Fragment Assembly and Shortest Common Superstrings

Course “Discrete Biological Models” (Modelli Biologici Discreti)

Zsuzsanna Lipták

Laurea Triennale in Bioinformatica
a.a. 2014/15, fall term

Shotgun sequencing of the human genome

From the DNA molecules (input of experiment) we want to get the sequence of the nucleotides (desired output).

$\Rightarrow$

```
...AACAGTACCATGCTAGGTCAATCGA...
...TTGTCATGGTACGATCCAGTTAGCT...
```

These slides are mainly based on the Setubal-Meidanis book, chapter 4.

Recall some molecular biology

```
5’ ...AACAGTACCATGCTAGGTCAATCGA...3’
3’ ...TTGTCATGGTACGATCCAGTTAGCT...5’
```

- 4 characters: `A C T G` (bases, nucleotides)
- double stranded
- `A` – `T` and `C` – `G` complementary (Watson-Crick pairs)
- length measured in bp (base pairs)
- orientation (read from 5’ to 3’ end)
- reverse complementary: $(\texttt{ACCTG})^{rc} = \texttt{CAGGT}$

Sanger sequencing technology

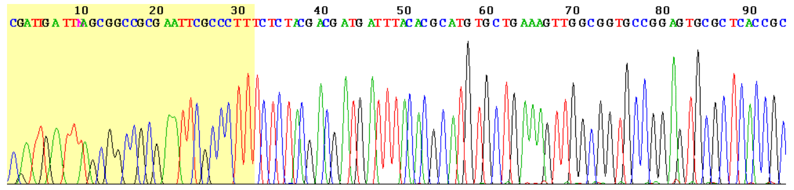


image source: Wikimedia commons

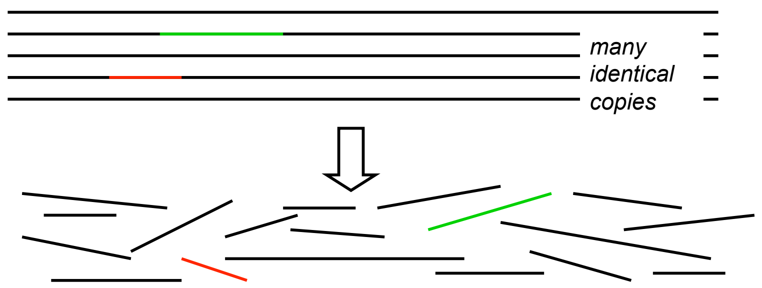
DNA sequences of length 300-1000 bp are sequenced via

- DNA amplification using PCR or vectors
- division of sample into 4 different sub-samples
- chain-termination using modified nucleotides (different one for each reaction)
- radioactive or fluorescent labeling
- gel electrophoresis

See Wikipedia article on ”Sanger sequencing” or Setubal-Meidanis 1.5.2 and 1.6.

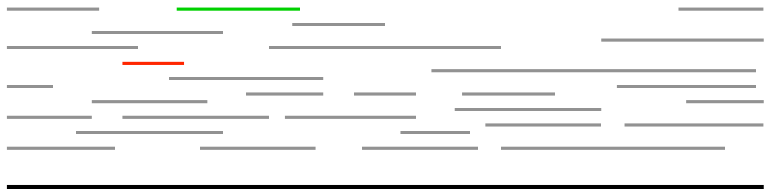
Shotgun sequencing

Typical DNA molecules are several 100 000 bp long, but only sequences of a few hundred ($\sim 300 - 1000$) bp can be sequenced. Solution: We make many identical copies, break them up in random places ("shotgun method") and sequence these shorter fragments.



The fragment assembly problem

Input:
Many short sequences/strings[1] (the fragments).



Goal:
Reconstruct original string (the target string).

[1]Recall that string = sequence, but substring $\neq$ subsequence.

An example

Given the four input strings on the left, one possible way of assembling them is shown on the right. This is called a *layout* (= a multiple alignment of the fragments).

```
ACCGT          --ACCGT--
CGTGC          ----CGTGC
TTAC           TTAC-----
TACCGT         -TACCGT--
               ---------
               TTACCGTGC
```

The sequence under the line (in blue) is called a *consensus sequence*. We'll see later why.

A different example

Here are two different consensus sequences for the same set of input strings.

```
TACC
ACTAC
CGGACT
ACGGA
```

A different example

Here are two different consensus sequences for the same set of input strings.

```
TACC        TACC----------
ACTAC       ------ACTAC---
CGGACT      ---CGGACT-----
ACGGA       ---------ACGGA
            --------------
            TACCGGACTACGGA
```

A different example

Here are two different consensus sequences for the same set of input strings.

```
TACC        TACC----------      ------TACC
ACTAC       ------ACTAC---      ----ACTAC-
CGGACT      ---CGGACT-----      -CGGACT---
ACGGA       ---------ACGGA      ACGGA-----
            --------------      ----------
            TACCGGACTACGGA      ACGGACTACC
```

A different example

Here are two different consensus sequences for the same set of input strings.

```
TACC        TACC----------      ------TACC
ACTAC       ------ACTAC---      ----ACTAC-
CGGACT      ---CGGACT-----      -CGGACT---
ACGGA       ---------ACGGA      ACGGA-----
            --------------      ----------
            TACCGGACTACGGA      ACGGACTACC
```

1. Which solution is better?
2. How can we find all solutions?

A different example

Here are two different consensus sequences for the same set of input strings.

```
TACC        TACC----------      ------TACC
ACTAC       ------ACTAC---      ----ACTAC-
CGGACT      ---CGGACT-----      -CGGACT---
ACGGA       ---------ACGGA      ACGGA-----
            --------------      ----------
            TACCGGACTACGGA      ACGGACTACC
```

1. Which solution is better? $\Rightarrow$ models
2. How can we find all solutions? $\Rightarrow$ algorithms

Complications

First we look at some complications:

- base call errors,
- chimeras and contamination,
- unknown orientation,
- repeats, and
- lack of coverage.

Complications 1: Base call errors 1

Sequencing errors (so-called *base call errors*) can be of 3 types: *substitution, insertion, or deletion* of a single base.

```
ACCGT                 --ACCGT--
CGTGC                 ----CGTGC
TTAC                  TTAC-----
TGCCGT                -TGCCGT--
                      ---------
                      TTACCGTGC
```

A substitution (of an A by a G) occurred in the last sequence. Majority vote will still produce the correct consensus sequence.

Majority vote: For every column, put that nucleotide which appears in the majority (absolute or simple) of the rows in the layout.

Complications 1: Base call errors 2

Sequencing errors (so-called *base call errors*) can be of 3 types: *substitution, insertion, or deletion* of a single base.

```
ACCGT                 --ACC-GT--
CAGTGC                ----CAGTGC
TTAC                  TTAC-----
TACCGT                -TACC-GT--
                      ----------
                      TTACC-GTGC
```

An insertion (of an A) occurred in the second sequence. Majority vote will still produce the correct consensus sequence (- in the consensus sequence will be removed).

Complications 1: Base call errors 3

Sequencing errors (so-called *base call errors*) can be of 3 types: *substitution, insertion, or deletion* of a single base.

```
ACCGT                           --ACCGT--
CGTGC                           ----CGTGC
TTAC                            TTAC-----
TACGT     deletion of a C       -TAC-GT--
                                ---------
                                TTACCGTGC
```

A deletion (of a C) occurred in the last sequence. Majority vote will still produce the correct consensus sequence ('-' in the consensus sequence will be removed).

Complications 2: Chimeras and contamination

Chimeras: Two sequences stick together at the $5'$ resp. $3'$ end, during the lab process.

Contamination: DNA of the vector, or of the human handling the samples, ends up in the input.

An example for a chimera:

```
ACCGT        --ACCGT--
CGTGC        ----CGTGC
TTAC         TTAC-----
TACCGT       -TACCGT--
TTATGC       ?????????
             ---------
             TTACCGTGC
             TTA   TGC
```

Note: Layout/consensus sequence/majority voting cannot deal with chimeras or contamination.

Complications 3: Unknown orientation

Any of the input sequences could be a substring of one strand or the other. Since we have decided for one of the strands which we are trying to reconstruct, if the input sequence is a substring of the other strand, then its *reverse complement* will be a substring of our strand.

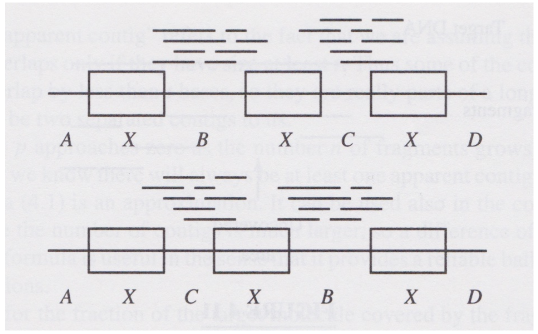
```
CACGT        CACGT--------
ACGT         -ACGT--------
ACTACG       --CGTAGT-----   rc
GTACT        -----AGTAC---   rc
ACTGA        --------ACTGA
CTGA         ---------CTGA
             -------------
             CACGTAGTACTGA
```

There are roughly 2^n many possibilities if we have n input strings.
(In actual fact, less: Why? How many distinct possibilities are there?)

Complications 4: Repeats

It is known that the genome has many repeats: Regions (substrings) which occur more than once. If these are longer than the fragments, then they often lead to ambiguities: It is impossible to decide, based on the input, which is the correct target string, even if we have error-free input strings and an unlimited quantity of them.

Complications 4: Repeats on same strand



The repeated region X is too long; therefore, no fragment covers it completely. The two consensus sequences $AXBXCXD$ or $AXCXBXD$ are equally possible.

Complications 4: Repeats on opposite strands (inverted repeats)

Repeats on opposite strands lead to inverted repeats on the same strand: of the form $AXB(X)^{rc}C$, where $(X)^{rc}$ is the reverse complement of X. We cannot distinguish between the two possible consensus sequences $AXB(X)^{rc}C$ and $AX(B)^{rc}(X)^{rc}C$ (below, the region B is marked in green).

```
----TCGCG--------
-------CGAAGA----
-----------------
ATACTCGCGAAGAGTCC
```

```
--------CGCGA----
----TCTTCG-------
-----------------
ATACTCTTCGCGAGTCC
```

Complications 5: Lack of coverage

```
GTACC----------
GTA------------
--ACC----------
-------ACTAC---
-------ACTA----
----------ACGGA
---------------
GTACCGGACTACGGA
```

The two `G`s in positions 6 and 7 are not covered by any fragment, so we have no information about this stretch. Now the best we can hope for is a good layout for each of the well covered regions, called *contigs* (see later). One way of measuring the quality of a layout is the minimum coverage; another (more common) the mean coverage, taken over all positions of the consensus string.

Quality measures 1: Minimum and mean coverage

```
TACC----------
------ACTAC---
---CGGACT-----
---------ACGGA
--------------
TACCGGACTACGGA
```

```
------TACC
----ACTAC-
-CGGACT---
ACGGA-----
----------
ACGGACTACC
```

Minimum cov. $= 1$
mean cov.: $\frac{20}{14} = 1.42$

Minimum cov. $= 1$
mean cov.: $\frac{18}{10} = 1.8$

Quality measures 2: Linkage

Definition

An overlap of two strings s, t is a string u s.t. u is prefix of s and suffix of t, or u is prefix of t and suffix of s. E.g. the strings `ACGCG` and `GCGTTAC` have three non-empty overlaps:

Quality measures 2: Linkage

Definition

An overlap of two strings s, t is a string u s.t. u is prefix of s and suffix of t, or u is prefix of t and suffix of s. E.g. the strings `ACGCG` and `GCGTTAC` have three non-empty overlaps: `GCG`, `AC`, and `G`.

Given a layout, the linkage is the minimum length of an overlap in the layout which is not contained in any other overlap in the layout. A t-contig is a layout with linkage t.

```
TACC----------
---CGGACT-----
------ACTAC---
---------ACGGA
--------------
TACCGGACTACGGA
```

a 1-contig

Quality measures 2: Linkage

Definition

An overlap of two strings s, t is a string u s.t. u is prefix of s and suffix of t, or u is prefix of t and suffix of s. E.g. the strings `ACGCG` and `GCGTTAC` have three non-empty overlaps:

Given a layout, the linkage is the minimum length of an overlap in the layout which is not contained in any other overlap in the layout. A t-contig is a layout with linkage t.

```
TACC----------        ------TACC
---CGGACT-----        ----ACTAC-
------ACTAC---        -CGGACT---
---------ACGGA        ACGGA-----
--------------        ----------
TACCGGACTACGGA        ACGGACTACC
```

a 1-contig   a 3-contig

Quality measures 2: Linkage

Definition

An overlap of two strings s, t is a string u s.t. u is prefix of s and suffix of t, or u is prefix of t and suffix of s. E.g. the strings `ACGCG` and `GCGTTAC` have three non-empty overlaps:

Given a layout, the linkage is the minimum length of an overlap in the layout which is not contained in any other overlap in the layout. A t-contig is a layout with linkage t.

```
TACC----------        ------TACC        -------ACC
---CGGACT-----        ----ACTAC-        ----ACTAC-
------ACTAC---        -CGGACT---        ACGGA-----
---------ACGGA        ACGGA-----        --GGAC----
--------------        ----------        ----------
TACCGGACTACGGA        ACGGACTACC        ACGGACTACC
```

a 1-contig   a 3-contig   a 2-contig

Quality measures 2: Linkage

Finally, we say that a collection $\mathcal{F}$ admits a t-contig if there is a layout which, for every $f \in \mathcal{F}$, uses either f or f^{rc}, and which is a t-contig.

E.g. the collection $\mathcal{F} = \{\texttt{TCAT}, \texttt{GAA}\}$ admits a 2-contig but not a 3-contig:

```
ATGA-
--GAA
-----
ATGAA
```

Quality measures 3: Length of consensus sequence

Finally, an often used measure is: the shorter the consensus sequence, the better.

```
TACC----------        ------TACC
------ACTAC---        ----ACTAC-
---CGGACT-----        -CGGACT---
---------ACGGA        ACGGA-----
--------------        ----------
TACCGGACTACGGA        ACGGACTACC
```

Models 1: Shortest Common Superstring

The simplest model is:

Shortest Common Superstring (SCS)

Input: A collection $\mathcal{F}$ of strings.
Output: A shortest possible string S s.t. for every $f \in \mathcal{F}$, S is a superstring of f.

N.B.

The problem is well-defined because there always exists *some* superstring. (Which?)

Models 1: Shortest Common Superstring

Example for SCS

$\mathcal{F} = \{\texttt{ACT}, \texttt{CTA}, \texttt{AGT}\}$. Then $S = \texttt{ACTAGT}$ is the (unique) shortest common superstring for $\mathcal{F}$.

Models 1: Shortest Common Superstring

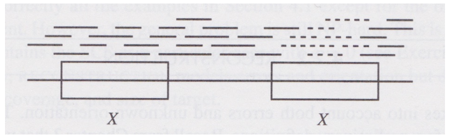
Example for SCS

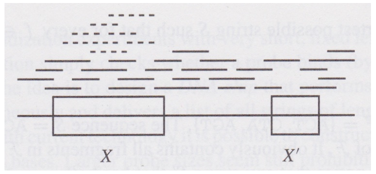
$\mathcal{F} = \{\texttt{ACT}, \texttt{CTA}, \texttt{AGT}\}$. Then $S = \texttt{ACTAGT}$ is the (unique) shortest common superstring for $\mathcal{F}$.

Proof: 1. Clearly, S is a superstring for all 3 strings. 2. Now any string that has both $u = \texttt{ACT}$ and $v = \texttt{AGT}$ as substring must have length at least 6, because they have no overlap. But if length is 6, then the string is either uv or vu. Since also CTA is a substring, the string must be $uv = S$.

Models 1: Shortest Common Superstring

The SCS model assumes no errors and known orientation. Moreover, it does not account for repeats in the target string: If there are long repeats, then the target string is not shortest possible, i.e. an algorithm for SCS will not produce the correct result.





The dashed lines show fragments which are contained in the repeat X. These all get aligned to the one (unique) copy of X. So the middle part of the second occurrence of X does not appear in the consensus sequence.

Models 1: Shortest Common Superstring

N.B.:
SCS is NP-hard but approximation algorithms exist.

NP-hard problems
We will learn exact meaning later. For now, it means "very difficult problem; we cannot hope to find exact solutions efficiently, i.e. fast." However, in this case, since problem is *approximable*: We can hope to find approximate solutions efficiently, i.e. not shortest superstring, but maybe we can find a superstring which is not much longer than a shortest superstring would be.

Models 2: Reconstruction

Now we want to account also for base call errors. Recall the edit distance between two strings:

Edit distance

$$d(u, v) = \text{minimum number of edit operations which turn } u \text{ into } v,$$

where edit operations can be substitutions, deletions, or insertions of bases.

Models 2: Reconstruction

Now we want to account also for base call errors. Recall the edit distance between two strings:

Edit distance

d(u, v) = minimum number of edit operations which turn u into v,

where edit operations can be substitutions, deletions, or insertions of bases.

Example
$d(\texttt{ACTCT}, \texttt{GACCT}) = 2$, because with one insertion and one deletion we can turn the first string into the second, and clearly there is no one operation that will do that. (In general, how do we compute $d(u, v)$?)

Models 2: Reconstruction

Substring edit distance

The same but now u has to be turned into a substring of v:

$$d_s(u, v) = \min\{d(u, v') : v' \text{ substring of } v\}.$$

Example

```
-----GC-GATAG----
CAGTCGCTGATCGTACG
```

$d_s(\mathtt{GCGATAG}, \mathtt{CAGTCGCTGATCGTACG}) = 2$: one insertion, one substitution.
(We have not proved that this is minimum, you just have to believe it.)
Note that this distance is not symmetric! (Upper bound on $d_s(u, v)$?)

Models 2: Reconstruction

Substring edit distance

The same but now u has to be turned into a substring of v:

$$d_s(u, v) = \min\{d(u, v') : v' \text{ substring of } v\}.$$

Example

```
-----GC-GATAG----
CAGTCGCTGATCGTACG
```

$d_s(\mathtt{GCGATAG}, \mathtt{CAGTCGCTGATCGTACG}) = 2$: one insertion, one substitution.
(We have not proved that this is minimum, you just have to believe it.)
Note that this distance is not symmetric! (Upper bound on $d_s(u, v)$?)

N.B. This is one type of *semiglobal alignment*, where gaps at beginning and end of second string are not penalized.

Models 2: Reconstruction

Reconstruction

Input: A collection $\mathcal{F}$ of strings and an error tolerance ϵ, $0 \le \epsilon \le 1$.
Output: A shortest possible string S s.t. for every $f \in \mathcal{F}$,

$$\min(d_s(f, S), d_s(f^{rc}, S)) \le \epsilon |f|,$$

where $|f|$ is the length of f.

So we want to align either f or its reverse complement to S. And if $\epsilon = 0.05$, then we are allowed 5 errors per 100 bp.

Models 2: Reconstruction

The Reconstruction model admits errors and orientation, but does not allow for chimeras, lack of coverage or repeats.

Reconstruction is NP-hard.

Models 3: Multicontig

Taking care of linkage

We want to partition $\mathcal{F}$ in the minimum number of t-contigs.

Example: $\mathcal{F} = \{\texttt{TAATG}, \texttt{TGTAA}, \texttt{GTAC}\}$.

Models 3: Multicontig

Taking care of linkage

We want to partition $\mathcal{F}$ in the minimum number of t-contigs.

Example: $\mathcal{F} = \{\texttt{TAATG}, \texttt{TGTAA}, \texttt{GTAC}\}$.

$$t = 3: \quad \begin{array}{ll} \texttt{--TAATG} & \texttt{GTAC} \\ \texttt{TGTAA--} & \\ \hline \texttt{TGTAATG} & \texttt{GTAC} \end{array}$$

Models 3: Multicontig

Taking care of linkage

We want to partition $\mathcal{F}$ in the minimum number of t-contigs.

Example: $\mathcal{F} = \{\texttt{TAATG}, \texttt{TGTAA}, \texttt{GTAC}\}$.

$$t = 3: \quad \begin{array}{ll} \texttt{--TAATG} & \texttt{GTAC} \\ \texttt{TGTAA--} & \\ \hline \texttt{TGTAATG} & \texttt{GTAC} \end{array} \qquad t = 2: \quad \begin{array}{ll} \texttt{TAATG---} & \texttt{GTAC} \\ \texttt{---TGTAA} & \\ \hline \texttt{TAATGTAA} & \texttt{GTAC} \end{array}$$

Models 3: Multicontig

Taking care of linkage

We want to partition $\mathcal{F}$ in the minimum number of t-contigs.

Example: $\mathcal{F} = \{\texttt{TAATG}, \texttt{TGTAA}, \texttt{GTAC}\}$.

$$t = 3: \quad \begin{array}{ll} \texttt{--TAATG} & \texttt{GTAC} \\ \texttt{TGTAA--} & \\ \hline \texttt{TGTAATG} & \texttt{GTAC} \end{array} \qquad t = 2: \quad \begin{array}{ll} \texttt{TAATG---} & \texttt{GTAC} \\ \texttt{---TGTAA} & \\ \hline \texttt{TAATGTAA} & \texttt{GTAC} \end{array}$$

$$t = 1: \quad \begin{array}{l} \texttt{TGTAA-----} \\ \texttt{--TAATG---} \\ \texttt{------GTAC} \\ \hline \texttt{TGTAATGTAC} \end{array}$$

Models 3: Multicontig

Taking care of linkage

We want to partition $\mathcal{F}$ in the minimum number of t-contigs.

Example: $\mathcal{F} = \{\mathtt{TAATG}, \mathtt{TGTAA}, \mathtt{GTAC}\}$.

$t = 3$:

```
--TAATG     GTAC
TGTAA--
-----------------
TGTAATG     GTAC
```

$t = 2$:

```
TAATG---     GTAC
---TGTAA
-----------------
TAATGTAA     GTAC
```

$t = 1$:

```
TGTAA-----
--TAATG---
------GTAC
----------
TGTAATGTAC
```

So for $t = 3, 2$, we get two contigs, for $t = 1$, we get just one contig.

Models 3: Multicontig

Now add errors to the model:

ϵ-consensus

Every f must have $d(f, u) \leq \epsilon|f|$, where u is the substring to which it has been aligned in the given layout.

Example

```
TGGAA-----
--TAATG---
------GTAA
----------
TGTAATGTAC
```

This is a 0.25-consensus (and not 0.2-consensus), because the last string, even though it *could* have been aligned with 0 errors, has distance 1 to the substring of the consensus to which it has been aligned *in this layout*.

Models 3: Multicontig

Multicontig

Input: A collection $\mathcal{F}$ of strings, and integer $t \geq 0$ and an error tolerance ϵ, $0 \leq \epsilon \leq 1$.
Output: A partition of $\mathcal{F}$ in the minimum number of subcollections $\mathcal{C}_i$, $1 \leq i \leq k$, s.t. every subcollection admits a t-contig with an ϵ-consensus.

Multicontig model admits errors, orientation, lack of coverage, and can partially deal with repeats.

Multicontig is NP-hard.