Probabilistic Methods for Computational Annotation of Genomic Sequences

Dissertation

zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
„Doctor rerum naturalium“
an der Georg-August-Universität Göttingen

vorgelegt von

Oliver Keller

aus Bremen

Göttingen 2010

Betreuungsausschuss:	Prof. Dr. Stephan Waack
	Prof. Dr. Carsten Damm
	Prof. Dr. Dieter Hogrefe
Referent:	Prof. Dr. Stephan Waack
Koreferent:	Prof. Dr. Burkhard Morgenstern

Tag der mündlichen Prüfung: 26. Januar 2011

Abstract

New sequencing techniques have increased enormously the speed that new genomic sequences are produced. As manual inspection is impossible for this amount of data, there is an ongoing need for computational tools that can annotate this data efficiently and accurately. Essential parts of the annotation process of genomes are the prediction of protein-coding genes, and the classification of the obtained protein sequences according to their function. Currently, computational predictions are not accurate enough to be considered overall reliable.

At the same time that new data is produced that needs to be analysed, the amount of available data that can be used to guide the prediction is growing as well. In particular, databases containing annotated proteins and functional descriptions of protein families, are widespread and easily accessible, and can provide additional input to gene prediction programs.

In the focus of this thesis is the introduction of a new method that uses protein profiles that can be generated from a set of related proteins to improve the accuracy of present gene prediction methods. It was implemented as an extension to the gene prediction program AUGUSTUS, called the "Protein Profile Extension" (PPX).

Since a correct classification of protein sequences relies on accurate gene predictions especially of regions typical for a class or family, this method can be viewed a combination of gene prediction and protein classification that is designed to improve classification rates.

Both gene prediction and protein classification commonly evaluate sequences based on probabilistic models, identifying sequences that have a high probability under the model. All these models have in common the Markov property, stating

that the direct neighbourhood determines the sequence composition at specific location, without long-distance dependencies. The thesis describes the specific models used in the presented methods.

In the context of this work, other problems arose involving probabilistic modelling and protein-based gene prediction, that became projects on their own. In this these, two of these are presented: SIGI-HMM, a method for classifying microbial genes examining the phenomenon of horizontally gene transfer, and Scipio, a tool for reproducing exact gene structures from given protein sequences. The main publications about the project are attached in an appendix to the thesis.

Acknowledgements

First and foremost, I want to thank my supervisor Professor Stephan Waack for leading me to the field of bioinformatics, accommodating me in his group, and supporting me in every aspect of my work and beyond.

I am much obliged to Professor Mario Stanke who for me was the authoritative instance in the field of sequence analysis; being the author of AUGUSTUS, his guidance was vital for the PPX project.

Special thanks I want to give to Dr. Martin Kollmar, for a fruitful cooperation, and for always giving me motivating feedback.

I want to thank my parents, Brigitte and Joachim Keller, for all the love and support over the years.

I want to dedicate this thesis to my wife Dana, who always helped me stay on the right track and always believed in me. I wouldn't have made it without her.

Contents

Abstract **3**

1 Introduction **8**

2 Local probabilistic models **13**

2.1 Markov Chains 13

2.2 Hidden Markov Models 20

2.3 Conditional semi-Markov chains 23

2.4 Algorithms for semi-Markov Chains 24

2.5 The Markov model *SIGI-HMM* 28

3 Gene Prediction and Protein Classification **31**

3.1 Ab-initio Gene Prediction 31

3.2 Scipio: Homology-based gene prediction 34

3.3 Protein classification 40

4 AUGUSTUS-PPX: A hybrid gene prediction method **45**

4.1 Motivation 45

4.2 Profile-DNA mappings 46

4.3 Integration into AUGUSTUS' state model 49

4.4 Speed-up and Memory-saving Strategies 52

4.5 An algorithm for fast block search 54

4.6 Discussion 54

Appendix: **56**

A Article about SIGI-HMM **57**

"Score-based prediction of genomic islands" 57

B Article about Scipio **70**

"Using protein sequences to determine exon structures" 70

C Article about AUGUSTUS-PPX **83**

"A novel hybrid gene prediction method employing MSA's" 83

Bibliography **94**

Chapter 1

Introduction

In every cell of a living organism, its main design information is stored digitally in the form of Deoxyribonucleic acid (DNA). The data structure described by a DNA molecule is a *string*, a sequence over the four-letter alphabet $\mathcal{N} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, each letter denoting a single unit called *nucleotide* or *basepair* referring to their biochemical meaning.

The full set of DNA sequences present in an organism is called its *genome*. In the human genome, 24 sequences[1] can be distinguished that have an average length of approximately 140 Mbps (million base pairs).

The discovery of the sequential structure of the DNA in the 1950's has had a revolutionary impact on genetical research, and efforts were started to read the genomic sequences (*DNA sequencing*), making them available in databases. In 2001, the Human Genome Project released the first draft of the human genome [Con01]. Since then, the genomes of hundreds, soon thousands, of species have been sequenced, as new sequencing techniques have led to an exponential growth of the available amount of genomic data [LCM+10].

The DNA sequence provides a blueprint for the actual building blocks of the organism, the *proteins*. Proteins are vital for virtually every process in a cell, and

[1]The human genome consists of 23 pairs of *chromosomes* (numbered 1 to 22, and X) each contributing two almost identical copies of a sequence; in the male genome, one single chromosome (Y) is present replacing one of the two copies of the X-chromosome. Not counted here is the very short (16 Kbps) DNA sequence present in the mitochondrion of the cell.

also constitute their structure. From a data-processing point of view, proteins are the same sequential objects as DNA, with the nucleotide alphabet replaced by the *amino acid* alphabet $\mathcal{A}$, consisting of 20 letters. In general, the physical structure of a protein, and hence its function, is determined by the amino acid composition of the sequence.

The biological process of protein synthesis is represented by *translating* a nucleotide sequence into a protein sequence. The mapping used for translation is clearly defined, and is the same for all organisms, with very few exceptions. This quasi-universal *genetic code* assigns to a nucleotide triplet (also called *codon*) $c \in \mathcal{N}^3$ one letter $a = \mathrm{tr}(c) \in \mathcal{A}$ from the amino acid alphabet. From the 64 possible codons, 61 are translated into amino acids, while the three *stop codons* (`TAG`, `TGA` and `TAA`) are not translated, causing the protein production process to terminate. As there are only 21 possible values, the mapping tr cannot be an injective function: the code is *degenerate*, with up to six different codons mapped to the same amino acid. These *synonymous* codons mostly differ from each other only on the third nucleotide, making the protein synthesis more robust to single mutations.

The full process of automated genome annotation is as follows. The DNA is read in short fragments that are assembled in several steps to longer sequences, and ultimately to full-length chromosomes. The second step, the *gene prediction*, is the determination of the coding segments of the genomic sequences, and their translation to protein sequences. Finally, the outcoming proteins are classified into groups according to their biological function. The methods presented in this thesis focus on the two latter tasks: gene prediction and protein classification.

While the translation of a coding nucleotide sequence to a protein sequence is computationally trivial, the determination of the coding parts of a given DNA sequence is a complex task. Here, a region containing the segments coding for one protein is called a *gene*.

Nucleotide sequences are found in living cells in two forms: DNA and RNA. A DNA molecule consists of two *strands* where each nucleotide in one strand is complemented by its counterpart in the other strand, with $\{\mathtt{A}, \mathtt{T}\}$ and $\{\mathtt{G}, \mathtt{C}\}$ form-

ing pairs of mutually complementary bases. Both strands are sequential objects containing the same information, with a defined order that is reversed in opposite strands. As a result, a DNA sequence cannot be distinguished from its *reverse complement*. RNA sequences are single-stranded nucleotide sequences.[2]

The protein biosynthesis is made complex by two processes that precede the translation. First, a region from one of the DNA strands is *transcribed*, i.e. copied into single-stranded RNA. Second, in a process called *splicing*, large segments called *introns* are cut out from the RNA, and the remaining ones, called *exons*, are concatenated. The actual coding sequence is now a contiguous subsequence of the RNA, a series of codons that will be translated, followed by a stop codon. A nucleotide triplet is considered a codon only if it is *in-frame*, i.e. its distance from the translation start is a multiple of three. Introns may also occur inside a codon. The *reading frame* of an intron is the position (0, 1, or 2) relative to the codon start that the intron is inserted.

The procedure varies between the two domains of living organisms, *prokaryotes* (single-cell organisms without cell membranes such as bacteria) and *eukaryotes* (organisms with more complex cell structures, including all multicellular organisms): in prokaryotes, splicing does not occur at all (with very few exceptions), but one RNA can contain several genes (coding for multiple proteins, often with related function).

A *gene structure* is a labelling of DNA segments according to their destination in the protein coding process. More labels can be defined, for example for exons and exon parts that are transcribed to RNA but not coding for proteins, or for other functional elements contained in DNA sequences. However, this thesis deals with the problem of finding location of the *coding* exons; the segments forming the *untranslated* regions (UTRs) are identified with intergenic regions.

Because of the huge amount of raw sequence data, computational methods have to be designed that find the gene structure of a genomic sequence automatically.

[2]In RNA sequences, the letter `T` is sometimes replaced by `U` to reflect a different chemical consistence but it would be misleading from the data processing point of view to distinguish them.

The task of predicting the genes remains a challenge, as exact gene structures can in the fewest cases be determined with certainty. The accuracy of prediction programs varies according to the amount and type of data available as input. If the target DNA sequence is the only input (*ab-initio gene prediction*), the problem is especially difficult, since the prediction relies merely on a probabilistic model. Such an approach exploits different nucleotide composition in coding and non-coding sequences, and the presence of signals that precede the start of a gene, or are located at exon-intron boundaries.

If other sequences are compared to the target sequence, they can provide additional information about the gene structure. For example, in cases that the protein sequence of the same or very closely related gene is known, the gene structure may in fact be reconstructed almost with certainty, by determining DNA segments that match the query protein. The program Scipio that will be described here was designed for this task. Methods that incorporate different kinds of extrinsic information become increasingly important as the amount of available information grows.

Finding the coordinates of protein-coding exons is often necessary to determine the protein sequence that can be used in further analysis of the gene function. This is achieved by comparing the protein sequence to repositories of related sequences with known function, following the assumption that sequence similarity implies similar function. An accurate gene prediction, especially for the regions conserved among members of the same protein family, is crucial for the classification.

This issue is addressed by a new method that will be introduced as a major part of this thesis. This method combines ab-initio prediction with the mapping of protein family profiles to the target sequence that are given as additional input to the program. This mapping was implemented as an integral part of the existing gene prediction program AUGUSTUS, by extending the internal probabilistic models.

The thesis is organized in three parts: First, the probabilistic models are introduced that form the basis of the prediction methods used. Second, the general methods are described that are used in gene prediction and protein classification.

Finally, the Protein Profile Extension is presented and described in detail. Each of the chapters is accompanied by a published work that is attached in the appendix to the thesis.

Chapter 2

Local probabilistic models used in genome analysis

If no sufficient extrinsic information is available by mapping of other sequences, a sequence has to be evaluated by *intrinsic* methods: applying models for nucleotide or amino acid composition. In a probabilistic approach, the target sequence itself is considered the result of a random process, and the prediction is a decision made by a statistical test, and experimentally confirmed training data sets are used to estimate parameters for the model.

In this chapter, I will introduce local probabilistic models that are widely used in biological sequence analysis, and then describe a specific model that I developped for the software COLOMBO classifying prokaryotic genes.

The models presented here are *local* in the sense that distributions have no dependencies over long distances; this is expressed by the Markov property for sequential data: given that values on neighbouring locations are known, the nucleotide distribution of a specific location is independent from the rest.

2.1 Markov Chains

Definition: A *Markov chain* (of order 1) is given by the following data:

- a finite *state* set Q

- a set of *transition* functions ($i \in \mathbb{N}$)

$$\begin{aligned} \mathrm{Tr}_i \colon Q \times Q &\to [0,1] \\ (q', q) &\mapsto \mathrm{Tr}(q \mid q') \end{aligned}$$

  such that $\sum_{q \in Q} \mathrm{Tr}_i(q \mid q') = 1$ for all $q' \in Q$ and $i \geq 1$

- an *initial* transition $\mathrm{Tr}_{\mathrm{init}} \colon Q \to [0,1]$, with $\sum_{q \in Q} \mathrm{Tr}_{\mathrm{init}}(q) = 1$.

If transition functions are specified only for $i = 1, \ldots, n-1$ together with the initial transitions, they constitute a *finite* Markov chain of length n. For each $n \in \mathbb{N}$, a Markov chain of length n (or more) defines a probability distribution on state sequences $\eta = (q_1, \ldots, q_n) \in Q^n$ by

$$P(\eta) = \mathrm{Tr}_{\mathrm{init}}(q_1) \cdot \mathrm{Tr}_1(q_2 \mid q_1) \cdot \ldots \cdot \mathrm{Tr}_{n-1}(q_n \mid q_{n-1})$$

More generally, if $\mathrm{Tr}_i \colon Q^k \times Q \to \mathbb{R}$ is a function of $k+1$ variables for each $i \geq k$, and $\mathrm{Tr}_{\mathrm{init}}$ depends on k variables, they define a k-th order Markov chain with the probability distribution

$$\begin{aligned} P(\eta) = \mathrm{Tr}_{\mathrm{init}}(q_1, \ldots, q_k) \cdot \mathrm{Tr}_1(q_{k+1} \mid q_1, \ldots, q_k) \cdot \ldots \\ \ldots \cdot \mathrm{Tr}_{n-k}(q_n \mid q_{n-k}, \ldots, q_{n-1}) \end{aligned}$$

In a Markov chain, the frequency that state q appears at position i, given the value at the preceding position (or the k preceding positions), is independent from the values at any other position. A Markov chain is called *homogenous*, if the transition functions Tr_i are the same for every position i, and *m-periodical* if $\mathrm{Tr}_i = \mathrm{Tr}_j$ for $i \equiv j \pmod{m}$.

Let $Q^* = \bigcup_{n \in \mathbb{N}} Q^n$ denote the set of state sequences of arbitrary length. If a Markov chain is given with state space $Q \cup \{q_{\mathrm{term}}\}$ such that q_{term} is a terminal state (which once entered cannot be left):

$$\mathrm{Tr}_i(q_{\mathrm{term}} \mid q_{\mathrm{term}}) = 1$$

then a probability distribution summing to 1 over all $\eta \in Q^*$, is defined by[1]

$$P(\eta) = \mathrm{Tr}_{\mathrm{init}}(q_1) \cdot \mathrm{Tr}_1(q_2|q_1) \cdot \ldots \cdot \mathrm{Tr}_{n-1}(q_n|q_{n-1})\, \mathrm{Tr}_n(q_{\mathrm{term}}|q_n). \tag{2.1}$$

Markov chains can be used directly to model nucleotide sequences of a fixed type, by using the nucleotide alphabet as the state set: $Q = \mathcal{N}$; or to model the succession of sequence types, by using the states for labelling, e.g. $Q = \{\text{exon, intron, intergenic}\}$.

Feature Representation of Markov chains

A Markov chain is characterized by the fact that the frequency that a random value is observed at some position in a sequence depends only on the values at neighbouring positions. More generally, any undirected graph G with n nodes can be used to describe dependencies: each node is assigned a random value from a state set Q, such that the values on any node v depend only on the values at nodes adjacent to v. Such a distribution on Q^n is called a *Markov random field*. A Markov field over a linear graph (with nodes $1, \ldots, n$ and edges $\{i, i+1\}$) is the same as a (finite) Markov chain.

Instead of transition probabilites, the distribution function of a random field in expressed as a product of *feature* functions, where each feature takes a non-negative value that depends only on the values on one *clique* (complete subgraph) of G. Any such set of feature functions defines (after normalizing by a constant) a probability distribution on Q^n that turns it into a random field based on G. In the case of a finite Markov chain, where cliques correspond to pairs of neighbouring positions, this can be seen as follows.

Consider $f\colon Q^n \to \mathbb{R}_{\geq 0}$ of the form

$$f(q_1, \ldots, q_n) = \prod_{i=1}^{n-1} f_i(q_i, q_{i+1}) \tag{2.2}$$

[1]Provided that infinite sequences have zero probability; this can be ensured by the additional condition that all terminal transition probabilities have a lower bound $\varepsilon > 0$: $\mathrm{Tr}_i(q_{\mathrm{term}}|q) > \varepsilon$.

choosing the normalization factor $c = \sum_{Q^n} f(q_1, \ldots, q_n)$ makes $P = f/c$ a probability distribution. Using the abbreviated notation

$$f_{[j..k]}(q_j, \ldots, q_k) := f_j(q_j, q_{j+1}) \cdot \ldots \cdot f_{k-1}(q_{k-1}, q_k)$$

and

$$f(\ldots, *, \ldots) := \sum_{q \in Q} f(\ldots, q, \ldots),$$

we set

$$\begin{aligned} \mathrm{Tr}_{\mathrm{init}}(q) &= \frac{1}{c}\, f(q, *, \ldots, *) \\ \mathrm{Tr}_i(q \mid q') &= \frac{f_{[i..n]}(q', q, *, \ldots, *)}{f_{[i..n]}(q', *, \ldots, *)}. \end{aligned} \tag{2.3}$$

Then $\sum_{q \in Q} \mathrm{Tr}_i(q \mid q') = 1$, and for all $(q_1, \ldots, q_{i+1}) \in Q^{i+1}$:

$$\mathrm{Tr}_i(q_{i+1} \mid q_i) = \frac{f(q_1, \ldots, q_i, q_{i+1}, *, \ldots, *)}{f(q_1, \ldots, q_i, *, *, \ldots, *)},$$

hence

$$P(q_1, \ldots, q_n) = \mathrm{Tr}_{\mathrm{init}}(q_1) \cdot \prod_{i=1}^{n-1} \mathrm{Tr}_i(q_{i+1} \mid q_i) \tag{2.4}$$

The expression (2.3) is not defined if $f_i(q', *) = 0$; in this case, $\mathrm{Tr}_i(\cdot|q')$ can be defined safely by $1/|Q|$ (or any arbitrary function summing to 1) without violating (2.4).

The following lemma states which sets of feature functions are equivalent, and gives a characterization of the partial product $f_{[1..i]}(q_1, \ldots, q_i)$. $q \in Q$ is called *observable* at position i if there are $q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n$ such that $P(q_1, \ldots, q_{i-1}, q, q_{i+1}, \ldots, q_n) > 0$.

Lemma: Two sets of feature functions $(f_1, \ldots, f_{n-1})$ and $(\tilde{f}_1, \ldots, \tilde{f}_{n-1})$ describe the same distribution if and only if there are positive functions

$$r_2, \ldots, r_n \colon Q \to \mathbb{R}_{>0}$$

with r_n constant, such that

$$\tilde{f}_{[1..i]}(q_1, \ldots, q_i) = f_{[1..i]}(q_1, \ldots, q_i) \cdot r_i(q_i), \tag{2.5}$$

if q_i is observable at i. Moreover, if q' and $q \in Q$ are observable at positions i and $i+1$:[2]

$$\begin{aligned} \tilde{f}_1(q', q) &= r_2(q) \cdot f_1(q', q) \\ \tilde{f}_i(q', q) &= \frac{r_{i+1}(q)}{r_i(q')} \cdot f_i(q', q) \end{aligned} \tag{2.6}$$

Proof: Clearly, if $\tilde{f}$ is given by equation (2.6), then (2.5) is true also; if one of $q_1, \ldots, q_{i-1}$ is not observable then both sides equal 0.

Conversely, let f and $\tilde{f}$ be equivalent feature sets such that

$$\tilde{f}(q_1, \ldots, q_n) = \frac{\tilde{c}}{c} f(q_1, \ldots, q_n)$$

for all $(q_1, \ldots, q_n) \in Q^n$. For any fixed q_i^* observable at i,

$$\frac{\tilde{f}_{[1..i]}(q_1, \ldots, q_{i-1}, q_i^*)}{f_{[1..i]}(q_1, \ldots, q_{i-1}, q_i^*)} = \frac{\tilde{c}}{c} \cdot \frac{f_{[i..n]}(q_i^*, q_{i+1} \ldots, q_n)}{\tilde{f}_{[i..n]}(q_i^*, q_{i+1} \ldots, q_n)}$$

is defined for suitable $q_1, \ldots, q_n$, and independent of the choice of the q_j (since the left side depends on $q_1, \ldots, q_{i-1}$ only, and the right side on $q_{i+1}, \ldots, q_n$). Hence

$$r_i(q) := \frac{\tilde{f}_{[1..i]}(q_1, \ldots, q_{i-1}, q)}{f_{[1..i]}(q_1, \ldots, q_{i-1}, q)}$$

is well-defined, so (2.5) is established, and the first line of (2.6) as a special case. Now let q' be observable at $i-1$, so that $(q_1, \ldots, q_{i-2})$ exist with $\tilde{f}_{[1..i-1]}(q_1, \ldots, q_{i-2}, q') > 0$. Then for $i > 1$

$$\begin{aligned} \tilde{f}_{i-1}(q', q) = \frac{\tilde{f}_{[1..i]}(q_0, \ldots, q_{i-2}, q', q)}{\tilde{f}_{[1..i-1]}(q_0, \ldots, q_{i-2}, q')} &= \frac{r_i(q) \cdot f_{[1..i]}(q_1, \ldots, q_{i-2}, q', q)}{r_{i-1}(q') \cdot f_{[1..i-1]}(q_1, \ldots, q_{i-2}, q')} \\ &= \frac{r_i(q)}{r_{i-1}(q')} \cdot f_{i-1}(q', q) \end{aligned}$$

Semi-Markov Chains

In gene prediction, the task is to find a segmentation of the sequence, together with a labelling of the segments, depending on the state. The labels correspond to the

[2] If q' or q has zero probability to occur at all, then $f_i(q', q) > 0$ could be replaced by *any* function.

sequence type. A sequence of states, equipped with coordinates $0 = t_0 < \ldots < t_n$ is called a *parse*:

$$\phi = (z_1, \ldots, z_n); \qquad z_i = (q_i, t_i) \in Q \times \mathbb{N}$$

Semi-Markov chains define probability distributions on parses, as follows:

Definition: Let $n \in \mathbb{N}$, and $N = \{1, \ldots, n\}$. A *finite semi-Markov chain* is a homogeneous Markov chain with state set $(Q \times N) \cup \{q_{\text{term}}\}$, and probability distribution as in equation (2.1) such that

$$\begin{aligned} \operatorname{Tr}\big((q,t) \mid (q',t')\big) &= 0 \qquad (t \leq t') \\ \operatorname{Tr}\big(q_{\text{term}} \mid (q',t')\big) &= \begin{cases} 0 & (t' < n) \\ 1 & (t' = n) \end{cases} \\ \operatorname{Tr}(q_{\text{term}} \mid q_{\text{term}}) &= 1 \end{aligned}$$

The second equation ensures that only complete parses (reaching the end of the interval N) will get a positive probability. In an *infinite* semi-Markov chain, N is given by $\mathbb{N}$, and the terminal state is dropped.[3]

In many applications of semi-Markov chains the t_i represent points in time, to model the duration of a transition or a phase; however, in this thesis they refer to sequence coordinates.

Although themselves defined as Markov chains, semi-markov chains are usually viewed as a generalization of (inhomogeneous) Markov chains, over the state set Q: Let the *maximal length* of a state q be given by

$$l_{\max}(q) = \max\{t - t' \mid \operatorname{Tr}(q, t \mid q', t') > 0\}. \tag{2.7}$$

Then a Markov chain of length n is the same as a semi-Markov chain where each state q has maximal length $l_{\max}(q) = 1$, by setting

$$\operatorname{Tr}_i^{\text{MC}}(q \mid q') = \operatorname{Tr}^{\text{sMC}}\big((q, i+1) \mid (q', i)\big)$$

[3]However, the distribution of infinite semi-Markov chains is on infinite parses, and the event represented by a finite parse is the union of all infinite parses having it as a prefix. In particular, events represented by parses of varying length are not necessarily disjoint and do not add up in general.

The semi-Markov chain is said to be *time-homogeneous* if

$$\mathrm{Tr}\big((q,t+l)\mid(q',t'+l)\big)=\mathrm{Tr}\big((q,t)\mid(q',t')\big)$$

i.e. if the choice of the state and the length of a phase is independent of the current point of time (or here, sequence coordinate). A Markov chain is homogeneous exactly if it is time-homogeneous when viewed as a semi-Markov chain. A feature representation for finite semi-Markov chains, analogously to (2.2), can be derived as follows:

Lemma: Let $N=\{1,\ldots,n\}$ for some $n\in\mathbb{N}$, and $f\colon (Q\times N)\times(Q\times N)\to\mathbb{R}$ satisfy

$$f\big((q',t'),(q,t)\big)=0 \text{ for all } t\le t'$$

Let F be defined on parses $\phi=(z_1,\ldots,z_m)\in(Q\times N)^*$ by

$$F(\phi)=f_0(z_1)\prod_{i=1}^{m-1}f(z_i,z_{i+1}) \tag{2.8}$$

and let $\delta\big((q,t)\big)=0$ for $t<n$ and $\delta\big((q,n)\big)=1$. Then there is a semi-Markov chain with probability distribution proportional to $F(\phi)\cdot\delta(z_m)$ (i.e., proportonal to F for complete parses, and equal to 0 for incomplete parses).

Proof: Define

$$\beta(z)=\delta(z)+\sum_{m=1}^{n}\sum_{\substack{z_1,\ldots,z_m\\ \in Q\times N}} f(z,z_1)\,f(z_1,z_2)\cdot\ldots\cdot f(z_{m-1},z_m)\;\delta(z_m). \tag{2.9}$$

Then, for $z'=(q',t')$ with $t'<n$:

$$\begin{aligned}
&\sum_{z\in Q\times N} f(z',z)\beta(z)\\
&\quad=\sum_{z\in Q\times N}\Big(f(z',z)\,\delta(z)+\sum_{m=1}^{n}\sum_{z_1,\ldots,z_m} f(z',z)\,f(z,z_1)\cdot\ldots\cdot f(z_{m-1},z_m)\,\delta(z_m)\Big)\\
&\quad=\sum_{m=1}^{n+1}\sum_{z_1,\ldots,z_m} f(z',z_1)\cdot\ldots\cdot f(z_{m-1},z_m)\,\delta(z_m) \qquad =\beta(z')
\end{aligned}$$

since the case $m > n$ contributes 0 (the length of a parse cannot exceed n), and $\delta(z') = 0$. By setting $c = \sum_{z \in Q \times N} f_0(z)\,\beta(z)$ and

$$\mathrm{Tr}_{\mathrm{init}}(z) = \frac{1}{c} f_0(z)\,\beta(z), \qquad \mathrm{Tr}(z \mid z') := \begin{cases} \dfrac{f(z', z)\,\beta(z)}{\beta(z')}, & (\beta(z') > 0) \\ 1/|Q|, & (\beta(z') = 0) \end{cases}$$

$\mathrm{Tr}_{\mathrm{init}}$ and $\mathrm{Tr}(\cdot \mid z')$ are well-defined and sum to 1, except in the case $z' = (q', n)$ where $\mathrm{Tr}(\cdot \mid z') \equiv 0$. (In case $\beta(z') = 0$, $\mathrm{Tr}(\cdot \mid z')$ may be defined in an arbitrary way). Together:

$$\mathrm{Tr}_{\mathrm{init}}(z_1) \cdot \mathrm{Tr}(z_2 \mid z_1) \cdot \ldots \cdot \mathrm{Tr}(z_m \mid z_{m-1}) = \frac{1}{c} F(z_1, \ldots, z_m)\,\beta(z_m) \tag{2.10}$$

The proof is completed by setting

$$\mathrm{Tr}(q_{\mathrm{term}} \mid (q, n)) = \mathrm{Tr}(q_{\mathrm{term}} \mid q_{\mathrm{term}}) = 1$$

since $\beta(z_m) = 1 = \delta(z_m)$ if $z_m = n$, and $\mathrm{Tr}(q_{\mathrm{term}} \mid z_m) = 0 = \delta(z_m)$ if $z_m < n$.

2.2 Hidden Markov Models

Hidden Markov Models are among the most widely used probabilistic models for the local analysis of sequential data, including but not restricted to biological data [DEKM99].

Definition: A *Hidden Markov Model* (HMM) consists of:

- a Markov chain $\mathbf{M} = (Q, (\mathrm{Tr}_i)_{i \in \mathbb{N}}, \mathrm{Tr}_{\mathrm{init}})$
- an alphabet $\mathcal{E}$, the *emission alphabet*
- *emission functions*

$$\begin{aligned} \mathrm{Em}_i \colon Q \times \mathcal{E} &\to \mathbb{R} \quad (i \geq 0), \\ (q, c) &\mapsto \mathrm{Em}_i(c \mid q) \end{aligned}$$

satisfying $\sum_{c \in \mathcal{E}} \mathrm{Em}_i(c \mid q) = 1$ for all $q \in Q$

An HMM defines a distribution on $\mathcal{E}^n$ dependent on a given state sequence $\eta = (q_0, \ldots, q_{n-1})$ by

$$P(\sigma \mid \eta) = \mathrm{Em}(\sigma_0 \mid q_0) \cdot \ldots \cdot \mathrm{Em}(\sigma_{n-1} \mid q_{n-1}). \tag{2.11}$$

σ is called the *emission*. In a typical application, the emitted sequence is known but the underlying state sequence is not ("hidden"). The task is usually to determine state sequences with a high conditional probability $P(\eta \mid \sigma)$, given the observed sequence (*a-posteriori* probability); this reversion of dependencies makes it a *Bayesian* approach (cf. Bayes' formula $P(A \mid B) = \frac{P(A)}{P(B)} P(B \mid A)$). In general, a Bayesian test uses a pre-defined *a-priori* distribution on hypotheses (here: the Markov chain on state sequences) to choose the most likely ones given the observation. As the observation is constant, this is equivalent to maximizing the *joint probability*

$$P(\sigma, \eta) = P_{\mathbf{M}}(\eta) \cdot P(\sigma \mid \eta) \tag{2.12}$$

Unless otherwise specified, Hidden Markov Models are *state-homogenous*, i.e. the underlying Markov chain is homogenous; if Em_i is the same function for all i, the HMM is said to have *homogenous emissions*.

Hidden Markov Models can be generalized in several ways. For the purposes of gene prediction it is convenient to allow the following:

- each state can emit a full sequence segment $\omega \in \mathcal{E}^*$, rather than a single character
- the emission probability Em may depend on the state preceding the last state
- Em may depend on the previously emitted sequence[4]

Definition: A *Generalized Hidden Markov Model* (GHMM) is given by the same data as an HMM, with an emission function generalized to the form

$$\begin{array}{rccc} \mathrm{Em}\colon & (Q \cup \{q_{\mathrm{init}}\}) \times Q \times \mathcal{E}^* \times \mathcal{E}^* & \to & [0,1] \\ & (\quad q', \qquad q, \quad \omega', \quad \omega\ ) & \mapsto & \mathrm{Em}(\omega \mid q', q, \omega') \end{array}$$

[4] Although the GHMMs presented here do have homogenous emission probabilities, they are a true generalization of ordinary HMMs with inhomogenous emissions since in those, i is the length of the previously emitted sequence.

The additional state q_{init} is a dummy state to be able to model the first emission probability where there is no predecessor; in this case, the previously emitted sequence is the empty string ϵ.

Joint probabilities in a GHMM refer to a segmented sequence. Let $\ell = |\sigma|$ be the length of σ, and $\sigma_{[t..u)}$ denote the substring $\sigma_t \cdots \sigma_{u-1}$. Let J be a sequence of indices

$$J = (t_0, \ldots, t_n), \qquad 0 = t_0 < t_1, < \ldots < t_n = \ell$$

segmenting σ into substrings:

$$\sigma = \omega_1 \cdots \omega_n, \qquad \omega_i = \sigma_{[t_{i-1}..t_i)},$$

such that (σ, J) can be identified with $(\omega_0, \ldots, \omega_{n-1})$. The GHMM defines a distribution on segmented sequences, subject to the state sequence η:

$$\begin{aligned} P(\sigma,\, J \mid \eta) = &\operatorname{Em}(\omega_1 \mid q_{\text{init}},\, q_1,\, \epsilon\,) \cdot \operatorname{Em}(\omega_2 \mid q_1,\, q_2,\, \omega_1\,) \cdot \ldots \\ &\ldots \cdot \operatorname{Em}(\omega_n \mid q_{n-1},\, q_n,\ \ \omega_1 \cdots \omega_{n-1}\,), \end{aligned} \tag{2.13}$$

and a joint probability for a parse $\phi = (J, \eta)$ and a sequence, by

$$P(\sigma, \phi) = P(\sigma, J, \eta) = P_{\mathbf{M}}(\eta)\; P(\sigma, J \mid \eta) \tag{2.14}$$

If the lengths of emission probabilities are independent of previous emissions ω', of the form

$$L(l \mid q', q) = \sum_{\omega \in \mathcal{E}^l} \operatorname{Em}(\omega \mid q',\, q,\, \omega')$$

then the global distribution of parses produced by a GHMM is distributed according to an infinite time-homogeneous semi-Markov chain, with transition probabilities:

$$\operatorname{Tr}\big((q,t) \mid (q',t')\big) = \operatorname{Tr}^{(\text{GHMM})}(q \mid q')\; L(t - t' \mid q', q)$$

$P_{\mathbf{M}}(\eta)$ can be a distribution on variable length sequences, as in (2.1), or add up to 1 for each n. In the latter case, the joint probability (2.14) can be interpreted as the probability to observe σ as the emission of a *partial* parse.

2.3 Conditional semi-Markov chains

A generative model like a Hidden Markov Model implicitly defines a distribution on the set of sequences, by summing equation (2.12) over all state sequences, or equation (2.14) over all parses:

$$P(\sigma) = \sum_{\eta \in Q^n} P(\sigma, \eta)$$

$$P(\sigma) = \sum_{\substack{\phi=(J,\eta) \\ J=(0=t_0,\ldots,t_n=|\sigma|) \\ \eta \in Q^n}} P(\sigma, \phi)$$

However, having a distribution on the set of sequences is not necessary when we are just looking for a labelled segmentation of a fixed sequence. With dependencies reversed, the state sequence is modelled subject to the observed sequence. This approach is pursued by *conditional* Markov chains and their generalizations, *conditional semi-Markov chains* (and *conditional random fields*).

Definition: A conditional (semi-)Markov chain consists of

- a set $\mathcal{H}$ of *observations*
- a state set Q
- for each $h \in \mathcal{H}$, a (semi-)Markov chain $\mathbf{M}_h$ on Q

$$q \mapsto \mathrm{Tr}_{\mathrm{init}}(z \mid h) \qquad (h \in \mathcal{H})$$

$$(q', q) \mapsto \mathrm{Tr}_i(z \mid z', h) \qquad (i \geq 1,\ h \in \mathcal{H})$$

where $z = (q, t) \in Q \times N_h$ in the semi-Markov case, or $z \in Q$ otherwise.

The conditional Markov chain defines, for each observation h, a distribution on Q^n (or $(Q \times N_h)^*$), by

$$P(\phi \mid h) = \mathrm{Tr}_{\mathrm{init}}(z_1 \mid h) \cdot \mathrm{Tr}_1(z_2 \mid z_1, h) \cdot \ldots \cdot \mathrm{Tr}_{n-1}(z_n \mid z_{n-1}, h)$$

The following lemma shows that GHMMs can be considered a special case of conditional semi-Markov chains.

Lemma: With a fixed observation sequence σ, the a-posteriori distribution on parses defined by a GHMM is a semi-Markov chain over the extended state set $Q \times N$, where $N = \{1, \ldots, \ell\}$ is the set of positions of σ. Hence, the set of a-posteriori distributions is a conditional semi-Markov chain with observation set $\mathcal{H} = \mathcal{E}^*$, and $N_\sigma = \{1, \ldots, |\sigma|\}$.

Proof: In order to see that the a-posteriori distribution can be represented by a semi-Markov chain, it is enough to show that it has a feature representation, as explained in the end of section 2.1. Set $c = P^{(\mathrm{GHMM})}(\sigma)$, and define

$$\begin{aligned} f_0\big((q,t)\big) &= \frac{1}{c}\ \mathrm{Tr}^{(\mathrm{GHMM})}_{\mathrm{init}}(q) \cdot \mathrm{Em}(\sigma_{[0..t)} \mid \epsilon,\, q_{\mathrm{init}},\, q) \\ f\big((q',t'),(q,t)\big) &= \mathrm{Tr}^{(\mathrm{GHMM})}(q',q) \cdot \mathrm{Em}(\sigma_{[t'..t)} \mid \sigma_{[0..t')},\, q',\, q) \\ f\big((q',t'),(q,t)\big) &= 0 \qquad (t \leq t) \end{aligned}$$

Then, the a-posteriori distribution can be written as

$$\begin{aligned} P^{(\mathrm{GHMM})}(\phi \mid \sigma) &= \frac{1}{c}\ P^{(\mathrm{GHMM})}(\sigma,\ \phi) \\ &= f_0(z_1) \cdot f(z_1, z_2) \cdot \ldots \cdot (z_{m-1}, z_m) \end{aligned}$$

where $z_i = (q_i, t_i)$ and $\phi = (z_1, \ldots, z_m)$, if the parse ends with $t_m = n$, and 0 otherwise.

2.4 Algorithms for semi-Markov Chains

Viterbi Algorithm

The Viterbi algorithm maximizes the probability of parses in a semi-Markov chain. Given a feature representation

$$F(\phi) = f_0(z_1) \cdot \prod_{i=1}^{m-1} f(z_i, z_{i+1}),$$

the Viterbi variables are defined for $z = (q, i) \in Q \times N$ as

$$\gamma(z) = \max\big\{F(z_1, \ldots, z_{m-1}, z) \mid m \geq 1;\ z_1, \ldots, z_{m-1} \in Q \times N\big\}$$

and can be calculated iteratively by

$$\gamma(z) = \max\{\gamma(z') \cdot f(z', z) \mid z' \prec z\} \cup \{f_0(z)\}$$

where $(q', i') \prec (q, i)$ is defined by $i - l(q) \leq i' < i$, and the state length $l(q)$ is given by equation(2.7); more restrictive definitions that are necessary for $f(z', z) > 0$ may be used if they can be determined efficiently.

The parse with maximal probability can then be determined by *backtracking*: Define iteratively $z^{(k)} = (q^{(k)}, i^{(k)})$, $k \leq 0$, by

$$z^{(0)} \in \underset{z \in Q \times \{n\}}{\operatorname{argmax}} \gamma(z)$$

$$z^{(k-1)} \in \underset{z' \in Q \times N}{\operatorname{argmax}} \gamma(z') \cdot f(z', z^{(k)})$$

and $K = \max\left\{k \leq 0 \mid \gamma(z^{(k-1)}) \cdot f(z^{(k-1)}, z^{(k)}) \leq f_0(z^{(k)})\right\}$. Then

$$\gamma(z^{(k+1)}) = \gamma(z^{(k)}) \cdot f(z^{(k)}, z^{(k+1)})$$

and $(z^{(K)}, \ldots, z^{(0)})$ maximizes F, and hence, P.

Forward and Backward algorithms

The *forward* and *backward algorithms* work the same way as the Viterbi algorithm, with maximum replaced by sums. The forward variables are defined as

$$\alpha(z) = \sum_{m=0}^{n-1} \sum_{\substack{z_1, \ldots, z_m \\ \in Q \times N}} F(z_1, \ldots, z_m, z) \tag{2.15}$$

(with the case $m = 0$ contributing $F(z) = f_0(z)$), while the backward variables $\beta(z)$ are defined as in equation (2.9).

An iterative formula for $\alpha(z)$ is given by

$$\alpha(z) = f_0(z) + \sum_{z' \prec z} \alpha(z') \cdot f(z', z)$$

Analogously, an iterative formula for $\beta(z)$ is given by

$$\beta\big((q, n)\big) = 1$$

$$\beta(z) = \sum_{z' \succ z} f(z, z') \cdot \beta(z), \qquad z = (q, i),\ i < n.$$

Algorithm 1. Pseudocode of the Viterbi algorithm, for state set $Q = \{q_1, \ldots, q_r\}$

`get_predecessor`(i, j):

  $v^* \leftarrow f_0(q_j, i), \quad i^* \leftarrow 0, \quad j^* \leftarrow -1$
  for $i' = \max\{i - l(q), 1\}, \ldots, i - 1$ **do**
    for $j' = 1, \ldots, r$ **do**
      $v \leftarrow \gamma[i', j'] \cdot f(q_{j'}, i', q_j, i)$
      if $v^* < v$ **then**
        $v^* \leftarrow v, \quad i^* \leftarrow i', \quad j^* \leftarrow j'$
      end if
    end for
  end for
  return (v^*, i^*, j^*)

`viterbi`:

  for $i = 1, \ldots, n$ **do**
    for $j = 1, \ldots, r$ **do**
      $(v, i', j') \leftarrow$ `get_predecessor`(i, j)
      $\gamma[i, j] \leftarrow v$
    end for
  end for
  $s \leftarrow \gamma[n, 1], \quad j \leftarrow 1$
  for $j' = 2, \ldots, r$ **do**
    if $s < \gamma[n, j']$ **then**
      $s \leftarrow \gamma[n, j'], \quad j \leftarrow j'$
    end if
  end for
  $i \leftarrow n, \quad \phi \leftarrow ()$
  repeat
    $z \leftarrow (q_j, i)$
    `push_front`(ϕ, z)
    $(v, i, j) \leftarrow$ `get_predecessor`(i, j)
  until $j = -1$
  return (ϕ, s)

Forward and backward variables can be used to calculate probabilities from features. The constant c of the feature function can be calculated as:

$$c = \sum_{z \in Q \times \{n\}} \alpha(z) = \sum_{\substack{z=(q,i) \\ 1 \le i \le l(q)}} f_0(z)\, \beta(z)$$

The probability to observe $z_1, \ldots, z_m$ as a subsequence of a parse is

$$\frac{1}{c}\, \alpha(z_1) \cdot f(z_i, z_{i+1}) \cdot \ldots \cdot f(z_{m-1}, z_m) \cdot \beta(z_m);$$

to express in addition that the parse starts with z_1, or ends with z_m, $\alpha(z_1)$ is replaced in this equation with $f_0(z_1)$, or $\beta(z_m)$ with $\delta(z_m)$, respectively. Defining *reverse* transition probabilities by

$$\mathrm{Tr}^{\mathrm{rev}}_{\mathrm{init}}(z) = \frac{1}{c}\, \alpha(z)\, \delta(z), \quad \mathrm{Tr}^{\mathrm{rev}}(z \mid z') = \frac{\alpha(z)\, f(z, z')}{\alpha(z')}, \quad \mathrm{Tr}^{\mathrm{rev}}(q_{\mathrm{term}} \mid z) = \frac{f_0(z)}{\alpha(z)}$$

the probability for a parse, in analogy to equation (2.10), can also be expressed by

$$\mathrm{Tr}^{\mathrm{rev}}_{\mathrm{init}}(z_m) \cdot \mathrm{Tr}^{\mathrm{rev}}(z_{m-1} \mid z_m) \cdot \ldots \cdot \mathrm{Tr}^{\mathrm{rev}}(z_1 \mid z_2) \cdot \mathrm{Tr}^{\mathrm{rev}}(q_{\mathrm{term}} \mid z_1). \tag{2.16}$$

The backward variables can be used to rescale viterbi variables to probabilities: equation (2.10) shows that multiplying the scoring function with $\beta(z)/c$ yields the probability for a partial parse ending in z. In particular, $\gamma(z)\beta(z)/c$ is the probability of the highest-scoring parse ending in z; even though scores (products of features) are not given as probabilities, maximizing scores for a fixed last state is the same as maximizing probabilities for partial scores.

In the case that features are given by transition probabilities, then $c = 1$; all backward variables equal 1, while the forward variables $\alpha(z)$ refer to the probability to observe a given state ending at a given position.

If features are given by the joint probability of a GHMM, then c is the probability of the observed sequence; the forward variables refer to the joint probability that a parse ending with state q at position i emits $\sigma_{[0..i)}$, and the backward variables refer to the conditional probability that $\sigma_{[i..n)}$ is emitted given that the former is the case. In particular, $\beta(z)/c$ is the ratio between joint probability and a-posteriori probability of a partial parse ending at z.

2.5 The Markov model *SIGI-HMM*

I conclude this chapter with presenting a Markov model for the classification of genes in a prokaryotic genome that we implemented in the program *SIGI-HMM*; the detailed analysis of results is shown in the publication attached in appendix A.

In prokaryotes, *horizontal gene transfer* is a frequent phenomenon: in a single event, genes from evolutionary distant sources are incorporated into the genome of an organism. In contrast to vertical transfer where the genomic material is transferred from one generation to the next, only subject to minor changes caused by mutations and recombination, the incorporation of alien genes results in a sudden addition of new, and possibly very dissimilar sequence to the genome which frequently changes the behaviour of the microbes. In particular, they can make them pathogenic. The detection of horizontal gene transfer is therefore an important task.

The transferred sequences can contain several genes. A set of consecutive genes that is conspicuous compared to the overall nucleotide composition of the genome is called a *genomic island*; when it is characteristic for a pathogen, also a *pathogenicity island*.

In SIGI-HMM, an algorithm based on a Markov model was implemented that performed the task of recognising genomic islands, given the set of genes of a prokaryotic genome; and labelling each gene according to its predicted source (the class of the donor species).

In prokaryotes, genes are contiguous coding subsequences of the genome, without introns in them, but seperated by intergenic regions. The genes are assumed to be located prior to running the algorithm, with the set of genes given in the order as they appear on the genome.

Alien genes are identified by a *codon usage* deviating from the genome under consideration, an approach previously suggested and tested successfully [Kar01]. For each of the 64 codons, the codon usage $\mathrm{CU}^{(\rho)}$ specifies the percentage of times this codon c was used to code for its amino acid: $\mathrm{CU}^{(\rho)}(c) = \frac{\#c}{\#a(c)}$, where $a(c)$ denotes the amino acid assigned to c by the genetic code, and # refers to the number of occurences of codon/amino acid in the genome(s) indexed by ρ. Thus defined,

codon usage is referred to more precisely as *synonymous codon usage* (frequency of codons coding for the same amino acid). Codon usage for candidate species was taken from the *Codon Usage Database* (`http://www.kazusa.or.jp/codon/`, [NGI00]).

However, specific genes native to the host genome are highly expressed, often coding for ribosomal proteins, can have a deviating codon usage [Kar01]. Hence, highly expressed genes need to be excluded explicitly from being labelled as alien. To this end, genes are marked as highly expressed if their codon usage resembles that of ribosomal genes.

In order to combine scoring according to codon usage with local dependencies (assuming that consecutive genes are transferred together in islands), the observed set of genes was modelled according to an HMM conditional on the observed amino acid sequences which is fixed by the model. The HMM is generalised in that emission probabilities refer to the full gene rather than single nucleotides, but since emission lengths are given and fixed, it can be also considered non-generalised if the set of hypothetical coding sequences is viewed as *alphabet*. Its architecture is as follows:

- the state set $Q = \{\mathtt{nat}, \mathtt{pA}_1, \ldots, \mathtt{pA}_r\}$ containing the candidate sources for putative alien genes, and one state labelling native genes
- inhomogeneous transition probabilities $\mathrm{Tr}_i(q \mid q')$, for $q, q' = \mathtt{nat}, \mathtt{pA}_\rho$. These reflect expected length of the islands compared to native stretches, with a bonus $\tau_{\rho,\alpha}$ depending on a user-specified sensitivity parameter α. They are inhomogeneous since the bonus depends on the observed amino-acid composition of the following gene.
- inhomogeneous emission probabilites $\mathrm{Em}_i(g \mid q)$, given by the probability that the nucleotide sequence was used to encode the fixed amino acid sequence: $\mathrm{Em}_i(c_1 \ldots c_l \mid q) = \mathrm{CU}^{(q)}(c_1) \cdot \ldots \cdot \mathrm{CU}^{(q)}(c_l)$

As shown before, a HMM described this way automatically defines a conditional semi-Markov chain (here subject to the series of genes, i.e. coding nucleotide

sequences), with feature function $f_i(q', q) = \mathrm{Tr}_i(q \mid q') \cdot \mathrm{Em}_i(g_i \mid q)$, in this case of a conditional inhomogeneous Markov chain. The Viterbi algorithm is now used to determine the most likely labelling of genes (with a putative donor, or as native). Genes that have been identified beforehand as highly expressed, carry a mark that is interpreted as additional emission that is restricted to native states. This way, they can be explicitly excluded from being predicted as alien.

Viewing the model as a conditional Markov chain defined by a scoring or feature function is more adequat than the pure Bayesian approach where the Markov chain underlying an HMM should reflect a-priori probabilities of the states. This is frequently unnecessary, or even undesired (it can cause the Viterbi algorithm to suffer from the Label Bias phenomenon; see also section 4.6)

Here, transition probabilities are modified by a sensitivity control, making this in fact a mixture of a Bayesian test and a test with a bounded error of second type (i.e., the failure to recognize an alien gene).

For example, if an isolated gene was to be tested for being alien in a binary classification test, then using the condition

$$\frac{\mathrm{Em}(g \mid q_0)}{\mathrm{Em}(g \mid q_\rho)} < \tau_{\rho,\alpha}$$

for the discrimination, would result in a guaranteed sensitivity. Replacing the threshold τ by the ratio of a-priori probabilities would be the Bayesian equivalent (and, in the case of the Markov chain, depend on the labels chosen for neighbouring genes).

With a bonus of τ being multiplied to the transition probabilities, the two approaches are sensibly combined but do not anymore reflect an a-priori distribution on states or labels; however, the advantage of the conditional Markov chain perspective is that an a-priori distribution is not needed.

Chapter 3

Gene Prediction and Protein Classification

In this chapter, I will discuss general methods that are part of the central steps of the annotation pipeline: gene prediction and protein classification.

Gene prediction is the task of finding a gene structure: a labelled segmentation $\sigma = \omega_1 \dots \omega_n$ of a genomic input sequence $\sigma \in \mathcal{N}^*$, according to their meaning for the protein coding process. Other functional sequence segments than protein coding genes are integrated into DNA [Jon06], but they will not be dealt with here.

Gene prediction methods can be distinguished according to the available input. *Ab-initio* (or intrinsic) methods do not need any input but the target sequence itself. *Extrinsic* methods use additional information gained from further sources. This information is frequently called *evidence* indicating certainty or considerably higher reliability of the prediction compared to intrinsic methods.

3.1 Ab-initio Gene Prediction

An ab-initio gene prediction can only be model-based, using probabilistic models as those introduced in chapter 2. In the most widespread ab-initio gene prediction programs (see [PP10] for an overview), the models are Generalized Hidden Markov Models, or more recently, other kinds of Conditional semi-Markov chains

[BCHP07], with gene structures represented by parses, sometimes in a one-to-one correspondence between states and labels. In these approaches, having defined a model, finding the optimal gene structure amounts to finding the highest-scoring parse, which is computed by the Viterbi algorithm.

Two components have to be distinguished in the description of a GHMM, the *architecture* which is fixed throughout one approach, and the *parameters* that have to be estimated. Commonly, transition probabilities, and the choice of state-dependent emission models are specified in the architecture, while parameters of emission models are trained.

Emission models generally fall into two categories:

- models for nucleotide composition differing in general between exons, introns and inter-genic sequence, frequently given as position-unspecific k-th order Markov chains on nucleotide level

- models for sequence motifs signalling a function, for example *translation start sites* or *splice sites* (borders between exons and introns), often given as explicit distributions at a defined distance from segment coordinates

In order to train the parameters, experimentally confirmed gene structures are evaluated. In the classical approach, nucleotide composition and signal motifs are estimated by counting occurences, compositional parameters averaged over all segments of the same type, and signal motifs at well-defined locations.

Recently, a discriminative learning approach (*online large-margin training*, [BCHP07]) has been proposed that optimizes all parameters simultaneously for single training examples. This requires scoring functions to depend on parameters linearly, which can be achieved by considering the logarithm of features. It also requires giving up a GHMM interpretation of features in general.

The work presented in the next chapter of this thesis is an extension to the ab-initio program AUGUSTUS that was introduced by Mario Stanke in 2003 [Sta03, SW03, SSMW06]. AUGUSTUS was initially based on a Generalized Hidden Markov Model (GHMM), but parameters have in some cases been retrained with the online large-margin method.

The architecture of AUGUSTUS' GHMM is shown in [Sta03], Figure 3.1. Its original state set consists of 47 states in total: 16 exon states, 30 intron states and one intergenic state.

Four types of exon states, initial, internal, terminal and single, are distinguished, all with seperate states for the two reading directions (strands). Exon states that can end inside a codon (initial and internal) are also distinguished by reading frame, since it affects the nucleotide composition.

Two types of introns are distinguished: short introns with an explicit length model, and long introns that have an implicit length model with geometrical distribution. In the latter case, an intron is represented in a parse by a fixed-length prefix followed by a segmentation into single nucleotides each separately emitted from the same state. In addition to these three states, two more intron states model the splice sites (the intron regions adjoining the exons). Introns need to be distinguished by reading frame in order to determine the reading frame for the following exon. The intergenic state also emits single nucleotides and transitions back to itself with probability close to 1.

AUGUSTUS executes the Viterbi algorithmus to determine the highest-scoring parse. A parse is not completely identical with a gene structure since multiple states can share the same label, consecutive occurences of the same state are subsumed (in the single-nucleotide emission models), and model coordinates may differ from biological coordinates. For example, the sequence region modelled by the initial exon state starts before the biological exon, in order to contain signals for the translation start located in the intergenic region.

AUGUSTUS can also execute a randomized sampling algorithm [SKG$^+$06] to procude a set of parses, each with probability proportional to their score. This sampling algorithm first calculates the forward variables, and then iterative determines $z^{(0)}, z^{(-1)}, \ldots$, proportional to the reverse transition probabilities given in (2.16).

3.2 Scipio: Homology-based gene prediction

While an ab-initio approach is based on a model, evidence is in most cases gained by *aligning* informant sequences to the target genome: coordinates of one sequence are mapped to the other in a way that the corresponding sequence segments show high, or even full identity.

Extrinsic approaches can be categorized according to the kind of the informant sequence:

- *Comparative* methods use alignments with genomic sequences of closely related species, exploiting the fact that coding sequences show a higher degree of cross-species conservation
- *Transcript-based* methods use alignments with RNA sequenced independently from the DNA; since mature RNA does not contain introns, matching segments of DNA are identified as exons
- *Homology-based* methods use alignments with RNA or protein sequences coming from already known genes found at other locations or in other genomes, DNA segments showing high similarity are likely to code for related proteins

The most accurate results are gained by approaches that combine extrinsic information with an ab-initio gene prediction. AUGUSTUS has the ability to use the evidence generated by external alignment tools, in the form of *hints* [SSMW06]: it can consider additional input that suggests, with specified reliability, labellings at indicated locations, and incorporate this into the probabilistic model.

Formally, an alignment is given by a pairwise parse: segmentations

$$s = \nu_1 \cdots \nu_n, \quad \sigma = \omega_1 \cdots \omega_n$$

of query sequence s and the target sequence σ, together with labellings q_j describing the relations between ν_j and ω_j. For example, labels might take the values MATCH (if ν_j equals ω_j, or the translation of ω_j in a protein-DNA alignment), MISMATCH (if ν_j differs from ω_j but both have same length), INSERTION (if ω_j is empty) or DELETION (if ν_j is empty), or UNALIGNED if there is no relation whatsoever;

an alignment can be assigned a score based on the labelling, similar to a semi-Markov chain but for pairs of segmentations. MISMATCH and MATCH labels not always distinguished, allowing MATCHes that are similar but not completely identical.

A *spliced* alignment considers introns as a special case of deletions; with only small penalties. A perfect spliced lignment is turned into a gene structure simply by relabelling MATCH to exon labels. Since introns might occur inside codons, it is convenient to specify spliced protein-DNA alignments in a virtual RNA coordinate system (even if the sequence forming the RNA is only known *after* aligning).

The strongest evidence for a gene can clearly be gained from an alignment with the coding (RNA) sequence itself, or with the resulting protein sequence. This puts the step of identifying a protein *before* the gene prediction step. Protein sequences may be available indepently from a gene prediction, for example when closely related genomes have already been analysed.

With the software Scipio, I am presenting in this thesis a protein-based gene prediction tool that is designed for the case of high homology. Finding a gene structure based on similarity to the translated sequence is desirable in many cases, for example:

- if protein sequences come from databases that do not contain the coordinates of the coding sequences (this information may have existed but is not as easily accessible)
- if protein or RNA sequences have been determined experimentally
- if sequences are only available for closely related species, for a cross-species search

If the original location of a protein sequence is determined in the genome, the task is to *recover* the gene structure rather than *predict* it; there is normally just one gene structure resulting in a specified protein sequence. The resulting gene structures can safely be used as reference.

The need for a tool that can determine the exact exon/intron boundaries from a given protein sequence arose during the work related with this thesis at several

occasions. Reference gene structures were needed to evaluate prediction quality. In addition, it was found also a very useful approach for directly predicting gene structures in cross-species analysis, for closely related sequences.

Scipio is a wrapper program for the alignment program BLAT written by Jim Kent [Ken02]. Postprocessing was needed since the protein-DNA alignment calculated by BLAT are not complete gene structures, for the following reasons:

- since BLAT aligns the protein sequence to the translated target sequence, it misses codons that have been split by an intron
- rather than returning one accurate hit, it returns a list of hits with varying accuracy, most of which can be regarded as false positives when searching for a specific gene
- genomes in an incomplete stage of assembly are scattered over short sequences (contigs) that have not yet been assembled, and may contain only partial genes
- sequencing or assembly errors can cause a single exon to be represented by multiple segments in the alignment
- exons that are too short to be significant hits alone are not found by BLAT

Scipio addresses these issues and turns the spliced alignments found by BLAT into gene structures, with precise exon/intron locations, by adding the split codons to the predicted exons. Segments seperated in a hit by short insertions or deletions are joined together to form a single exon.

Hits found by BLAT are sorted by score, and the highest scoring is returned. More than one hit is returned only if the hits are compatible, i.e. they are partial hits referring to non-overlapping parts of the query sequence, and are located on different contigs such that the unaligned contig ends do not get too long (allowing an interpretation as intron).

With the postprocessing performed by Scipio, the output provided by BLAT was transformed into exact gene structures, turning the spliced alignment into an

actual *homology-based gene prediction*. Details about the accuracy of the gene structures produced were published in the article attached in appendix B.

Short exons that may have been missed by BLAT are searched for by a version of the exhaustive algorithm designed by Needleman and Wunsch [NW70], a global dynamic programming algorithm with many analogies to the Viterbi algorithm. It can be used efficiently only to fill short gaps in the alignment since its running time is linear in the length of both sequences. In Scipio, its use is restricted to cases where only short parts of query and target sequences are unaligned; only these unaligned segments are the input to the algorithm. In the remainder of this section, I will describe the implementation that I designed for Scipio.

The generic procedure is as follows: Given two sequences r and s, and a penalty function $\rho(\nu,\omega)\colon \mathcal{E}_1^* \times \mathcal{E}_2^* \to \mathbb{R} \cup \{\infty\}$, the dynamic programming (DP) variables $\gamma(r,t)$ are defined as:

$$\gamma(r,t) = \min\{\ \rho(\nu_1,\omega_1) + \ldots + \rho(\nu_k,\omega_k) \mid s_{[0..r)} = \nu_1 \ldots \nu_k,\ \sigma_{[0..t)} = \omega_1 \ldots \omega_k\}$$

and iteratively calculated by

$$\begin{aligned}\gamma(0,0) &= 0;\\ \gamma(r,t) &= \min\{\gamma(r',t') + \rho(s_{[r'..r)}, \sigma_{[t'..t)}) \mid r' \le r, t' \le t\}\end{aligned}$$

The optimal alignment is calculated by backtracking through the maximizing arguments: Let

$$(r',t') = \mathrm{pred}(r,t) = \operatorname*{argmin}_{r'\le r, t' \le t} \big(\gamma(r',t') + \rho(s_{[r'..r)}, \sigma_{[t'..t)})\big)$$

Then sequences $(r_0,\ldots,r_n)$, $(t_0,\ldots,t_n)$ are determined iteratively, by starting with $(r_n,t_n) = (|s|,|\sigma|)$, and prepending the predecessor $(r_{k-1},t_{k-1}) = \mathrm{pred}(r_k,t_k)$ until $(r_0,t_0) = (0,0)$ is reached. Then

$$\nu_i = s_{[i-1..i)}; \qquad \omega_i = \sigma_{[i-1..i)}.$$

Given a label set Q characterizing alignments, scores for labels depend only on the associated segments and not on neighbouring labellings, and ρ can simply be defined as

$$\rho(\nu,\omega) = \min_{q\in Q} \rho(\ \nu,\ \omega \mid q).$$

ρ is assigned the value ∞ for cases that are explicitly excluded.[1]

For the sake of efficiency, additional restrictions for the scoring function are required. The lengths of admissible segments must be bounded, depending on the label, and pairs of empty strings must be excluded.

The label set used by Scipio is

$$Q = \{\text{MAP}, \text{INS}, \text{GAP}, \text{FS}, \text{INTRON}_0, \text{INTRON}_1, \text{INTRON}_2\},$$

and the penalty function $\rho(\,\nu,\, \omega \mid q)$ is defined subject to the conditions

$$\begin{array}{lll}
q = \text{MAP}: & |\nu| = 1, & |\omega| = 3 \\
q = \text{INS}: & |\nu| = 1, & \omega = \epsilon \\
q = \text{GAP}: & \nu = \epsilon, & |\omega| = 3 \\
q = \text{FS}: & |\nu| \in \{0, 1\}, & |\omega| \in \{1, 2\} \\
q = \text{INTRON}_0: & \nu = \epsilon, & |\omega| \geq l_{\min} \\
q = \text{INTRON}_{1,2}: & |\nu| = 1, & |\omega| \geq l_{\min} + 3
\end{array}$$

where ϵ is the empty string, and $l_{\min}$ is the minimum intron length.

The labels q =INS, GAP, FS are used to allow insertions, deletions and frameshifts, at the cost of some constant penalty $\rho(\cdot|q) = c_q$. Furthermore,

$$\rho(\nu, \omega \mid \text{MAP}) = \delta^{\text{tr}}(\nu, \omega) = \begin{cases} 0, & \text{if } \nu = \text{tr}(\omega) \\ c_{\text{MISM}}, & \text{otherwise} \end{cases}$$

where $\text{tr}(\cdot)$ denotes translation. Introns are distinguished by reading frame; the intron penalty is computed evaluating the splice sites (considering that 99% of all introns start with `GT` and end with `AG`):

$$p_{\text{INTRON}}(\omega) = p_{\text{INTSTART}}(\omega_{[0,1]}) + p_{\text{INTEND}}(\omega_{[t-2,t-1]}) + c_{\text{INTRON}}$$

[1] A technical difference between the scores defined here and for the probabilistic models is that here, for practical and historical reasons, additive scores are used, and minimized rather than maximized, with 0 being replaced by ∞ with the usual rules for calculation. The Viterbi algorithm can use additive variables as well, using log-probabilities as feature functions.

If $q = \text{INTRON}_j$ $(j = 1, 2)$, then the given nucleotide sequence is interpreted as an intron inside a codon $n_1n_2n_3$. ω has the form $n_1 \ldots n_2n_3$ $(j = 1)$, or $n_1n_2 \ldots n_3$ $(j = 2)$. The three reading frames give rise to different intron penalties and a potential mismatch penalty:

$$\left.\begin{array}{r} \rho\,(\epsilon,\ \omega \mid \text{INTRON}_0) \\ \rho\,(\nu,\ n_1\,\omega\,n_2n_3 \mid \text{INTRON}_1) \\ \rho\,(\nu,\ n_1n_2\,\omega\,n_3 \mid \text{INTRON}_2) \end{array}\right\} = \delta^{\text{tr}}(\nu,\ n_1n_2n_3) + p_{\text{INTRON}}(\omega)$$

Since the length of introns is not bounded from above, the DP variables cannot be computed directly, without t' iterating back to the start of the target sequence. This is resolved by introducing DP variables keeping track of potential intron scores at positions *inside* them ($n \in \mathcal{N}$):

$$\begin{aligned} \gamma^{(0)}(r,t) &= \min\{\ \gamma(r,t') + p_{\text{INTSTART}}(\sigma_{[t',t'+1]}) \mid t' \le t - l_{\min}\ \} \\ \gamma^{(1)}(r,t,n) &= \min\{\ \gamma(r,t') + p_{\text{INTSTART}}(\sigma_{[t'+1,t'+2]}) \mid t' \le t - l_{\min} - 3,\ \sigma_{[t']} = n\ \} \\ \gamma^{(2)}(r,t,n) &= \min\{\ \gamma(r{-}1,t') + p_{\text{INTSTART}}(\sigma_{[t'+2,t'+3]}) + \delta^{\text{tr}}(s_{[r-1]}, \sigma_{[t',t'+1]}n) \\ &\qquad\qquad \mid t' \le t - l_{\min} - 3\ \} \end{aligned}$$

leading to the following recursions

$$\gamma^{(0)}(r,t) = \min\{\ \gamma^{(0)}(r,t{-}1),\ \gamma(r,t') + p_{\text{INTSTART}}(\sigma_{[t',t'+1]})\ \} \qquad (t' = t - l_{\min})$$

$$\gamma^{(1)}(r,t,n) = \begin{cases} \gamma^{(1)}(r,t-1,n); & \sigma_{[t']} \neq n \\ \min\{\ \gamma^{(1)}(r,t-1,n), & (t' = t{-}l_{\min}{-}3) \\ \qquad \gamma(r,t') + p_{\text{INTSTART}}(\sigma_{[t'+1,t'+2]})\}; & \sigma_{[t']} = n \end{cases}$$

$$\begin{aligned} \gamma^{(2)}(r,t,n) = \min\{\ &\gamma^{(2)}(r,t-1,n), \\ &\gamma(r-1,t') + p_{\text{INTSTART}}(\sigma_{[t'+2,t'+3]}) + \delta^{\text{tr}}(\sigma_{[t',t'+1]}n\ \} \quad (t' = t{-}l_{\min}{-}3) \end{aligned}$$

and finally,

$$\gamma(r,t) = \min \begin{cases} \gamma(r-1,t-3) + \delta^{\mathrm{tr}}(s_{[r-1]}, \sigma_{[t-3,t-2,t-1]}), \\ \gamma(r-1,t) + c_{\mathrm{INS}}, \\ \gamma(r,t-3) + c_{\mathrm{GAP}}, \\ \min\{\ \gamma(r,t-1),\ \gamma(r,t-2),\ \gamma(r-1,t-1),\ \gamma(r-1,t-2)\ \} + c_{\mathrm{FS}}, \\ \gamma^{(0)}(r,t) + c_{\mathrm{INTRON}} + p_{\mathrm{INTEND}}(\sigma_{[t-2,t-1]}), \\ \min\{\ \gamma^{(1)}(r-1,t,n) + \delta^{\mathrm{tr}}(\ s_{[r-1]},\ n\,\sigma_{[t-2,t-1]}\ ) \mid n \in \mathcal{N}\} \\ \qquad + c_{\mathrm{INTRON}} + p_{\mathrm{INTEND}}(\sigma_{[t-4,t-3]}), \\ \gamma^{(2)}(r,t,\sigma_{[t-1]}) + c_{\mathrm{INTRON}} + p_{\mathrm{INTEND}}(\sigma_{[t-3,t-2]}), \end{cases}$$

During the iterations, the predecessor index (r', t') and the corresponding label $q(r,t) \in Q$ are stored, corresponding to the argument maximizing $\gamma(r,t)$. Then the optimal alignment is determined by the backtracking procedure described above. This way, the remaining gaps in the gene prediction are filled.

3.3 Protein classification

After genes have been predicted on the DNA, protein sequence are determined that then can be used for further analysis, such as classification. Conversely, homology-based approaches use the feedback from protein resources for the gene prediction.

By mapping single highly homologous, or identical protein sequences to the genome, the gene structure can in most cases be reconstructed, as the results obtained with Scipio have shown.

Similarity shared by *all* members of a given class can be a type of information that can guide the gene prediction in the case of higher evolutionary distance, effectively performing the protein classification in the same step. This was the main idea behind designing AUGUSTUS-PPX, that I will present in detail in the next chapter. In this section, I will describe models that can be used to represented these similarities.

Relationship between proteins is commonly reflected by sequence similarity: as the sequence determines most of the three-dimensional structure of a protein,

sequence similarity generally implies similar or identical function. Hence proteins with similar sequences are grouped into families, and membership of a query protein to a family is assessed by sequence models.

Along the full length of the sequences, the degree of conservation can vary according to the importance of the sequence regions for the particular function shared by the proteins. These *domains* are characteristic for the protein function.

Protein signatures are representations of protein families that can be used to classify query sequences. They are commonly probabilistic models based on multiple sequence alignments (MSAs). In an MSA, related protein sequences are arranged in a way that conserved sequence positions appear vertically aligned. The gap character "-" is inserted into sequences in an alignment to shift the corresponding motifs underneath each other.

Many publicly accessible protein databases, the most important of them integrated into the meta-database InterPRO (http://www.ebi.ac.uk/interpro/, [HAA+09]), offer services to classify query sequences provided by the user, according to the precomputed signatures.

Profile HMMs

The majority of protein databases use *profile HMMs* to represent the families [DEKM99, GKHC01, MLUL+05, BCD+04, QSP+05]. A profile HMM defines a distribution on member sequences where the underlying state sequence of the HMM represents the way a generated sequence is aligned to the MSA. Parameters of a typical profile HMM are

- for each column $j \in N$ of the MSA, a distribution

  $$P_j \colon \mathcal{A} \to [0, 1]$$

  summing to 1, estimated from the number of occurences of amino acids in column j

- deletion probabilities $P_j^{(\mathrm{Del})}$, estimated from the number of gap characters found in column j

- an insertion probability $P^{(\mathrm{Ins})}$

The architecture of the HMM is defined by:

- a state set

$$Q = \{\mathrm{Del}_1, \ldots, \mathrm{Del}_n\} \cup \{\mathrm{Mat}_1, \ldots, \mathrm{Mat}_n\} \cup \{\mathrm{Ins}_0, \ldots, \mathrm{Int}_n\}$$

  where n is the number of columns of the MSA. The *deletion* states Del_j correspond to a column that is deleted in the query, the *insertion* states Ins_j correspond to extra characters inserted to the query between columns, and the *match* states correspond to a column aligned to a character in the query.

- transition probabilities

$$\begin{aligned}
\mathrm{Tr}_{\mathrm{init}}(\mathrm{Ins}_0) &= P^{(\mathrm{Ins})} & \mathrm{Tr}(\mathrm{Ins}_j \mid \mathrm{Ins}_j) &= P^{(\mathrm{Ins})} \\
\mathrm{Tr}_{\mathrm{init}}(\mathrm{Del}_1) &= P_1^{(\mathrm{Del})} & \mathrm{Tr}(\mathrm{Mat}_j \mid \mathrm{Ins}_{j-1}) &= 1 - P^{(\mathrm{Ins})} \\
\mathrm{Tr}_{\mathrm{init}}(\mathrm{Mat}_1) &= 1 - P^{(\mathrm{Ins})} - P_1^{(\mathrm{Del})} & & \\
\mathrm{Tr}(\mathrm{Ins}_j \mid \mathrm{Mat}_j) &= P^{(\mathrm{Ins})} & \mathrm{Tr}(\mathrm{Del}_j \mid \mathrm{Del}_{j-1}) &= P_j^{(\mathrm{Del})} \\
\mathrm{Tr}(\mathrm{Del}_j \mid \mathrm{Mat}_{j-1}) &= P_j^{(\mathrm{Del})} & \mathrm{Tr}(\mathrm{Mat}_j \mid \mathrm{Del}_{j-1}) &= 1 - P_j^{(\mathrm{Del})} \\
\mathrm{Tr}(\mathrm{Mat}_j \mid \mathrm{Mat}_{j-1}) &= 1 - P^{(\mathrm{Ins})} - P_j^{(\mathrm{Del})} & &
\end{aligned}$$

  By introducing a terminal state as in 2.1, a variable number of insertions is made possible at the end of the sequence: $\mathrm{Tr}(q_{\mathrm{init}} \mid \mathrm{Mat}_n) = \mathrm{Tr}(q_{\mathrm{init}} \mid \mathrm{Ins}_n) = 1 - P^{(\mathrm{Ins})}$, $\mathrm{Tr}(q_{\mathrm{init}} \mid \mathrm{Del}_n) = 1$.

- emission probabilities

$$\begin{aligned}
\mathrm{Em}(a \mid \mathrm{Mat}_j) &= P_j^{(\mathrm{Mat})}(a) \\
\mathrm{Em}(\epsilon \mid \mathrm{Del}_j) &= 1 \\
\mathrm{Em}(a \mid \mathrm{Ins}_j) &= P_{\mathrm{back}}(a)
\end{aligned}$$

  where ϵ is the empty string, and $P_{\mathrm{back}}(a)$ is the background (random) distribution of amino acids, given by their global frequencies

Note that this HMM is generalized by allowing the deletion states to emit empty strings only (*silent* states). By choosing appropriate transition probabilites and eliminating the silent states from Q, it can be regarded as an HMM in the original sense. The Viterbi algorithm can be used to produce an alignment, by assigning to the coordinates of the query sequence the aligned columns of the MSA, given by the state sequence. Moreover, the joint probability reflects the quality of the alignment, and can be used as classificator, after choosing an appropriate threshold.

Block profiles

In AUGUSTUS-PPX, protein families are modelled by block profiles. In this context, a *block* is an ungapped and highly conserved section of a multiple sequence alignment. The concept of a block was first introduced by Henikoff [HH91], and then used to classify proteins in the Blocks database [PHH96, HHP99].

Very similarly, collections of blocks, together with the ranges of admissible distances between consecutive blocks, have been used as protein signatures, referred to as *fingerprint*[AB94] and currently collected in the PRINTS database,[ABF+03] a member database of InterPRO.

A *block profile* is a set of n frequency matrices, each generated from one block, containing the column-specific frequencies of amino acids: $(P_i^{(b)}(a))$, $b = 0, \ldots, n-1$, $i = 0, \ldots, w-1$, $a \in \mathcal{A}$ ($w = w_b$ denoting the block width). Let

$$R_i^{(b)}(a) = \frac{P_i(a)}{P_{\text{back}}(a)}$$

denote the odds ratio obtained by dividing by the global distribuion P_{back} of amino acids. R serves as a scoring matrix for sequence motifs: given an amino acid sequence $s = s_0 \ldots s_{w-1}$, its similarity to the block is expressed by

$$\rho^{(b)}(s) = R_0(s_0) \cdot \ldots \cdot R_{w-1}(s_{w-1}).$$

More generally, a *partial block score* $\rho^{(b)}_{[j..k]}(s) = R_j(s_0) \cdot \ldots \cdot R_k(s_{k-j})$ can be defined if s is a (shorter) sequence of length $k - j + 1$.

The order of the blocks is assumed to be preserved throughout all sequences in the family. For each b, an interval $I_b = [d_b^{\min}, d_b^{\max}]$ specifies the range of admissible distances between consecutive block motifs (or, in the cases $b = 0$, $b = n$, between the first/last block motif and the sequence start/end, respectively).

A mapping of the profile to a protein sequence s is defined by a series of block starts $J = (j_0, \ldots, j_{n-1})$, giving rise to a *profile score*

$$\rho(s; J) = \rho^{(0)}(m_0) \cdot \ldots \cdot \rho^{(n-1)}(m_{n-1}) \tag{3.1}$$

where $m_b = s_{[j_b,\ldots,j_b+w_b-1]}$ is the motif mapped to block b, provided that consecutive motifs are in admissible distance from each other:

$$j_0 \in I_0; \quad j_b - j_{b-1} - w_{b-1} \in I_b \quad (b = 1, \ldots n)$$

(setting $j_n := |s|$ to the sequence length of s).

In contrast to profile HMMs, this profile score is not Bayesian: there is no a-priori distribution on mappings. While in a profile HMM, insertions and deletions are penalized, here the score of a mapping depends only on the sequence motifs the mapping aligns to blocks. The odds score reflects the ratio of probabilities in the background model to the model defined by the mapping; both sequence models are identical in inter-block regions.

Chapter 4

AUGUSTUS-PPX: A new hybrid gene prediction method

In this chapter, a new method is described that combines ab-initio gene prediction with the information contained in a protein profile. The method has been implemented, and integrated into the existing ab-initio gene prediction program AUGUSTUS. It was tested successfully, showing considerably improved prediction results. The publication based on this work is attached in Appendix C, and gives a detailed overview on the obtained results.

4.1 Motivation

Because of the enormous speed that new genomic data is produced, only a very small part of it can be annotated manually. On the other hand, the amount of data available as extrinsic input to automated annotation methods is equally growing. It is therefore crucial to explore ways to integrate all possible kinds of information that can improve the accuracy of gene prediction.

Since evidence is not available for all genomic regions, ab-initio methods are still an essential part of the prediction process. Based on an ab-initio approach, AUGUSTUS can be viewed as a platform for the integration of all types of evidence produced by third-party tools (usually alignment-based), considering these as *hints* of adjustable reliability [Sta03, SSMW06]

As the next step in the annotation pipeline, the classification of newly predicted genes depends strongly on the accuracy of the gene prediction. Existing protein resources can help guiding the prediction. While many researchers are focused on particular protein families of interest, comprehensive databases containing protein data are widespread, and easily accessible, for example via InterPRO.

The protein profile extension (PPX) to AUGUSTUS is designed to combine AUGUSTUS' ab-initio model with a second probabilistic model for membership to a protein family. The additional input that AUGUSTUS-PPX takes is a protein signature in the form of a block profile. In contrast to the *hints* approach, the evidence is not produced by a separate program but created in parallel to the ab-initio prediction, allowing mutual interaction of the models. However, hints from complementary sources can work together with this approach to include more kinds of extrinsic information.

Two goals are pursued with the protein extension. First, the identification of family members during the gene prediction, and second, more importantly, a potential of correcting the prediction in a way that improves identification rates for these genes.

The problem addressed by this combined approach is the prediction of gene structures subject to the condition that the encoded protein sequence fits to the protein model. To this end, the similarity of candidate genes to the profile is evaluated based on candidate mappings of its blocks to the DNA; gene structures equipped with a valid mapping are awarded a bonus to their ab-initio score. An extended version of the Viterbi algorithm takes into account the bonus of potential profile mappings for the determination of the highest-scoring parse.

4.2 Profile-DNA mappings

A block profile is mapped to the DNA sequence σ by specifiying the start locations $(t_0, \ldots, t_{n-1})$ of the segments coding for potential block motifs. We denote any such mapping by ψ. Since blocks may be interrupted by introns, the full mapping is well-defined only when also a candidate gene structure ϕ is given, as depicted in Figure 4.1. In this case, ψ is *valid* if all t_b lie on exons belonging to the same

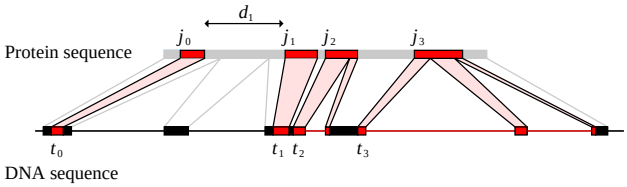

Figure 4.1: A profile-DNA mapping. The translated transcript (above) contains four block motifs (in red) $m_0,\ldots,m_3$, separated by inter-block sequence (in grey) of length d_b. The coding gene (below) consists of six exons; sequences coding for the block motifs are shown in red. Block hits may appear inside an exon (m_0,m_1) or be disconnected by one or more introns (m_2,m_3).

gene, and the inter-block regions on the protein sequence have admissible lengths $d_b^{\min} \leq d_b \leq d_b^{\max}$.

The score of the mapping is defined by equation (3.1): if s is the protein sequence of the gene, and $\tilde{\psi}$ are the protein coordinates of the mapping ψ, then we write

$$\rho(\sigma;\phi;\psi) = \rho(s,\tilde{\psi}) = \rho^{(0)}(m_0)\cdot\ \ldots\ \cdot\rho^{(n-1)}(m_{n-1}). \tag{4.1}$$

Several genes may be equipped with mappings $\psi_1,\psi_2,\ldots,\psi_r$, resulting in a total score of

$$\rho(\sigma;\phi;\psi_1,\ldots,\psi_r) = \rho(\sigma;\phi;\psi_1)\cdot\ \ldots\ \cdot\rho(\sigma;\phi;\psi_r).$$

Formally, ρ can be defined to be 0 for invalid mappings. When provided with a protein profile, the Viterbi algorithm will search for the best-scoring combination of gene structure and compatible profile-DNA mappings. More precisely, it determines $\phi,\psi_1,\ldots,\psi_r$ maximizing the combined score

$$P(\sigma,\phi)\cdot\rho(\sigma;\phi;\psi_1,\ldots,\psi_r), \tag{4.2}$$

where the joint probability $P(\sigma,\phi)$ for sequence and gene structure is multiplied with the *profile bonus* $\rho(\sigma,\phi,\psi)$ for each candidate transcript equipped with a valid mapping ψ. Thus, only if a gene is compatible to a profile mapping with a score high enough to compensate for a lower ab-initio score, the prediction with the profile will differ from the prediction without. In particular, on sequences where no members of the protein family are identified, the result will be identical

to the ab-initio prediction. In the underlying sequence model, multiplying with ρ effectively amounts to replacing the background model with the block model, in the exon sequence model.

For performance reasons, only mappings that consist of *block hits* are considered for the evaluation of the profile bonus. To classify a given protein sequence s of length w as a block hit, we turn the scoring function into a decision function by requiring $\rho(s) > \tau$, with a block-specific threshold $\tau = \tau^{(b)}$.

Two global parameters $\theta_0 (= \theta_{\text{spec}})$ and $\theta_1 (= \theta_{\text{sens}})$ independent from b, are specified by the user, designed to ensure that estimated error rates are low.

We consider two competing models, H_0 describing random sequences distributed according to P_{back}, and H_1 describing block motifs from a block b under consideration, distributed according to $P_{\text{block}} = (P_i)_{i=0,\ldots,w}$, the frequency matrix given in the profile. $P_i(a)$ denotes the probability to observe amino acid a at position i of a block motif, while $P_{\text{back}}(a)$ denotes the global probability for a to appear at any position. The odds-ratio is given by $R_i(a) = \frac{P_i(a)}{P_{\text{back}}(a)}$; for convenience, we consider in the following the log-odds ratio $L_i(a) = \log R_i(a)$, turning the product $\rho(s) = R_0(s_0) \cdot \ldots \cdot R_{w-1}(s_{w-1})$ into a sum $\ell(s) = \log \rho(s) = L_0(s_0) + \ldots + L_{w-1}(s_{w-1})$. Each of the two models H_j gives rise to a different expectation value $\mu_j = \mathrm{E}_j(L)$ and variance $\sigma_j^2 = \mathrm{Var}_j(L)$ for the log-scores.

By putting the global parameters into the block-specific scale, we obtain thresholds $\tau^- = \mu_0 + \theta_0\,\sigma_0,\ \tau^+ = \mu_1 - \theta_1\,\sigma_1$. A score exceeding τ^- is *at least* θ_0 standard deviations above the expected score in the background model, and a score below τ^- is *at most* θ_1 standard deviations less than the expected score for a block motif. The probability for a random sequence (in either model) to have a score in this range can be approximated with the Gaussian distribution by

$$1 - \Phi(\theta_0) \qquad \text{and} \qquad 1 - \Phi(\theta_1),$$

where Φ is the cumulative Gaussian distribution function; hence, these numbers are bounding the estimated error rates.

For example, the values $\theta_0 = 4.5$ and $\theta_1 = 1.5$ used as default values correspond to a false-positive rate less than $1 - \Phi(4.5) = 3.3 \cdot 10^{-6}$ (one block hit in a random

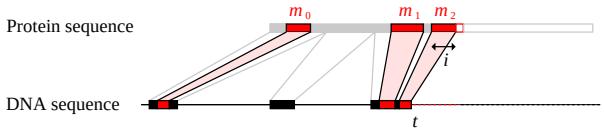

Figure 4.2: Calculating Viterbi variables at DNA position t, for substate (b, i), $b = 2$, $i > 0$. The score is maximized for all gene structures that have a mapping to the truncated block motif m_2 of length i at the end of the exon. Any parse continuing from here must have the next exon starting with the remainder of the motif. The bonus awarded to the Viterbi variable for this mapping is $\rho^{(0)}(m_0) \cdot \rho^{(1)}(m_1) \cdot \rho^{(2)}_{[0..i-1]}(m_2)$.

amino acid sequence of 300 000 residues), and to a sensitivity of at least $\Phi(1.5) = 93.3\%$.

In order to fulfill both conditions, τ must satisfy $\tau^- \leq \tau \leq \tau^+$. Provided that $\tau^- \leq \tau^+$, we set $\tau = \frac{1}{2}(\tau^- + \tau^+)$. Any sequence satisfying the condition $\ell^{(b)}(s) > \tau$, or equivalently, $\rho^{(b)}(s) > \exp(\tau)$, is then considered a block hit. In the case $\tau^- > \tau^+$, or if $w < 6$, the block is removed from the profile for the evaluation.

4.3 Integration into AUGUSTUS' state model

Iterating over all DNA positions, the Viterbi algorithm computes the scores of optimal partial parses (candidate gene structures) ending in the current position, and stores them in variables indexed by position and last state, each state representing a different sequence type, distinguished by strand and, if applicable, by reading frame.

When equipped with a protein profile, the score assigned to a parse is modified as described above: for each gene in the parse that is compatible with a profile-DNA-mapping, the score is to be multiplied with the best possible profile bonus; each exon in the gene contributes the bonus for the hits for a full or partial block mapped onto it.

When the algorithm arrives at a position in the DNA, it must consider all partial profile mappings started before that can be continued beyond this position, having a particular position within the profile aligned to the current DNA position,

as in Figure 4.2. This position determines the conditions for the mapping to be continued; thus, a separate score has to be stored at that position for each profile position. To this end, the original state space of AUGUSTUS was extended by a set U of *secondary states* attached to the main state, representing the position in the profile:

$$U = \{u = (b, i) \mid b = 0, \ldots, n;\ i = -d^{(b)}_{\max}, \ldots, w_b\} \cup \{X\}$$

where b denotes the block ($b = n$ the region behind the last block), and i the position relative to the block. The state X represents an undefined position, for genes not matching the profile.

Positive i denotes a position inside block b, while negative i represents a position before block b, determined by the position of the previous block. In the latter case, $-i$ refers to the *maximum* allowed inter-block columns to follow; the earliest admissible block start is at $i = -d^{(b)}_{\max} + d^{(b)}_{\min}$, the latest is at $i = 0$.

For a partial profile mapping ending at position (b, i), the profile score multiplied to the ab-initio score is computed as in equation (4.1), except that the product is truncated at the current position:

$$\rho^{(0)}(m_0) \cdot \ \ldots \ \cdot \rho^{(b-1)}(m_{b-1}) \cdot \rho^{(b)}_{[0..i-1]}(m_b)$$

(where the last factor is included only in the case $i > 0$, as in Figure 4.2).

If a gene structure is equipped with a profile-DNA mapping, they define a sequence of secondary states in the form of profile positions aligned to the DNA coordinates specified in the parse; an *extended parse* is a parse together with such a sequence of secondary states:

$$\phi^+ = (\phi; u_1, \ldots, u_n) = \big((q_1, t_1, u_1), \ldots, (q_n, t_n, u_n)\big);$$

by setting $u_j = X$ for all j, a parse without profile hits can be viewed an extended parse as well.

We define the score of an extended parse as the maximum of the combined scores of all profile-DNA mappings that agree with it: Let q is an exon state, and u', u profile positions at exon start t' and end t. The exon bonus $\rho(u', t', u, t)$ is

defined as follows: If $i' > 0$, then the exon must start with the a block suffix, and contributes the partial block score $\rho^{(b')}_{[i'..w_{b'})}(m)$, where m is the translation of the exon part overlapping with the block; an admissible mapping starts with block $b'+1$ at a suitable distance. If $i' \leq 0$, then it determines the admissible block start for block b'. Consecutive blocks may be mapped onto the exon, contributing $\rho^{(b)}(m)$ to the exon bonus, provided they satisfy the distance constraints given in the profile. There may also be no block starting on the exon if it is short enough to fit into the admissible inter-block region.

Each admissible mapping defines a profile position u at the exon end, depending on the last mapped block. $\rho(u', t'; u, t)$ is is the contribution of the block scores aligned to the exon (partial block scores, if it starts or ends inside a block). If several mappings are compatible with (u', u) on the same exon, then maximum of them is taken.

The extended feature function is then given by $f(z', u', z, u) = f(z', z) \cdot \rho(u', t'; u, t)$, $z = (q, t)$, if q is an exon state.

Intron states inside a profile mapping are not contributing a bonus, but need to be equipped with secondary states as well, in order to keep track of the profile position for the following exon. For introns, we have and $f(z', u', z, u) = f(z', z)$ if $u' = u$, and $f(u', t', u, t) = 0$ otherwise.

It follows that

$$P(\sigma, \phi^+) = P(\sigma, \phi) \cdot \rho(\sigma, \phi, \psi_1, \ldots, \psi_r)$$

if $\psi_1, \ldots, \psi_r$ is the highest scoring set of profile-DNA mapping compatible with the secondary state sequence $u_1, \ldots, u_n$ (including the case $r = 0$ that no gene matches the profile, corresponding to $u_j = X$).

The Viterbi variables used in the extended model are indexed by state and secondary state:

$$\gamma(z, u) = \max_{z', u'} \gamma(z', u') \cdot f(z', u', z, u)$$

Each pair of secondary states give rise potentially to a different profile bonus for the same exon. To compute the extended variables, the combined scores have to be maximized iterating over all predecessor DNA and profile positions.

Genes on the backward strand are evaluated essentially in the same way as genes on the forward strand; however, in this case, the profile is mapped to the DNA "backwards", starting with the C-terminus. Correspondingly, the secondary states on the backward strand refer to blocks and columns in reverse order: the first motif is evaluated by the last block in the profile, starting with the last block column.

4.4 Speed-up and Memory-saving Strategies

Since the model attaches a high number of secondary states to every main state of the original model, an exhaustive evaluation of all combinations of block locations and secondary states would become prohibitively slow; therefore, most of the secondary state entries are eliminated or shared between main states in order to control the computational cost. Secondary scores are stored in the Viterbi matrix dynamically, reserving memory only for nonzero entries.

Precomputing block hit collections

To prepare the search for whole blocks found on the same exon, the target sequence is searched for block hits in parallel to the main algorithm, and the hits are stored in collections of consecutive hits satisfying the distance conditions. Motifs that do not score over the threshold τ defined in the end of section 4.2, are not considered for the collections. In practice, this commonly leads to exon candidates with very few block hits or none at all allowed on them, preventing the vast majority of secondary state entries to be created.

Scores for blocks truncated at exon borders are calculated only once for every location. Furthermore, similar to the case of full blocks, truncated blocks are subject to filtering with thresholds if they exceed a minimum length; again, this leaves only few values for $i > 0$ actually stored as secondary states.

Removing dominated states

In the situation that inter-block positions $u = (b, i)$ with $i < 0$ are calculated, i constrains the admissible block start on the upcoming exon(s).

We call $u = (b, i)$ *dominated* by two neighbouring positions (b, i') and (b, i'') if these cover the admissible block starts defined by u.

This is the case if $i' < i < i''$ and $i'' - i' \leq d_b^{\max} - d_b^{\min}$. If both of the dominating profile positions have a higher score, the entry at (b, i) may be deleted from the Viterbi table.

Pruning dead state graph branches

In order to reduce the memory needed for the extended Viterbi table, we remove from it all entries that are not contained in any parse reaching the current DNA location. To this end, for each entry, a counter is installed and incremented every time the entry is maximizing the partial score for some successor state. An entry for a secondary state is deleted if its successor count is zero at the time the algorithm has proceded to a location in the target sequence beyond the maximal state length from the entry. This helps saving considerable amounts of memory in predictions on long sequences.

Sharing substate tables

As the profile location is constant throughout an intron, memory can be saved by sharing the state tables between consecutive intron states with a fixed length. AUGUSTUS' state model has intron states emitting a single nucleotide, and candidate parses evaluated in the course of the Viterbi algorithm contain long sequences of the single-nucleotide intron states in every long intron, mostly with identical entries for secondary states, just differing by a constant factor. Instead of using a separate copy for the secondary states attached to the main intron state, for each nucleotide position, only the constant factor is stored, and a link to the table containing the secondary states of the predecessor (the last DNA location that an exon candidate contributed to the intron score).

4.5 An algorithm for fast block search

To make it possible to run AUGUSTUS-PPX on a whole-genome scale, we devised a fast preliminary search algorithm for determining the genomic regions that are likely to contain genes matching the profile, without determining detailed gene structures. The algorithm was implemented in a separate executable that can be used before running AUGUSTUS. Given a target sequence and a protein profile, it performs the following steps:

- Building a seed collection: For each of the 8 000 possible triplets of amino acids, the positions in the profile are stored where that triplet is likely to occur.
- Determining block hit candidates: iterating over the target sequence, each 9-tuple is tested for being a seed. Any segment covered at least 25% by seeds for a block is considered for exhaustive evaluation. This prefiltering step is very fast, since it consists of a simple lookup with amino acid triples as keys.
- Exhaustive evaluation of candidates: given a block start offset, partial scores $\rho_{[j..k]}(a_j..a_k)$ are maximized where $a_0..a_{w-1}$ is the translated DNA segment. This interval is stored as a partial block hit if its size reaches the minimal block width and the (partial) block threshold is exceeded.
- Assembling block hits to profile hits: each block hit is extended to a series of block hits by joining neighbouring hits, allowing for blocks being skipped. Using dynamic programming, sequences of block hits are determined that are highest-scoring for all contained blocks, and returned as profile hits.

This algorithm is fast enough to be run on whole genomes with a high number of profiles, in order to determine the regions AUGUSTUS-PPX is then run on.

4.6 Discussion

In order to take into account information about membership to protein families of predicted genes, a block profile representing one family of interest was accepted as

additional input to the gene prediction program AUGUSTUS. The blocks of such a profile were mapped to the DNA, imposing length constraints and deviating nucleotide composition to the exon model. Gene structures following the modified model were assigned a series of secondary states.

In general, secondary states can be used to model side conditions that influence the score of a parse. For example, constraints on the total gene length may be imposed, or they can keep track of split intron information, for the exclusion of spliced stop codons. Here, secondary states are aligning the profile to the exon ends.

Except for technicalities, secondary or *sub*-states can be characterized by the propery that they do not follow the Bayesian approach: While in a GHMM, an *a-priori* distribution for parses is defined by transition probabilities and state length distributions, any legal sequence of secondary states may be assigned may be assigned to the parse, without affecting the a-priori distribution. The result is a mixture of a Bayesian and a maximum-likelihood approach: Given a parse, the choice of secondary state depends only on the target sequence; all secondary sequences have the same chance to be used.

The reason for this choice is the *label bias* [LMP01]: a problem occuring in some GHMMs when states that have many successor states compete with states that do not. The parse returned by the Viterbi algorithm is the most likely single parse, and will less frequently contain states with many successor states, since all these successor states have to share the "mass" coming from the predecessor. This is especially undesired if the successor states share the same label (what the user sees as output), and are distinguished merely to implement side constraints.

In order to assess the prediction accuracy of AUGUSTUS-PPX, it was equipped with the profile of several protein families, and test runs were performed on genomes at different evolutionary distances. The results showed a significant increase in prediction quality, compared to the ab-initio approach, but also to competing software using protein profiles. The number of genes predicted with high accuracy showed a dramatic increase. For detailed results, see the published article in appendix C.

Appendix

Appendix A

Article about SIGI-HMM

"Score-based prediction of genomic islands in prokaryotic genomes using hidden Markov models"

published in BMC Bioinformatics (2006), **7**, 142.

Score-based prediction of genomic islands in prokaryotic genomes using hidden Markov models

Stephan Waack[1], Oliver Keller[1], Roman Asper[1], Thomas Brodag[1], Carsten Damm[2], Wolfgang Florian Fricke[3], Katharina Surovcik[1], Peter Meinicke[4] and Rainer Merkl*[5]

Address: [1]Institut für Informatik, Universität Göttingen, Lotzestr. 16–18, 37083 Göttingen, Germany, [2]Institut für Numerische und Angewandte Mathematik, Universität Göttingen, Lotzestr. 16–18, 37083 Göttingen, Germany, [3]Göttingen Genomics Laboratory, Universität Göttingen, Grisebachstr. 8, 37077 Göttingen, Germany, [4]Institut für Mikrobiologie und Genetik, Universität Göttingen, Goldschmidtstr. 1, 37077 Göttingen, Germany and [5]Institut für Biophysik und Physikalische Biochemie, Universität Regensburg, Universitätsstr. 31, 93053 Regensburg, Germany

Email: Stephan Waack - waack@cs.uni-goettingen.de; Oliver Keller - keller@cs.uni-goettingen.de; Roman Asper - asper@cs.uni-goettingen.de; Thomas Brodag - Thomas.Brodag@T-Online.de; Carsten Damm - damm@math.uni-goettingen.de; Wolfgang Florian Fricke - wfricke@gwdg.de; Katharina Surovcik - surovcik@cs.uni-goettingen.de; Peter Meinicke - pmeinic@gwdg.de; Rainer Merkl* - Rainer.Merkl@biologie.uni-regensburg.de

* Corresponding author

Published: 16 March 2006

BMC Bioinformatics 2006, **7**:142 doi:10.1186/1471-2105-7-142

Received: 09 December 2005
Accepted: 16 March 2006

This article is available from: http://www.biomedcentral.com/1471-2105/7/142

Abstract

Background: Horizontal gene transfer (HGT) is considered a strong evolutionary force shaping the content of microbial genomes in a substantial manner. It is the difference in speed enabling the rapid adaptation to changing environmental demands that distinguishes HGT from gene genesis, duplications or mutations. For a precise characterization, algorithms are needed that identify transfer events with high reliability. Frequently, the transferred pieces of DNA have a considerable length, comprise several genes and are called genomic islands (GIs) or more specifically pathogenicity or symbiotic islands.

Results: We have implemented the program SIGI-HMM that predicts GIs and the putative donor of each individual alien gene. It is based on the analysis of codon usage (CU) of each individual gene of a genome under study. CU of each gene is compared against a carefully selected set of CU tables representing microbial donors or highly expressed genes. Multiple tests are used to identify putatively alien genes, to predict putative donors and to mask putatively highly expressed genes. Thus, we determine the states and emission probabilities of an inhomogeneous hidden Markov model working on gene level. For the transition probabilities, we draw upon classical test theory with the intention of integrating a sensitivity controller in a consistent manner. SIGI-HMM was written in JAVA and is publicly available. It accepts as input any file created according to the EMBL-format.

It generates output in the common GFF format readable for genome browsers. Benchmark tests showed that the output of SIGI-HMM is in agreement with known findings. Its predictions were both consistent with annotated GIs and with predictions generated by different methods.

Conclusion: SIGI-HMM is a sensitive tool for the identification of GIs in microbial genomes. It allows to interactively analyze genomes in detail and to generate or to test hypotheses about the origin of acquired genes.

Background

Horizontal gene transfer (HGT) is a process that results in the acquisition of novel genes originating from perhaps taxonomically unrelated species. This phenomenon is frequent among microbes and is considered a means of rapid adaptation to changing environmental demands [1]. Pieces of DNA acquired *via* HGT frequently have a considerable length. These patches have been called genomic islands (GI) or due to their role and more specifically pathogenicity islands [2] or symbiotic islands [3].

Several methods have been developed for the prediction of GIs based on different approaches to identify putatively alien (pA) genes [4-12]. Each of these concepts has specific preferences and drawbacks; for recent reviews see [13,14]. In the following, we describe an approach which relies on the genome theory postulating a rather homogeneous codon usage within a genome [15]. The algorithm exploits taxon specific differences in codon usage for the identification of pA genes and the prediction of their putative origin. Hidden Markov models (HMMs) are a state of the art concept in computational learning theory. A sequence of observations is considered as being emitted from the states of an invisible Markov chain. The Viterbi algorithm efficiently computes a sequence of states that have the maximal posteriori probability given a certain sequence of observations and fixed transition and emission probabilities. The challenge in designing a HMM is representing the real situation adequately in order to generate relevant predictions. HMM have proved useful in many applications. In the case of predicting eukaryotic genes, for example, the programs GENSCAN [16,17], HMMGene [18,19], GenomeScan [20], AUGUSTUS [21], and AUGUSTUS+ [22] are HMM-based.

It has been shown that HMMs allow to predict GIs [9]. GIs have typically a considerable length, therefore we have decided to implement a HMM assessing GI prediction on the *gene* level. GIs can originate from a variety of *a priori* unknown donors. Therefore, it is difficult to assure sufficient test statistics. We will describe an approach named SIGI-HMM. To some extent, it is based on principles introduced with SIGI [23]. This program was used to analyze individual genomes [24,25] and to study the content of genomic islands in general [26] as well as to characterize gene-flux between bacteria and archaea [27]. For SIGI-HMM we substituted a heuristic approach with a HMM. SIGI-HMM has only few parameters to adjust. The most relevant one is a sensitivity controller which affects transitions of the HMM in a consistent manner. We will demonstrate and assess the performance of SIGI-HMM by analyzing genomes in detail.

Implementation

We have implemented SIGI-HMM in Java as a first module of our software suite COLOMBO intended as a workbench for the statistical analysis of genomic data. The program can be downloaded from [28]. The download package contains also the program Artemis [29], which is used to visualize the output of SIGI-HMM. After the installation, a genomic dataset formatted in EMBL-format can be loaded and analyzed. SIGI-HMM creates several lists containing the predictions in GFF-format or tabulated. Predictions are classified according to the categories NATIVE and PUTAL. In addition, a modified EMBL-formatted file is generated containing both the original annotation and the predictions. This file can be fed into Artemis in order to color-code and visualize genome content. Thus, the user can interactively study the composition of genomes. Intentionally, only few parameters can be manipulated by the user: The sensitivity controller and the gap length which decides on merging single GIs to larger ones. In addition, the user can supplement the list of putative donors we have deduced from the CUTG database (see below). The default value of the the sensitivity controller was chosen to give predictions consistent with published results; see Table 1. If it is known that the genome under study contains GIs, we propose the following approach in order to optimize sensitivity of SIGI-HMM: Starting from a low value, sensitivity should be increased until all known GIs appear. If new islands emerge, they show the same degree of codon usage bias and should be considered GIs.

Results

The following text is organized as follows: First we introduce data models, the scoring system and the architecture of the HMM. Then we evaluate the predictive power of the algorithm and present analyses of several genomes.

Stochastic data models

Let $\mathcal{G}$ be a series of genes as deduced from a genome coding for proteins $\mathcal{P}$. For each codon c we count its occurrence $\#c$ in $\mathcal{G}$. We define the *synonymous frequency* $q_{ac} \in [0,1]$ as the ratio of $\#c$ divided by the occurrence of the amino acid a encoded by c in $\mathcal{P}$. The *frequency* $q_c \in [0,1]$ of c in $\mathcal{G}$ is defined as $\#c$ divided by the occurrence of all codons in $\mathcal{P}$.

Now let $\mathcal{G}_0$ be a prokaryotic genome whose genomic islands have to be predicted and let $\mathcal{G}_1$, $\mathcal{G}_2$, ..., $\mathcal{G}_r$ be genomes assumed to be the donors for pA genes occurring in $\mathcal{G}_0$. We consider $\mathcal{G}_1$ to $\mathcal{G}_r$ as representatives of taxa $\mathcal{T}_1$ to $\mathcal{T}_r$ which are assumed to be the putative sources of $\mathcal{G}_0$'s alien genes. For each protein (i.e. sequence of

Table 1: A comparison of pA predictions for prokaryotic species. SIGI-HMM was used to identify GIs. The accumulated length of genes constituting GIs is given in percent in column pA DNA. This transformation allows to compare results with entries of column Foreign DNA, which was reproduced from [41]. The column Length lists the genome size im Mbp.

Species	Length [Mbp]	Foreign DNA [%]	pA DNA [%]
Escherichia coli K-12	4.64	12.8	9.3
Bacillus subtilis	4.21	7.5	7.6
Synechocystis PCC6803	3.57	16.6	5.0
Deinococcus radiodurans	2.65	5.2	4.8
Archaeoglobus fulgidus	2.18	5.2	4.2
Aeropyrum pernix	1.67	3.2	1.5
Thermotoga maritima	1.86	6.4	1.0
Pyrococcus horikoshii	1.74	2.7	2.9
Methanobacterium thermoautotrophicum	1.75	9.4	1.6
Haemophilus influenzas Rd KW20	1.83	4.5	1.6
Helicobacter pylori 26695	1.67	6.2	0.0
Aquifex aeolicus	1.55	9.6	1.8
Methanocaldococcus jannaschii	1.66	1.3	0.2
Treponema pallidum	1.14	3.6	0.3
Borrelia burgdorferi	0.91	0.1	8.5
Rickettsia prowazekii	1.11	0.0	0.0
Mycoplasma pneumoniae	0.82	11.6	3.8
Mycoplasma genitalium	0.58	0.0	0.2

amino acids) $\pi = a_1, a_2, ..., a_n$ that is encoded by a gene g of genome $\mathcal{G}_0$ (given by the sequence of codons $c_1, c_2, ..., c_n$), and for each $\rho = 0, 1, ..., r$, we define the probability

$$P_\rho(g \mid \pi) := q^{(\rho)}_{a_1c_1} \cdot q^{(\rho)}_{a_2c_2} \cdot \ldots \cdot q^{(\rho)}_{a_nc_n}, \qquad (1)$$

where $q^{(\rho)}_{ac} \in [0,1]$ is the synonymous frequency in genome $\mathcal{G}_\rho$ as defined above.

Scoring scheme

We utilize the odds ratio

$$\frac{P_0(g \mid \pi)}{P_\rho(g \mid \pi)} = \frac{q^{(0)}_{a_1c_1} \cdot q^{(0)}_{a_2c_2} \cdot \ldots \cdot q^{(0)}_{a_nc_n}}{q^{(\rho)}_{a_1c_1} \cdot q^{(\rho)}_{a_2c_2} \cdot \ldots \cdot q^{(\rho)}_{a_nc_n}}$$

in the following way as a *scoring scheme*. The codon usage of g originating from $\mathcal{G}_0$ resembles more the prevalences of $\mathcal{G}_\rho$ if

$$\tau_{\rho,\alpha} > \frac{P_0(g \mid \pi)}{P_\rho(g \mid \pi)}. \qquad (2)$$

If this is the case for some ρ and if

$$\rho^* = \arg\min_{\rho \in \{1,2,\ldots,r\}} \frac{P_0(g \mid \pi)}{P_\rho(g \mid \pi)},$$

then gene g is considered to be pA originating from taxon $\mathcal{T}_{\rho^*}$ represented by genome $\mathcal{G}_{\rho^*}$. This principle of deducing the putative donor has previously been introduced and validated [23].

How to choose the thresholds $\tau_{\rho,\alpha}$ needed in Equation 2? Let $\mathcal{F}_\pi$ be the set of all theoretically possible genes coding for protein π. For each $\rho \in \{1, 2, ..., r\}$, we consider the statistic

$$t_\rho(G) := \ln \frac{P_0(G \mid \pi)}{P_\rho(G \mid \pi)},$$

where $G \in \mathcal{F}_\pi$ is a random element distributed according to $P_\rho(\cdot \mid \pi)$. Having computed the mean μ_ρ and the standard deviation σ_ρ of $t_\rho(G)$, we apply the central limit theorem: The random variable $1/\sigma_\rho(t_\rho(G) - \mu_\rho)$ is approximately distributed according to the standard normal distribution with the cumulative distribution function Φ. We determine the value $\tau_{\rho,\alpha}$ such that

$$\alpha = 1 - \Phi\left(\frac{\ln \tau_{\rho,\alpha} - \mu_\rho}{\sigma_\rho}\right).$$

The parameter α serves as SIGI-HMM's sensitivity controller. It can be adjusted by the user. Please note that the

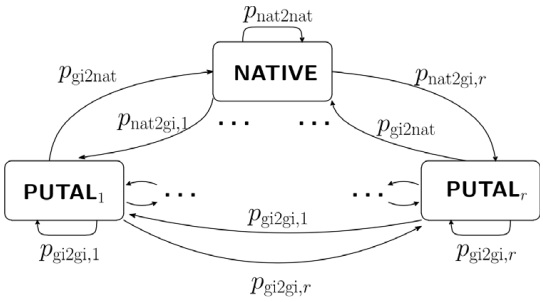

Figure 1

States and transition probabilities of SIGI-HMM's Markov chain. The state NATIVE represents genes which are unsuspicious with respect to synonymous codon frequencies. For ρ = 1, 2,..., *r*, the state PUTAL$_\rho$ models genes, whose codon usage resembles more the prevalences of genomes $\mathcal{G}_\rho$ which represents taxon $\mathcal{T}_\rho$. Each transition from state *x* to state *y* is characterized by its transition probability p_{x2y}. In order to model the mosaic structure of GI composition, transitions from any state PUTAL$_\rho$ to any other one PUTAL$_\sigma$ are allowed.

impact of parameter α onto the decision is independent of $\mathcal{G}_0$ and $\mathcal{G}_\rho$.

Eliminating putatively highly expressed genes

In several genomes, highly expressed genes show a specific codon usage which deviates from the average one and resembles codon prevalences observed in genes coding for ribosomal proteins; see e.g. [8]. We name these genes *putatively highly expressed* (PHX). On the one hand, it is unlikely that these genes were acquired *via* HGT. On the other hand, methods based on codon usage tend to classify them as pA. This needs to be prevented explicitly. We use an approach similar to the GCB score introduced in [30]. It was shown that this methods is one of the best to predict gene expressivity [31]. Let $q_{ac}^{(0,\mathrm{rib})}$ be the synonymous codon frequencies for the ribosomal genes of genome $\mathcal{G}_0$ and let

$$P_{0,\mathrm{rib}}(g \mid \pi) := q_{a_1c_1}^{(0,\mathrm{rib})} \cdot q_{a_2c_2}^{(0,\mathrm{rib})} \cdot \ldots \cdot q_{a_3c_3}^{(0,\mathrm{rib})}. \qquad (3)$$

If

$$t_{\mathrm{rib}}(g) := \ln \frac{P_{0,\mathrm{rib}}(g \mid \pi)}{P_\rho(g \mid \pi)} > \theta,$$

we consider the gene *g* as not alien (see [2,8]).

The threshold θ is determined as follows: Let μ_0 and $\mu_{0,\mathrm{rib}}$ be the mean values and σ_0 and $\sigma_{0,\mathrm{rib}}$ be the standard deviations of the test statistic $t_{\mathrm{rib}}(G)$, where *G* is distributed according to $P_0(\cdot \mid \pi)$ and $P_{0,\mathrm{rib}}(\cdot \mid \pi)$, respectively. The distribution functions of $1/\sigma_0(t_{\mathrm{rib}}(G) - \mu_0)$ and $1/\sigma_{0,\mathrm{rib}}(t_{\mathrm{rib}}(G) - \mu_{0,\mathrm{rib}})$ are approximately standard normal. We choose θ in such a way that

$$1 - \Phi\left(\frac{\theta - \mu_0}{\sigma_0}\right) = \Phi\left(\frac{\theta - \mu_{0,\mathrm{rib}}}{\sigma_{0,\mathrm{rib}}}\right). \qquad (4)$$

Thus, the error of the first and second kind are of equal size.

Architecture of the HMM

Figure 1 depicts the architecture of the implemented HMM. The state NATIVE corresponds to genes having an unsuspicious codon usage. The states PUTAL$_1$, PUTAL$_2$,..., PUTAL$_r$ represent putatively alien genes originating from taxa $\mathcal{T}_1$ to $\mathcal{T}_r$. GIs frequently have a mosaic structure which is due to their generation in a multistep process (see [2]). Therefore, we allow transitions from any PUTAL (i.e. donor) state to any other one.

In order to implement our sensitivity controller, we let the transition probabilities depend on the protein under consideration. Thus, the Markov chain presented in Figure 1 is in fact an inhomogeneous one driven by the series $\mathcal{P}_0$ of proteins encoded by $\mathcal{G}_0$. To simplify notation, we have omitted the index π, which refers to the protein. Instead, we identify a protein by its index originating from $\mathcal{P}_0$.

Solving some linear equations, the transition probabilities given in Figure 1 can be determined in such a way that

$$a \cdot \tau_{\rho,\alpha} = \frac{p_{\mathrm{gi2gi},\rho}}{p_{\mathrm{gi2na}}} \quad \text{and} \quad b \cdot \tau_{\rho,\alpha} = \frac{p_{\mathrm{na2gi},\rho}}{p_{\mathrm{na2na}}}.$$

a and *b* are positive constants which were chosen appropriately to generate GIs which are at mean shorter than the surrounding regions of native genes. The probabilities p_{x2y} correspond to transitions from state *x* to *y* (see Figure 1).

We extend the Markov chain X_1, X_2, ..., $X\ell$ driven by the state diagram given in Figure 1 to a HMM X_1, Y_1, X_2, Y_2 ..., $X\ell$, $Y\ell$ in the following way: For π = 1, 2..., ℓ, the random *emission* Y_π takes values in the sample space $\mathcal{F}_\pi$ defined above. For ρ = 1, 2,..., *r*, the *emission probabilities* are defined by means of Equation 1 as follows:

$P(Y_\pi = g \mid X_\pi = \text{NATIVE}) = P_0(g \mid \pi)$ and $P(Y_\pi = g \mid X_\pi = \text{PUTAL}_\rho) = P_\rho(g \mid \pi)$. (5)

As already explained, PHX genes have to be eliminated. Our test for putatively highly expressed genes classifies genes as phx or ¬phx. In order to integrate these predictions into the HMM, we interpret the outputs as a random sequence $H_1, H_2, \ldots, H\ell$ of *hints*. Please note that an emission is now a combination of a gene and a hint. Hints are interpreted the following way: For the native state we define

$$P(H_\pi = \text{phx} \mid X_\pi = \text{NATIVE}, Y_\pi = g_\pi) = \begin{cases} 1 & \text{if } t_{\text{rib}}(g_\pi) > \theta; \\ 0 & \text{otherwise.} \end{cases}$$

For $\rho = 1, 2, \ldots, r$, the *emission probability* given a pA state is defined by

$$P(H_\pi = \neg\text{phx} \mid X_\pi \in \{\text{PUTAL}_1, \text{PUTAL}_2, \ldots, \text{PUTAL}_r\}) = 1.$$

It is biological evidence, which led to the above definitions. The products of highly expressed genes are involved in complex interactions. Therefore, it is highly unlikely that these genes can be replaced by HGT. Please note that the algorithm has – due to our design – to consider each hint.

Determination of the codon-specific core and atypical genes

It might be that some pA genes originate from sources not characterized by our set of putative donors (see below). In order to identify these atypical genes, we determine the *codon-specific core* (*CSC*) of a genome, which consists of those genes having an unsuspicious codon usage. Having chosen a protein $\pi \in \mathcal{P}_0$ and the related gene $g \in \mathcal{G}_0$, we consider a random element G of the set $\mathcal{F}_\pi$ distributed according to $P_0(\cdot \mid \pi)$ (see Equation 1). For the following test, we identifed those amino acids a encoded by more than one codon and occurring at least 5 times ($n_a \geq 5$) in the protein. For each codon c which encodes amino acid a we introduce a random variable $\text{count}_c(G) = \#c$, which follows a binomial distribution characterized by the expected value $n_a q_{ac}^{(0)}$ and variance $n_a q_{ac}^{(0)}(1 - n_a q_{ac}^{(0)})$. The statistic

$$\phi_c(G) := \frac{\text{count}_c(G) - n_a q_{ac}^{(0)}}{\sqrt{n_a q_{ac}^{(0)}(1 - q_{ac}^{(0)})}}$$

is approximately distributed according to the standard normal distribution. For each $\delta \in (0, 1)$ there is exactly one $\theta_\delta > 0$ such that

$$P(|\phi_c(G)| \geq \theta_\delta) = 2\Phi(-\theta_\delta) = \frac{\delta}{\gamma},$$

where γ is the occurrence of those amino acids considered in this section. In analogy to [32], we name the gene g *δ-typical* ($\delta \in (0, 1)$), if for all codons c

$$|\phi_c(g)| < \theta_\delta.$$

This is why the probability of being not δ-typical is for a random gene G less than or equal to δ. Setting δ to $10/\ell$, where ℓ is the number of $\mathcal{G}_0$'s genes, turned out to be adequate. Only few genes (< 1%) were labelled as atypical (see Results). Therefore, the exact value of δ is uncritical. This observation confirms that our selection of codon usage tables covers the prevalences of putative donors to a great extent.

The algorithm for computing the CSC of genome $\mathcal{G}_0$ first removes all genes from $\mathcal{G}_0$ that are not δ-typical. Then the synonymous codon frequencies of the remaining genes $\mathcal{G}_{\text{typ}}$ are recomputed and the genes not δ-typical with respect to the new frequencies are removed from $\mathcal{G}_{typ}$. This is done as long as there are such genes in $\mathcal{G}_{typ}$. Our experiments showed that this algorithm converged for all completely sequenced genomes to a CSC $\mathcal{G}_{typ}$ containing at least 75% of all genes. The *atypical* genes are those not contained in the CSC $\mathcal{G}_{typ}$.

Predicting genomic islands

Using the Viterbi algorithm (see e.g. [33,34]), SIGI-HMM computes at first the Viterbi path (i.e. the most probable sequence of states). All genes labeled as atypical and all genes assigned to one of the states PUTAL_ρ ($\rho = 1, 2, \ldots, r$) are considered as belonging to GIs. Since it is reasonable to expect inside GIs genes with a codon usage similar to native ones, GIs separated by less than four native genes can optionally be merged. This merging distance can be set by the user.

Selecting putative donors

For each genome $\mathcal{G}_0$, an individual set of putative donors $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_r$ has to be selected. As these donors are reduced to their specific codon usage tables, we utilized the Codon Usage Database (CUTG) (Release 149.0, September 26, 2005) [35]. Those entries were extracted that

consisted of more than 6,400 codons. If a species was represented by more than one table, we took the entry sampling the largest number of codons. This pre-computing phase resulted in the selection of $z = 690$ codon usage tables. Then, a $z \times z$ dissimilarity matrix D was set up. For each pair i, j of species, we calculated the value $D_{ij} = 1/2 - \eta_{ij}$. In order to compute the discriminative error η_{ij}, we first considered the set of all "synthetic" genes each comprising 50 codons. Each of the 50 codons was independently selected according to the codon frequencies $q_c^{(k)}$. We then determined a probability distribution P_k for each species k on this set. These distributions were utilized to determine η_{ij} in analogy to Equation 4.

Hierarchical divisive clustering [36] was now applied to analyze the dissimilarity matrix D. As it was our aim to generate clusters representing taxonomically related species, we used the data basis of the taxonomy browser of the NCBI [37,38] for the following procedure. First, we eliminated all entries, which could not be related to a taxonomical class. Then, we generated for the initiation of the diversification process "class"-clusters consisting of species (i.e. synonymous codon frequency tables) belonging to the same taxonomical class. To test homogeneity of the clusters G, we computed for each entry i the average dissimilarity $\mathrm{diss}_G^{(av)}(i)$ (see [39]) according to

$$\mathrm{diss}_G^{(av)}(i) := \frac{1}{|G \setminus \{i\}|} \sum_{j \in G \setminus \{i\}} D_{ij}.$$

In order to initiate the split of a cluster G, the element $i \in G$ having the maximal $\mathrm{diss}_G^{(av)}(i)$ value was chosen. This i was the first element of a new cluster H. As long as the condition

$$\max_{k \in G} \left(\mathrm{diss}_G^{(av)}(k) - \mathrm{diss}_H^{(av)}(k) \right) \geq 0$$

was true, the element k generating the maximal $\mathrm{diss}_G^{(av)}(k) - \mathrm{diss}_H^{(av)}(k)$ value was transferred from G to H. Starting with the initial set of class-clusters described above, the split procedure was applied to that cluster G having maximal diameter

$$\mathrm{diam}\, G := \max_{i,j \in G} D_{ij}$$

as long as that maximal diameter was greater than or equal to a threshold d_1 (see [40]).

The procedure resulted in $\tilde{r} = 99$ clusters. In order to select a typical example for each cluster, the frequency table having the lowest dissimilarity value to the barycenter of the cluster was chosen. The resulting $\tilde{r}$ codon usage tables were regarded as representatives for putative sources of aliens genes.

To prevent false predictions, clusters with a composition too similar to the input genome $\mathcal{G}_0$ have to be eliminated. Therefore, the set of $\tilde{r}$ codon usage tables was preprocessed during the initialization phase for $\mathcal{G}_0$. Those elements were deleted, whose dissimilarity to the frequency table of $\mathcal{G}_0$ was less than a threshold d_2. This procedure resulted in a $\mathcal{G}_0$-specific set of r putative sources.

Testing performance and analyzing genomes

To assess accuracy, SIGI-HMM's predictions were compared with results published in [41]. In nearly all cases, the fraction of pA genes determined by SIGI-HMM was lower; compare results listed in Table 1. This might be due to the focusing of SIGI-HMM on the prediction of GIs. However, for the genome of *Borrelia burgdorferi* SIGI-HMM predicts a significantly higher fraction of pA genes. The organization of this genome is unusual, it consists of 20 mainly linear replicons and is subject to frequent genomic rearrangements [42]. During these reorganization events integration of alien DNA might take place making a larger fractions of pA genes for the *B. burgdorferi* genome plausible. In the following, we report in more detail findings deduced for genomic data sets of the following microbial genomes: *Vibrio cholerae*, *Bacillus subtilis*, *Escherichia coli* K-12, *Methanosarcina mazei*, *Thermus thermophilus* and *Propionibacterium acnes*. The genome of *V. cholerae* consists of two chromosomes with a pronounced asymmetry in the distribution of coding elements with respect to the replicons [43]. Most genes required for growth and virulence are located on chromosome I, whereas chromosome II contains a larger fraction of hypothetical genes.

Interestingly, SIGI-HMM predicted 4.6% pA genes for chromosome I and 21.1% pA genes for chromosome II. Two predicted genomic islands on chromosome I comprise a gene cluster for a toxin-coregulated pilus (VC0813 – VC0845) and fragments of a temperate filamentous phage (VC1455 – VC1457, VC1464, VC1477 – VC1481). Both clusters are closely associated with the pathogenicity of *V. cholerae* [44]. Many of the hypothetical genes encoded on chromosome II are located within a large integron island comprising gene products that might be

involved in drug resistance, DNA metabolism and virulence [43]. One of the predicted GIs on chromosome II, which consists of genes VCA0283 – VCA0507, overlaps to a great extent the integron described above. SIGI-HMM identified two additional GIs comprising genes VCA0198 – VCA0202 and VCA0790 – VCA0797, which contain homologs for putative transposases. As transposases are often encoded in genetically mobile IS-elements, these genes are likely candidates for alien genes. For both chromosomes, SIGI-HMM predicts similar distributions of putative donors. The largest fractions belong to the class of bacilli (51% or 61%), whereas the taxonomical class of *V. cholerae*, the γ-proteobacteria, accounts for 34% or 37% of all pA genes.

For *B. subtilis*, 10 integrated prophages have been reported (see [4,45,46], and [47]), whose identification is based either on experimental evidence or theoretical considerations. A profound analysis of chromosomal heterogeneities has been accomplished by Nicolas *et al.* [9], using a HMM on the nucleotide level. All genomic islands identified by Nicolas *et al.* were largely confirmed by SIGI-HMM. Both approaches detected nine of the putative prophages and several other islands assigned to functions in cell wall biosynthesis, competence and resistance. In contrast to Nicolas *et al.*, SIGI-HMM identified pA genes, which belong to the experimentally reported integrated prophage PBSX [47]. In summary, SIGI-HMM predicted for *B. subtilis* 9.5% of the genes as being pA, most of them originating from the class of bacilli (316 pA genes, 81%).

Based on a combination of parameters measuring computational complexity, Lawrence and Ochman [4] had estimated that about 18% of the *E. coli* K-12 genome have been imported *via* lateral gene transfer. In contrast, SIGI-HMM predicted 580 (13.4%) pA genes which were mostly organized in small clusters of less than ten genes. 521 pA genes (92%) seem to originate from γ-proteobacteria, the taxonomical class *E. coli* belongs to. The largest GIs included the cryptic prophages CP4-6 (262 – 297 kbp), DLP12 (557 – 584 kbp), e14 (1,196 – 1,221), Rac (1,410 – 1,433 kbp), Qin (1,631 – 1,651 kbp), CP4-44 (2,064 – 2,069 kbp), CPS-53 (2,465 – 2,475 kbp), Eut (2,556 – 2,563 kbp), CP4-57 (2,752 – 2,775 kbp), and the phage-like element KpLE2 (4,494 – 4,544 kbp) (for review see [48]). 44 IS-elements have been annotated within the genome of *E. coli* K-12, SIGI-HMM predicted 34 of them correctly.

T. thermophilus is an extreme thermophilic bacterium living as a halotolerant in an extreme ecological niche. Two *T. thermophilus* strains, namely HB27 [45] and HB8 [46], have been sequenced so far. SIGI-HMM predicted for both strains a small fraction of pA genes (HB27 1.0%; HB8 1.7%). The largest pA cluster consists of 6 genes in case of HB27 (TTC0277 – TTC0278, TTC0280 – TTC0283) and of 5 genes in case of HB8 (TTHA0644 – TTHA0648). The GIs share no sequence similarity and contain genes that are associated with functions in cell wall biosynthesis. Most pA genes seem to originate from the class of the δ-proteobacteria (HB27 5 genes; HB8 18 genes). In both genomes no donor was predicted for 12 pA genes, respectively.

It has been suggested that HGT plays an important role in the evolution of the mesophilic archaeon *M. mazei* [49]. The analysis of protein sequences *via* BLAST showed that 31% of the archeal sequences were more similar to bacterial than to archeal ones. SIGI-HMM predicted for *M. mazei* only 8.4% pA genes. Please note that the two analyses used different approaches for pA prediction and that SIGI-HMM focuses on the analysis of GIs only. These systematic differences may explain the findings. Interestingly and in agreement with the above analysis, only 21% of the pA genes seem to originate from the archeal domain. 27% of the pA genes were predicted to originate from the class of shingobacteria, 23% from chlamydia and 11% from clostridia. This finding is also in agreement with the postulated gene flux from mesophilic bacteria to mesophilic archaea [27].

P. acnes is a major inhabitant of the adult human skin, living in sebaceous follicles [50]. Usually the bacterium is harmless; however it is involved in acne vulgaris formation. The genome harbors genes whose products are involved in degrading host molecules and pore-forming factors. It also contains surface-associated and other immunogenic factors, which might be responsible for acne inanimation and other *P. acnes*-associated diseases. SIGI-HMM predicted 4.1% pA genes clustered in five larger GIs and several smaller islands of less than five genes. 47% (45 genes) of them are predicted to originate from the α-proteobacteria, but only 13% (12 pA genes) from the taxonomic class of *P. acnes*, the actinobacteria. Interestingly, four of the larger GIs and two of the smaller islands are flanked by tRNA-genes in direct or close vicinity. tRNAs are considered to be hot spots for recombination events that can result in horizontal gene transfer. SIGI-HMM found these anomalies although it does not interpret sequences besides protein coding genes. Of the larger GIs, the first (at position 28 – 34 kbp) contains genes without functional assignment, the second (874 – 880 kbp) harbors genes for several transport systems among others for iron(III)dicitrate (PPA0792 – PPA0794) and the third (921 – 941 kbp) for an ABC-type transport system (PPA0843 – PPA0845), putative conjugal transfer proteins (PPA0846 – PPA0848) and two putative transposases (PPA2354, PPA0858). The forth GI (1,390 – 1,407 kbp) contains a gene cluster for a putative non-ribosomal peptide synthetase (NRPS) (PPA1287 – PPA1290). NRPSs are involved in the biosynthesis of

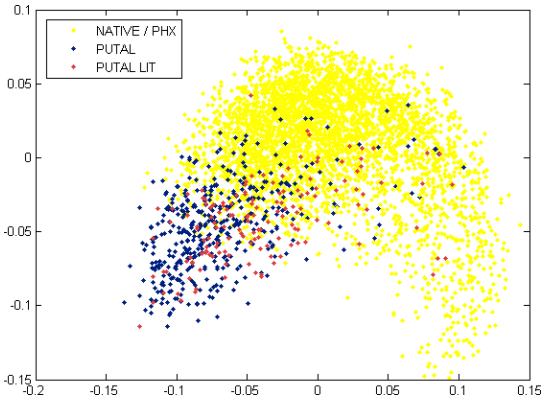

Figure 2
Kernel-based scatter plot visualization of SIGI-HMM predictions for *E. coli* K-12. Blue points (PUTAL) represent pA genes as predicted by SIGI-HMM, red points (PUTAL LIT) indicate predicted pA genes with additional evidence from the current literature as described in the text. Yellow points (NATIVE / PHX) refer to genes which are predicted to be native or highly expressed.

complex secondary metabolites. As many of the genes clustered in the fifth GI (1,707 – 1,731 kbp) are annotated as phage-associated proteins (PPA1593 – PPA1596, PPA1604 – PPA1605), the GI may be attributed to an integrated prophage.

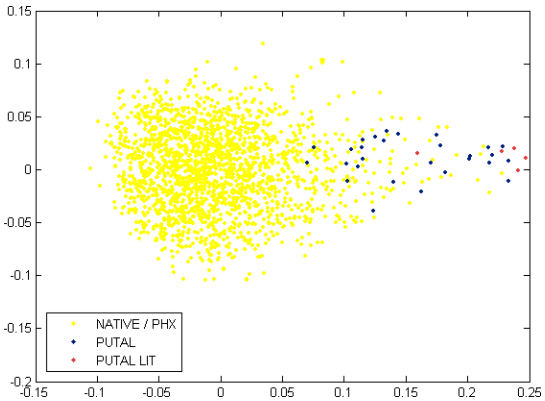

Figure 3
Kernel-based scatter plot visualization of SIGI-HMM predictions for *T. thermophilus*. Blue points (PUTAL) represent pA genes as predicted by SIGI-HMM, red points (PUTAL LIT) indicate predicted pA genes with additional evidence from the current literature as described in the text. Yellow points (NATIVE / PHX) refer to genes which are predicted to be native or highly expressed.

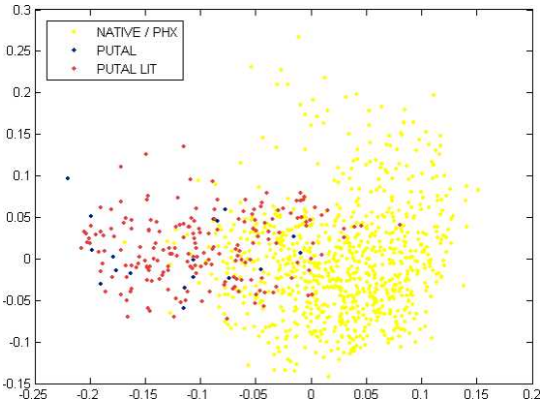

Figure 4
Kernel-based scatter plot visualization of SIGI-HMM predictions for *V. cholerae* (chromosome II). Blue points (PUTAL) represent pA genes as predicted by SIGI-HMM, red points (PUTAL LIT) indicate predicted pA genes with additional evidence from the current literature as described in the text. Yellow points (NATIVE / PHX) refer to genes which are predicted to be native or highly expressed.

For visualization of the HMM-based predictions we use scatter plot repesentations providing an overview of codon usage similarities between all genes of a genome. By means of a newly developed kernel for measuring similarity of codon usage tables [51], we perform a kernel principal component analysis (see e.g. [52]) to compute the resulting 2D coordinates of all genes. In that representation, nearby points indicate a similar codon usage of the corresponding genes. It is important to note that the kernel-based approach does not use any information about the location of genes on the genome. Instead, codon usage correlations between different amino acids are used to derive the two-dimensional representation. This approach is different from the concept of SIGI-HMM. Therefore, a clustering of SIGI-HMM predicted pA genes which becomes visible in the scatter plots (see Figure 2, 3, and 4) confirms the corresponding predictions.

Figure 2 is a plot of all genes of the E. coli K-12 genome. The general form resembles the "rabbit head" trimodal shape described earlier for the genome of B. subtilis [53]. Most genes belonging to integrated prophages are located in the lower left "ear". PHX genes are clustered in the lower right corner.

T. thermophilus is one of the genomes with lowest pA content. The plot depicted in Figure 3 represents the genome of *T. thermophilus* and has a quite specific shape. This finding indicates that the overall shape of the plot is massively modulated by the fraction of genes acquired *via* HGT. The

Table 2: Prophages and prophage-like elements integrated into the genome of *B. subtilis*. Column 1 lists the elements flanked by sequence repeats. Column 2 gives the location of the repeats. Column 3 and 4 list the number of pA and pN genes predicted for these GIs by SIGI-HMM. The two last columns indicate the offset of the GI from the sequence repeats. An offset of -1 means that the GI predicted by SIGI-HMM starts (ends) one gene after (before) the repeat. Positions are as in [9] and given in kbp.

Element	Repeats	# pA	# pN	Offset Begin	Offset End
P1	202 – 213	10	1	0	-1
P2	555 – 567	10	1	-1	0
P6	2050 – 2060	9	0	0	0
Skin	2654 – 2701	32	31	0	0
P7	2725 – 2735	7	6	-6	0

pA genes as predicted by SIGI-HMM are mainly located in a long tail with low point density on the right hand side of the plot.

As already mentioned, the genome of *V. cholerae* consists of two chromosomes. Most essential genes are located on chromosome I and codon usage of genes on chromosome II is rather inhomogeneous. Again, the overall shape of the plot, which represents chromosome II, reflects this situation (compare Figure 4) and shows a well-clustering fraction of pA genes located in the lower left corner of the plot. Please note that the positioning of pA genes predicted by SIGI-HMM only and those pA genes supported by additional evidence from the literature corresponds to a great extent in all plots.

Assessing the patchiness of GIs

Genomic islands are thought to be the result of constant genetic rearrangement events, which account for their observed mosaic structure. As these rearrangements could also take place at hot spots for the integration of alien DNA in the host genome, patches of genes having a codon usage similar to the host have to be expected inside GIs. This fact makes it difficult to determine the number of false negatives, even in annotated GIs. The number of false positives is difficult to deduce too, as it is hard to proof that a stretch of pA genes has not been acquired *via* HGT. In order to illustrate the problem and the patchiness of GIs, we compare in more detail some predictions with published findings.

Chromosome II of *V. cholerae* contains an integron island of size 125.3 kbp, which includes genes VCA0271 to VCA0491 [43]. Of these 214 genes, SIGI-HMM labels 188 as pA (87%), 1 as AT (atypical) and 25 as pN (putatively native). SIGI-HMM did subdivide the integron island into the following patches: VCA0271 – VCA0282 pN, VCA0283 – VCA0286 pA, VCA0287 – VCA0291 pN, VCA0292 – VCA0324 pA, VCA0325 – VCA0329 pN, VCA0330 – VCA0379 pA, VCA0380 – VCA0385 pN, VCA0386 – VCA0507 pA. From the remaining 611 genes on the chromosome, 42 were predicted as pA.

The chromosome of *Mesorhizobium loti* consists of 6.725 protein coding genes. It contains a 611 kbp DNA segment which is, as the authors put it, "a highly probable candidate of a symbiotic island" [3]. SIGI-HMM predicted 5.561 genes as pN, 1.161 (17%) as pA and 30 as AT. Of the symbiotic island, 145 genes were pN, 421 pA (72%) and 14 AT. The pA genes were clustered in 29 GIs ranging in size from 2 to 108 genes.

As already mentioned, ten integrated prophages or prophage-like elements were reported for the genome of *B. subtilis* [9]. Five of these elements are flanked by sequence repeats which we considered as the original integrations sites indicating the actual borders of the GIs. Table 2 summarizes composition and location of related GIs predicted by SIGI-HMM. Skin prophage and P7 have a mosaic structure and harbour ≈ 50% pN genes. In four of the five cases, the borders of the predicted GIs are in good agreement with the location of the repeats.

Discussion

Analysis of codon usage reliably allows to identify most HGT events

We have to stress that our approach entirely relies on the analysis of codon usage. SIGI-HMM does not interpret additional signals like direct repeats or disrupted tRNA sequences frequently flanking GIs. Therefore, the outcome of the HMM analysis are DNA regions showing atypical codon usage. This fact has two consequences: 1) SIGI-HMM is unable to identify GIs having an unsuspicious codon usage and 2) the rationale of naming these stretches GIs merely depends on the correlation with biological findings.

However, we have shown that DNA regions identified by SIGI-HMM as suspicious correspond to known cases of horizontally transferred elements like phages. Our approach of focusing on the analysis of codon usage is not a completely new one. There exist several methods to identify horizontally transferred genes. These approaches rely on the analysis of codon or amino acid sequences or the construction of phylogenetic trees. For a comparison see e.g. [14]. Each approach has individual drawbacks and it might be that each method identifies a specific class of genes acquired in a different time of genome evolution [13]. It was argued that codon usage is no reliable indicator for the study of HGT [54]. However, it was shown that related methods identify pA genes to a great extent [55]. The assumption that methods analyzing codon usage might overlook horizontally acquired genes could be valid for more ancient events. For these genes, the effect of amelioration [56] might have rendered codon usage

unsuspicious. Lawrence and Ochman estimated the age of imported genes [4]. Their conclusion was that most were relatively recent, i.e. acquired within the last few million years; see also [57]. This suggests that older imports have been purged presumably because the acquired genes did not improve fitness. If this argument is true, there is no need to search for larger amounts of ancient pA genes. Therefore, methods based on the analysis of codon usage should have the potential of identifying a great fraction of horizontally transferred genes. Low values of pA content can frequently be explained with biological findings. It was argued that species populating extreme ecological niches tend to have relative small genomes [58]. The size of the sequenced *T. thermophilus* genomes support this notion. If selective pressure minimizes genome size, it will also effect acquisition and conservation of foreign DNA. The low fraction of pA genes determined for both strains is in agreement with the above hypothesis.

The methods will fail at alien genes having a codon usage undistinguishable from the host's preferences. Among them might be ancient pA genes. Because of the amelioration process, ancient pA genes are harder to detect. These pA genes, surviving the selection process may actually constitute important and useful genes. In order to complete the set of identified HGT events and to reduce the number of false negatives, it will be necessary to use a completely different approach like the construction of phylogenetic trees.

If not processed correctly, highly expressed genes could be a source for false positive predictions. It is known that these genes show a distinct codon usage by preferring a species-specific set of major codons. In order to reduce the rate of false positive predictions, we use a filter which is based on a method [30] shown to be effective in predicting gene expressivity [31]. We have adjusted the parameters (see Equation 4) in such a way that the errors of the first and second kind are equally likely. Highly expressed genes belong to the core of a genome and it is unlikely that these genes are subject to HGT. Nevertheless, the user may disable this filter in order to study its influence on GI prediction.

Focusing on the prediction of GIs is biologically reasonable and reduces the risk of false predictions

Intrinsically, increasing the sensitivity of a test also increases the risk of predicting false positives. For the prediction of pA genes, the risk can however be minimized, if an algorithm focuses on the prediction of genomic islands as SIGI-HMM does. The pieces of DNA acquired *via* HGT typically have a considerable length. Examples are the symbiotic island of size 611 kbp described for the genome of *M. loti* or the integron island of size 125 kbp found on chromosome II of *V. cholerae* (see Results). Genes responsible for pathogenicity are also agglomerated in islands; see [2] and references therein. Therefore, a focusing on predicting GIs rather than all pA genes is an appropriate strategy to avoid false positives without missing relevant HGT events. Consequently, this argument was considered for the design of recently introduced algorithms [23,59]. However, the rate of false positive predictions will increase, if codon usage of a genome is inhomogeneous. To avoid this situation, it is important, to determine the CSC of a genome.

Codon usage is a reliable indicator to predict the origin of pA genes

For each completely sequenced genome, we have computed a variant of the CSC defined above; see [60]. It consisted of those genes having a homogeneous codon usage. The results obtained with the classification of genes from CSCs show that codon usage hints at the origin of genes. First tests indicate that prediction quality is high, as long as the CSC contains at least 70% of the genes. In addition, the results of performance tests (see [23]) carried out to demonstrate SIGI's ability of predicting the putative donor are also valid for SIGI-HMM.

Omelchenko *et al.* [61] used BLAST on the protein level to determine HGT events in the genome of *T. thermophilus* HB27. The protein sequences of many genes were similar to those of hyperthermophilic archaea. Taxonomical classification of donors for genes constituting GIs predicted by SIGI-HMM was rather inhomogeneous. The putative donors belonged to bacteria, archaea and eukaryota. It will be necessary to evaluate methods for pA prediction with a standardized test bed. Artificial genomes as introduced recently [62] may constitute the basis for such a validation, which may lead to a contest of methods for pA prediction.

Conclusion

An inhomogeneous HMM on gene level allows to identify GIs in microbial genomes and to predict the putative donor of horizontally transferred genes. The predictions are consistent with known findings and do not depend on the optimization of many parameters. Our implementation as a freely available tool written in Java allows an independent inspection of genomes in great detail. The genome-specific predictions can be used for further analysis or the comparison of several methods.

Authors' contributions

SW and RM specified the problem and the solution strategy. SW developed the HMM together with OK and KS and provided resources. OK and TB implemented the HMM. The donor selection was conceptualized by SW, RA and CD, and implemented by RA. WFF analyzed the genomes. PM decisively contributed to the methods meas-

uring similarity of codon usage tables. RM conducted the performance tests and contributed substantially to the manuscript which was prepared together with SW and KS. All authors read and approved the final manuscript.

Acknowledgements

The research was partially supported by the grant "ELAN – E-Learning Academic Network" of the Lower Saxony Ministry of Science, and by DFG Graduate Program "Identification in mathematical models: Synergy of stochastics and numerical methods".

References

1. Gogarten J, Doolittle W, Lawrence J: **Prokaryotic evolution in light of gene transfer.** *Mol Biol Evol* 2002, **19:**2226-2238.
2. Hacker J, Kaper JB: **Pathogenicity islands and the evolution of microbes.** *Annu Rev Microbiol* 2000, **54:**641-679.
3. Kaneko T, Nakamura Y, Sato S, Asamizu E, Kato T, Sasamoto S, Watanabe A, Idesawa K, Ishikawa A, Kawashima K, Kimura T, Kishida Y, Kiyokawa C, Kohara M, Matsumoto M, Matsuno A, Mochizuki Y, Nakayama S, Nakazaki N, Shimpo S, Sugimoto M, Takeuchi C, Yamada M, Tabata S: **Complete genome structure of the nitrogen-fixing symbiotic bacterium *Mesorhizobium loti*.** *DNA Res* 2000, **7:**381-406.
4. Lawrence JG, Ochman H: **Molecular archaeology of the *Echerichia coli* genome.** *Proc Nat Acad Sci USA* 1998, **95:**9413-9417.
5. Hooper SD, Berg OG: **Detection of genes with atypical nucleotide sequence in microbial genomes.** *J Mol Evol* 2002, **54:**365-375.
6. Mrázek J, Karlin S: **Detecting alien genes in bacterial genomes.** *Ann NY Acad Sci* 1999, **870:**314-329.
7. Garcia-Vallvé S, Romeu A, Palau J: **Horizontal gene transfer in bacterial and archaeal complete genomes.** *Genome Res* 2000, **10:**1719-1725.
8. Karlin S: **Detecting anomalous gene clusters and pathogenicity islands in diverse bacterial genomes.** *Trends Microbiol* 2001, **9(7):**335-343. Jul
9. Nicola P, Bize L, Muri F, Hoebeke M, Rodolhe F, Ehrlic SD, Prum B, Bessièrs P: **Mining *Bacillus subtilis* chromosome heterogeneities using hidden Markov models.** *Nucleic Acids Res* 2002, **30:**1418-1426.
10. Nesbø CL, L'Haridon S, Stetter KO, Doolittle WF: **Phylogenetic analysis of two "archaeal" genes in *Thermotoga maritima* reveal multiple transfers between archaea and bacteria.** *Mol Biol Evol* 2001, **18:**362-375.
11. Sandberg R, Winberg G, Bräden C, Kaske A, Ernberg I, Cöster J: **Capturing whole-genome characteristics in short sequences using a naive Bayesian classifier.** *Genome Res* 2001, **11:**1404-1409.
12. Dufraigne C, Fertil B, Lespinats S, Giron A, Deschavanne P: **Detection and characterization of horizontal transfers in prokaryotes using genomic signature.** *Nucleic Acids Res* 2005, **33:**e6.
13. Ragan MA: **Detection of lateral gene transfer among microbial genomes.** *Curr Opin Genet Dev* 2001, **11:**620-626.
14. Ragan MA: **On surrogate methods for detecting lateral gene tranfer.** *FEMS Microbiol Lett* 2001, **201:**187-191.
15. Grantham R, Gautier C, Gouy M, Mercier R, Pave A: **Codon catalog usage and the genome hypothesis.** *Nucleic Acids Res* 1980, **8:**R49-R62.
16. Burge C: **Identification of genes in a human genome DNA.** In *PhD thesis* Stanford University; 1997.
17. Burge C, Karlin S: **Prediction of complete gene structures in human genomic DNA.** *J Mol Biol* 1997, **268:**78-94.
18. Krogh A: **Two methods for improving performance of an HMM and their application for gene finding.** *Proc Int Conf Intell Syst Mol Biol* 1997, **5:**179-186.
19. Krogh A: **Using data base matches with HMMGene for automated gene detection in *Drosophila*.** *Genome Res* 2000, **10:**523-528.
20. Yeh R, Lim L, Burge C: **Computational inference of homologous gene structures in the human genome.** *Genome Res* 2001, **11:**803-816.
21. Stanke M, Waack S: **Gene prediction with a hidden Markov model and new intron submodel.** *Bioinformatics* 2003, **19:**ii215-ii225.
22. Stanke M, Schöffman O, Dahms S, Morgenstern B, Waack S: **Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources.** *BMC Bioinformatics* 2006, **7:**62.
23. Merkl R: **SIGI: score-based identification of genomic islands.** *BMC Bioinformatics* 2004, **5:**22.
24. Collins N, Liebenberg J, de Villiers E, Brayton K, Louw E, Pretorius A, Faber F, van Heerden H, Josemans A, van Kleef M, Steyn H, van Strijp M, Zweygarth E, Jongejan F, Maillard J, Berthier D, Botha M, Joubert F, Corton C, Thomson N, Allsopp M, Allsopp B: **The genome of the heartwater agent *Ehrlichia ruminantium* contains multiple tandem repeats of actively variable copy number.** *Proc Natl Acad Sci USA* 2005, **102:**838-843.
25. Veith B, Herzberg C, Steckel S, Feesche J, Maurer K, Ehrenreich P, Baumer S, Henne A, Liesegang H, Merkl R, Ehrenreich A, Gottschalk G: **The complete genome sequence of *Bacillus licheniformis* DSM13, an organism with great industrial potential.** *J Mol Microbiol Biotechnol* 2004, **7:**204-211.
26. Merkl R: **A comparative categorization of protein function encoded in bacterial or archeal genomic islands.** *J Mol Evol* 2006, **62:**1-14.
27. Wiezer A, Merkl R: **A comparative categorization of gene flux in diverse microbial species.** *Genomics* 2005, **86:**462-475.
28. **Colombo homepage** [http://www.tcs.informatik.uni-goettingen.de/colombo]
29. Rutherford K, Parkhill J, Crook J, Horsnell T, Rice P, Rajandream MA, Barrell B: **Artemis: sequence visualisation and annotation.** *Bioinformatics* 2000, **16:**944-945.
30. Merkl R: **A survey of codon and amino acid frequency bias in microbial genomes focusing on translational efficiency.** *J Mol Evol* 2003, **57:**453-466.
31. Supek F, Vlahovicek K: **Comparison of codon usage measures and their applicability in prediction of microbial gene expressivity.** *BMC Bioinformatics* 2005, **6:**182.
32. Welsh D: *Codes and Cryptography* New York: Oxford University Press; 1987.
33. Durbin R, Eddy S, Krogh A, Mitchinson G: *Biological Sequence Analysis* Cambridge: Cambridge University Press; 1998.
34. Merkl R, Waack S: *Bioinformatik interaktiv – Algorithmen und Praxis* Weinheim: Wiley-VCH; 2003.
35. Nakamura Y, Gojobori T, Ikemura T: **Codon usage tabulated from the international DNA sequences databases and predictions.** *Nucleic Acids Res* 1999, **27:**292.
36. Hastie T, Tibshirani R, Friedman J: *The Elements of Statistical Learning* New York, Berlin, Heidelberg: Springer; 2001.
37. Wheeler D, Chappey C, Lash A, Leipe DD, Madden T, Schuler G, Tatusova T, Rapp B: **Database resources of the National Center for Biotechnology Information.** *Nucleic Acids Res* 2000, **28:**10-14.
38. Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Rapp BA, Wheeler DL: **GenBank.** *Nucleic Acids Res* 2000, **28:**15-18.
39. MacNaughton-Smith P, Williams W, Dale M, Mockett L: **Dissimilarity analysis: a new technic of hierarchical subdivision.** *Nature* 1964, **202:**1034-1035.
40. Kaufman L, Rousseeuw P: *Finding Groups in Data* New York: Wiley; 1990.
41. Ochman H, Lawrence JG, Groisman EA: **Lateral gene transfer and the nature of bacterial innovation.** *Nature* 2000, **405:**299-304.
42. Chaconas G: **Hairpin telomeres and genome plasticity in *Borrelia*: all mixed up in the end.** *Mol Microbiol* 2005, **58:**625-635.
43. Heidelberg JF, Eisen JA, Nelson WC, Clayton RA, Gwinn ML, Dodson RJ, Haft DH, Hickey EK, Peterson JD, Umayam L, Gill SR, Nelson KE, Read TD, Tettelin H, Richardson D, Ermolaeva MD, Vamathevan J, Bass S, Qin H, Dragoi I, Sellers P, McDonald L, Utterback T, Fleishmann RD, Nierman WC, White O, Salzberg SL, Smith HO, Colwell RR, Mekalanos JJ, Venter JC, Fraser CM: **DNA sequence of both chromosomes of the cholera pathogen *Vibrio cholerae*.** *Nature* 2000, **406:**477-483.
44. Waldor M, Mekalanos J: **Lysogenic conversion by a filamentous phage encoding cholera toxin.** *Science* 1996, **272:**1910-1914.
45. Kunst F, Ogasawara N, Moszer I, Albertini A, Alloni G, Azevedo V, Bertero M, Bessieres P, Bolotin A, Borchert S, Borriss R, Boursier L, Brans A, Braun M, Brignell S, S B, Brouillet S, Bruschi C, Caldwell B, Capuano V, Carter N, Choi S, Codani J, Connerton I, Danchin A, *et*

al.: **The complete genome sequence of the gram-positive bacterium *Bacillus subtilis*.** *Nature* 1997, **390:**249-256.
46. Takemaru K, Mizuno M, Sato T, Takeuchi M, Kobayashi Y: **Complete nucleotide sequence of a skin element excised by DNA rearrangement during sporulation in *Bacillus subtilis*.** *Microbiology* 1995, **141:**323-327.
47. Wood HE, Dawson MT, Devine K, McConnell D: **Characterization of PBSX, a defective prophage of *Bacillus subtilis*.** *J Bacteriol* 1990, **172:**2667-2674.
48. Casjens S: **Prophages and bacterial genomics: what have we learned so far?** *Mol Microbiol* 2003, **49:**277-300.
49. Deppenmeier U, Johann A, Hartsch T, Merkl R, Schmitz R, Martinez-Arias R, Henne A, Wiezer A, Bäumer S, Jacobi C, Brüggemann H, Lienard T, Christmann A, Bömecke M, Steckel S, Bhattacharyya A, Lykidis A, Overbeck R, Klenk HP, Gunsalus RP, Fritz HJ, Gottschalk G: **The genome of *Methanosarcina mazei*: evidence for lateral gene transfer between archaea and bacteria.** *J Mol Microbiol Biotechnol* 2002, **4:**453-461.
50. Brüggemann H, Henne A, Hoster F, Liesegang H, Wiezer A, Strittmatter A, Hujer S, Dürre P, Gottschalk G: **The complete genome sequence of *Propionibacterium acnes*, a commensal of human skin.** *Science* 2004, **305:**671-673.
51. Meinicke P, Brodag T, Fricke WF, Waack S: **Kernel-based visualization of codon usage data.** . Submitted
52. Schölkopf B, Smola AJ, Müller KR: **Nonlinear component analysis as a kernel eigenvalue problem.** *Neural Computation* 1998, **10:**1299-1319.
53. Moszer I, Rocha E, Danchin A: **Codon usage and lateral gene transfer in *Bacillus subtilis*.** *Curr Opin Microbiol* 1999, **2:**524-8.
54. Wang B: **Limitations of compositional approach to identifying horizontally transferred genes.** *J Mol Evol* 2001, **53:**244-250.
55. Daubin V, Perrière G: **G+C3 structuring along the genome: a common feature in prokaryotes.** *Mol Biol Evol* 2003, **20:**471-483.
56. Lawrence JG, Ochman H: **Amelioration of bacterial genomes: rates of change and exchange.** *J Mol Evol* 1997, **44:**383-397.
57. de la Cruz F, Davies J: **Horizontal gene transfer and the origin of species: lessons from bacteria.** *Trends Microbiol* 2000, **8:**128-133.
58. Bentley S, Parkhill J: **Comparative genomic structure of prokaryotes.** *Annu Rev Genet* 2004, **38:**771-792.
59. Nakamura Y, Itoh T, Matsuda H, Gojobori T: **Biased biological functions of horizontally transferred genes in prokaryotic genomes.** *Nat Genet* 2004, **36:**760-766.
60. Waack S, Brodag T, Surovcik K, Merkl R: **Assessing homogeneity and species-specifity of codon usage in prokaryotic genomes.** . submitted
61. Omelchenko M, Wolf Y, Gaidamakova E, Matrosova V, Vasilenko A, Zhai M, Daly M, Koonin E, Makarova K: **Comparative genomics of *Thermus thermophilus* and *Deinococcus radiodurans*: divergent routes of adaptation to thermophily and radiation resistance.** *BMC Evol Biol* 2005, **5:**.
62. Azad R, Lawrence J: **Use of artificial genomes in assessing methods for atypical gene detection.** *PLoS Comput Biol* 2005, **1:**e56.

Appendix B

Article about Scipio

"Scipio: Using protein sequences to determine the precise exon/intron structures of genes and their orthologs in closely related species"

published in BMC Bioinformatics (2008), **9**, 278.

Scipio: Using protein sequences to determine the precise exon/intron structures of genes and their orthologs in closely related species

Oliver Keller[1], Florian Odronitz[2], Mario Stanke[3], Martin Kollmar*[2] and Stephan Waack*[1]

Address: [1]Universität Göttingen, Institut für Informatik, Lotzestr. 16-18, 37083 Göttingen, Germany, [2]Max-Planck-Institut für Biophysikalische Chemie, Abteilung NMR-basierte Strukturbiologie, Am Fassberg 11, 37077 Göttingen, Germany and [3]Universität Göttingen, Institut für Mikrobiologie und Genetik, Abteilung Bioinformatik, Goldschmidtstr. 1, 37077 Göttingen, Germany

Email: Oliver Keller - keller@cs.uni-goettingen.de; Florian Odronitz - flod@nmr.mpibpc.mpg.de; Mario Stanke - mstanke@gwdg.de; Martin Kollmar* - mako@nmr.mpibpc.mpg.de; Stephan Waack* - waack@cs.uni-goettingen.de

* Corresponding authors

Published: 13 June 2008

BMC Bioinformatics 2008, **9**:278 doi:10.1186/1471-2105-9-278

Received: 8 February 2008
Accepted: 13 June 2008

This article is available from: http://www.biomedcentral.com/1471-2105/9/278

Abstract

Background: For many types of analyses, data about gene structure and locations of non-coding regions of genes are required. Although a vast amount of genomic sequence data is available, precise annotation of genes is lacking behind. Finding the corresponding gene of a given protein sequence by means of conventional tools is error prone, and cannot be completed without manual inspection, which is time consuming and requires considerable experience.

Results: Scipio is a tool based on the alignment program BLAT to determine the precise gene structure given a protein sequence and a genome sequence. It identifies intron-exon borders and splice sites and is able to cope with sequencing errors and genes spanning several contigs in genomes that have not yet been assembled to supercontigs or chromosomes. Instead of producing a set of hits with varying confidence, Scipio gives the user a coherent summary of locations on the genome that code for the query protein. The output contains information about discrepancies that may result from sequencing errors. Scipio has also successfully been used to find homologous genes in closely related species. Scipio was tested with 979 protein queries against 16 arthropod genomes (intra species search). For cross-species annotation, Scipio was used to annotate 40 genes from *Homo sapiens* in the primates *Pongo pygmaeus abelii* and *Callithrix jacchus*. The prediction quality of Scipio was tested in a comparative study against that of BLAT and the well established program Exonerate.

Conclusion: Scipio is able to precisely map a protein query onto a genome. Even in cases when there are many sequencing errors, or when incomplete genome assemblies lead to hits that stretch across multiple target sequences, it very often provides the user with the correct determination of intron-exon borders and splice sites, showing an improved prediction accuracy compared to BLAT and Exonerate. Apart from being able to find genes in the genome that encode the query protein, Scipio can also be used to annotate genes in closely related species.

Background

In the post-genome era, sequence data is the entry point for many studies. Often, it is essential to obtain the correct genomic DNA sequences of eukaryotic genes because of the information contained in non-coding regions. For example, the intron regions contain important sites for the regulation of gene transcription, like enhancers, repressors, and silencers [1]. Transcription initiator sequences are located upstream from the target gene [2]. The determination of the exon/intron structures of genes is also important in comparative genomic analyses like the identification of ancient exons [3].

Today, over 340 eukaryotic genome sequencing projects have resulted in genome assemblies [4]. For many of these project, genomic sequences of genes are only available by ab-initio gene predictions, if at all. However, it has been shown that automatically annotated sequences are often wrong because of sequencing and assembly errors, and mispredictions of exons and introns [5]. Correct protein sequences have in many cases been derived from manual annotation of the genes of interest or from full-length cDNAs. But experimentally obtained cDNA sequences often do not completely correspond to annotated genes, for example because previously undescribed alternative splice forms have been isolated. In many cases, it might also be interesting to look at the genes of evolutionary closely related species. If these species have not been annotated yet, it is, however, very time-consuming to identify and manually annotate the corresponding homologous genes.

Currently, a few programs are available for the retrieval of non-coding sequence. The Java application Retrieval of Regulative Regions (RRE) parses annotation and homology data from NCBI [6]. RRE requires local installation and local copies of the desired genome and annotation files. The web application of RRE only hosts a small number of eukaryotic genomes and annotation data only from NCBI. Recently, the non-coding sequences retrieval system (NCSRS) has been published [7] that has 16 genomes and annotation data from both NCBI and Ensembl [8]. In summary, both tools rely on annotation files provided by NCBI and Ensembl, with all possible errors, for only a few organisms. In addition, Ensembl and the UCSC browser [9] allow to search for genes and to recover any part of the gene of interest. When searching with descriptive terms, accession numbers, or other search terms the output is mainly based on results of gene prediction programs, often supported by evidence from cDNA or manual curation. Both web-interfaces also allow searching the genomes with any protein query with either BLAST or BLAT. However, the quality of the resulting gene structure is limited by these programs. There are also further species-specific genome pages that offer retrieving the genomic DNA corresponding to search terms. But there is no service offering the retrieval of the gene structures corresponding to protein queries of almost all sequenced and assembled eukaryotic genomes.

If transcript or protein sequences are available, the task of automatic identification of gene-related sequences can be accomplished by spliced alignment tools much more accurately and faster than, for example, by ab initio gene prediction. Splign [10] is a program that aligns a cDNA query determining the exact location of splice sites. GeneWise [11] uses protein queries. The most notable tools that can do both are Exonerate [12], partly based on the algorithm used in GeneWise, and BLAT [13]. The suitable variants of BLAST [14] (TBLASTN etc.) are probably still the most widely used tools for the task, though they are more useful for detecting weak homology and yield separate contiguous hits rather than a single complete spliced alignment. All approaches have in common a tradeoff between high speed that can only be reached using heuristics, and optimal accuracy that can only be achieved by exact algorithms that are prohibitively slow for many applications. Exonerate is very flexible as it implements a variety of models to choose from in a particular setting. BLAT is specialized on queries with high sequence similarity, and is considered the fastest tool for this task.

An accurate protein spliced alignment tool is particularly useful when a cDNA sequencing project precedes a genome sequencing project. In those cases, protein sequences can be constructed using the cDNA but the genomic location of the exons and introns is yet unknown. If the genome assemby is fragmented and genes are split onto different contigs, then any gene-finding or alignment method that considers each contig separately makes insufficient use of the available protein sequence information. Here, a spliced alignment should take contig-spanning introns and exons into account for maximal accuracy at the contig boundaries. Another application scenario for accurate protein spliced alignment is the problem of annotating a new assembly of an already annotated genome. Frequently, a complete reannotation is time-consuming as it often requires different groups to run different gene finders and integrating all available evidence. Simply mapping the previously known protein sequences to the new genome is a fast and easy alternative, at least for those genes that can be mapped with 100% identity.

Some sequencing approaches sequence only filtered genomic regions that are enriched for genes like the methyl-filtrated sequencing or high-cot analysis used for many plants (e.g. maize and tobacco). In these cases the assembly will remain fragmented and many genes are

split up onto different contigs, requiring that a spliced alignment tool takes this fragmentation into account.

Here, we present Scipio, a tool for the retrieval of the genome sequence corresponding to a protein query. It uses BLAT to perform a spliced alignment that results in a list of candidate hits, which are then refined, filtered and assembled to produce a prediction as accurate as possible. No annotation data is required, and genes are identified correctly even if they span several genome contigs, and contain mismatches and frameshifts. With these capabilities, Scipio is not only able to correctly identify the gene in the genome corresponding to the protein query but also to identify homologous genes in the genomes of closely related organisms. Moreover, it provides a comprehensive output, both machine- and human-readable, already containing the genomic sequences searched for.

Implementation

In most instances, the task of determining the gene structure of a query protein within a DNA target sequence is a special case of the search for a spliced alignment. Since several tools performing this task have already been available for a long time, writing another one would mean reinventing the wheel. Our choice was to depend on a BLAT search.

However, in the example of BLAT, when performing a search for the protein in the translated DNA, the output does not coincide with the exon structure of a single gene. Usually, multiple hits are found for each query, varying in accuracy, and exon boundaries are given only on amino acid level, missing those codons that are split by introns. Hence, manual processing was still needed until now in the majority of cases to determine the exact location of the query. In cases where the genomic sequence is in an early stage of the assembly process, several parts of one particular gene are often found on different target sequences (contigs), making this task very tedious and time consuming.

The Scipio script

We designed the perl script Scipio to automate this process and output the results in both human- and machine-readable output formats [see Additional file 1]. The summary of the process is depicted in the diagram in Fig. 1. We chose to run BLAT to provide us with the spliced alignments because it is specialized for the case of high sequence identity, which is obviously the case when locating genes of the same species (where mismatches are mainly due to sequencing errors), but it turned out to be very applicable also for the case of closely related species.

Stage one: hit refinement

After running BLAT, Scipio processes the query protein and target DNA sequences, and the file containing the BLAT hits. In the first stage, each hit is then "refined" by a number of steps. A BLAT hit is a collection of consecutive matchings of the protein sequence aligned to the translated DNA. We do not want to include hits with low quality. Therefore, everything with an accuracy below a given threshold is discarded at this stage. The refinement then consists of the following steps:

• Unaligned parts of the target sequence between the matchings that form a BLAT hit are analysed. A hit is only likely to be considered if they consist of a longer piece of DNA corresponding to at most one residue of the query, so they can be regarded as introns. Scipio tries to determine the exact location of the splice site by looking for a known splice site pattern (see below). This way, codons that are split by an intron, and are only joined after splicing, can be revealed. In cases where all residues are aligned by BLAT but a splice site pattern is missing, Scipio tries to improve the prediction by shifting the splice sites in single nucleotide steps. If an exact location can not be found, a heuristic is used to determine a trade-off between the number of additional mismatches and the presence of the splice site pattern.

• In addition, two more types of unaligned target sequence are distinguished: First, actual gaps with significant parts of the query sequence unaligned (mostly due to low coverage of sequencing resulting in gaps between contigs represented by contiguous N's in supercontigs/chromosomes). Second, short gaps resulting from sequencing and assembly errors leading to additional or missing bases or codons, with or without a frameshift. Additional DNA in this case is not interpreted as intron an but as a sequence shift of the query against the target.

• Scipio tries to locate very short exons where the BLAT hit misses parts of the query sequence. This is done by simple pattern matching. Thus only pieces with full identity are added. Terminal exons are added only when an intact splice site is found.

The filtering during the first stage ensures that nothing will be shown that cannot be regarded as a good match. If no hit is left after filtering, Scipio simply considers the gene non-existent in the target sequence, and no further processing is done.

Stage two: hit filtering and assembly

All BLAT hits that survive the first stage are subsequently filtered in the second stage to determine those that form the gene corresponding to the protein query. If only complete chromosomes were considered, one could expect a

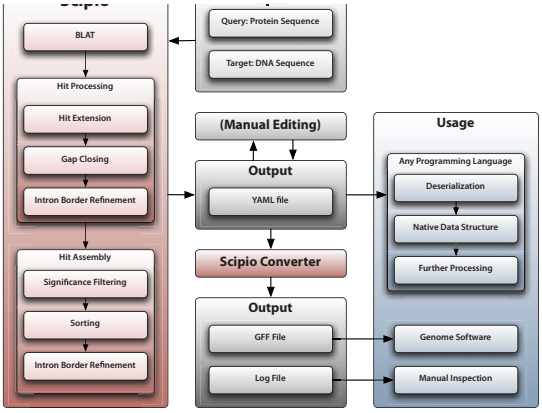

Figure 1
The Scipio workflow. This diagram depicts the data flow of a Scipio run. Scipio needs two FASTA files as input, one containing the protein query and one containing the genome sequence. Scipio starts BLAT and processes the BLAT results in a series of steps, successively refining and assembling the hits. Scipio's output is a YAML file, which can further be converted into a GFF file or a log file. YAML files can also be manually edited and read by a parser of which many exist for all modern programming languages. The resulting data structure can then be further processed.

single optimal BLAT hit coinciding with that gene; however, in cases without a complete assembly, partial hits on different targets need to be taken into account.

First, all hits are sorted by a score proportional to the number of matches, with a penalty subtracted for each mismatch. Second, all incompatible hits are discarded in the order just determined. Hits are incompatible if their queries overlap but their targets do not. An exception is the complete identity on DNA level at the ends of two contigs. This could result from an incomplete assembly, and the possibility of an overlap is taken into account. At the end of this step, we come out with a small number (usually just one) of non-overlapping hits forming the best gene candidate.

The final part of stage two is another refinement step: by assembling multiple hits, sequence parts may have been identified as parts of an intron that is split on different targets, the first half at the end of one target, the second at the beginning of the next. After the assembly Scipio uses the same method as in stage one to determine the exact splice site locations.

Output
The output contains target names, and location coordinates (genomic and protein) of all features: introns, exons, and gaps; exons can have sub-features: sequence shifts, mismatches, or undetermined positions. In addition, it contains the genomic DNA for all regions (including up- and downstream of the hit) and the translation of the coding sequence.

For the output format we defined two essential requirements: Human readability and machine readability. We chose YAML as it is a format that is complex enough to express our data structures and at the same time simple enough to be human readable and editable. YAML can be parsed easily and has numerous bindings to any modern programming language. The resulting native data structures can be used to further process the data generated by Scipio.

Conversion tools
Scipio provides two tools to convert YAML files:

• yaml2log: Converts YAML files into an easily readable log file with summary information about the results and clearly arranged sequence alignments.

• yaml2gff: Converts YAML files into GFF Format which can be read by a wide range of genome-related software packages.

Results and discussion
In many biological studies, protein sequences have been obtained by isolating mRNAs and sequencing the reverse-transcribed cDNAs. Also, large-scale cDNA sequencing projects resulted in thousands of supposed-to-be full-length cDNA sequences for some eukaryotes [15,16]. Protein sequences might have also been obtained by manual annotation. Sets of genomic DNA sequences of genes exist for some annotation projects. However, for many eukaryotic sequencing projects, the annotation process is lacking years behind the sequencing and assembly. In addition, experimentally obtained cDNA sequences often differ from annotated sequences because new alternatively spliced forms have been isolated. Therefore, for subsequent studies it might be useful or crucial to obtain the genomic DNA and the gene structure corresponding to the protein of interest.

Scipio has been designed for this task, and based on its differentiated processing capabilities it is able to cope with genes spanning multiple contigs as well as various kinds of sequencing and assembly errors. Scipio has been developed for the correct identification of eukaryotic genes. It can also be used for bacterial and archaeal genes although these genes are easily identified manually based on their simple single-exon structure. Depending on the similarity of the protein sequences, Scipio is also often

able to correctly identify homologous genes in closely related organisms.

We have implemented the following features:

A. If the query is distributed on several targets, the target contigs will be assembled guided by the alignments to the query. Untranslated regions from the last exon on a contig to the contig end and from the beginning of the next contig to the next exon are regarded as intronic. Scipio is also able to resolve overlapping contig ends if they consist of coding sequence, hence contributing to an improvement of the assembly.

B. The yaml2log script identifies cases from a list of alignment discrepancies and mismatches between query and target sequence that can result from sequencing/assembly errors (Figure 2). The simplest case is that amino acids differ (cases 1 to 3), or that they are missing in either the target or the query (cases 4 and 5). Sequencing/assembly errors may lead to additional or missing bases. These frameshifts are represented by an X in the translation corresponding to one or two nucleotides. The query sequence might have either been obtained from cDNA sources thus leading to a mismatch between query and translated target (cases 6, 8, 10, and 12), or the sequencing errors might have already been interpreted represented by an X in the query (cases 7, 9, 11, and 13). The target sequence might also contain in-frame stop codons (cases 14 to 17). These can be the result of sequencing errors or real stop codons as they appear in pseudogenes. In all these cases, the stop codon is shown as an asterisk ('*') in the translation.

C. Scipio interprets splice site patterns to determine intron locations. Exons borders are chosen so that the splice sites belong to one of the following classes, in decreasing priority: GT-AG, GC-AG, AT-AC, GA-AG, GG-AG. In cases where the translation of the adjacent intronic sequence was identical to the query, it was necessary to shift the intron location predicted by BLAT by several codons to determine the splice site location.

D. Scipio searches for stop codons at the end of genes. This helps evaluating the completeness of the query sequence.

E. Scipio tries to locate very short exons that are not recognized by BLAT. These short exons might either appear inbetween longer exons or at the ends of the gene. For example, very often genes start with an N-terminal methionine that is the only translated codon in the first exon. Scipio locates N-terminal methionines only if matching splice sites are found.

Insect genomes

To develop and test Scipio we used a test set of 16 arthropod genomes and a set of 979 proteins (Figure 3). The genome sequences (the newest collections of contigs as submitted to NCBI) differ in quality and completeness and are thus representative for straight-forward and for difficult identifications of the genes.

Drosophila melanogaster is an example of a perfect genome sequence with all reads assembled to chromosomes and almost all gaps closed. *Bombyx mori p50T* was used as example for a very preliminary assembly with many short contigs. The other genome sequences represent all stages in-between these extreme cases. For example, the genomes of *Drosophila persimilis* and *Drosophila sechellia* are quite complete which is visible from their number of contigs, but they have a low sequence coverage and/or contain many sequencing errors leading to high numbers of mismatches and frameshifts in the identified genes (Figure 3). In total, almost all query sequences have been identified correctly by Scipio (90.9 %), although many are spread on several target contigs (e.g. see *Aedes aegyptii* and *Culex pipiens*). 4.7 % of the genes have correctly been identified but the target DNA sequence contains sequencing or assembly errors. Another 1.7 % has not completely been found with the standard BLAT settings (BLAT tilesize of 5) because these genes contain very short exons. After changing the BLAT tilesize to 3 or 4 these genes have also completely been identified. Further 1.7 % of the genes could not be identified correctly, because the query sequence has been derived from manual annotation thus having incorporated EST data, data from other genome assemblies (e.g. newer data from the sequencing centers), or errors in the manual annotation process. E.g., the *Bombyx mori p50T* genome data is very incomplete but a lot of EST data is available. Thus, the query protein sequences have been built to a large part on these EST data. EST data has also been used to close gaps in the *Apis mellifera* and *Drosophila virilis* genomes. Errors resulting from this process are not due to problems in the implementation of Scipio. Two sequences (0.2 %) could not be identified correctly, because the genome sequences shows an large number of sequencing errors resulting in a succession of frame shifts in this particular region. The query protein sequences have correctly been identified based on EST data. The remaining 7 sequences (0.7 %) contain very long overlapping regions due to problems in the genome assemblies. Currently, Scipio handles overlapping hits by choosing the one with the higher overall score, in some cases discarding the one with fewer mismatches in the overlap region. The other cases in which Scipio did not resolve the complete gene structure are those, where a frame shift exists very close to an intron border. BLAT does not include the stretches past the frameshifts since they are

```
1: mismatch
gDNA        AAA ttt GGG
translation  K   F   G
             |   X   |
query        K   A   G

2: undetermined query
gDNA        AAA ttt GGG
translation  K   F   G
             |       |
query        K   X   G

3: undetermined target
gDNA        AAA nnn GGG
translation  K   X   G
             |       |
query        K   A   G

4: additional codon in target
gDNA        AAA ttt GGG
translation  K   F   G
             |       |
query        K   -   G

5: unmatched query
gDNA        AAA --- GGG
translation  K   -   G
             |       |
query        K   A   G

6: frameshift (+1) target only
gDNA        AAA t-- GGG
translation  K   X   G
             |       |
query        K   -   G

7: frameshift (+1) target/query
gDNA        AAA t-- GGG
translation  K   X   G
                     |
query        X   -   G

8: frameshift (+2) target only
gDNA        AAA tt- GGG
translation  K   X   G
             |       |
query        K   -   G

9: frameshift (+2) target/query
gDNA        AAA tt- GGG
translation  K   X   G
                     |
query        X   -   G

10: frameshift (-2) target only
gDNA        AAA t-- GGG
translation  K   X   G
             |       |
query        K   A   G

11: frameshift (-2) target/query
gDNA        AAA t-- GGG
translation  K   X   G
             |       |
query        K   X   G

12: frameshift (-1) target only
gDNA        AAA tt- GGG
translation  K   X   G
             |       |
query        K   A   G

13: frameshift (-1) target/query
gDNA        AAA tt- GGG
translation  K   X   G
             |       |
query        K   X   G

14: stopcodon target/query
gDNA        AAA tag GGG
translation  K   *   G
             |   |   |
query        K   *   G

15: stopcodon, target only
gDNA        AAA tag GGG
translation  K   *   G
             |   X   |
query        K   D   G

16: stopcodon, undetermined query
gDNA        AAA tag GGG
translation  K   *   G
             |       |
query        K   X   G

17: additional stopcodon
gDNA        AAA tag GGG
translation  K   *   G
             |       |
query        K   -   G
```

Figure 2
Types of discrepancies. This chart lists all types of discrepancies between the protein query and target translation/DNA that are known to Scipio. The identifiers as written into the log files are given.

smaller than the tile size used for searching. Scipio was not able to place the missing residues between the exons.

Cross species search

To test the ability of Scipio to correctly predict orthologous genes in closely related organisms we have annotated the myosins in the recently assembled primates *Pongo pygmaeus abelii* and *Callithrix jacchus* [17]. As a query, we used the 40 manually annotated myosins from *Homo sapiens* [5], which can be obtained from CyMoBase [18,19]. Although the genome assembly is not complete and most of the sequencing/assembly errors described above have been seen, Scipio correctly predicted and identified most orthologs of the human myosins in the two primates and located all parts of the genes if they were distributed on several target contigs, It also correctly identified two rare splice sites (AT-AC and GG-AG) that are specific for vertebrate sequences in two myosin classes. In the tails of the class-15 and class-35 myosins very small and divergent gaps had to be filled manually.

Figure 4A, shows the completeness compared to the sequence of the human ortholog and demonstrates the incompleteness of the genome sequences of *Pongo pygmaeus abelii* and *Callithrix jacchus*. Figure 4B and 4C show how gene completeness as determined by manual annotation depends on sequence identity. Exceptionally low values are largely due to gaps in the genome sequence (see Figure 4A), values exceeding 100% are due to stretches in homologous genes that were found by Scipio. Comparison between Figure 4B and 4C illustrates how a more thorough search can improve the recognition accuracy of 'difficult' genes (e.g. genes with very short exons) but at the same adds false positive exons to genes.

Comparison with existing tools

To assess the improvement on prediction quality that Scipio achieves, a comparative study was carried out with a selection of genomes varying in size and quality of assembly (Table 1). Running Scipio, standalone BLAT, and Exonerate, a number of myosin and kinesin queries were searched for in these genomic sequences.

Table 2 shows the accuracy on amino acid level. The improvement over BLAT is mainly due to the identification of split codons at intron boundaries. BLAT is only able to align unbroken codons, but will predict split codons as (false) matches whenever the splice site matches the query sequence. By the postprocessing, Scipio was able to reduce the number of residues missed by BLAT (including false positives) from 2.16 % to 1.67 %. Depending on the quality of the genomic sequences (some of which contained frameshifts, undetermined nucleotides, or even lacked larger parts), this is already the best that can be achieved theoretically; in the example of *Drosophila melanogaster* where complete chromosomes were available, 100 % of the residues could be matched.

Exonerate is able to find introns with exact boundaries. Hence, its accuracy is somewhat higher than BLAT's. However, in our setting, Exonerate had running times of about three to ten times longer than Scipio. Scipio adds less than ten percent to BLAT's running time.

It is important to note that manual postprocessing of BLAT's and Exonerate's results was used to gain comparable values: while BLAT and Exonerate come out with a collection of possibly overlapping hits with strongly varying accuracy, Scipio is able to filter and assemble non-overlapping partial hits on multiple targets together to a single prediction for each query sequence. This additional feature of Scipio had to be emulated by manually choosing a collection of best-scoring hits without large overlaps, and adding up the numbers of matches found, from the outputs of the two other programs.

In the case of a genome in a very early stage of assembly (here: the silkworm *Bombyx mori*), this assembling of hits increased the number of matching residues found to 92 % from 64 % which would be found taking only a single hit for each query. In the genome of *Macaca mulatta*, although target sequences were complete chromosomes, Exonerate still failed sometimes to return a single hit comprising the entire query sequence, and hence assembling of partial hits increased the number of residues found here, too. Manual inspection showed that Scipio produced false positives in rare cases when it rather had declared the gene not present in the target sequences. 248 residues (equaling 2 % of the missing residues, or 0.04 % of all) accounted for false exons.

The results on sequence level (Table 3) show strikingly how much the workload for manual postprocessing is reduced by Scipio, given that every sequence not matched completely will have to be looked at again, and alignments marked as completed (without mismatches, missing codons, sequence shifts or doubtful splice site patterns) will not.

While BLAT was unable to yield a complete prediction including the correct splice site locations in 95 % of the query sequences, Exonerate, by modeling introns, did better but would still fail to give a complete prediction in about half of the queries, even when partial hits were combined manually. The reason why assembling hits does not improve the number of completed queries is that in cases when a genomic sequence (e.g., a contig) ends in the middle of an intron, Exonerate will not use the intron model. Instead, at the cost of mismatches, the alignment

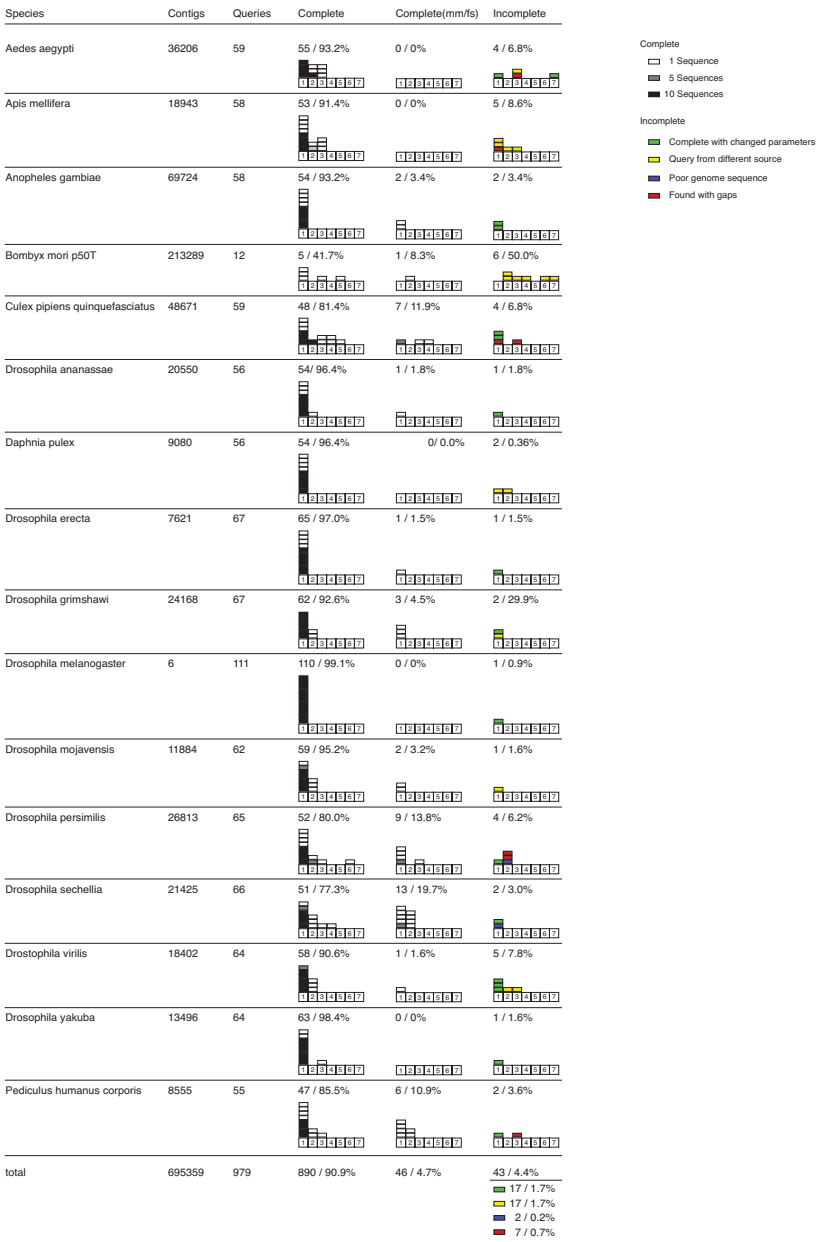

Figure 3
In-Species Performance. This chart shows Scipio's performance when searching in-species. The charts shows histograms depicting how many sequences where found on a particular number of contigs in the genome. Black rectangles represent ten, grey ones five and white ones single sequences. 'Complete' means the queries where found without discrepancies. 'Complete (mm/fs)' means that Scipio found the complete gene without gaps but with discrepancies like mismatches or framshifts. 'Incomplete' means that Scipio could not determine the complete gene structure with standard parameters.

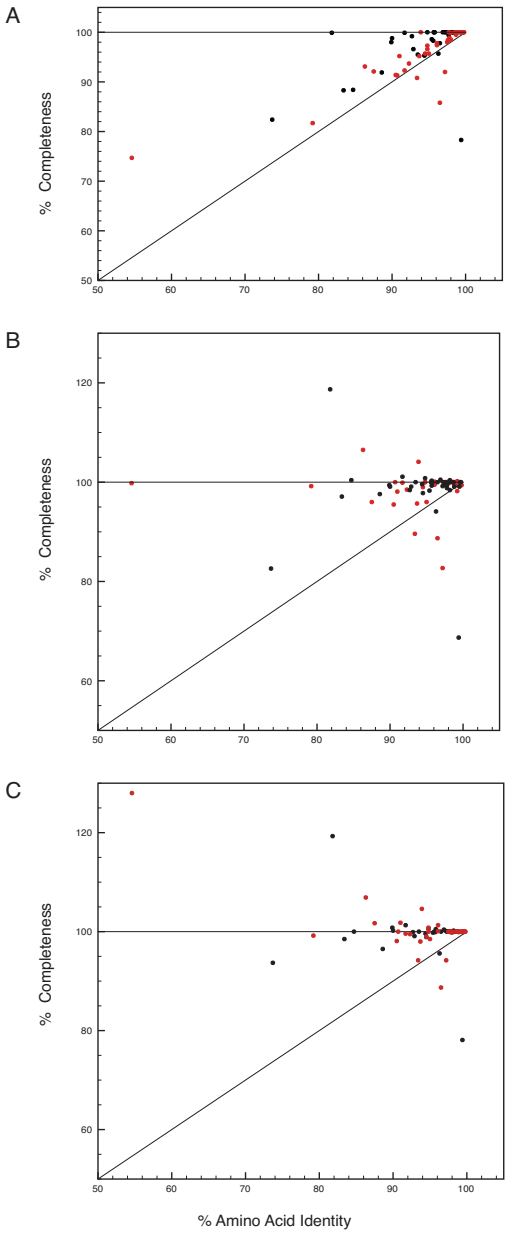

Figure 4
Cross-Species Performance. This chart shows Scipio's performance when searching cross-species. The charts show the dependency of the completeness of the gene reconstruction on the identity of the protein sequences. Red dots show searches with human sequences against the genome of *Pongo pygmaeus abelii*, black dots show searches with human sequences against the genome of *Callithrix jacchus*. A: Completeness compared to the query sequence. B: Completeness compared to the manually annotated sequence. A BLAT tile size of 7 was used. C: As in B, but with tile size 5.

Table 1: Sequences tested in comparative analysis.

Species	typ. target size (kbp)	# of target sequences	# of query sequences
Bombyx mori p50T	2.8	213 289	8
Drosophila simulans	9.4	31 198	37
Bombyx mori str. Dazao	14	66 482	32
Nasonia vitripennis (contigs)	27	26 605	35
Nasonia vitripennis (scaffolds)	1 827	6 181	35
Drosophila sechellia	70	21 425	38
Aedes aegypti	113	36 206	40
Daphnia pulex	825	9 080	42
Anopheles gambiae	6 969	69 724	37
Drosophila melanogaster	21 575	13	38
Aspergillus niger	2 409	143	15
Homo sapiens	153 287	24	40
Macaca mulatta	150 724	22	40

A list of 9 insect, one fungi, and two primate genomes, searched for kinesin and myosin genes to compare the performance of Scipio with that of BLAT and Exonerate. Genomic target sequences were taken from different stages of assembly, as can be seen from the different typical target sizes (*D. melanogaster*, *H. sapiens* and *M. mulatta* were given as sets of complete chromosomes; of the genome of *N. vitripennis*, two versions were compared; the genome of *A. gambiae* was given partly in chromosomes, partly in small contigs). The protein query sequences were taken from the same species as the genome.

Table 2: Percentage of residues left unmatched.

		BLAT	Exnrt.	BLAT	Exnrt.
Species	Scipio (automatic assembling)	with manual assembling		without assembling	
Bombyx mori p50T	15.03	15.41	17.03	43.61	43.64
Drosophila simulans	7.11	7.44	8.14	31.83	30.33
Bombyx mori str. Dazao	7.93	8.43	7.86	36.09	36.38
Nasonia vitripennis (contigs)	0.20	0.60	0.45	10.17	9.38
Nasonia vitripennis (scaffolds)	1.11	1.50	1.38	3.70	3.89
Drosophila sechellia	0.11	0.45	0.95	9.05	10.14
Aedes aegypti	0.73	1.01	0.21	10.43	10.22
Daphnia pulex	1.26	1.84	1.33	1.84	1.88
Anopheles gambiae	0.02	0.31	0.26	0.31	0.63
Drosophila melanogaster	0.00	0.32	0.05	0.32	0.05
Aspergillus niger	0.01	0.18	0.08	0.18	2.63
Homo sapiens	0.06	0.92	0.10	0.92	0.10
Macaca mulatta	2.29	3.09	2.31	3.09	7.82
total	1.67	2.16	1.87	8.24	8.71

The percentage of residues of the query sequences that the compared tools failed to recover from the target sequence. To gain comparable results, the hits proposed by BLAT and Exonerate were assembled together manually: a collection of best-scoring non-overlapping hits was chosen for each query. The last two columns show the results if only the best-scoring hit for each query was used.

is extended into the intron, yielding false boundaries and consequently, a false prediction.

In the case of *D. melanogaster*, where Scipio completely recovers all 38 query sequences, Exonerate misses five short exons; while these account only for 0.05 % of all residues, it reduces its rate of success on sequence level to 33 of 38, or 86.8 %.

These results show that Scipio, apart from providing a more detailed and well arranged output, can improve the prediction rate to 100 % by searching for short exons. In the case of fragmented genomes, the feature of hit assembly significantly improves the chances of retrieving the complete genomic sequence belonging to a protein query.

Future plans

The primary aim of the upcoming version of Scipio is to eliminate false positive predictions and to close more gaps still left in the prediction.

Table 3: Percentage of perfectly aligned queries.

		BLAT	Exnrt.	BLAT	Exnrt.
Species	Scipio (automatic assembling)	with manual assembling		without assembling	
Bombyx mori p50T	25.0	0.0	25.0	0.0	12.5
Drosophila simulans	37.8	2.7	21.6	2.7	16.2
Bombyx mori str. Dazao	15.6	3.1	12.5	0.0	9.4
Nasonia vitripennis (contigs)	71.4	11.4	57.1	11.4	54.3
Nasonia vitripennis (scaffolds)	68.6	11.4	65.7	11.4	65.7
Drosophila sechellia	63.2	7.9	55.3	2.6	55.3
Aedes aegypti	87.5	7.5	52.5	7.5	50.0
Daphnia pulex	88.1	0.0	76.2	0.0	76.2
Anopheles gambiae	94.6	8.1	73.0	8.1	73.0
Drosophila melanogaster	100.0	10.5	86.8	10.5	86.8
Aspergillus niger	86.7	6.7	73.3	6.7	73.3
Homo sapiens	62.5	0.0	50.0	0.0	50.0
Macaca mulatta	12.5	0.0	10.0	0.0	10.0
total	64.5	5.5	51.7	4.8	50.3

The number of query sequences that were predicted by the programs exactly at the correct location, with 100 % matching residues, without frameshifts or false positives. This figure reveals the amount of workload needed for manual postprocessing of the hits.

Eukaryotic genes contain far more information than is encoded in the sequence of one expressed protein. Most of this information is contained in the untranslated regions. Therefore, our future developments will focus on analyzing the untranslated regions to provide the user with additional gene-related information. Thus, Scipio will be developed to identify untranslated exons and, in addition, to determine mutually exclusive exons, and other alternatively spliced exons.

A web interface for Scipio is currently under development to address a wider audience and to make Scipio more user-friendly.

Conclusion

Scipio is a tool for the determination of gene structure and annotation of genes for a given protein sequence. Based on the widely used program BLAT, it performs exhaustive processing to ensure the best possible mapping of the protein onto the genome. By the ability of assembling partial hits ranging over multiple target sequences, Scipio goes beyond the scope of present spliced alignment tools and presents the user with a coherent set of matches that are often accurate to the level of single bases. Having a certain level of tolerance, Scipio can handle mismatches and frameshifts that often result from sequencing errors in genomes and cDNA. The same tolerance can be used to track down homologous genes in closely related species, allowing for cross-species annotation.

Availability and requirements

Project name: Scipio

Project home page: http://www.webscipio.org

Operating system: Platform independent

Programming language: Perl

Software requirements: Installation of BLAT and BioPerl.

Hardware requirements: BLAT may demand several times the genome size in RAM. If the RAM size is limiting, the most reasonable way is to split the genome and run Scipio against the split files.

License: Scipio may be obtained upon request and used under a Creative Commons License.

Any restrictions to use by non-academics: Using Scipio by non-academics requires permission.

Abbreviations

BLAT: BLAST like alignment tool; YAML: YAML ain't markup language.

Authors' contributions

OK, FO and MK set the requirements for the system, performed testing, and wrote the manuscript. MK, MS and SW initiated the project, OK wrote software. FO and MK carried out the analysis of the insect genomes and the cross-species searches, OK the comparative tests. All authors read and approved the final version of the manuscript.

Additional material

Additional file 1

Source code of the software, two Scipio output conversion tools, and a manual on the usage of the Scipio script.
Click here for file
[http://www.biomedcentral.com/content/supplementary/1471-2105-9-278-S1.zip]

Acknowledgements

MK has been funded by grants KO 2251/3-1 and KO 2251/6-1, and SW by grant WA 766/6-1 of the Deutsche Forschungsgemeinschaft.

The sequence data for *Pongo pygmaeus abelii* and *Callithrix jacchus* were produced by the Genome Sequencing Center at Washington University School of Medicine in St. Louis [17].

References

1. Fedorova L, Fedorov A: **Introns in gene evolution.** *Genetica* 2003, **118(2–3):**123-31.
2. Sandelin A, Carninci P, Lenhard B, Ponjavic J, Hayashizaki Y, Hume DA: **Mammalian RNA polymerase II core promoters: insights from genome-wide studies.** *Nature reviews* 2007, **8(6):**424-36.
3. Irimia M, Rukov J, Penny D, Roy S: **Functional and evolutionary analysis of alternatively spliced genes is consistent with an early eukaryotic origin of alternative splicing.** *BMC Evol Biol* 2007, **7:**188.
4. Odronitz F, Hellkamp M, Kollmar M: **diArk-a resource for eukaryotic genome research.** *BMC Genomics* 2007, **8:**103.
5. Odronitz F, Kollmar M: **Drawing the tree of eukaryotic life based on the analysis of 2269 manually annotated myosins from 328 species.** *Genome Biol* 2007, **8(9):**R196.
6. Lazzarato F, Franceschinis G, Botta M, Cordero F, Calogero RA: **RRE: a tool for the extraction of non-coding regions surrounding annotated genes from genomic datasets.** *Bioinformatics (Oxford, England)* 2004, **20(16):**2848-50.
7. Doh ST, Zhang Y, Temple MH, Cai L: **Non-coding sequence retrieval system for comparative genomic analysis of gene regulatory elements.** *BMC bioinformatics* 2007, **8:**94.
8. Flicek P, Aken BL, Beal K, Ballester B, Caccamo M, Chen Y, Clarke L, Coates G, Cunningham F, Cutts T, Down T, Dyer SC, Eyre T, Fitzgerald S, Fernandez-Banet J, Graf S, Haider S, Hammond M, Holland R, Howe KL, Howe K, Johnson N, Jenkinson A, Kahari A, Keefe D, Kokocinski F, Kulesha E, Lawson D, Longden I, Megy K, Meidl P, Overduin B, Parker A, Pritchard B, Prlic A, Rice S, Rios D, Schuster M, Sealy I, Slater G, Smedley D, Spudich G, Trevanion S, Vilella AJ, Vogel J, White S, Wood M, Birney E, Cox T, Curwen V, Durbin R, Fernandez-Suarez XM, Herrero J, Hubbard TJP, Kasprzyk A, Proctor G, Smith J, Ureta-Vidal A, Searle S: **Ensembl 2008.** *Nucleic Acids Res* 2008:D707-14.
9. Karolchik D, Kuhn RM, Baertsch R, Barber GP, Clawson H, Diekhans M, Giardine B, Harte RA, Hinrichs AS, Hsu F, Kober KM, Miller W, Pedersen JS, Pohl A, Raney BJ, Rhead B, Rosenbloom KR, Smith KE, Stanke M, Thakkapallayil A, Trumbower H, Wang T, Zweig AS, Haussler D, Kent WJ: **The UCSC Genome Browser Database: 2008 update.** *Nucleic Acids Res* 2008:D773-9.
10. Kapustin Y, Souvorov A, Tatusova T, Lipman D: **Splign: algorithms for computing spliced alignments with identification of paralogs.** *Biol Direct* 2008, **3(1):**20.
11. Birney E, Clamp M, Durbin R: **GeneWise and Genomewise.** *Genome Res* 2004, **14(5):**988-995.
12. Slater GSC, Birney E: **Automated generation of heuristics for biological sequence comparison.** *BMC Bioinformatics* 2005, **6:**31.
13. Kent WJ: **BLAT-the BLAST-like alignment tool.** *Genome research* 2002, **12(4):**656-64.
14. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ: **Basic local alignment search tool.** *J Mol Biol* 1990, **215(3):**403-410.
15. Gerhard DS, Wagner L, Feingold EA, Shenmen CM, Grouse LH, Schuler G, Klein SL, Old S, Rasooly R, Good P, Guyer M, Peck AM, Derge JG, Lipman D, Collins FS, Jang W, Sherry S, Feolo M, Misquitta L, Lee E, Rotmistrovsky K, Greenhut SF, Schaefer CF, Buetow K, Bonner TI, Haussler D, Kent J, Kiekhaus M, Furey T, Brent M, Prange C, Schreiber K, Shapiro N, Bhat NK, Hopkins RF, Hsie F, Driscoll T, Soares MB, Casavant TL, Scheetz TE, Brown-stein MJ, Usdin TB, Toshiyuki S, Carninci P, Piao Y, Dudekula DB, Ko MSH, Kawakami K, Suzuki Y, Sugano S, Gruber CE, Smith MR, Simmons B, Moore T, Waterman R, Johnson SL, Ruan Y, Wei CL, Mathavan S, Gunaratne PH, Wu J, Garcia AM, Hulyk SW, Fuh E, Yuan Y, Sneed A, Kowis C, Hodgson A, Muzny DM, McPherson J, Gibbs RA, Fahey J, Helton E, Ketteman M, Madan A, Rodrigues S, Sanchez A, Whiting M, Madari A, Young AC, Wetherby KD, Granite SJ, Kwong PN, Brinkley CP, Pearson RL, Bouffard GG, Blakesly RW, Green ED, Dickson MC, Rodriguez AC, Grimwood J, Schmutz J, Myers RM, Butterfield YSN, Griffith M, Griffith OL, Krzywinski MI, Liao N, Morin R, Palmquist D, Petrescu AS, Skalska U, Smailus DE, Stott JM, Schnerch A, Schein JE, Jones SJM, Holt RA, Baross A, Marra MA, Clifton S, Makowski KA, Bosak S, Malek J: **The status, quality, and expansion of the NIH full-length cDNA project: the Mammalian Gene Collection (MGC).** *Genome Res* 2004, **14(10B):**2121-2127.
16. Ota T, Suzuki Y, Nishikawa T, Otsuki T, Sugiyama T, Irie R, Wakamatsu A, Hayashi K, Sato H, Nagai K, Kimura K, Makita H, Sekine M, Obayashi M, Nishi T, Shibahara T, Tanaka T, Ishii S, Yamamoto Ji, Saito K, Kawai Y, Isono Y, Nakamura Y, Nagahari K, Murakami K, Yasuda T, Iwayanagi T, Wagatsuma M, Shiratori A, Sudo H, Hosoiri T, Kaku Y, Kodaira H, Kondo H, Sugawara M, Takahashi M, Kanda K, Yokoi T, Furuya T, Kikkawa E, Omura Y, Abe K, Kamihara K, Katsuta N, Sato K, Tanikawa M, Yamazaki M, Ninomiya K, Ishibashi T, Yamashita H, Murakawa K, Fujimori K, Tanai H, Kimata M, Watanabe M, Hiraoka S, Chiba Y, Ishida S, Ono Y, Takiguchi S, Watanabe S, Yosida M, Hotuta T, Kusano J, Kanehori K, Takahashi-Fujii A, Hara H, Tanase To, Nomura Y, Togiya S, Komai F, Hara R, Takeuchi K, Arita M, Imose N, Musashino K, Yuuki H, Oshima A, Sasaki N, Aotsuka S, Yoshikawa Y, Matsunawa H, Ichihara T, Shiohata N, Sano S, Moriya S, Momiyama H, Satoh N, Takami S, Terashima Y, Suzuki O, Nakagawa S, Senoh A, Mizoguchi H, Goto Y, Shimizu F, Wakebe H, Hishigaki H, Watanabe T, Sugiyama A, Takemoto M, Kawakami B, Yamazaki M, Watanabe K, Kumagai A, Itakura S, Fukuzumi Y, Fujimori Y, Komiyama M, Tashiro H, Tanigami A, Fujiwara T, Ono T, Yamada K, Fujii Y, Ozaki K, Hirao M, Ohmori Y, Kawabata A, Hikiji T, Kobatake N, Inagaki H, Ikema Y, Okamoto S, Okitani R, Kawakami T, Noguchi S, Itoh T, Shigeta K, Senba T, Matsumura K, Nakajima Y, Mizuno T, Morinaga M, Sasaki M, Togashi T, Oyama M, Hata H, Watanabe M, Komatsu T, Mizushima-Sugano J, Satoh T, Shirai Y, Takahashi Y, Nakagawa K, Okumura K, Nagase T, Nomura N, Kikuchi H, Masuho Y, Yamashita R, Nakai K, Yada T, Nakamura Y, Ohara O, Isogai T, Sugano S: **Complete sequencing and characterization of 21,243 full-length human cDNAs.** *Nat Genet* 2004, **36:**40-45.
17. **Genome sequencing centre at the Washington University School of Medicine** [http://genome.wustl.edu]
18. Odronitz F, Kollmar M: **Pfarao: a web application for protein family analysis customized for cytoskeletal and motor proteins (CyMoBase).** *BMC Genomics* 2006, **7:**300 [http://www.cymobase.org].
19. **CyMoBase – a database for cytoskeletal and motor proteins** [http://www.cymobase.org]

Appendix C

Article about AUGUSTUS-PPX

"A novel hybrid gene prediction method employing protein multiple sequence alignments"

published in Bioinformatics (2011), doi: 10.1093/bioinformatics/btr010

A novel hybrid gene prediction method employing protein multiple sequence alignments

Oliver Keller [1,*], Martin Kollmar [2], Mario Stanke [3,*] and Stephan Waack [1]

[1]Institute of Computer Science, University of Göttingen, Goldschmidtstr. 7, 37077 Göttingen, Germany
[2]Department of NMR-based Structural Biology, Max-Planck-Institute for Biophysical Chemistry, Am Fassberg 11, 37077 Göttingen, Germany
[3]Institute of Mathematics and Computer Science, University of Greifswald, Walther-Rathenau-Str. 47, 17487 Greifswald, Germany

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Motivation: As improved DNA sequencing techniques have increased enormously the speed of producing new eukaryotic genome assemblies, the further development of automated gene prediction methods continues to be essential.

While the classification of proteins into families is a task heavily relying on correct gene predictions, it can at the same time provide a source of additional information for the prediction, complementary to those presently used.

Results: We extended the gene prediction software AUGUSTUS by a method that employs block profiles generated from multiple sequence alignments as a protein signature to improve the accuracy of the prediction. Equipped with profiles modelling human Dynein Heavy Chain (DHC) proteins and other families, AUGUSTUS was run on the genomic sequences known to contain members of these families. Compared both to AUGUSTUS' ab-initio version and Genewise in HMM mode, the rate of genes predicted with high accuracy showed a dramatic increase.

Availability: The AUGUSTUS project webpage is located at http://augustus.gobics.de, with the executable program as well as the source code available for download.

Contact: keller@cs.uni-goettingen.de, mario.stanke@uni-greifswald.de

1 INTRODUCTION

With ever faster and cheaper sequencing techniques, the amount of available nucleotide sequence data is growing rapidly. In comparison, the process of accurately annotating the generated data is still lagging far behind. Fully automated annotation is essential here as the sheer amount of data makes manual inspection, even as a secondary step, impossible on the whole. Thus, improving gene prediction tools that perform automated annotation of protein-coding regions as accurate as possible is becoming increasingly important for the generation, for example, of the corresponding protein sequence data.

In eukaryotes, the computational identification of the *gene structure* (in simplified terms, the locations of the protein-coding exons in a nucleotide sequence) is a complex and error-prone task. The most direct approach is pursued by *ab-initio* methods that do not need any input but the genomic target sequence itself. Commonly, sequences are considered the result of a random process, and the outcome is the gene structure that is most likely to randomly produce the target sequence. Parameters for the underlying probabilistic model, often a Generalized Hidden Markov Model, are derived from a training set of gene structures verified as accurate. As in any probabilistic approach, the prediction accuracy is limited already by constraints inherent to the model, even if it is a perfect description of the data.

In order to overcome these theoretical bounds, it is necessary to employ extrinsic sources of information that give hints whether an interval of the sequence is, for example, a coding exon. These gene finding methods are usually based on alignments with informant sequences: *comparative* methods make use of similarities with the DNA of closely related species, *transcript-based* methods map sequences to the target genome that are known to be expressed (such as ESTs or RNA-Seq) while *homology-based* methods try to map transcripts of related genes. Current approaches combine these methods, some including an ab-initio gene prediction. Gene structures found by these methods are the starting point for the pipelines generating reference annotations for genomes (Harrow *et al.*, 2009).

RNA-Seq (transcriptome sequencing with next-generation sequencing methods, Metzker, 2010) promises major advances for gene finding; however, currently the accuracy of RNA-Seq based gene prediction suffers from mapping ambiguities and partially contained introns, especially in complex genomes.

Homology-based gene predictors, such as Genewise (Birney *et al.*, 2004) and Exonerate (Slater and Birney, 2005), can determine gene structures by mapping a single protein sequence to the target genome, others like Projector (Meyer and Durbin, 2004) combine homology-based and comparative approaches. Cui *et al.* (2007) presented a combined homology-based and comparative gene finding method that extends the prediction beyond the homologous part to a complete gene structure but requires an established gene structure for the informant sequence. Protein queries highly identical to the target sequence can be mapped with BLAT (Kent, 2002); the

*to whom correspondence should be addressed

software Scipio (Keller *et al.*, 2008) can refine a mapping provided by BLAT such that the precise exon-intron structure of a gene is automatically recovered from the query.

The gene prediction program AUGUSTUS (Stanke and Waack, 2003) is able to incorporate *hints* from external sources to combine them with an ab-initio prediction (Stanke *et al.*, 2006b,a, 2008). The method employed in AUGUSTUS is completely generic as to the source of the hints and the way they have been generated; in practice, however, they are almost always derived from alignment-based methods (Stanke *et al.*, 2006a, 2008), including the use of peptides from proteomics experiments (Castellana *et al.*, 2008).

At a later stage of the genome annotation pipeline, an important task is the classification of proteins into families and subfamilies based on sequence similarities. A correct classification obviously relies on accurate gene predictions. Conversely, membership to a family is a potential source of information that can be made available already to the gene prediction. This information is commonly stored in protein family databases that are easily accessible, for example via InterPro (http://www.ebi.ac.uk/interpro/, Hunter *et al.*, 2009), quickly growing, and often equipped with precomputed models (also called *signatures*) in the form of profile-HMMs, MSAs and similar representations. Furthermore, researchers specialised on specific families will be interested in tools that enable them to use their existing sequence repositories to improve the prediction on newly available nucleotide data.

While almost every resource of protein family signatures offers its own methods to classify *protein* query sequences supplied by the user (bundled by InterProScan, Quevillon *et al.*, 2005), these methods cannot be applied directly to (eukaryotic) *genomic* sequences, without the prior knowledge of the gene structure, and hence the coding sequence. On the other hand, most protein-based gene finding methods map single sequences rather than protein signatures. The program Genewise has an HMM mode that can perform a combination of gene prediction and protein signature recognition using a profile-HMM (of one family at a time) in place of a single protein sequence.

Here, we present a novel approach that uses what we call a hybrid method as it combines the existing ab-initio model with the protein signature as an additional model for protein family membership of the resulting transcripts. It is implemented as an extension (the protein profile extension, PPX) to the program AUGUSTUS. In this approach, evidence from complementary sources, such as RNA-Seq, can be used simultaneously.

2 APPROACH

With the integration of a protein model into gene prediction, we pursue two goals: first, the identification of members of a given protein family, and more importantly, an increased accuracy of the prediction designed especially to improve identification rates.

In AUGUSTUS-PPX, protein families are modelled by *block profiles*. In this context, a *block* is an ungapped and highly conserved section of a multiple sequence alignment (MSA). The concept of a block was first introduced by Henikoff and Henikoff (1991), and then used to classify proteins in the Blocks database (Pietrokovski *et al.*, 1996; Henikoff *et al.*, 1999).

Very similarly, collections of blocks, together with the ranges of admissible distances between consecutive blocks, have been used as protein signatures, referred to as *fingerprint* (Attwood and Beck, 1994), and currently collected in the PRINTS database (Attwood *et al.*, 2003), a member database of InterPRO.

A block profile is a collection of position-specific frequency matrices, each describing the amino acid distribution in a block, similar to a profile-HMM. However, in contrast to profile-HMMs, sequence motifs modelled by blocks have a fixed length with no insertions or deletions permitted inside a block. Along the full length of a multiple sequence alignment, non-conserved regions alternate with blocks. These inter-block sequence parts are modelled in a block profile only by constraining their length.

In the extended version of AUGUSTUS, one block profile at a time can be provided as an additional input, representing a particular protein family of interest. Here, we will refer to it as a *protein profile* as it models a gene with respect to its protein sequence.

AUGUSTUS predicts genes using a Generalized Hidden Markov Model (GHMM), in which each of the states corresponds to the biological meaning of the sequence (exon, intron, intergenic, etc.); this turns a gene structure into a sequence of states, together with their sequence coordinates, also called a *parse*. The well-known Viterbi algorithm is used to compute the highest-scoring among all possible parses. In a GHMM, the score corresponds to the joint probability that the model generates the target sequence using the parse.

The profile extension to AUGUSTUS evaluates, for each candidate gene structure, a similarity score of the predicted transcript to the profile, giving a bonus to genes matching the profile. While the gene prediction takes place in genomic space, the protein profile models the *protein* sequence that the predicted gene translates to.

Although profile-HMMs can be a more powerful sequence model, block profiles were chosen here as a protein signature because the integration into the existing GHMM requires a reduced complexity. This is compensated by the use of the ab-initio model that can better predict the less conserved sequence parts.

The coordinates of the protein model are mapped to the input DNA sequence when considering a candidate gene structure. Genes consisting of multiple exons will induce a mapping of partial profiles, where blocks may be disconnected by introns. Inter-block regions impose a constraint on exon length between blocks, but modelling nucleotide composition in them is left to AUGUSTUS' exon model.

Genes that show no evidence for similarity to all of the blocks in the profile are predicted the same way they would without the profile extension. This is an advantage over purely homology-based approaches which cannot predict regions without sufficient homology. On the other hand, exons containing distant blocks can be forced to belong to the same gene, addressing the split gene problem common to ab-initio approaches: while the predicted coding regions of a long gene may largely agree, they are frequently mispredicted as several shorter ones.

Instead of using the output of a separate program as source of extrinsic information, as is the case in the hints approach introduced by Stanke *et al.* (2006b), the mapping of the block profile to the target sequence is created *in parallel* to the ab-initio prediction, with a mutual interaction between both.

The profile can be complemented with information about conserved intron positions (relative to the protein sequence), among the members of a protein family. This *intron profile* is a type of information that is already lost when dealing with MSAs but can be very valuable for the gene prediction.

3 METHODS

3.1 Block profiles representing protein families

Scoring function defined by block profiles. A set of n blocks can be transformed into a set of frequency matrices, one for each block, containing the column-specific frequencies of amino acids.

The order of the blocks is assumed to be preserved throughout all sequences in the family. For each b, the interval $I_b = [d_b^{\min}, d_b^{\max}]$ specifies the range of admissible distances between consecutive block motifs (or, in the cases $b = 0$, $b = n$, between the first/last block motif and the sequence start/end, respectively).

We call such a collection of frequency matrices, together with the range intervals I_b, a *block profile*, in analogy to other profiles generated from multiple sequence alignments. For the sake of brevity, we will use the term *block* also for the particular matrix that represents it. A block profile does not contain probabilities for insertions or deletions, and it does not model the sequence regions between blocks.

From each frequency matrix, *odd-ratios* are obtained by dividing each entry by the random (background) distribution of amino acids, and used as the scoring matrix $R^{(b)}$ for block b. If $s = s_0 \dots s_{w-1}$ is an amino acid sequence, its similarity to block b of length w is expressed by the odds ratio score of s, the product of the respective entries of the scoring matrix: $\rho^{(b)}(s) = R_0(s_0) \cdot \ \dots \ \cdot R_{w-1}(s_{w-1})$. More generally, a *partial block score* $\rho^{(b)}_{[j..k]}(s) = R_j(s_0) \cdot \dots \cdot R_k(s_{k-j})$ can be defined if s is a (shorter) sequence of length $k - j + 1$.

To classify a given protein sequence s of length w as a *block hit*, we turn the scoring function into a decision function by requiring $\rho(s) > \tau$, with a block-specific threshold $\tau = \tau^{(b)}$, which is controlled by global parameters θ_{sens} and θ_{spec} that give upper bounds for the expected error rates in the respective models for blocks and background sequences (for details see the supplementary material).

Generating block profiles. To convert a MSA into a block profile that can be used as input to AUGUSTUS, we provide separate tools that compute frequency matrices from each conserved ungapped region in the alignment. By the definition of a block, each member sequence of the family must contain all motifs that are part of a block, without insertions or deletions. Only a minimum number (usually set to 6–10) of subsequent gapless columns will actually form a block. These columns we call *usable* for conversion to a block profile. Additional restrictions might be imposed, for example on the degree of conservation in a block column.

Large protein families may be composed of subfamilies characterised by domains that are shared only by a subset of all sequences. If no domain is present in all member sequences, a family cannot be described appropriately by one single block profile. A possible solution is to convert the alignment associated with each subfamiliy into a separate block profile. In the next subsection, we present a different approach of determining a "core" alignment that can be transformed into a profile.

Once blocks have been extracted from an alignment, or retrieved from the PRINTS or Blocks databases, PSSMs can be calculated by determining column-wise relative frequencies. In our conversion scripts we follow the methods used for the Blocks database: a position-specific weighting scheme (Henikoff *et al.*, 1990) is applied in order to avoid over-representation of similar motifs, and pseudo counts are determined by regularizing the position-specific counts with BLOSUM matrices. Range intervals are determined from an alignment by taking minimum and maximum lengths of inter-block sequence parts among all aligned sequences, or taken directly from a database.

The restriction that no insertions or deletions occur in a block motif can be relaxed, either by splitting a block into two to allow insertions in a central position of a large block, or by merging amino acid distributions of neighbouring positions. This way, also profile-HMMs (e.g., given in HMMER format) – also containing frequency tables, but equipped with deletion and insertion states – can be used for conversion into block profiles.

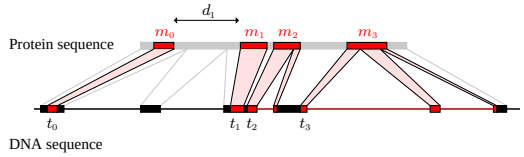

Figure 1. A profile-DNA mapping. The translated transcript (above) contains four block motifs (in red) $m_0,\dots,m_3$, separated by inter-block sequence (in grey) of length d_b. The coding gene (below) consists of six exons; sequences coding for the block motifs are shown in red. Block hits may appear inside an exon (m_0,m_1) or be disconnected by one or more introns (m_2,m_3).

Preprocessing of Multiple Sequence Alignments. For the preparation of MSAs found in PFAM, we implemented an algorithm that iteratively discards sequences from an alignment until both a required number of usable columns is reached (see above), and the estimated overall profile size (the number of usable columns multiplied with the number of used sequences) reaches a maximum. To this end, each column in an input alignment is categorized: Either

1. it contains only a small number of gaps (removing the corresponding sequences would turn it into a block column), or
2. it contains only a small number of non-gap characters (making it a candidate for complete removal from the alignment).

Columns that do not fall into these two categories, are already determined to be inter-block columns at this stage, as well as isolated columns that cannot be extended to a block of minimal length.

Now, each column in the alignment can be considered as a potential start of a block of minimal length. Each sequence is *in conflict* with the block starting there if there is a deletion (the sequence has a gap in a column of category 1) or an insertion (the sequence has a residue character in a column of category 2) within the minimum number of columns after the fixed block start.

In each iteration, we use a heuristic to pick a new block start to be introduced by removing its conflicting sequence set from the alignment, based on the expected new profile size of the alignment; this is repeated until that size cannot be increased further by removing more sequences. The resulting sub-alignment can now be described by a block profile.

3.2 Evaluating profile-DNA mappings

A block profile is mapped to the DNA sequence σ by specifiying the start locations $(t_0, \dots, t_{n-1})$ of the segments coding for potential block motifs. We denote any such mapping by ψ. Since blocks may be interrupted by introns, the full mapping is well-defined only when also a candidate gene structure ϕ is given, as depicted in Figure 1. In this case, ψ is *valid* if all t_b lie on exons belonging to the same gene, and the inter-block regions on the protein sequence have admissible lengths $d_b^{\min} \le d_b \le d_b^{\max}$.

The score of the mapping is defined by the log-odds of the candidate block motifs m_b on the protein sequence

$$\rho(\sigma;\phi;\psi) = \rho^{(0)}(m_0) \cdot \ \dots \ \cdot \rho^{(n-1)}(m_{n-1}). \qquad (1)$$

Several genes may be equipped with mappings ψ_1,ψ_2,$\dots$,ψ_m, resulting in a total score of

$$\rho(\sigma;\phi;\psi_1,\dots,\psi_m) = \rho(\sigma;\phi;\psi_1) \cdot \ \dots \ \cdot \rho(\sigma;\phi;\psi_m).$$

Formally, ρ can be defined to be 0 for invalid mappings. When provided with a protein profile, the Viterbi algorithm will search for the best-scoring combination of gene structure and compatible profile-DNA mappings. More

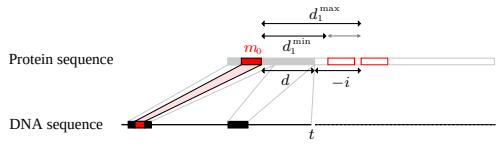

Figure 2. Calculating Viterbi variables at DNA position t, for substate (b, i), $b = 1$, $i < 0$. The score is maximized for all gene structures that have a mapping to block motif m_0 at a fixed distance to the current position. In any parse continuing from this substate, the transcript must have a block motif m_1 starting between $d_1^{\min}$ and $d_1^{\max}$ from m_0. This determines the admissible range of block starts on the following exon (unless it is short enough to fit into the inter-block region). The bonus awarded to the Viterbi variable for this mapping is $\rho^{(0)}(m_0)$.

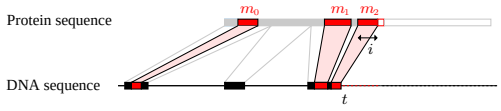

Figure 3. Calculating Viterbi variables at DNA position t, for substate (b, i), $b = 2$, $i > 0$. The score is maximized for all gene structures that have a mapping to the truncated block motif m_2 of length i at the end of the exon. Any parse continuing from here must have the next exon starting with the remainder of the motif. The bonus awarded to the Viterbi variable for this mapping is $\rho^{(0)}(m_0) \cdot \rho^{(1)}(m_1) \cdot \rho^{(2)}_{[0..i-1]}(m_2)$.

precisely, it determines $\phi, \psi_1, \ldots, \psi_m$ maximizing the combined score

$$P(\phi, \sigma) \cdot \rho(\sigma; \phi; \psi_1, \ldots, \psi_m),$$

where the joint probability $P(\phi, \sigma)$ for gene structure and sequence is multiplied with the *profile bonus* $\rho(\sigma, \phi, \psi)$ for each candidate transcript that permits a valid mapping ψ. Thus, only if a gene is compatible to a profile mapping with a score high enough to compensate for a lower ab-initio score, the prediction with the profile will differ from the prediction without. In particular, on sequences where no members of the protein family are identified, the result will be identical to the ab-initio prediction. For performance reasons, we consider for the evaluation of the profile bonus only mappings that consist of block hits (each of the $\rho^{(b)}$ must exceed their threshold $\tau^{(b)}$).

In the underlying sequence model, multiplying with ρ effectively amounts to replacing the background model with the block model, for the emission probabilities.

3.3 Integration into AUGUSTUS’ state model

Iterating over all DNA positions, the Viterbi algorithm computes the scores of optimal partial parses (candidate gene structures) ending in the current position, and stores them in variables indexed by position and last state, each state representing a different sequence type, strand, or reading frame. The state model AUGUSTUS uses has been described in Stanke and Waack (2003).

When equipped with a protein profile, the score assigned to a parse is modified as described above: for each gene in the parse that is compatible with a profile-DNA-mapping, the score is to be multiplied with the best possible profile bonus; each exon in the gene contributes the bonus for the block (part) hits that overlap with it.

When the algorithm arrives at a position in the DNA, it must consider all partial profile mappings started before that can be continued beyond this position, having a particular position within the profile aligned to the current DNA position (see Figure 2). This position determines the conditions for the mapping to be continued; thus, a separate score has to be stored at that position for each profile position. To this end, the original state space of AUGUSTUS was extended by a set of substates attached to the main state, representing the position in the profile, specified as a pair of integers (b, i) denoting block (b) and position i relative to the block ($i \leq 0$ before a block, and $i > 0$ inside a block). As shown in Figure 2, if i is negative, it determines the admissible range for the start of block b on the next exon.

The substates serve as additional indices for the Viterbi table labelling the entries that the *combined* scores are stored. For a partial profile mapping ending at the substate (b, i), the profile score multiplied to the ab-initio score is given by equation (1), except that the product is truncated at the current position: $\rho^{(0)}(m_0) \cdot \ldots \cdot \rho^{(b-1)}(m_{b-1}) \cdot \rho^{(b)}_{[0..i-1]}(m_b)$ (the last factor is a partial block score included only in the case $i > 0$, see Figure 3). The maximum of all combined scores ending in the same substate is then the value stored in the table.

In general, substates can be used to model side conditions that influence the score of a parse, for example if constraints on the exon length are imposed, or for the exclusion of spliced stop codons. Here, substates are aligning the profile to the exon ends.

The emission probability of an exon in the extended version depends on the substates on both ends: it gets a bonus equal to the maximum score of all profile mappings on the exon that are compatible to the substates. Each pair of substates gives rise potentially to a different profile bonus. New substate variables are calculated by maximizing the combined score, over all predecessor positions and predecessor substates.

In introns, the emission probabilities remain unchanged, but intron states as well need to be equipped with substates to indicate a potential profile position they are mapped to.

Genes on the backward strand are evaluated essentially in the same way as genes on the forward strand; however, in this case, the profile is mapped to the DNA “backwards”, starting with the C-terminus. Correspondingly, the substates on the backward strand refer to blocks and columns in reverse order: the first motif is evaluated by the last block in the profile, starting with the last block column.

Since the model attaches a high number substates to every main state of the original model, an exhaustive evaluation of all combinations of block locations and substates would cause an explosion of running time and memory requirements; therefore, most of the substate entries are eliminated or shared between main states in order to control the computational cost. We store substate scores dynamically, reserving memory only for nonzero entries. See the supplementary material for details on speed-up strategies.

3.4 Fast preliminary block hit search

To make it possible to run AUGUSTUS-PPX on a whole-genome scale, a fast preliminary search algorithm was devised for determining the genomic regions that are likely to contain genes matching the profile, without determining detailed gene structures. The algorithm was implemented in a separate executable that can be used before running AUGUSTUS. Given a target sequence and a protein profile, it performs the following steps:

- Building a seed collection: For each of the 8 000 possible triplets of amino acids, the positions in the profile are stored where that triplet is likely to occur.
- Determining block hit candidates: iterating over the target sequence, each 9-tuple is tested for being a seed. Any segment covered at least 25% by seeds for a block is considered for exhaustive evaluation. This prefiltering step is very fast, since it consists of a simple lookup with nucleotides triple as keys.
- Exhaustive evaluation of candidates: given a block start offset, partial scores $\rho_{[j..k]}(a_j..a_k)$ are maximized where $a_0..a_{w-1}$ is the translated DNA segment. Ths interval is stored as a partial block hit if its size reaches the minimal block width and the (partial) block threshold is exceeded.

- Assembling block hits to profile hits: each block hit is extended to a series of block hits by joining neighbouring hits, allowing for blocks being skipped. Using dynamic programming, sequences of block hits are determined that are highest-scoring for all contained blocks, and returned as profile hits.

This algorithm is fast enough to be run on whole genomes with a high number of profiles, in order to determine the regions AUGUSTUS-PPX is then run on.

4 DATA SETS

4.1 Dynein Heavy Chain (DHC) family

Dynein Heavy Chain (DHC) proteins belong to the longest proteins in eukaryotes comprising more than 4000 amino acids. They can be grouped into several subfamilies that are either responsible for intracellular transport and mitosis or part of the complex microtubule-based structures in cilia and axonemes. In mammals most of the DHC genes are spread in up to hundred exons over several hundred-thousand of base pairs. The 16 human DHC sequence fall into 10 subfamilies DHC1 to DHC9 and DHC11.

The DHC genes have all been manually assembled and verified because existing gene prediction programs were not able to correctly predict the gene structures over such long distances. For 13 of the 16 human DHC sequences, cDNA data is available and has been used for prediction. The remaining three DHC sequences, and the DHC genes of the other organisms, have been manually assembled based on comparative analysis of the metazoan homologs in the subfamily.

The manually created and curated multiple sequence alignment of the DHC family proteins, currently comprising more than 1 600 DHC sequences, has been the best control and guide during this process. These manually assembled and verified DHC sequences are regarded as almost correct predictions and taken as reference sequences for the test runs. The DHC data is available from CyMoBase (http://www.cymobase.org, Odronitz and Kollmar, 2006).

From a complete sequence alignment of all Dynein Heavy Chain members available from CyMoBase, the alignment of the human sequences was converted into block profiles that were used as input in the test runs of AUGUSTUS.

A total of 96 DHC proteins out of six species were chosen as reference sequences; these were human, mouse, chicken, the Clawed frog, and the Owl limpet, a sea snail. The range of members and subfamilies varies slightly between species.

4.2 PFAM alignments

To assess the performance of the profile extension on a wider set of protein families, five additional profiles were produced from MSAs downloaded from the PFAM database. The families were chosen randomly from the set of all families that had a minimum average length of at least 400 residues, and a minimum of 30 human representatives in them. Short alignments that cover only single domains were not considered, since AUGUSTUS-PPX is designed for the case of full-length protein signatures.

From each alignment, a core alignment was produced with the procedure described in section 3.1. Typically, about 60% of the sequences were discarded in order to maximize the size of the usable part of the alignment. Among the core sequences, a total of 46 human sequences remained that were taken as reference and were downloaded from UniProt.

4.3 Genomic reference sequences

From the protein sequences chosen as reference set, i.e. the 96 DHC proteins from six species, and the 46 human sequences from the PFAM alignments tested, we produced reference gene structures. We used the program Scipio (Keller *et al.*, 2008; Odronitz *et al.*, 2008) which is able to establish the exact exon/intron structure of a given protein.

The reference genes for the DHC family consisted of 75 exons on average, spanning a length from 30 up to 450 kbps.

Depending on the quality of the assembly, the reference gene structures in some cases could not fully be determined; the corresponding parts in the protein sequence had been determined from unmapped contigs of the genome genome, or from cDNA analysis. When they were missing completely, their length had been estimated from the alignment to orthologous sequences. Furthermore, the accuracy of gene structures might be affected by frameshifts caused by sequencing errors.

5 RESULTS AND DISCUSSION

5.1 Setup of runs

AUGUSTUS was run on the regions containing the reference genes, both in the ab-initio and PPX versions, and the results were compared to the reference genes. The DHC genes were chosen as examples as their size makes their prediction prone to difficulties like the split gene problem. The human DHC used in the runs consisted of 35 blocks with a total length of 1180 sites (columns), the largest block having a motif length of 134.

In a second set of runs, allowing for the scenario that no ortholog to the target sequence is known, the human ortholog to the test sequence was removed from the alignment used to generate the profile, so that the remaining sequences all had less than 60% identity to the target sequence ("ex-ortholog").

We also ran Genewise in HMM mode on the reference regions, supplied with a profile-HMM in HMMER format, automatically generated from the same alignments, including the ex-ortholog ones. While Genewise has been succeeded by faster tools such as Exonerate in the case of single-protein queries, we are not aware of any other program that can perform a spliced alignment of a profile-HMM to a genomic sequence. On average, running times of Genewise in HMM mode were about 200 times longer than those of AUGUSTUS-PPX.

In order to compare a single-query mapping approach designed for high homology, cross-species searches with human queries for their orthologs were executed with Scipio, using default parameters.

Finally, to examine a random set of various reference proteins, runs were performed with the profiles generated from the five PFAM alignments, comparing AUGUSTUS-PPX to AUGUSTUS ab-initio and Genewise. We ran a low-similarity scenario here as well, by taking out from the alignments all sequences with more than 60% identity.

5.2 Assessment of the prediction quality

Results of the DHC runs. The task of recognizing a gene as a member of the DHC protein family was accomplished by AUGUSTUS-PPX in almost all cases. Four sequences from subfamilies with lesser sequence identity to the rest, and one case with an incomplete

Table 1. Accuracy of DHC runs

species	AUGUSTUS-PPX		AUG.	Scipio	Genewise	
	full	ex-ortho	ab-initio	cross-sp.	full	ex-ortho
exon level sensitivity (%)						
human	92.9	91.2	84.3	N/A	83.7	52.7
mouse	93.3	91.3	85.8	86.0	80.4	50.1
chicken	88.8	87.3	80.5	66.0	68.8	46.2
frog	84.4	82.9	77.8	48.2	64.3	48.0
zebrafish	89.0	88.9	80.3	49.2	68.5	49.1
snail	86.0	84.5	78.5	31.6	59.3	49.1
total	89.0	87.6	81.3	63.5	70.9	49.4
exon level specificity (%)						
range	76–93	74–90	77–88	52–88	74–85	68–73
average	85.5	84.0	83.3	76.5	79.3	69.7
highly accurate genes (%)						
human	62.5	62.5	31.3	N/A	93.8	12.5
mouse	72.2	66.7	27.8	88.9	55.6	0.0
chicken	41.7	33.3	16.7	8.3	16.7	0.0
frog	38.9	38.9	0.0	5.6	11.1	0.0
zebrafish	57.1	50.0	57.1	7.1	14.3	0.0
snail	44.4	44.4	11.1	0.0	11.1	0.0
total	53.1	50.0	14.6	36.5	34.4	2.1

Comparing the results of AUGUSTUS-PPX with AUGUSTUS ab-initio, single-sequence cross-species search (Scipio, based on BLAT), and Genewise in HMM mode. “ex-ortho” refers to runs with orthologs of target genes removed from the MSA. Highly accurate genes are those predicted with at least 95% sensitivity and 85% specificity.

genomic reference sequence were not identified as DHCs. Instead, these genes were predicted identically to the ab-initio version.

Scipio (run only where human orthologs were there) and Genewise failed to identify the same set of sequences, but did not make a prediction in these cases. In the ex-ortholog scenario, Genewise left 14 sequences unidentified.

Table 1 shows prediction accuracy of the testing scenarios, compared to the ab-initio version, Scipio, and Genewise. At exon level, there is significant gain in sensitivity compared to the ab-initio version, with almost 40% of the previously missed or mispredicted exons corrected, throughout the tested species. Specificity was overall increased slightly, but showed a decrease in some of the more distant species. Genewise is somewhat weaker in predicting exact splice sites, resulting in lower accuracy than AUGUSTUS ab-initio, even when using the full profile.

With genes consisting of more than 70 exons, the requirement that a gene is predicted entirely correctly has to be relaxed somewhat when assessing prediction quality at the gene level. We calculated the rate of *highly accurate* genes, meaning that the overlap of prediction and reference is at least 95% of the reference and 85% of the prediction. Here, the prediction quality of the human sequences of both Genewise and AUGUSTUS-PPX was dramatically better than the ab-initio prediction; the protein-based prediction tools benefit especially from joining predicted exons to a single gene. However, the accuracy of Genewise dropped significantly on the more distant species, and especially in the “ex-ortholog” scenario. In the single-query approach pursued by running Scipio, prediction quality at all levels deteriorates strongly with evolutionary distance.

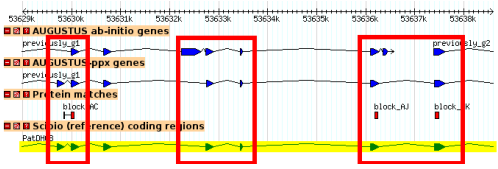

Figure 4. A section of an example result of AUGUSTUS-PPX, shown in GBrowse. Compared to the ab-initio prediction, one exon containing a block hit is added by the extension, one false positive exon removed to satisfy the distance constraints, and two genes are joined into one.

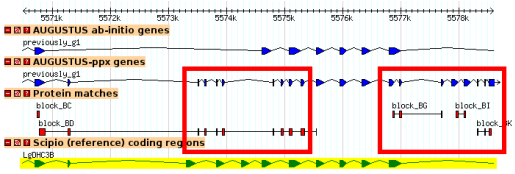

Figure 5. False positive exons due to a missing block enforced by the profile extension.

Detailed analysis of results There are various ways the profile can improve the prediction, as illustrated in Figure 4. A gene with complementary block hits on them, previously mispredicted as two genes, is now joined to one. This is an important advantage that protein-based gene finders have in comparison to ab-initio tools.

Overall, specificity was improved to a lesser extent; in some genes, we observed a deterioration of specificity at exon level when using the profile, while in others it was clearly improved. These heterogenous results are caused by block hits enforced on the sequence by the profile extension, leading to a number of false-positive exons added to the prediction, as can be seen in Figure 5. This occurred in the more distant species and in the ex-ortholog scenario, and also when the assembly quality was low.

We addressed this issue by performing a third set of runs where we executed a fast block search (see section 3.4) prior to the actual AUGUSTUS run (shown as supplementary data). The search can be used to determine the regions for the gene prediction, but it also outputs a list of block hits found there. The missing blocks were then removed from the DHC profile used in the AUGUSTUS-PPX prediction. With the filtered profile, no exons were added by enforced block hits, resulting in a higher exon specificity in all species.

Runs with profiles generated from PFAM alignments. Results of the runs on the 46 reference genes from the five PFAM protein families are shown in Table 2. Exon level sensitivity and rate of highly accurate genes were improved, to varying extent, in all five cases. The number of completely correct genes rose from 10 (21.7%) to 14 (30.4%). In all but one of the 46 cases, the genes were identified by AUGUSTUS-PPX as members of their families. Results deteriorated only slightly when we restricted the profile to low identity, and occasionally even improved (removing similar sequences can lead to more blocks in the profile, or prevent false positive block hits); two more sequences were not recognised as members. Genewise did not reach the accuracy of AUGUSTUS ab-initio.

Table 2. Accuracy of PFAM runs

family	AUGUSTUS-PPX full	AUGUSTUS-PPX <60%	AUGUSTUS ab-initio	Genewise
exon level sensitivity (%)				
HSP70	92.1	89.5	84.2	10.5
Aldedh	92.6	90.1	86.0	48.8
AA_permease	83.9	85.4	81.0	24.4
Cullin	87.7	86.9	83.1	56.9
Sec1	93.9	94.7	87.1	44.7
total	89.0	88.8	83.9	39.3
exon level specificity (%)				
range	72–92	71–89	65–89	25–77
average	80.5	78.9	75.9	57.7
highly accurate genes (%)				
	56.5	54.3	39.1	0.0
completely correct genes (%)				
	30.4	28.3	21.7	0.0

AUGUSTUS-PPX was compared to AUGUSTUS ab-initio and Genewise. "<60" refers to a profile with a maximum sequence identity of 60% to the target sequence, analogously to "ex-ortho" above.

Further testing. In order to verify the interoperability with external evidence, we ran AUGUSTUS on the human sequences with manually edited hints and the DHC profile simultaneously. When supplied with the hints for the exons still missing in the original predictions those were predicted correctly in general, unless there were non-standard splice sites. This showed that in principle there is the potential of adding the advantages of complementary methods, such as RNA-based evidence.

6 CONCLUSION

The gene prediction program AUGUSTUS was extended by a method combining protein-family based gene finding with an ab-initio prediction. Equipped with protein signatures, prediction accuracy could be improved considerably, especially on full-gene level on very long genes. The extrinsic protein data significantly improves the gene prediction compared to existing programs when sequence data only from distant species was available.

The presented approach is complementary to transcript-based methods, and easily combined with them, offering the potential of a further improvement of prediction accuracy.

Block profiles are a protein signature suitable for aiding gene prediction. The approach for extending the model is generic and can be used to describe other types of constraints, for example on coding sequence length. Future plans include the integration of intron profiles, containing information about conserved intron positions.

ACKNOWLEDGEMENT

OK wishes to thank Shmuel Pietrokovski for sharing his knowledge about blocks and granting access to tools for evaluating them.

All authors thank Florian Odronitz for helpful discussions and comments.

Funding: This work has been funded by the Deutsche Forschungsgemeinschaft [KO 2251/3-1 and KO 2251/6-1 to MK, and WA 766/6-1 to SW].

REFERENCES

Attwood, T. K. and Beck, M. E. (1994). Prints–a protein motif fingerprint database. *Protein Eng*, **7**(7), 841–848.

Attwood, T. K., Bradley, P., Flower, D. R., Gaulton, A., Maudling, N., Mitchell, A. L., Moulton, G., Nordle, A., Paine, K., Taylor, P., Uddin, A., and Zygouri, C. (2003). Prints and its automatic supplement, preprints. *Nucleic Acids Res*, **31**(1), 400–402.

Birney, E., Clamp, M., and Durbin, R. (2004). Genewise and genomewise. *Genome Res*, **14**(5), 988–995.

Castellana, N. E., Payne, S. H., Shen, Z., Stanke, M., Bafna, V., and Briggs, S. P. (2008). Discovery and revision of arabidopsis genes by proteogenomics. *Proc Natl Acad Sci U S A*, **105**(52), 21034–21038.

Cui, X., Vinar, T., Brejov, B., Shasha, D., and Li, M. (2007). Homology search for genes. *Bioinformatics*, **23**(13), i97–103.

Harrow, J., Nagy, A., Reymond, A., Alioto, T., Patthy, L., Antonarakis, S. E., and Guig, R. (2009). Identifying protein-coding genes in genomic sequences. *Genome Biol*, **10**(1), 201.

Henikoff, S. and Henikoff, J. G. (1991). Automated assembly of protein blocks for database searching. *Nucleic Acids Res*, **19**(23), 6565–6572.

Henikoff, S., Wallace, J. C., and Brown, J. P. (1990). Finding protein similarities with nucleotide sequence databases. *Methods Enzymol*, **183**, 111–132.

Henikoff, S., Henikoff, J. G., and Pietrokovski, S. (1999). Blocks+: a non-redundant database of protein alignment blocks derived from multiple compilations. *Bioinformatics*, **15**(6), 471–479.

Hunter, S., Apweiler, R., Attwood, T. K., Bairoch, A., Bateman, A., Binns, D., Bork, P., Das, U., Daugherty, L., Duquenne, L., Finn, R. D., Gough, J., Haft, D., Hulo, N., Kahn, D., Kelly, E., Laugraud, A., Letunic, I., Lonsdale, D., Lopez, R., Madera, M., Maslen, J., McAnulla, C., McDowall, J., Mistry, J., Mitchell, A., Mulder, N., Natale, D., Orengo, C., Quinn, A. F., Selengut, J. D., Sigrist, C. J. A., Thimma, M., Thomas, P. D., Valentin, F., Wilson, D., Wu, C. H., and Yeats, C. (2009). Interpro: the integrative protein signature database. *Nucleic Acids Res*, **37**(Database issue), D211–D215.

Keller, O., Odronitz, F., Stanke, M., Kollmar, M., and Waack, S. (2008). Scipio: using protein sequences to determine the precise exon/intron structures of genes and their orthologs in closely related species. *BMC Bioinformatics*, **9**, 278.

Kent, W. J. (2002). Blat–the blast-like alignment tool. *Genome Res*, **12**(4), 656–664.

Metzker, M. L. (2010). Sequencing technologies - the next generation. *Nat Rev Genet*, **11**(1), 31–46.

Meyer, I. M. and Durbin, R. (2004). Gene structure conservation aids similarity based gene prediction. *Nucleic Acids Res*, **32**(2), 776–783.

Odronitz, F. and Kollmar, M. (2006). Pfarao: a web application for protein family analysis customized for cytoskeletal and motor proteins (cymobase). *BMC Genomics*, **7**, 300.

Odronitz, F., Pillmann, H., Keller, O., Waack, S., and Kollmar, M. (2008). Webscipio: An online tool for the determination of gene structures using protein sequences. *BMC Genomics*, **9**(1), 422.

Pietrokovski, S., Henikoff, J. G., and Henikoff, S. (1996). The blocks database–a system for protein classification. *Nucleic Acids Res*, **24**(1), 197–200.

Quevillon, E., Silventoinen, V., Pillai, S., Harte, N., Mulder, N., Apweiler, R., and Lopez, R. (2005). Interproscan: protein domains identifier. *Nucleic Acids Res*, **33**(Web Server issue), W116–W120.

Slater, G. S. C. and Birney, E. (2005). Automated generation of heuristics for biological sequence comparison. *BMC Bioinformatics*, **6**, 31.

Stanke, M. and Waack, S. (2003). Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics*, **19 Suppl 2**, 215–215.

Stanke, M., Tzvetkova, A., and Morgenstern, B. (2006a). Augustus at egasp: using est, protein and genomic alignments for improved gene prediction in the human genome. *Genome Biol*, **7 Suppl 1**, 1–8.

Stanke, M., Schöffmann, O., Morgenstern, B., and Waack, S. (2006b). Gene prediction in eukaryotes with a generalized hidden markov model that uses hints from external sources. *BMC Bioinformatics*, **7**, 62–62.

Stanke, M., Diekhans, M., Baertsch, R., and Haussler, D. (2008). Using native and syntenically mapped cdna alignments to improve de novo gene finding. *Bioinformatics*, **24**(5), 637–644.

Bibliography

[AB94] T. K. Attwood and M. E. Beck. Prints–a protein motif fingerprint database. *Protein Eng*, 7(7):841–848, Jul 1994.

[ABF+03] T. K. Attwood, P. Bradley, D. R. Flower, A. Gaulton, N. Maudling, A. L. Mitchell, G. Moulton, A. Nordle, K. Paine, P. Taylor, A. Uddin, and C. Zygouri. Prints and its automatic supplement, preprints. *Nucleic Acids Res*, 31(1):400–402, Jan 2003.

[BCD+04] Alex Bateman, Lachlan Coin, Richard Durbin, Robert D Finn, Volker Hollich, Sam Griffiths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik L L Sonnhammer, David J Studholme, Corin Yeats, and Sean R Eddy. The pfam protein families database. *Nucleic Acids Res*, 32(Database issue):D138–D141, Jan 2004.

[BCHP07] Axel Bernal, Koby Crammer, Artemis Hatzigeorgiou, and Fernando Pereira. Global discriminative learning for higher-accuracy computational gene prediction. *PLoS Comput Biol*, 3(3):e54, Mar 2007.

[Con01] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, Feb 2001.

[DEKM99] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Camrbridge University Press, 1999.

[GKHC01] J. Gough, K. Karplus, R. Hughey, and C. Chothia. Assignment of homology to genome sequences using a library of hidden markov models that represent all proteins of known structure. *J Mol Biol*, 313(4):903–919, Nov 2001.

[HAA+09] Sarah Hunter, Rolf Apweiler, Teresa K Attwood, Amos Bairoch, Alex Bateman, David Binns, Peer Bork, Ujjwal Das, Louise Daugherty, Lauranne Duquenne, Robert D Finn, Julian Gough, Daniel Haft, Nicolas Hulo, Daniel Kahn, Elizabeth Kelly, Aurlie Laugraud, Ivica Letunic, David Lonsdale, Rodrigo Lopez, Martin Madera, John Maslen, Craig McAnulla, Jennifer McDowall, Jaina Mistry, Alex Mitchell, Nicola Mulder, Darren Natale, Christine Orengo, Antony F Quinn, Jeremy D Selengut, Christian J A Sigrist, Manjula Thimma, Paul D Thomas, Franck Valentin, Derek Wilson, Cathy H Wu, and Corin Yeats. Interpro: the integrative protein signature database. *Nucleic Acids Res*, 37(Database issue):D211–D215, Jan 2009.

[HH91] S. Henikoff and J. G. Henikoff. Automated assembly of protein blocks for database searching. *Nucleic Acids Res*, 19(23):6565–6572, Dec 1991.

[HHP99] S Henikoff, J G Henikoff, and S Pietrokovski. Blocks+: a non-redundant database of protein alignment blocks derived from multiple compilations. *Bioinformatics*, 15(6):471–479, Jun 1999.

[Jon06] Steven J M Jones. Prediction of genomic functional elements. *Annu Rev Genomics Hum Genet*, 7:315–338, 2006.

[Kar01] S. Karlin. Detecting anomalous gene clusters and pathogenicity islands in diverse bacterial genomes. *Trends Microbiol*, 9(7):335–343, Jul 2001.

[Ken02] W J Kent. Blat–the blast-like alignment tool. *Genome Res*, 12(4):656–664, Apr 2002.

[LCM+10] Konstantinos Liolios, I-Min A Chen, Konstantinos Mavromatis, Nektarios Tavernarakis, Philip Hugenholtz, Victor M Markowitz, and Nikos C Kyrpides. The genomes on line database (gold) in 2009: status of genomic and metagenomic projects and their associated metadata. *Nucleic Acids Res*, 38(Database issue):D346–D354, Jan 2010.

[LMP01] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, 2001.

[MLUL+05] Huaiyu Mi, Betty Lazareva-Ulitsky, Rozina Loo, Anish Kejariwal, Jody Vandergriff, Steven Rabkin, Nan Guo, Anushya Muruganujan, Olivier Doremieux, Michael J Campbell, Hiroaki Kitano, and Paul D Thomas. The panther database of protein families, subfamilies, functions and pathways. *Nucleic Acids Res*, 33(Database issue):D284–D288, Jan 2005.

[NGI00] Y. Nakamura, T. Gojobori, and T. Ikemura. Codon usage tabulated from international dna sequence databases: status for the year 2000. *Nucleic Acids Res*, 28(1):292, Jan 2000.

[NW70] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3):443–453, Mar 1970.

[PHH96] S Pietrokovski, J G Henikoff, and S Henikoff. The blocks database–a system for protein classification. *Nucleic Acids Res*, 24(1):197–200, Jan 1996.

[PP10] Ernesto Picardi and Graziano Pesole. Computational methods for ab initio and comparative gene finding. *Methods Mol Biol*, 609:269–284, 2010.

[QSP+05] E. Quevillon, V. Silventoinen, S. Pillai, N. Harte, N. Mulder, R. Apweiler, and R. Lopez. Interproscan: protein domains identifier. *Nucleic Acids Res*, 33(Web Server issue):W116–W120, Jul 2005.

[SKG+06] M Stanke, O Keller, I Gunduz, A Hayes, S Waack, and B Morgenstern. Augustus: ab initio prediction of alternative transcripts. *Nucleic Acids Res*, 34(Web Server issue):435–439, Jul 2006.

[SSMW06] M Stanke, O Schöffmann, B Morgenstern, and S Waack. Gene prediction in eukaryotes with a generalized hidden markov model that uses hints from external sources. *BMC Bioinformatics*, 7:62–62, 2006.

[Sta03] M Stanke. *Gene Prediction with a Hidden Markov Model*. PhD thesis, Universität Göttingen, 2003.

[SW03] M Stanke and S Waack. Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics*, 19 Suppl 2:215–215, Oct 2003.