Gene Identification: Classical and Computational Intelligence Approaches

Sanghamitra Bandyopadhyay, *Senior Member, IEEE*, Ujjwal Maulik, *Senior Member, IEEE*, and Debadyuti Roy

***Abstract*—Automatic identification of genes has been an actively researched area of bioinformatics. Compared to earlier attempts for finding genes, the recent techniques are significantly more accurate and reliable. Many of the current gene-finding methods employ computational intelligence techniques that are known to be more robust when dealing with uncertainty and imprecision. In this paper, a detailed survey on the existing classical and computational intelligence based methods for gene identification is carried out. This includes a brief description of the classical and computational intelligence methods before discussing their applications to gene finding. In addition, a long list of available gene finders is compiled. For the convenience of the readers, the list is enhanced by mentioning their corresponding Web sites and commenting on the general approach adopted. An extensive bibliography is provided. Finally, some limitations of the current approaches and future directions are discussed.**

***Index Terms*—Bioinformatics, case-based reasoning, decision tree, gene finding, genetic algorithms (GAs), neural networks.**

I. Introduction

IDENTIFYING genes from DNA sequences is an important problem in bioinformatics [1], [2]. As the human genome project came to an end in 2003, all the human chromosomes have been sequenced [3], pegging the estimated number of genes to 20 000–25 000 [4]. Thus, the development of reliable automated techniques for interpreting long anonymous genomic sequences (i.e., partitioning them into genes, promoters, regulatory elements, intergenic region, etc.) became imperative. As a consequence, computational approaches to gene finding have attracted a lot more attention in the molecular biology and genomics community [5]–[7]. Significant advances in gene-finding methodology have taken place, and the current methods are considerably more accurate, reliable, and useful than those available in the past. Hidden Markov model (HMM), decision trees, and dynamic-programming-based approaches for gene finding are discussed in [8]. A summary of the available programs for gene finding is also reported in [8]. A comparative study of the gene structure prediction methods is available in [9]. Gelfand [10] provides an extensive review of several methods for prediction of functional sites, tRNA, and protein-coding genes. A brief overview of some of the computational methods for gene finding is available in [11]. A review of the existing approaches to predict genes in eukaryotic genomes and the limitations of these methods are given in [12]. An extensive bibliography and many more related resources on computational gene recognition are available in [13] and [14].

In addition to the conventional gene-finding methods like those based on HMM and dynamic programming, approaches based on computational intelligence techniques have gained popularity in recent times. Computational intelligence methods are known to be more robust, and they provide greater approximation of the results of the problem under consideration. For example, a neural-network-based protein secondary structure prediction method (PHDsec, a component of a suite of programs called PHD for predicting the one-dimensional structure of proteins [15]) was the first to surpass a level of 70% overall three-state per-residue accuracy. Moreover, even if some of the links fail, the performance of the network degrades gradually. Computational intelligence techniques also work well for the gene-finding problem because they can handle approximate nature, incompleteness, and uncertainty of data.

In this paper, the problem of gene identification, along with the issues involved in it, are first described. Thereafter, the classical approaches based on HMM, Bayesian networks, and dynamic programming are discussed. Finally, a review of some of the computational intelligence techniques for this problem is provided. Most of the previous reviews on the topic have concentrated on a few conventional gene-finding methods. Subsequently, several sophisticated techniques like those based on computational intelligence approaches have been developed. However, no review of such approaches has been presented in the last few years. The recent developments in various gene-finding methods, especially using computational intelligence techniques like neural networks and genetic algorithms, have been summarized in the present paper. A brief history, as well as a description of these methods, is provided. Web addresses for most of the gene-finding software and a fairly extensive bibliography are also included. Some of the limitations of the current methods have been mentioned.

II. Background and Problem Definition

In this section, some basic terminology related to the problem of gene identification are given first. The problem is then stated formally.

A. Basic Terminology

This section provides a brief overview of the basic concepts and terminologies in genetics, as well as some of the issues involved in finding genes in DNA sequences.

Proteins are considered the building blocks of life and are the most abundant type of organic molecules in living organisms. A *protein* is made up of one or more polypeptide chains. *A polypeptide* chain is a chain of amino acids. An *amino acid* is an organic molecule consisting of a carbon atom bound to one hydrogen atom, a carboxyl group, an amino group, and a side group that varies from amino acid to amino acid. When an organism reproduces, it is imperative that the instructions for building its proteins are reproduced accurately and completely. These instructions are largely maintained by another type of organic molecule called *nucleotides*.

Nucleotides are joined together to form large molecules called *nucleic acids*. The two most common types of nucleic acids are deoxyribonucleic acid (DNA) and ribonucleic acid (RNA). DNA is made up of four nucleotides: adenine, guanine, cytosine, and thymine, abbreviated A, G, C, and T, respectively. In DNA, A pairs with T and G pairs with C, because of the presence of hydrogen bonds. A molecule of DNA is organized in the form of two complementary chains of nucleotides wound in a double helix. One of these chains (or strands) is called the sense strand, and other the antisense strand or the template strand. The antisense strand is the one that contains the genetic code of a gene, and is transcribed. Note that, in general, at any place in a DNA molecule, either of the two strands may be serving as the antisense strand. A *gene* is a segment of DNA that codes for a specific functional product (e.g., a protein or noncoding RNA molecule). Though genes lie linearly along chromosomes, they are not necessarily contiguous. *Intergenic regions* are the regions of DNA in between genes. These regions contain a few or no genes, and parts of it sometimes control genes that are close by. However, most of the intergenic region has no currently known function. *Introns* are segments of DNA found within genes. Introns are transcribed into RNA along with the rest of the gene but must be removed from the RNA before the mature RNA product is complete. Fig. 1 illustrates the organization of a gene on a DNA strand. After the introns are spliced out, the remaining segments of RNA are *exons*. The intron exon boundary is often called the splice site. Exons are joined together to become the mature RNA product that is usually changed to the corresponding protein via the process of translation. Though most of the gene-finding software use the term exon to denote only the coding parts, in reality, there are some exons (or parts of them) that are noncoding. In this review, however, we have referred to the correct biological definition of exons and explicitly mention when only their coding part is concerned. The basic principle by which cellular DNA is converted to its functional component, the protein, is known as the central dogma of molecular biology [16], depicted in Fig. 2. The process is described, in some detail, in Section II- C. For more details, the readers may refer to [16] and [17].

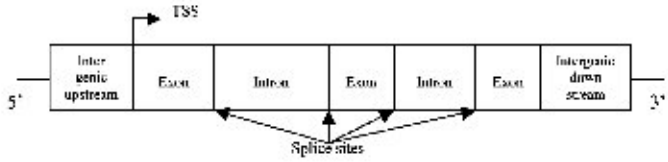

Fig. 1. Basic structure of a gene on a DNA strand.

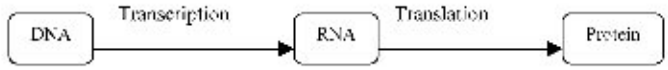

Fig. 2. Central dogma of molecular biology.

B. Tasks in Bioinformatics and Their Applications

The different biological tasks considered within the scope of bioinformatics that are within the purview of computational molecular biology can be broadly classified into two categories: genomics and proteomics, which involve the study of genes, proteins, nucleic acid structure prediction, and molecular design with docking. For raw DNA sequences, investigations include, among others, the following:

1) separating coding and noncoding regions;
2) identification of introns, exons, and promoter regions for annotating genomic DNA;
3) gene product prediction;
4) analysis of genomic sequences of simple organisms simplifies the problem of understanding complex genomes (its applications are mainly in gene product prediction and forensic analysis).

Besides these, phylogenetics, microarray analysis, gene regulatory networks, and metabolic pathways are other fields that are much studied in this domain. Some other areas of immense interest within bioinformatics, which are well above the aforementioned level, include the optimization of drug administration schedule based on drug response data, automatic determination of cancer types from image data (e.g., skin cancer, breast cancer), and determining the stage of cancer from cellular images of the infected areas (e.g., cervical cancer).

Since the purpose of this paper is to provide an overview of the different techniques for gene identification, this is described later in more detail. Note that the present review deals with only protein-coding gene-finding techniques. A completely different class of algorithms are used for noncoding RNA gene finding, which is outside the purview of this paper.

C. Problem of Finding Protein-Coding Genes

The general problem of finding protein-coding genes is as follows. Given a sequence of DNA, how does one determine the location of protein-coding genes, which are the regions containing information that code for proteins. At a very general level, nucleotides can be classified as belonging to:

- coding regions in a gene, i.e., exons;
- noncoding regions in a gene, i.e., introns;
- intergenic regions.

The problem can then be formally stated as follows.

Input: A sequence of DNA

$$X = (x_1, \ldots, x_n) \in \sum^*, \quad \text{where } \sum = \{A, T, C, G\}$$

Output: Correct labeling of each element in X as belonging to a coding region, noncoding region, or intergenic region.

As shown in Fig. 1, on a very high level, genes in human DNA and many other organisms have a relatively regular structure. The beginning of a gene is at the transcription start site (TSS). This is followed by an exon, which is a sequence of DNA that is mostly expressed in the final protein product (or an RNA product). A series of intron–exon pairs may then follow, where introns separate the exons. At the intron–exon boundaries, there are splice junctions (or acceptor/donor sites) that aid the process of RNA splicing. Transcription initiation of protein-coding genes in eukaryotic cells is a complex process. The different steps of gene transcription, on a general level, can be summarized as follows. It starts with the binding of several transcription factors (TFs) on the "upstream" promoter that could be as far as 1 kilo base pairs (kb) upstream of the start site [6]. This, in conjunction with enhancers, silencers and insulators (all of which are stretches of DNA that could be several thousand base pairs away from the genes they control), controls the binding of a preinitiation complex, usually composed of RNA polymerase II, transcription factor IID and other general transcription factors, on the core promoter that lies approximately 100 base pairs (bp) on either side of the TSS [6]. These help in opening up the double helix, which is followed by a movement of the RNA polymerase from the 3′-end to the 5′-end on the antisense strand of the DNA. As the RNA polymerase proceeds, it assembles ribonucleotides into a strand of pre-mRNA, following certain rules of base pairing. These rules are: 1) a C on the DNA strand results in a G being inserted into the RNA strand, 2) a G on the DNA strand results in a C being inserted into the RNA strand, 3) a T on the DNA strand results in an A being inserted into the RNA strand, and 4) an A on the DNA strand results in a U (uracil) being inserted into the RNA strand. The RNA is synthesized in a 5′ to 3′ direction. The pre-mRNA undergoes a series of processing steps to be converted into an mRNA. These are the capping of the 5′-end of the pre-mRNA, the removal of introns and splicing of exons, and finally the attachment of a polyadenine tail at a special site, called the polyadenylation site, of the pre-mRNA. The transcript is cut at this point, and forms the mRNA that undergoes translation to be converted into a protein. A protein is composed of a number of amino acids. There are 20 amino acids in practice. A set of three consecutive nucleotides in an mRNA, called a codon, codes for an amino acid. As there are a total of 64 possible codons, but only 20 different amino acids, coding redundancy exists. Moreover, some of the codons do not code for any amino acid, they just signal some special event. For example, start and stop codons signal the beginning and end of translation.

It may be noted that our knowledge about the genomic transcription process is limited. For example, in vertebrates, the internal exons are small (typically about 140 nucleotides long), while the introns are much larger (some being more than 100 kb long) [6]. The mechanism by which the splicing machinery recognizes the exons from within vast stretches of introns is still not fully clear. Accurate information regarding the promoters of most genes is severely limited. There is no strong consensus sequence at the splice junctions. We still do not know how many genes are there in the human genome, and the estimate of this value is getting revised frequently. These, and several other, issues make the task of identifying genes extremely challenging.

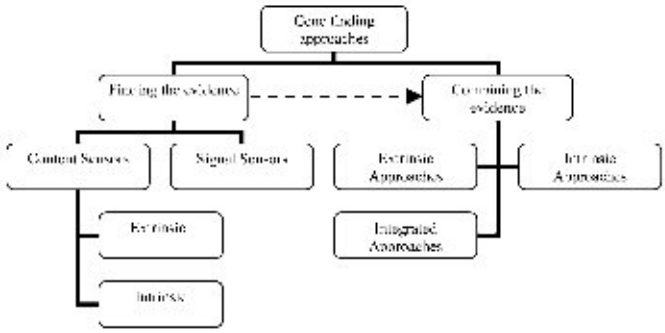

Fig. 3. Classical approaches to gene finding.

III. General Overview of the Classical Gene-Finding Approaches

As shown in Fig. 3, the classical approaches for gene finding can be roughly divided into finding the evidence for a gene and combining several evidences of genes in order to better predict the structure of a gene [12]. (As shown by dashed arrow in Fig. 3, the latter may use techniques in the former.) These are now discussed briefly.

A. Finding the Evidence

Essentially, two different types of information, viz., content sensors and signal sensors are currently used to try to identify genes in a given genomic sequence.

1) Content Sensors: Content sensors are those measures that try to classify a DNA region into different types, e.g., coding versus noncoding [12]. The existence of sufficient similarity with a biologically characterized sequence has been the main means of obtaining such a classification. Content sensors can be further classified as extrinsic and intrinsic content sensors.

a) Extrinsic content sensors: Extrinsic content sensors consider a genomic sequence region and compute its similarity to a protein or DNA sequence present in a database in order to determine whether the region is transcribed and/or coding. The basic tools for detecting this similarity between sequences are local alignment methods, ranging from the optimal Smith–Waterman algorithm [18] to fast heuristic approaches such as FASTA [19] and BLAST [20]. The obvious weakness of such extrinsic approaches is that nothing will be found if the database does not contain a sufficiently similar sequence. Even when a good similarity is found, the expansion of the regions of similarity, which should indicate exons, are not always very precise and do not enable an accurate identification of the structure of the gene. Small exons are also easily missed. Moreover, some databases may contain information of poor quality that could easily mislead these approaches. Finally, the choice of the similarity measure(s) is an important consideration itself.

b) Intrinsic content sensors: Intrinsic content sensors were basically defined for prokaryotic genomes where only protein coding regions and intergenic regions are usually

considered. Here, genes define (long) uninterrupted coding regions with no-stop codons; hence, the simplest approach for finding potential coding sequences is to look for sufficiently long open reading frames (ORFs). In contrast, in eukaryotic sequences, the translated regions may be very short, and the absence of stop codons becomes meaningless [21]. In these cases, other measures based on the content of the sequences are computed for identifying whether it is coding or not. Some typical measures are the GC content, frequency of k-mers, especially hexamers, base occurrence periodicities.

2) Signal Sensors: Signal sensors are measures that try to detect the presence of the functional sites specific to a gene [12]. The basic and natural approach to finding a signal that may represent the presence of a functional site is to search for a match with a consensus sequence (with possible variations allowed), the consensus being determined from a multiple alignment of functionally related documented sequences. This type of method is used for splice site prediction in SPLICEVIEW [22] and a logitlinear-model-based approach [23]. Positional weight matrices (PWMs) offer another representation of signals. PWMs capture the probability of appearance of a base in a particular location. In order to capture the possible dependencies between adjacent positions of a signal, one may use higher order Markov models. The so-called weight array model (WAM) is essentially an inhomogeneous higher order Markov model. It was first proposed by Zhang and Marr [24], and later, used by Salzberg [25], who applied it in the VEIL [26] and MORGAN [27] software. Genscan [28] also uses a modified WAM to model acceptor splice sites and a second-order WAM to represent branch point information.

B. Combining the Evidence to Predict Gene Structures

For a given sequence, different types of signal sensors and/or similarity-based methods can be used to predict the occurrence of genes. The most important evidences of the presence of genes are undoubtedly translation starts and stops, and splice sites since they define the boundaries of coding regions. All such evidences can be combined for predicting the complicated gene structures; this contrasts with the earlier approaches for identifying individual exons. In theory, each consistent pair of detected signals defines a potential gene region (intron, exon, or coding part of an exon). If one considers that all these potential gene regions can be used to build a gene model, the number of potential gene models grows exponentially with the number of predicted exons. In practice, this is slightly reduced by the fact that "correct" gene structures must satisfy the following set of properties [12]: there are no overlapping exons, coding exons must be frame compatible, and merging two successive coding exons will not generate an in-frame stop at the junction. This approach can further be classified into three categories, namely, extrinsic, intrinsic, and integrated approaches.

a) Extrinsic approaches: Pioneered by Gelfand *et al.* with PROCRUSTES [29], many programs based on similarity searches have emerged during the last decade. The principle of most of these programs is to combine similarity information with signal information obtained by signal sensors. This information will be used to refine the region boundaries. These programs inherit all the strengths and weaknesses of the sensors used and may, for example, fail when noncanonical splice sites are present.

b) Intrinsic approaches: Intrinsic gene finders aim at locating all the gene elements that occur in a genomic sequence, including possible partial gene structures at the border of the sequence. To efficiently deal with the exponential number of possible gene structures defined by potential signals, almost all intrinsic gene finders use dynamic programming (DP) to identify the most likely gene structures according to the evidence defined by both content and signal sensors. All such gene-modeling strategies can be formulated with a graph language [30], [31]. Such approaches are said to be exon based or signal based depending on whether a gene structure is considered to be an assembly of segments defining the coding part of the exons or by the presence of a succession of signals separated by "homogeneous" regions, respectively [32]. In theory at least, the segment assembly process can be defined as the search for an optimal path in a directed acyclic graph where vertices represent exons and edges represent compatibility between exons. The search is done using the famous Viterbi algorithm [33], which produces a most likely gene structure and can be considered as a specific instance of the older Bellman shortest path algorithm [34]. This is the approach adopted in GeneId [35], GenView [36], GAP III [37], FGENES [38], and DAGGER [39] programs.

c) Integrated approaches: In view of the added value provided by database similarities, authors are now combining both intrinsic and extrinsic approaches in recent gene predictors; in older software, updates are made to add information from homology. An important work in this regard is TWINSCAN [40], [41] program. Yeh *et al.* have extended Genscan to yield GenomeScan [42] that incorporates similarity with a protein retrieved by BLASTX or BLASTP. Genes predicted by GenomeScan have a maximum probability conditional on such similarity information. GenomeScan is, thus, able to accurately predict coding regions missed by both Genscan and BLASTX used alone. A highly integrative approach is used in the EuGene program reported in [43]. Very recently, a technique is proposed in [44] that uses statistical methods to combine the gene predictions of *ab initio* gene finders, protein sequence alignments, expressed sequence tag and cDNA alignments, splice site predictions, and other evidences.

IV. Some Classical Approaches

In this section, we present a brief overview of some classical gene-finding techniques.

A. Hidden Markov Models

A Markov model is a stochastic model, which assumes that the probability of appearance of a given base (A, T, G, or C) at a given position depends only on the k previous nucleotides (k is called the order of the Markov model). Such a model is defined by the conditional probabilities $P(X|k \text{ previous nucleotides})$, where $X =$ A, T, G, or C. In order to build a Markov model, a learning set of sequences on which these probabilities will be

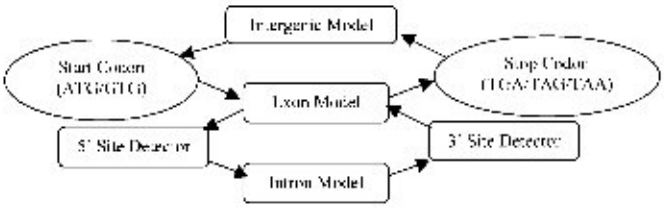

Fig. 4. Macroscopic view of the model of human genomic DNA. Noncircular nodes indicate regions of more details that have been omitted.

estimated is required. Given a sequence and a Markov model, one can then simply compute the probability that this sequence has been generated according to this model, i.e., the likelihood of the sequence, given the model [45].

Since HMMs are designed to process sequences of information, researchers in computational biology use them extensively for the analysis of DNA and protein sequences. For example, they have already been used for motif finding [46], multiple sequence alignment [47], and protein structure prediction [48]–[50]. At the present time, there is no publicly available HMM system (either commercially or public domain) that can handle the size and topological complexity of the model in its entirety needed for gene finding. Presently, the full model is decomposed into three submodels, viz., for detecting exons, introns, and intergenic regions [26]. A macroscopic view of the model's topology is shown in Fig. 4.

To train the HMM models, the exons, introns, and other noncoding regions in the training set are separated out, and three smaller models are trained separately. The intergenic model is very simple, containing two disconnected ten-state chains (representing the upstream and downstream intergenic pieces) with loops on the ends to absorb extra bases. The exon model contains 89 states, and the intron model contains only 10 states.

1) HMM Algorithms: The Viterbi and expectation maximization (EM) algorithms are used for computing with HMM during its training and testing [33], [51], [52].

a) Training the models: The EM algorithm is used to solve what is known as the learning problem in HMMs, i.e., to determine reasonable values for all the probabilities in an HMM. The model topology is fixed, and all of the output probabilities and transition probabilities [53] are initialized to random values. (If any prior estimates of these probabilities are available, it is usually beneficial to initialize them to these estimates.) Upon being presented with a set of DNA sequences, the EM algorithm reestimates all of these probabilities. Briefly, it runs each training sequence through the model λ and computes the posterior probability $P(s/\lambda)$ for each sequence s. These values are then multiplied together to produce $P(S/\lambda)$, where S represents the set of all the sequences. The reestimation procedure then adjusts all the probabilities in order to increase $P(S/\lambda)$. The data are then run through the model again and the probabilities are further refined. The process is iterated until $P(S/\lambda)$ is maximized. The EM algorithm is guaranteed to converge to a locally optimal estimate of all the probabilities in the model. Usually, it is assumed that the multiple observations in the training data are independent of each other. However, the assumption of independence may not always hold in practice. A formal treatment of HMM training without imposing the independence assumption is provided in [54].

b) Testing the models: After training, the model is ready to be used to interpret new sequences where the Viterbi algorithm is often used. It is a dynamic programming algorithm that efficiently aligns any sequence to an HMM. Given a sequence and a trained HMM, the Viterbi algorithm will find the most likely sequence of states through the model for that particular sequence. (In addition, the Viterbi algorithm computes the probability of the model producing the sequence via that path.) Although there are an exponential number of distinct paths through the model, the Viterbi algorithm finds the best one in $O(ne)$ time, where n is the length of one sequence and e is the number of edges in the path. In order to efficiently search for the best path, the Viterbi algorithm builds a data structure called a trellis, which requires $O(ne)$ storage space.

B. Dynamic Programming

The use of dynamic programming in gene finding is briefly reviewed in [8]. The dynamic programming algorithm [55] is a well-established recursive procedure for finding the optimal (e.g., minimal cost or top scoring) pathway among a series of weighted steps. GeneParser [56] that employs dynamic programming techniques uses coding measures and signal strengths to compute scores for all subintervals in the test sequence. A neural network is first used to combine the various measures into the log-likelihood ratio for each subinterval to exactly represent an intron or exon. A dynamic programming approach is then used to find the optimal combination of introns and exons. Ranked suboptimal solutions can also be generated by the program. Gelfand and Roytberg [57] have reviewed the use of dynamic programming in gene prediction, and suggested "vector dynamic programming" to combine multiple exon quality indices without the time-consuming training of a neural network. These ideas have been implemented in CASSANDRA [58], a program to predict protein-coding segments, and the experimental gene structure prediction program GREAT [59]. The GenView system [36] is again based on the prediction of splicable ORFs ranked by the strength of their splice signal and their coding potential ("in phase" hexamer measure). The best gene structure is then constructed using dynamic programming to sift through the numerous possible exon assemblies. Dynamic programming methods are also used in GRAIL II [60], GeneParser [56], FGENESH [38], and recent versions of GeneId [35]. Finally, the gene assembly program GAP III [37] also uses dynamic programming (as well as heuristics) to construct optimal gene models from the candidate exons predicted by GRAIL II.

C. Bayesian Networks

A Bayesian network [61] is a graphical model that encodes probabilistic relationships among variables of interest. When used in conjunction with statistical techniques, the graphical model has several advantages for data analysis.

1) Since the model encodes dependencies among all variables, it readily handles situations where some data entries are missing.
2) A Bayesian network can be used to learn causal relationships, and hence, can be used to gain understanding about a problem domain and to predict the consequences of intervention.
3) Since the model has both causal and probabilistic semantics, it is an ideal representation for combining prior knowledge (which often comes in causal form) and data.
4) Bayesian statistical methods in conjunction with Bayesian networks offer an efficient and principled approach for avoiding the overfitting of data.

A Bayesian network for a set of variables $V = \{V_1, \ldots, V_n\}$ consists of: 1) a network structure S that encodes a set of conditional independence assertions about variables in V and 2) a set P of local probability distributions associated with each variable. Together, these components define the joint probability distribution for V. The network structure S is a directed acyclic graph. The nodes in S are in one-to-one correspondence with the variables V. V_i denotes both the variable and its corresponding node, and Pa_i denote the parents of node V_i in S as well as the variables corresponding to those parents. The lack of possible arcs in S encodes conditional independencies. In particular, given structure S, the joint probability distribution for V is given by

$$p(V) = \Pi_{i=1}^{n} p(V_i|Pa_i).$$

The local probability distributions P are the distributions corresponding to the terms in the product of the previous equation. Consequently, the pair $(S;P)$ encodes the joint distribution $p(V)$. Methods of learning probabilities in a Bayesian network, and techniques for learning with incomplete data, are studied in detail in [62]. A Bayesian network framework for combining gene predictions from multiple systems is given in [63], where the approach adopted is that of combining the advice of multiple experts.

V. Computational Intelligence Techniques for Gene Identification

Computational intelligence is a fairly new name with no consensus (yet) on what computational intelligence exactly is. There is a widely accepted view on which areas belong to it: evolutionary computing, fuzzy computing, and neurocomputing. While some techniques within computational intelligence are often counted as artificial intelligence techniques (e.g., genetic algorithms or neural networks), there is a clear difference between these techniques and traditional, logic-based artificial intelligence techniques. In general, typical artificial intelligence techniques are top-to-bottom, where the structure of models, solutions, etc., is imposed from above. Computational intelligence techniques are generally bottom-up, where order and structure emerges from an unstructured beginning. Here, we present not only some of the computational intelligence techniques but also some other computational methods that are frequently used for gene identification.

A. Case-Based Reasoning (CBR)

Case-based reasoning (CBR) [64], [65] is a model of reasoning where the systems' expertise is embodied in a library of past cases (stored as a case base) already experienced by the system, rather than being encoded explicitly as rules, or implicitly as decision boundaries. In CBR, a problem is solved by first matching it to problems that were encountered in the past, and then retrieving one or a small set of similar cases. The retrieved cases are used to suggest a solution to the present problem, which is tested, and if necessary, revised. The present problem and its solution are updated in the case base as a new case.

An application of CBR to the gene-finding problem has been investigated in [66]. It employs a case library of nucleotide segments that have previously been categorized as coding (exon) or noncoding (intron), in order to locate the coding regions of a new DNA strand. A similarity metric for nucleotide segments is established, and the results of multiple cases are combined to categorize entire new DNA strands. For a CBR system to work effectively, it is necessary to be able to compare the case (e.g., a known exon), with the query strand of DNA in which we want exons to be identified. Costello and Wilson [67] have investigated a number of possible approaches to similarity, including longest common subsequence [68] and sequence alignment methods, but chose an edit distance method for the initial work. Given a measure of sequence similarity, it is needed to employ the case library segments in a way that will enable one to isolate regions of a sequence of DNA and point to them as potential protein-coding regions. Since library exons are likely to be much shorter than a new strand, an approach that combines many retrieved cases in order to arrive at the new solution was adopted in [69]. The CBR framework has been applied to the problem of annotating genes and the regulatory elements in their proximal promoter regions. CBR was chosen for several reasons: the problem with sparse data is nowhere more evident than in the study of patterns in DNA, such as those in the proximal promoter region, necessary for the regulation of gene expression; and it is the goal of such research to discover these mechanisms.

A database called EpoDB is described in [70], where a large amount of information available for the genes that are expressed in vertebrate red blood cells are collected and curated to provide high-quality data for sequence analysis. The goal is to create an informatics system for the study of gene expression in and functional analysis of red blood cell differentiation, a process termed erythropoiesis. Here, the genes and the database can be considered to be analogous to the cases and the case-base in CBR. More details on EpoDB can be found in [71]. A detailed survey of the applications of CBR in molecular biology, including gene identification, is provided in [72].

B. Neural Networks

Artificial neural networks (ANNs) [73] are computer algorithms based loosely on modeling the neuronal structure of natural organisms. They are stimulus-response transfer functions that accept some input and yield some output. They are typically used to learn an input–output mapping over a set of examples. In

general, if given sufficient complexity, there exists an ANN that will map every input pattern to its appropriate output pattern, so long as the input output mapping is not one-to-many. ANNs are, therefore, well suited for use as detectors and classifiers.

Multilayer perceptrons, also sometimes described as feedforward networks, are the most common architecture used in supervised learning applications (where exemplar patterns are available for training). Each computational node sums N weighted inputs, subtracts a threshold value, and passes the result through a logistic (sigmoid) function. Single perceptrons form decision regions separated by a hyperplane. If the different data classes being input are linearly separable, a hyperplane can be positioned between the classes by adjusting the weights and bias terms. If the input data are not linearly separable, a least mean square (LMS) solution is typically generated to minimize the mean square error (MSE) between the calculated output of the network and the actual desired output.

An earlier attempt at computer-aided gene recognition, such as the well-known GRAIL software, used an ANN to combine a number of coding indicators calculated within a fixed sequence window [74]. Fickett and Tung [75] noted that at the core of most gene recognition algorithm are one or more coding measures: functions that calculate, for any window of the sequence, a number or vector intended to measure the "codingness" of the sequence. Common examples of these measures include *codon usage*, *base composition vector*, etc. An exon-recognition method includes both a coding measure and a method of deciding between "coding" or "noncoding" regions for each vector. Such an approach to evolve ANNs capable of identifying coding and noncoding regions is available in [76]. The classification process using evolved ANNs proceeded as follows. A sequence of DNA was interrogated using a window of 99 nucleotides. The ANN was used to classify the nucleotide in the center of the window as either coding or noncoding. For this analysis, the neural network architecture was fixed and consisted of nine input nodes (corresponding to nine features), 14 hidden nodes, and one output node. The output decision was normalized from -1 (noncoding) to $+1$ (coding) for each position in the sequence. If the output value was less than -0.5 (or $+0.5$), it was classified as coding (or, noncoding). In evolved ANN, genetic algorithms (GAs) have been used for determining the appropriate network architecture. "Offspring" ANN architectures are created from the parent networks through random mutation. The number of layers, nodes, and the values for the associated parameters (e.g., weights and biases of a MLP, weights, biases, means, and standard deviations of a radial basis function network) are encoded in the chromosomes, and their appropriate values are evolved using GAs. The architecture is fixed to nine input nodes, 14 hidden nodes, and one output node, while the interconnections weights and the biases are evolved using GAs. The coding indicators of this system, used as the set of input features, are *Frame bias matrix*, *Fickett feature*, *coding sextuple word preferences*, *word preferences in frame 1*, *word preferences in frame 2*, *word preferences in frame 3*, *maximum word preferences in frames*, *sextuple word commonality*, and *repetitive sextuple word* [76]. A flowchart of the entire gene identification procedure is given in Fig. 5. In the postprocessing step, spikes in the output vector

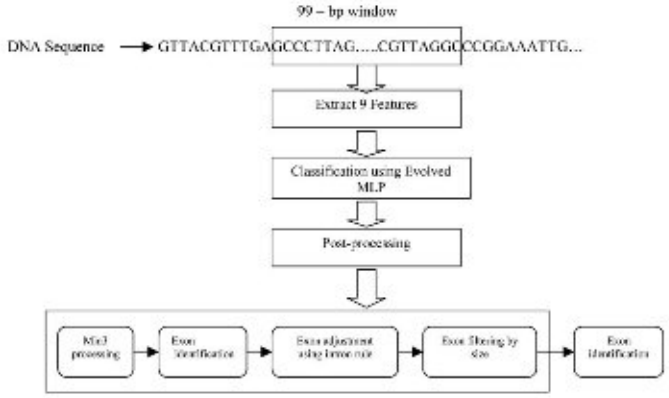

Fig. 5. Flowchart of gene identification technique by evolved neural networks.

are first removed by replacing the value of a central nucleotide to the minimum over its own value and those of its left and right neighbors (the Min3 processing step). In the exon-identification step, a set of continuous coding nucleotides was predicted as a putative exon, and its start and end positions were noted. Across each such exon, a window of 50 nucleotides was considered, and statistical measures for intron filtering (based on the frequency of occurrence of nucleotides at the intron/exon boundaries) was used to adjust the location of the putative exon to a more appropriate location within the window. Finally, the domain knowledge that a majority of exons in human DNA was more than 15 nucleotides long was used to reject all predicted exons of length less than 15.

ANNs combined with a rule-based system has been used for splice site prediction in human *Arabidopsis thaliana* by using a joint prediction scheme where the prediction of transition regions between introns and exons regulates a cutoff level for local splice site assignment [77]. This is followed by a rule-based refinement that uses splice site confidence values, prediction scores, coding context, and distances between potential splice sites. This has been further improved by the incorporation of information regarding the branch point consensus sequence found by a noncircular approach using HMM [78].

The application of a time-delay-architecture-based feedforward neural network for analysis of the *Drosophilia melanogaster* genome has been presented in [79]. It was tested on the Adh region of 2.9 Mbases of the Drosophila genome, where it was found to provide a recognition rate of 75% with a false positive rate of 1/547 bases. Recently, a neural-network-based multiclassifier system has been proposed for gene identification in *E. coli* by locating the promoters of the genes [80]. A set of 324 known *E. coli* promoters and 429 known nonpromoters was coded using four different coding techniques, and four different neural classifiers were trained on each set. The final classification was an aggregate of the individual classifications obtained using a variant of the logarithmic opinion pool method. Some other applications of neural networks for gene finding are also discussed in [81].

An attempt to employ ANNs to predict the regions that overlap with the first exon of a gene or lies in its close

proximity was reported in [82] and [83]. Dragon gene start finder (DGSF) [82], [83] combines three systems to achieve this goal. First, it uses a promoter finder system to estimate the transcription start site (TSS). Another system estimates the presence of CpG islands (which are stretches of DNA containing a significantly high frequency of CG sequence, often located around the promoters of genes that perform general cell functions) on the DNA strand. Several signals are extracted from the predicted TSS and CpG islands. The data are then normalized and transformed using principal component analysis. Thereafter, a four-layer neural network is used to make predictions whether the combination of the CpG island and the predicted transcription start site indicates the presence of gene starts.

C. Decision Trees

Decision tree algorithms are important, well-established machine learning techniques that have been used for a wide range of applications, especially for classification problems [84]–[87]. Decision trees have been found to accurately distinguish between coding and noncoding DNA for sequences as short as 54 bp [88]. In that study, the task of distinguishing between subsequences that are either entirely encoding or entirely noncoding was addressed. MORGAN [27], an integrated system for finding genes, uses a variety of techniques, the most distinctive of which is a decision tree algorithm. It uses an OC1 decision tree system developed in [86] for solving the problem of discriminating coding and noncoding DNA. The optimal segmentation is dependent on a separate scoring function that takes a subsequence and assigns to it a score reflecting the probability that the sequence is an exon. The scoring functions in MORGAN are sets of decision trees that are combined to give a probability estimate. The internal nodes of a decision tree are property values that are tested for each subsequence passed to the tree. Properties can be various coding measures (e.g., hexamer frequency) or signal strengths. The bottom nodes (leaves) of the tree contain class labels to be finally associated with the subsequence. Once classified, the various components are assembled into an optimal gene model using a dynamic programming approach. Another well-known gene finder, developed specifically for eukaryotes, which uses decision trees hybridized with interpolated Markov model (IMM) and dynamic programming, is GlimmerM [89]. It uses dynamic programming to consider all possible exons for inclusion in a gene model, and chooses the best of all these combinations. The decision about what gene model is best is a combination of the strength of the splice sites and the scores of the exons produced by IMM. A scoring function based on decision trees is built in order to estimate the probability that a DNA subsequence is coding or not. Five types of subsequences are evaluated: introns, initial exons, internal exons, final exons, and single exons. The probabilities obtained with the decision trees are averaged to produce a smoothed estimate of the probability that the given subsequence is of a certain type. A gene model is only accepted if the IMM score over all coding sequences exceeds a fixed threshold.

D. Genetic Algorithms

A genetic or evolutionary algorithm, first proposed by J. H. Holland, applies the principles of evolution found in nature to finding an optimal solution to an optimization problem [90], [91]. In a GA the problem is encoded in a series of bit strings that are manipulated by the algorithm; in an evolutionary algorithm, the decision variables and problem functions are used directly. The basic GA can be outlined in a very simple way.

1) *Start*: Generate random population of n chromosomes (suitable solutions for the problem).
2) *Fitness*: Evaluate the fitness $f(x)$ of each chromosome x in the population.
3) *New population*: Create a new population by repeating following steps until the new population is complete.
    a) *Selection*: Select two parent chromosomes according to their fitness.
    b) *Crossover*: With a probability crossover, the parents to form new offspring (children).
    c) *Mutation*: With a probability mutate new offspring at each position.
    d) *Fitness*: Evaluate the fitness $f(x)$ of each chromosome x in the new population.
4) *Test*: If the end condition is satisfied, then stop and return the best solution in the current population.
5) *Loop*: Go to step 3.

For the gene-mapping problem, Gunnels *et al*. [92] compared GA with simulated annealing (SA) [93], and found that the GA-based method always converges to a good solution fast, since it is able to take advantage of the extra information to construct good local maps that can then be used to construct good global maps. Using a GA, Kel *et al*. [94] designed the 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. A method for recognizing promoter regions of eukaryotic genes using a GA with an application on *Drosophila melanogaster* is described in [95]. Its novelty lies in realizing the genetic algorithm to search for an optimal partition of a promoter region into local nonoverlapping fragments and selection of the most significant dinucleotide frequencies for the aforesaid fragments.

The evolved ANN software, mentioned in Section V-B, mainly uses neural network techniques for gene finding. It also makes use of GAs to decide about the network architecture and its other parameters. The fitness of the hidden layers is computed using a GA. ANN architectures are created from the parent networks through random mutation, and variations are applied to the number of layers and nodes and other associated parameters. A comparative study between the performance of evolved ANN and GRAIL, in terms of correlation coefficient (CC), sensitivity, specificity, fraction of exactly predicted exons (including start and stop codons), and fraction of predicted exons that overlap with actual exons, is shown in Table I. Let TP, TN, FP, and FN denote the true positives (number of coding nucleotides detected as coding), true negatives (number of noncoding nucleotides

TABLE I
PERFORMANCE OF GRAIL EVALUATED RELATIVE TO THE BEST-EVOLVED ANN ON TWO DIFFERENT DATA SETS [74]

	Test set I		Test set II	
Program	GRAIL	Evolved ANN	GRAIL	Evolved ANN
Correlation coefficient	0.65	0.46	0.52	0.45
Sensitivity	0.56	0.24	0.48	0.62
Specificity	0.85	0.38	0.88	0.12
Exons wrong	0.00	0.00	0.00	0.00
Exons overlapped	0.70	0.83	0.50	0.67

detected as noncoding), false positives (number of noncoding nucleotides detected as coding), and false negatives (number of coding nucleotides detected as noncoding). Then

$$\text{specificity} = \text{TP}/(\text{TP}+\text{FP}) \quad \text{sensitivity} = \text{TP}/(\text{TP}+\text{FN})$$

$$\text{CC} = \frac{(\text{TP}\times\text{TN})-(\text{FP}\times\text{FN})}{\sqrt{(\text{TP}+\text{FN})\times(\text{TN}+\text{FP})\times(\text{TP}+\text{FP})\times(\text{TN}+\text{FN})}}.$$

All test sequences were taken from human genes and are available in [74]. Test set I contained sequences used in the testing of GRAIL and GeneID, while test set II contained genes with complete protein coding regions and at least two exons. These sequences were without pseudogenes, multiple coding sequence fields, putative coding sequence fields, or alternative splicing forms, and were first used in [96].

E. Comparative Performance of Some Gene Finders

A detailed comparative study of a number of computer programs for the prediction of gene structure in genomic sequences is provided in [8] and [9]. The programs were tested on a set of 524 vertebrate sequences ranging in length from about 420 to 21 720 bp [97]. The sensitivity (in percent) and specificity (in percent) of GRAIL II, a very popular program, were found to be 51% and 57%, respectively [8]. An advancement of GRAIL II/GAP was actually no better than GRAIL II in terms of sensitivity, specificity, and missed exons (percentage of actual exons not overlapping any predicted exon). But in terms of wrong exons (percentage of predicted exons not overlapping any actual exon), GRAIL II/GAP (10%) was better than GRAIL (28%). The performances of GeneID and GeneParser were not better than any other programs in any respect. MORGAN's performance was at par with GRAIL II and GAP. The best overall performance was shown by GENSCAN (gene structure prediction) and MZEF (individual exon finder) [98]. Both have high sensitivity (78%), but specificity of MZEF is slightly better (86%) than that of GENSCAN (81%). Scores, in terms of missed exons and wrong exons, are slightly better in GENSCAN than in MZEF. The comparison of UNVEIL, a relatively recent HMM-based gene finder, with some other methods including GlimmerM and also GENSCAN on 300 *Arabidopsis thaliana* full-length cDNAs reveals that it performs well in terms of nucleotide accuracy, exon accuracy, and whole-gene accuracy [99]. GRAIL and evolved ANN both are based on neural network methodologies—sensitivity of evolved ANN program is found to be much higher than that of GRAIL, but specificity and correlation coefficient of GRAIL is greater than that of evolved ANN (as can be observed from Table I for both the test sets). Overall, GENSCAN and MZEF perform better than any other program. A more detailed study of the comparative performance of different methods can be found in [8], though a caveat is also added that test sets vary in size, and complexity or (G + C) composition, thus making the results difficult to interpret. It may be noted in this context that the predicted accuracies of the gene finders have been found to vary in different investigations, sometimes being substantially lower than those obtained with more limited test sets [8].

VI. DISCUSSION AND CONCLUSION

The identification of genes is an important problem in bioinformatics. Several methods of gene identification, e.g., those based on HMM and dynamic programming, have been developed in the past. Given the difficulty of the problem, computational-intelligence-based methods have also been applied in recent times because of their robustness and ability to handle noisy and incomplete/uncertain data. This paper provides a comprehensive review of the different methods of gene identification, with special emphasis on computational intelligence techniques. An extensive bibliography along with a list of important URLs are also provided in this regard. Table II lists various gene-finding programs and their references, and the kind of approaches used by them.

In general, training sets used for various methods of gene finding consists of almost an equal number of coding and noncoding nucleotides. But, in reality, it has been found that only about 2% of human DNA is coding, and the remainder is noncoding. The use of equal proportion of coding and noncoding data may lead to a biased system resulting in an increased sensitivity toward the coding data and decreased capability of classifying noncoding data. As a result, the system might often classify noncoding nucleotides incorrectly as coding ones, thereby leading to an increase in the false positive count. Therefore, the ratio of coding to noncoding data in the data set should be altered in order to remove such bias.

Several issues, both biological and database related, make the problem of eukaryotic gene finding extremely difficult [12]. In the current practice, the promoter is generally viewed as appearing in the intergenic region, immediately upstream of the gene, and not overlapping with it. This is a simplification of reality. Often, the genes are very long, e.g., the largest human gene, the dystrophin gene, is composed of 79 exons spanning nearly 2.3 MB. Moreover, there may be situations where the introns are very long, e.g., in the human dystrophin gene, some introns are more than 100 kB long, and more than 99% of the gene is composed of introns. Some genes are characterized by very short exons that may be easily missed, especially if bordered by large introns. The more difficult cases are those where the length of a coding exon is a multiple of three (typically 3, 6, or 9 bp long), because missing such exons will not cause a problem in the exon assembly, as they do not introduce any change in the frame. Moreover, it has been found that a significant percentage

TABLE II
VARIOUS GENE-FINDING PROGRAMS/RESOURCES

Programs	References	Approaches used	Comments (along with URLs wherever available)
Genefinder	[100]	Maximum likelihood estimation	A species specific gene prediction tool
Geneid	[35]	Dynamic Programming	Uses Viterbi algorithm to search for optimal path
GeneView	[36]	Dynamic Programming	Uses Viterbi algorithm to search for optimal path
FGENES/FGENESH	[38]	Dynamic Programming	Uses Viterbi algorithm to search for optimal path
GlimmerM	[89]	Dynamic Programming, Interpolated Markov Model (IMM) and Decision Trees	Considers all combinations of possible exons for inclusion in a gene model http://www.tigr.org/software/glimmerm/
TWINSCAN	[40]	Integrated approach	Combines alignment based approach with HMM
EuGene	[43]	Integrated approach	Combines the information of various software and user information using a weighted directed acyclic graph
CASSANDRA	[58]	Dynamic Programming	
GAPII	[37]	Dynamic Programming	Uses Viterbi algorithm to search for optimal path http://www.gcg.com/
GRAIL/GRAIL II	[60]	Neural Network	GRAIL II is the latest advancement on GRAIL
GenomeScan	[42]	Integrated approaches in database similarities	Extension of Genscan
GeneMark	[101]	Non-homogeneous Markov chain model for protein coding DNA and homogeneous Markov chain model for non-coding DNA	The programs of the Genmark family that is not based on the presence of homologues in the database
MORGAN	[27]	Decision Trees and Dynamic Programming	Also uses fixed order Markov models to filter out potential signals http://www.tigr.org/~salzberg/morgan.html
Genscan	[28]	HMM	Predicts complete gene structure. Uses modified Weight Array Model http://CCR-081.mit.edu/GENSCAN.html
MZEF	[98]	Individual exon finder	http://argon.cshl.org/genefinder/
PROCRUSTES	[29]	Splice alignment argument	Provides 99% accurate recognition of mammalian gene if a related gene from another mammalian species is known http://www.bro.use.edu/software/procrustes/index.html
VEIL	[26]	HMM	Uses Weight Array Model http://www.tigr.org/~salzberg/veil.html
UNVEIL	[98]	HMM	Loosely based on VEIL http://www.tigr.org/software/Unveil/index.shtml
Genie	[102]	Generalized HMM	http://www.fruitfly.org/seq_tools/genie.html
GeneParser	[56]	Dynamic Programming	Program for identification of protein coding regions in genomic DNA http://beagle.colorado.edu/~eesnyder/GeneParser.html
SPLICEVIEW	[22]	Signal Sensor methods	For splicing signal prediction
Splice Predictor	[23]	Signal Sensor methods based on logitlinear models	For splicing signal prediction
DAGGER	[39]	DAG shortest paths	Uses Viterbi algorithm to search for optimal path
GREAT	[59]	Dynamic Programming	
AAT	[103]	Integrated approaches in database similarities	Analyzing and annotating genomic sequences http://www.tigr.org/software/
Evolved ANN	[74]	Neural network and genetic algorithm	Use of Genetic Algorithm on GRAIL

(35%) of human genes are alternatively spliced that makes the problem further complicated [104]. A recent related study on human genome is reported in [105] that states that many of the splice variants could be a result of aberrant rather than regulated splicing.

The arrangement of genes in genomes is also prone to exceptions. Although usually separated by an intergenic region, there are examples of genes nested within each other and overlapping genes [106], [107]. The presence of pseudogenes (nonfunctional sequences resembling real genes), which are distributed in numerous copies throughout the genome, further complicates the identification of true protein-coding genes. A recent study on the sequence and biological annotation of human chromosome 1 shows that the ratio of the number of pseudogenes to the number of true genes is significantly high, and that the coding sequences often overlap [108]. Polycistronic gene arrangement presents an added level of complexity for gene finding. These have been found for snoRNA genes in plants [109], as well as in mammals [110].

Another major concern is that existing gene finders nearly always rely on already known sequences, either in the form of training sets or of databases against which homology searches are performed, and are therefore, prone to errors in the databases. As an example, some sequences stored in databases may contain either sequencing errors or simply errors made when editing the sequence, resulting in the introduction of artificial frameshifts. Such frameshifts greatly increase the difficulty of the computational gene-finding problem by producing erroneous statistics

and masking true solutions [12]. A gene-finding method described in [111] can handle this situation. A way of tackling this problem may be to adopt a hierarchical approach. In the first level, one of these "conservative" methods can be adopted to make a prediction. Thereafter, when an unusual situation occurs, the possibility of noncanonical cases should be examined in the second level, allowing a reconsideration of the prediction proposed in the first level.

It has already been revealed in past investigations that there may be more than one unique gene model [112]. Moreover, instead of using a single large data set for training a model, it may be beneficial to consider several partitioned data sets. Training a single model on the entire data may be confusing, leading to poor generalization capabilities. In contrast, training several models on small, homogeneous partitions of the data may provide better stability of the models. The issues over here are how to partition the data sets in homogeneous groups, and how to combine the predictions of the several models for annotating unknown sequences. For partitioning, an approach like the one used in [113] can be adopted. For combining the predictions, a possible approach could be to find out the maximum scoring over all the models, and then, to make the corresponding hypothesis. Finally, there is a great need for "clean" sequence databases, i.e., for databases that are not redundant, contain reliable and relevant annotations, and provide all necessary links to further data. For example, a clean data set of verified splice sites for the human EST sequences, and the standards used for the cleanup procedure are reported in [114].

With the huge amount of expressed sequence tag (EST) and cDNA sequences now available, programs based on the existence of homology with such expressed sequences are playing an ever increasingly crucial role in current genome annotations, at least for genes for which expression can be shown. Even in the case of genes that are scarcely or tissue specifically expressed, complementary information can be provided by similarity between genomic sequences. Consequently, a great deal of effort is now expended on trying to gather information from genome comparisons. This is particularly true in the case of the human genome annotation process, where the availability of other complete vertebrate genomes, such as those of mouse and fish, is a great advantage. With many genome sequencing projects currently under way, and although there are still problems to be solved, the comparative genome approach seems to be a very promising approach not only in the field of gene prediction but also for the identification of regulatory sequences and the deciphering of the so-called junk DNA. The latter has been largely ignored until now; yet much may be expected to be learnt from its analysis. At present, more interest is devoted to noncoding RNAs, and this probably stands among the main future directions.

ACKNOWLEDGMENT

The authors gratefully acknowledge the informative comments of the reviewers that helped in improving the quality of the manuscript.

REFERENCES

[1] P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*. Cambridge, MA: MIT Press, 1998.
[2] A. M. Campbell and L. J. Heyer, *Discovering Genomics, Proteomics & Bioinformatics*. Singapore: Pearson Education, 2004.
[3] (2007). [Online]. Available: http://www.ornl.gov/sci/techresources/Human_Genome/project/progress.shtml
[4] L. D. Stein, "Human genome: End of the beginning," *Nature*, vol. 431, pp. 915–916, 2004.
[5] E. V. Koonin and M. Y. Galperin, *Sequence–Evolution–Function: Computational Approaches in Comparative Genomics*. Boston, MA: Kluwer Academic, 2002.
[6] M. Q. Zhang, "Computational prediction of eukaryotic protein-coding genes," *Nat. Rev. Genet.*, vol. 3, pp. 698–710, 2002.
[7] G. D. Stormo, "Gene-finding approaches for eukaryotes," *Genome Res.*, vol. 10, no. 4, pp. 394–397, 2000.
[8] J. M. Claverie, "Computational methods for the identification of genes in vertebrate genomic sequences," *Hum. Mol. Gene.*, vol. 6, pp. 1735–1744, 1997.
[9] M. Burset and R. Guigo, "Evaluation of gene structure prediction programs," *Genomics*, vol. 34, pp. 353–367, 1996.
[10] M. S. Gelfand, "Prediction of function in DNA sequence analysis," *J. Comput. Biol.*, vol. 2, no. 1, pp. 87–115, 1995.
[11] D. Haussler, Computational genefinding. vol. 23, no. 1, pp. 12–15, 1998. Available: http://www.cse.ucsc.edu/~haussler/grpaper.pdf. A short version of this paper appeared in *Trends in Biochemical Sci., Suppl. Guide to Bioinformatics*.
[12] C. Mathé, M.-F. Sagot, T. Schiex, and P. Rouzé, "Current methods of gene prediction, their strengths and weaknesses," *Nucleic Acids Res.*, vol. 30, no. 19, pp. 4103–4117, 2002.
[13] W. Li (2007). A bibliography on computational gene finding. [Online]. Available: http://www.nslij-genetics.org/gene/
[14] M. S. Gelfand, FANS-REF: Functional analysis of nucleotide sequences reference data bank. [Online]. Available: http://www-hto.usc.edu/software/procrustes/fans_ref/.
[15] B. Rost, "PHD: Predicting one-dimensional protein structure by profile based neural networks," *Methods Enzymol.*, vol. 266, pp. 525–539, 1996.
[16] J. C. Setubal and J. Meidanis, *Introduction to Computational Molecular Biology*. Boston, MA: PWS, 1996.
[17] N. C. Jones and P. A. Pevzner, *An Introduction to Bioinformatics Algorithms*. San Diego, MA: MIT Press, 2004.
[18] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, pp. 195–197, 1981.
[19] W. R. Pearson and D. J. Lipman, "Improved tools for biological sequence comparison," *Proc. Natl. Acad. Sci. USA*, vol. 85, pp. 2444–2448, 1988.
[20] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *J. Mol. Biol.*, vol. 215, pp. 403–410, 1990.
[21] J. W. Fickett, "ORFs and genes: How strong a connection?" *J. Comput. Biol.*, vol. 2, no. 1, pp. 117–123, 1995.
[22] I. B. Rogozin and L. Milanesi, "Analysis of donor splice signals in different organisms," *J. Mol. Evol.*, vol. 45, pp. 50–59, 1997.
[23] J. Kleffe, K. Hermann, W. Vahrson, B. Witting, and V. Brendel, "Logitlinear models for the prediction of splice sites in plant pre-mRNA sequences," *Nucleic Acids Res.*, vol. 24, no. 23, pp. 4709–4718, 1996.
[24] M. Q. Zhang and T. G. Marr, "A weight array method for splicing signal analysis," *Comput. Appl. Biosci.*, vol. 9, pp. 499–509, 1993.
[25] S. L. Salzberg, "A method for identifying splice sites and translational start sites in eukaryotic mRNA," *Comput. Appl. Biosci.*, vol. 13, pp. 365–376, 1997.
[26] J. Henderson, S. Salzberg, and K. H. Fasman, "Finding genes in DNA with a hidden Markov model," *J. Comput. Biol.*, vol. 4, pp. 127–141, 1997.
[27] S. Salzberg, A. Delcher, K. H. Fasman, and J. Henderson, "A decision tree system for finding genes in DNA," *J. Comput. Biol.*, vol. 5, no. 4, pp. 667–680, 1998.
[28] C. Burge and S. Karlin, "Prediction of complete gene structures in human genomic DNA," *J. Mol. Biol.*, vol. 268, pp. 78–94, 1997.
[29] M. S. Gelfand, A. A. Mironov, and P. A. Pevzner, "Gene recognition via spliced sequence alignment," *Proc. Natl. Acad. Sci. USA*, vol. 93, no. 17, pp. 9061–9066, 1996.
[30] M. A. Roytberg, T. V. Astakhova, and M. S. Gelfand, "Combinatorial approaches to gene recognition," *Comput. Chem.*, vol. 21, pp. 229–235, 1997.
[31] C. Grasso, B. Modrek, Y. Xing, and C. Lee, "Genome-wide detection of alternative splicing in expressed sequences using partial order multiple

sequence alignment graphs," in *Proc. Pacific Symp. Biocomput.*, 2004, pp. 29–41.
[32] R. Guigo, "Assembling genes from predicted exons in linear time with dynamic programming," *J. Comput. Biol.*, vol. 5, pp. 681–702, 1998.
[33] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 2, pp. 260–269, Apr. 1967.
[34] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.
[35] R. Guigo, S. Knudsen, N. Drake, and T. Smith, "Prediction of gene structure," *J. Mol. Biol.*, vol. 226, pp. 141–157, 1992.
[36] L. Milanesi, N. A. Kolchanov, I. B. Rogozin, I. V. Ischenko, A. E. Kel, Y. L. Orlov, M. P. Ponomarenko, and P. Vezzoni, "GenView: A computing tool for protein-coding regions prediction in nucleotide sequences," in *Proc. 2nd Int. Conf. Bioinformatics, Supercomput. Complex Genome Anal.*, H. A. Lim, J. W. Fickett, C. R. Cantor, and R. J. Robbins, Eds. Singapore, 1993, pp. 573–588.
[37] Y. Xu, R. J. Mural, and E. C. Uberbaker, "Constructing gene models from accurately predicted exons: An application of dynamic programming," *Comput. Appl. Biosci.*, vol. 10, pp. 613–623, 1994.
[38] A. A. Salamov and V. V. Solovyev, "Ab initio gene finding in *Drosophila* genomic DNA," *Genome Res.*, vol. 10, no. 4, pp. 516–522, 2000.
[39] J. S. Chuang and D. Roth, "Gene recognition based on DAG shortest paths," *Bioinformatics*, vol. 1, pp. 1–9, 2001.
[40] I. Korf, P. Flicek, D. Duan, and M. R. Brent, "Integrating genomic homology into gene structure prediction," *Bioinformatics*, vol. 17, no. 1, pp. 140–148, 2001.
[41] A. E. Tenney, R. H. Brown, C. Vaske, J. K. Lodge, T. L. Doering, and M. R. Brent, "Gene prediction and verification in a compact genome with numerous small introns," *Genome Res.*, vol. 14, pp. 2330–2335, 2004. [Online]. Available: http://genes. cs.wustl.edu/.
[42] R.-F. Yeh, L. P. Lim, and C. B. Burge, "Computational inference of homologous gene structures in the human genome," *Genome Res.*, vol. 11, no. 5, pp. 803–816, 2001.
[43] T. Schiex, A. Moisan, L. Duret, and P. Rouze, "EuGene: An eukaryotic gene finder that combines several sources of evidence," in *Lecture Notes in Computational Science*, vol. 2066, New York: Springer-Verlag, 2000.
[44] J. E. Allen, M. Pertea, and S. L. Salzberg, "Computational gene prediction using multiple sources of evidence," *Genome Res.*, vol. 14, pp. 142–148, 2004.
[45] T. Koski, *Hidden Markov Models for Bioinformatics*. New York: Kluwer Academic, 2002.
[46] P. Bucher, K. Karplus, N. Moeri, and K. Hoffman, "A flexible motif search technique based on generalized profiles," *Comput. Chem.*, vol. 20, no. 1, pp. 3–24, 1996.
[47] E. L. L. Sonnhammer, S. R. Eddy, E. Birney, A. Bateman, and R. Durbin, "PFAM: Multiple sequence alignments and HMM-profiles of protein domains," *Nucleic Acids Res.*, vol. 26, no. 1, pp. 320–322, 1998.
[48] V. D. Francesco, P. J. Munson, and J. Garnier, "FORESST: Fold recognition from secondary structure predictions of proteins," *Bioinformatics*, vol. 15, no. 2, pp. 131–140, 1999.
[49] G. Pollastri, P. Baldi, A. Vullo, and P. Frasconi, "Prediction of protein topologies using GIOHMMs and GRNNs," in *Advances in Neural Information Processing Systems*, vol. 15, Cambridge, MA: MIT Press, 2003.
[50] K. Karplus, R. Karchin, J. Draper, J. Casper, Y. Mandel-Gutfreund, M. Diekhans, and R. Hughey, "Combining local-structure, fold-recognition, and new fold methods for protein structure prediction," *Proteins*, vol. 53, no. 6, pp. 491–496, 2003.
[51] S. Cawley, A. Wirth, and T. Speed, "Phat—A gene finding program for *Plasmodium falciparum*," *Mol. Biochem. Parasitol.*, vol. 118, no. 2, pp. 167–174, 2001.
[52] J. A. Bilmes, *A Gentle Tutorial of the EM Algorithm and Its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*, Tech. Rep. No. ICSI-TR-97–021, Int. Comput. Sci. Inst. Berkeley CA, USA, 1998.
[53] L. A. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.
[54] X. Li, M. Parizeau, and R. Plamondon, "Training hidden Markov models with multiple observations-a combinatorial method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 4, pp. 371–377, Apr. 2000.
[55] M. A. Trick, "A tutorial on dynamic programming," *Mini V*, 1997, [Online]. Available: http://mat.gsia.cmu.edu/classes/dynamic/dynamic.html
[56] E. Snyder and G. Stormo, "Identification of protein coding regions in genomic DNA," *J. Mol. Biol.*, vol. 248, pp. 1–18, 1995.
[57] M. S. Gelfand and M. A. Roytberg, "Prediction of the exon–intron structure by a dynamic programming approach," *Biosystems*, vol. 30, pp. 173–182, 1993.
[58] M. S. Gelfand, T. V. Astakhova, and M. A. Roytberg, "An algorithm for highly specific recognition of protein-coding regions," *Genome Inf.*, vol. 7, pp. 82–87, 1996.
[59] M. S. Gelfand, L. I. Podolsky, T. V. Astakhova, and M. A. Roytberg, "Recognition of genes in human DNA sequences," *J. Comput. Biol.*, vol. 3, no. 2, pp. 223–234, 1996.
[60] Y. Xu, J. R. Einstein, M. Shah, and E. C. Uberbacher, "An improved system for exon recognition and gene modeling in human DNA sequences," in *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 1994, pp. 376–383.
[61] R. E. Neapolitan, *Learning Bayesian Networks*. Upper Saddle River, NJ: Prentice-Hall, 2003.
[62] D. Heckerman, D. Geiger, and D. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," *Mach. Learn.*, vol. 20, pp. 197–243, 1995.
[63] V. Pavlović, A. Garg, and S. Kasif, "A Bayesian framework for combining gene predictions," *Bioinformatics*, vol. 18, no. 1, pp. 19–27, 2002.
[64] J. Kolodner, *Case-Based Reasoning*. San Francisco, CA: Morgan Kaufmann, 1993.
[65] I. Watson, *Applying Case-Based Reasoning: Techniques for Enterprise Systems*. San Francisco, CA: Morgan Kaufman, 1997.
[66] C. G. Overton and J. Haas, "Case-based reasoning driven gene annotation," in *Computational Methods in Molecular Biology*, S. Salzberg, D. Searls, and S. Kasif, Eds. Amsterdam, The Netherlands: Elsevier, 1998, pp. 65–86.
[67] E. Costello and D. C. Wilson, "A case-based approach to gene finding," in *Proc. Workshop CBR Health Sci.* ICCBR'03, pp. 19–28.
[68] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and S. Clifford, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.
[69] A. Ram and A. Francis, "Multi-plan retrieval and adaptation in an experience-based agent," in *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, D. Leake, Ed. CA, AAAI Press, Menlo Park, CA, 1996.
[70] C. J. Stoeckert, Jr., F. Salas, B. Brunak, and G. C. Overton, "EpoDB: A prototype database for the analysis of genes expressed during vertebrate erythropoiesis," *Nucleic Acids Res.*, vol. 27, no. 1, pp. 200–203, 1999.
[71] (1999). [Online]. Available: http://www.cbil.upenn.edu/EpoDB/
[72] I. Jurisica and J. I. Glasgow, "Applications of case-based reasoning in molecular biology," *AI Mag.*, vol. 25, no. 1, pp. 85–96, 2004.
[73] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Singapore: Pearson Education, 2002.
[74] E. C. Uberbacher, Y. Xu, and R. J. Mural, "Discovering and understanding genes in human DNA sequence using GRAIL," *Methods Enzymol*, vol. 266, pp. 259–281, 1996.
[75] J. W. Fickett and C.-S. Tung, "Assessment of protein coding measures," *Nucleic Acids Res.*, vol. 20, pp. 6441–6450, 1992.
[76] G. B. Fogel, K. Chellapilla, and D. W. Corne, "Identification of coding regions in DNA sequences using evolved neural networks," in *Evolutionary Computation in Bioinformatics*, G. B. Fogel and D. W. Corne, Eds. San Fransisco, CA: Morgan Kaufmann, 2002, pp. 195–218.
[77] S. M. Hebsgaard, P. G. Korning, N. Tolstrup, J. Engelbrecht, P. Rouze, and S. Brunak, "Splice site prediction in *Arabidopsis thaliana* pre mRNA by combining local and global sequence information," *Nucleic Acids Res.*, vol. 24, no. 17, pp. 3439–3452, 1996.
[78] N. Tolstrup, P. Rouze, and S. Brunak, "A branch point consensus from *Arabidopsis* found by non-circular analysis allows for better prediction of acceptor sites," *Nucleic Acids Res.*, vol. 25, no. 15, pp. 3159–3163, 1997.
[79] M. G. Reese, "Application of a time-delay neural network to promoter annotation in the *Drosophila melanogaster* genome," *Comput. Chem.*, vol. 26, no. 1, pp. 51–56, 2001.
[80] R. Ranawana and V. Palade, "A neural network based multi-classifier system for gene identification in DNA sequences," *Neural Comput. Appl.*, vol. 14, no. 2, pp. 122–131, 2005.
[81] A. Sherriff and J. Ott, "Applications of neural networks for gene finding," *Adv. Genet.*, vol. 42, pp. 287–297, 2001.
[82] V. B. Bajic and S. Hong Seah, "Dragon gene start finder: An advanced system for finding approximate locations of the start of gene transcriptional units," *Genome Res.*, vol. 13, pp. 1923–1929, 2003.
[83] V. B. Bajic and S. Hong Seah, "Dragon gene start finder identifies approximate locations of the 5′ ends of genes," *Nucleic Acids Res.*, vol. 31, no. 13, pp. 3560–3563, 2003.

[84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth Int., 1984.
[85] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Los Altos, CA: Morgan Kaufmann, 1993.
[86] D. Heath, S. Kasif, and S. Salzberg, "Induction of oblique decision trees," in *Proc. 13th Intl. Joint Conf. Artif. Intell.*, 1993, pp. 1002–1007.
[87] S. K. Murthy, S. Kasif, and S. Salzberg, "A system for the induction of oblique decision trees," *J. Artif. Intell. Res.*, vol. 2, pp. 1–33, 1994.
[88] S. L. Salzberg, "Locating protein coding regions in human DNA using a decision tree algorithm," *J. Comput. Biol.*, vol. 2, no. 3, pp. 473–485, 1995.
[89] S. L. Salzberg, M. Pertea, A. L. Delcher, M. J. Gardner, and H. Tettelin, "Interpolated Markov models for eukaryotic gene finding," *Genomics*, vol. 59, pp. 24–31, 1999.
[90] J. H. Holland, *Adaptation in Natural and Artificial System*. Ann Arbor, MI: Univ. Michigan Press, 1975.
[91] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press, 1998.
[92] J. Gunnels, P. Cull, and J. L. Holloway, "Genetic algorithms and simulated annealing for gene mapping," in *Proc. 1st IEEE Conf. Evol. Comp*, Piscataway, NJ: IEEE Press, 1994, pp. 385–390.
[93] E. H. Aarts, H. M. Korst, and P. J. van Laarhoven, "Simulated annealing," in *Local Search in Combinatorial Optimization*, E. H. Aarts and J. K. Lenstra, Eds. Hoboken, NJ: Wiley, 1997, pp. 121–136.
[94] 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.
[95] 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.
[96] E. E. Snyder and G. D. Stormo, "Identifying genes in genomic DNA sequences," in *Nucleic Acid and Protein Sequence Analysis: A Practical Approach*, M. J. Bishop and C. J. Rawlings, Eds. Oxford, U.K.: Oxford Univ. Press, 1995, pp. 209–224.
[97] (2007). [Online]. Available: http://genome.imim.es/datasets/genomics96/#SEQS
[98] M. Q. Zhang, "Using MZEF to find internal coding exons," in *Current Protocols in Bioinformatics*, A. D. Baxevanis and D. B. Davison, Eds. Hoboken, NJ: Wiley, 2003, pp. 1–18.
[99] W. H. Majoros, M. Pertea, C. Antonescu, and S. L. Salzberg, "GlimmerM, exonomy, and unveil: Three ab initio eukaryotic genefinders," *Nucleic Acids Res.*, vol. 31, no. 13, pp. 3601–3604, 2003.
[100] C. Wilson, L. Hilyer, and P. Green. (1995). *Genefinder Documentation*, [Online]. Available: http://weeds.mgh.harvard.edu/doc/genefinder.doc.html
[101] J. Besemer and M. Borodovsky, "GeneMark: Web software for gene finding in prokaryotes, eukaryotes and viruses," *Nucleic Acids Res. (Web server issue)*, vol. 33, pp. W451–W454, 2005.
[102] M. G. Reese, D. Kulp, H. Tammana, and D. Haussler, "Genie-gene finding in Drosophila melanogaster," *Genome Res.*, vol. 10, no. 4, pp. 391–393, 2000.
[103] X. Huang, M. Adams, H. Zhou, and A. Kerlavage, "A tool for analyzing and annotating genomic sequences," *Genomics*, vol. 46, pp. 37–45, 1997.
[104] A. A. Mironov, J. W. Fickett, and M. S. Gelfand, "Frequent alternative splicing of human genes," *Genome Res.*, vol. 9, no. 12, pp. 1288–1293, 1999.
[105] R. Sorek, R. Shamir, and G. Ast, "How prevalent is functional alternative splicing in the human genome?" *Trends Genet.*, vol. 20, no. 2, pp. 68–71, 2004.
[106] V. Veeramachaneni, W. Makalowski, M. Galdzicki, R. Sood, and I. Makalowska, "Mammalian overlapping genes: The comparative perspective," *Genome Res.*, vol. 14, pp. 280–286, 2004.
[107] P. Yua, D. Maa, and M. Xua, "Nested genes in the human genome," *Genomics*, vol. 86, pp. 414–422, 2005.
[108] S. G. Gregory *et al.*, "The DNA sequence and biological annotation of human chromosome 1," *Nature*, vol. 441, pp. 315–321, 2006.
[109] D. J. Leader, G. P. Clark, J. Watters, A. F. Beven, P. J. Shaw, and J. W. Brown, "Clusters of multiple different small nucleolar RNA genes in plants are expressed as and processed from polycistronic pre-snoRNAs," *EMBO J.*, vol. 16, pp. 5742–5751, 1997.
[110] T. Blumenthal, "Gene clusters and polycistronic transcription in eukaryotes," *Bioessays*, vol. 20, pp. 480–487, 1998.
[111] A. A. Mironov, P. S. Novichkov, and M. S. Gelfand, "Pro-Frame: Similarity-based gene recognition in eukaryotic DNA sequences with errors," *Bioinformatics*, vol. 17, pp. 13–15, 2001.
[112] C. Mathé, A. Peresetsky, P. Déhais, M. Van Montagu, and P. Rouzé, "Classification of Arabidopsis thaliana gene sequences: Clustering of coding sequences into two groups according to codon usage improves gene prediction," *J. Mol. Biol.*, vol. 285, pp. 1977–1991, 1999.
[113] 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.
[114] T. A. Thanaraj, "A clean data set of EST-confirmed splice sites from Homo sapiens and standards for clean up procedures," *Nucleic Acids Res.*, vol. 27, no. 13, pp. 2627–2637, 1999.

Sanghamitra Bandyopadhyay (M'99–SM'05) received the Bachelor's degree in physics and computer science, both from Calcutta University, India, in 1988 and 1991, respectively, and the Master's degree in computer science from the Indian Institute of Technology (IIT), Kharagpur, in 1993, and the Ph.D. degree in computer science from the Indian Statistical Institute, Kolkata, in 1998.

She is currently an Associate Professor at the Indian Statistical Institute. She has worked in the Los Alamos National Laboratory, Los Alamos, NM, in 1997; the University of New South Wales, Sydney, Australia, in 1999; the Department of Computer Science and Engineering, University of Texas at Arlington in 2001; the University of Maryland, Baltimore County, in 2004; the Fraunhofer Institute AiS, St. Augustin, Germany, in 2005; and Tsinghua University, China, in 2006. She has also delivered lectures at the Imperial College, London, U.K.; Monash University, Australia; the University of Aizu, Japan; the University of Nice, France; and the University Kebangsaan Malaysia, Kuala Lumpur, Malaysia. She is the coauthor of three books and more than 100 research publications. Her current research interests include pattern recognition, data mining, evolutionary and soft computation, bioinformatics, and parallel and distributed systems.

Dr. Bandyopadhyay is the first recipient of the Dr. Shanker Dayal Sharma Gold Medal and the Institute Silver Medal for the Best All Round Post Graduate Performer in IIT, Kharagpur, in 1994. She also 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 has been the Guest Editor of special issues of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, PART B. She has been the Program Chair, Tutorial Chair, and a member of the program committees of many international conferences.

Ujjwal Maulik (M'99–SM'05) received the Bachelor's degree in physics and computer science, both from Calcutta University, India, in 1986 and 1989, respectively, and the Master's and Ph.D. degrees in computer science, both from Jadavpur University, India, in 1991 and 1997, respectively.

He was the Head of the School of Computer Science and Technology Department, Kalyani Government Engineering College, Kalyani, India, during 1996–1999. He has worked in the Center for Adaptive Systems Application, Los Alamos, NM, in 1997; the University of New South Wales, Sydney, Australia, in 1999; the University of Texas at Arlington in 2001; the University of Maryland, Baltimore County, in 2004; and the Fraunhofer Institute AiS, St. Augustin, Germany, in 2005. He is currently a Professor in the Department of Computer Science and Engineering, Jadavpur University, Kolkata, India. He is the coauthor of two books and about 100 research publications. His current

research interests include artificial intelligence and combinatorial optimization, soft computing, pattern recognition, data mining, bioinformatics, very large-scale integration, and distributed system.

Dr. Maulik is a Fellow of the Institution of Electronics and Telecommunication Engineers, India. He is the recipient of the Government of India Better Opportunities for Young Scientists in Chosen Areas of Science and Technology Fellowship in 2001. He has been the Program Chair, Tutorial Chair, and a member of the program committees of many international conferences and workshops.

Debadyuti Roy received the B.Sc. degree from St. Xaviers' College, Kolkata, India, and the M.Sc. degree from the Chennai Mathematical Institute, Chennai, India, in 2002 and 2004, respectively, both in computer science. She is currently working toward the Ph.D. degree at the Department of Statistics, University of Michigan, Ann Arbor.

Her current research interests include computational statistics, bioinformatics, and machine learning.