To appear in: Proceedings of the Workshop on Knowledge Discovery and Emergent Complexity in Bioinformatics (KDECB / Benelearn'06), Lecture Notes in Bioinformatics, Springer-Verlag, 2007

Advancing the State of the Art in Computational Gene Prediction

William H. Majoros, Uwe Ohler

Center for Bioinformatics and Computational Biology
Institute for Genome Sciences and Policy, Duke University
101 Science Drive, Durham, NC 27708, USA
{bmajoros, uwe.ohler}@duke.edu

Abstract. Current methods for computationally predicting the locations and intron-exon structures of protein-coding genes in eukaryotic DNA are largely based on probabilistic, state-based generative models such as hidden Markov models and their various extensions. Unfortunately, little attention has been paid to the optimality of these models for the gene-parsing problem. Furthermore, as the prevalence of alternative splicing in human genes becomes more apparent, the "one gene, one parse" discipline endorsed by virtually all current gene-finding systems becomes less attractive from a biomedical perspective. Because our ability to accurately identify all the isoforms of each gene in the genome is of direct importance to biomedicine, our ability to improve gene-finding accuracy both for human and non-human DNA clearly has a potential to significantly impact human health. In this paper we review current methods and suggest a number of possible directions for further research that may alleviate some of these problems and ultimately lead to better and more useful gene predictions.

1. Introduction

The growing availability of large quantities of genomic sequence data for both human and non-human species has promoted a renewed interest in purely computational methods for finding protein-coding genes in raw DNA. In the case of vertebrate genomes, the problem has been fairly likened to that of finding the proverbial needle in a haystack, with the additional complication that each needle has an internal structure which also needs to be predicted.

Of the methods which have been investigated for solving this difficult problem, those based on probabilistic models of gene composition and structure have largely come to dominate, with the emphasis in the field now being on the use of hidden Markov models (HMMs) and their various extensions—in particular, those permitting the incorporation of various forms of external evidence such as patterns of evolutionary conservation between related genomes. As the field continues along this track, a number of difficulties have emerged which suggest that the use of purely generative models for heuristic parsing may not be an ideal framework for automated gene prediction.

In particular, the widespread existence of alternative splicing in mammalian genes, the suboptimality of maximum likelihood HMMs for Viterbi parsing, and the lack of efficient discriminative training procedures for stochastic parsers all seem to be conspiring to keep the predictive accuracy of practical gene-finding systems substantially below what is needed by the users of these systems. In the case of biomedical applications, our ability to overcome these limitations may translate into significant impacts on human health.

In this paper we suggest a number of possible directions for further research that may alleviate some of these problems and ultimately lead to better and more useful gene predictions in eukaryotic DNA.

2. Background

2.1 The Problem of Finding and Parsing Eukaryotic Protein-coding Genes

The human genome comprises 23 chromosomes, each consisting of a single DNA molecule which is in turn formed out of a linear series of *nucleotides*. Nucleotides come in four varieties: *adenine* (A), *cytosine* (C) *guanine* (G), and *thymine* (T). If each nucleotide is denoted by a single letter from the DNA alphabet

α={A,C,G,T}, the entire genome can then be represented by a sequence of approximately 2.9 billion letters. Embedded within this enormous sequence—at seemingly random intervals—are the actual *genes*, which encode the proteins used by the cell to mediate the building and operation of a complete organism. Expression of a gene begins with its transcription into *messenger RNA* (*mRNA*), which may then be *spliced* by the eukaryotic *spliceosome* to remove stretches of nonfunctional DNA within the gene known as *introns*. The two ends of the mRNA are then specially processed and the message is exported out of the nucleus to await *translation* by a molecular complex called a *ribosome*. This latter process pairs off individual *amino acids* with each triple of nucleotides (called a *codon*) along the message. The concatenation of these amino acids forms a polypeptide which finally folds into a functional protein. In this way, the precise sequence of nucleotides comprising a gene, and the precise way in which that gene's mRNA is spliced, determine the final form of the protein product and thus influence the operation of the cell. Fig. 1 summarizes this process.

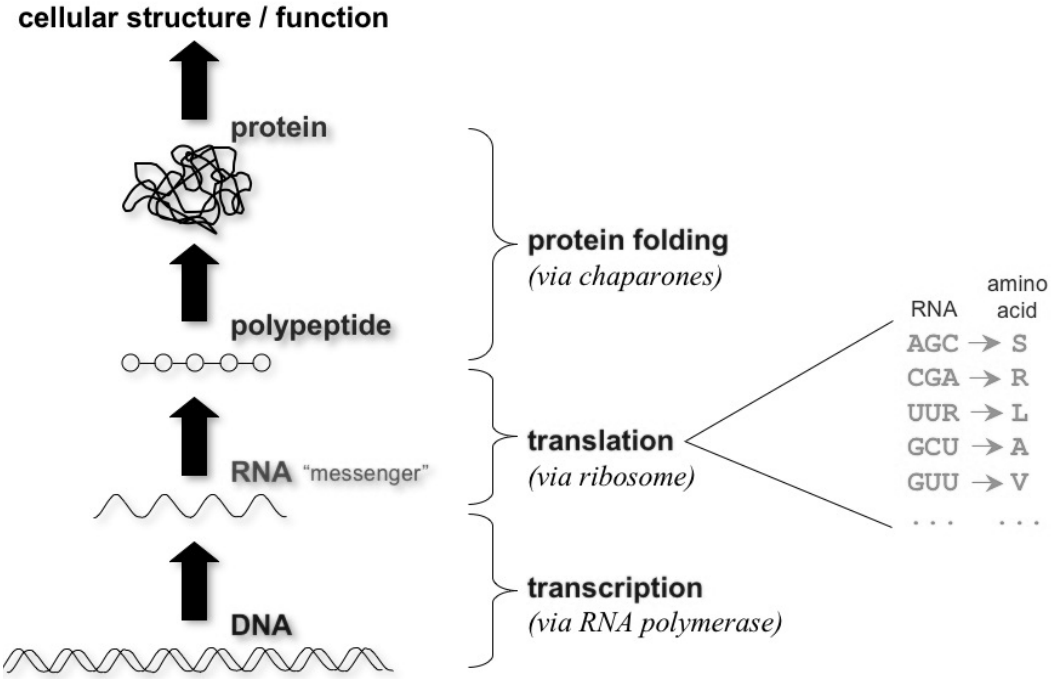

Fig. 1. The central dogma of molecular biology: DNA gives rise to RNA messages, which are translated into polypeptides that then fold into functional proteins. Source: Majoros WH, *Methods for Computational Gene Prediction*, Cambridge University Press (forthcoming), reproduced with permission.

The human gene-finding problem is a difficult one for two reasons: (1) the genes comprise less than 2% of our 2.9 billion letter genome, and (2) once a gene is found, the locations of the introns within the gene must be precisely determined before the protein product of the gene may be accurately deduced. The problem is thus one of *parsing*—i.e., partitioning an input sequence into a series of "words" (non-overlapping intervals of various types). The top portion of Fig. 2 shows a sample parse of a DNA sequence; rectangular boxes represent *exons* (non-intronic regions of a gene), the line segments separating pairs of exons represent introns, and the white spaces to the left and right of the gene represent *intergenic* regions.

Shaded portions of exons represent the parts of the gene which are actually translated into amino acids; in typical eukaryotic organisms, only the region between the *start codon* (ATG) and the *stop codon* (one of TGA, TAG, or TAA) is translated. Hatched portions of exons in the figure therefore represent *untranslated regions* (*UTRs*), and are generally not predicted by current gene-finding programs (though preliminary work in this direction shows some promise—e.g., [1]). The bottom portion of the figure emphasizes the *signals*, or fixed-length nucleotide motifs, which serve as boundaries for individual exons and introns. Most eukaryotic introns begin with a GT dinucleotide (called a *donor site*) and end with an AG (called an *acceptor site*).

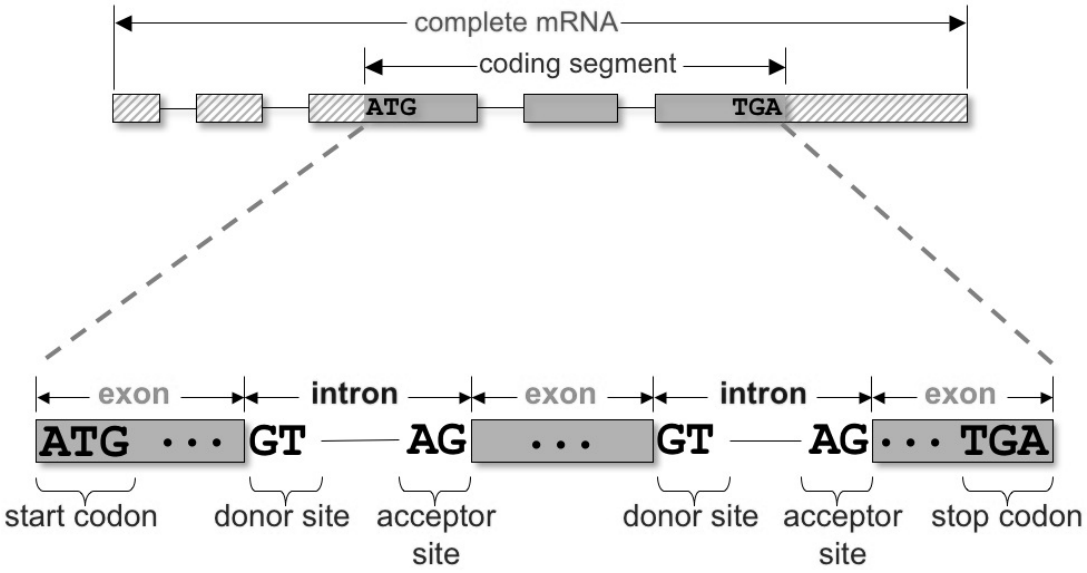

Fig. 2. The gene-parsing problem. A complete mRNA consists of one or more exons (rectangles). Portions of these exons may be coding (gray) or noncoding (hatched), with only the former giving rise to amino acids during translation. The coding segment extends from a start codon (ATG) to a stop codon (TGA, TAG, or TAA), with one or more introns (GT to AG) in between. Introns are spliced out prior to translation into a protein. Source: Majoros WH, *Methods for Computational Gene Prediction*, Cambridge University Press (forthcoming), reproduced with permission.

A gene *parse* thus consists of a syntactically valid series of signals from the set V={ATG, GT, AG, TGA, TAA, TAG} which have been identified in the input sequence. The necessary syntactic constraints on the parse of a genomic sequence are:

ATG→TAG
ATG→GT
GT→AG
AG→GT
AG→TAG
TAG→ATG

where the rule $X \rightarrow Y$ indicates that signal X may be followed by signal Y in a syntactically valid parse (rules for genes on the opposite DNA strand are easily obtained from these). The set of all valid parses for a given input sequence may be represented using a *parse graph* (Fig. 3) in which vertices represent putative signals and edges represent possible exons, introns, and intergenic regions.

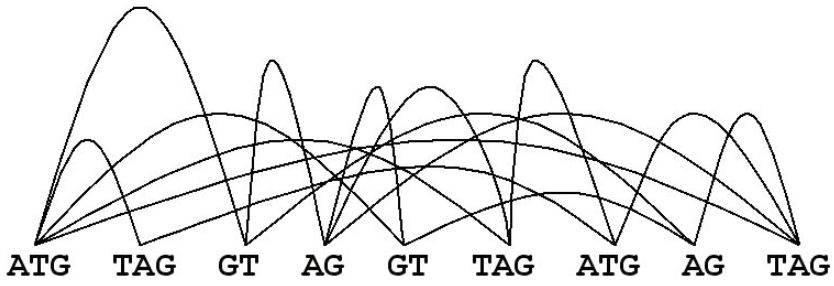

Fig. 3. An example parse graph. Vertices are shown as dinucleotide or trinucleotide motifs at the bottom. Edges denote exons, introns, or intergenic regions. Source: Majoros WH, *Methods for Computational Gene Prediction*, Cambridge University Press (forthcoming), reproduced with permission.

Because not every ATG/GT/AG/TAG/TGA/TAA occurring in a sequence is a true start codon, donor site, acceptor site, or stop codon, as recognized by the living cell, the gene-parsing problem is a highly ambiguous one. For this reason, stochastic parsers based on probabilistic models of DNA have largely come to dominate the gene-finding field. Several of the most popular types of model for this task are described in the sections that follow.

2.2 Hidden Markov Models

A *hidden Markov model* (*HMM*) is a state-based generative model which emits symbols over a finite alphabet. Formally, a hidden Markov model $M=(Q,\alpha,P_t,P_e)$ operates by beginning in the special state $q_0 \in Q$, transitioning stochastically from state to state (i.e., between elements of $Q=\{q_0,q_1,\ldots,q_{m-1}\}$) according to the *transition distribution*, $P_t(q_j \mid q_i)$, and emitting a single symbol $c \in \alpha$ according to the *emission distribution*, $P_e(c \mid q)$, upon entering state $q \in Q$. The machine ceases operation when it re-enters state q_0 (which emits no symbols). Gene-finding with an HMM is accomplished by positing that the DNA sequence S under study was generated by a particular model M having alphabet $\alpha=\{\text{A,C,G,T}\}$ and then identifying the most probable *path* (series of states) ϕ^* by which M could have generated S:

$$\begin{aligned}\phi^* &= \underset{\phi}{argmax}\, P(\phi \mid S) = \underset{\phi}{argmax}\, \frac{P(\phi,S)}{P(S)} = \underset{\phi}{argmax}\, P(\phi,S) \\ &= \underset{\phi}{argmax}\, P(S \mid \phi)P(\phi).\end{aligned} \tag{1}$$

That is, were the HMM to emit sequence S, the most probable way for it to do so would be for it to pass through precisely the series of states specified by ϕ^*. Eq. (1) can be further factored into a product of emission and transition probabilities along a prospective path ϕ by decomposing $P(\phi)$ into P_t terms, and $P(S \mid \phi)$ into P_e terms:

$$\phi^* = \underset{\phi}{argmax}\, P_t(q_0 \mid y_{|S|}) \prod_{i=0}^{|S|-1} P_e(x_i \mid y_{i+1}) P_t(y_{i+1} \mid y_i)\ , \tag{2}$$

where $S = x_0 \ldots x_{|S|-1}$ is a sequence of length $|S|$, for nucleotides x_i, and $\phi=(y_0,\ldots,y_{|S|+1})$ for states $y_i \in Q$; $y_0 = q_0$ and $y_{|S|+1} = q_0$ are assumed since the machine must begin and end in state q_0. Actually finding the optimal path (or "parse") ϕ^* can be carried out using Viterbi's dynamic programming algorithm [2], which entails the computation of two matrices, $V(i,k)$ for the path probabilities and $T(i,k)$ for the *traceback pointers* which allow us to reconstruct the optimal path once the matrices have been computed:

$$V(i,k) = \begin{cases} \max\limits_j V(j,k-1) P_t(q_i \mid q_j) P_e(x_k \mid q_i) & \text{if } k > 0, \\ P_t(q_i \mid q_0) P_e(x_0 \mid q_i) & \text{if } k = 0. \end{cases} \tag{3}$$

$$T(i,k) = \begin{cases} \arg\max\limits_j V(j,k-1) P_t(q_i \mid q_j) P_e(x_k \mid q_i) & \text{if } k > 0, \\ 0 & \text{if } k = 0. \end{cases} \tag{4}$$

Reconstruction of the optimal path proceeds by starting at the highest-scoring cell (i,k) in the last column of the V matrix and iteratively assigning $i \leftarrow T(i,k)$ and $k \leftarrow k-1$ until the first column (k=0) is reached; the successive i visited during this traversal correspond to the states q_i in the optimal path (in reverse order).

Training of an HMM is most commonly carried out using maximum likelihood estimation (MLE). In the simplest case, in which individual nucleotides in the training sequences are labeled with corresponding states in the model, MLE can be performed simply by tabulating the number of times $C(q_i,q_j)$ that state q_i was followed by q_j in the training set, and also the number of times $C(s_k,q_i)$ that nucleotide s_k was labeled with state q_i, for alphabet $\alpha=\{s_k \mid 0 \le k < size(\alpha)\}$. Normalizing these counts produces the desired probability estimates:

$$P(q_j \mid q_i) \approx \frac{C(q_i,q_j)}{\sum_{h=0}^{|Q|-1} C(q_i,q_h)}, \qquad P_e(s_k \mid q_i) \approx \frac{C(s_k,q_i)}{\sum_{h=0}^{|\alpha|-1} C(s_h,q_i)}. \tag{5}$$

More sophisticated methods such as *Viterbi training* or the use of an *expectation maximization* (*EM*) algorithm [3] are required when labeled training data are not available [4].

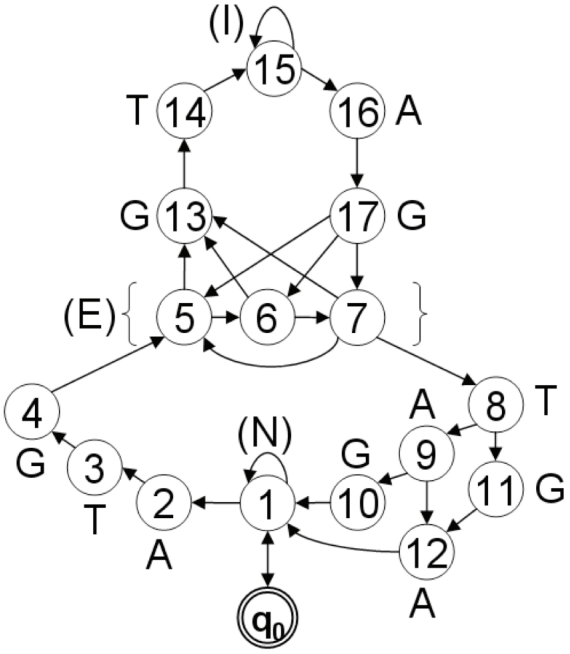

Fig. 4. A simple HMM for gene finding. States are represented as circles and transitions as arrows. Probabilities are omitted for clarity. States which emit only one symbol are shown with the corresponding symbol next to the state. The special state q_0 is the start/stop state, which emits no symbols. Source: Majoros WH, *Methods for Computational Gene Prediction*, Cambridge University Press (forthcoming), reproduced with permission.

A simple HMM for gene finding is depicted in Fig. 4. The state labeled (N) represents intergenic regions. The machine may self-transition any number of times while in this state to generate arbitrarily long intergenic regions. Following the path $q_2 \rightarrow q_3 \rightarrow q_4$ produces a start codon (ATG) and places the machine in the exon states (q_5, q_6, q_7—three states to represent the three codon positions). Generation of an intron begins with a donor site (GT; $q_{13} \rightarrow q_{14}$) followed by an arbitrarily long intronic region ("I", q_{15}) and then an acceptor site (AG; $q_{16} \rightarrow q_{17}$). The reader can easily verify that states $\{q_8, q_9, q_{10}, q_{11}, q_{12}\}$ generate only the three eukaryotic stop codons, TGA, TAA, and TAG. Note that states labeled with a specific nucleotide in the figure can generate only that symbol (e.g., T for state q_{14}).

Such a simple HMM can be extended in various ways to improve gene-finding accuracy, primarily through the more detailed modeling of statistical biases in nucleotide composition within gene features. An example is the use of higher-order emission probabilities:

$$P_e(x_i \mid x_{i-n}x_{i-n+1}\ldots x_{i-1}, q_j) \approx \frac{C(x_{i-n}x_{i-n+1}\ldots x_i, q_j)}{\sum_{s\in\alpha} C(x_{i-n}x_{i-n+1}\ldots x_{i-1}s, q_j)} \quad , \tag{6}$$

where $P_e(x_i \mid x_{i-n}x_{i-n+1}\ldots x_{i-1}, q_j)$ denotes the probability of state q_j emitting symbol x_i, given that the subsequence $x_{i-n}\ldots x_{i-1}$ has just been emitted; counts $C(x_{i-n}x_{i-n+1}\ldots x_i, q_j)$ for all $(n+1)$-letter sequences $x_{i-n}x_{i-n+1}\ldots x_i$ may be derived from the training data as before.

An unfortunate aspect of gene modeling with HMMs is the fact that variable-length features (such as exons or introns) are implicitly modeled as having geometrically distributed lengths, as enforced via the compounding of repeated transition probabilities during generation of a variable-length feature. Generalized HMMs (*GHMMs*—see below) solve this problem while also allowing for greater modeling flexibility.

2.3 Generalized Hidden Markov Models

GHMMs improve on HMMs by abstracting the generation of entire gene features into single states; i.e., upon entering a state q_i the machine may emit an entire subsequence S_i before making the next transition. In this way, feature lengths may be explicitly modeled via arbitrary distributions (not necessarily geometric), the syntactic and statistical properties of individual features may be encapsulated within each state in an arbitrary (i.e., non-Markovian) way, and the number of states required to implement a production-quality gene-finding system can be kept relatively small.

Formally, a GHMM is a stochastic generative model $M=(Q, \alpha, P_t, P_e, P_d)$ in which all terms are as defined for the HMM case, except that individual state emissions are entire substrings (rather than individual symbols) over α, with those emissions having lengths distributed according to the state-specific duration distribution $P_d(L \mid q)$, $L\in\mathbb{N}$, $q\in Q$. Decoding (i.e., finding the optimal path) with a GHMM is similar to the HMM case:

$$\phi^* = \underset{\phi}{argmax}\, P_t(q_0 \mid y_n)\prod_{i=1}^{n} P_t(y_i \mid y_{i-1})P_d(d_i \mid y_i)P_e(S_i \mid y_i, d_i) \ , \tag{7}$$

for putative parse $\phi=(y_0, \ldots, y_{n+1})$, $\forall_i y_i\in Q$, where it can be seen that the emission term $P_e(S_i \mid y_i, d_i)$ is now additionally conditioned on the duration $d_i=|S_i|$ of the subsequence S_i emitted by state y_i; $S=S_1S_2\ldots S_n$. Note that the parse again begins and ends in (silent) state q_0: $y_0=y_{n+1}=q_0$. An efficient dynamic programming heuristic exists for the GHMM case [5,6] which first identifies high-scoring putative signals in the input sequence and links these into a continuously-pruned parse graph; by weighting the vertices and edges of this graph with corresponding terms from Eq. (7) we obtain a structure that can be searched very quickly to find the optimal parse.

Training of a GHMM is most often carried out using MLE by separately estimating the P_t, P_e, and P_d parameters from labeled training data. The P_d distribution is commonly represented via a smoothed histogram constructed from feature lengths in the training data; P_t is easily estimated by observing transition counts in the training data and normalizing these into probabilities, as in the HMM case. Because most GHMM-based gene finders utilize some form of *Markov chain* (a two-state, higher-order HMM in which transition probabilities are ignored) as the submodel within each variable-length state of the GHMM (i.e., states for exons, introns, or intergenic regions), estimation of P_e is rendered trivial; interpolation techniques are also sometimes employed to mitigate the effects of sampling error when using higher-order

models [7]. Fixed-length states of the GHMM, which correspond to signals such as start/stop codons and donor/acceptor sites, are typically represented using a *weight matrix* (*WMM*) [8], in which each position of a fixed-length signal window is described by a position-specific emission distribution, possibly conditional on the symbols residing at other positions within the window [9]. Thus, for most GHMM implementations, MLE parameter estimation may be performed without the need for iterative methods such as Viterbi training or EM.

2.4 Pair HMMs and Generalized Pair HMMs

A significant increase in predictive accuracy can often be achieved by modeling evolutionary trends as observed in the genomic sequence of related organisms. This is due to the fact that natural selection tends to operate more stringently on the coding (versus noncoding) regions of any genome. When predicting genes in some target genome S, an informant genome I from some related organism may be employed by aligning portions of S and I for which *homology* (evolutionary commonality of descent) may be inferred via sequence similary. In this case, the optimal parse may be defined as that ϕ which maximizes $P(\phi|S,I)$, which we may factor as:

$$\begin{aligned}\phi^* &= \arg\max_{\phi} P(\phi \mid S, I) = \arg\max_{\phi} \frac{P(\phi, S, I)}{P(S, I)} = \arg\max_{\phi} P(\phi, S, I) \\ &= \arg\max_{\phi} P(\phi) P(S, I \mid \phi),\end{aligned} \tag{8}$$

where $P(\phi)$ is merely the product of transition probabilities incurred along the path ϕ just as before, leaving only the problem of evaluating $P(S,I|\phi)$. A particularly elegant method for modeling the latter joint probability is by positing a special type of Markov model $M=(Q,\alpha,P_t,P_e)$ in which each state $q\in Q$ emits pairs of symbols $(s_1,s_2)\in\alpha\times\alpha$ rather than individual symbols as in a standard HMM. Replacing α with the augmented alphabet $\alpha^- = \{A,C,G,T,-\}$, for '$-$' the gap symbol representing unaligned positions or gaps in an alignment, we arrive at a model capable of emitting aligned sequences with gaps. Such a model is called a *Pair HMM* (*PHMM*); an example is shown in Fig. 5.

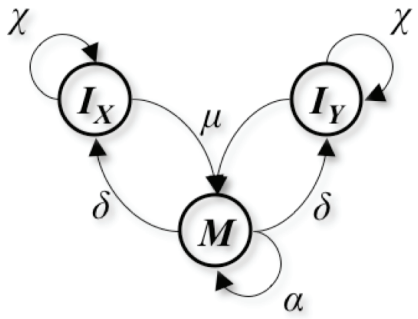

Fig. 5. A simple Pair HMM. State M emits matched or mismatched symbols into an alignment; I_X and I_Y emit gapped alignment positions (i.e., gaps in sequence X for I_X and in sequence Y for I_Y). Transition probabilities are indicated using letters α, χ, δ, and μ. Source: Majoros WH, *Methods for Computational Gene Prediction*, Cambridge University Press (forthcoming), reproduced with permission.

Decoding with a PHMM may be described as:

$$\phi^* = \underset{\phi=\{y_0,\ldots,y_{n+1}\}}{\arg\max}\ P_t(q_0 \mid y_n)\prod_{i=1}^{n} P_e(a_{i,1}, a_{i,2} \mid y_i)P_t(y_i \mid y_{i-1}) , \tag{9}$$

where $a_{i,j}$ denotes the i^{th} symbol in the j^{th} track ($j \in \{1,2\}$) of the alignment formed by a putative parse ϕ. Unfortunately, a dynamic programming solution to this optimization problem requires a three-dimensional matrix and therefore significantly greater computational resources than for a standard HMM. Heuristics are thus commonly employed to prune the matrix, as illustrated in Fig. 6. The heuristic aligner *BLAST* [10] is often used to precompute a set of *guide alignments* (black bars in the figure); portions of the dynamic programming matrix which are deemed too distant from these guide alignments are pruned from the matrix and never evaluated.

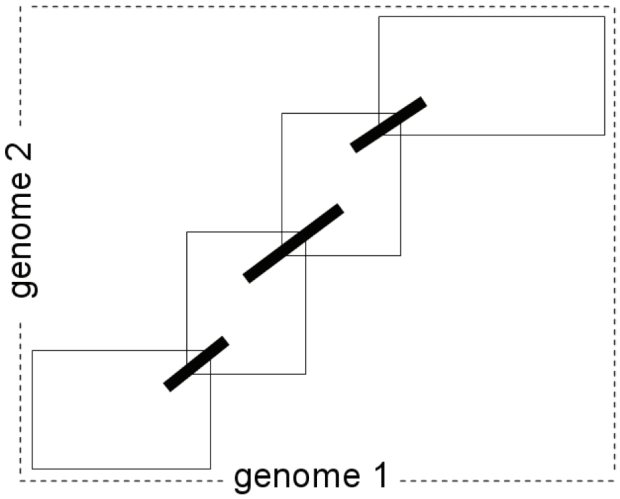

Fig. 6. Pruning an alignment matrix. Precomputed alignments are shown as solid bars; rectangles denote the portion of the alignment matrix which are actually evaluated. The third dimension, corresponding to states of the PHMM, is omitted for clarity. Source: Majoros WH, *Methods for Computational Gene Prediction*, Cambridge University Press (forthcoming), reproduced with permission.

Generalized PHMMs (GPHMMs) have also been employed for gene prediction [11,12]. A GPHMM may be obtained by embedding a PHMM within each state of a GHMM, so that each GPHMM state will emit pairs of aligned genomic features (e.g., exons, introns, or intergenic regions). The corresponding decoding optimization is given by:

$$\phi^* = P_t(q_0 \mid y_n)\underset{\phi}{\arg\max}\prod_{i=1}^{n} P_e(S_{i,1}, S_{i,2} \mid y_i, d_{i,1}, d_{i,2}) \\ P_t(y_i \mid y_{i-1})P_d(d_{i,1}, d_{i,2} \mid y_i). \tag{10}$$

One dynamic programming solution for GPHMMs proceeds by constructing a parse graph (Fig. 3) for each of the two input sequences and then aligning these graphs in such a way that like vertices (e.g., ATG-ATG, GT-GT, etc.) are permitted to align and unlike vertices (e.g., ATG-TAG) are not, with Eq. (10) serving as the objective function of the alignment process [12]. The resulting alignment between parse graphs will outline an isomorphism corresponding to a parse in each of the two graphs. Precomputed guide alignments

are generally also required for GPHMMs in order to achieve acceptable time-space complexity via pruning of the dynamic-programming matrix.

2.5 Phylogenetic HMMs

Whereas PHMMs and GPHMMs incorporate homology evidence from a single informant genome, *Phylogenetic HMMs* (*PhyloHMMs*) permit evidence from any number of informants to be utilized, with a Bayesian network being employed to reduce bias due to the non-independence of the informants. Precomputed alignments are again used; unlike PHMMs and GPHMMs, however, current PhyloHMM implementations adhere strictly to the precomputed alignments, rather than merely using them as guides for the purpose of pruning the search space.

The decoding derivation for a PhyloHMM is:

$$\begin{aligned}\phi^* &= \arg\max_{\phi} P(\phi \mid S, I^{(1)}, \ldots, I^{(n)}) \\ &= \arg\max_{\phi} \frac{P(\phi, S, I^{(1)}, \ldots, I^{(n)})}{P(S, I^{(1)}, \ldots, I^{(n)})} \\ &= \arg\max_{\phi} P(\phi, S, I^{(1)}, \ldots, I^{(n)}) \\ &= \arg\max_{\phi} P(\phi) P(S, I^{(1)}, \ldots, I^{(n)} \mid \phi) \\ &= \arg\max_{\phi} P(\phi) P(S \mid \phi) P(I^{(1)}, \ldots, I^{(n)} \mid S, \phi),\end{aligned} \tag{11}$$

for target genome S and informants $I^{(1)}, \ldots, I^{(n)}$. The $P(\phi)P(S \mid \phi)$ term can be evaluated using a standard GHMM decoder. The remaining term, $P(I^{(1)} \ldots I^{(n)} \mid S, \phi)$, can be evaluated as follows:

$$P(I^{(1)}, \ldots, I^{(n)} \mid S, \phi) = \prod_{y_i \in \phi} \prod_{j=b_i}^{e_i} F(I_j^{(1)}, \ldots, I_j^{(n)} \mid S_j, \psi_i), \tag{12}$$

where the second product is over columns b_i through e_i of the precomputed alignment, according to the emission of state $y_i \in \phi$. The evolution model ψ_i typically differs between coding states (i.e., ψ_{coding}) and non-coding states ($\psi_{noncoding}$) so as to model the differences in rates of evolution between the coding and noncoding portions of genomes. These rates are reflected in the $F(\bullet)$ term, which is known as *Felsenstein's algorithm* [13], and is used to compute the likelihood of a single column in the alignment:

$$F(I_j^{(1)}, \ldots, I_j^{(n)} \mid S_j, \psi_i) = \sum_{unobservables} \left(\prod_{\substack{\text{nonroot} \\ v}} P(v_j \mid parent(v_j), \psi_i) \right), \tag{13}$$

where the summation is over all possible assignments of nucleotide sequences to the (unobserved) ancestral species in a *phylogenetic tree* (or *phylogeny*) describing the evolutionary relationships among the target and informant genomes; the phylogeny effectively serves as a Bayesian network for modeling evolutionary dependencies. v_j is the residue in column j of the alignment for any non-root vertex v in the tree; these v_j

thus correspond to the (observable) informants as well as their (unobservable) common ancestors in the phylogeny. Summing over all the possible nucleotides in the ancestral genomes permits us to evaluate this formula in the presence of unobservables, by effectively computing an expectation. Before this computation may be performed the phylogeny must first be re-rooted so that the target genome is at the root of the phylogeny and the informant genomes are at the leaves [14], reflecting the dependence of the informants on the target.

The actual dependence of each genome on its parent genome—denoted $P(v_j|parent(v_j), \psi_i)$ in Eq. (13)—may be represented at the individual nucleotide level using a substitution matrix **M** in which the entry $\mathbf{M}_{a,b}$ gives the probability of nucleotide *a* evolving into nucleotide *b* during a period of time equivalent to the evolutionary distance between the two genomes. Non-independence of the columns in the alignment may be modeled as well, by conditioning the substitution matrix on one or more preceding nucleotides in the parent genome, similar to the higher-order Markov models described earlier.

The substitution matrices comprising the evolutionary models ψ_i of a PhyloHMM may be independently trained from aligned features of the appropriate type (e.g., aligned coding exons for ψ_{coding}) using standard maximum likelihood techniques developed prevously for phylogeny reconstruction [15]. A general-purpose gradient ascent procedure may thus be employed to maximize the likelihood of the training data using Eq. (12) as the objective function of the optimizer.

2.6 Ad hoc "Combiner" Methods

Integration of other forms of evidence besides evolutionary conservation between genomes—such as expression evidence in the form of messenger RNAs and proteins culled from living cells of the target organism—can be incorporated as well, though current methods tend to be largely *ad hoc* in nature and therefore defy (at present) any concise, unified description such as those given in the preceding sections. These programs are referred to as *combiners*, since they may combine many disparate sources of evidence, including predictions from other gene-finding programs. Despite their typically *ad hoc* nature, some combiner programs have proven to be among the most accurate systems currently available for predicting gene structure [16,17]. It seems a curious fact that, despite their not conforming (in most cases) to a rigorous probabilistic formulation as in the case of Markov models and their various relatives described earlier, combiner-type programs can perform so well. As we will discuss in greater detail below, this may be due (in part) to the fact that combiner systems are typically trained *discriminatively* via extensive manual tuning of evidence weights, with the goal of the manual tuner being to maximize the accuracy of the gene predictions when the system is applied to the sequences in the training set, as opposed to maximizing the *likelihood* of the training data as in MLE. Another likely reason for the success of combiners is their integration of all available forms of evidence in arriving at a prediction; it is in reference to this latter property that combiner-type programs are often referred to as being *integrative*. Unfortunately, because the "gold standard" against which gene finders are often measured—namely, test sets of previously annotated genes—is often produced (or at least heavily influenced) by a combiner-like "annotation pipeline" (see section 4.5), the superiority of integrative systems may in fact be somewhat over-estimated.

3. Limitations of Current Methods

3.1 MLE+Viterbi Is Not Optimal

As described above, most state-of-the-art gene-finding systems are at present based on Markovian models of one type or another (i.e., HMMs, GHMMs, PHMMs, GPHMMs, PhyloHMMs). The vast majority of systems based on these models are trained via MLE and are then subjected to some form of Viterbi decoding, with the latter being extended in various ways to incorporate external evidence such as informant sequences (e.g., PHMMs and PhyloHMMs) as well as modeling enhancements such as explicit state duration (e.g., GHMMs). Much evidence suggests, however, that these MLE-trained systems are not optimal in practice, in that the use of non-maximum-likelihood parameters can often improve the accuracy of a given probabilistic parser when the parser is later utilized for Viterbi-based prediction. Indeed, the suboptimality of the MLE+Viterbi strategy has been well-documented for some time now in the field of

speech recognition, in which HMM-based systems are fairly routinely subjected to one of several non-MLE forms of training collectively known as *discriminative training* [18-20].

Whereas the goal of maximum likelihood training is to maximize the joint likelihood of the training set T (consisting of pairs of sequences S and their "correct" parse ϕ) given the model parameters θ—e.g.,

$$\theta^*_{MLE} = \arg\max_{\theta} \left(\prod_{(S,\phi)\in T} P(\phi, S \mid \theta) \right), \tag{14}$$

the goal of discriminative training is to maximize the *expected accuracy* of the resulting parser. This latter goal can be formalized in a number of ways. A common formulation is the so-called *conditional maximum likelihood* (*CML*):

$$\theta^*_{CML} = \arg\max_{\theta} \left(\prod_{(S,\phi)\in T} P(\phi \mid S, \theta) \right) = \arg\max_{\theta} \left(\prod_{(S,\phi)\in T} \frac{P(\phi, S \mid \theta)}{P(S \mid \theta)} \right), \tag{15}$$

in which we require the parameterization θ^*_{CML} under which the correct parses of the training sequences are most probable, given the sequences and the model parameters. Unfortunately, methods for directly optimizing Eq. (15) for an HMM are not known [4], and while a number of heuristics have been developed within the field of speech recognition for this or similar objective functions (e.g., *maximum mutual information*, *MMI* [18]; *minimum classification error*, *MCE* [19]), these tend to be unstable in practice so that convergence is typically not guaranteed without manual tuning of additional parameters (e.g., [19, 21]). It should also be noted that for practical gene finders the number of model parameters to be optimized can be in the high thousands in the case of higher-order models, making thorough discriminative training of such models seem highly daunting at best.

Explicit discriminative training for HMM-based gene finders has thus been largely ignored (see [21] for a rare example). In the case of GHMMs and more sophisticated probabilistic models for gene finding, much anecdotal evidence suggests that a very crude form of discriminative training is typically performed via manual tuning of a small number of model parameters by the authors of these systems so as to improve the observed prediction accuracy on the training set or on a separate test set. In the case of *comparative* gene-finding systems (i.e., those incorporating external evidence apart from the target genome), such manual tuning is commonly performed by introducing one or more "fudge factors" to allow for the artificial weighting of the various components of the decoding objective function such as the informant component (e.g., "coding bias" in *ExoniPhy* [15]; "conservation score coefficient" in *N-SCAN* [22]; non-maximum-likelihood value for P_{match} in *TWAIN* [12]). Though these "fudge factors" appear to serve no theoretical role in the probabilistic formulation of the model, such manipulations can sometimes dramatically increase the accuracy of the resulting parser.

Automated discriminative training procedures for generalized HMMs and comparative systems such as pair HMMs and PhyloHMMs have received little or no attention as of yet. A rare exception involved the use of a crude gradient-ascent approach to optimize a handful of the thousands of parameters making up a GHMM-based gene finder [23]. Given the simplistic and *ad hoc* nature of the "fudge factor" approach described above for PhyloHMMs and other sophisticated probabilistic gene parsers, investigations into more comprehensive means of discriminatively optimizing these systems would seem to be well justified.

Alternatively, one might consider the very need for discriminative training of Markovian gene-finding models to be an indication that this family of models is perhaps not an ideal one for the gene-finding application. Investigations into explicitly discriminative, non-Markovian frameworks such as *conditional random fields* have recently produced promising preliminary results [24, 25]. The use of alternate HMM decoders (i.e., in place of Viterbi) remains another possibility, though experiments by ourselves with two recently-proposed alternate decoders (*posterior Viterbi* [26], *optimal accuracy decoder* [27]) suggest that these decoders do not provide an appreciable gain in predictive accuracy for eukaryotic gene finding, and in particular do not obviate the need for discriminative training of the model (unpublished data).

3.2 Reliance on Precomputed Alignments

As mentioned earlier, the PhyloHMM framework, and to a lesser extent the PHMM and GPHMM frameworks, rely on pre-computed alignments of the target and informant genomes to be used during gene prediction. In the case of Pair HMMs and GPHMMs, the pre-computed alignments serve largely as guides, so that the actual pairing off of target and informant nucleotides resulting from a decoding run of the system may differ to some degree from that prescribed by the pre-computed alignment, though in practice the aggressive pruning of the dynamic programming matrix around the guide alignments may preclude all but the smallest divergence from the pre-computed alignment. In the case of PhyloHMMs, all known implementations at present adhere to the pre-computed alignment precisely, so that alignment errors by the external alignment tool may give rise to spurious evolutionary patterns as seen by the PhyloHMM decoder. Ideally, one would like the gene prediction and alignment phases to proceed simultaneously, so as to mutually inform one another, as in the case of (non-pruned) PHMM decoding. Methods for efficiently achieving this in the case of PhyloHMMs have yet to be investigated.

3.3 Simplifying Assumptions

A number of simplifying assumptions are typically made in formulating a gene-finding model, most often for the purpose of reducing the computational complexity of the decoding process. In particular, various models assume that:

1. feature lengths are geometrically distributed (HMMs)
2. exon-intron structure does not change over evolutionary time (GPHMMs, PhyloHMMs)
3. pre-computed alignments are correct (PhyloHMMs; also to some degree GPHMMs and PHMMs)
4. each locus has exactly one correct parse (one "isoform")
5. the target sequence contains no frameshifts
6. genes do not overlap
7. non-consensus splice sites do not occur
8. stop codons do not code for any amino acid

Though all of these assumptions can be shown to be false in at least one biologically valid instance, few efforts have been undertaken to relax these assumptions. Known exceptions include the modeling of non-geometrically distributed intron lengths [28] and the modeling of genes which overlap on opposite strands [29], neither of which have seen widespread adoption in mainstream eukaryotic gene finders as of yet. In the case of non-consensus splice sites, though several software implementations do permit the user to explicitly request the modeling of non-consensus splice sites, a thorough analysis of the impact of this feature on prediction accuracy has yet to be performed, while conventional wisdom holds that the sensitivity gains can be more than offset by the loss in specificity.

Because Markovian-based gene finders utilize a Viterbi decoding step to find the single most promising parse of an input sequence, any genes which are predicted as part of the parse will be assigned a single exon-intron stucture by the gene finder. Unfortunately, many human genes (perhaps as many as 80%) can be spliced in multiple ways to produce distinct intron-exon structures, or *isoforms*. The issue of multiple isoforms is discussed in more depth in the next section.

The assumption that stop codons do not code for any amino acid is untrue in the very rare case of *selenocysteine*—an amino acid coded by the codon TGA (UGA in the mRNA). In general, gene finders do not predict genes containing in-frame stop codons (i.e., stop codons residing at a distance d from the beginning of the coding portion of the spliced gene, in which d is divisible by 3), except for the in-frame stop codon occurring at the very end of the gene. For most organisms, to allow the prediction of genes with in-frame stop codons (other than the termination codon at the end of the gene) would very likely result in a significant degradation in predictive accuracy, since for most sequenced genomes to date, the majority of known genes do not contain in-frame stop codons. A rare example of a gene-finding system which can predict selenocysteine-bearing genes has been described [30] in which homology evidence and other information from the UTR of a putative gene were used to limit the large number of possible in-frame stop-codon-bearing genes to a more reasonable number.

The assumption that genes do not overlap is specific to eukaryotic gene finders; because overlapping genes appear to be more common in prokaryotes, prokaryotic gene-finding programs have modeled overlapping genes for some time now [31, 32] and gene finders for eukaryotic viruses such as HIV also must deal with the phenomenon of overlapping genes [33]. In the case of eukaryotes, nested genes and genes which overlap other genes on the opposite strand are not just rare exceptions (e.g., in *Drosophila melanogastor* [34]), though most eukaryotic gene finders do not predict them. Two exceptions are SNAP [29] and AUGUSTUS [28], which can be run in a special single-strand mode, in which genes are independently predicted on either strand, so that a gene prediction on one strand may overlap a prediction the other strand.

Reliance on pre-computed alignments has already been discussed; the somewhat related issue of conservation of exon-intron structure in GPHMMs and PhyloHMMs is similarly vexing. Fig. 7 illustrates the problem for a pair of *Aspergillus* homologues. The upper track in the figure depicts the exon-intron structure of a particular gene in *A. oryzae*; the lower track depicts the homologous gene in *A. fumigatus*, where it can be seen that a number of structural changes have been effected since these organisms diverged from their common ancestor, though the encoded proteins have remained identical. Efficient GPHMM implementations generally do not permit the prediction of homologues with different exon-intron structures, since to do so would largely eliminate any opportunity for pruning the search space, resulting in dynamic programming matrices which are often too large to evaluate in a reasonable amount of time. In the case of PhyloHMMs, the potential for such structural changes would at the least seem to present a challenge for the alignment pre-processing phase. More specifically, the need for incorporating amino acid conservation into the alignment phase would seem to be greater than is perhaps recognized at present.

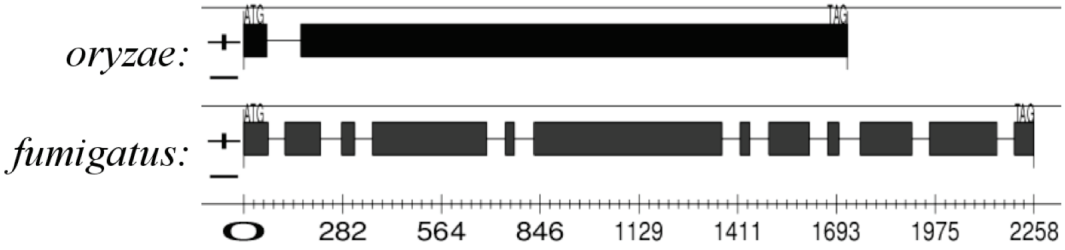

Fig. 7. An example of exon-intron structure divergence. These two genes from *Aspergillus oryzae* and *A. fumigatus* encode the same protein, but have accumulated a number of structural changes since their last common ancestor. Many comparative gene finders cannot easily model such structural changes. Source: Majoros WH, *Methods for Computational Gene Prediction*, Cambridge University Press (forthcoming), reproduced with permission.

3.4 The Existence of Alternative Splicing

The propensity for human genes to encode multiple, distinct proteins via alternative splicing (as well as alternative polyadenylation and alternative transcription/translation initiation) is now well documented [35]; Fig. 8 illustrates some of the potential effects of alternative splicing and related phenomena.

Each potential splicing pattern gives rise to a unique *isoform* for the locus. Some loci can have very many isoforms [36], and there is even evidence that exons from distinct loci in the human genome may sometimes be spliced together to encode a "chimeric" protein [37]. It has been suggested that the propensity for a locus to encode multiple proteins may account for the seemingly large mismatch between the estimated number of human genes (~25000) and the number of proteins (>100000), and is therefore a particularly important issue for human gene finding.

Despite the prevalence of these phenomena in human genes, however, virtually all state-of-the-art eukaryotic gene finders continue to enforce a one-gene-one-parse discipline via their use of Viterbi (or Viterbi-like) decoding to find the single optimal parse of the input sequence. We will address possible methods for relaxing this discipline in section 4.1.

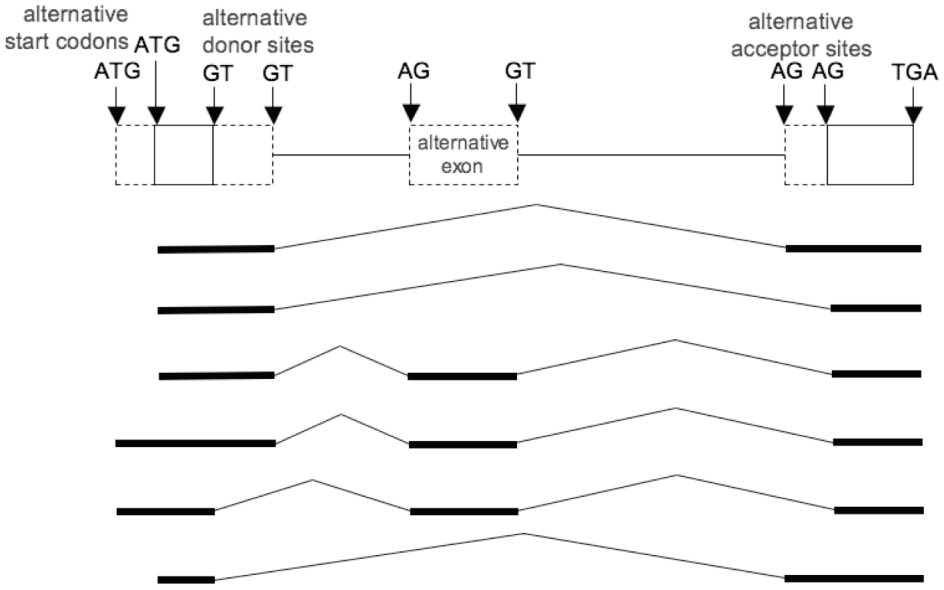

Fig. 8. Some possibilities for alternative splicing of coding segments (i.e., ignoring UTRs). Many isoforms may potentially be produced from a single locus in a combinatorial fashion. Source: Majoros WH, *Methods for Computational Gene Prediction*, Cambridge University Press (forthcoming), reproduced with permission.

4. Some Possible Future Directions

4.1 Redefining the Problem

The earliest "gene finding" systems were actually exon finders: that is, rather than predicting complete gene structures, they instead predicted individual exons, and left the task of assembling exons into complete genes to the end user. As Markov-based systems gained in popularity it became more feasible to predict whole gene structures via the well-established Viterbi decoding algorithm. As the prevalence of alternative splicing in mammalian genomes becomes better appreciated, however, the suitability of a Viterbi-based approach is increasingly cast into doubt. A modified version of Viterbi decoding which permits the efficient identification of the *N* best (rather than the single best) parses has been suggested as one possible means of addressing the issue of alternative splicing within current gene-finding frameworks [38]. However, not all possible valid alternative isoforms are actually produced in an organism, and without additional splicing-specific information, we will not be able to deduce the set of isoforms which are actually produced.

One possible remedy lies in redefining the problem so as to focus on the identification of likely exons in isolation—i.e., predicting individual exons without regard to their compatibility (i.e., whether they overlap, whether they maintain a consistent reading frame, etc.) with other predicted exons in a complete gene parse. The task of assembling these exon predictions into one or more predicted isoforms for a locus can then be left for downstream software, or for human annotators in the case of well-funded genome projects. Although this redefinition of the problem would seem to be a step backward toward the earlier exon-finding approaches mentioned above, there are a number of potential advantages to this change.

The most obvious advantage of such an approach, for organisms exhibiting appreciable levels of alternative splicing, is that it facilitates the identification of multiple isoforms by downstream analyses after exon prediction has been performed. For instance, the last few years have seen the development of a number of algorithms which allow the predictions of which individual exons are subject to alternative splicing [39-41], and additionally other alternative splicing patterns such as *intron retention* [42]. That is, the identification of likely exons and the assembling of exons into multiple isoforms become effectively decoupled, thereby entailing many of the advantages of modular software design (i.e., division of labor,

ease of development and debugging, efficiency gains through parallelization, etc.). Given a set of high-confidence exon predictions from an exon finder, research into optimal methods for combining these into multiple-isoform predictions may proceed without the need to repeatedly perform the content-scoring analyses encapsulated within the exon finder, perhaps significantly easing the computational load of development and research efforts. Indeed, were the exon predictions from one or more exon finders to be collected into publicly available data banks for each genome project, the annotation (and re-annotation) of these genomes at the whole-gene level may be considerably eased, since the exon-finding phase need not be performed anew as alternative parameterizations of the exon-assembly process are explored. Exons predicted by different exon finders may also be considered for combination by automated methods into coherent isoform predictions (thereby addressing the not-uncommon situation in which one gene finder correctly predicts one exon of a gene while another gene finder correctly predicts another, but neither program predicts the entire gene correctly).

Predicting individual exons for later use by an exon-assembly process poses the question of how best to settle the tradeoff between sensitivity and specificity. Many, if not most, exon-finding approaches require that the user or designer impose a scoring threshold below which a putative exon is not reported. In situations in which a later automated exon-assembly process is to be performed, a reasonably liberal threshold would presumably be of greatest value, so as to avoid limiting sensitivity. In a similar vein, one might view an *ensemble* of exon predictions much like a "particle cloud" in statistical physics, in which a particle's position is not precisely defined, but is instead characterized by a probability distribution. In a similar way, one or more exon finders may be used to induce a probability distribution on the set of all possible *open reading frames* (i.e., possible coding exons) in a sequence. To the extent that an exon finder cannot identify exact exon boundaries with absolute certainty (e.g., in cases of alternative splicing affecting the choice of either 5' or 3' splice site), some form of "exon cloud" representation may be appropriate so as not to unduly constrain a downstream exon-assembly process. Because optimal exon assembly in the case of genes with multiple isoforms is not yet a solved problem, such an ensemble-based approach to exon prediction may indeed be a promising starting point. As our knowledge about splicing regulatory factors and their cis-regulatory sequences increases (see, e.g., [43]), we can use information about, e.g., their expression values as evidence to infer condition-specific isoforms.

4.2 A Greater Role for Machine Learning

The redefinition of the gene-finding problem via the decoupling of exon finding from the later assembly of exons into one or more isoforms for each putative gene would in some ways seem to permit a greater role for alternative machine-learning approaches in the gene prediction process. Although a number of machine learning methods have been utilized within gene finders in the past (e.g., decision trees in *GlimmerM* [44]; neural networks in *GRAIL* [45]), the newest generation of gene-finding systems are based primarily on Markov models and generally do not incorporate any other machine learning algorithms. One obstacle to the greater utilization of other machine learning methods in gene finding appears to be the fundamental mismatch between the *classification-oriented* formulation of many machine-learning algorithms (at least the more popular ones such as support vector machines and the like) and the *parsing-oriented* interface of HMMs provided by Viterbi decoding. Because alternative splicing was for a number of years considered a rare exception to the one-gene-one-protein "rule," the single-parse approach enforced by Viterbi decoding became well entrenched in the gene-finding field. Exon finding, on the other hand, permits a very natural interpretation within the classification framework: given an open reading frame, an exon finder aims to accurately classify the interval as being an exon (class 1) or not being an exon (class −1).

Reformulating the problem as one of classification would permit designers of exon-finding software to draw more fully on the vast body of research from the machine-learning field. In particular, the use of maximum discrimination classifiers may produce appreciable accuracy gains as compared to the standard MLE-trained Markov models which currently dominate the field. This in turn highlights yet another advantage of a move away from the MLE+Viterbi strategy for whole-gene prediction, which as we noted earlier can be characterized as sub-optimal in certain regards.

A particularly popular machine-learning method, *support vector machines* (*SVM*s) [46], has been applied to the problems of exon prediction [47], start codon prediction [48], splice site prediction [49], and the prediction of specific forms of alternative splicing [39]. The discriminative nature of SVMs and the high

accuracy rates which have been observed in a number of applications suggest that further investigations into their use for gene and exon prediction may indeed be worthwhile.

4.3 Focus on Integrative Methods

As we noted earlier, the *ad hoc* methods exemplified by so-called "combiner" systems have proven in some cases to be exceptionally effective at producing highly accurate gene predictions, though it seems obvious that much of the advantage enjoyed by these systems derives not so much from their *ad hoc* nature as from their access to multiple forms of evidence (e.g., homology evidence, known proteins, other gene predictions) in making informed decisions regarding the most likely exonic structure for a gene. Despite the success of integrative approaches utilizing all available evidence, much attention in the field remains focused on systems utilizing only limited forms of evidence—e.g., nucleotide-based conservation in the case of PhyloHMMs and other comparative gene finders. A greater emphasis on the further development of integrative approaches to computational gene prediction may thus be useful, though it is acknowledged that in the case of genomes for which little additional evidence besides the primary genomic sequence is available, the advantage of integrative approaches dwindles.

4.4 Interoperability

Yet another possible avenue for advancing the state of the art in computational gene finding is through the use of explicit graph-based representations of genome content. Recall from section 2.1 our definition of a parse graph as a directed acyclic graph in which individual vertices represent putative splice sites and start/stop codons, and edges denote putative exons, introns, and intergenic regions. While not all gene finders explicitly construct such a graph, it is arguably the case that most, if not all, state-of-the-art whole-gene prediction systems construct such a graph implicitly during their processing of the input sequence. For many of these systems, at the point in their decoding algorithms (whether Viterbi or otherwise) when they select an optimal predecessor signal for linking into the "trellis" which is later used to retrace the optimal parse, if the potential predecessors of the current signal are instead linked to the current signal via a weighted edge (with some function of each predecessor's inductive score serving as the weight), then a parse graph would be automatically induced, and could be emitted by the program in addition to (or even instead of) the gene prediction corresponding to the optimal parse.

Such weighted parse graphs could be immensely useful for later re-processing, especially as additional evidence becomes available which was not present at the time the gene finder was originally run. Parse graphs from multiple gene finders (perhaps based on different training sets or utilizing different classes of model) could conceivably be combined with each other and/or with additional evidence (e.g., homology evidence, expression evidence, etc.) to produce a re-weighted graph that may permit more accurate decoding by virtue of the integrative nature of the graph's construction. Decoding of (i.e., extracting a gene prediction from) parse graphs can be done very simply and efficiently using a specialized shortest-path algorithm entirely anologous to Viterbi decoding [6]. Given a standard file format for the storage of such graphs, decoding of any graph could then be performed by a "universal decoder" program, which need not be aware of the actual methods employed in weighting any particular graph. Given the existence of such a "universal decoder," the implementation of a decoder in any given graph-emitting gene finder then becomes unnecessary, since the universal decoder may be applied to the emitted graph. Were such a graph-based interface to be adopted by a sufficient number of gene-finding systems, entire pipelines may conceivably be constructed in which the graphs from one or more gene finders are subjected to any number of re-weighting processes to incorporate additional information such as the existence of *genomic repeats* [50] or other genome-level features not commonly utilized by the primary gene-finding programs, or which were not available when the programs were trained. The last stage in such a pipeline would presumably involve the use of a graph-based decoder to extract one or more gene predictions.

The utility of a graph-based representation for the identification of alternative splicing should be fairly obvious. Indeed, graph-based methods for the identification of alternative splicing have already been proposed, though not in an overtly Markovian setting [51]. In our own research we have observed a tendency for our graph-based gene finders to often rank the "correct" gene parse very highly, while ranking another, incorrect parse only slightly higher, so that were the program to emit the top N parses, for some

reasonably small *N*, instead of the single highest-scoring parse, the correct parse would very often be among the top *N*. Because most state-of-the-art eukaryotic gene finders emit only the single highest-scoring parse, the "correct" parse (which might be recognized by a human annotator as correct, due to his or her access to additional evidence) is effectively lost. Methods for sampling parses from an HMM have been explored, and their possible utility to the detection of alternative splicing suggested [38], though the actual adoption of these methods by mainstream gene finders has for the most part not occurred. The proposed practice of emitting an entire parse graph (after applying a reasonable amount of pruning so as to keep the size of the graph manageable while eliminating very unlikely parses) may be viewed as an extreme variant of the sampling approach.

Finally, we would speculate that the availability of pre-computed parse graphs for a large number of organisms in some publicly-available repository—much like the precomputed whole-genome alignments maintained at such sites as the UCSC [52]—may prove useful in enabling researchers to re-analyze genomes at a later date when additional evidence becomes available, without having to deal with the often vexing problem of re-aquiring an older gene finder which had been used in an earlier analysis, or even having to recompile old, possibly poorly-maintained source code in order to run such programs on newer assemblies of a previously annotated genome.

Yet other advantages to graph-based gene prediction conceivably exist which we have not here enumerated. Unless and until a sufficient number of gene-finding software systems adopt such an interface, these advantages will of course prove elusive.

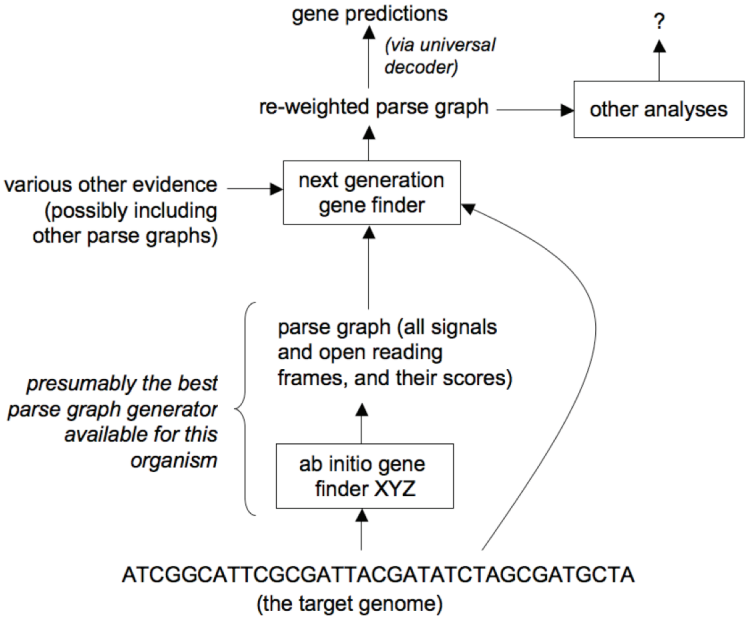

Fig. 9. Some possible uses of parse graphs as a data interchange format for computational gene prediction. Graphs produced via one gene finder may be re-weighted by other downstream programs through the incorporation of additional evidence. Eventually a graph may be supplied to a "universal decoder" to extract an optimal parse. Source: Majoros WH, *Methods for Computational Gene Prediction*, Cambridge University Press (forthcoming), reproduced with permission.

4.5 Improved Evaluation Protocols

It is an unfortunate (and often quite vexing) fact that the unbiased evaluation of gene-finder accuracy can often be very difficult to achieve. Sustained progress in any field depends to a significant degree on our ability to accurately measure progress when it is made. In the case of gene prediction, verification of predicted genes in the laboratory can be rather expensive, so that accuracy assessments are most often made by applying a new (or newly retrained or modified) program to a "test set" of genes for which the intron-exon structures are more-or-less known. Unfortunately, many genes for which we believe we know the "correct" intron-exon structure may in fact be alternatively spliced, so that the predictions obtained for a

particular locus which do not agree with the known structure of the gene may in fact match a valid, but unknown, isoform for that gene. In other cases, the "known" structure of a test gene may in fact derive from an earlier gene prediction which had been elevated to "known gene" status by an over-eager human annotator; a number of these "hypthetical" gene structures may in fact be false, again distorting our assessment of the predictive accuracy of a new gene finder when it is tested against these annotated gene structures. In the case of combiner-type programs, a further possibility for bias their evaluation exists—namely, the fact that many gene annotations in curated gene sets derive from annotation pipelines that are effectively combiner programs themselves, so that a combiner program under evaluation is effectively assessed by the degree to which the program agrees with some other combiner-like program upon which the human annotators (if any) heavily depended during genome annotation.

In order to improve this situation, a set of standardized gene sets—more than one, and ideally more than a few—need to be generated and rigorously maintained as new isoforms of existing genes are discovered. Such standard test sets should come from a variety of organisms, and should also be accompanied by corresponding training sets. Large-scale gene-finder competitions (e.g., *GASP* [53], *EGASP* [16]) whichattempt to evaluate and rank sets of gene finders on a common test set generally do not (and, out of practical reasons, typically cannot) control for the difference in training sets used by the authors of the various programs, even though it has been well-documented that the details of the training regime applied to a particular gene finder can significantly affect the accuracy of the resulting system [23]. More generally, the practice of comparing different gene-finding algorithms by applying completely different software systems embedding those approaches to a common test set fails to account for the many minute modeling decisions which are made by different software authors in implementing their highly complex software systems. Thus, a comparison between program X implementing a model of type M_X and a program Y implementing a different class of model M_Y may be so severely influenced by implementation details of the two software systems as to invalidate, or at least distort, any conclusions which are drawn about the fundamental capabilities of methods M_X and M_Y. The ideal scenario for comparing algorithmic and modeling approaches would involve the implementation of the alternative approaches within the same software code-base, so that differences in accuracy between the different versions of a single software system utilizing different gene-finding strategies may be less influenced by implementation details (e.g., [17]); ideally, such single-code-base experiments should be replicated across several independently-developed code-bases. The availability of larger numbers of open-source gene-finding software systems will hopefully make the latter types of experiments more feasible.

5. Summary and Conclusions

We have reviewed the major approaches currently in popular use for automated gene prediction in eukaryotic DNA. While much progress has certainly been made over the past two decades in building accurate gene-parsing systems, much room yet remains for progress. We have enumerated a number of shortcomings inherent in current state-of-the-art systems, and suggested a number of very broad avenues for possible future research. We have focused in particular on the existence of alternative splicing in mammalian genomes, since the existence of potentially many uncharacterized alternative splice forms in human genes poses a potential barrier to biomedical advances aimed at improving human health. To the extent that alternative splicing is still not adequately addressed by current gene-finding systems, the need for creative proposals for the advancement of the field should be manifestly clear.

References

1. Davuluri RV, Grosse I, Zhang MQ (2001) Computational identification of promoters and first exons in the human genome. *Nature Genetics* 29:412-417.
2. Viterbi A (1967) Error bounds for convolutional codes and an assymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 260-269.
3. Dempster A, Laird N, Rubin D (1977) Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society (Series B)* 39:1–38.

4. Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77:257-286.
5. Kulp D, Haussler D, Reese M, Eeckman F (1996) A generalized hidden Markov model for the recognition of human genes in DNA. *ISMB '96*.
6. Majoros WM, Pertea M, Delcher AL, Salzberg SL (2005) Efficient decoding algorithms for generalized hidden Markov model gene finders. *BMC Bioinformatics* 6:16.
7. Salzberg SL, Pertea M, Delcher AL, Gardner MJ, Tettelin H (1998) Interpolated Markov models for eukaryotic gene finding. *Genomics* 59:24-31.
8. Staden R (1984) Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Research* 12:505-519.
9. Zhang MQ, Marr TG (1993) A weight array method for splicing signal analysis. *Computer Applications in the Biosciences* 9:499-509.
10. Altschul SF, Madden TL, Schaffer AA, Zhang J, Anang Z, Miller W, Lipman DJ (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research* 25:3389-3402.
11. Alexandersson M, Cawley S, Pachter L (2003) SLAM: Cross-species gene finding and alignment with a generalized pair hidden Markov model. *Genome Research* 13:496-502.
12. Majoros WM, Pertea M, Salzberg SL (2005) Efficient implementation of a generalized pair hidden Markov model for comparative gene finding. *Bioinformatics* 21:1782-1788.
13. Felsenstein J (1981) Evolutionary trees from DNA sequences. *Journal of Molecular Evolution* 17:368-376.
14. Durbin R, Eddy S, Krogh A, Mitchison G (1998) *Biological sequence analysis*. Cambridge University Press.
15. Siepel A, Haussler D (2004) Computational identification of evolutionarily conserved exons. *RECOMB'04*, March 27-31, 2004, San Diego.
16. Guigó R, Flicek P, Abril JF, Reymond A, Lagarde J, Denoeud F, Antonarakis S, Ashburner M, Bajic VB, Birney E, Castelo R, Eyras E, Gingeras TR, Harrow J, Hubbard T, Lewis S, Ucla C, Reese MG (2006) EGASP: The human ENCODE genome annotation assessment project. *Genome Biology* 7(Suppl 1):S2.
17. Allen JE, Majoros WH, Pertea M, Salzberg SL (2006) JIGSAW, GeneZilla, and GlimmerHMM: puzzling out the features of human genes in the ENCODE regions. *Genome Biology 7(Suppl 1):S9*.
18. Bahl LR, Brown PF, de Souza PV, Mercer RL (1986) Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing* 1986, pp 49-52.
19. Reichl W, Ruske G (1995) Discriminative training for continuous speech recognition. In: Proceedings of the Fourth European Conference on Speech Communication and Technology (EUROSPEECH-95): 18-21 September 1995; Madrid. Amsterdam: Institute of Phonetic Sciences. pp 537-540.
20. Normandin Y (1996) Maximum mutual information estimation of hidden Markov models. In: *Automatic Speech and Speaker Recognition*. Lee C-H, Soong FK, Paliwal KK (eds). Klewer Academic Publishers, Norwell. pp 58-81.
21. Krogh A (1997) Two methods for improving performance of an HMM and their application for gene finding. In: *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*. Gaasterland T, Karp P, Karplus K, Ouzounis C, Sander C, Valencia A (eds). American Association for Artificial Intelligence. pp 179-186.
22. Gross SS, Brent MR (2005) Using multiple alignments to improve gene prediction. *RECOMB'05*. pp 374-388.
23. Majoros WM, Salzberg SL (2004) An empirical analysis of training protocols for probabilistic gene finders. *BMC Bioinformatics* 5:206.
24. Vinson J, DeCaprio D, Luoma S, Galagan JE (2006) Gene prediction using conditional random fields (abstract). In: *The Biology of Genomes*, Cold Spring Harbor Laboratory, New York, May 10-14, 2006.
25. Culotta A, Kulp D, McCallum A (2005) Gene prediction with conditional random fields. *Technical Report UM-CS-2005-028*. University of Massachusetts, Amherst.
26. Fariselli P, Martelli PL, Casadio R (2005) The posterior-Viterbi: a new decoding algorithm for hidden Markov models. *BMC Bioinformatics* 6 Suppl 4:S12.
27. Käll L, Krogh A, and Sonnhammer ELL (2005) An HMM posterior decoder for sequence feature prediction that includes homology information. *Bioinformatics* **21** Suppl. 1, i251-i257.
28. Stanke M, Waack S (2003) Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics* 19:II215-II225.
29. Korf I (2004) Gene finding in novel Genomes. *BMC Bioinformatics* 5:59.
30. Castellano S, Lobanov AV, Chapple C, Novoselov SV, Albrecht M, Hua D, Lescure A, Lengauer T, Krol A, Gladyshev VN, Guigó R (2005) Diversity and functional plasticity of eukaryotic selenoproteins: Identification and characterization of the SelJ family. *Proc Natl Acad Sci* 102:16188–16193.
31. Delcher A, Harmon D, Kasif S, White O, Salzberg SL (1999) Improved microbial gene identification with GLIMMER. *Nucleic Acids Research* 27:4636-4641.

32. Shmatkov AM, Melikyan AA, Chernousko FL, Borodovsky M (1999) Finding prokyarotic genes by the 'frame-by-frame' algorithm: targeting gene starts and overlapping genes. *Bioinformatics* 15:874-886.
33. McCauley S, Hein J (2006) Using hidden Markov models and observed evolution to annotate viral genomes. *Bioinformatics* 22:1308-1316.
34. Misra S, Crosby MA, Mungall CJ, Matthews BB, Campbell KS, Hradecky P, Huang Y, Kaminker JS, Millburn GH, Prochnik SE, Smith CD, Tupy JL, Whitfied EJ, Bayraktaroglu L, Berman BP, Bettencourt BR, Celniker SE, de Grey AD, Drysdale RA, Harris NL, Richter J, Russo S, Schroeder AJ, Shu SQ, Stapleton M, Yamada C, Ashburner M, Gelbart WM, Rubin GM, Lewis SE (2002) Annotation of the Drosophila melanogaster euchromatic genome: a systematic review. Genome Biology 3:RESEARCH0083.
35. Thanaraj TA, Stamm S, Clark F, Riethoven JJM, Le Texier V, Muilu J (2004) ASD: the Alternative Splicing Database. *Nucleic Acids Research* 32:D64-D69.
36. Wojtowicz WM, Flanagan JJ, Millard SS, Zipursky SL, Clemens JC (2004) Alternative splicing of Drosophila Dscam generates axon guidance receptors that exhibit isoform-specific homophilic binding. *Cell* 118:619-33.
37. Parra G, Reymond A, Dabbouseh N, Dermitzakis ET, Castelo R, Thomson TM, Antonarakis SE, Guigo R (2006) Tandem chimerism as a means to increase protein complexity in the human genome. *Genome Research* 16:37-44.
38. Cawley SE, Pachter L (2003) HMM sampling and applications to gene finding and alternative splicing. *ECCB 2003*:36-41.
39. Dror G, Sorek R, Shamir R (2004) Accurate identification of alternatively spliced exons using support vector machines. *Bioinformatics* 21:897-901.
40. Yeo GW, Van Nostrand E, Holste D, Poggio T, Burge CB (2005) Identification and analysis of alternative splicing events conserved in human and mouse. *PNAS* 102:2850-2855.
41. Rätsch G, Sonnenburg S, Schölkopf B (2005) RASE: recognition of alternatively spliced exons in C.elegans. *Bioinformatics* 21 Suppl 1:i369-377.
42. Ohler U, Shomron N, Burge CB (2005) Recognition of unknown conserved alternatively spliced exons. *PLoS Computational Biology* 1:113-22.
43. Wang Z, Rolish ME, Yeo G, Tung V, Mawson M, Burge CB (2004) Systematic identification and analysis of exonic splicing silencers. Cell 119:831-845.
44. Pertea M, Salzberg SL (2002) Computational gene finding in plants. *Plant Molecular Biology* 48:49-48.
45. Uberbacher EC, Mural RJ (1991) Locating protein coding regions in human DNA sequences using a multiple-sensor neural network approach. *PNAS* 88:11261-11265.
46. Vapnik V (1998) *Statistical Learning Theory*. John Wiley and Sons.
47. Jaakkola TS, Haussler D (1999) Exploiting generative models in discriminative classifiers. *Advances in Neural Information Processing Systems* 11:487-493.
48. Zien A, Rätsch G, Mika S, Scholkopf B, Lengauer T, Muller K-R (2000) Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics* 16:799-807.
49. Sun YF, Fan XD, Li YD (2003) Identifying splicing sites in eukaryotic RNA: support vector machine approach. *Comput Biol Med.* 33:17-29.
50. Bedell JA, Korf I, Gish W (2000) MaskerAid: a performance enhancement to RepeatMasker. *Bioinformatics* 16:1040-1041.
51. Heber S, Alekseyev M, Sze SH, Tang H, Pevzner PA (2002) Splicing graphs and EST assembly problem. *Bioinformatics* 18 Suppl 1:S181-8.
52. Karolchik D, Baertsch R, Diekhans M, Furey TS, Hinrichs A, Lu YT, Roskin KM, Schwartz M, Sugnet CW, Thomas DJ, Weber RJ, Haussler D, Kent WJ (2003) The UCSC genome browser database. *Nucleic Acids Research* 31:51-54.
53. Reese MG, Eeckman FH, Kulp D, Haussler D (1997) Improved splice site detection in Genie. *Journal of Computational Biology* 4:311-323.