Read Mapping

de Bruijn Graph-based de novo Assembly

Peter N. Robinson

Institut für Medizinische Genetik und Humangenetik
Charité Universitätsmedizin Berlin

Genomics: Lecture #3 WS 2014/2015

Today

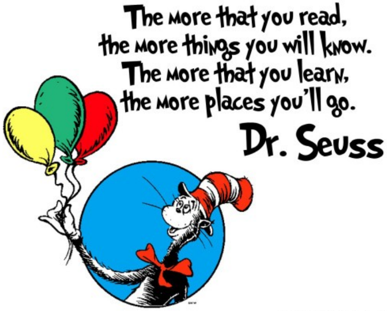
Last time we looked at some basic concepts of genome sequencing and assembly. It should however be clear that just looking for a Eulerian path in a de Bruijn graph will not solve all problems. Today, we will look at some ideas and concepts to use de Bruijn graphs for practical assembly algorithms

Main source for today:

Pevzner PA, Tang H, Waterman MS (2001) An Eulerian path approach to DNA fragment assembly. *PNAS* **98**:9748-53.

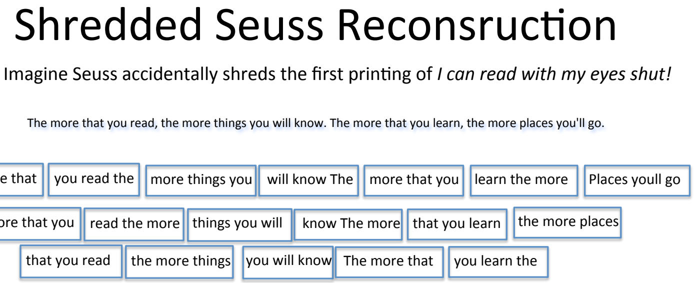
The Dr. Seuss-Ome

Just to review the topics of the last lecture, we will adapt a brilliant idea of Michael Schatz (CHSL), who used the first sentence of Dicken's *A Tale of Two Cities* to illustrate De Bruijn graphs



Dr. Seuss (Theodor Seuss Geisel), American writer of children's books. *I Can Read With My Eyes Shut!* (1978)

The Dr. Seuss-Ome



Shredded Seuss Reconsruction

Imagine Seuss accidentally shreds the first printing of *I can read with my eyes shut!*

The more that you read, the more things you will know. The more that you learn, the more places you'll go.

The more that | you read the | more things you | will know The | more that you | learn the more | Places youll go

more that you | read the more | things you will | know The more | that you learn | the more places

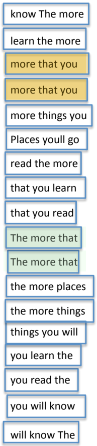
that you read | the more things | you will know | The more that | you learn the

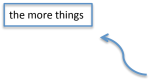
- Many different copies of the book are shredded into three word fragments (“3-mer” subsequences)
- Start position of the fragments is random
- Goal: find overlaps to reconstruct the Seussome

The Dr. Seuss-Ome

Greedy Reconsruction

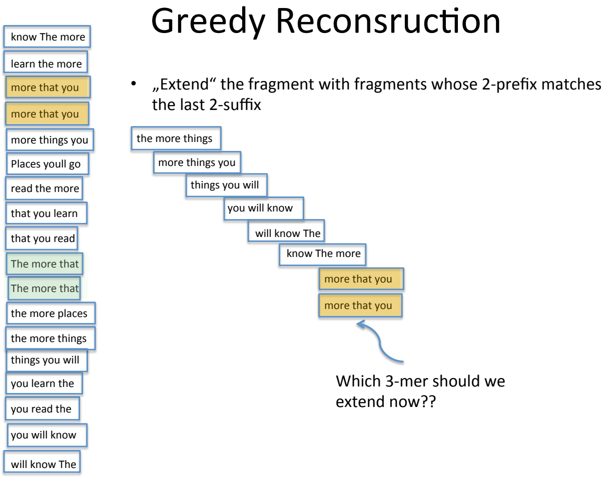
- Let's try to reconstruct the original text on the basis of overlaps
- Start with an arbitrary fragment
- „Extend“ the fragment with fragments whose 2-prefix matches the last 2-suffix





We choose this 3-mer „at random“

The Dr. Seuss-Ome



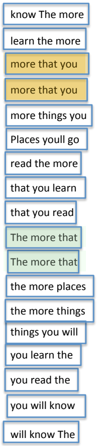
Greedy Reconsruction

- „Extend" the fragment with fragments whose 2-prefix matches the last 2-suffix

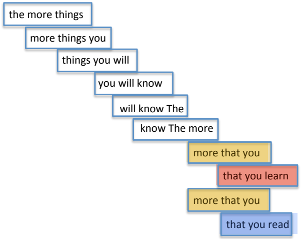
Which 3-mer should we extend now??

The Dr. Seuss-Ome

Greedy Reconsruction



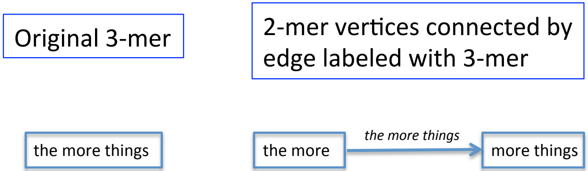
- „Extend“ the fragment with fragments whose 2-prefix matches the last 2-suffix



The repeated k-mer makes the correct reconstruction ambiguous

The more things you will know the more that you [***learn/read***]

The Dr. Seuss-Ome



- $G = (V, E)$
- V all length $k-1$ fragments (here: $k-1=2$)
- E directed edges between consecutive subfragments with labels of length k (here: $k=3$)
- Note that vertices overlap by $k-2$ words (here: $k-2=1$)

The Dr. Seuss-Ome

- How do we choose a value for k in real life?
- "Big enough": the $k-1$ mer sequences should mainly be unique
- However, memory usage grows as $\mathcal{O}(nk)$, or about $n \approx 2.4 \times 10^9$ nucleotides, k-mer size $k = 27$, requiring about 15 GB (nk/4 bytes) of memory to store the nodes alone.
- Repeats in typical genomes are larger than individual reads, so even if we could hope to sequence without errors, we would not quite yet have a complete solution to the assembly problem

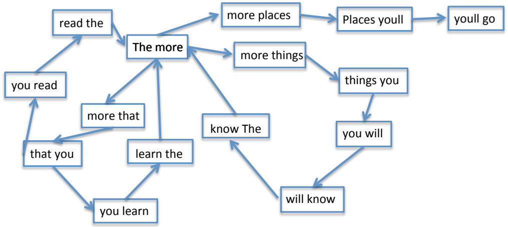
We will now construct a de Bruijn graph (DBG) from the Seussome as explain in the previous lecture

The Dr. Seuss-Ome

know The more
learn the more
more that you
more that you
more things you
Places youll go
read the more
that you learn
that you read
The more that
The more that
the more places
the more things
things you will
you learn the
you read the
you will know
will know The

DBG Reconstruction

The more that you read, the more things you will know. The more that you learn, the more places you'll go.



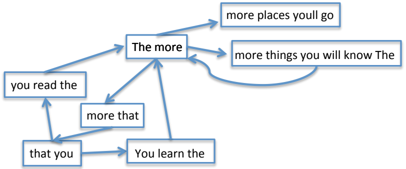
A particular Eulerian tour of the graph reconstructs the original text but there are multiple such tours

The Dr. Seuss-Ome

DBG Compression

know The more
learn the more
more that you
more that you
more things you
Places youll go
read the more
that you learn
that you read
The more that
The more that
the more places
the more things
things you will
you learn the
you read the
you will know
will know The

The more that you read, the more things you will know. The more that you learn, the more places you'll go.



After reconstruction, many edges are unambiguous and can be compressed

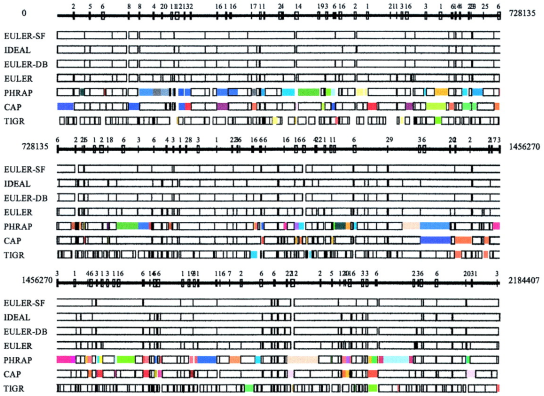
Back to real life...

Unfortunately, there is a time when it is necessary to wake up to the realities of real life...In the rest of the lecture we will cover the EULER algorithm as developed by Pevzner, Tang and Waterman (2001).

The authors showed that EULER was superior to previous algorithms based the overlap-layout-consensus paradigm. There were two key ideas

- Intelligent way of dealing with repetitive regions
- Intelligent way of dealing with sequence errors
- Superpath approach to disambiguating the de Bruijn graph

EULER



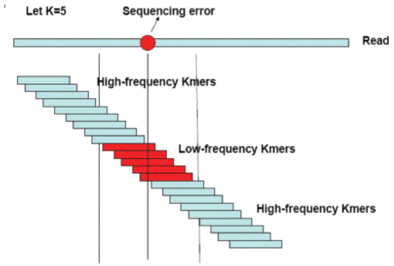
Comparative analysis of euler, phrap, cap, and tigr assemblers

- Every box corresponds to a contig in *Neisseria meningitidis* assembly
- colored boxes correspond to assembly errors

EULER

Unfortunately, the straightforward Eulerian path approach, although very promising, did not scale up well. The problem is that sequencing errors transform a simple de Bruijn graph into a tangle of erroneous edges.

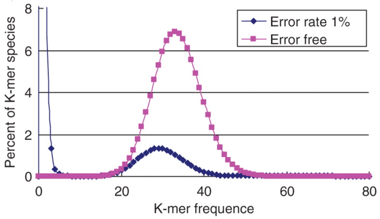
EULER



- Li et al. (2011) Briefings in functional genomics (2012) 11 (1): 25-37.

- The 5 k-mers which crossed the error base appear in low frequency
- the surrounding k-mers appear in high frequency.
- In practice, the situations are often more complex than this

EULER

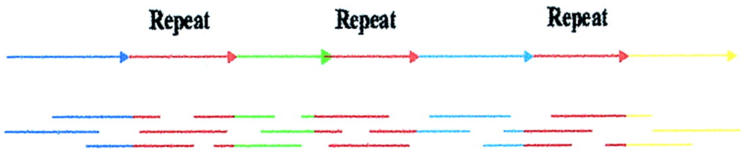


- Li et al. (2011) Briefings in functional genomics (2012) 11 (1): 25-37.

- Distribution of 17-mer frequency for error free and 1% erroneous data
- In the 1% error curve, about 80% k-mer species have frequency below five, most of which are caused by sequencing errors.

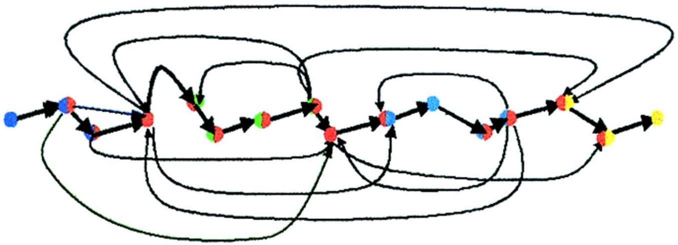
Therefore, an obvious heuristic is to remove low-frequency k-mers from the assembly

EULER



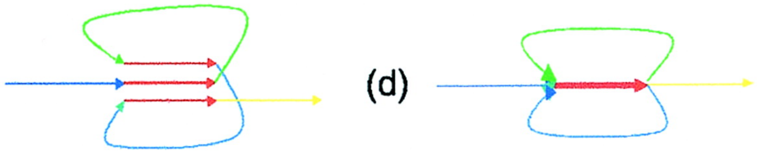
- Consider this DNA sequence, which consists of four unique segments A, B, C, D, and one triple repeat R (in red).
- We perform WGS and obtain 16 reads
- The reads are here conveniently colored according to their sequence of origin

EULER



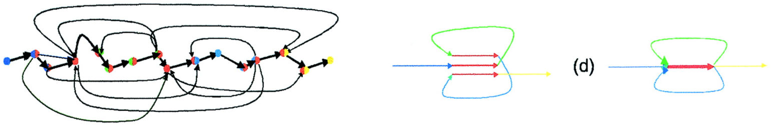
- Every read corresponds to a vertex in the **overlap graph**
- two vertices are connected by an edge if the corresponding reads overlap.
- The fragment assembly problem is thus cast as finding a path in the overlap graph visiting every vertex exactly once, a **Hamiltonian Path Problem**.
- The Hamiltonian Path Problem is NP-complete
- This is why fragment assembly of highly repetitive genomes is a notoriously difficult problem.

EULER



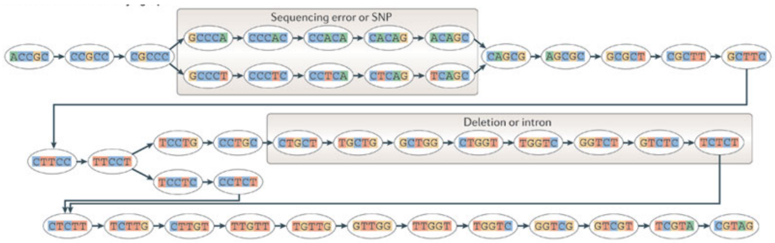
- In an informal way, one can visualize the construction of the de Bruijn graph by representing a DNA sequence as a "thread" with repeated regions covered by a "glue" that "sticks" them together
- The resulting de Bruijn graph consists of $4 + 1 = 5$ edges (we assume that the repeat edge is obtained by gluing three repeats and has multiplicity three).
- In this approach, every repeat corresponds to an edge rather than a collection of vertices in the layout graph.

EULER



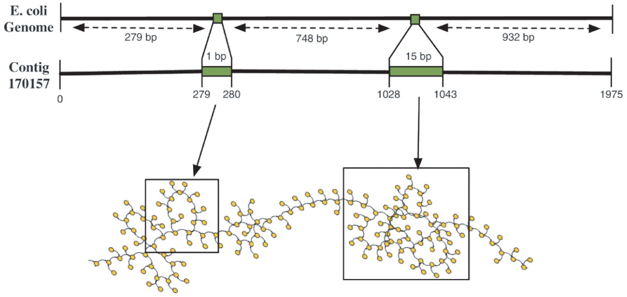
- Obviously, de Bruijn graph is a much simpler representation of repeats than the overlap graph
- fragment assembly is now cast as finding a path visiting every edge of the graph exactly once, an Eulerian Path Problem.
- There are two Eulerian paths in the graph: one of them corresponds to the sequence reconstruction ARBRCRD, whereas the other one corresponds to the sequence reconstruction ARCRBRD.
- In contrast to the Hamiltonian Path Problem, the Eulerian path problem is easy to solve even for graphs with millions of vertices, because there exist linear-time Eulerian path algorithms

EULER



- Read errors cause characteristic patterns in the de Bruijn graph
- The numer of nodes “explodes”

EULER



Ronen R (2012) Bioinformatics 28:i188-96.

- Up to 90% of nodes in de Bruijn assembly grphs may stem from sequence errors
- Here: The alignment of a 1975 bp contig from the assembly with Velvet and k=31, showing two insertions in the alignment, having respective lengths 1 bp and 15 bp.
- The de Bruijn graph constructed from the set of permissively aligned reads to this contig contains bulges and whirls at regions corresponding to the insertions in the contigs

EULER: error correction

- If we knew the genome sequence and could somehow correctly align the reads, it would be relatively easy to perform error correction
- But of course we do not know the true genome seqeunce G
- If we knew the set of all k-mers present in G, we could also try to correct the reads accordingly
- We can approximate this set based on the evidence in the reads data using the assumption that true k-mers are present multiple times in the collection of reads but error-related k mers have low counts in the data

$\boxed{G_k}$: the set of all k-tuples in genome G

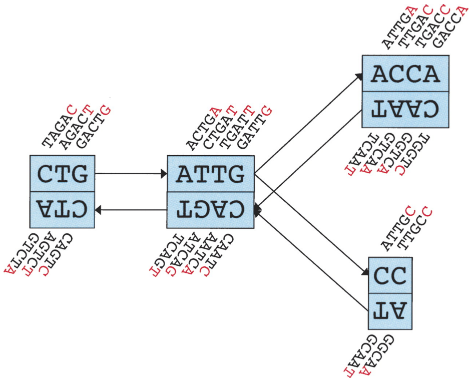
EULER: error correction

$\boxed{G_k}$: the set of all k-tuples in genome G

- An k-mer is called **solid** if it belongs to more than M reads and **weak** otherwise.
- M is a threshold, e.g., 4
- A natural approximation for G_k is the set of all solid k-mers from a sequencing project.
- Now let T be a collection of k-mers called a **spectrum**.
- A string s is called a T-string if all its k-mers belong to T.

EULER: Strand ambiguity

- WGS is typically not strand-specific
- Solution: Treat each k-mer as actually being two k-mers, the original sequence and the reverse complement



Zerbino DR, Birney E (2008) Genome Res. 18, 821–829

EULER: error correction

Spectral Alignment Problem.
Given a string s and a spectrum T, find the minimum number of mutations in s that transform s into a T-string.

We are given a collection of reads (strings) $S = \{s_1, \ldots, s_n\}$ from a sequencing project and an integer k.

- The **spectrum** of S is a set S_k of all k-mers from the reads $s_1, \ldots, s_n$ and $\overline{s}_1, \ldots, \overline{s}_n$
- $\overline{s}$ denotes a reverse complement of read s.
- Let Δ be an upper bound on the number of errors in each DNA read

EULER: error correction

Error Correction Problem.
Given S, Δ, and k, introduce up to Δ corrections in each read in S in such a way that $|S_k|$ is minimized.

- An error in a read s affects at most k individual k-mers in s and k more k-mers in $\overline{s}$
- Thus, an error usually creates $2k$ erroneous k-mers that point to the same sequencing error
- If the error is close to the end of a read, less erroneous k-mers are created ($2d$ for positions within a distance $d < k$ from the endpoint of the reads)
- A greedy approach for the Error Correction Problem is to look for error corrections in the reads that reduce the size of S_k by $2k$ (or $2d$ for positions close to the endpoints). This simple procedure already eliminates 86.5% of the errors in sequencing reads.

EULER: error correction

Greedy Error Correction.
Look for error corrections in the reads that reduce the size of S_k by $2k$ (or $2d$ for positions close to the endpoints).

- Why does this work?
- This approach eliminated 86.5% of errors in the test bacterial genomes used
- Similar heuristics improve error correction to 98% (look for k-mers that are nearly identical to other k-mers with high multiplicity)
- For details see Pevzner PA, Tang H, Waterman MS (2001) A new approach to fragment assembly in DNA sequencing. In Proceedings of the Fifth International Conference on Computational Biology (RECOMB 2001, Montreal). pp. 256265.

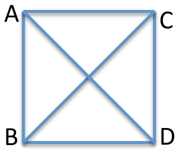
EULER: Eulerian Superpath

Recall first the Eulerian **path**

- We are given a collection of reads (strings) $S = \{s_1, \dots, s_n\}$
- Define the de Bruijn graph $G(S_k)$ with vertex set S_{k1} (the set of all $(k1)$-tuples from S) as follows.
  1. An $(k1)$-tuple $v \in S_{k1}$ is joined by a directed edge with an $(k1)$-tuple $w \in S_{k1}$, if S_k contains an l-tuple for which the first $k1$ nucleotides coincide with v and the last $k1$ nucleotides coincide with w.
  2. Each k-tuple from S_k corresponds to an edge in $G(S_k)$.
- We first need to recall what a Eulerian path and cycle is, and what a Chinese postman path is.

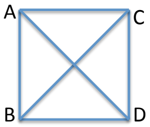
Traversable graph

A traversable graph is one that can be drawn without taking a pen from the paper and without retracing the same edge. In such a case the graph is said to have an **Eulerian path** (a trail in a graph which visits every edge exactly once.)



- Is this graph traversable?

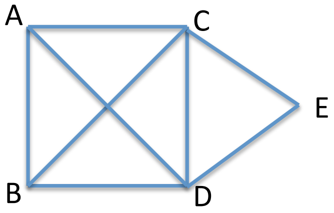
Traversable graph



Vertex	Degree
A	3
B	3
C	3
D	3

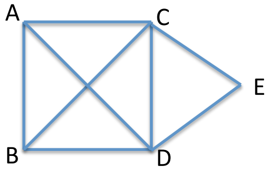
- This graph does not have a Eulerian path

Traversable graph



- Is this graph traversable?

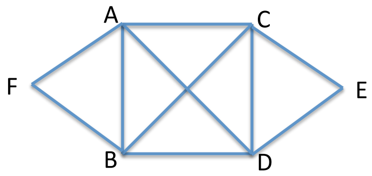
Traversable graph



Vertex	Degree
A	3
B	3
C	4
D	4
E	2

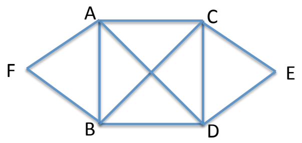
- This graph has a Eulerian path from A to B or vice versa

Traversable graph



- Is this graph traversable?

Traversable graph



Vertex	Degree
A	4
B	4
C	4
D	4
E	2
F	2

- This graph has a Eulerian cycle

Traversable graph

The pattern is related, of course, to the degrees of the vertices of the graph

- When the order of all the vertices is even, the graph is traversable and we can draw it.
- If there are two vertices of odd degree and all other vertices are of even degree, there is a Eulerian path
- If there are more than two odd vertices the graph cannot be traversed without repeating an edge.

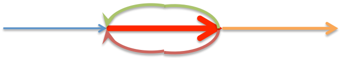
Chinese postman problem

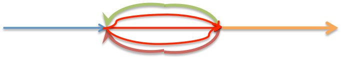
The Chinese postman problem (CPP), postman tour or route inspection problem is to find a shortest closed path or circuit that visits every edge of a (connected) undirected graph. Each edge **must** be visited once but it **can** be visited multiple times

- the problems was named 'Chinese Postman' because it was originally studied by the Chinese mathematician Kwan Mei-Ko in 1962
- There are several relatively straightforward algorithms for the CPP (we will try one in the recital)

Chinese postman problem

- The Chinese Postman Problem is closely related to the problem of finding a path visiting every edge of a graph exactly once, an Eulerian Path Problem
- One can transform the Chinese Postman Problem into the Eulerian Path Problem by introducing multiplicities of edges in the de Bruijn graph.
- For example, one can substitute every edge in the de Bruijn graph by k parallel edges, where k is the number of times the edge is used in the Chinese Postman path.
- If S contains the only sequence s_1, this operation creates m "parallel" edges for every k-tuple repeating m times in s_1





Chinese postman problem

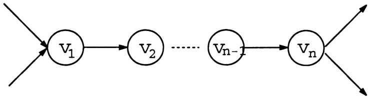
Nomenclature

- A vertex v is called a ...
  - source if $\text{indegree}(v) = 0$
  - sink if $\text{outdegree}(v) = 0$
  - branching vertex if $\text{indegree}(v)\ \text{outdegree}(v) > 1$

For the NM genome, the de Bruijn graph has 502,843 branching vertices for original reads (for k-tuple size 20). Error corrections simplify this graph and lead to a graph with 382 sources and sinks and 12,175 branching vertices.

Chinese postman problem

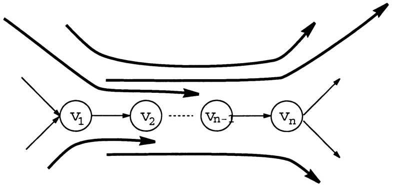
A path $v_1, \ldots, v_n$ in the de Bruijn graph is called a **repeat** if indegree$(v_1) > 1$, outdegree$(v_n) > 1$, and indegree $(v_1) =$ outdegree$(v_i) = 1$ for $1 \leq i \leq n1$. Edges entering the vertex v_1 are called **entrances** into a repeat, whereas edges leaving the vertex v_n are called **exits** from a repeat.



An Eulerian path visits a repeat a few times, and every such visit defines a pairing between an entrance and an exit.

Chinese postman problem

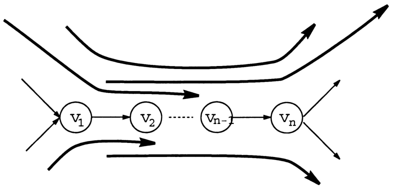
Repeats may create problems in fragment assembly, because there are a few entrances in a repeat and a few exits from a repeat, but it is not clear which exit is visited after which entrance in the Eulerian path.



- A read-path **covers** a repeat if it contains an entrance into and an exit from this repeat
- Every covering read-path reveals some information about the correct pairings between entrances and exits.
- A repeat is called a **tangle** if there is no read-path containing this repeat

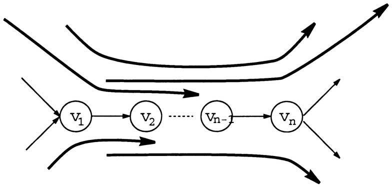
Chinese postman problem

QUESTION: Which read keeps this repeat from being a tangle?



Chinese postman problem

QUESTION: Which read keeps this repeat from being a tangle?



ANSWER: The uppermost path covers the repeat and defines the correct pairing between the corresponding entrance and exit. If this path were not present, the repeat $v_1, \ldots, v_n$ would become a tangle.

Eulerian Superpath Problem

Eulerian Superpath Problem
Given an Eulerian graph and a collection of read-paths in this graph, find an Eulerian path in this graph that contains all these paths as subpaths.

To solve the Eulerian Superpath Problem, we transform both the graph G and the system of paths $\mathcal{P}$ in this graph into a new graph G_1 with a new system of paths $\mathcal{P}_1$. Such transformation is called equivalent if there exists a one-to-one correspondence between Eulerian superpaths in $(G,\mathcal{P})$ and $(G_1,\mathcal{P}_1)$. Our goal is to make a series of equivalent transformations

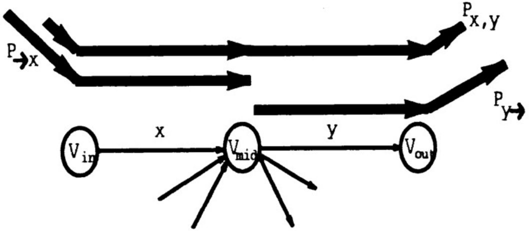
$$(G,\mathcal{P}) \to (G_1,\mathcal{P}_1) \to \ldots \to (G_k,\mathcal{P}_k)$$

that lead to a system of paths $\mathcal{P}_k$, with every path being a **single edge**.

Eulerian Superpath Problem

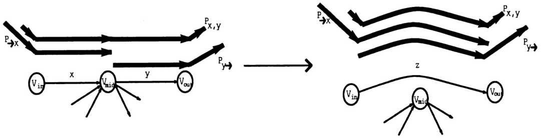
We will describe a simple equivalent transformation of the graph

Let $x = (v_{in}, v_{mid})$ and $y = (v_{mid}, v_{out})$ be two consecutive edges in graph G, and let $\mathcal{P}_{x,y}$ be a collection of all paths from $\mathcal{P}$ that include both these edges as a subpath.



x, y-detachment

Informally, x, y-detachment bypasses the edges x and y via a new edge z and directs all paths in $\mathcal{P}_{x,y}$ through z, thus simplifying the graph.



However, this transformation affects other paths and needs to be defined carefully.

- Define $\mathcal{P}_{\to x}$ as a collection of paths from $\mathcal{P}$ that end with x
- Define $\mathcal{P}_{y \to}$ as a collection of paths from $\mathcal{P}$ that start with y
- x, y-detachment is a transformation that adds a new edge $z = (v_{in}, v_{out})$ and deletes the edges x and y from G
  1. substitute z for x, y in all paths from $\mathcal{P}_{x,y}$
  2. substitute z for x in all paths from $\mathcal{P}_{\to x}$
  3. substitute z for y in all paths from $\mathcal{P}_{y \to}$

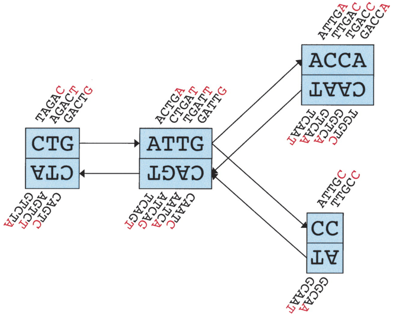
x, y-detachment

Because every detachment reduces the number of edges in G, the detachments will eventually shorten all paths from $\mathcal{P}$ to single edges and will reduce the Eulerian Superpath Problem to the Eulerian Path Problem.

- The EULER paper and several other papers from that group describe a number of clever bells and whistles for genome assembly by de Bruijn graph analysis that we have not covered here
- Especially interesting are strategies for pairwise end sequencing, known colloquially as double-barrel shotgun sequencing, and for resolving repeats
- See references at the end of these slides

Velvet

We will now discuss the Velvet algorithm of Zerbino and Birney, in particular the strategies for error correction, which are different from those of EULER



- Velvet uses a slightly different formulation of the de Bruijn graph (not relevant for the rest of the discussion today)

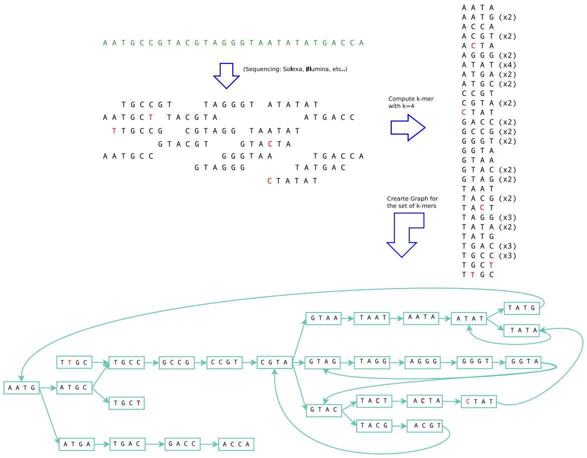
Velvet

EULER relied upon erroneous *k*-mers haveing a low coverage. Velvet instead focuses on topological features.

- Erroneous data create three types of structures: “tips” due to errors at the edges of reads
- “bulges” due to internal read errors or to nearby tips connecting
- erroneous connections due to cloning errors or to distant merging tips

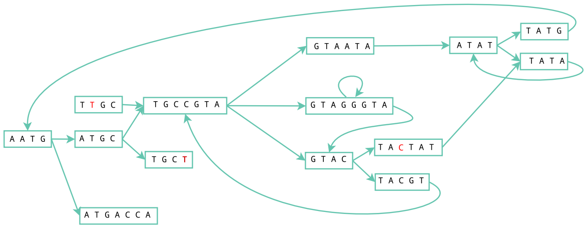
The three features are removed consecutively.

Velvet



graphic: wikipedia

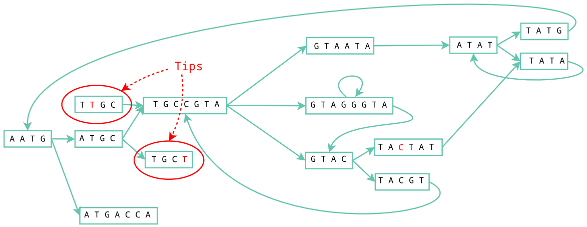
Velvet



graphic: wikipedia

- Whenever a node A has only one outgoing arc that points to another node B that has only one ingoing arc, the two nodes (and their twins) are merged. Iteratively, chains of blocks are collapsed into single blocks.

Velvet

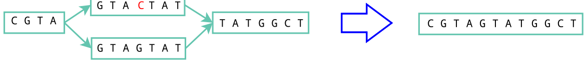


graphic: wikipedia

A node is considered a tip and should be erased if

- it is disconnected on one of its ends
- the length of the information stored in the node is shorter than 2k
- and the arc leading to this node has a low multiplicity (In other words, starting from that node, going through the tip is an alternative to a more common path)

Velvet: bubble

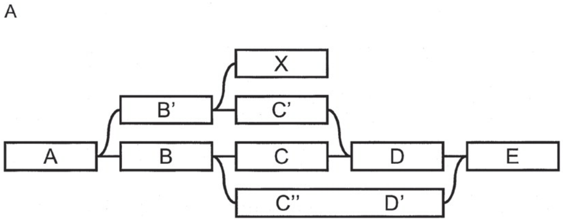


graphic: wikipedia

- We consider two paths redundant if they start and end at the same nodes (forming a “bubble”) and contain similar sequences.
- Such bubbles can be created by errors or biological variants, such as SNPs or cloning artifacts prior to sequencing.

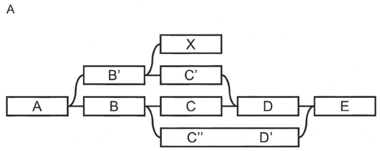
Velvet: bubble

Erroneous bubbles are removed by an algorithm called **Tour Bus**.



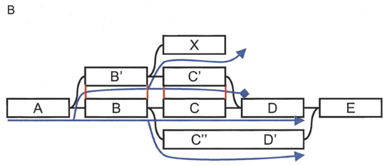
Let us consider this graph

Velvet: bubble



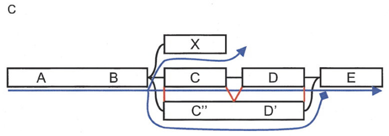
- Detection of redundant paths is done through a Dijkstra-like breadth-first search.
- The algorithm starts from an arbitrary node and progresses along the graph, visiting nodes in order of increasing distance from the origin.
- The distance between two consecutive nodes A and B is the length of $s(B)$ divided by the multiplicity of the arc leading from A to B.
- This ad hoc metric gives priority to higher coverage, more reliable, paths.

Velvet: bubble



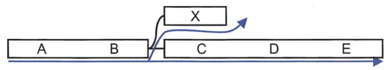
- The search (BFS) starts from A and spreads toward the right.
- The progression of the top path (through B' and C') is stopped because D was previously visited.
- The nucleotide sequences corresponding to the alternate paths B'C' and BC are extracted from the graph, aligned, and compared.

Velvet: bubble



- The two paths are judged similar, so the longer one, B'C', is merged into the shorter one, BC.
- The merging is directed by the alignment of the consensus sequences, indicated in red lines in B.
- Note that node X, which was connected to node B', is now connected to node B.
- The search progresses, and the bottom path (through C'' and D') arrives second in E. Once again, the corresponding paths, C''D' and CD are compared.

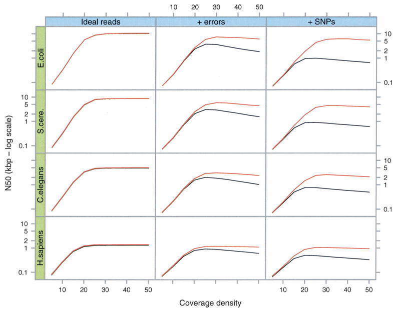
Velvet: bubble



- CD and C"D' are judged similar enough.
- The longer path is merged into the shorter one.

Velvet

We have discussed 2 of the three error correction heuristics of Velvet. Error correction could be shown to substantially improve the N50 on simulated datasets



Summary

Today we have discussed de Bruijn graphs for genome assembly and touched on some of the bells and whistles that are required to get this techniqwue to work on real world data (error correction, repeats...).

- There are at least 25 academic de novo genome assemblers, each possessing its own range of application, are developed for short reads datasets from different sequencing platforms in the last few years
- Thus, genome assembly is still an active area of algorithmic research and development
- New algorithms will be required to get the most out of longer NGS reads in the near future (nanopore etc)

Overview: Zhang W et al (2011) A practical comparison of de novo genome assembly software tools for next-generation sequencing technologies. *PLoS One*. **6**:e17915.

Finally

- Email: peter.robinson@charite.de
- Office hours by appointment

Further reading

- Pevzner PA, Tang H, Waterman MS (2001) An Eulerian path approach to DNA fragment assembly *Proc Natl Acad Sci U S A* **98**:9748-53
- Compeau PE, Pevzner PA, Tesler G (2011) How to apply de Bruijn graphs to genome assembly. *Nat Biotechnol* **29**:987-91
- Zerbino DR, Birney E (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res* **18**:821-829.