Velvet Algorithm

CS1820

Previous Sequencing Methods

- Sanger sequencing, pyrosequencing
- Whole genome shotgun fragment assembly
  - Piecing together fragments randomly extracted from sample to form **continguous sequences (contigs)**
  - Overlap-Layout-Consensus
- Euler Assembly
  - de Bruijn Graphs

Problem

- Not suited for very short reads (~25-50bp)
  - More repeats and errors
- Cost effective to use very short reads

Velvet Algorithm

- de Bruijn Graph Assembly
- Very Short (~25-50bp), Paired Reads
- Eliminate errors and resolve repeats

Structure

- De Bruin Graph
  - Each node N represents series of overlapping k-mers
  - Adjacent k-mers overlap by k-1
  - Each node attached to “twin node”
    - the reverse complement
    - Form “block”

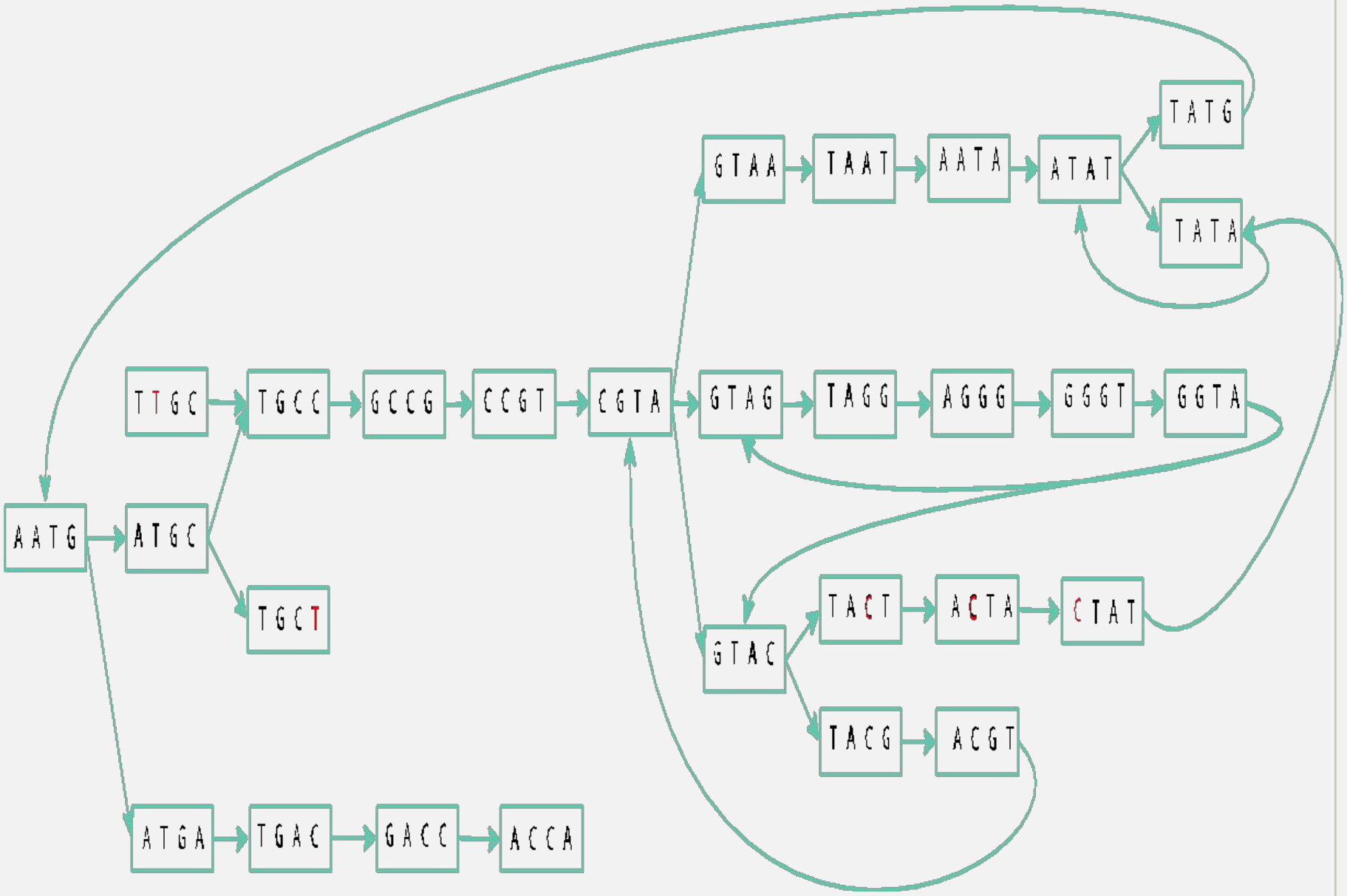
ATTGA
TTGAC
TGACC
GACCA
ACCA
CAAT
TGGTC
GGTCA
TCAAT
TAGAC
AGACT
GACTG
CTG
CTA
CAGTC
AGTCT
GTCTA
ACTGA
CTGAT
TGATT
GATTG
ATTG
CAGT
CAATC
AATCA
ATCAG
TCAGT
ATTGC
TTGCC
CC
AT
GGCAA
GCAAT

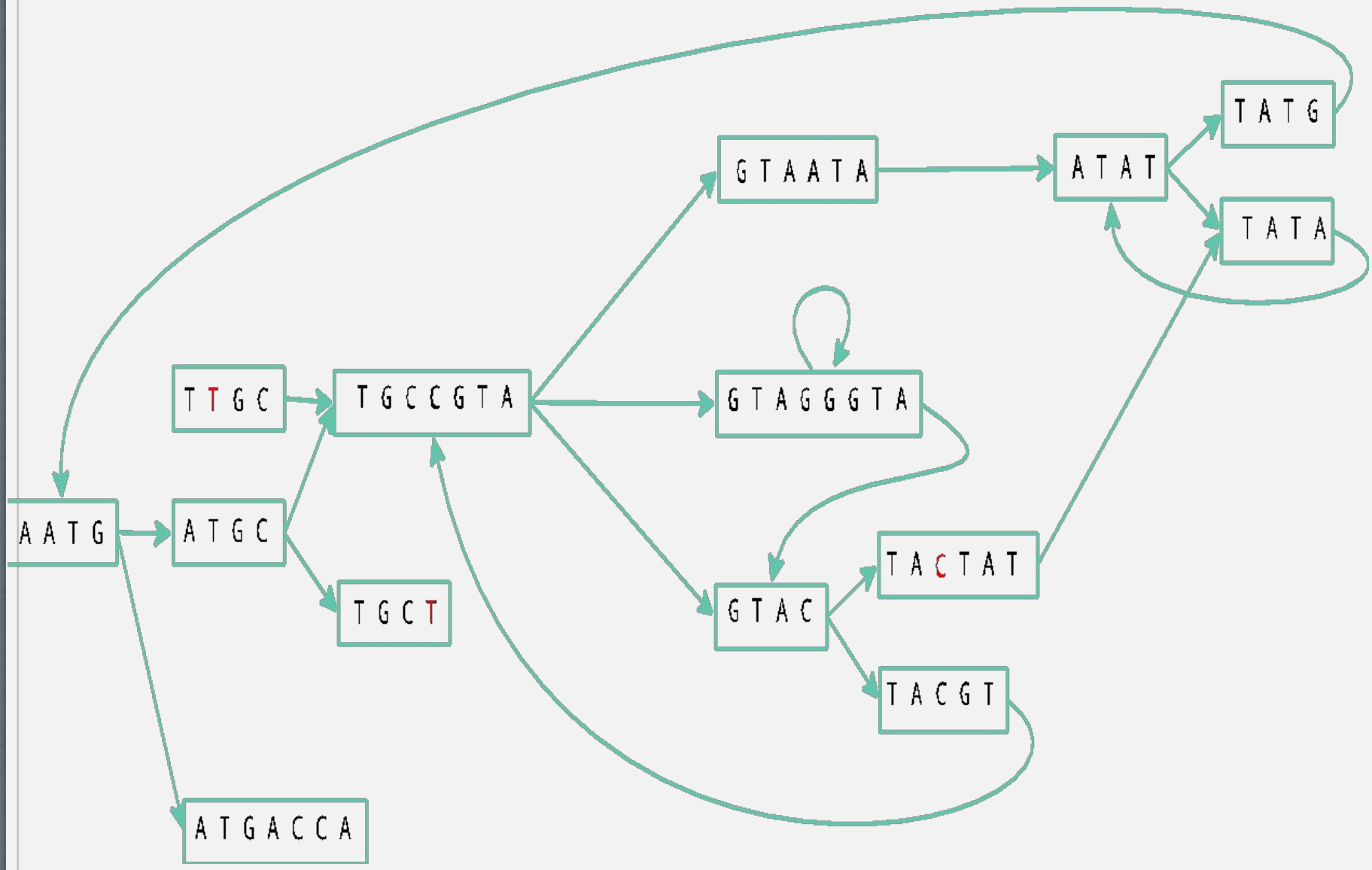
Construction

- K-mer length
  - Big debate: sensitivity vs specificity
  - Usually 21bp k-mer for 25bp length reads
- 2 Hash Tables
  - 1st: Originial k-mers, positions in reads, and complements
  - 2nd: k-mers that are not overlapped by other reads

Simplification

- Combine nodes that have only one outgoing and one ingoing node


AATG
ATGC
TGCT
TTGC
TGCC
GCCG
CCGT
CGTA
ATGA
TGAC
GACC
ACCA
GTAA
TAAT
AATA
ATAT
TATG
TATA
GTAG
TAGG
AGGG
GGGT
GGTA
GTAC
TACT
ACTA
CTAT
TACG
ACGT

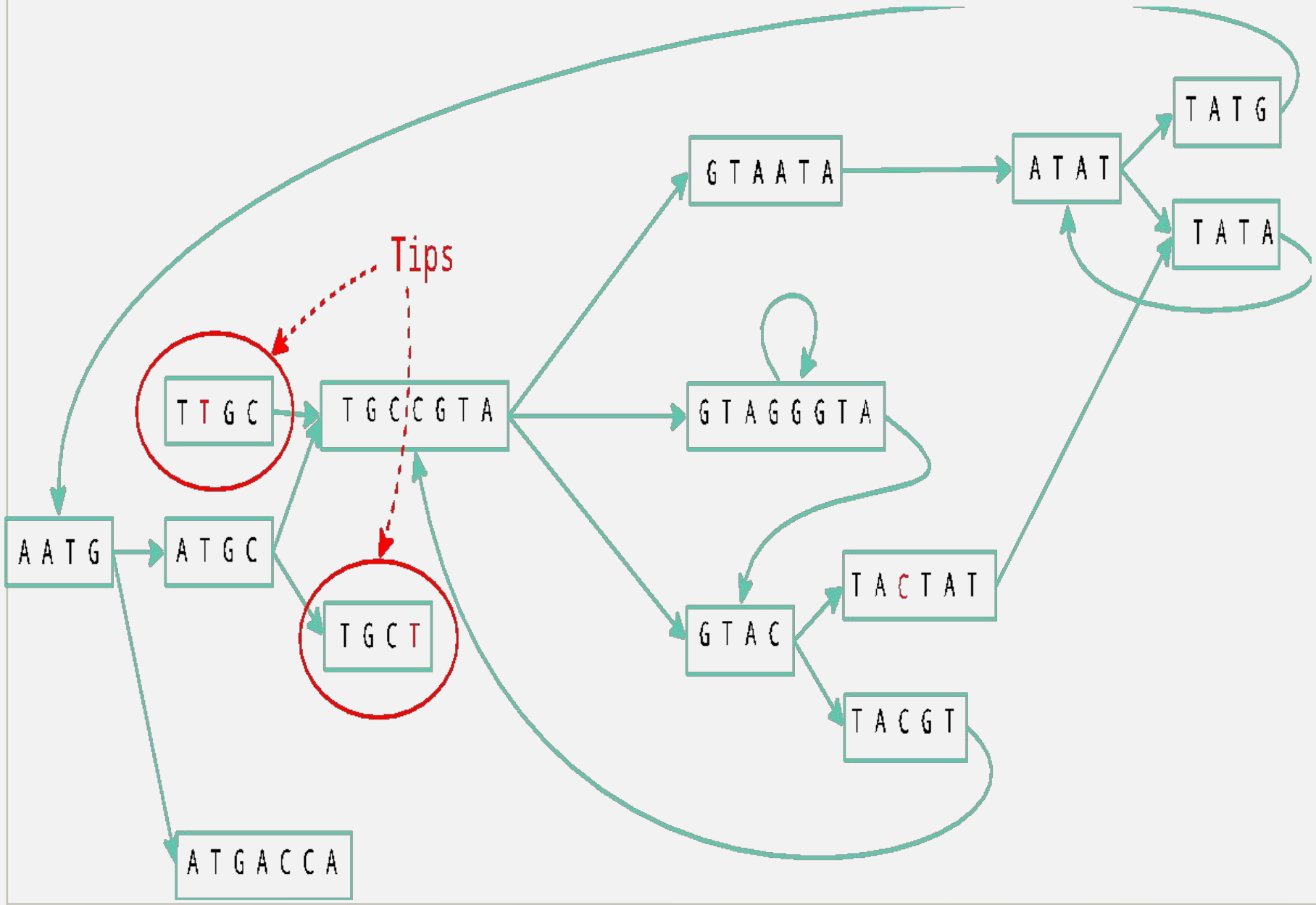

AATG
ATGC
TTGC
TGCCGTA
TGCT
ATGACCA
GTAATA
GTAGGGTA
GTAC
ATAT
TATG
TATA
TACTAT
TACGT

Error Removal

- Removing Tips
- Removing Bubbles
  - Tour Bus Algorithm

Tips

- Either do not have outgoing or do not have incoming node
- Remove if:
  - Shorter than 2k
  - The arc leading to the tip has multiplicity inferior to at least one of the other arcs connected to the junction node


Tips
AATG
ATGC
TTGC
TGCCGTA
TGCT
ATGACCA
GTAATA
GTAGGGTA
GTAC
TACTAT
TACGT
ATAT
TATG
TATA

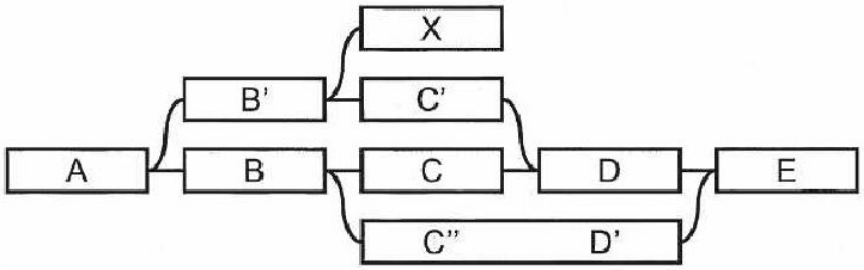
Bubbles

- Formed when two paths start and end at the same nodes
- Can be created by sequencing errors or biological mutations, such as SNPs
- Removed with “Tour Bus Algorithm”

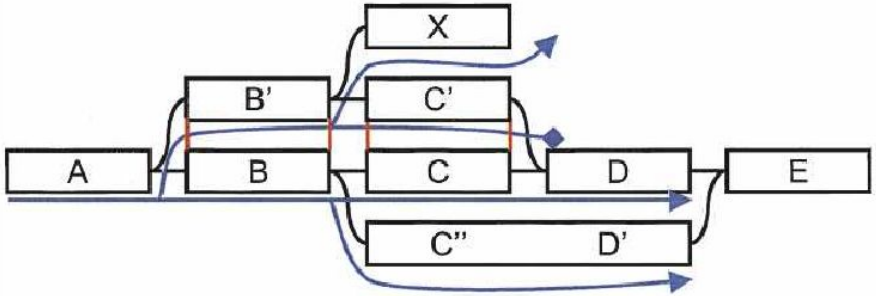
Tour Bus Algorithm

- When a bubble is encountered
  - Return to closest common ancestor
  - Extract sequences of bubble and align
  - If sequences are similar enough, merge the sequences
    - Path that reaches end node first, "shortest path," is used because it has higher coverage

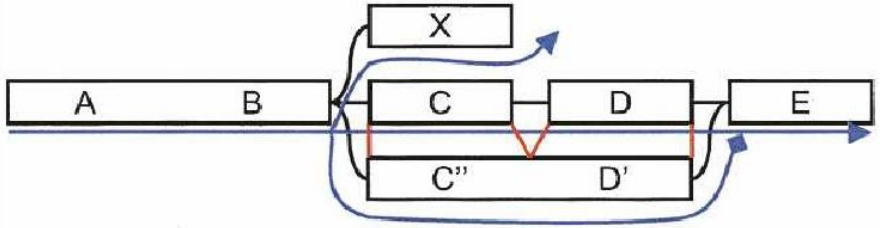
A



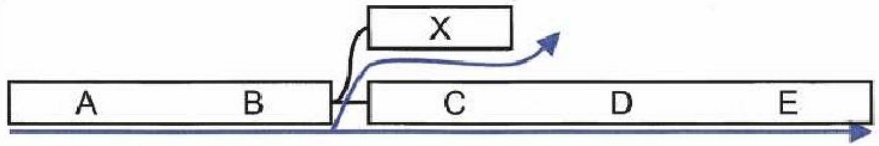
B



C



D



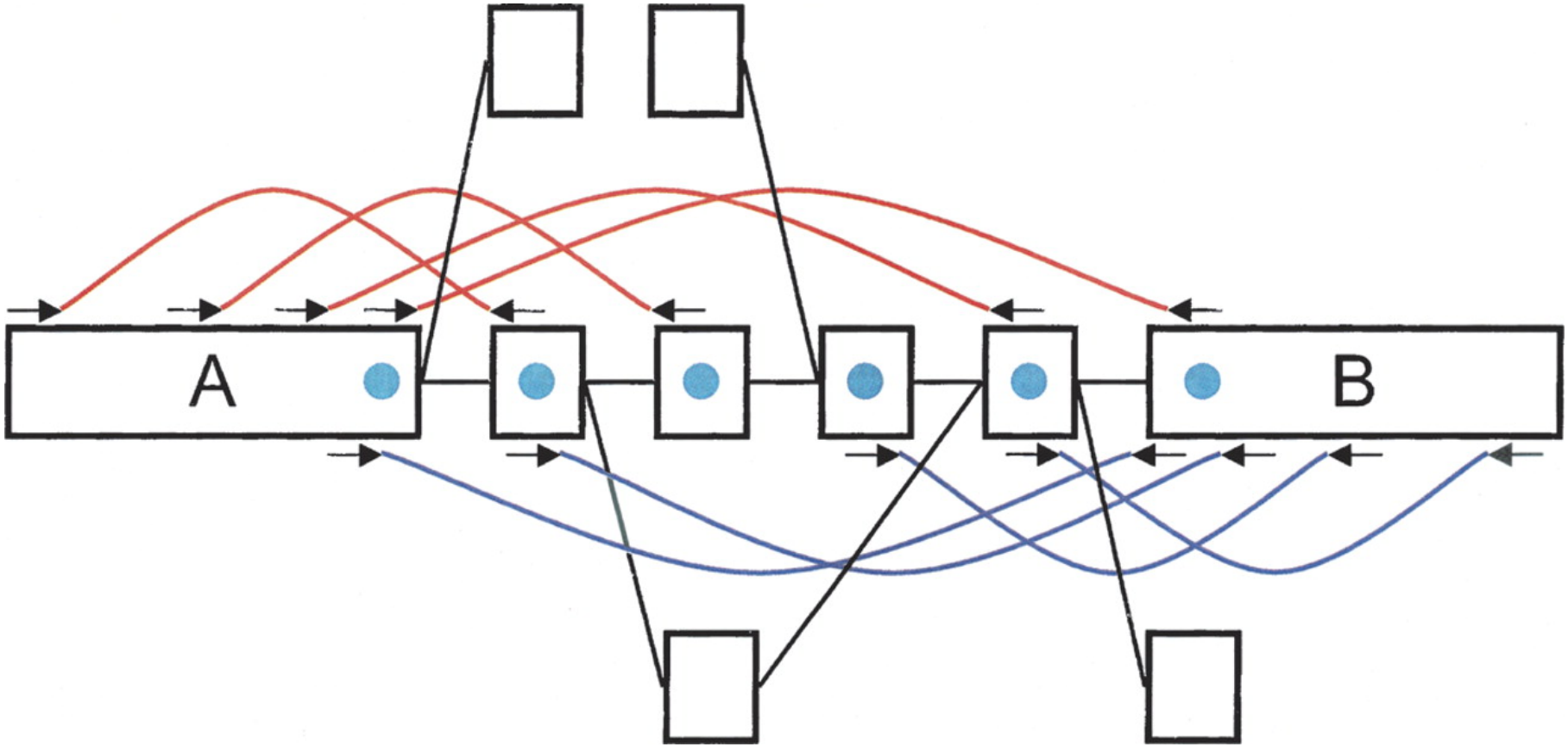
Removing Erroneous Connections

- The erroneous connections left after Tour Bus cannot be identified via the graph topology
  - Cannot find a recognizable loop or structure
  - Cannot be associated directly to a corresponding correct path
- Solution:
  - Create a coverage cutoff
  - Currently set by the user
- Concern: doesn't this undo Tour Bus's goal of preserving 'important' low coverage nodes?
  - No, Tour Bus turns those into long straight nodes which have near average coverage

Breadcrumb!

- Assembly is limited by the repeat structure in the sequenced genome
- We need to be able to resolve these repeats!
- We need to be able to connect contiguous regions through repeated regions, otherwise these repeats would create tangles in the de Bruijn graph.
- How do we do this? With Breadcrumb!

Breadcrumb algorithm.



Daniel R. Zerbino, and Ewan Birney Genome Res. 2008;18:821-829

The Breadcrumb Algorithm

- We start by selecting a cutoff length longer than most of the inserts, and designate “long nodes” to be the nodes that are longer than the cutoff.
- Using the read pairs, Breadcrumb starts by pairing up the long nodes. Because we did not set any restriction on uniqueness, some long nodes may consistently connect to several other long nodes, but they are simply flagged as ambiguous and left untouched. This selection eliminates some duplicated nodes, but not necessarily all of them.
- Select only unambiguous long nodes, Breadcrumb flags all the nodes containing the mate reads of the reads in that long node. If a single opposite long node is available, then all the nodes that pair up to it are also flagged. Because of the node length constraint, between two long contigs nearly all paired reads map onto a read on either of the long reads.
- Breadcrumb then extends the unique node by going as far as possible from one connected flagged node to the next and stopping if there are no connections or if the node is flagged. In the best case, a simple path can be found to the opposite long node, and the two contigs can be merged.

Breadcrum cont'd

- We must allow for erroneous reads!
- All reads detected while removing erroneous connections are marked unreliable.
- Long nodes may be erroneously connected to very few paired reads, in which case we discard the weak connections between long nodes.
- However, error also occurs in low complexity regions, so it is necessary to apply Tour Bus to the flagged nodes only, where we flag them instead of destroying them.

Time Complexity

- Main bottleneck is graph construction- the initializer graph for streptococcus needs 2 Gb of RAM alone!

- Tour bus uses a modified version of Dijkstra which has a time complexity of O(NlogN), where N is the number of nodes in the graph. This N depends on read coverage, error rate, and number of repeats.

- The runtime on breadcrumb is dependent on some subset K of N, where K is the size of the set of long nodes (remember that we set a threshold from which to select our K). It is a near constant amount of time per long node though, so its around O(K).

Why Velvet?

- There are other short read assemblers
- SSAKE and VCAKE explore a de Bruijn graph by searching for reads in a hash table.
- Since Velvet builds this graph, it uses more memory but tends to be much faster and produces longer contigs without misassembly!
- Velvet makes connected supercontigs out of very short reads. The connectivity of the supercontigs is super useful!