

Complex-Network Modelling and Inference

Lecture 6: Application: Genome Reconstruction

Matthew Roughan

`<matthew.roughan@adelaide.edu.au>`

https://roughan.info/notes/Network_Modelling/

School of Mathematical Sciences,
University of Adelaide

August 18, 2021

Section 1

Genome Reconstruction

Genomes

Definition

A *genome* is the genetic material in an organism.

- Consists of DNA for us (RNA for viruses)
- Made up of sequences of nucleotides
 - ▶ Humans have about 3 billion nucleotides in our genome
- (roughly) DNA has made up of sequences of 4 molecules (the “bases”)
 - ▶ **A**denine
 - ▶ **G**uanine
 - ▶ **C**ytosine
 - ▶ **T**hymine

which appear in a double-helix, but in fixed pairs, so we can write a DNA sequence as a sequence of these letters, e.g.,

AAGCTTAAGTC

Genome sequencing

- We can't just read the sequence
- There are various approaches to reading, but generally
 - ▶ they can only see a small part at a time: a “read”
 - ▶ we don't know where in the sequence a read comes from
 - ▶ if we get lots, then they will have lots of overlaps
- Practicalities
 - ▶ real reads contain errors
 - ▶ we can't guarantee that reads cover all the genome
 - ▶ you don't know which side of the DNA a read comes from

But we will ignore these problems for the moment

Terminology

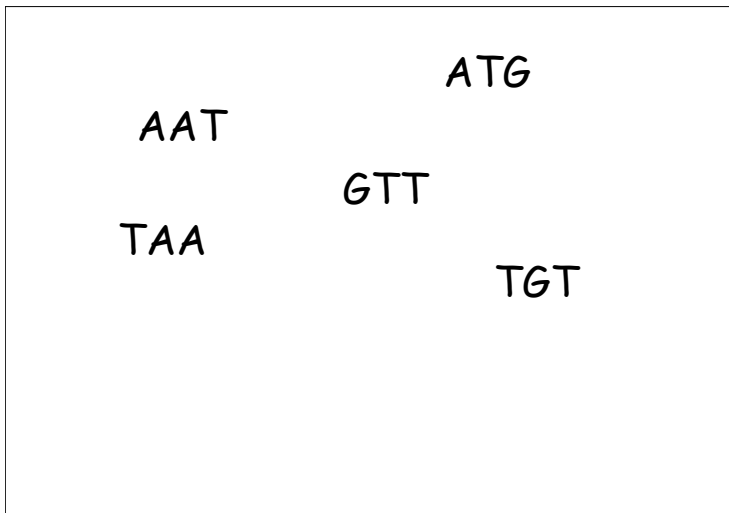
- A sequence of k symbols is called a *k -mer*. Also called an n -gram
 - ▶ e.g., in English, 3-mers are sequences like “abc”, “rtb”, ...
 - ▶ e.g., in binary, 3-mers are sequences like “010”, “111”, ...
 - ▶ e.g., in DNA, 3-mers are sequences like “ATG”, “TTT”, ...

Usually we are interested in sub-sequences from a longer sequence.

- A prefix is the start of a sequence, and a suffix the end
 - ▶ when we are dealing with k -mers, we will mean a $k - 1$ -prefix or suffix
- Given n symbols, how many possible k -mers are there?

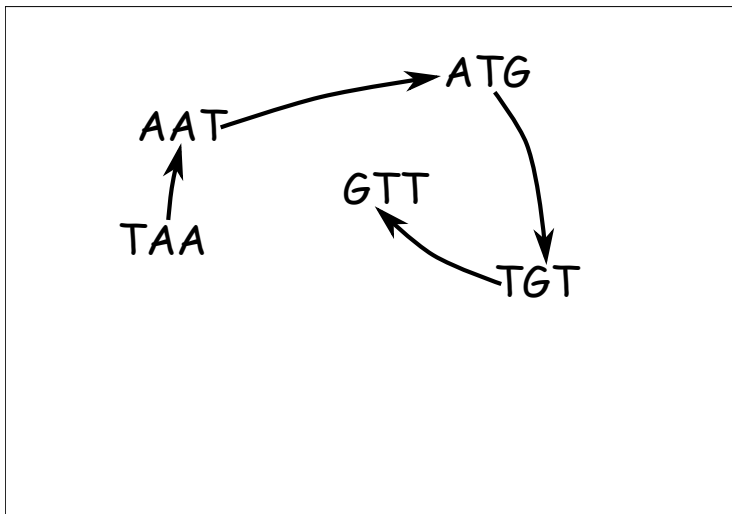
Example 1

Assume our reads provide all 3-mers from a DNA sequence



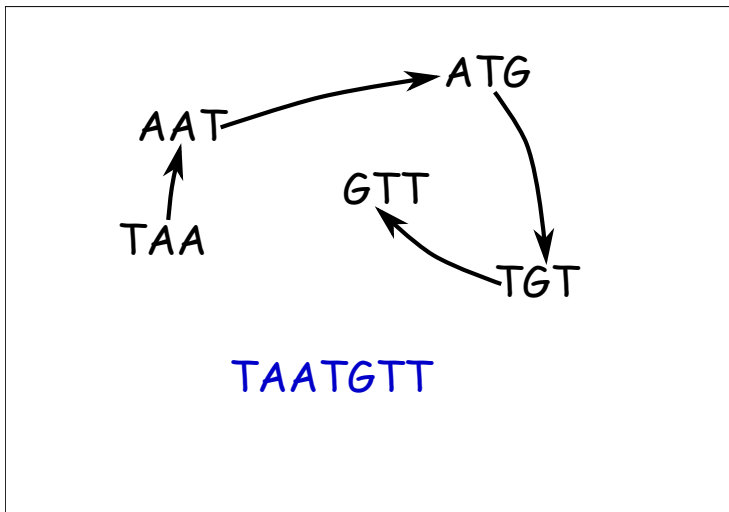
Example 1

Create a graph by linking suffix \rightarrow prefix



Example 1

The sequence is just the path through this graph



Are we done?

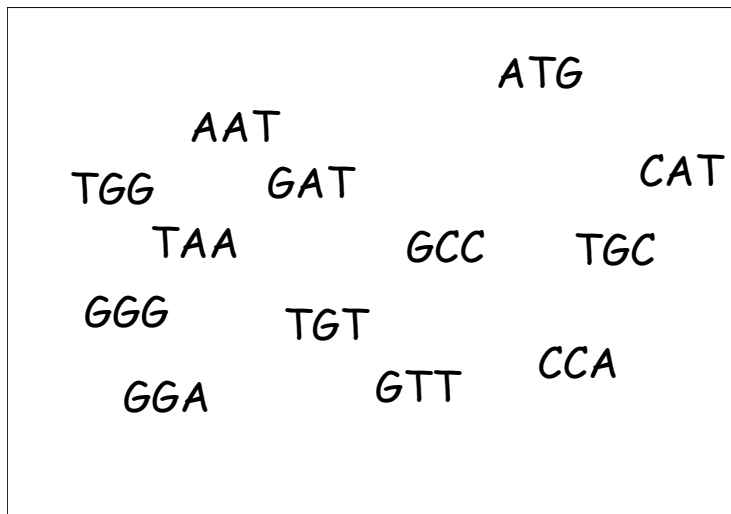
No!

- k -mers are repeated in a long sequence
 - ▶ the above assumed that each appeared exactly once in the sequence
 - ▶ there can be more than one prefix/suffix match
- Above also assumes
 - ▶ we have all k -mers
 - ▶ no errors

but for the moment, lets ignore these issues, as the first one is big enough

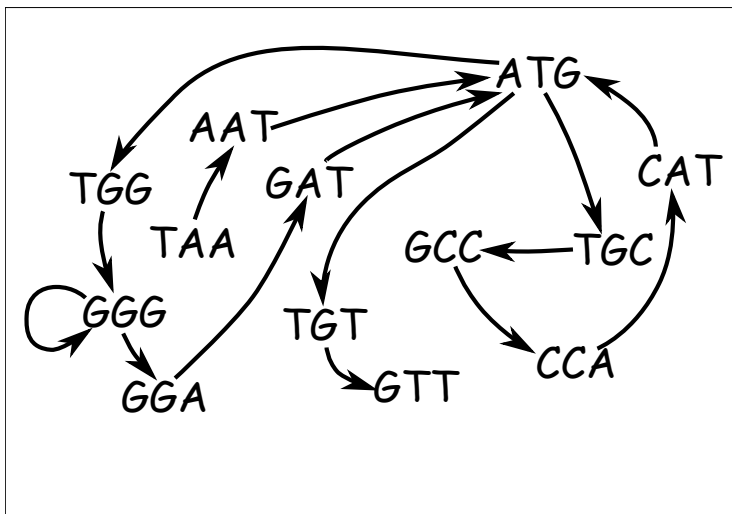
Example 2

A (slightly) bigger example



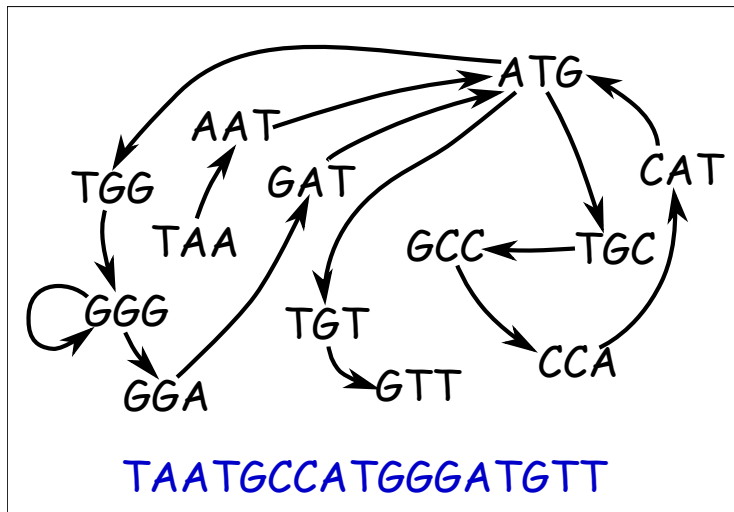
Example 2

Harder to spot the "walk" (it isn't cycle-free, so not a "path")



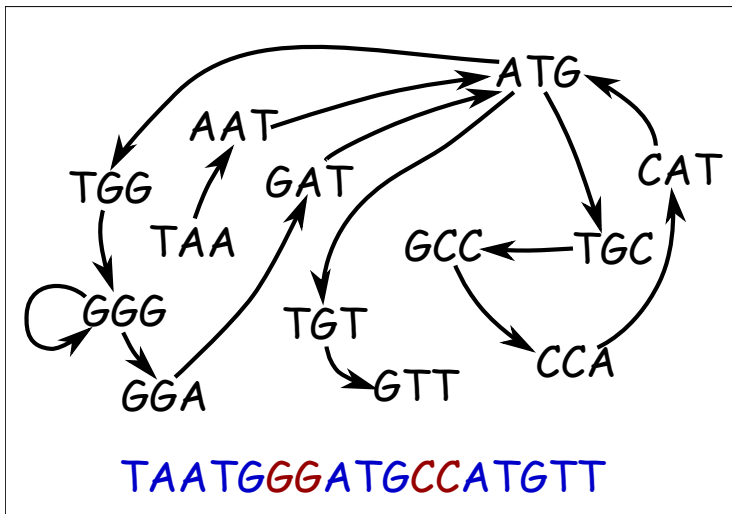
Example 2

Here is one possibility



Example 2

But there is ambiguity



A how to

- If we knew multiplicity of ATG we would create 3 nodes.
 - ▶ this simplifies some problems
 - ★ e.g., we know we don't loop through GGG
 - ▶ it makes the graph quite a bit more complicated
 - ★ 2 extra nodes
 - ★ lots of extra links
 - ▶ it doesn't resolve all ambiguities
 - ★ only way to avoid is to have longer sequences
- Given this new graph the problem becomes one of finding a *Hamiltonian path*

Hamiltonian paths and cycles

Definition

A *Hamiltonian path* is a path that visits each node exactly once. A *Hamiltonian cycle* visits each node exactly once, and then returns to the start.

The problem of finding a Hamiltonian path is *NP-complete*

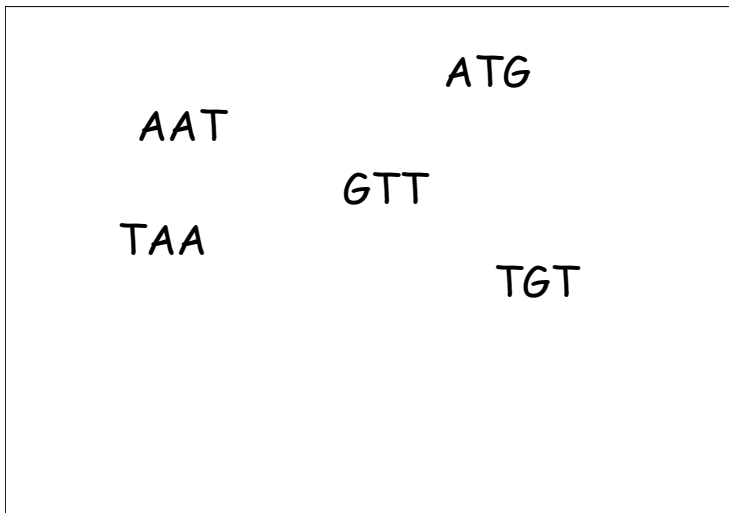
- this is a hard class of problems (computationally)
- there are no known polynomial-time algorithms
 - ▶ but it is easy to check a given cycle is correct
- this is the problem early genome sequencers attempted, but you can't (practically) solve big NP-complete problems
- so they came up with an alternative graph: the de Bruijn graph

de Bruijn graph

- Often one set of data can be represented by multiple different graphs
- This is a classic example
 - ▶ the obvious graph is not the right one to work with
- *de Bruijn graph*
 - ▶ the nodes are prefixes and suffixes
 - ▶ the edges are the k -mers
 - ★ they link their prefix to their suffix

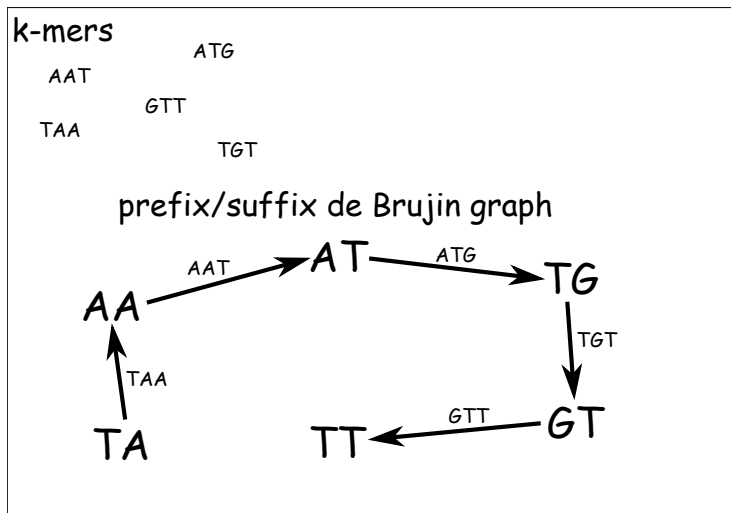
Example 1

Construct a de Bruijn graph



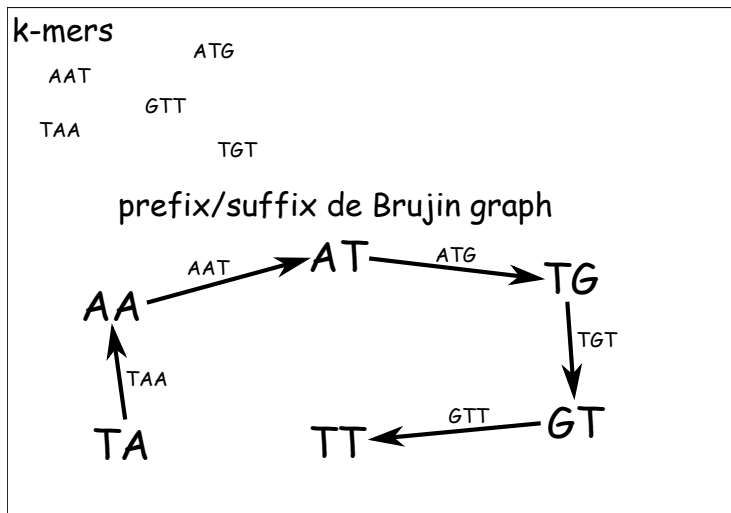
Example 1

Construct a de Bruijn graph



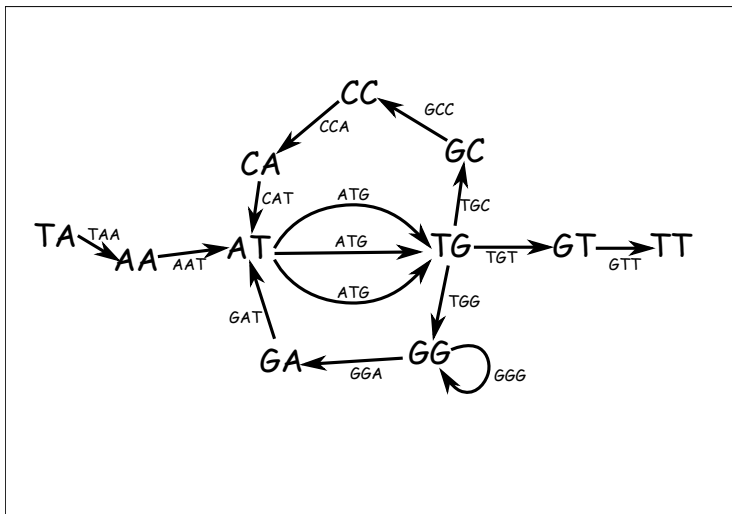
Example 1

Now we just look for an Eulerian path in this graph



Example 2

de Bruijn graph of second example



Eulerian paths

- The sequence on the de Bruijn graph can be found by taking finding an Eulerian path
 - ▶ a path that goes along each edge exactly once
 - ▶ need to include multiplicity of edges in construction
- Eulerian paths are easy to construct
- We need a minor adaptation here for directed graphs

Definition

A digraph is *balanced* if the in-degree of each node is the same as its out degree.

Theorem

An Eulerian cycle exists on a digraph if and only if it is balanced and strongly connected.

I leave the proof as an exercise, as well as the extension to an Eulerian path.

Further reading I



Phillip Compeau and Pavel Pevzner, *Bioinformatics algorithms: An active learning approach*, Active Learning Publishers, 2014.