodes which represent the hypothesized ancestors. These kind of labeled binary trees are called phylogenetic trees [10]. Phylogenetic analysis is used to study the evolutionary relationship.

Phylogenies are reconstructed based on comparisons between present-day objects. The term object is used to denote the units for which one wants to reconstruct the phylogeny. Input data required for constructing phylogeny are classified into two main categories [10]. 1) Discrete character, such as beak shape, number of fingers of 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. 2) Comparative numerical data, called distances between objects. The resulting matrix is called a distance matrix.

Given data (character state matrix or distance matrix) for n taxa (objects), the phylogenetic tree reconstruction problem is to find the particular permutation of taxa that optimize the criteria (distance). The problem is equivalent to the problem of TSP. 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.

Methods: Exhaustive search of the space of phylogenetic trees is generally not possible for more than 11 taxa, and so algorithms for efficiently searching the space of trees must be developed. Phylogeny reconstruction is a difficult computational problem, because the number of possible solutions (permutations) increases with the number of included taxa (objects) [54]. Branch-and-bound methods can reasonably be applied for up to about 20 taxa, so scientists generally rely on heuristic algorithms, such as stepwise-addition and star-decomposition methods. However, such algorithms generally involve a prohibitive amount of computation time for large problems and often find trees that are only locally optimal. Heuristic search strategies using GAs [54]–[57] can overcome the aforementioned problems by faster reconstruction of the optimal trees with less computing power.

In [57], each chromosome in GA is encoded as a permutation of 15 taxas (the same as TSP); and selection, crossover, and mutation operations are performed to minimize the distance among the taxas. Here, each taxa is an amino acid sequence taken from the GenBank, and distance between them is computed as an alignment score using CLUSTAL W [26]. The GA population consisted of 20 trial trees. A crossover probability of 0.5 and mutation probability of 0.2 has been used. Optimal trees are obtained after 138 generations. The only difference with TSP is that the end points of the chromosome GA are relevant in phylogenetic trees as they represent the starting and the end points of evolutionary relationship. GAs has also been used [58] for automatic self-adjustment of the parameters of the optimization algorithm of phylogenetic trees.

G. DNA Structure Prediction

DNA structure plays an important role in a variety of biological processes. Different dinucleotide and trinucleotide scales have been described to capture various aspects of DNA structure including base stacking energy, propeller twist angle, protein deformability, bendability, and position preference [59]. three-dimension DNA structure and its organization into chromatin fibres is essential for its functions, and is applied in protein binding sites, gene regulation, triplet repeat expansion diseases, etc. DNA structure depends on the exact sequence of nucleotides and largely on interactions between neighboring base pairs. Different sequences can have different intrinsic structures. Periodic repetitions of bent DNA in phase with the helical pitch will cause DNA to assume a macroscopically curved structure. Flexible or intrinsically curved DNA is energetically more favorable to wrap around histones than rigid and unbent DNA.

The curvature of a space line is defined as the derivative, dt/dl, of the tangent vector t, along the line l. Its modulus is the inverse of the curvature radius, and its direction is that of the main normal to the curve [61]. In the case of DNA, the line corresponds to the helical axis and the curvature is a vectorial function of the sequence. The curvature represents the angular deviation ($|C(n)|$) between the local helical axes of the nth and $(n+1)$th base pairs (Fig. 4). Under similar external conditions, the intrinsic curvature function represents the differential behavior of different DNA tracts and corresponds to the most stable superstructure. The physical origin of curvature is still a matter of debate [60]; it is, however, a result of the chemical and, consequently, stereochemical, inhomogeneity of the sequence, which gives rise to different macroscopic manifestations. These manifestations change with the thermodynamic conditions such as pH, the ionic force, the kind of counterions, and obviously the temperature as a result of perturbations on the intrinsic curvature depending on the sequence-dependent bendability. Therefore, it is generally useful to characterize a DNA superstructure with the so-called intrinsic curvature function [60].

Methods: The 3-D spatial structure of a methylene-acetal-linked thymine dimer present in a 10 basepair (bp) sense–antisense DNA duplex was studied in [62] with a GA designed to interpret nuclear Overhauser effect (NOE) inter-proton distance restraints. Trial solutions (chromosomes in GAs) are encoded on bit strings which represents torsion angles between atoms. From these torsion angles, atomic coordinates, needed for the fitness function are calculated using the DENISE program. The problem is to find a permutation of torsion angles (eight torsion angles for each nucleotide in DNA) that minimizes the atomic distance between protons of neucleotides. The GA minimizes the difference between distances in the trial structures and distance restraints for a set of 63 proton–proton distance restraints defining the methylene-acetal-linked thymine dimer. The torsion angles were encoded by Gray coding and the GA population consisted of 100 trial structures. Uniform crossover with a probability of 0.9 and mutation rate of 0.04 was used. It was demonstrated that the bond angle geometry around the methylene-acetal linkage plays an important role in the optimization.

A hybrid technique involving artificial neural networks (ANN) and GA is described in [63] for optimization of DNA curvature characterized in terms of the reliability (RL) value. In this approach, first an ANN approximates (models) the nonlinear relationship(s) existing between its input and output example data sets. Next, the GA searches the input space of the ANN with a view to optimize the ANN output. Using this methodology, a number of sequences possessing high RL values have been obtained and analyzed to verify the existence of features known to be responsible for the occurrence of curvature.

H. RNA Structure Prediction

An RNA molecule is considered as a string of n characters $R = r_1 r_2 \cdots r_n$ such that $r_i \varepsilon A, C, G, U$. Typically n is in the hundreds, but could also be in thousands. The secondary structure of the molecule is a collection S of a set of stems and each stem consisting of a set of consecutive base pairs $(r_i r_j)$ (e.g., GU, GC, AU). Here, $1 \leq i \leq j \leq n$ and (r_i and r_j) are connected through hydrogen bonds. If $(r_i, r_j) \varepsilon S$, in principle we should require that r_i be a complement to r_j and that $j - i > t$, for a certain threshold t (because it is known that an RNA molecule does not fold too sharply on itself). With such an assumption [10], the total free energy E of a structure S is given by

$$E(s) = \sum_{(r_i, r_j) \in S} \alpha(r_i, r_j) \tag{3}$$

where $\alpha(r_i, r_j)$ gives the free energy of base pair (r_i, r_j). Generally, the adopted convention is $\alpha(r_i, r_j) < 0$, if $i \neq j$, and $\alpha(r_i, r_j) = 0$, if $i = j$.

Attempts to predict automatically the RNA secondary structure can be divided in essentially two general approaches. The first involves the overall free energy minimization by adding contributions from each base pair, bulged base, loop, and other elements [64]. EAs are found to be suitable for this purpose. Chromosomes in EAs are encoded to represent the RNA structure and fitness of each chromosome is evaluated in terms of free energy (3). The second type of approach [65] is more empirical and it involves searching for the combination of nonexclusive helices with a maximum number of base pairings, satisfying the condition of a tree like structure for the bio-molecule. Within the latter, methods using dynamic programming (DP) are the most common [65], [66]. While DP can accurately compute the minimum energy within a given thermodynamic model, the natural fold of RNA is often in a suboptimal energy state and requires soft computing EAs rather than hard computing DP.

RNA may enter intermediate conformational states that are key to its functionality. These states may have a significant impact on gene expression. The biologically functional states of RNA molecules may not correspond to their minimum energy state, and kinetic barriers may exist that trap the molecule in a local minimum. In addition, folding often occurs during transcription, and cases exist in which a molecule will undergo transitions between one or more functional conformations before reaching its native state. Thus, methods for simulating the folding pathway of an RNA molecule and locating significant intermediate states are important for the prediction of RNA structure and its associated function.

Methods: The possibilities of using GAs for the prediction of RNA secondary structure were investigated in [67] and [68]. The implementations used a binary representation for the solutions (chromosomes in GAs). The algorithm, using the procedure of stepwise selection of the most fit structures (similarly to natural evolution), allows different models of fitness for determining RNA structures. The analysis of free energies for intermediate foldings suggests that in some RNAs, the selective evolutionary pressure suppresses the possibilities for alternative structures that could form in the course of transcription. The algorithm had inherent incompatibilities of stems due to the binary representation of the solutions.

Wiese *et al.* [69] used GAs to predict the secondary structure of RNA molecules, where the secondary structure is encoded as a permutation similar to path representation in TSP (each helix is associated to one imaginary city) to overcome the inherent incompatibilities of binary representation for RNA molecule structure prediction. They showed that the problem can be decomposed into a combinatorial problem of finding the subset of helices from a set of feasible helices leading to a minimum energy [using (3)] in the molecule. More specifically, the algorithm predicts the specific canonical base pairs that will form hydrogen bonds and build helices. Different combinations of crossover and mutation probabilities ranging from 0.0 to 1.0 in increments of 0.01 and 0.1 were tested for 400 generations with a population size of 700 (maximum). Results on RNA sequences of lengths 76, 210, 681, and 785 nucleotides were provided. It was shown that the keep-best reproduction operator has similar benefits as in the traveling salesman problem domain. A comparison of several crossover operators was also provided.

A massively parallel GA for the RNA folding problem has been used in [70]–[72]. The authors demonstrated that the

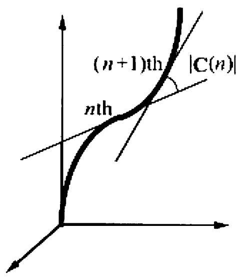

Fig. 4. Representation of the DNA curvature in terms of angular deviation between the local helical axes of the turn centered on the nth and $(n+1)$th basepairs [60].

GA with an improved mutation operator predicts more correct (true-positive) stems and more correct base pairs than could have been a predicted with DP algorithm.

I. Protein Structure Prediction and Classification

Identical protein sequences result in identical 3-D structures. So it follows that similar sequences may result in similar structures, and this is usually the case. The converse, however, is not true: identical 3-D structures do not necessarily indicate identical sequences. It is because of this that there is a distinction between "homology" and "similarity." There are examples of proteins in the databases that have nearly identical 3-D structures, and are therefore homologous, but do not exhibit significant (or detectable) sequence similarity. Pairwise comparisons do not readily show positions that are conserved among a whole set of sequences and tend to miss subtle similarities that become visible when observed simultaneously among many sequences. Thus, one wants to simultaneously compare several sequences.

Structural genomics is the prediction of the 3-D structure of a protein from the primary amino acid sequence [73]. This is one of the most challenging tasks in bioinformatics. The five levels of protein structure are described below. Three of them are illustrated in Fig. 5.

1) Primary structure is the sequence of amino acids that compose the protein.
2) The secondary structure of a protein is the spatial arrangement of the atoms constituting the main protein backbone. Linus Pauling was the first to develop a hypothesis for different potential protein secondary structures. He developed the α-helix structure and later the β-sheet structure for different proteins. An α-helix is a spiral arrangement of the protein backbone in the form of a helix with hydrogen bonding between side-chains. The β-sheets consist of parallel or antiparallel strands of amino acids linked to adjacent strands by hydrogen bonding. Collagen is an example of a protein with β-sheets serving as its secondary structure.
3) The super-secondary structure (or motif) is the local folding pattern built up from particular secondary structures. For example, the EF-hand motif consists of an α-helix, followed by a turn, followed by another α-helix.
4) Tertiary structure is formed by packing secondary structural elements linked by loops and turns into one or several compact globular units called domains; i.e., the folding of the entire protein chain.
5) A final protein may contain several protein subunits arranged in a quaternary structure.

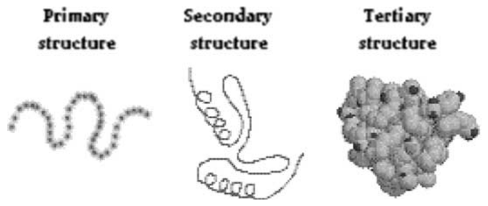

Fig. 5. Three levels of protein structure.

Protein sequences almost always fold into the same structure in the same environment. Hydrophobic interaction, hydrogen bonding, electrostatic, and other Van der Waals-type interactions also contribute to determine the structure of the protein. Many efforts are underway to predict the structure of a protein, given its primary sequence. A typical computation of protein folding would require computing all the spatial coordinates of atoms in a protein molecule, starting with an initial configuration and working up to a final minimum-energy folding configuration [10]. Sequence similarity methods can predict the secondary and tertiary structures based on homology to known proteins. Secondary structure predictions methods include Chou–Fasman [73], neural network [74], [75], nearest neighbor methods [76], [77], and Garnier–Osguthorpe–Robson [78]. Tertiary structure prediction methods are based on energy minimization, molecular dynamics, and stochastic searches EAs of conformational space.

Proteins clustered together into families are clearly evolutionarily related. Generally, this means that pairwise residue identities between the proteins are 30% and greater. Proteins that have low sequence identities, but whose structural and functional features suggest that a common evolutionary origin is probable, are placed together in superfamilies.

Methods: The work of Unger *et al.* [79]–[81] is one of the earlier investigations that discussed the reduced 3-D lattice protein folding problem for determining tertiary structure of protein in a GA framework. In this model, the energy function of protein chains is optimized. The encoding proposed by Unger *et al.* is a direct encoding of the direction of each peptide from the preceding peptide (five degrees of freedom, disallowing back move). Peptides are represented as single point units without side chains. Each peptide is represented by three bits to encode five degrees of freedom. The evaluation function solely evaluates nonsequential hydrophobe to hydrophobe contacts and is stated as a negative value (-1 per contact) with larger negative values indicating better energy conformations (thus stating the problem in terms of minimization). The algorithm begins with a population of identical unfolded configurations. Each generation begins with a series of K mutations being applied to each individual in the population, where K is equal to the length of the encoding. These mutations are filtered using a Monte Carlo acceptance algorithm which disallows lethal configurations (those with back move), always accepts mutations resulting in better energy, and accepts increased energy mutations based upon a threshold on the energy gain which becomes stricter over time. One-point crossover with an additional random mutation at the crossover point follows, producing a single offspring for each selected pair of parents; however, lethal configurations are rejected. In this situation, the crossover operation is retried for a given pair of parents until a nonlethal offspring can be located. Offspring are accepted using a second Monte Carlo filter which accepts all reduced energy confirmations and randomly accepts increased energy offspring again using a cooling threshold on the energy gain. The algorithm uses 100% replacement of all individuals in a generation through crossover except the single best, elitist, individual. Test data consisted of a series of ten randomly produced 27 length sequences and ten randomly produced 64 length sequences. The algorithm operated on each of the 27 and 64 length sequence for roughly 1.2 million and 2.2 million function evaluations, respectively, using a population size of 200. Performance comparisons were given between the above algorithm and a pure Monte Carlo approach which greatly favored the former. While the encoding and evaluation function proposed by Unger and Moult are fairly straightforward, the algorithm differs from a standard GA approach in several aspects. Most notable are the nonrandom initialization, the high level of mutation, and the Monte Carlo filtering of both the mutation and crossover results, which resembles typical simulated annealing approaches.

Patton *et al.* [82] determined tertiary structures of proteins based on the concept of Unger *et al.* [36], [40]. They enlarged the representation from three to seven bits per peptide in order to encode one of the 120 permutations of the five allowable directions for each. It was shown that the GA indeed appears to be effective for determining the tertiary structure with far fewer computational steps than that reported by Unger *et al.*

Natalio *et al.* [83], [84] investigated the impact of several algorithmic factors for a simple protein structure prediction problem: the conformational representation, the energy formulation, and the way in which infeasible conformations are penalized. Their analysis leads to specific recommendations for both GAs and other heuristic methods for solving PSP on the HP model. A detailed comparison between the work of Unger *et al.* and Patton *et al.* and an algorithm using GAs to overcome their limitations has also been presented [84].

A hill-climbing GA for simulation of protein folding has been described in [85]. The program builds a set of Cartesian points to represent an unfolded polypeptide's backbone. The dihedral angles determining the chain's configuration are stored in an array of chromosome structures that is copied and then mutated. The fitness of the mutated chain's configuration is determined by its radius of gyration. A four-helix bundle was used to optimize the simulation conditions. The program ran 50% faster than the other GA programs, and tests on 100 nonredundant structures produced results comparable to that of other GAs.

In [86], features are extracted from protein sequences using a position specific weight matrix. Thereafter, a genetic algorithm based fuzzy clustering scheme [87] is used for generating prototypes of the different superfamilies. Finally, superfamily classification of new sequences is performed by using the nearest neighbor rule.

Other investigations on protein structure prediction are available in [88]–[100]. An overview and state-of-the-art of the applications of EAs only for the protein folding problem is described in [101], whereas the relevance of GAs in several bioinformatics tasks is discussed in the present article.

J. Molecular Design and Molecular Docking

When two molecules are in close proximity, it can be energetically favorable for them to bind together tightly. The molecular docking problem is the prediction of energy and physical configuration of binding between two molecules. A typical application is in drug design, in which one might dock a small molecule that is a described drug to an enzyme one wishes to target. For example, HIV protease is an enzyme in the AIDS virus that is essential to its replication. The chemical action of the protease takes place at a localized active site on its surface. HIV protease inhibitor drugs are small molecules that bind to the active site in HIV protease and stay there, so that the normal functioning of the enzyme is prevented. Docking software allows us to evaluate a drug design by predicting whether it will be successful in binding tightly to the active site in the enzyme. Based on the success of docking, and the resulting docked configuration, designers can refine the drug molecule [102].

Molecular design and docking is a difficult optimization problem, requiring efficient sampling across the entire range of positional, orientational, and conformational possibilities [103]. The major problem in molecular binding is that the search space is very large and the computational cost increases tremendously with the growth of the degrees of freedom. A docking algorithm must deal with two distinct issues: a sampling of the conformational degrees of freedom of molecules involved in the complex, and an objective function (OF) to assess its quality.

For molecular design, the structure of a flexible molecule is encoded by an integer-valued or real-valued chromosome in GA, the ith element of which contains the torsion angle for the ith rotable bond. The energy for the specified structure (conformation) can be calculated using standard molecular modeling package, and this energy is used as the fitness function for the GA. GAs try to identify a set of torsion angle values that minimize the calculated energy. GA is becoming a popular choice for the heuristic search method in molecular design and docking applications [104]. Both canonical GAs and evolutionary programming methods are found to be successful in drug design and docking. Some of them are described below.

Methods: A novel and robust automated docking method that predicts the bound conformations (structures) of flexible ligands to macromolecular targets has been developed [105]. The method combines GAs with a scoring function that estimates the free energy change upon binding. This method applies a Lamarckian model of genetics, in which environmental adaptations of an individual's phenotype are reverse transcribed into its genotype and become inheritable traits. Three search methods, *viz.*, Monte Carlo simulated annealing, a traditional GA, and the Lamarckian GA were considered, and their performance was compared in dockings of seven protein-ligand test systems having known three-dimensional structure. The chromosome is composed of a string of realvalued genes: three Cartesian coordinates for the ligand translation; four variables defining a quaternion specifying the ligand orientation; and one real-value for each ligand torsion, in that order. The order of the genes that encode the torsion angles is defined by the torsion tree created by AUTOTORS (a preparatory program used to select rotatable bonds in the ligand). Thus, there is a one-to-one mapping from the ligand's state variables to the genes of the individuals chromosome. An Individual's fitness is the sum of the intermolecular interaction energy between the ligand and the protein, and the intramolecular interaction energy of the ligand. In the GA and LGA dockings, an initial population of 50 random individuals, a maximum number of 1.5×10^6 energy evaluations, a maximum number of 27 000 generations, a mutation rate of 0.02, and a crossover rate of 0.80 have been used. Proportional selection was used, where the average of the worst energy results was calculated over a window of the previous 10 generations.

Bagchi *et al.* [106], [107] presented an evolutionary approach for designing a ligand molecule that can bind to the active site of a target protein. A two-dimensional (2-D) model was considered. A variable string length genetic algorithm (VGA) was used for evolving an appropriate arrangement of the basic functional units of the molecule to be designed. The method is superior to fixed string length GA for designing a ligand molecule to target the human rhinovirus strain 14 (causative agent for AIDS).

Chen *et al.* [108] derived a population based annealing genetic algorithm (PAG) using GAs and simulated annealing (SA). They applied it to find binding structures for three drug protein molecular pairs, including the anti-cancer drug methotrexate (MTX). All of the binding results keep the energy at low levels, and have a promising binding geometrical structure in terms of number of hydrogen bonds formed. One of the design methods of PAG, which incorporates an annealing scheme with the normal probability density function as the neighbor generation method, was described in [109]. The algorithm was used for computer-aided drug design. Using a dihydrofolate reductase enzyme with the anti-cancer drug methotrexate and two analogs of the antibacterial drug trimethoprim, PAGs can find a drug structure within several hours. A similar work is available in [110].

Christopher *et al.* [111] evaluated the use of GAs with local search in molecular docking. They investigated several GA-local search hybrids and compared results with those obtained from simulated annealing in terms of optimization success, and absolute success in finding the true physical docked configuration.

Other relevant investigations are available in [104], [112]–[120]. A survey on the application of GAs for molecular modeling, docking of flexible ligands into protein active sites, and for *de novo* ligand design is described in [121]. Advantages and limitations of GAs are mentioned for the aforementioned tasks. In contrast, the present article provides a broader overview and state-of-the-art of the applications of EAs for several bioinformatics tasks.

IV. Conclusion

The increasing availability of annotated genomic sequences has resulted in the introduction of computational genomics and proteomics, large-scale analysis of complete genomes, and the proteins that they encode for relating specific genes to diseases. The rationale for applying computational approaches to facilitate the understanding of various biological processes mainly includes the following:

1) to provide a more global perspective in experimental design;
2) to capitalize on the emerging technology of database-mining: the process by which testable hypotheses are generated regarding the function or structure of a gene or protein of interest by identifying similar sequences in better characterized organisms.

GAs appear to be a very powerful artificial intelligence paradigm to handle these issues. This article provides an overview of different bioinformatics tasks and the relevance of GAs to handle them efficiently.

Even though the current approaches in biocomputing using EAs are very helpful in identifying patterns and functions of proteins and genes, the output results are still far from perfect. There are three general characteristics that might appear to limit the effectiveness of GAs. First, the basic selection, crossover, and mutation operators are common to all applications. Therefore, research is now focussed on designing problem specific operators to get better results. Second, a GA requires extensive experimentation for the specification of several parameters so that appropriate values can be identified. Third, GAs involve a large degree of randomness and different runs may produce different results, so it is necessary to incorporate problem specific domain knowledge into GA to reduce randomness and computational time and current research is going on in this direction also. The methods are not only time-consuming, requiring UNIX workstations to run on, but might also lead to false interpretations and assumptions due to necessary simplifications. It is therefore still mandatory to use biological reasoning and common sense in evaluating the results delivered by a biocomputing program. Also, for evaluation of the trustworthiness of the output of a program, it is necessary to understand its mathematical/theoretical background to finally come up with a useful and sense-full analysis.

Other potential bioinformatics tasks for which EA can be used include the following:

1) characterization of protein content and metabolic pathways between different genomes;
2) identification of interacting proteins;
3) assignment and prediction of gene products;
4) large-scale analysis of gene expression levels;
5) mapping expression data to sequence, structural and biochemical data.

References

[1] P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*. Cambridge, MA: MIT Press, 1998.

[2] R. B. Altman, A. Valencia, S. Miyano, and S. Ranganathan, "Challenges for intelligent systems in biology," *IEEE Intell. Syst.*, vol. 16, no. 6, pp. 14–20, Nov./Dec. 2001.

[3] D. Goldberg, *Genetic Algorithms in Optimization, Search, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[4] D. Bhandari, C. A. Murthy, and S. K. Pal, "Genetic algorithm with elitist model and its convergence," *Int. J. Pattern Rcognit. Artif. Intell.*, vol. 10, no. 6, pp. 731–747, 1996.

[5] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," *Artif. Intell.*, vol. 40, no. 1–3, pp. 235–282, 1989.

[6] M. Mitchell, S. Forrest, and J. H. Holland, "The royal road for genetic algorithms: Fitness landscapes and GA performance," in *Proc. 1st Eur. Conf. Artificial Life*, 1992.

[7] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Mach. Learn.*, vol. 3, pp. 95–100, 1988.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA: MIT Press, 1992.

[9] L. Davis, *Handbook of Genetic Algorithm*. New York: Van Nostrand Reinhold, 1991.

[10] J. Setubal and J. Meidanis, *Introduction to Computational Molecular Biology*. Boston, MA: Thomson, 1999.

[11] A. S. Wu and R. K. Lindsay, "A survey of intron research in genetics," in *Proc. 4th Conf. Parallel Problem Solving from Nature*, pp. 101–110, 1996.

[12] J. Chen, E. Antipov, B. Lemieux, W. Cedeno, and D. H. Wood, "DNA computing implementing genetic algorithms," in *Evolution as Computation*, New York: Springer-Verlag, pp. 39-49, 1999.

[13] R. J. Parsons, S. Forrest, and C. Burks, "Genetic algorithms, operators, and DNA fragment assembly," *Mach. Learn.*, vol. 21, no. 1–2, pp. 11–33, 1995.

[14] ——, "Genetic algorithms for DNA sequence assembly," in *Proc. 1st Int. Conf. Intelligent Systems in Molecular Biology*, pp. 310–318, 1993.

[15] R. J. Parsons and M. E. Johnson, "DNA fragment assembly and genetic algorithms. New results and puzzling insights," in *Int. Conf. Intelligent Systems in Molecular Biology*, 1995, pp. 277–284.

[16] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Biol.*, vol. 48, pp. 443–453, 1970.

[17] T. F. Smith and M. S. Waterman, "Identification of common molecular sequences," *J. Mol. Biol.*, vol. 147, pp. 195–197, 1981.

[18] E. Aart and V. P. Laarhoven, *Simulated Annealing: A Review of Theory and Applications*. Norwell, MA: Kluwer, 1987.

[19] C. Lawrence, S. Altschul, M. Boguski, J. Liu, A. Neuwald, and J. Wootton, "Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment," *Science*, vol. 262, pp. 208–214, 1993.

[20] C. Notredame and D. G. Higgins, "SAGA: Sequence alignment by genetic algorithm," *Nucleic Acids Res.*, vol. 24, no. 8, pp. 1515–1524, 1996.

[21] C. Zhang and A. K. C. Wong, "A genetic algorithm for multiple molecular sequence alignment," *Bioinformatics*, vol. 13, pp. 565–581, 1997.

[22] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed., 1989, pp. 61–69.

[23] T. Yokoyama, T. Watanabe, A. Taneda, and T. Shimizu, "A web server for multiple sequence alignment using genetic algorithm," *Genome Inf.*, vol. 12, pp. 382–383, 2001.

[24] O. O'Sullivan, K. Suhre, C. Abergel, D. G. Higgins, and C. Notredame, "3DCoffee: Combining protein sequences and structures within multiple sequence alignments," *J. Mol. Biol.*, vol. 340, no. 2, pp. 385–395, 2004.

[25] C. Notredame, E. A. O'Brien, and D. G. Higgins, "RAGA: RNA sequence alignment by genetic algorithm," *Nucleic Acids Res.*, vol. 25, no. 22, pp. 4570–4580, 1997.

[26] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic Acids Res.*, vol. 22, pp. 4673–4680, 1994.

[27] C. Zhang and A. K. C. Wong, "A technique of genetic algorithm and sequence synthesis for multiple molecular sequence alignment," in *Proc. IEEE Int. Conf. Syst. Man, and Cybernetics*, vol. 3, 1998, pp. 2442–2447.

[28] ——, "Toward efficient multiple molecular sequence alignment: A system of genetic algorithm and dynamic programming," *IEEE Trans. Syst. Man, Cybern. B*, vol. 27, no. 6, pp. 918–932, Dec. 1997.

[29] T. Murata and H. Ishibuchi, "Positive and negative combination effects of crossover and mutation operators in sequencing problems," *Evol. Comput.*, vol. 20–22, pp. 170–175, 1996.

[30] C. Zhang, "A genetic algorithm for molecular sequence comparison," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, vol. 2, 1994, pp. 1926–1931.

[31] J. D. Szustakowski and Z. Weng, "Protein structure alignment using a genetic algorithm," *Proteins*, vol. 38, no. 4, pp. 428–440, 2000.

[32] K. Hanada, T. Yokoyama, and T. Shimizu, "Multiple sequence alignment by genetic algorithm," *Genome Inf.*, vol. 11, pp. 317–318, 2000.

[33] L. A. Anbarasu, P. Narayanasamy, and V. Sundararajan, "Multiple molecular sequence alignment by island parallel genetic algorithm," *Current Sci.*, vol. 78, no. 7, pp. 858–863, 2000.

[34] H. D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga, "A parallel hybrid genetic algorithm for multiple protein sequence alignment," in *Proc. Congress Evolutionary Computation*, vol. 1, 2002, pp. 309–314.

[35] C. Gaspin and T. Schiex, "Genetic algorithms for genetic mapping," in *Proc. 3rd Eur. Conf. Artificial Evolution*, 1997, pp. 145–156.

[36] J. Gunnels, P. Cull, and J. L. Holloway, "Genetic algorithms and simulated annealing for gene mapping," in *Proc. 1st IEEE Conf. Evolutionary Computation*, 1994, pp. 385–390.

[37] H. Murao, H. Tamaki, and S. Kitamura, "A coevolutionary approach to adapt the genotype-phenotype map in genetic algorithms," in *Proc. Congress Evolutionary Computation*, vol. 2, 2002, pp. 1612–1617.

[38] J. Fickett and M. Cinkosky, "A genetic algorithm for assembling chromosome physical maps," in *Proc. 2nd Int. Conf. Bioinformatics, Supercomputing, and Complex Genome Analysis*, 1993, pp. 272–285.

[39] J. W. Fickett, "Finding genes by computer: The state of the art," *Trends Genetics*, vol. 12, no. 8, pp. 316–320, 1996.

[40] A. Kel, A. Ptitsyn, V. Babenko, S. Meier-Ewert, and H. Lehrach, "A genetic algorithm for designing gene family-specific oligonucleotide sets used for hybridization: The G protein-coupled receptor protein superfamily," *Bioinformatics*, vol. 14, no. 3, pp. 259–270, 1998.

[41] V. G. Levitsky and A. V. Katokhin, "Recognition of eukaryotic promoters using a genetic algorithm based on iterative discriminant analysis," *In Silico Biol.*, vol. 3, no. 1–2, pp. 81–87, 2003.

[42] S. Knudsen, "Promoter2.0: For the recognition of PolII promoter sequences," *Bioinformatics*, vol. 15, pp. 356–361, 1999.

[43] N. M. Luscombe, D. Greenbaum, and M. Gerstein, "What is bioinformatics? A proposed definition and overview of the field," in *Yearbook Medical Informatics*: Edmonton, AB, Canada: IMIA, 2001, pp. 83–100.

[44] J. Quackenbush, "Computational analysis of microarray data," *Nat. Rev. Genetics*, vol. 2, pp. 418–427, 2001.

[45] H. K. Tsai, J. M. Yang, and C. Y. Kao, "Applying genetic algorithms to finding the optimal order in displaying the microarray data," in *Proc. GECCO*, 2002, pp. 610–617.

[46] H. K. Tsai, J. M. Yang, Y. F. Tsai, and C. Y. Kao, "An evolutionary approach for gene expression patterns," *IEEE Trans. Inf. Technol. Biomed.*, vol. 8, no. 2, pp. 69–78, Jun. 2004.

[47] A. S. Wu and I. Garibay, "The proportional genetic algorithm: Gene expression in a genetic algorithm," *Genetic Programm. Evol. Hardware*, vol. 3, no. 2, pp. 157–192, 2002.

[48] T. Akutsu, S. Miyano, and S. Kuhara, "Identification of genetic networks from a small number of gene expression patterns under the boolean network model," in *Proc. Pacific Symp. Biocomputing*, vol. 99, 1999, pp. 17–28.

[49] S. Ando and H. Iba, "Inference of gene regulatory model by genetic algorithms," in *Proc. Congress Evolutionary Computation*, vol. 1, 2001, pp. 712–719.

[50] N. Behera and V. Nanjundiah, "Trans gene regulation in adaptive evolution: A genetic algorithm model," *J. Theore. Biol.*, vol. 188, pp. 153–162, 1997.

[51] S. Ando and H. Iba, "Quantitative modeling of gene regulatory network-identifying the network by means of genetic algorithms," presented at the 11th Genome Informatics Workshop, 2000.

[52] ——, "The matrix modeling of gene regulatory networks-reverse engineering by genetic algorithms-," in *presented at the Atlantic Symp. Computational Biology and Genome Information Systems and Technology*, 2001.

[53] D. Tominaga, M. Okamoto, Y. Maki, S. Watanabe, and Y. Eguchi, "Nonlinear numerical optimization technique based on a genetic algorithm for inverse problems: Towards the inference of genetic networks," *Comput. Science and Biology (Proc. German Conf. Bioinformatics)*, 1999, pp. 127–140.

[54] P. O. Lewis, "A genetic algorithm for maximum likelihood phylogeny inference using nucleotide sequence data," *Mol. Biol. Evol.*, vol. 15, no. 3, pp. 277–283, 1998.

[55] A. R. Lemmon and M. C. Milinkovitch, "The metapopulation genetic algorithm: An efficient solution for the problem of large phylogeny estimation," *Proc. Nat. Acad. Sci.*, vol. 99, no. 16, pp. 10516–10521, 2002.

[56] K. Katoh, K. Kuma, and T. Miyata, "Genetic algorithm-based maximum-likelihood analysis for molecular phylogeny," *J. Mol. Evol.*, vol. 53, no. 4-5, pp. 477–484, 2001.

[57] H. Matsuda, "Protein phylogenetic inference using maximum likelihood with a genetic algorithm," *Pacific Symp. Biocomputing*, 1996, pp. 512–523.

[58] A. Skourikhine, "Phylogenetic tree reconstruction using self-adaptive genetic algorithm," in *IEEE Int. Symp. Bio-Informatics and Biomedical Engineering*, 2000, pp. 129–134.

[59] P. Baldi and P. F. Baisnee, "Sequence analysis by additive scales: DNA structure for sequences and repeats of all lengths," *Bioinformatics*, vol. 16, pp. 865–889, 2000.

[60] C. Anselmi, G. Bocchinfuso, P. De Santis, M. Savino, and A. Scipioni, "A theoretical model for the prediction of sequence-dependent nucleosome thermodynamic stability," *J. Biophys.*, vol. 79, no. 2, pp. 601–613, 2000.

[61] L. D. Landau and E. M. Lifshitz, *Theory of Elasticity*. New York: Pergamon, 1970.

[62] M. L. Beckers, L. M. Buydens, J. A. Pikkemaat, and C. Altona, "Application of a genetic algorithm in the conformational analysis of methylene-acetal-linked thymine dimers in DNA: Comparison with distance geometry calculations," *J. Biomol. NMR*, vol. 9, no. 1, pp. 25–34, 1997.

[63] R. V. Parbhane, S. Unniraman, S. S. Tambe, V. Nagaraja, and B. D. Kulkarni, "Optimum DNA curvature using a hybrid approach involving an artificial neural network and genetic algorithm," *J. Biomol. Struct. Dyn.*, vol. 17, no. 4, pp. 665–672, 2000.

[64] J. P. Adrahams and M. Breg, "Prediction of RNA secondary structure including pseudoknotting by computer simulation," *Nucleic Acids Res.*, vol. 18, pp. 3035–3044, 1990.

[65] M. Waterman, "RNA structure prediction," in *Methods in Enzymology*. San Diego, CA: Academic, vol. 164, 1988.

[66] M. Zuker and P. Stiegler, "Optimal computer folding of large RNA sequences using thermo-dynamics and auxiliary information," *Nucleic Acids Res.*, vol. 9, pp. 133–148, 1981.

[67] V. Batenburg, A. P. Gultyaev, and C. W. A. Pleij, "An APL-programmed genetic algorithm for the prediction of RNA secondary structure," *J. Theoret. Biol.*, vol. 174, no. 3, pp. 269–280, 1995.

[68] A. P. Gultyaev, V. Batenburg, and C. W. A. Pleij, "The computer simulation of RNA folding pathways using an genetic algorithm," *J. Mol. Biol.*, vol. 250, pp. 37–51, 1995.

[69] K. C. Wiese and E. Glen, "A permutation-based genetic algorithm for the RNA folding problem: A critical look at selection strategies, crossover operators, and representation issues," *Biosystems*, vol. 72, no. 1–2, pp. 29–41, 2003.

[70] B. A. Shapiro and J. Navetta, "A massively parallel genetic algorithm for RNA secondary structure prediction," *J. Supercomput.*, vol. 8, pp. 195–207, 1994.

[71] B. A. Shapiro and J. C. Wu, "An annealing mutation operator in the genetic algorithms for RNA folding," *Comput. Appl. Biosci.*, vol. 12, pp. 171–180, 1996.

[72] B. A. Shapiro, J. C. Wu, D. Bengali, and M. J. Potts, "The massively parallel genetic algorithm for RNA folding: MIMD implementation and population variation," *Bioinformatics*, vol. 17, no. 2, pp. 137–148, 2001.

[73] P. Chou and G. Fasmann, "Prediction of the secondary structure of proteins from their amino acid sequence," *Adv. Enzymol.*, vol. 47, pp. 145–148, 1978.

[74] S. K. Riis and A. Krogh, "Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments," *J. Comput. Biol.*, vol. 3, pp. 163–183, 1996.

[75] N. Qian and T. J. Sejnowski, "Predicting the secondary structure of globular proteins using neural network models," *J. Mol. Biol.*, vol. 202, no. 4, pp. 865–884, 1988.

[76] A. Salamov and V. Solovyev, "Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignments," *J. Mol. Biol.*, vol. 247, pp. 11–15, 1995.

[77] S. Salzberg and S. Cost, "Predicting protein secondary structure with a nearest-neighbor algorithm," *J. Mol. Biol.*, vol. 227, pp. 371–374, 1992.

[78] J. Garnier, J. F. Gibrat, and B. Robson, "GOR method for predicting protein secondary structure from amino acid sequence," in *Methods in Enzymology*, vol. 266, 1996, pp. 540–553.

[79] R. Unger and J. Moult, "On the applicability of genetic algorithms to protein folding," in *Proc. Hawaii Int. Conf. System Sciences*, vol. 1, 1993, pp. 715–725.

[80] ——, "Genetic algorithms for protein folding simulations," *J. Mol. Biol.*, vol. 231, no. 1, pp. 75–81, 1993.

[81] ——, "A genetic algorithms for three dimensional protein folding simulations," in *Int. Conf. Genetic Algorithms*, 1993, pp. 581–588.

[82] A. Patton, W. P., III, and E. Goldman, "A standard GA approach to native protein conformation prediction," in *Proc. Int. Conf. Genetic Algorithms*, 1995, pp. 574–581.

[83] N. Krasnogor, W. E. Hart, J. Smith, and D. A. Pelta, "Protein structure prediction with evolutionary algorithms," in *Proc. Genetic and Evolutionary Computation*, vol. 2, 1999, pp. 1596–1601.

[84] N. Krasnogor, D. Pelta, P. M. Lopez, P. Mocciola, and E. Canal, "Genetic algorithms for the protein folding problem: A critical view," in *Proc. Engineering Intelligent Systems*, 1998

[85] L. Cooper, D. Corne, and M. Crabbe, "Use of a novel hill-climbing genetic algorithm in protein folding simulations," *Comput. Biol. Chem.*, vol. 27, no. 6, pp. 575–580, 2003.

[86] S. Bandyopadhyay, "An efficient technique for superfamily classification of amino acid sequences: Feature extraction, fuzzy clustering and prototype selection," *Fuzzy Sets Syst.*, vol. 152, pp. 5–16, 2005.

[87] U. Maulik and S. Bandyopadhyay, "Fuzzy partitioning using real coded variable length genetic algorithm for pixel cassification," *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 5, pp. 1075–1081, May 2003.

[88] H. Iijima and Y. Naito, "Incremental prediction of the side-chain conformation of proteins by a genetic algorithm," in *Proc. IEEE Conf. Evolutionary Computation*, vol. 1, 1994, pp. 362–367.

[89] I. Ono, H. Fujiki, M. Ootsuka, N. Nakashima, N. Ono, and S. Tate, "Global optimization of protein 3-dimensional structures in NMR by a genetic algorithm," in *Proc. Congress Evolutionary Computation*, vol. 1, 2002, pp. 303–308.

[90] B. Contreras-Moreira, P. W. Fitzjohn, M. Offman, G. R. Smith, and P. A. Bates, "Novel use of a genetic algorithm for protein structure prediction: Searching template and sequence alignment space," *Proteins*, vol. 53, no. 6, pp. 424–429, 2003.

[91] P. Saxena, I. Whang, Y. Voziyanov, C. Harkey, P. Argos, M. Jayaram, and T. Dandekar, "Probing Flp: A new approach to analyze the structure of a DNA recognizing protein by combining the genetic algorithm, mutagenesis and non-canonical DNA target sites," *Biochem. Biophys. Acta.*, vol. 1340, no. 2, pp. 187–204, 1997.

[92] J. T. Pedersen and J. Moult, "Protein folding simulations with genetic algorithms and a detailed molecular description," *J. Mol. Biol.*, vol. 269, no. 2, pp. 240–259, 1997.

[93] M. Khimasia and P. Coveney, "Protein structure prediction as a hard optimization problem: The genetic algorithm approach," *Mol. Simul.*, vol. 19, pp. 205–226, 1997.

[94] R. Konig and T. Dandekar, "Improving genetic algorithms for protein folding simulations by systematic crossover," *BioSystems*, vol. 50, pp. 17–25, 1999.

[95] C. A. Del Carpio, "A parallel genetic algorithm for polypeptide three dimensional structure prediction: A transputer implementation," *J. Chem. Inf. Comput. Sci.*, vol. 36, no. 2, pp. 258–269, 1996.

[96] Rabow and H. A. Scheraga, "Improved genetic algorithm for the protein folding problem by use of a cartesian combination operator," *Protein Sci.*, vol. 5, pp. 1800–1815, 1996.

[97] J. R. Gunn, "Sampling protein conformations using segment libraries and a genetic algorithm," *J. Chemi. Phys.*, vol. 106, pp. 4270–4281, 1997.

[98] A. C. W. May and M. S. Johnson, "Improved genetic algorithm-based protein structure comparisons: Pairwise and multiple superpositions," *Protein Eng.*, vol. 8, pp. 873–882, 1995.

[99] M. J. Bayley, G. Jones, P. Willett, and M. P. Williamson, "Genfold: A genetic algorithm for folding protein structures using NMR restraints," *Protein Sci.*, vol. 7, no. 2, pp. 491–499, 1998.

[100] Z. Sun, X. Xia, Q. Guo, and D. Xu, "Protein structure prediction in a 210-type lattice model: Parameter optimization in the genetic algorithm using orthogonal array," *J. Protein Chem.*, vol. 18, no. 1, pp. 39–46, 1999.

[101] S. Schulze-Kremer, "Genetic algorithms and protein folding. Methods in molecular biology," *Protein Structure Prediction: Methods and Protocols*, vol. 143, pp. 175–222, 2000.

[102] A. M. Lesk, *Introduction to Bioinformatics*. London, U.K.: Oxford Univ. Press, 2002.

[103] Y. Xiao and D. Williams, "Genetic algorithms for docking of actinomycin D and deoxyguanosine molecules with comparison to the crystal structure of actinomycin ddeoxyguanosine complex," *J. Phys. Chem.*, vol. 98, pp. 7191–7200, 1994.

[104] D. R. Westhead, D. E. Clark, D. Frenkel, J. Li, C. W. Murray, B. Robson, and B. Waszkowycz, "PRO-LIGAND: An approach to de novo molecular design. 3. A genetic algorithm for structure refinement," *J. Comput.-Aided Mol. Design*, vol. 9, no. 2, pp. 139–148, 1995.

[105] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olsoni, "Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function," *J. Comput. Chem.*, vol. 19, no. 14, pp. 1639–1662, 1998.

[106] A. Bagchi, S. Bandyopadhyay, and U. Maulik, "Determination of molecular structure for drug design using variable string length genetic algorithm," in *Workshop on Soft Computing, High Performance Computing (HiPC) Workshops 2003: New Frontiersin High-Performance Computing,* Hyderabad, India, 2003, pp. 145–154.

[107] S. Bandyopadhyay, A. Bagchi, and U. Maulik, “Active site driven ligand design: An evolutionary approach,” *J. Bioinf. Comput. Biol.*, vol. 3, no. 5, pp. 1053–1070, 2005.
[108] C. Chen, L. H. Wang, C. Kao, M. Ouhyoung, and W. Chen, “Molecular binding in structure-based drug design: A case study of the population-based annealing genetic algorithms,” in *Proc. IEEE Int. Conf. Tools with Artificial Intelligence*, 1998, pp. 328–335.
[109] L. H. Wang, C. Kao, M. Ouh-Young, and W. Chen, “Molecular binding: A case study of the population-based annealing genetic algorithms,” in *Proc. IEEE Int. Conf. Evolutionary Computation*, 1995, pp. 50–55.
[110] L. H. Wang, C. Kao, M. Ouh-Young, and W. C. Cheu, “Using an annealing genetic algorithm to solve global energy minimization problem in molecular binding,” in *Proc. 6th Int. Conf. Tools with Artificial Intelligence*, 1994, pp. 404–410.
[111] C. D. Rosin, R. S. Halliday, W. E. Hart, and R. K. Belew, “A comparison of global and local search methods in drug docking,” in *Proc. Int. Conf. Genetic Algorithms*, 1997, pp. 221–228.
[112] J. M. Yang and C. Y. Kao, “A family competition evolutionary algorithm for automated docking of flexible ligands to proteins,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 4, no. 3, pp. 225–237, Sep. 2000.
[113] C. M. Oshiro, I. D. Kuntz, and J. S. Dixon, “Flexible ligand docking using a genetic algorithm,” *J. Comput.-Aided Mol. Design*, vol. 9, no. 2, pp. 113–130, 1995.
[114] D. E. Clark and D. R. Westhead, “Evolutionary algorithms in computer-aided molecular design,” *J. Comput.-Aided Mol. Design*, vol. 10, no. 4, pp. 337–358, 1996.
[115] V. Venkatasubramanian, K. Chan, and J. Caruthers, “Computer aided molecular design using genetic algorithms,” *Comput. Chem. Eng.*, vol. 18, no. 9, pp. 833–844, 1994.
[116] D. M. Deaven and K. O. Ho, “Molecular-geometry optimization with a genetic algorithm,” *Phys. Rev. Lett.*, vol. 75, no. 2, pp. 288–291, 1995.
[117] G. Jones, P. Willett, and R. C. Glen, “Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation,” *J. Mol. Biol.*, vol. 245, pp. 43–53, 1995.
[118] G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor, “Further development of a genetic algorithm for ligand docking and its application to screening combinatorial libraries,” *American Chemical Society Symposium Series*, vol. 719, Washington, DC: ACS, 1999, pp. 271–291.
[119] D. B. McGarrah and R. S. Judson, “Analysis of the genetic algorithm method of molecular conformation determination,” *J. Comput. Chem.*, vol. 14, no. 11, pp. 1385–1395, 1993.
[120] T. Hou, J. Wang, L. Chen, and X. Xu, “Automated docking of peptides and proteins by using a genetic algorithm combined with a tabu search,” *Protein Eng.*, vol. 12, pp. 639–647, 1999.
[121] P. Willet, “Genetic algorithms in molecular recognition and design,” *Trends Biotechnol.*, vol. 13, no. 12, pp. 516–521, 1995.

Sankar K. Pal (M’81–SM’84–F’93) received the Ph.D. degree in radio physics and electronics from the University of Calcutta, Calcutta, India, in 1974, and the Ph.D. degree in electrical engineering along with DIC from Imperial College, University of London, London, U.K., in 1982.

He is the Director and Distinguished Scientist of the Indian Statistical Institute, Calcutta. He founded the Machine Intelligence Unit in 1993, and the Center for Soft Computing Research: A National Facility in 2004 at the Institute in Calcutta. He worked at the University of California, Berkeley and the University of Maryland, College Park, in 1986–1987; the NASA Johnson Space Center, Houston, TX, in 1990–1992 and 1994; and in the U.S. Naval Research Laboratory, Washington, DC, in 2004. He is a co-author of 13 books and about 300 research publications. Since 1997 he has been serving as a Distinguished Visitor of the IEEE Computer Society for the Asia-Pacific Region, and held several visiting positions in Hong Kong and Australian universities. He is an Associate Editor of *Pattern Recognition Letters, Neurocomputing, Applied Intelligence, Information Sciences, Fuzzy Sets and Systems, Fundamenta Informaticae and the International Journal of Computational Intelligence and Applications;*

Prof. Pal is a Fellow of the Third World Academy of Sciences, International Association for Pattern recognition, and all the four National Academies for Science/Engineering in India. He has received the 1990 S.S. Bhatnagar Prize (which is the most coveted award for a scientist in India), and many prestigious awards in India and abroad including the 1999 G. D. Birla Award, 1998 Om Bhasin Award, 1993 Jawaharlal Nehru Fellowship, 2000 Khwarizmi International Award from the Islamic Republic of Iran, 2000–2001 FICCI Award, 1993 Vikram Sarabhai Research Award, 1993 NASA Tech Brief Award, 1994 IEEE TRANSACTION NEURAL NETWORKS Outstanding Paper Award, 1995 NASA Patent Application Award, 1997 IETE-R. L. Wadhwa Gold Medal, and the 2001 INSA-S.H. Zaheer Medal. He is an Associate Editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, and the IEEE TRANSACTIONS NEURAL NETWORKS He is a Member, Executive Advisory Editorial Board, IEEE TRANSACTIONS FUZZY SYSTEMS, *Int. Journal on Image and Graphics*, and *Int. Journal of Approximate Reasoning*; and a Guest Editor of *IEEE Computer.*

Sanghamitra Bandyopadhyay (SM’05) received the B.Sc. and B.Tech. degrees in physics and computer science in 1988 and 1992, respectively, the Masters degree in computer science from the Indian Institute of Technology (IIT), Kharagpur, in 1994, and the Ph.D. degree in computer science from the Indian Statistical Institute, Calcutta, in 1998.

Currently, she is an Associate Professor at the Indian Statistical Institute. She has worked for Los Alamos National Laboratory, in 1997, as a Graduate Research Assistant, in the University of New South Wales, in 1999, as a Post Doctoral Fellow, in the Department of Computer Science and Engineering, University of Texas at Arlington, in 2001 as Researcher, and in the Department of Computer Science and Engineering, University of Maryland, in 2004 as Visiting Research Faculty. Her research interests include evolutionary and soft computation, pattern recognition, data mining, bioinformatics, parallel and distributed systems and VLSI. She has published over 70 articles in international journals, conference, and workshop proceedings, edited books and journal special issues, and served on the committees of several conferences and workshops. She is on the editorial board of the *International Journal on Computational Intelligence*.

Dr. Bandyopadhyay is the first recipient of the Dr. Shanker Dayal Sharma Gold Medal and Institute Silver Medal for being the best all round post graduate performer in IIT, Kharagpur in 1994. She received the Indian National Science Academy and the Indian Science Congress Association Young Scientist Awards in 2000, as well as the Indian National Academy of Engineering Young Engineers’ Award in 2002. She is serving as the Program Co-Chair of the 1st International Conference on Pattern Recognition and Machine Intelligence, 2005, to be held in Kolkata, India, and has served as the Tutorial Co-Chair, World Congress on Lateral Computing, 2004, held in Bangalore, India.

Shubhra Sankar Ray received the M.Sc. and M.Tech. degrees in electronic science and radiophysics and electronics from the University of Calcutta, Kolkata, India, in 2000 and 2002, respectively.

Since June 2003, he has been a Senior Research Fellow of the Council of Scientific and Industrial Research, New Delhi, India, working at the Machine Intelligence Unit, Indian Statistical Institute, Kolkata. His research interests include bioinformatics, evolutionary computation, neural networks, and data mining.