Comparative methods for gene structure prediction in homologous sequences

Christian N. S. Pedersen*† Tejs Scharling*

July 2, 2002

Abstract

The increasing number of sequenced genomes motivates the use of evolutionary patterns to detect genes. We present a series of comparative methods for gene finding in homologous prokaryotic or eukaryotic sequences. Based on a model of legal genes and a similarity measure between genes, we find the pair of legal genes of maximum similarity. We develop methods based on genes models and alignment based similarity measures of increasing complexity, which take into account many details of real gene structures, e.g. the similarity of the proteins encoded by the exons. When using a similarity measure based on an exiting alignment, the methods run in linear time. When integrating the alignment and prediction process which allows for more fine grained similarity measures, the methods run in quadratic time. We evaluate the methods in a series of experiments on synthetic and real sequence data, which show that all methods are competitive but that taking the similarity of the encoded proteins into account really boost the performance.

1 Introduction

At the molecular level a gene consists of a length of DNA which encodes a protein, or in some cases, a tRNA or rRNA molecule. A gene on a prokaryotic genomic sequence is a stretch of nucleotides flanked by start and stop codons. A gene on a eukaryotic genomic sequence is also a stretch of nucleotides flanked by start and stop codons, but it is further divided into an alternating sequence of blocks of coding nucleotides (exons) and non-coding nucleotides (introns). Each intron starts at a donor site and ends at an acceptor site.

*Bioinformatics Research Center (BiRC), www.birc.dk, funded by Aarhus University Research Foundation, Department of Computer Science, University of Aarhus, Ny Munkegade, 8000 Århus C, Denmark. E-mail: {cstorm,tejs}@birc.dk

†Basic Research In Computer Science (BRICS), Center of the Danish National Research Foundation, Department of Computer Science, University of Aarhus, Ny Munkegade, 8000 Århus C, Denmark.

Identifying the location and structure of genes in the genome of an organism is an important task, and a lot of work have been devoted to develop efficient and good computational gene finders. Many methods focus on constructing signal sensors for identifying e.g. possible acceptor and donor site positions in eukaryotic sequences, and context sensors for classifying smaller sequences as being e.g. coding or non-coding. Constructing such signal and context sensors based on statistics of known genes, e.g. nucleotides frequencies or shared motifs, have lead to various methods, see [6] for an overview. Neural networks have been successfully used in signal sensors, e.g. NetGene [4], and hidden Markov models have been used in context sensors, and to make sure that the predicted gene structures obey a proper gene syntax, e.g. HMMgene [14] and Genscan [5]. The increasing number of sequenced genomes has lead to methods which use sequence homology and evolutionary patterns as context sensors and guidelines for identifying genes. These comparative gene finders search for genes by comparing homologous DNA sequences looking for similar segments which can be assembled into genes, or by using proteins from related organisms to search for gene structures which encode similar proteins, see e.g. [1, 2, 3, 8, 13, 16].

In this paper we consider a simple but flexible approach to comparative gene finding: Given two homologous DNA sequences, we identify the most likely pair of orthologous genes based on a model of legal gene structures and a similarity measure between genes. For eukaryotic gene pairs, we furthermore identify the most likely exon structures. We develop methods based on gene models and alignment based similarity measures of increasing complexity which allow us to take into account many details of real gene structures that are often neglected in comparative methods. E.g. that insertions and deletions in exons do not only happen at codon boundaries, that stop codons do not occur in exons, and that improper modeling of insertion and deletions in exons may produce stop codons. We develop methods where the gene similarity measure is based on an existing alignment of the sequences, as well as methods where the gene similarity measure is the optimal score of an alignment of a pair of legal genes, which is computed as an integral part of the prediction process.

Using an existing alignment is simple but only allows us to assign similarity to legal gene pairs which are aligned by the given alignment. Computing the alignments of legal gene pairs as an integral part of the prediction process is more complicated but allows us to assign similarity of all legal gene pairs. In both cases it is possible to employ ideas from existing alignment score functions, which we use this to construct a gene similarity measure that explicitly models that the concatenated exons encode a protein by using a DNA- and protein-level score function as proposed by Hein in [10]. All our methods can be implemented using dynamic programming such that gene finding using a similarity measure based on an existing alignment takes time proportional to the number of columns in the given alignment, and gene finding using a similarity measures based on integrating the alignment and prediction process takes time $O(nm)$, where n and m are the lengths of the two sequences.

To evaluated the quality of the develop methods, we have performed a series of experiments on both simulated and real pairs of eukaryotic sequences which show that all methods are competitive. The best results are obtained when integrating the alignment and prediction process using a similarity measure which takes the encoded proteins into account. In all the experiments we used very simple pattern based signal sensors to indicate the possible start and stop positions of exons and introns. An increased performance might be achieved by using better signal sensors, e.g. splice site detectors such as NetGene [4], as a preprocessing step. However, since our focus is to examine the quality of our pure comparative approach to gene finding, we do not consider this extension in this paper. The comparative approach can also be extended by comparison of multiple sequences which is beyond the scope of this paper.

The rest of the paper is organized as follows. In Section 2 we introduce legal gene structures and similarity measures. In Section 3 we present our basic methods for gene finding in prokaryotic and eukaryotic sequences. In Section 4 we consider a number of extensions of the basic methods. In Section 5 we evaluate our methods through a series of experiments on synthetic and real sequence data.

2 Preliminaries

Let $a = a_1 a_2 \cdots a_n$ and $b = b_1 b_2 \cdots b_m$ be two homologous genomic sequences which contain a pair of orthologous genes of unknown structure. We want to identify the most likely pair of genes based on a model of legal gene structures and a gene similarity measure between pairs of legal genes.

Model of legal gene structures An important feature in biological and computational gene identification is functional signals on the genomic sequence. Typically a signal is made up of a certain pattern of nucleotides, e.g. the start codon *atg*, and these signal flanks the structural elements of the gene. Unfortunately most signals are not completely determined by a simple two or three letter code (not even the start-codon). Various (unknown) configurations of nucleotides in the neighborhood determine if e.g. *atg* is a start codon. We leave the discussion of how to identify potential signals for Section 5, and settle here with a more abstract definition of the relevant signals. We consider four types of signals: start codons, stop codons, acceptor sites, and donor sites. Donor and acceptor sites are only relevant for eukaryotic sequences. Potential start and stop codons on sequence a are given by the indicator variables $G_{a,i}^{start}$ and $G_{a,i}^{stop}$, where $G_{a,i}^{start}$ is true when a_i is the last nucleotide of a potential

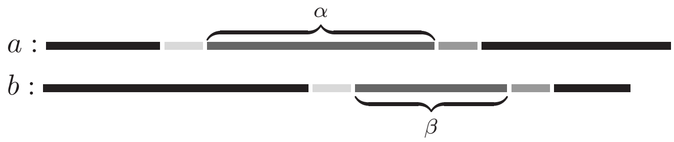

Figure 1: We want to find a pair of legal prokaryotic genes, α and β, of highest similarity. A legal prokaryotic gene is a substring flanked by certain signals. In the simplest case these signals can be modeled as just the start and stop codons.

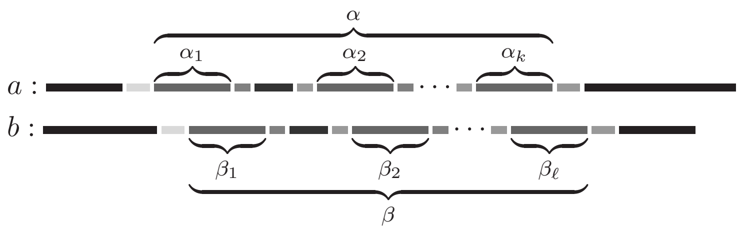

Figure 2: We want to find a pair of legal eukaryotic genes, α and β, of highest similarity, and their exon structures $\alpha_1, \ldots, \alpha_k$ and $\beta_1, \ldots, \beta_\ell$. A legal eukaryotic gene is a substring flanked by certain signals. In the simplest case these signals can be modeled as just the start and stop codons. A legal exon structure is a sequence of substrings of the gene fulfilling that the remaining substrings of the gene, the introns, all start at a donor site and end at an acceptor site.

start codon and $G^{stop}_{a,i}$ is true when a_i is the first nucleotide of a potential stop codon. Similarly, potential donor and acceptor sites are given by the indicator variables D^a_i and A^a_i, where D^a_i is true when a_i is the first nucleotide of a potential donor site, and A^a_i is true when a_i is the last nucleotide of a potential acceptor site.

A *legal prokaryotic gene* on a genomic sequence a is a substring $\alpha = a_i \cdots a_j$, where $G^{start}_{a,i-1}$ and $G^{stop}_{a,j+1}$ are true. See Figure 1. Similarly, a *legal eukaryotic gene* on a genomic sequence a is a substring $\alpha = a_i \cdots a_j$, where $G^{start}_{a,i-1}$ and $G^{stop}_{a,j+1}$ are true, that is divided into substrings which are alternating labeled intron and exon, such that each intron is a substring $a_h \cdots a_k$, for some $i \leq h < k \leq j$, where D^a_h and A^a_k are true. The *exon structure* of a eukaryotic gene α is the sequence of substrings $\alpha_1, \alpha_2, \ldots, \alpha_k$ that are labeled exons. See Figure 2. In Section 4 we refine our definition of legal gene structures to exclude gene structures which contain in-frame stop codons, and gene structures where the coding nucleotides do not constitute an integer number of codons, i.e. a number of coding nucleotides which is not a multi-plum of three.

Gene similarity measures Let $\mathrm{Sim}(\alpha, \beta)$ be a similarity measure which quantifies the similarity between two genes α and β. We use two alignment based approaches to define $\mathrm{Sim}(\alpha, \beta)$. The first approach is to use an existing alignment. Let $(\hat{a}, \hat{b})$ be an alignment of a and b, where $\hat{a}$ and $\hat{b}$ is the first and second row of the alignment matrix respectively. We say that two substrings α and β of a and b respectively are aligned by $(\hat{a}, \hat{b})$ if there is a sub-alignment $(\hat{\alpha}, \hat{\beta})$ of $(\hat{a}, \hat{b})$, i.e. a consecutive block of columns, which is an alignment of α and β. We define the similarity of α and β, $\mathrm{Sim}(\alpha, \beta)$, as the score of the alignment $(\hat{\alpha}, \hat{\beta})$ induced by $(\hat{a}, \hat{b})$ cf. a given alignment score function. Note that this only defines the similarity of substrings which are aligned by $(\hat{a}, \hat{b})$. The advantage of defining the similarity measure based on an existing alignment is of course that we can use any existing and well proven alignment method to construct the alignment. A drawback is that it only defines the similarity of substrings aligned by $(\hat{a}, \hat{b})$, which implies that we cannot consider the similarity of all pairs of possible legal gene structures. To circumvent this problem, we as the second approach simply define the similarity of α and β as the score of an optimal alignment of α and β.

3 Methods

In this section we present methods for solving the following problem. Given two sequences $a = a_1 a_2 \cdots a_n$ and $b = b_1 b_2 \cdots b_m$, find a pair of legal genes α and β in a and b respectively such that $\mathrm{Sim}(\alpha, \beta)$ is maximized. For eukaryotic genes we also want to find their exon structures. For ease of presentation, the methods in this section do not take into account that in frame stop codons must not occur in exons, that the combined length of exons must be a multiple of three, that a codon can be split by an intron, or that the exons encode a protein. These extensions are considered in Section 4.

To define $\mathrm{Sim}(\alpha, \beta)$ we use a column based alignment score which allows us to assign scores to alignments of α and β by summing the score of each column in the alignment. The assumption of coding regions being well-conserved and non-coding regions being divergent can be expressed in two probabilistic models, one for coding alignment columns $P\binom{x}{y} = p_{xy}$, and one for non-coding alignment columns $P\binom{x}{y} = q_x q_y$, where q_x is the overall probability of observing x in an alignment. Note that an alignment column can represent an insertion or deletion, i.e. either x or y can be a gap. Since the chance of inserting or deleting a nucleotide depends on the background frequency of the nucleotide but not its kind, we set $p_{x-} = p_{-x} = \sigma q_x$, for some constant σ. We formulate these models in terms of their log-odds ratio, and use this as a scoring scheme.

We observe that the log-odds ratio of a non-coding column is $-\log \frac{q_x q_y}{q_x q_y} = 0$. The log-odds ration for a coding column consisting of two nucleotides is de-

$$
\begin{aligned}
\hat{a} &= a\ g\ t\ c\ \overbrace{a\,t\,g}^{\text{start}}\ \overbrace{c\ a\ a\ t\ a\ c}^{\text{exon}}\ \overbrace{g\ t\ t\ a\ g\ -\ -\ -\ -}^{\text{intron}}\ \overbrace{c\,t\,c}^{\text{exon}}\ \overbrace{t\,a\,a}^{\text{stop}}\ a\,g\,g\,-\,- \\
\hat{b} &= -\ -\ g\,c\ a\,t\,g\ c\,-\,-\,-\ t\,c\ -\,-\,-\,-\,-\,-\ g\ t\ a\ g\ g\,t\,c\ t\,g\,a\ t\,c\,a\ g\ a
\end{aligned}
$$

$$\text{log-odds ratio} = \delta_{c,c} + 3 \cdot \lambda + \delta_{a,t} + \delta_{c,c} + \gamma + 9 \cdot 0 + \gamma + \delta_{c,g} + \delta_{t,t} + \delta_{c,c}$$

Figure 3: Calculating the log-odds ratio of an alignment of two eukaryotic sequence with known gene structures. Only coding columns and switching between coding and non-coding models contribute to the score. The score of non-coding columns are zero.

scribed by $\delta_{x,y} = -\log \frac{p_{xy}}{q_x q_y}$, and a column including a gap by $\lambda = -\log \frac{\sigma}{q_-}$. Hence, the cost of a coding column is given by a substitution cost $\delta_{x,y}$ and a gap cost λ, and the cost of a non-coding column is zero. Switching between the coding and non-coding models is penalized by γ which denotes the probability (in log-odds ratio) of observing a splice-site. In Sect. 4 we consider extensions of this simple score function. Given an alignment $(\hat{a}, \hat{b})$ and a known gene structure, its log-odds ratio can be calculated by summing over columns aligning the genes, see Fig. 3.

Finding prokaryotic genes

When working with prokaryotic sequences a and b, we know that legal genes are substrings of coding nucleotides. Hence, we only have to use the coding model when computing the similarity between possible pairs of genes.

Prokaryotic Method 1 – Using an existing alignment Let $\mathcal{A} = (\hat{a}, \hat{b})$ be an alignment of a and b. A sub-alignment $\mathcal{A}_{i,j}$ from column i to j is an alignment of the substrings $\hat{a}_i \cdots \hat{a}_j$ and $\hat{b}_i \cdots \hat{b}_j$ with gaps removed. We denote these aligned substrings $\alpha = a_{\mathrm{a}(i)} \cdots a_{\mathrm{a}(j)}$ and $\beta = b_{\mathrm{b}(i)} \cdots b_{\mathrm{b}(j)}$ respectively, where $\mathrm{a}(\cdot)$ and $\mathrm{b}(\cdot)$ are functions which translate column indices in $\mathcal{A}$ to string indices in a and b respectively. We say that $\mathcal{A}_{i,j}$ is a legal sub-alignment if the aligned substrings α and β are legal genes, i.e. if $G^{start}_{\mathrm{a}(i)-1,\mathrm{b}(i)-1}$ and $G^{stop}_{\mathrm{a}(j)+1,\mathrm{b}(j)+1}$ are true. The score $\mathrm{Sim}(\hat{a}_i \cdots \hat{a}_j, \hat{b}_i \cdots \hat{b}_j)$ of $\mathcal{A}_{i,j}$ is the sum of the score of each of the $j - i + 1$ columns. We want to find a legal sub-alignment of maximum score.

We define $S(i) = \max_{h<i} \mathrm{Sim}(\hat{a}_h \cdots \hat{a}_i, \hat{b}_h \cdots \hat{b}_i)$ where $G^{start}_{\mathrm{a}(h)-1,\mathrm{b}(h)-1}$ is true, i.e. $S(i)$ is the maximum score of a subalignment ending in column i of substrings starting at legal gene start positions. Since our alignment score is column based, we can compute $S(i)$ using dynamic programming based on

the recursion:

$$
\begin{aligned}
S(i) &= \max \begin{cases} S(i-1) + \delta_{\hat{a}_i, \hat{b}_i} & \\ 0 & \text{if } G^{start}_{\mathrm{a}(i),\mathrm{b}(i)} \end{cases} \\
i'' &= \operatorname*{argmax}_i \left\{ S(i) \mid G^{stop}_{\mathrm{a}(i)+1,\mathrm{b}(i)+1} \right\}
\end{aligned}
\tag{1}
$$

where i'' is the index of the last column in a legal sub-alignment of maximal score. By finding the maximum $i' < i''$ where $S_{i'} = 0$ and $G^{start}_{\mathrm{a}(i'),\mathrm{b}(i')}$ is true, we have that $\mathcal{A}_{i'+1,i''}$ is a legal sub-alignment of maximal score, and that the corresponding pair of substrings is a pair of legal genes of maximum similarity. Computing $S(i)$ and finding i' and i'' takes time proportional to the number of columns in $\mathcal{A}$, i.e. the total running time is $O(n+m)$.

Prokaryotic Method 2 – Maximizing over all possible gene pairs We define the similarity $\mathrm{Sim}(\alpha, \beta)$ between legal genes α and β as the score of an optimal alignment of α and β using the column based alignment score function introduced above. Finding a pair of legal genes of maximum similarity is similar to the local alignment problem [18] with the additional condition that we only maximize the similarity over pairs of substrings which are legal genes.

We define $S(i,j) = \max_{h<i,k<j} \mathrm{Sim}(a_h \cdots a_i, b_k \cdots b_j)$ where $G^{start}_{h-1,k-1}$ is true, i.e. $S(i,j)$ is the maximum similarity of a pair of substrings ending in positions i and j and starting at legal gene start positions. Since our similarity measure is a column based alignment score, we can compute $S(i,j)$ using dynamic programming based on the recursion:

$$
\begin{aligned}
S(i,j) &= \max \begin{cases} S(i-1,j-1) + \delta_{a_i,b_j} & \\ S(i-1,j) + \lambda & \\ S(i,j-1) + \lambda & \\ 0 & \text{if } G^{start}_{i,j} \end{cases} \\
(i'',j'') &= \operatorname*{argmax}_{i,j} \left\{ S(i,j) \mid G^{stop}_{i+1,j+1} \right\}
\end{aligned}
\tag{2}
$$

where (i'', j'') are the last positions in a pair of legal genes of maximum similarity. We can find the start positions of these genes by back-tracking through the recursive computation of $S(i'', j'')$ searching for the maximum indices $i' < i''$ and $j' < j''$ where $S(i', j') = 0$ and $G^{start}_{i',j'}$ is true. The substrings $a_{i'+1} \cdots a_{i''}$ and $b_{j'+1} \cdots b_{j''}$ is by construction legal genes of maximum similarity. Computing $S(i,j)$ and finding (i'', j'') takes time $O(nm)$ using dynamic programming, back-tracking through the recursive computation to find (i', j') takes time $O(n+m)$, i.e. the total running time is $O(nm)$. The space consumption is $O(nm)$ but can be reduced to $O(n+m)$ using the technique in [12].

Finding eukaryotic genes

When working with eukaryotic sequences a and b, we not only want to find a pair of legal genes α and β of highest similarity, but also their most likely exon structures. We solve this problem by making it possible to switch between the coding and non-coding score model at splice-sites when computing the similarity between possible pairs of genes.

Eukaryotic Method 1 – Using an existing alignment Let $\mathcal{A} = (\hat{a}, \hat{b})$ be an alignment of a and b. A legal sub-alignment $\mathcal{A}_{i,j}$ is defined as above. The score of a column in $\mathcal{A}_{i,j}$ is computed by the coding or non-coding score model cf. these rules: (1) the score of the first column, i.e. column i, can be computed using the coding model. (2) the score of column k can be computed using the score model used to compute the score of column $k-1$. (3) the score of column k can be computed in the non-coding model if $D^a_{\mathrm{a}(k)}$ and $D^b_{\mathrm{b}(k)}$ are true, i.e. if we are at a donor site. (4) the score of column k can be computed in the coding model if $A^a_{\mathrm{a}(k)-1}$ and $A^b_{\mathrm{b}(k)-1}$ are true, i.e. if we have just passed an acceptor site.

The score $\mathrm{Sim}(\hat{a}_i \cdots \hat{a}_j, \hat{b}_i \cdots \hat{b}_j)$ of $\mathcal{A}_{i,j}$ is defined as the sum of the score of each of the $j + i - 1$ columns maximized over all possible divisions into coding and non-coding columns. The division of $\mathcal{A}_{i,j}$ into coding or non-coding columns which yields the maximum score also yields the exon structures of the aligned substrings. We want to find a legal sub-alignment of maximum score and the corresponding division into coding and non-coding columns. We define $S(i) = \max_{h<i} \mathrm{Sim}(\hat{a}_h \cdots \hat{a}_i, \hat{b}_h \cdots \hat{b}_i)$ where $G^{start}_{\mathrm{a}(h)-1,\mathrm{b}(h)-1}$ is true, and column i is either a coding column or the last column in an intron, i.e. $S(i)$ is the maximum score of a sub-alignment ending in column i of substrings starting at legal gene start positions and ending in an exon or at an acceptor site. We can compute $S(i)$ using dynamic programming based on the recursions:

$$
\begin{aligned}
I(i) &= \max \begin{cases} \mathcal{I}: & I(i-1 \\ \mathcal{I}: & S(i-1) + \gamma \quad \text{if } D^a_{\mathrm{a}(i)} \wedge D^b_{\mathrm{b}(i)} \end{cases} \\
S(i) &= \max \begin{cases} \mathcal{E}: & S(i-1) + \delta_{\hat{a}_i,\hat{b}_i} \\ \mathcal{I}: & I(i-1) + \gamma \qquad \text{if } A^a_{\mathrm{a}(i)} \wedge A^b_{\mathrm{b}(i)} \\ & 0 \qquad\qquad\qquad \text{if } G^{start}_{\mathrm{a}(i),\mathrm{b}(i)} \end{cases} \\
i'' &= \operatorname*{argmax}_i \left\{ S(i) \mid G^{stop}_{\mathrm{a}(i)+1,\mathrm{b}(i)+1} \right\}
\end{aligned}
\tag{3}
$$

where i'' is the index of the last column in a legal sub-alignment of maximal score. Recall that the constant γ is the probability (in log-odds ratio) of observing a splice-site. We can find the start positions of these genes by back-tracking through the recursive computation of $S(i'')$ searching for the maximum index $i' < i''$ where $S(i') = 0$ and $G^{start}_{\mathrm{a}(i'),\mathrm{b}(j')}$ is true. By construction the sub-alignment $\mathcal{A}_{i'+1,i''}$ is a legal sub-alignment of maximal score, and the corresponding pair of substrings is a pair of legal genes of maximum similarity. The exon structures of these genes is determined while back-tracking, where the annotation, $\mathcal{E}$ or $\mathcal{I}$, of each applied recursion indicates if the corresponding column in $\mathcal{A}_{i'+1,i''}$ is coding or non-coding. Computing $S(i)$ and finding i' and i'' takes time proportional to the number of columns in $\mathcal{A}$, i.e. the total running time is $O(n+m)$.

Eukaryotic Method 2 – Maximizing over all possible gene pairs We define the similarity $\mathrm{Sim}(\alpha, \beta)$ between legal genes α and β as the score of an optimal alignment of α and β, where we use the coding and non-coding score models, and make it possible to switch between them at donor and acceptor sites. Since the non-coding score model sets the score of a non-coding column to zero, we can ignore the possibility of non-coding substitution columns when computing an optimal alignment. Hence, we only have to consider alignments where non-coding nucleotides are in gap columns cf. the alignment in Figure 3. Intuitively, this approach makes it possible to skip introns by allowing free gaps starting at a donor site and ending at an acceptor site. We define $S(i,j) = \max_{h<i,k<j} \mathrm{Sim}(a_h \cdots a_i, b_k \cdots b_j)$ where $G^{start}_{h-1,k-1}$ is true and a_i and b_j are either coding or the last symbols in introns. Hence, $S(i,j)$ is the maximum similarity of a pair of substrings ending in positions i and j and starting at legal gene start positions, that would be a pair of legal genes if $G^{stop}_{i+1,j+1}$ was true. We can compute $S(i,j)$ using dynamic programming based on the recursions:

$$I^a(i,j) = \max \begin{cases} \mathcal{I}: & I^a(i-1,j) \\ \mathcal{I}: & S(i-1,j) + \gamma \quad \text{if } D^a_i \end{cases}$$

$$I^b(i,j) = \max \begin{cases} \mathcal{I}: & I^a(i,j-1) \\ \mathcal{I}: & S(i,j-1) + \gamma \quad \text{if } D^b_j \end{cases}$$

$$S(i,j) = \max \begin{cases} \mathcal{E}: & S(i-1,j-1) + \delta_{a_i,b_j} \\ \mathcal{E}: & S(i-1,j) + \lambda \\ \mathcal{E}: & S(i,j-1) + \lambda \\ \mathcal{I}: & I^a(i-1,j) + \gamma & \text{if } A^a_i \\ \mathcal{I}: & I^b(i,j-1) + \gamma & \text{if } A^b_j \\ & 0 & \text{if } G^{start}_{i,j} \end{cases} \tag{4}$$

$$(i'', j'') = \operatorname*{argmax}_{i,j} \left\{ S(i,j) \mid G^{stop}_{i+1,j+1} \right\}$$

where $I^a(i,j)$ and $I^b(i,j)$ is the maximum similarity of a pair of substrings ending in positions i and j inside an intron in sequence a and b respectively, and starting at legal gene start positions. The pair (i'', j'') are the last positions in a pair of legal genes of maximum similarity. We can find the start positions of these genes by back-tracking through the recursive computation of $S(i'', j'')$ searching for the maximum indices $i' < i''$ and $j' < j''$ where $S(i', j') = 0$ and $G^{start}_{i',j'}$ is true. The substrings $a_{i'+1} \cdots a_{i''}$ and $b_{j'+1} \cdots b_{j''}$ is by construction legal genes of maximum similarity. The exon structures of these genes is determined when back-tracking where the annotation, $\mathcal{E}$ or $\mathcal{I}$, of each applied recursion indicates if the symbols in the corresponding column are coding or non-coding. The back-tracking also yields an optimal alignment of the identified gene pair. Similar to *Prokaryotic Method 2*, the time and space complexity of the method is $O(nm)$, where the space consumption can be improved to $O(n + m)$.

4 Extensions

The model of legal gene structures and the coding and non-coding score functions presented in the previous sections are deliberately made simple. However, there are some obvious extensions to both of them. The codon structure is an important feature of gene structures. The coding parts of a gene can be completely divided into subsequent triplets of nucleotides denoted codons. Each codon encodes an amino acid which evolves slower than its underlying nucleotides. If a codon is a stop codon the translation, and thus the gene, must end. Moreover, the codon structure implies that deleting a number of nucleotides which is not a multi-plum of three changes all encoded amino acids downstream of the deletion (called a frame shift). This is believed to be a very rare event. The coding score function should describe a more evolved treatment of evolutionary events, e.g. take changes on the encoded amino acids into account when scoring an event, and allow insertion/deletion of consecutive nucleotides.

Taking the encoded amino acids into account The alignment score function by Hein in [11] is a reasonable and fairly simple way of taking the protein level into account. The idea is to think of an amino acid encoded as being "attached" to the middle nucleotide of the codon encoding it. When matching two middle nucleotides a_2 and b_2, i.e. nucleotides in position two in predicted codons $a_1a_2a_3$ and $b_1b_2b_3$, the similarity between the amino acids A and B encoded by these codons is taken into account. For example, if $\delta_{a,b}$ is the similarity between two nucleotides and $\Delta_{A,B}$ is the similarity between two amino acids, then the cost of matching two middle nucleotides a_2 and b_2 in codons encoding amino acids A and B is $\delta_{a_2,b_2} + \Delta_{A,B}$, while the cost of matching any two non-middle nucleotides a_i and b_j is δ_{a_i,b_j}. By keeping track of the codon-position of the current nucleotides, we can distinguish the middle nucleotide from the other two and thus incorporate Δ.

Avoiding stop codons and obeying length requirements Keeping track of the codon-position as explained above also enables us to identify and exclude possible stop codons. Furthermore by only allowing genes to end in nucleotides that are in codon position three, we ensure that the total length of the coding regions is a multi-plum of three.

Avoiding frame shifts Frame shifts can be avoided if we only allow gaps to come in triplets. That is to consider three columns in an alignment at the time when producing gaps in coding regions. When working on a given alignment there is however the problem that frame shifts may be introduces by errors in the preceding alignment step. If we forbid these frame shifts, we might discard the true genes because of an alignment error. Instead we can penalize "frame shifts" (alignment errors) with an additional cost ϕ.

To implement the above three extensions we let $p \in \{1, 2, 3\}$ denote the codon-position of the nucleotides in the current alignment column, and let the operator $\ominus$ be defined as $p \ominus 1 = p - 1$ except $1 \ominus 1 = 3$. Adding this extra bookkeeping and incorporating the extensions to the *Prokaryotic Method 1* yields the recursion:

$$
\begin{aligned}
S_p(i) &= \max \begin{cases}
S_{p\ominus 1}(i-1) + \delta_{\hat{a}_i,\hat{b}_i} & \text{if } p \neq 2 \wedge \hat{a}_i, \hat{b}_i \text{ nucleotides} \\
S_{p\ominus 1}(i-1) + \delta_{\hat{a}_i,\hat{b}_i} + \Delta_{A_{\mathrm{a}(i)},B_{\mathrm{b}(i)}} & \text{if } p = 2 \wedge \hat{a}_i, \hat{b}_i \text{ nucleotides} \\
S_{p\ominus 1}(i-1) + \lambda + \phi & \text{if } \hat{a}_i \vee \hat{b}_i \text{ gaps} \\
S_p(i-3) + 3\lambda & \text{if } \hat{a}_{i-2} \cdots \hat{a}_i \vee \hat{b}_{i-2} \cdots \hat{b}_i \text{ gaps} \\
0 & \text{if } p = 3 \wedge G^{start}_{\mathrm{a}(i),\mathrm{b}(i)}
\end{cases} \\
i'' &= \operatorname*{argmax}_i \left\{ S_3(i) \mid G^{stop}_{\mathrm{a}(i)+1,\mathrm{b}(i)+1} \right\}
\end{aligned}
\tag{5}
$$

Adding the same three extensions to the *Eukaryotic Method 2* is strait forward. Since we produce the alignment ourself, we can avoid frame shifts by producing gap columns in triplets only. We get the following set of recursions:

$$I_p^a(i,j) = \max \begin{cases} \mathcal{I}: & I_p^a(i-1,j) \\ \mathcal{I}: & S_p(i-1,j) + \gamma \quad \text{if } D_i^a \end{cases}$$

$$I_p^b(i,j) = \max \begin{cases} \mathcal{I}: & I_p^a(i,j-1) \\ \mathcal{I}: & S_p(i,j-1) + \gamma \quad \text{if } D_j^b \end{cases}$$

$$S_p(i,j) = \max \begin{cases} \mathcal{E}: & S_{p\ominus 1}(i-1,j-1) + \delta_{a_i,b_j} & \text{if } p \neq 2 \\ \mathcal{E}: & S_{p\ominus 1}(i-1,j-1) + \delta_{a_i,b_j} + \Delta_{A_i,B_j} & \text{if } p = 2 \\ \mathcal{E}: & S_p(i-3,j) + 3\lambda & \\ \mathcal{E}: & S_p(i,j-3) + 3\lambda & \\ \mathcal{I}: & I_p^a(i-1,j) + \gamma & \text{if } A_i^a \\ \mathcal{I}: & I_p^b(i,j-1) + \gamma & \text{if } A_j^b \\ & 0 & \text{if } G_{i,j}^{start} \wedge p = 3 \end{cases}$$

$$(i'',j'') = \operatorname*{argmax}_{i,j} \left\{ S_3(i,j) \mid G_{i+1,j+1}^{stop} \right\} \tag{6}$$

The implementations of our gene finding method used for experiments in Section 5 incorporate the above three extensions plus the following two extensions.

Affine gap-cost In the probabilistic model underlying our coding score model it was assumed that gap-symbols, reflecting insertions or deletions of nucleotides through evolution, were independent. However, it is a well-known fact that insertions and deletions often involve blocks of nucleotides, thus imposing a dependence between neighboring gaps. In the alignment literature an affine gap-cost function is often used to model this aspect of evolution. Affine gap cost means that k consecutive gap-columns has similarity $\mu + \lambda \cdot k$ instead of $\lambda \cdot k$, i.e. probability $p_{\text{gap}} \cdot (p_{x,-})^k$ instead of $(p_{x,-})^k$ cf. [7]. By using the technique by Gotoh in [9] we can incorporate affine gap cost in all our methods without increasing the asymptotical time complexity.

Keeping track of codons across exon boundaries In a eukaryotic gene, a codon can be split between two exons. This implies that nucleotides in a codon not necessarily are consecutive in the sequence as there between any two nucleotides in a codon can be a number of nucleotides forming an intron. The problem of keeping track of the codon-position of the current nucleotide becomes more difficult when this is possible. It can be solved by doing some additional bookkeeping on the different ways to end an exon (with respect to nucleotides in an possibly unfinished codon). For example, the recursion for $I_1^a(i,j)$ can be divided into sixteen recursions $I_{1,x,y}^a(i,j)$, for $x, y \in \{a,g,t,c\}$, where each represents a different way for two exons to end with one “hanging” nucleotide each. Similarly, $I^a_{2,x_1x_2,y_1y_2}(i,j)$ represent the cases where each exon has two hanging nucleotides. Handling hanging nucleotides properly is a technically hard extension, but it only increases the running time of our methods by a constant factor.

Adding all five extensions to the *Eukaryotic Method 2* yields the set of recursions below. Again the pair (i'', j'') is the last positions in a pair of legal genes of maximum similarity and we can find the start positions of these genes by back-tracking through the recursive computation of $S(i'', j'')$.

$$
\begin{aligned}
I^a_{p,x,y}(i,j) &= \max \begin{cases} \mathcal{I}: & I^a_{p,x,y}(i-1,j) \\ \mathcal{I}: & S_{p,x,y}(i-1,j) & \text{if } D^a_i \end{cases} \\
I^b_{p,x,y}(i,j) &= \max \begin{cases} \mathcal{I}: & I^b_{p,x,y}(i,j-1) \\ \mathcal{I}: & S_{p,x,y}(i,j-1) & \text{if } D^b_j \end{cases} \\
S^{\text{ins}}_{p,x}(i,j) &= \max \begin{cases} \mathcal{E}: & S^{\text{ins}}_{p,x}(i,j-3) + 3\lambda \\ \mathcal{E}: & \max_y S_{1,x,y}(i,j-3) + 3\lambda + \mu & \text{if } p=1 \\ \mathcal{E}: & S_{2,x,b_{j-2}}(i,j-3) + 3\lambda + \mu & \text{if } p=2 \\ \mathcal{E}: & S_{3,\epsilon,\epsilon}(i,j-3) + 3\lambda + \mu & \text{if } p=3 \end{cases} \\
S^{\text{del}}_{p,y}(i,j) &= \max \begin{cases} \mathcal{E}: & S^{\text{del}}_{p,y}(i-3,j) + 3\lambda \\ \mathcal{E}: & \max_x S_{1,x,y}(i-3,j) + 3\lambda + \mu & \text{if } p=1 \\ \mathcal{E}: & S_{2,a_{i-2},y}(i-3,j) + 3\lambda + \mu & \text{if } p=2 \\ \mathcal{E}: & S_{3,\epsilon,\epsilon}(i-3,j) + 3\lambda + \mu & \text{if } p=3 \end{cases} \\
S^{\text{mch}}_{p,x,y}(i,j) &= \max \begin{cases} \mathcal{E}: & S_{3,\epsilon,\epsilon}(i-1,j-1) + \delta_{a_i,b_j} & \text{if } p=1 \\ \mathcal{E}: & \max_{x',y'} S_{1,x',y'}(i-1,j-1) + \delta_{a_i,b_j} + \Delta_{x'a_ix,y'b_jy} & \text{if } p=2 \\ \mathcal{E}: & S_{2,a_i,b_j}(i-1,j-1) + \delta_{a_i,b_j} & \text{if } p=3 \end{cases} \\
S_{p,x,y}(i,j) &= \max \begin{cases} & S^{\text{mch}}_{p,x,y}(i,j) \\ & S^{\text{ins}}_{p,x}(i,j) & \text{if } p=1 \Rightarrow y=b_j \\ & S^{\text{del}}_{p,y}(i,j) & \text{if } p=1 \Rightarrow x=a_i \\ \mathcal{I}: & I^a_{p,x,y}(i-1,j)+\gamma & \text{if } A^a_i \\ \mathcal{I}: & I^b_{p,x,y}(i,j-1)+\gamma & \text{if } A^b_j \\ & 0 & \text{if } G^{start}_{i,j} \wedge p=3 \end{cases} \\
(i'', j'') &= \operatorname*{argmax}_{i,j} \left\{ S_{3,\epsilon,\epsilon}(i,j) \mid G^{stop}_{i+1,j+1} \right\}
\end{aligned} \tag{7}
$$

To model affine gap-cost we divide S into $S^{\text{ins}}_{p,x}$, $S^{\text{del}}_{p,y}$ and $S^{\text{mch}}_{p,x,y}$ which are conditioned by the last column that must represent an insertion, a deletion or a match respectively. As in (6) we condition all recursions with the codon position p of the last coding column. But we also condition with the content of the codons "under construction". If $p = 1$ we let x and y represent the first nucleotide in the two codons under construction, i.e. $S_{1,x,y}(i,j)$ is the maximum similarity of a pair of substrings ending in positions i and j inside an exon in codon position one in sequence a and b, where x and y is the last coding nucleotide in $a[1\,..\,i]$ and $b[1\,..\,j]$ respectively. On the other hand, if $p = 2$ we let x and y be the assumed last nucleotides in the two codons under construction, i.e. $S_{2,x,y}(i,j)$ is the maximum similarity of a pair of substrings ending in positions i and j inside an exon in codon position two in sequence a and b, where x and y is the assumed next coding nucleotide a and b respectively. If $p = 3$ the codons are completed and we let both x and y be ϵ.

By keeping track of the two codons under constructions using x and y as described above, we can incorporate the protein level score function. When matching two nucleotides in codon position two ($p = 2$), we can easily consider all possible last codons by iterating over the possible choices of the first and last nucleotides. This is done in the recursion $S^{\mathrm{mch}}_{2,x,y}(i,j) = \max_{x',y'} S_{1,x',y'}(i-1, j-1) + \delta_{a_i,b_j} + \Delta_{x' a_i x, y' b_j y}$. Keeping track of the two codons under construction also make it possible to avoid stop codons in exons.

5 Experiments

To evaluate the quality of our comparative gene finding method, we examine the performance of eukaryotic gene finding based on *Eukaryotic Method 1* and *Eukaryotic Method 2* including the extensions of Section 4, on two different data sets; a set of simulated sequence pairs, and a set of human-mouse sequence pairs. The performance of the two methods is evaluated using two different alignment score functions; one score function which only considers the nucleotide level, and one score function which also incorporates the protein level. In total we thus examine four different implementations of eukaryotic gene finding. The implementations are available as GenePair via `http://www.birc.dk/Software/`.

Data Set 1 – Simulated sequences The first data set consists of a time series of simulated sequence pairs. For steps of 0.1 in the time interval $[0\,;2]$ we simulate 20 sequence pairs. Each simulation starts with a random generated DNA-sequence (1000 nucleotides in length) with uniform background frequencies. Four regions (total length of about 250 nucleotides) of the sequence are exons and thus bounded with appropriate signals, i.e. start/stop-codons and splice sites. Given a time parameter two copies of the sequence undergo a Jukes-Cantor substitution process and a insertion/deletion Poisson-process with geometrical length distribution. Non-synonymous mutations changing

Signal	Pattern	Definition
Start codon	atg	$G^{start}_{x,i} = \{i \mid x_{i-3}x_{i-2}x_{i-1} = \text{atg}\}$
Stop codon	taa, tag, tga	$G^{stop}_{x,i} = \{i \mid x_{i+1}x_{i+2}x_{i+3} \in \{\text{taa,tag,tga}\}\}$
Donor site	gt	$D^x_i = \{i \mid x_i x_{i+1} = \text{gt}\}$
Acceptor site	ag	$A^x_i = \{i \mid x_{i-1}x_i = \text{ag}\}$

Table 1: Signals are defined as short patterns of nucleotides.

amino acid σ to σ' are kept with probability $\exp(-d(\sigma, \sigma')/c)$, where d is the Grantham distance matrix from PAML [19]. The parameter c lets us control the dN/dS ratio, i.e. the difference between the evolutionary patterns in coding and non-coding regions. Mutations that change signals or introduce frame shifts are not allowed. Simulation parameters are chosen such that evolution time 1 yields coding regions with a mutation rate of 0.25 per nucleotide, a dN/dS ratio of 0.20, and 20 per cent gaps.

Data Set 2 – Human and mouse sequences The second data set is the ROSETTA gene set from [2], which consists of 117 orthologous sequence pairs from human and mouse. The average length of the sequences is about 4750 nucleotides of which 900 are coding, and on average there are four exons per gene. Estimates in [17] give a dN/dS ratio of 0.12 and a mutation rate of 0.22 per nucleotide.

Parameters Parameter estimation is a crucial aspect of most gene finding and alignment methods. Since our gene finding methods are based on probabilistic models of sequence evolution, we can do maximum likelihood estimation of the parameters if the true relationship between data are known, i.e. if the alignments and gene structures of the sequence pairs are known. This is the case for the simulated sequence pairs in Data Set 1. For each time step we simulate an additional 100 sequence pairs for the purpose of parameter estimation. The situation is more subtle when working with the human and mouse sequences in Data Set 2 since the alignments and gene structures are unknown. Usually the evolutionary distance between two genes is estimated and a corresponding set of predefined parameters are used, e.g. PAM similarity matrices. For our experiments, we use the parameters estimated in [2].

Signals Much work is concerned about identifying functional sites of a gene, and several customized programs exist for identifying particular sites, see e.g. [15]. In our gene finding methods, we need to identify the start/stop codons and the donor/acceptor sites of a gene. We could perform this iden-

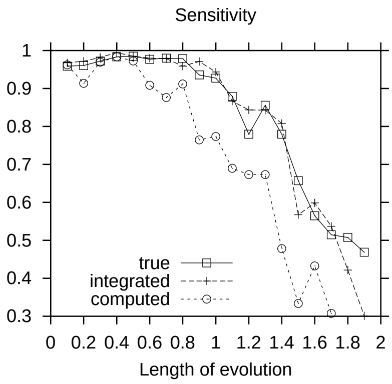

Figure 4: **DNA level score:** Nucleotide sensitivity and specificity of the predictions made on simulated sequences. The alignment score function only considers the nucleotide level. Each data point is the average over 20 predictions.

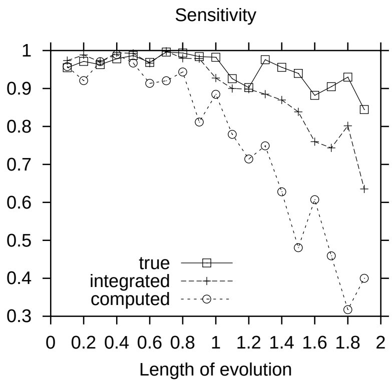

Figure 5: **DNA and protein level score:** Nucleotide sensitivity and specificity of the predictions made on simulated sequences. The alignment score function also considers the protein level. Each data point is the average over 20 predictions.

tification by using a customized program on the genes in question to predict the positions of potential signals, i.e. determine the sets D_i^x, A_i^x, $G_{x,i}^{start}$ and $G_{x,i}^{stop}$. However, this approach would make it difficult to compare the accuracy of our gene finding method with the accuracy of other methods, since a good prediction might be the result of a good signal detection by the customized program. To avoid this problem, we use a very general and simple definition of functional sites, where each site is just a pattern of two or three nucleotides as summarized in Table 1.

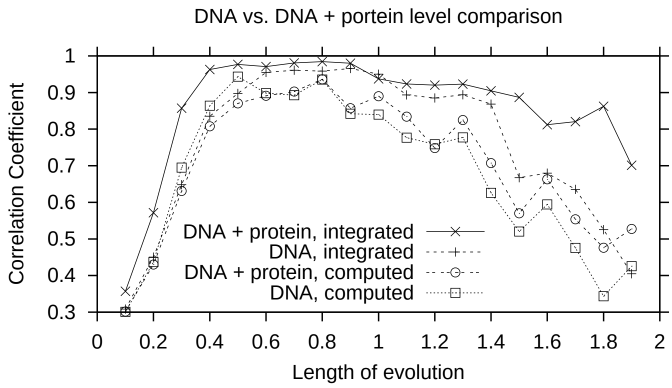

Figure 6: Correlation coefficient of predictions made on the time series of simulated sequences. Each data point is the average over 20 predictions.

Performance evaluation To compare prediction performance, we measure the nucleotide level sensitivity $\mathrm{Sn} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FN})$, i.e. the fraction of true coding nucleotide predicted to be coding, and the nucleotide level specificity $\mathrm{Sp} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP})$, i.e. the fraction of predicted nucleotides actually true, for each method. TP (true positives) is the number of coding nucleotides predicted to be coding, FP (false positives) is the number of non-coding nucleotides predicted to be coding, and FN (false negatives) is the number of coding nucleotides predicted to be non-coding. Neither Sn or Sp alone constitute a good measure of global accuracy. We use the correlation coefficient, $CC = (\mathrm{TP} \cdot \mathrm{TN} - \mathrm{FP} \cdot \mathrm{FN})/\sqrt{(\mathrm{TP} + \mathrm{FP}) \cdot (\mathrm{TN} + \mathrm{FN}) \cdot (\mathrm{TP} + \mathrm{FN}) \cdot (\mathrm{TN} + \mathrm{FP})}$, as a global measure of similarity. It is a well-known measure in the gene finding literature, see e.g. [6]. All three measures take their values between 0 and 1, where values close to 1 indicate good predictions.

Experiments on Data Set 1 This data set consists of a set of simulated sequence pairs with known alignments gene structures. For each sequence pair we use the four implementations of our gene finding approach to predict the gene structures of the sequences. The two implementations which use a given alignment (*Eukaryotic Method 1*) are tested with the *true* alignment known from the simulations, and a *computed* alignment obtained by a standard alignment method [9]. The true alignment is presumable the best possible alignment, and the computed alignment is presumable the best possible inferred alignment because optimal alignment parameters is estimated and used. The other two implementations *integrate* the alignment and prediction steps (*Eukaryotic Method 2*).

Figure 4 and 5 show the plots of the accuracy statistics for the predictions.

Gene finding method	Specificity	Sensitivity	Correlation
DNA level score Computed alignment	0.89	0.92	0.88
DNA level score Glass alignment	0.92	0.93	0.90
DNA level score Integrated alignment	0.88	0.97	0.91
DNA and protein level score Computed alignment	0.89	0.93	0.88
DNA and protein level score Glass alignment	0.92	0.93	0.90
DNA and protein level score Integrated alignment	0.92	0.98	0.94
GLASS/ROSETTA	0.97	0.95	–
Genscan	0.89	0.98	–

Table 2: Accuracy statistics for gene predictions in the human/mouse data set. The statistics for GLASS/ROSETTA and Genscan are from [2].

For all four implementations, we observe over-prediction for small evolutionary distances. This is a consequence of our founding assumption of non-coding regions being fare divergent, which does not hold when the overall evolutionary distance is small. In general, very closely related sequences will always cause problems in a pure comparative gene finder because different evolutionary patterns cannot be observed. The experiments also indicate that better predictions are made by combining the alignment and prediction steps instead of using a pre-inferred alignment (unless when the true alignment is used). This is confirmed by Figure 6 which shows the global accuracy of the four implementations by plotting the correlation coefficient of the predicted structures. The experiments also indicate that including the protein level in the alignment score function results in improved predictions.

Experiments on Data Set 2 This data set consists of a set of 117 sequence pairs from human and mouse with known gene structures but unknown alignments. Again we use the four implementations of our gene finding approach in a total of six different settings to predict the gene structures of the 117 sequence pairs. The two implementations which use a given alignment are tested with a *computed* alignment obtained by a standard alignment method [9], and an alignment found using the GLASS alignment program from [2]. Table 2 shows the prediction accuracy of our gene finders measured in sensitivity, specificity and correlation coefficient. The table also shows the prediction accuracy of two existing gene prediction programs, GLASS/ROSETTA [2] and Genscan [5]. GLASS/ROSETTA is a comparative approach which first aligns two genomic sequences using GLASS and then predicts genes on the basis of the aligned sequences using ROSETTA. The idea behind ROSETTA is similar to our similarity measure that is based on an given alignment, but with a more elaborated signal prediction scheme. Genscan is a HMM based gene prediction program. It analyzes each sequence separately and does not utilize any comparative methods.

The prediction results on the 117 pairs of human and mouse sequences lead to the same conclusions as the prediction results on the simulated sequences. Predictions made by combining the alignment and prediction steps instead of using a pre-inferred alignment yield better results. Including the protein level in the alignment score function further improves the prediction accuracy. The experiments show that our gene finding methods can compete with existing gene finding programs, and in some cases outperform them.

References

[1] V. Bafna and D. H. Huson. The conserved exon method for gene finding. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 3–12, 2000.

[2] S. Batzolou, L. Pachter, J. P. Mesirov, B. Berger, and E. S. Lander. Human and mouse gene structure: Comparative analysis and application to exon prediction. *Genome Research*, 10:950–958, 2000.

[3] P. Blayo, P. Rouzé, and M.-F. Sagot. Orphan gene finding - an exon assembly approach. Unpublished manuscript, 1999.

[4] S. Brunak, J. Engelbrecht, and S. Knudsen. Prediction of human mRNA donor and acceptor sites from the DNA sequence. *Journal of Molecular Biology*, 220:49–65, 1991.

[5] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, (268):78–94, 1997.

[6] M. Burset and R. Guigó. Evaluation of gene structure prediction programs. *Genomics*, 34:353–367, 1996.

[7] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, chapter 1-6. Cambridge University Press, 1998.

[8] M. S. Gelfand, A. A. Mironov, and P. A. Pevzner. Gene recognition via spliced sequence alignment. *Proceedings of the National Academy of Science of the USA*, 93:9061–9066, 1996.

[9] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708, 1982.

[10] J. Hein. An algorithm combining DNA and protein alignment. *Journal of Theoretical Biology*, 167:169–174, 1994.

[11] J. Hein and J. Støvlbæk. Combined DNA and protein alignment. In *Methods in Enzymology*, volume 266, pages 402–418. Academic Press, 1996.

[12] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communication of the ACM*, 18(6):341–343, 1975.

[13] I. Korf, P. Flicek, D. Duan, and M. R. Brent. Integrating genomic homology into gene structure prediction. *Bioinformatics*, 17:140–148, 2001.

[14] A. Krogh. A hidden Markov model that finds genes in e. coli DNA. *Nucleic Acids Research*, 22:4768–4778, 1994.

[15] L. Milanesi and I. Rogozin. Prediction of human gene structure. In *Guide to Human Genome Computing*, chapter 10. Academic Press Limited, 2nd edition, 1998.

[16] L. Pachter, M. Alexandersson, and S. Cawley. Applications of generalized pair hidden Markov models to alignment and gene finding problems. In *Proceedings of the 5th Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 241–248, 2001.

[17] J. S. Pedersen and J. Hein. Gene finding with hidden Markov model of genome structure and evolution. Unpublished manuscript, submitted to *Bioinformatics*.

[18] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[19] Z. Yang. *Phylogenetic Analysis by Maximum Likelihood (PAML)*. University College London, 3.0 edition, may 2000.