Introduction to Gene-Finding-Problem
What Do Genes Look Like?
Statistical Features of Genes
The One-Dimensional Chaining Problem
Simple Approach to Gene Finding
Gene Finding with HMMs
Generalized HMMs
Model Design
Training

Vorlesung Genomanalyse vom 6.12.2011

Mario Stanke
Institut für Mathematik und Informatik
Universität Greifswald
### Prokaryotes

**Prokaryotes** are the set of species that lack a cell nucleus.

\[ \{\text{prokaryotes}\} = \{\text{bacteria}\} \cup \{\text{archea}\} \]

### Eukaryotes

**Eukaryotes** are the set of species whose cells have a nucleus. May be unicellular (e.g. some algae) or multicellular (plants and animals).
Prokaryotes, Eukaryotes

- the structure of prokaryotic genes is less complex than those of eukaryotes.
Prokaryotes, Eukaryotes

- the structure of prokaryotic genes is less complex than those of eukaryotes.
- prokaryotic gene finding is
  - easier,
  - algorithmically less interesting
  - and can be considered a special case (missing introns).
Prokaryotes, Eukaryotes

- the structure of prokaryotic genes is less complex than those of eukaryotes.
- prokaryotic gene finding is
  - easier,
  - algorithmically less interesting
  - and can be considered a special case (missing introns).
- We will therefore restrict lecture to eukaryotes
Structure of a eukaryotic gene

DNA ...

...actatacatactattttgaaggtgctaggacatgctctttttcatgaatgattggcaaatgtcattttttctagttcatggttggcaaacagtgggatcctgagagtcagataattgaattggctctgcctttaattatttgtcagcaagcccctgtccctttaggtgggaatatgtatgagggaccatatttggggttctggtagctccacagggatgcggtgatgagcgctgaatttatgacgtactag...
Structure of a eukaryotic gene

DNA

...ataaatacatattctgtagctgatggtttaaggggatgcaaataggtgctact...gaattggc...actaatagacatctatttcgagtcaaggtgtaggcaatgtccttttttctagtcatggttggcaaacagtg

... attatttgttcaagcaagcccctgtccctttaggtgggaatatgtatgagggaccatatttggggttctggtagctccacagggatgcggtgatgagcgctgaatttatgacgtactag...
Structure of a eukaryotic gene

DNA...actaagatgcgtgaaggtggagctgcacagagc...gene A...intergenic region...intergenic region...gene B...intergenic region...intergenic region
Structure of a eukaryotic gene

DNA

pre mRNA

gene A

intergenic region

intergenic region

transcription

1:1

gene B

intergenic region

UTR = UnTranslated Region = part of mRNA that is not translated
CDS = CcoDing Sequence = part of mRNA (exon) that is translated
Structure of a eukaryotic gene

DNA region
intergenic region
pre mRNA
mRNA
Structure of a eukaryotic gene

DNA

pre mRNA

mRNA

gene A

intergenic region

coding sequence of gene A

intergenic region

gene B

intergenic region

coding sequence of gene B

transcription

splicing
Structure of a eukaryotic gene
Structure of a eukaryotic gene

UTR = UnTranslated Region = part of mRNA that is not translated
CDS = Coding Sequence = part of mRNA (exon) that is translated

DNA → pre mRNA → mRNA → protein

transcription

DNA → pre mRNA

DNA → pre mRNA

pre mRNA → mRNA

pre mRNA → mRNA

mRNA → protein

mRNA → protein

coding sequence of gene A

protein A

coding sequence of gene B

protein B

exon

introns

exon

introns

exon

introns

exon

introns

exon

introns

exon

introns

exon

introns

exon

introns

exon

introns

exon

introns

UTR

UTR

UTR

UTR

UTR

UTR

UTR

UTR

UTR

UTR

UTR
Structure of a eukaryotic gene

DNA

pre mRNA

mRNA

coding sequence of gene A

coding sequence of gene B

protein A

protein B

transcription

splicing

translation and folding

UTR UTR UTR UTR UTR

intron intron intron intron intron

exon exon exon exon exon

CDS CDS CDS CDS CDS

intergenic region

intergenic region

Intergenic region
Translation

- **coding RNA sequence**
- **translation**
- **amino acid sequence**
- **folding**
- **protein**

- **codons**
- **one of 3 stop codons only at end**

```
RNA sequence
aug uau gag ...
gga uga
```

```
translation
folding
...
protein
```

```
“universeller” genetischer Code

<table>
<thead>
<tr>
<th>Kodon (DNA)</th>
<th>Aminosäure</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa</td>
<td>K</td>
</tr>
<tr>
<td>aac</td>
<td>N</td>
</tr>
<tr>
<td>aag</td>
<td>K</td>
</tr>
<tr>
<td>aat</td>
<td>N</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>atg</td>
<td>M</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

61 20
Kodons Aminosäuren
Translation

coding RNA sequence

translation

amino acid sequence

folding

protein

one of 3 stop codons only at end

codons

intron

"universeller" genetischer Code

<table>
<thead>
<tr>
<th>Kodon (DNA)</th>
<th>Aminosäure</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa</td>
<td>K</td>
</tr>
<tr>
<td>aac</td>
<td>N</td>
</tr>
<tr>
<td>aag</td>
<td>K</td>
</tr>
<tr>
<td>aat</td>
<td>N</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>atg</td>
<td>M</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Kodons Aminosäuren

61 20
Translation

- Coding RNA sequence
- Amino acid sequence
- Protein

- Introns
- Codons
- One of 3 stop codons only at end

```
<table>
<thead>
<tr>
<th>Kodon (DNA)</th>
<th>Aminosäure</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa</td>
<td>K</td>
</tr>
<tr>
<td>aac</td>
<td>N</td>
</tr>
<tr>
<td>aag</td>
<td>K</td>
</tr>
<tr>
<td>aat</td>
<td>N</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>atg</td>
<td>M</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```

"Universeller" genetischer Code

Translation in an RNA sequence with codons and amino acids, including the folding of the protein.
Genomanalyse
Mario Stanke

Introduction to Gene-Finding-Problem
What Do Genes Look Like?
Statistical Features of Genes
The One-Dimensional Chaining Problem
Simple Approach to Gene Finding
Gene Finding with HMMs
Generalized HMMs
Model Design
Training

Translation

coding
RNA sequence

translaction

amino acid sequence

folding

protein

codons

intron

one of 3 stop codons only at end

aug uau gag ... gga uga

“universeller” genetischer Code

<table>
<thead>
<tr>
<th>Kodon (DNA)</th>
<th>Aminosäure</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa</td>
<td>K</td>
</tr>
<tr>
<td>aac</td>
<td>N</td>
</tr>
<tr>
<td>aag</td>
<td>K</td>
</tr>
<tr>
<td>aat</td>
<td>N</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>atg</td>
<td>M</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

61 20 Kodons Aminosäuren
Signals

chr2L: a fruitfly gene

FlyBase Protein-Coding Genes

CG5001

example from fruit fly
Signals

transcription start site

translation start site
donor (5') splice site
acceptor (3') splice site
transcription termination site

donor (5') splice site

branch point region
acceptor (3') splice site

exon → intron

AGGTGAG

CAG

donor splice site (DSS) signal

GCAG

GTTG

acceptor splice site (ASS) signal

Frequency of the nucleotides at positions relative to splice site.

from green algae Chlamydomonas
Branch point: upstream of 3’ splice site, a single conserved adenine at variable distance to 3’ splice site ($\approx -30$), a splicing complex binds to it, pyrimidine (C,T) rich in human
Transcription start site: Transcription from DNA to RNA by RNA polymerase starts here facilitated by promoter elements. Promoter elements are diverse and their profiles tend to contain little info:

- diverse transcription factor binding sites at very variable positions
- sometimes TATA-box
- “CpG islands”
Transcription termination site (TTS):

- cleavage of the transcript.
- some non-templated A’s are appended (polyadenylation).
- polyadenylation is triggered in many species in many genes by the hexamer aataaa roughly 15 bp upstream of the TTS.
Signals

transcription start site

translation start site

donor (5') splice site

acceptor (3') splice site

translation termination site

donor (5') splice site

branch point region

acceptor (3') splice site

Start and stop codon:

- start codon: ATG
- stop codons: TAA, TAG, TGA

In some species the genetic code is altered and a “stop codon” is actually coding for an amino acid.
Nucleotide Composition of Coding and Noncoding Regions

Sequence Content

Besides the signals, position-unspecific frequencies of nucleotide patterns can be used to guess biological classification (e.g. CDS, non-coding, CpG-island) of longer sequence intervals.

Example (GC content in red flour beetle)

Typically, higher order patterns are examined:
E.g. reading-frame dependent $k$-mer frequencies ($k = 5, 6$) for protein-coding regions.

Remark

Sequence content is usually only indirect evidence.
Problems and General Ansatz

<table>
<thead>
<tr>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>• known signal models do not carry much information</td>
</tr>
</tbody>
</table>

Problems

- known signal models do not carry much information

Ansatz

- combine all individual weak info to boost discriminatory power
- enforce standard gene structure:
  - reading frame consistency between exons
  - minimal splice site consensus (GT/AG, maybe GC/AG)
  - no in-frame stop codons
  - minimal intron length (≈ 40 bp)
Problems and General Ansatz

Problems

- known signal models do not carry much information
- false positive signals because of low number of true positives

Ansatz

- combine all individual weak info to boost discriminatory power
- enforce standard gene structure:
  - reading frame consistency between exons
  - minimal splice site consensus (GT/AG, maybe GC/AG)
  - no in-frame stop codons
  - minimal intron length ($\approx 40$ bp)
Problems and General Ansatz

Problems

- known signal models do not carry much information
- false positive signals because of low number of true positives
- sequence content can be misleading (pseudogenes, repeats)

Ansatz

- combine all individual weak info to boost discriminatory power
- enforce standard gene structure:
  - reading frame consistency between exons
  - minimal splice site consensus (GT/AG, maybe GC/AG)
  - no in-frame stop codons
  - minimal intron length ($\approx 40$ bp)
Problems and General Ansatz

Problems

- known signal models do not carry much information
- false positive signals because of low number of true positives
- sequence content can be misleading (pseudogenes, repeats)

Ansatz

- combine all individual weak info to boost discriminatory power
Problems and General Ansatz

Problems

- known signal models do not carry much information
- false positive signals because of low number of true positives
- sequence content can be misleading (pseudogenes, repeats)

Ansatz

- combine all individual weak info to boost discriminatory power
- enforce standard gene structure:
  - reading frame consistency between exons
  - minimal splice site consensus (GT/AG, maybe GC/AG)
  - no in-frame stop codons
  - minimal intron length ($\approx 40$ bp)
The One-Dimensional Chaining Problem

Definition

Let \( \mathcal{B} = \{B_1, B_2, \ldots, B_n\} \) be a set of intervals with boundaries given by \( B_j = [\ell_j, r_j) \) and \( \ell_j < r_j, \ (j = 1, \ldots, n) \). Let \( s_j \in \mathbb{R} \) be the score of interval \( B_j \).
The One-Dimensional Chaining Problem

**Definition**

Let \( B = \{B_1, B_2, \ldots, B_n\} \) be a set of intervals with boundaries given by \( B_j = [\ell_j, r_j) \) and \( \ell_j < r_j, (j = 1, \ldots, n) \).

Let \( s_j \in \mathbb{R} \) be the score of interval \( B_j \).

A **chain** \( \Gamma = (B_{j_1}, B_{j_2}, \ldots, B_{j_d}) \) is a sorted sequence of non-overlapping intervals (i.e. \( r_{j_i} \leq \ell_{j_{i+1}} \)).
The One-Dimensional Chaining Problem

**Definition**

Let \( \mathcal{B} = \{ B_1, B_2, \ldots, B_n \} \) be a set of intervals with boundaries given by \( B_j = [\ell_j, r_j) \) and \( \ell_j < r_j, (j = 1, \ldots, n) \). Let \( s_j \in \mathbb{R} \) be the score of interval \( B_j \).

A chain \( \Gamma = (B_{j_1}, B_{j_2}, \ldots, B_{j_d}) \) is a sorted sequence of non-overlapping intervals (i.e. \( r_{j_i} \leq \ell_{j_{i+1}} \)).

The score of a chain is the sum of the scores of its intervals:
\[
s(\Gamma) = \sum_{i}^{d} s_{j_i}
\]
The One-Dimensional Chaining Problem

**Definition**

Let $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ be a set of intervals with boundaries given by $B_j = [\ell_j, r_j]$ and $\ell_j < r_j$, ($j = 1, \ldots, n$).

Let $s_j \in \mathbb{R}$ be the score of interval $B_j$.

A chain $\Gamma = (B_{j_1}, B_{j_2}, \ldots, B_{j_d})$ is a sorted sequence of non-overlapping intervals (i.e. $r_{j_i} \leq \ell_{j_{i+1}}$).

The score of a chain is the sum of the scores of its intervals: $s(\Gamma) = \sum_{i}^{d} s_{j_i}$

**Definition (One-dimensional Chaining Problem)**

For a given set of scored intervals $\mathcal{B}$ find a chain with maximal score.
Example Chaining Problem

Example

\[ B_1 = [0, 1), s_1 = 1 \]
\[ B_2 = [0, 3), s_2 = 2 \]
\[ B_3 = [2, 4), s_3 = 2 \]
\[ B_4 = [2, 6), s_4 = 2 \]
\[ B_5 = [5, 8), s_5 = 3 \]
\[ B_6 = [7, 8), s_6 = 2 \]
\[ B = \{ B_1, \ldots, B_6 \} \]

\[ \Gamma = (B_1, B_3, B_5) \] is the chain with maximal score.
Example Chaining Problem

Example

\[ B_1 = [0, 1), s_1 = 1 \]
\[ B_2 = [0, 3), s_2 = 2 \]
\[ B_3 = [2, 4), s_3 = 2 \]
\[ B_4 = [2, 6), s_4 = 2 \]
\[ B_5 = [5, 8), s_5 = 3 \]
\[ B_6 = [7, 8), s_6 = 2 \]
\[ \mathcal{B} = \{B_1, \ldots, B_6\} \]

\[ \Gamma = (B_1, B_3, B_5) \text{ is the chain with maximal score.} \]
How to Solve the Chaining Problem?

- **brute force** too slow: There are $2^n$ possible chains.
How to Solve the Chaining Problem?

- **brute force** too slow: There are $2^n$ possible chains.
- **greedy** approach does not correctly solve the problem:

$$\Gamma \leftarrow ()$$

```markdown
repeat
    insert highest-scoring interval into $\Gamma$ that does not overlap any interval already in $\Gamma$
until no more interval can be inserted
```
How to Solve the Chaining Problem?

- **brute force** too slow: There are $2^n$ possible chains.
- **greedy** approach does not correctly solve the problem:

\[
\Gamma \leftarrow ()
\]

**repeat**

insert highest-scoring interval into $\Gamma$ that does not overlap any interval already in $\Gamma$

**until** no more interval can be inserted

trivial counterexample:

```
\[ B_1 \quad 2 \quad B_2 \quad 3 \quad B_3 \quad 2 \]
```
Chaining Algorithm

One-Dimensional Chaining Algorithm

1: \( P \leftarrow \text{sort} \{\ell_1, r_1, \ell_2, r_2, \ldots, \ell_n, r_n\} \) increasingly
2: \( S \leftarrow q \leftarrow q_1 \leftarrow \cdots \leftarrow q_n \leftarrow S_1 \leftarrow \cdots \leftarrow S_n \leftarrow 0 \)
3: \( \text{while } P \text{ not empty do} \)
4: \( b \leftarrow \text{remove smallest element in } P \)
5: \( \text{for all } j \text{ such that } r_j = b \text{ do} \)
6: \( \text{if } S_j > S \text{ then} \)
7: \( S \leftarrow S_j \)
8: \( q \leftarrow j \)
9: \( \text{end if} \)
10: \( \text{end for} \)
11: \( \text{for all } j \text{ such that } \ell_j = b \text{ do} \)
12: \( S_j \leftarrow s_j + S \)
13: \( q_j \leftarrow q \)
14: \( \text{end for} \)
15: \( \text{end while} \)
16: \( \text{output } S \text{ as score of best chain} \)
Chaining Algorithm

17: \( \Gamma \leftarrow () \)
18: \textbf{while} \( q \neq 0 \) \textbf{do}
19: \hspace{1em} \text{push} \( B_q \) onto \( \Gamma \)
20: \hspace{1em} \text{q} \leftarrow q
21: \textbf{end while}
22: \text{reverse order of} \( \Gamma \)
23: \text{output} \( \Gamma \) as highest-scoring chain

Backtracking
## Correctness

### Invariants of the Algorithm

1. After very iteration of the main loop in line 3, \( S \) is the score of the best chain without interval boundaries beyond \( b \).
2. After every iteration of the main loop in line 3, \( S_j \) is the score of the best chain, that ends with interval \( B_j \) for all \( j \) with \( \ell_j \leq b \).

Proof by induction on the iteration of the main loop in line 3. It follows that after the last iteration \( S \) is the score of the overall best chain.

### Pointers for Backtracking

Unless undefined \( (q_j = 0) \), \( q_j \) is the index of the interval immediately left of \( B_j \) in a best chain that contains \( B_j \).
Example Algorithm Run

Example

After initialization (line 2):
\[ P = (0, 1, 2, 3, 4, 5, 6, 7, 8) \]
\[ S = 0 \]
\[ q = 0 \]
Example Algorithm Run

Example

After 1st iteration of main loop (line 3):

$S = 0$
$q = 0$

$b = 0$
Example Algorithm Run

Example

After 2nd iteration of main loop (line 3):

\( S = 1 \)
\( q = 1 \)

\[
\begin{align*}
S_1 &= 1, \quad q_1 = 0 \\
B_1, s_1 &= 1 \\
S_2 &= 2, \quad q_2 = 0 \\
B_1, s_2 &= 2 \\
B_3, s_3 &= 2 \\
B_4, s_4 &= 2 \\
B_5, s_5 &= 3 \\
B_6, s_6 &= 2 \\
b &= 1
\end{align*}
\]
Example Algorithm Run

Example

After 3rd iteration of main loop (line 3):
\( S = 1 \)
\( q = 1 \)

0 1 2 3 4 5 6 7 8

\( b = 2 \)
Example Algorithm Run

Example

After 4th iteration of main loop (line 3):

\[ S = 2 \]
\[ q = 2 \]

\[ b = 3 \]
Example Algorithm Run

Example

After 5th iteration of main loop (line 3):

\[ S = 3 \]
\[ q = 3 \]
Example Algorithm Run

Example

After 6th iteration of main loop (line 3):

\[ S = 3 \]
\[ q = 3 \]

```
B_1, s_1 = 1
B_2, s_2 = 2
B_3, s_3 = 2
B_4, s_4 = 2
B_5, s_5 = 3
```

\[ b = 5 \]
Example Algorithm Run

Example

After 7th iteration of main loop (line 3):

\[ S = 3 \]
\[ q = 3 \]

\( b = 6 \)
Example Algorithm Run

After 8th iteration of main loop (line 3):

\( S = 3 \)
\( q = 3 \)

Example

\( S_1 = 1, q_1 = 0 \)
\( S_2 = 2, q_2 = 0 \)
\( S_3 = 3, q_3 = 1 \)
\( S_4 = 3, q_4 = 1 \)
\( S_5 = 6, q_5 = 3 \)
\( S_6 = 5, q_6 = 3 \)

\( B_1, s_1 = 1 \)
\( B_2, s_2 = 2 \)
\( B_3, s_3 = 2 \)
\( B_4, s_4 = 2 \)
\( B_5, s_5 = 3 \)
\( B_6, s_6 = 2 \)

\( b = 7 \)
Example Algorithm Run

**Example**

After last iteration of main loop (line 3):

- $S = 6$
- $q = 5$

<table>
<thead>
<tr>
<th>i</th>
<th>$B_i,s_i$</th>
<th>$S_i$</th>
<th>$q_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$(1, 2)$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$(2, 0)$</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$(3, 1)$</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$(4, 1)$</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>$(5, 3)$</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>$(6, 3)$</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>
Example Algorithm Run

Example

Backtracking:
Follow \( q_j \) pointers starting from \( q = 5 \) until \( q = 0 \).
\[
\Gamma = (B_1, B_3, B_5)
\]

\[
\begin{align*}
S_1 &= 1, q_1 = 0 \quad &B_1, s_2 &= 2 \\
B_1, s_1 &= 1 \quad &S_2 &= 2, q_2 = 0 \quad &S_4 &= 3, q_4 = 1 \\
B_3, s_3 &= 2 \quad &B_4, s_4 &= 2 \quad &S_5 &= 6, q_5 = 3 \\
S_3 &= 3, q_3 = 1 \quad &B_6, s_6 &= 2 \quad &S_6 &= 5, q_6 = 3
\end{align*}
\]
Running Time

Sorting of interval boundaries (line 1):

1.16  Running Time

Sorting of interval boundaries (line 1):

Overall time in main loop (lines 3-15):

Backtracking:

Overall running time:

Remarks:

• The linear running time of the main loop can be realized when for each interval boundary in \( P \) a list of intervals ending and starting at \( b \) is stored. For each interval the loops 5-10 and 11-14 are then executed exactly once each (amortized analysis).

• Special but important case: the intervals have integers as boundaries (sequence positions) in the range 1..\( t \) ⇒ sorting can be done in \( O(t + n \log n) \) using Bucket Sort ⇒ faster if \( t = o(n \log n) \) (dense intervals)
## Running Time

### Running Time

<table>
<thead>
<tr>
<th>Sorting of interval boundaries (line 1): $O(n \log n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall time in main loop (lines 3-15):</td>
</tr>
</tbody>
</table>

Remarks:
- The linear running time of the main loop can be realized when for each interval boundary in $P$ a list of intervals ending and starting at $b$ is stored. For each interval the loops 5-10 and 11-14 are then executed exactly once each (amortized analysis).
- Special but important case: the intervals have integers as boundaries (sequence positions) in the range $1 \ldots t$ ⇒ sorting can be done in $O(t + n)$ using Bucket Sort ⇒ faster if $t = o(n \log n)$ (dense intervals)
Running Time

Sorting of interval boundaries (line 1): $O(n \log n)$
Overall time in main loop (lines 3-15): $O(n)$
Backtracking:
## Running Time

<table>
<thead>
<tr>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorting of interval boundaries (line 1): $O(n \log n)$</td>
</tr>
<tr>
<td>Overall time in main loop (lines 3-15): $O(n)$</td>
</tr>
<tr>
<td>Backtracking: $O(n)$</td>
</tr>
</tbody>
</table>

**Overall running time:**

- The linear running time of the main loop can be realized when for each interval boundary in $P$ a list of intervals ending and starting at $b$ is stored. For each interval the loops 5-10 and 11-14 are then executed exactly once each (amortized analysis).
- Special but important case: the intervals have integers as boundaries (sequence positions) in the range 1..$t$ ⇒ sorting can be done in $O(t + n \log n)$ using Bucket Sort ⇒ faster if $t = o(n \log n)$ (dense intervals).
### Running Time

<table>
<thead>
<tr>
<th>Step</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorting of interval boundaries (line 1)</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Overall time in main loop (lines 3-15)</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Backtracking</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>

**Overall running time:** $O(n \log n)$
Running Time

Sorting of interval boundaries (line 1): $O(n \log n)$
Overall time in main loop (lines 3-15): $O(n)$
Backtracking: $O(n)$

**Overall running time:** $O(n \log n)$

Remarks:

- The linear running time of the main loop can be realized when for each interval boundary in $P$ a list of intervals ending and starting at $b$ is stored. For each interval the loops 5-10 and 11-14 are then executed exactly once each (amortized analysis).
Running Time

Running Time

Sorting of interval boundaries (line 1): $O(n \log n)$
Overall time in main loop (lines 3-15): $O(n)$
Backtracking: $O(n)$
**Overall running time:** $O(n \log n)$

Remarks:

- The linear running time of the main loop can be realized when for each interval boundary in $P$ a list of intervals ending and starting at $b$ is stored. For each interval the loops 5-10 and 11-14 are then executed exactly once each (amortized analysis).

- Special but important case: the intervals have integers as boundaries (sequence positions) in the range $1..t$
  $\Rightarrow$ sorting can be done in $O(t + n)$ using Bucket Sort
  $\Rightarrow$ faster if $t = o(n \log n)$ (dense intervals)
Simple Approach to Gene Finding

• only predict protein-coding part of genes (easier)
Simple Approach to Gene Finding

- only predict protein-coding part of genes (easier)
- interpret gene structure as chain of CDS
Simple Approach to Gene Finding

- only predict protein-coding part of genes (easier)
- interpret gene structure as chain of CDS
- gene boundaries are implied by CDS boundaries (stop codon)
Simple Approach to Gene Finding

- only predict protein-coding part of genes (easier)
- interpret gene structure as chain of CDS
- gene boundaries are implied by CDS boundaries (stop codon)
- CDS candidate defined by sequence (integer) interval $B_j = [\ell_j, r_j]

score $j$-th CDS candidate:

$$s_j = \text{score of signal at } \ell_j \quad (\text{e.g. ASS or start codon})$$
$$+ \text{score of signal at } r_j \quad (\text{e.g. DSS or stop codon})$$
$$+ \text{score of sequence content in } [\ell_j, r_j]$$
Simple Approach to Gene Finding

- only predict protein-coding part of genes (easier)
- interpret gene structure as chain of CDS
- gene boundaries are implied by CDS boundaries (stop codon)
- CDS candidate defined by sequence (integer) interval $B_j = [\ell_j, r_j)$

score $j$-th CDS candidate:

$$s_j = \text{score of signal at } \ell_j \text{ (e.g. ASS or start codon)} + \text{score of signal at } r_j \text{ (e.g. DSS or stop codon)} + \text{score of sequence content in } [\ell_j, r_j)$$

- find highest-scoring chain of CDS as gene prediction
Simple Approach to Gene Finding

Signal Score

A number $s$ assigned to a sequence position $p$ that is used to decide whether the signal is present at $p$.

Usually: $s = s(w)$, where $w$ is a sequence window around $p$.

Aims:

1. The larger the score, the more likely is it that there is a true signal.
2. $s(w)$ is “small” for positions $p$ without the signal.
Example Signal Score

Example (DSS position weight matrix)

\( p = \) candidate donor splice site position
\( w = \) seq window 2 pos upstream and 5 pos downstream of DSS

Have position specific scoring matrix for DSS

\[
m(i, b) \quad (i = 1, 2, \ldots, 7, b \in A, C, G, T),
\]

\[
m(i, A) + m(i, C) + m(i, G) + m(i, T) = 1
\]

Have “background” distribution of nucleotides \( q(b) \)

\[
q(A) + q(C) + q(G) + q(T) = 1
\]

Define log-odds score: \( s = \log \prod_{i=1}^{7} \frac{m(i, w_i)}{q(w_i)} \)
Base composition is frame-dependent

<table>
<thead>
<tr>
<th>f = 0</th>
<th>f = 1</th>
<th>f = 2</th>
<th>all f</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.248</td>
<td>0.291</td>
<td>0.146</td>
</tr>
<tr>
<td>C</td>
<td>0.264</td>
<td>0.243</td>
<td>0.351</td>
</tr>
<tr>
<td>G</td>
<td>0.321</td>
<td>0.201</td>
<td>0.312</td>
</tr>
<tr>
<td>T</td>
<td>0.166</td>
<td>0.265</td>
<td>0.190</td>
</tr>
</tbody>
</table>
**Example Content Score**

**Example (frame-dependent Markov chain of order $k$)**

Let $w$ be the DNA word of length $n$ to be scored as CDS. Let $f \in \{0, 1, 2\}$ be the frame of the first position of $w$.

$$P(w) := p_f(w_1, \ldots, w_k) \cdot \prod_{i=k+1}^{n} p_f(i)(w_i | w_{i-k}, \ldots, w_{i-1})$$

- $p_f$ is a start probability for the first $k$ bases

Here:

- $f(i) \in \{0, 1, 2\}$ such that $f(i) \equiv f - 1 + i \mod 3$ is the frame of the $i$-th position of $w$

Define $s(w) = \log(P(w)/Q(w))$, where $Q(w)$ is the probability of $w$ in a “background” model (e.g. non-coding).

**Remark:** division by background $\Rightarrow$ good exon candidates get positive score
Example Content Score - Continued

**Example**

\[ w = \text{ATTCTGC} \]
frame \( f = 2 \), i.e. with these codon breaks: \( \text{A}||\text{TTC}||\text{TGC} \)
\( k = 2 \)

\[
P(\text{ATTCTGC}) = p_2(\text{AT})p_1(\text{T|AT})p_2(\text{C|TT})
\]
\[
p_0(\text{T|TC})p_1(\text{G|CT})p_2(\text{C|TG})
\]

- if \( k \geq 2 \) above content model can reflect codon usage
- typical: \( k = 4 \) or \( k = 5 \)
- probabilities \( p_r(x \mid y_1, \ldots, y_k) \) can be estimated on known coding sequences
Problems with Simple Approach

- reading frame consistency not enforced
Problems with Simple Approach

- reading frame consistency not enforced
- $\Rightarrow$ output can be biologically “senseless”
Problems with Simple Approach

- reading frame consistency not enforced
- ⇒ output can be biologically “senseless”
- ⇒ less accurate when this info is ignored
Problems with Simple Approach

- reading frame consistency not enforced
- ⇒ output can be biologically “senseless”
- ⇒ less accurate when this info is ignored
- CDS candidates with negative score are never used
Problems with Simple Approach

- reading frame consistency not enforced
- ⇒ output can be biologically “senseless”
- ⇒ less accurate when this info is ignored
- CDS candidates with negative score are never used

Need extension to chaining algorithm to enforce consistency.
### Example (exon candidates in a DNA of length 2000)

- color at left and right end (red, green, blue)
  - specify exon phase at left and right end
- arrow tips and heads denote start and stop codons

Exon candidates of the program GENEID
Consistent Exon Chain

Extension of Algorithm

It is possible to extend the chaining algorithm so that it only considers **consistent chains**:
Every two consecutive exons in a chain must be (frame-)compatible.

Example

\[
\begin{align*}
\text{suc}(1) &= f_0^+ = \text{pre}(2) \\
\text{suc}(2) &= f_2^+ = \text{pre}(3) \\
\text{suc}(3) &= \text{boundary} = \text{pre}(4) \\
\text{suc}(4) &= f_2^- = \text{pre}(5)
\end{align*}
\]

\[
\begin{array}{ccc}
\text{ATG} & \cdots & \text{***} \\
B_1 & & \\
\text{***} & \cdots & \text{**} \\
B_2 & & \\
\text{* ***} & \cdots & \text{TAG} \\
B_3 & & \\
\text{CTA} & \cdots & \text{****} \\
B_4 & & \\
\text{**} & \cdots & \text{CAT} \\
B_5 & &
\end{array}
\]
Issues of the Exon Chaining Approach

Problematic:

- **introns** are not modelled at all:
  - no length distribution considered
  - no difference to intergenic region
## Issues of the Exon Chaining Approach

### Problematic:

- **introns** are not modelled at all:
  - no length distribution considered
  - no difference to intergenic region
- **UTRs**: How can one accommodate for exons like these?

| UTR | CDS |
### Issues of the Exon Chaining Approach

#### Problematic:

- **introns** are not modelled at all:
  - no length distribution considered
  - no difference to intergenic region

- **UTRs**: How can one accommodate for exons like these?

- dividing by *background* probability implicitly assumes that there are only two alternatives, e.g. exon ↔ noncoding but there are **more than two alternatives** for a region
Reminder: Hidden Markov Model

- A HMM is a probabilistic model of a word \( y = y_1 y_2 \cdots y_n \) ("emission") over some alphabet \( \Sigma \) and of a state sequence \( x = (x_1, x_2, \cdots, x_n) \) over some discrete set of states \( Q \).
A HMM is a probabilistic model of a word \( y = y_1 y_2 \cdots y_n \) ("emission") over some alphabet \( \Sigma \) and of a state sequence \( x = (x_1, x_2, \cdots, x_n) \) over some discrete set of states \( Q \).

The joint distribution of \( x \) and \( y \) is of the form

\[
P(x, y) = \prod_{i=1}^{n} p(x_i|x_{i-1}) \cdot p(y_i|x_i),
\]

where the \( p(x_i|x_{i-1}) \) are the transition probabilities of a Markov chain and the \( p(y_i|x_i) \) are all emission probabilities.

\((x_0 \text{ is a start state to simplify notation})\)
Reminder: Hidden Markov Model

Algorithms

- In applications, normally $y$ is observed and $x$ is unobserved/hidden.
Reminder: Hidden Markov Model

Algorithms

- In applications, normally $y$ is observed and $x$ is unobserved/hidden.
- The Viterbi algorithm computes a most likely state sequence $\hat{x} \in \text{arg max}_x P(x|y)$ in time $O(n)$. 

Reminder: Hidden Markov Model

Algorithms

- In applications, normally $y$ is observed and $x$ is unobserved/hidden.
- The Viterbi algorithm computes a most likely state sequence $\hat{x} \in \arg \max_x P(x|y)$ in time $O(n)$.
- The Forward algorithm can be used to compute $P(x, y)$ in time $O(n)$. 


**Reminder: Hidden Markov Model**

**Algorithms**

- In applications, normally $y$ is observed and $x$ is unobserved/hidden.
- The *Viterbi algorithm* computes a most likely state sequence $\hat{x} \in \arg \max_x P(x|y)$ in time $O(n)$.
- The *Forward algorithm* can be used to compute $P(x, y)$ in time $O(n)$.
- The *Forward and Backward algorithms* can be used to compute posterior probabilities $P(x_i = q|y)$ in time $O(n)$. 

Reminder: Hidden Markov Model

**Algorithms**

- In applications, normally $y$ is observed and $x$ is unobserved/hidden.
- The *Viterbi algorithm* computes a most likely state sequence $\hat{x} \in \arg \max_x P(x|y)$ in time $O(n)$.
- The *Forward algorithm* can be used to compute $P(x, y)$ in time $O(n)$.
- The *Forward and Backward algorithms* can be used to compute posterior probabilities $P(x_i = q|y)$ in time $O(n)$. 


Hidden Markov Model for Eukaryotic Gene Prediction

Example

Genome analysis
Mario Stanke

Introduction to Gene-Finding-Problem
What Do Genes Look Like?
Statistical Features of Genes
The One-Dimensional Chaining Problem
Simple Approach to Gene Finding

Gene Finding with HMMs
Generalized HMMs
Model Design
Training
Why GHMMs?

- A HMM is a **special case of a GHMM.**
Why GHMMs?

- A HMM is a special case of a GHMM.
- In gene finding and for alignment tasks GHMMs are often used because...
Reminder: Generalized Hidden Markov Model

Why GHMMs?

- A HMM is a **special case of a GHMM**.
- In gene finding and for alignment tasks, GHMMs are often used because they allow a detailed **modelling of the length** distribution of exons and other biological intervals.
Reminder: Generalized Hidden Markov Model

Why GHMMs?

- A HMM is a special case of a GHMM.
- In gene finding and for alignment tasks, GHMMs are often used because
  1. they allow a detailed modelling of the length distribution of exons and other biological intervals
  2. they accommodate for “silent” or “delete” states required to model alignment gaps
Generalized Hidden Markov Model

**Definition (Parse)**

Let $y = y_1 y_2 \cdots y_n$, $\Sigma$, $Q$ be as before. A parse $x$ of $y$ is a sequence

$$x = ((q_1, v_1), (q_2, v_2), \ldots, (q_t, v_t)),$$

with $q_i \in Q$, $v_i \in \mathbb{N}_0$ such that $v_1 \leq v_2 \leq \cdots \leq v_t = n$. 
**Generalized Hidden Markov Model**

**Definition (Parse)**

Let $y = y_1y_2 \cdots y_n$, $\Sigma$, $Q$ be as before. A parse $x$ of $y$ is a sequence

$$x = ((q_1, v_1), (q_2, v_2), \ldots, (q_t, v_t)),$$

with $q_i \in Q$, $v_i \in \mathbb{N}_0$ such that $v_1 \leq v_2 \leq \cdots \leq v_t = n$. 

\[ y_1 | y_2 | y_3 | \cdots | y_{v_1} | \cdots | y(v_{i-1}, v_i) | \cdots | y_n \]

\[ \downarrow \quad d_i \quad \uparrow \]
**Generalized Hidden Markov Model**

**Definition (Parse)**

Let \( y = y_1 y_2 \cdots y_n, \Sigma, Q \) be as before.

A parse \( x \) of \( y \) is a sequence

\[
  x = ((q_1, v_1), (q_2, v_2), \ldots, (q_t, v_t)),
\]

with \( q_i \in Q, v_i \in \mathbb{N}_0 \) such that \( v_1 \leq v_2 \leq \cdots \leq v_t = n \).

\[
\begin{array}{cccccccccc}
  v_0 = 0 & q_1 & v_1 & q_2 & v_2 & v_{i-1} & q_i & v_i & v_{t-1} & q_t & v_t \\
  y_1 y_2 y_3 \cdots y_{v_i} & \cdots & y(v_{i-1}, v_i] & \cdots & y_n \\
\end{array}
\]

- observe that \( y \) decomposes via \( x \) into
  \[
  y = y(v_0, v_1] y(v_1, v_2] \cdots y(v_{n-1}, v_n] \quad (v_0 := 0)
  \]
Generalized Hidden Markov Model

Definition (Parse)

Let \( y = y_1 y_2 \cdots y_n, \Sigma, Q \) be as before.

A parse \( x \) of \( y \) is a sequence

\[
x = ((q_1, v_1), (q_2, v_2), \ldots, (q_t, v_t)),
\]

with \( q_i \in Q, v_i \in \mathbb{N}_0 \) such that \( v_1 \leq v_2 \leq \cdots \leq v_t = n \).

- observe that \( y \) decomposes via \( x \) into
  \[
y = y(v_0, v_1]y(v_1, v_2] \cdots y(v_{n-1}, v_n] \quad (v_0 := 0)
  \]
- we say that state "\( q_i \) ends at \( v_i \)"
Generalized Hidden Markov Model

Definition (Parse)

Let $y = y_1 y_2 \cdots y_n$, $\Sigma$, $Q$ be as before.

A parse $x$ of $y$ is a sequence

$$x = ((q_1, v_1), (q_2, v_2), \ldots, (q_t, v_t)),$$

with $q_i \in Q$, $v_i \in \mathbb{N}_0$ such that $v_1 \leq v_2 \leq \cdots \leq v_t = n$.

- observe that $y$ decomposes via $x$ into
  $$y = y(v_0, v_1)y(v_1, v_2)\cdots y(v_{n-1}, v_n) \quad (v_0 := 0)$$
- we say that state "$q_i$ ends at $v_i$"
- we call $d_i := v_i - v_{i-1}$ the length of the $i$-th emission
Generalized Hidden Markov Model (Semi-HMM)

Definition (GHMM)

A GHMM is a joint distribution of a word \( y \) and a parse \( x \) of \( y \) of the form

\[
P(x, y) = \prod_{i=1}^{t} P_{\text{trans}}(q_i | q_{i-1}) \cdot P_{\text{emi}}(y(v_{i-1}, v_i) | q_i),
\]

where \( P_{\text{trans}}(\cdot | q) \) is a probability distribution (transition probabilities) over \( Q \) for all \( q \in Q \) and where \( P_{\text{emi}}(\cdot | q) \) is a probability distribution (emission probabilities) over \( \Sigma^* \) for all \( q \in Q \).

\( q_0 \) is a special start state

\( \Sigma^* = \{ \text{all strings with letters in } \Sigma \} \) (includes empty string)

Remark: We explicitly allow \( d_i = 0 \). A state \( q \) with \( P_{\text{emi}}(\epsilon | q) = 1 \) is called a silent state (\( \epsilon \) is the empty string of length 0).
When is a GHMM called a HMM?

- A HMM is a GHMM in which $d_i \equiv 1$ for all $i$, i.e. all emissions are a single character. In that special case the parse $x$ can be identified with the state sequence, which has the same length as $y$.

- Sometimes in the literature a GHMM, in which $d_i \in \{0, 1\}$, is still called a HMM only with some special modifications to the algorithms. Example: “delete” state in profile HMMs.
Algorithms for GHMM

### Algorithms

1. Usually, the word $y$ is observed. Now: A *concatenation* of the emissions, not the sequence of emissions. Contrast to HMM: The emissions cannot be inferred from $y$ alone.
Algorithms for GHMM

1. Usually, the word $y$ is observed. Now: A concatenation of the emissions, not the sequence of emissions. Contrast to HMM: The emissions cannot be inferred from $y$ alone.

2. $x$ is unobserved, neither the states nor their boundaries are known.
Algorithms for GHMM

### Algorithms

1. Usually, the word $y$ is observed. Now: A *concatenation* of the emissions, not the sequence of emissions.
   Contrast to HMM: The emissions cannot be inferred from $y$ alone.

2. $x$ is unobserved, neither the states nor their boundaries are known.

3. Analogous Viterbi, Forward and Backward algorithms exist that all run in $O(n^2)$. Important special case: they run in $O(n)$ if all $d_i$ are bounded from above by a constant.
Algorithms for GHMM

Algorithms

1. Usually, the word $y$ is observed. Now: A concatenation of the emissions, not the sequence of emissions. Contrast to HMM: The emissions cannot be inferred from $y$ alone.

2. $x$ is unobserved, neither the states nor their boundaries are known.

3. Analogous Viterbi, Forward and Backward algorithms exist that all run in $O(n^2)$. Important special case: they run in $O(n)$ if all $d_i$ are bounded from above by a constant.

4. A prerequisite for point 3 above is that no loops of states with just empty-word-emissions are possible. We will ensure that by the design of the model topology.
A Simple GHMM for Gene Finding: Model Topology

Model for (multiple) eukaryotic genes on forward strand:

(Arrows denote the transitions with non-zero transition probability.)
What (Most) Eukaryotic Species Have in Common?

### In Common:

- same genetic code, including start and stop codons
What (Most) Eukaryotic Species Have in Common?

**In Common:**

- same genetic code, including start and stop codons
- genes can have introns, may have many

In Common:

- same genetic code, including start and stop codons
- genes can have introns, may have many
What (Most) Eukaryotic Species Have in Common?

In Common:

- same genetic code, including start and stop codons
- genes can have introns, may have many
- genes rarely overlap in sequence
What (Most) Eukaryotic Species Have in Common?

In Common:

- same genetic code, including start and stop codons
- genes can have introns, may have many
- genes rarely overlap in sequence
- introns start almost always with GT, end with AG (some introns GC/AG)
What (Most) Eukaryotic Species Have in Common?

In Common:

- same genetic code, including start and stop codons
- genes can have introns, may have many
- genes rarely overlap in sequence
- introns start almost always with GT, end with AG (some introns GC/AG)
- more non-coding sequence than coding sequence
How Species-Specific Must Gene Finding Models Be?

Differences:

- distribution at signals, e.g. branch point region

  top: human / bottom: fly
How Species-Specific Must Gene Finding Models Be?

Differences:

- distribution at signals, e.g. branch point region
- GC content highly variable
How Species-Specific Must Gene Finding Models Be?

Differences:

- distribution at signals, e.g. branch point region
- GC content highly variable
- number and length distribution of introns

**top: human / bottom: C. elegans**
How Species-Specific Must Gene Finding Models Be?

**Differences:**

- distribution at signals, e.g. branch point region
- GC content highly variable
- number and length distribution of introns
- length distribution of UTRs
How Species-Specific Must Gene Finding Models Be?

**Differences:**

- distribution at signals, e.g. branch point region
- GC content highly variable
- number and length distribution of introns
- length distribution of UTRs
- gene density
### Training: Estimate Species-Specific Parameters

#### “Training Set”

- input: set of annotated sequences
  \[
  (x^{(k)}, y^{(k)})_{k=1,\ldots,N},
  \]
  such that the parse \( x^{(k)} \) represents the gene structure of DNA sequence \( y^{(k)} \).
- frequently a few hundred genes constructed from cDNA alignments

---

**Introduction to Gene-Finding-Problem**

- What Do Genes Look Like?
- Statistical Features of Genes
- The One-Dimensional Chaining Problem
- Simple Approach to Gene Finding

**Gene Finding with HMMs**

- Generalized HMMs
- Model Design
- Training