ATLAS to MetagenomeScope: Visualizing Metagenome Assembly Graphs

Todd J Treangen

Assistant Professor

Department of Computer Science – Rice University

Ready, set, download!

- **First**:
  - ./Chiron/bin/metacompass_interactive
  - python3 /opt/MetaCompass/go_metacompass.py

- **Second**:
  - cd /output
  - wget https://osf.io/xwk7m/download
  - wget https://osf.io/6dmh5/download

- **Third**:
  - wget http://tiny.cc/l65faz
  - wget http://tiny.cc/sa6faz

Docker details

- Where do I save my output?

/output

- How do I exit the container?

exit

- How do I access my output after exiting the container?

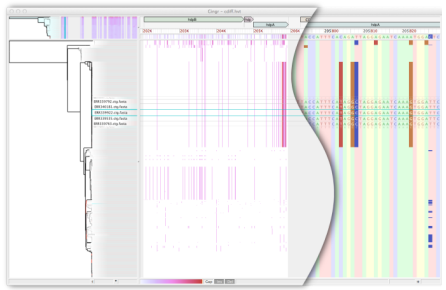
cd /home/stamps19/chiron/metacompass

Acknowledgements

- Mihai Pop (UMD): Metagenomics!
- Marcus Fedarko (UCSD): MetagenomeScope
- Victoria Cepeda (UMD): MetaCompass
- Jay Ghurye (Dovetail Genomics): MetaCarvel

Treangen Lab Members @Rice (Houston, TX)

Research background/interests



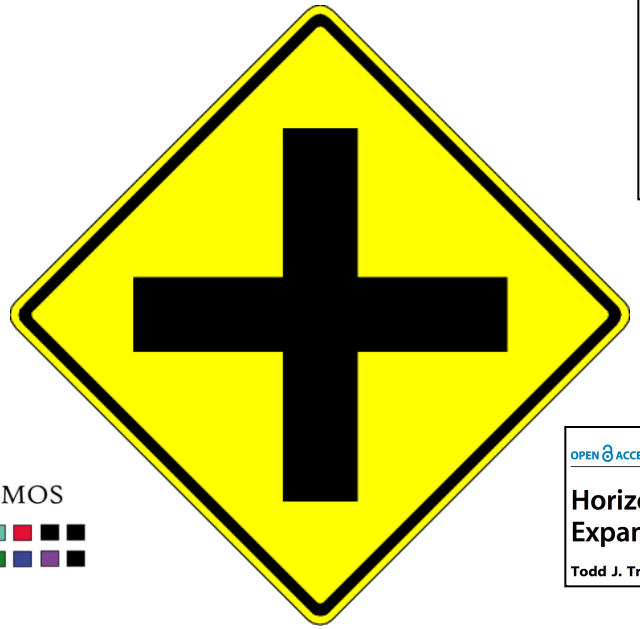
Pathogen Detection

RESEARCH ARTICLE

Identification and Genomic Analysis of a Novel Group C Orthobunyavirus Isolated from a Mosquito Captured near Iquitos, Peru

Todd J. Treangen[1]*, George Schoeler[2¤a], Adam M. Phillippy[1¤b], Nicholas H. Bergman[1], Michael J. Turell[3]

Software Engineering



Microbial Ecology & Evolution

:metAMOS:

a metagenomic assembly pipeline for AMOS

OPEN ACCESS Freely available online PLoS GENETICS

Horizontal Transfer, Not Duplication, Drives the Expansion of Protein Families in Prokaryotes

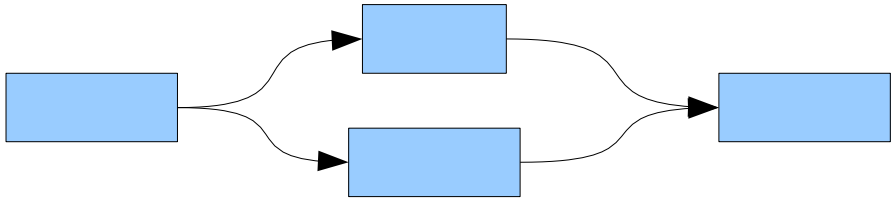
Todd J. Treangen[1,2,3¤]*, Eduardo P. C. Rocha[1,2,3]

Bioinformatics

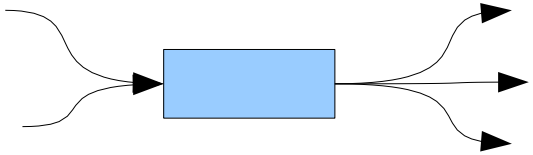
Goals for a metagenomic assembler

- Must work well for clonal data
  - handle repeats
  - handle errors
  - deal with low coverage regions

- Must deal with polymorphisms
  - distinguish between errors and polymorphisms



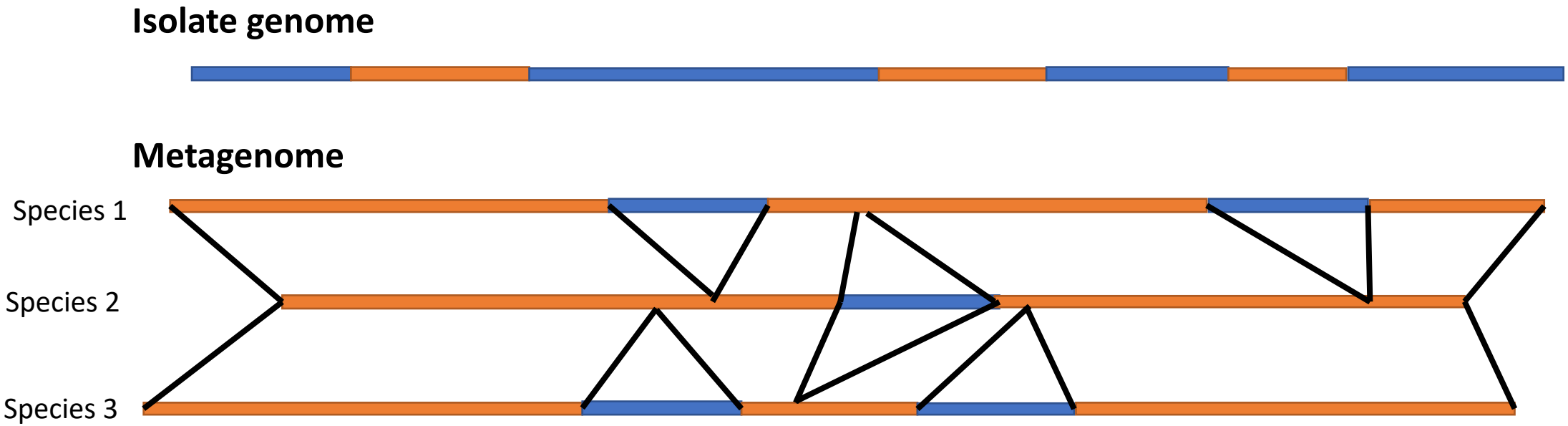
  - distinguish between repeats and polymorphisms



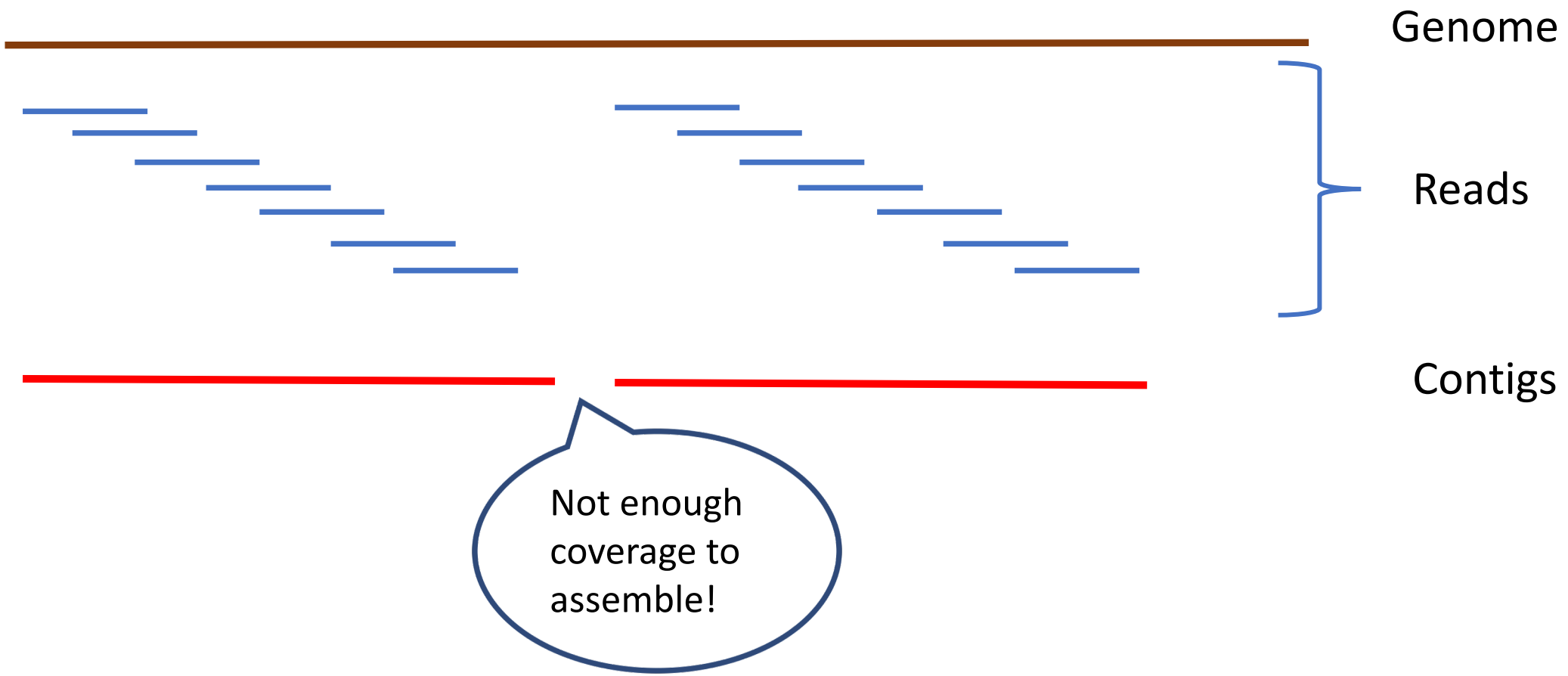
  - enable discovery of polymorphisms/variation

Repeats in metagenome

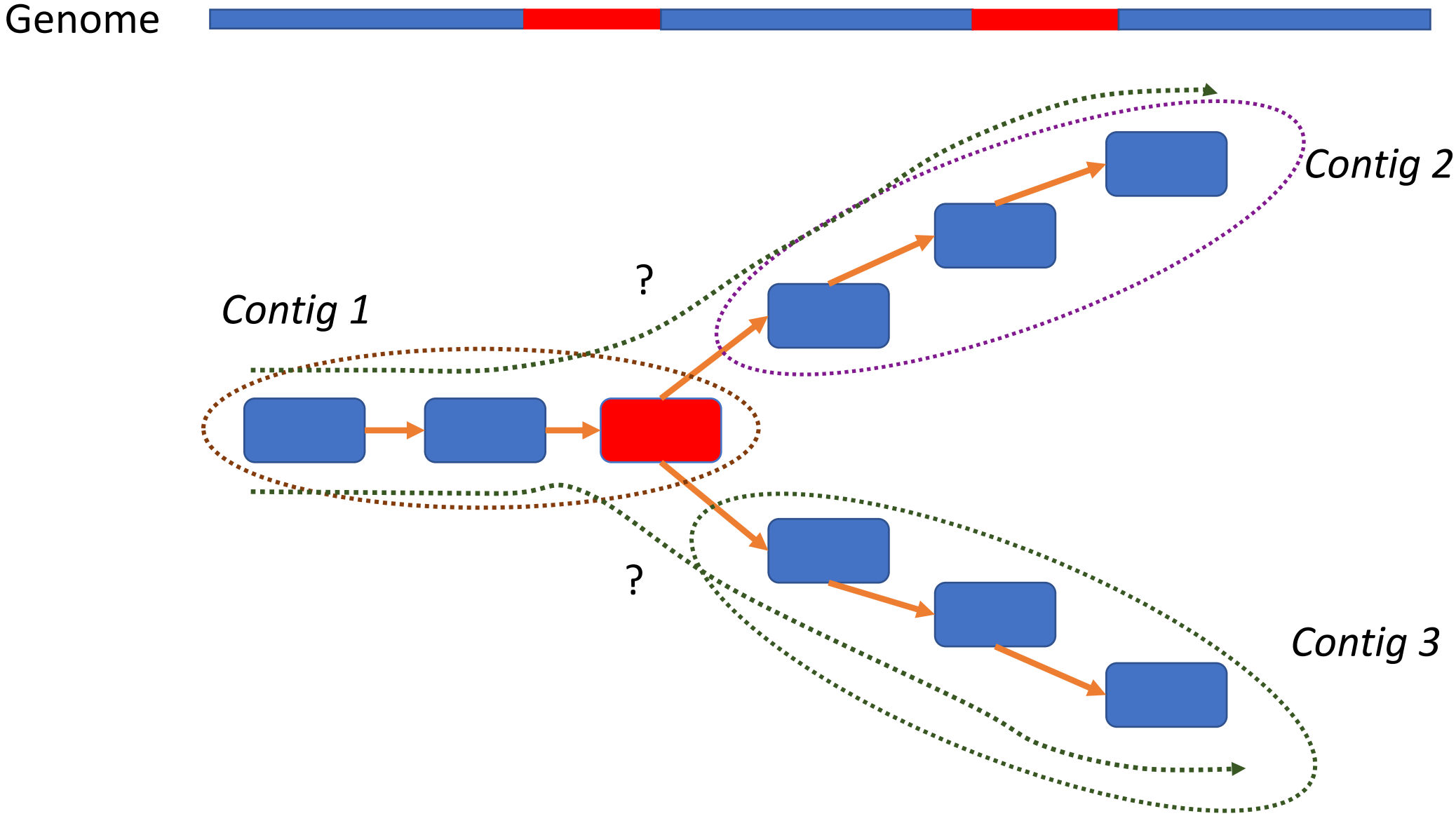
- High coverage regions -> abundant organisms
- Repeats are genome sized due to closely related strains in the sample



Why assemblers break contigs?

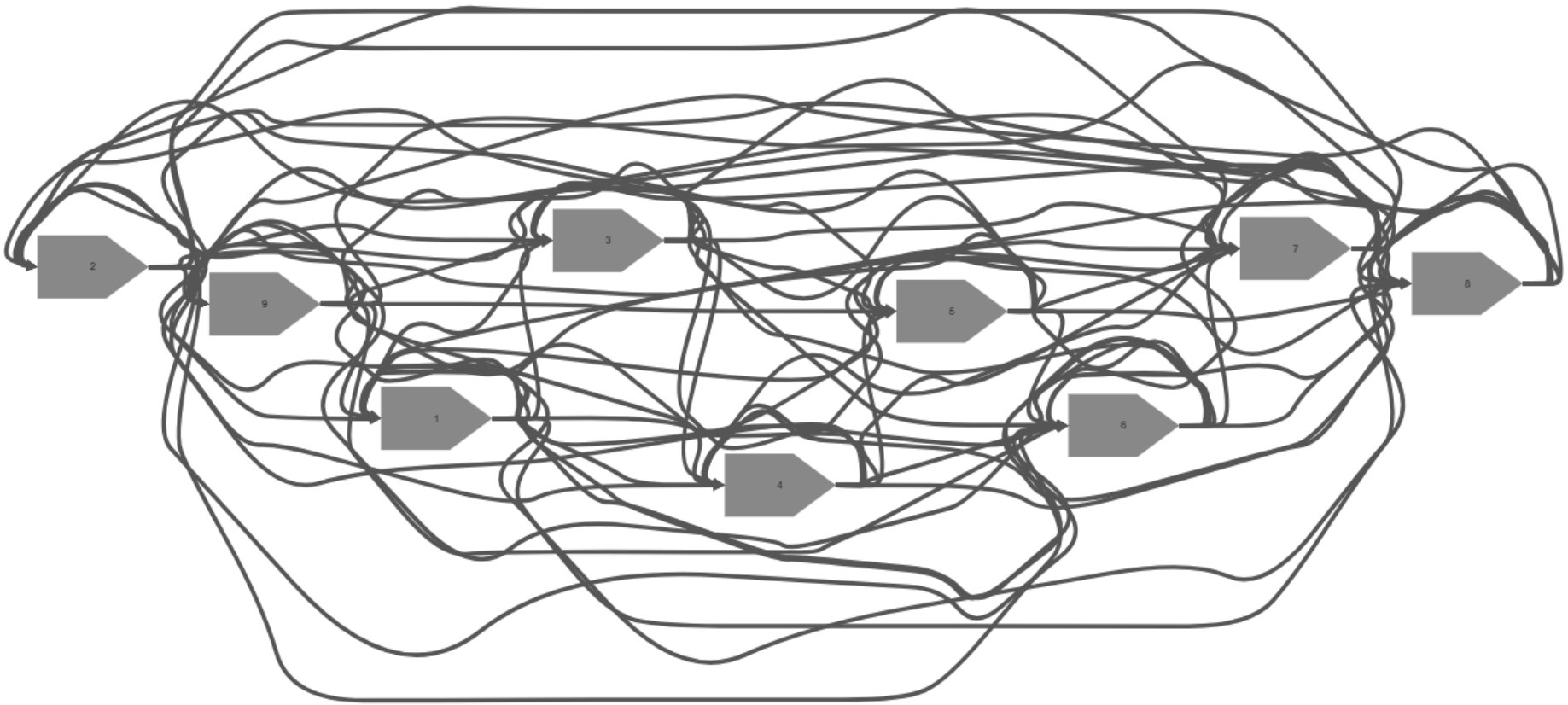


Why assemblers break contigs?



Example 1: AAAAAAAAAAAAAA (no unique 3-mer)

- 4bp reads:
  - (1) AAAA, (2) AAAA, (3) AAAA, (4) AAAA, (5) AAAA, (6) AAAA, (7) AAAA, (8) AAAA, (9) AAAA
- 3-mers:
  - AAA
- Overlaps:
  - (1)->(2),(1)->(3),(1)->(4),(1)->(5),(1)->(6),(1)->(7),(1)->8,(1)->(9)
  - (2)->(1),(2)->(3),(2)->(4),(2)->(5),(2)->(6),(2)->(7),(2)->8,(2)->(9)
  - ….
  - (9)->(1),(9)->(2),(9)->(3),(9)->(4),(9)->(5),(9)->(6),(9)->(7),(9)->(8)



Slide credits: MetagenomeScope

Example 2: AATCCGTTCGGA (no 3-mer repeats)

- 4bp reads:
  - (1) AATC, (2) ATCC, (3) TCCG, (4) CCGT, (5) CGTT, (6) GTTC, (7) TTCG, (8) TCGG, (9) CGGA
- 3-mers (10)
  - (i) AAT, (ii) ATC, (iii) TCC, (iv) CCG, (v) CGT, (vi) GTT, (vii) TTC, (viii) TCG, (ix) CGG, (x) GGA
- Overlaps
  - (1)->(2)
  - (2)->(3)
  - (3)->(4)
  - (4)->(5)
  - (5)->(6)
  - (6)->(7)
  - (7)->(8)
  - (8)->(9)

Slide credits: MetagenomeScope

```
AATCCGTTCGGA

AATC
 ATCC
  TCCG
   CCGT
    CGTT
     GTTC
      TTCG
       TCGG
        CGGA
```

Example 3: AATCCGTTCGGA (sequencing error)

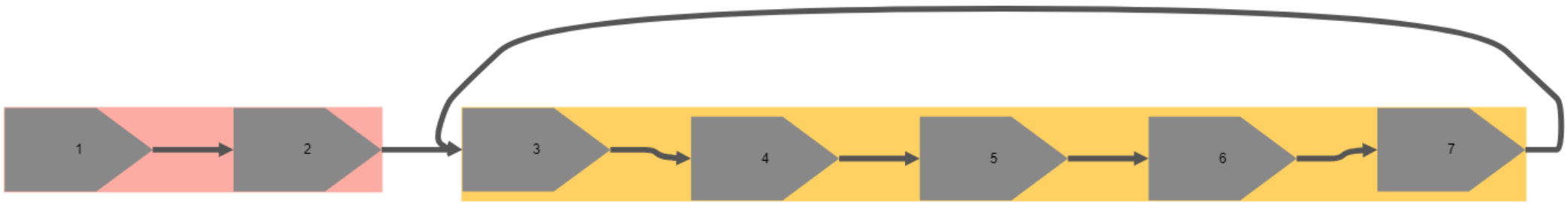
- 4bp reads:
  - (1) AAT**G**, (2) ATCC, (3) TCCG, (4) CCGA, (5) CGTT, (6) GTTC, (7) TTCG, (8) TCGG, (9) CGGA
- 3-mers (10)
  - (i) AAT, (ii) ATC, (iii) TCC, (iv) CCG, (v) ATG, (vi) CGT, (vii) GTT, (viii) TTC, (xi) TCG, (x) CGG, (xi) GGA
- Overlaps
  - ~~(1)->(2)~~
  - (2)->(3)
  - (3)->(4)
  - (4)->(5)
  - (5)->(6)
  - (6)->(7)
  - (7)->(8)
  - (8)->(9)

```
ATCCGTTCGGA


ATCC
 TCCG
  CCGT
   CGTT
    GTTC
     TTCG
      TCGG
       CGGA
```

Example 4: AATCCGTTCGGA (sequencing error)

- 4bp reads:
  - (1) AATC, (2) ATCC, (3) TCCG, (4) CCGT, (5) CGTT, (6) GTTC, (7) TTC**C**, (8) TCGG, (9) CGGA
- 3-mers (10)
  - (i) AAT, (ii) ATC, (iii) TCC (X2), (iv) CCG, (v) CGT, (vi) GTT, (vii) TTC, (viii) TCG, (ix) CGG, (x) GGA
- Overlaps
  - (1)->(2)
  - (2)->(3)
  - (3)->(4)
  - (4)->(5)
  - (5)->(6)
  - (6)->(7)
  - ~~(7)->(8)~~
  - (7)->(3)
  - (8)->(9)



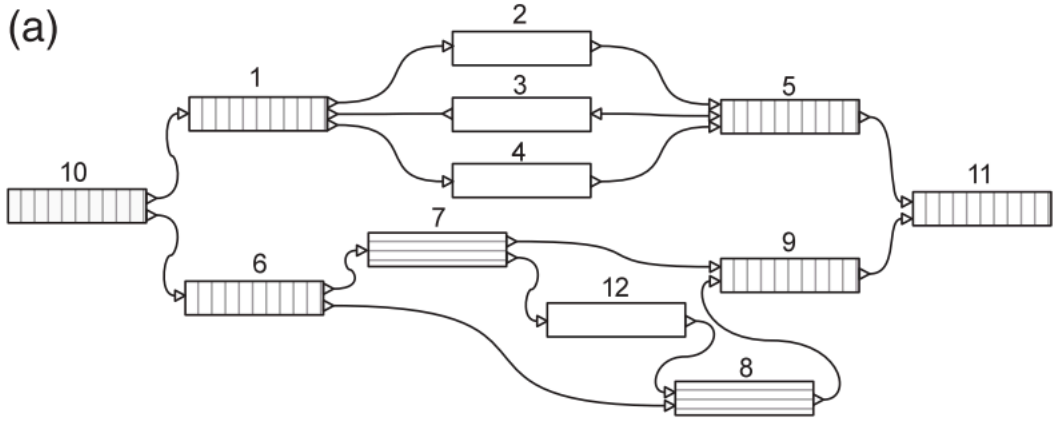
Slide credits: MetagenomeScope

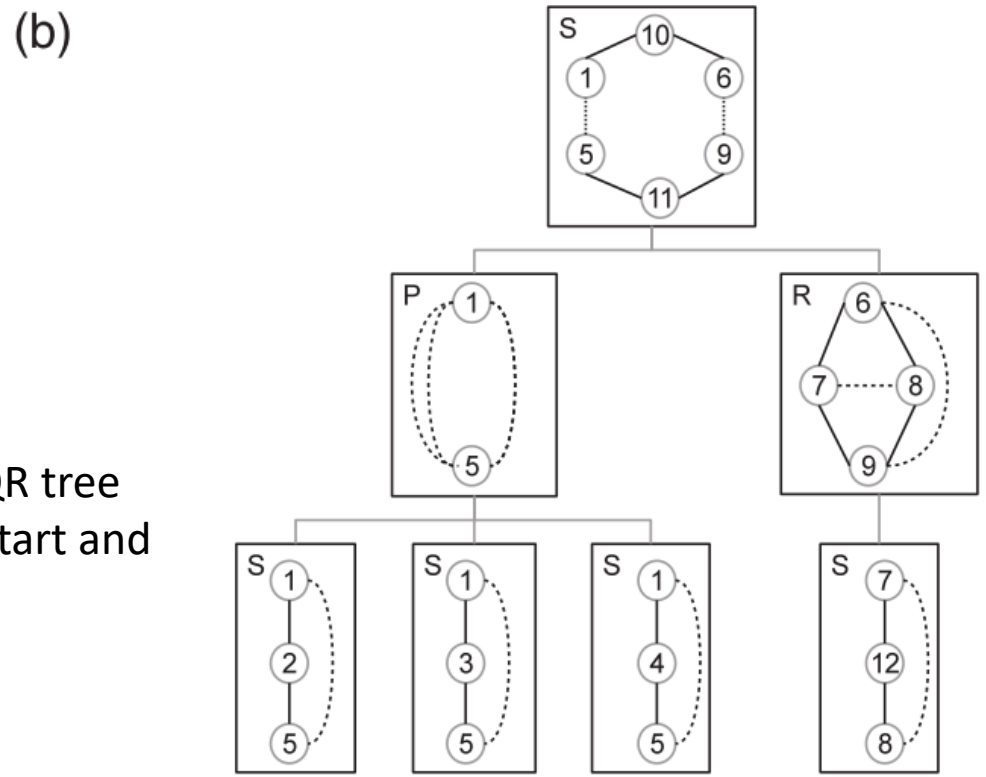
Example 5: AATCCGTTCGGA (coverage gap)

- 4bp reads:
  - (1) AATC, (2) ATCC, (3) TCCG, ~~(4) CCGT, (5) CGTT~~, (6) GTTC, (7) TTCG, (8) TCGG, (9) CGGA
- 3-mers (10)
  - (i) AAT, (ii) ATC, (iii) TCC, (iv) CCG, (v) TTC, (vi) TCG, (vii) CGG, (viii) GGA
- Overlaps
  - (1)->(2)
  - (2)->(3)
  - ~~(3)->(4)~~
  - ~~(4)->(5)~~
  - ~~(5)->(6)~~
  - (6)->(7)
  - (7)->(8)
  - (8)->(9)

Variant Detection Using Graph Topology

SPQR Tree





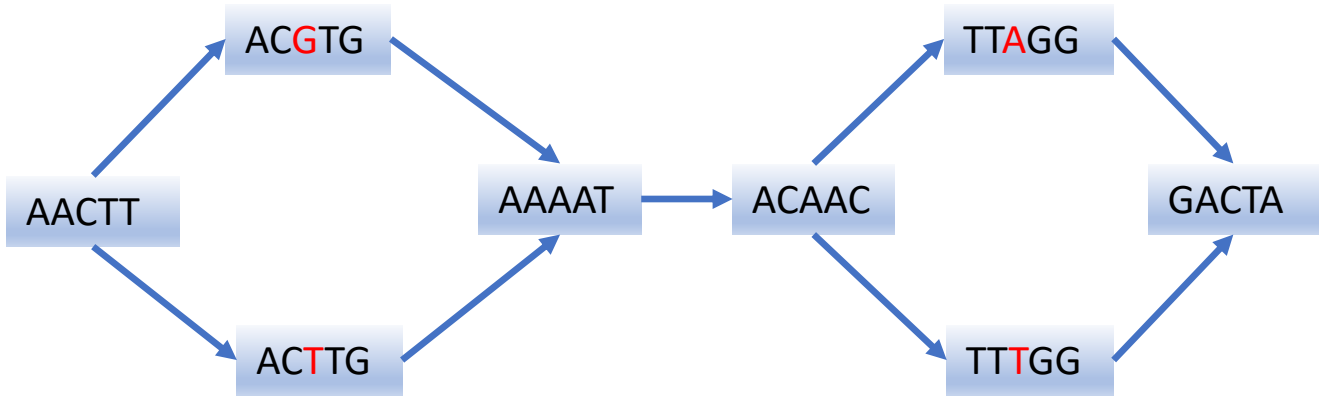
If two nodes lie in the same node of SPQR tree and share a virtual edge, then they are start and end of a bubble!

Figure credits: Nijkamp et al., 2013

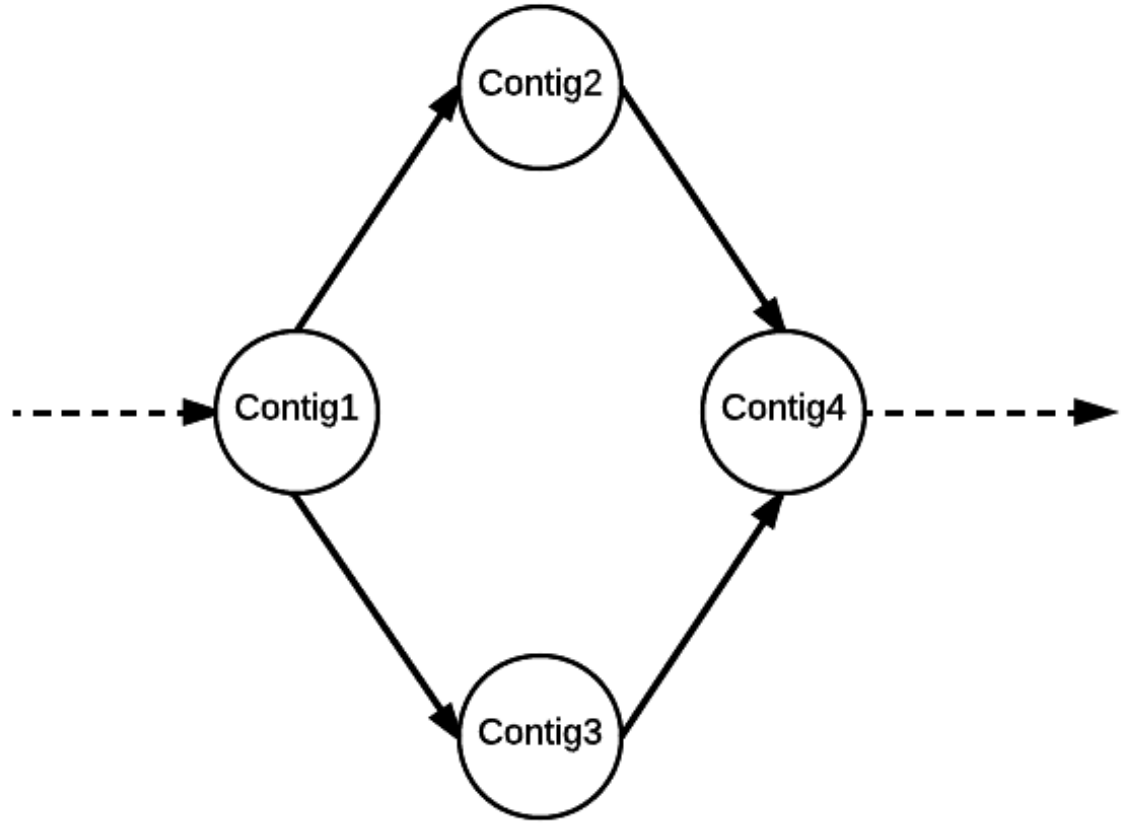
Variant detection using graph topology

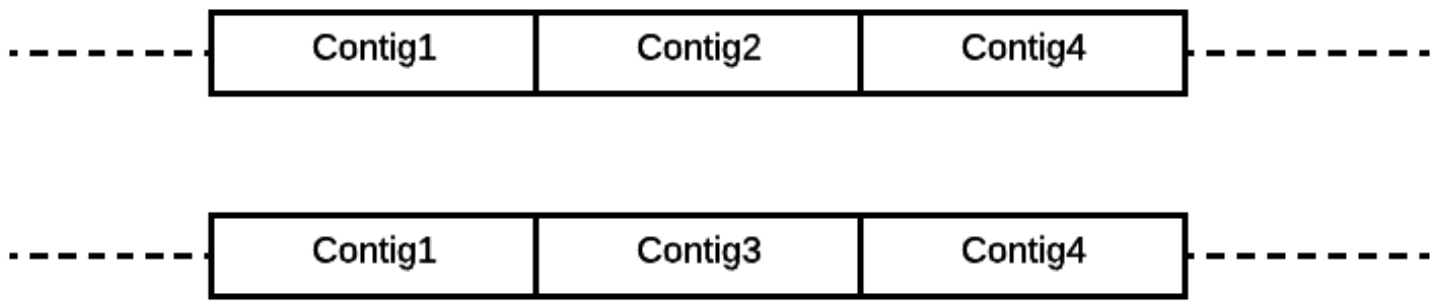
Species 1 …AACTTACGTGAAAATACAACTTAGGACTA…

Species 2 …AACTTACTTGAAAATACAACTTTGGACTA..

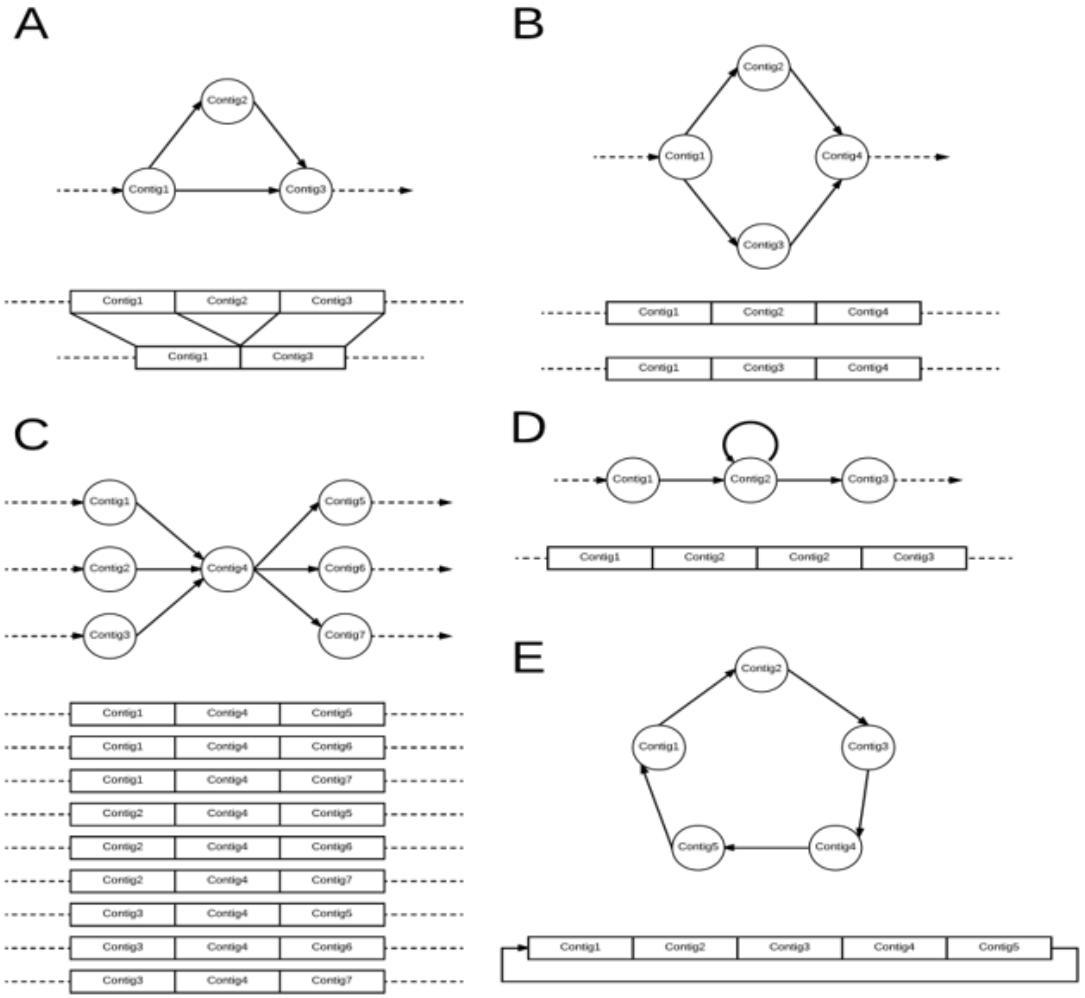


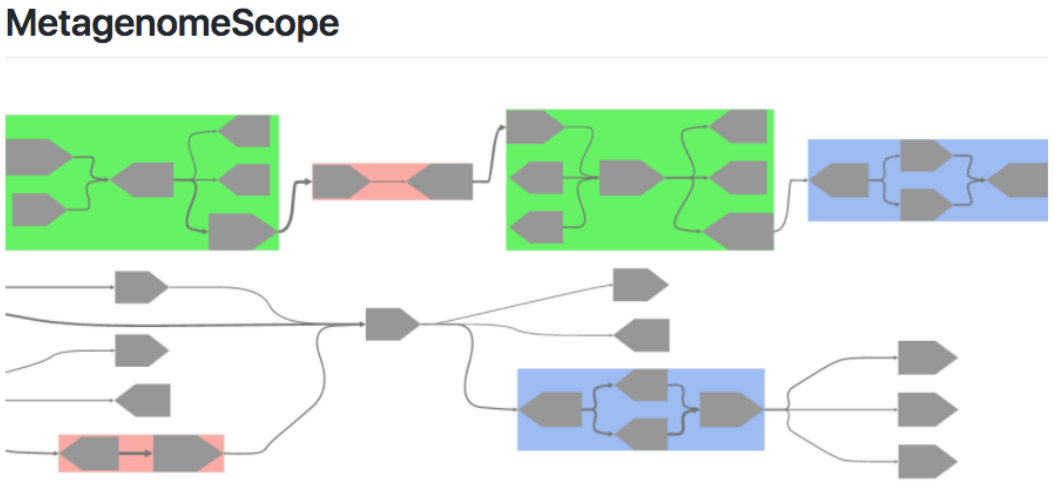
- Variations in the sequences cause bubble in the graph
- Naïve search for all bubble is exponential in the number of nodes
- Use a SPQR tree based approach to find out bubbles (Nijkamp et al. 2013)
- Runs in O(V+E) time (Gutwenger et al. 2001)


Contig2
Contig1
Contig4
Contig3


Contig1
Contig2
Contig4
Contig1
Contig3
Contig4

MetaCarvel and MetagenomeScope





Overview of the different types of variants detected by MetaCarvel.

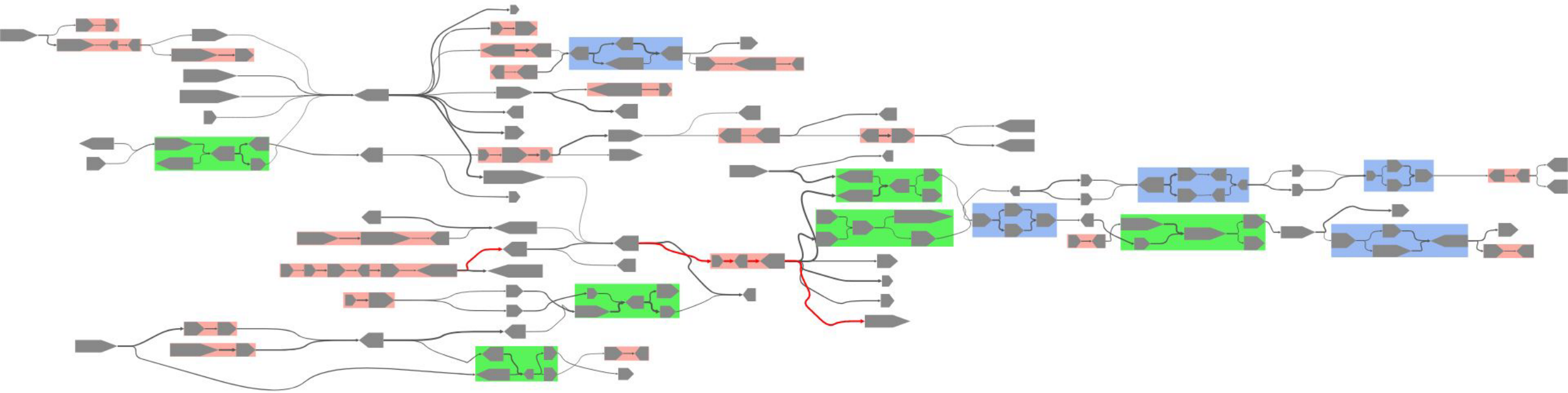
We focus on five types of variants:
A) Triangular bubbles,
B) Simple four bubbles
C) High centrality nodes
D) Simple Cycles, and
E) Multinode cycles

https://github.com/marbl/MetaCarvel

https://github.com/marbl/MetagenomeScope

Assembly Graphs

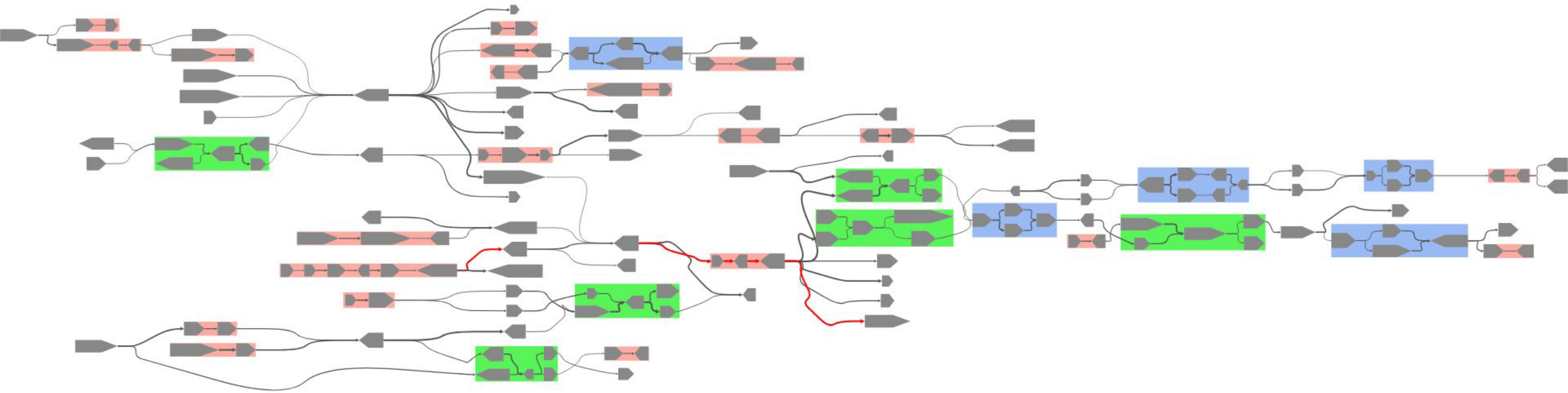
(Meta)genome assembly can be represented as a graph traversal



Assembly Graphs

(either a **de Bruijn graph** or an **overlap graph)**

(Meta)genome assembly can be represented as a graph traversal

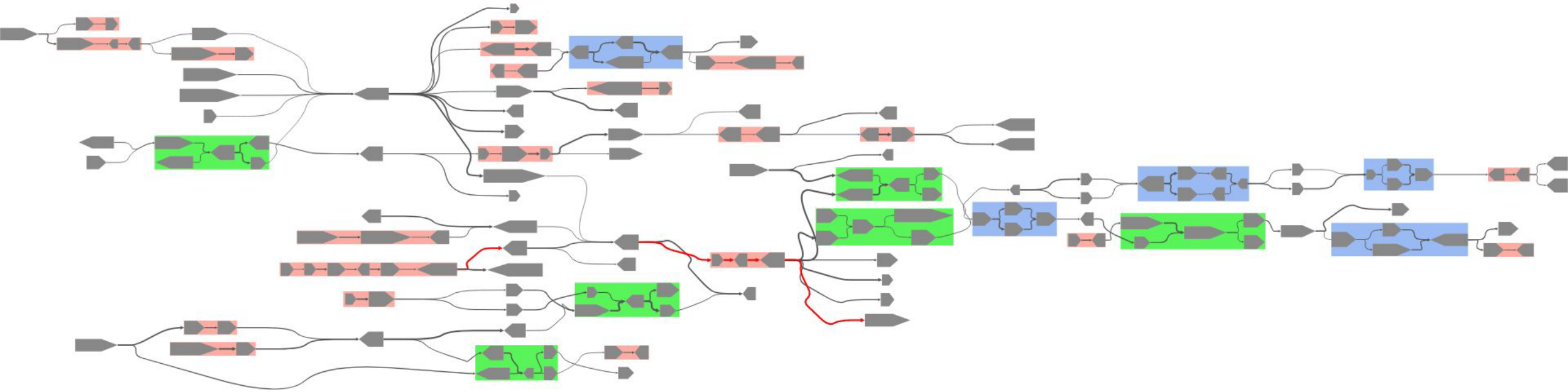


Assembly Graphs

(either a **de Bruijn graph** or an **overlap graph)**

(Meta)genome assembly can be represented as a graph traversal

In either case...



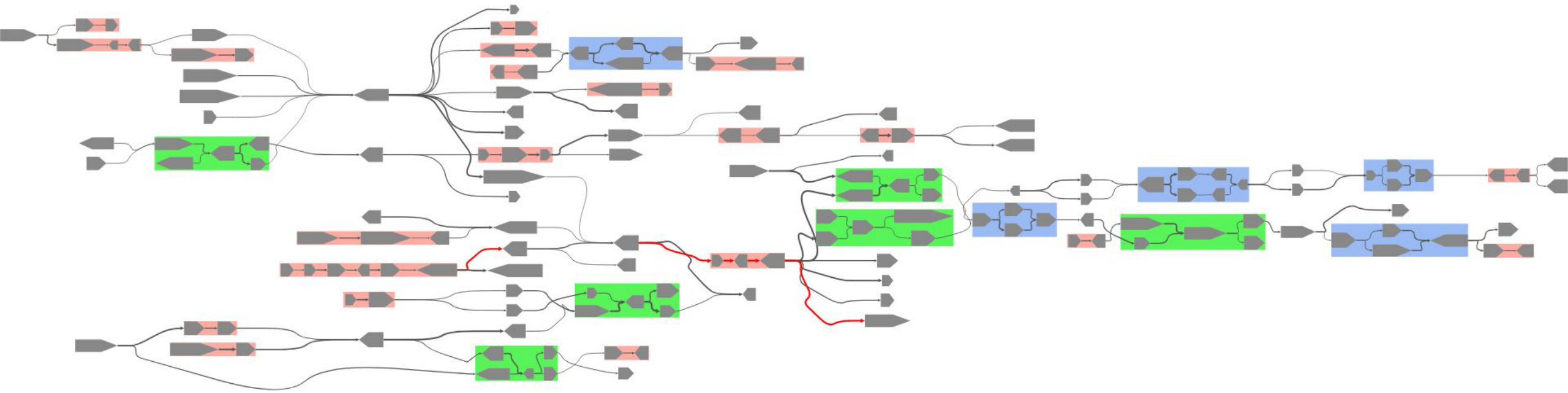
Assembly Graphs

(either a **de Bruijn graph** or an **overlap graph)**

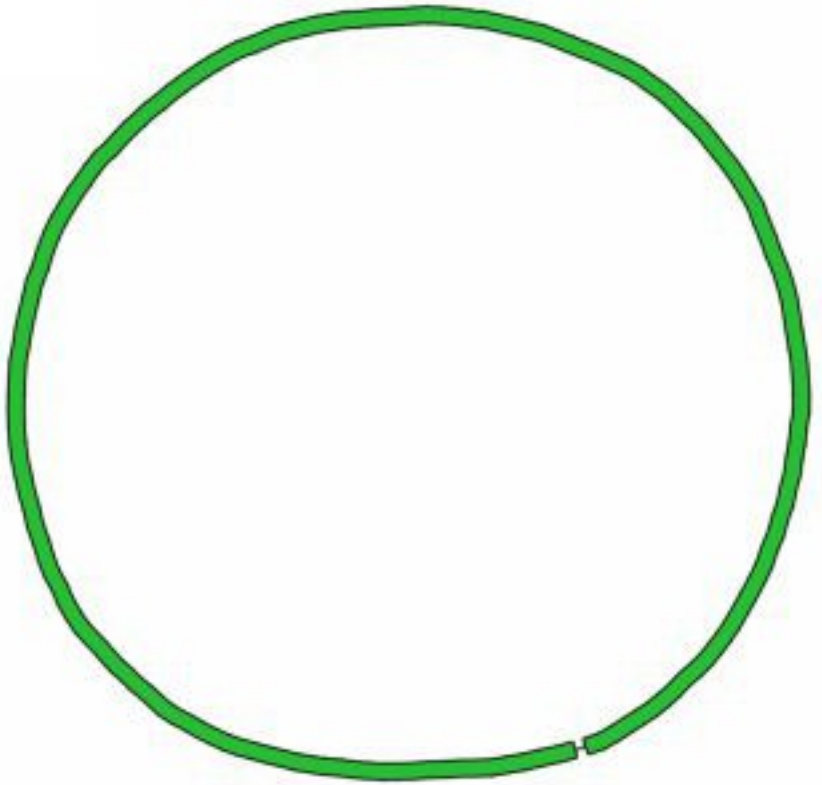
(Meta)genome assembly can be represented as a graph traversal

Nodes (*contigs*) - fragments of DNA obtained from assembly

Edges - overlaps between contigs' sequences



Assembly Graphs: The Good



Ideal genome assembly graph:

One node per sequence, maybe with an edge to itself

Ideal metagenome assembly graph:

c graphs, where each is an “ideal genome assembly graph”

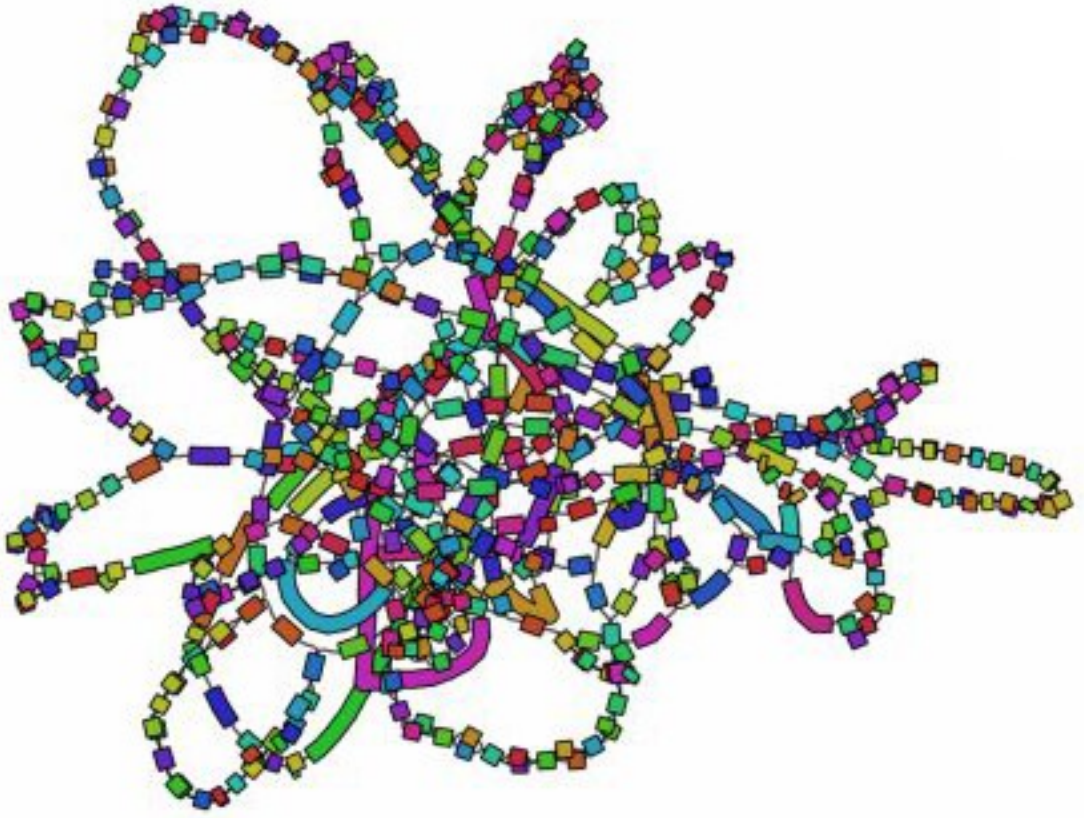
(assuming the input metagenome consists of *c* genomes)

These figures c/o the Bandage manuscript (Wick et al. 2015)

Assembly Graphs: The Bad /ugly

Actual genome assembly graph:

Lots of possible paths due to complexities like repeats, mutations, assembly errors, ...



Actual metagenome assembly graph:

More potential for these complexities to “compound” → even more fragmented

These figures c/o the Bandage manuscript (Wick et al. 2015)

Problem

How can we visualize metagenome assembly graphs in a way that highlights both

1. **small-scale details**, and
2. **large-scale structure**?

Your turn!

- https://gitlab.com/treangen/stamps2019/README.md#part1

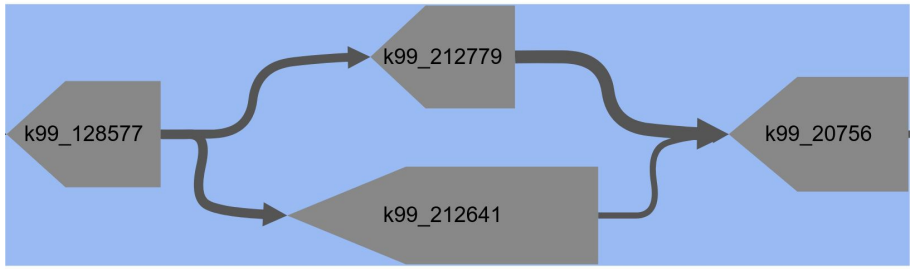
Exercise #1: Drawing Something

1. Open MetagenomeScope's viewer interface at mgsc.umiacs.io.
2. Pick a demo assembly graph using the "Demo .db" button.
3. Draw one of the graph's connected components[1]!

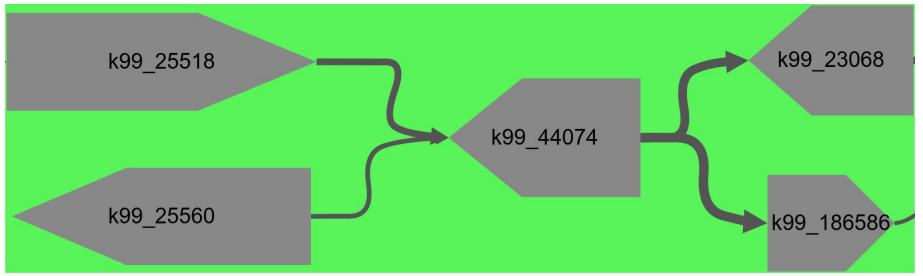
[1] "Weakly" connected components

Identifying Structural Patterns in Assembly Graphs

(See Miller, Koren, Sutton 2010)

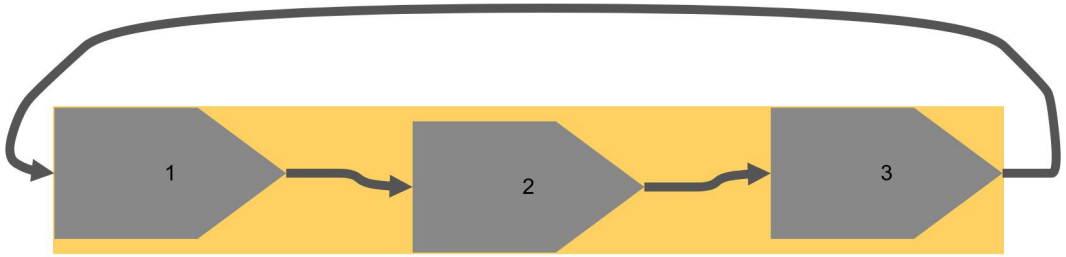


Bubble



Frayed Rope

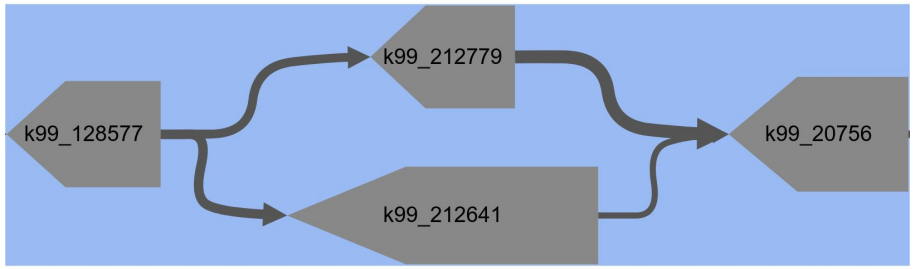
Chain



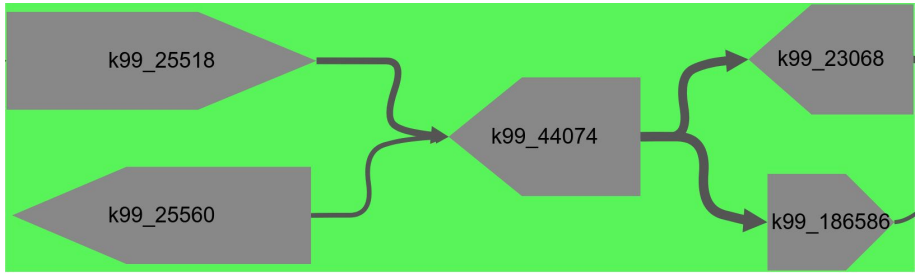
Cyclic Chain

Identifying Structural Patterns in Assembly Graphs

(See Miller, Koren, Sutton 2010)

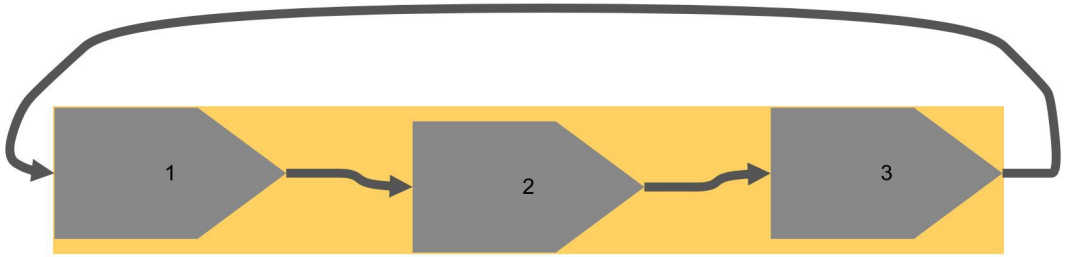


Bubble

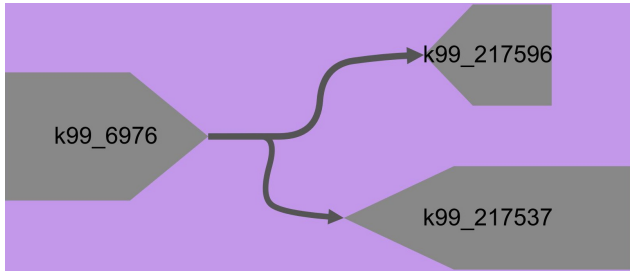


Frayed Rope

Chain



Cyclic Chain

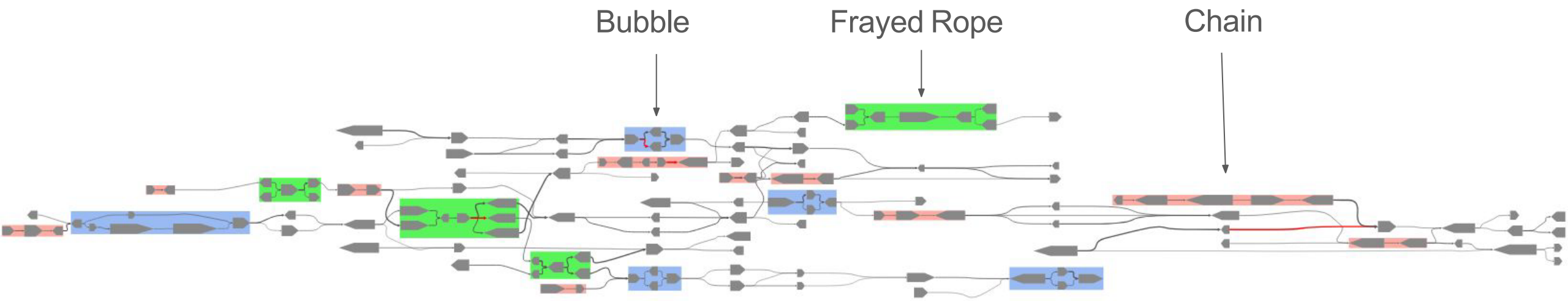


User-Defined Structures

Identifying Structural Patterns in Assembly Graphs

(See Miller, Koren, Sutton 2010)

Patterns can be collapsed/uncollapsed to decrease/increase graph complexity.

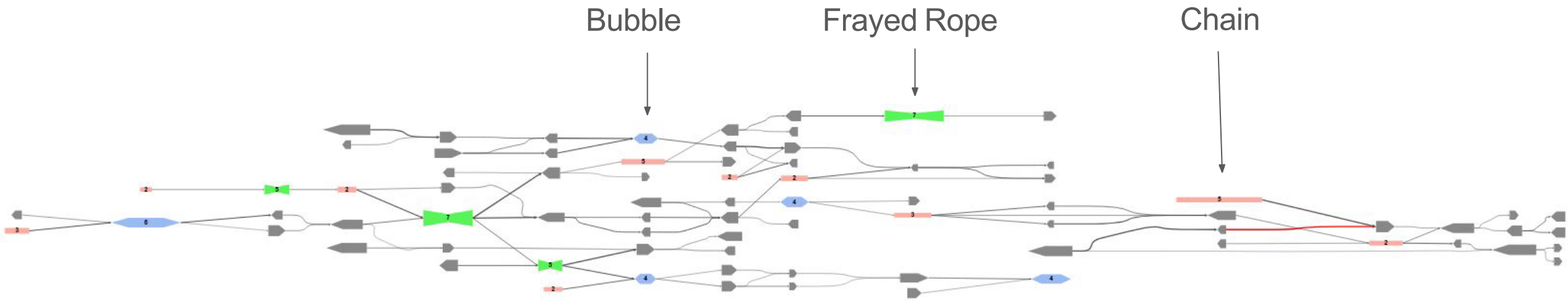


region of an assembly graph created from Human Microbiome Project (HMP) data, accession ID SRS049959

Identifying Structural Patterns in Assembly Graphs

(See Miller, Koren, Sutton 2010)

Patterns can be collapsed/uncollapsed to decrease/increase graph complexity.

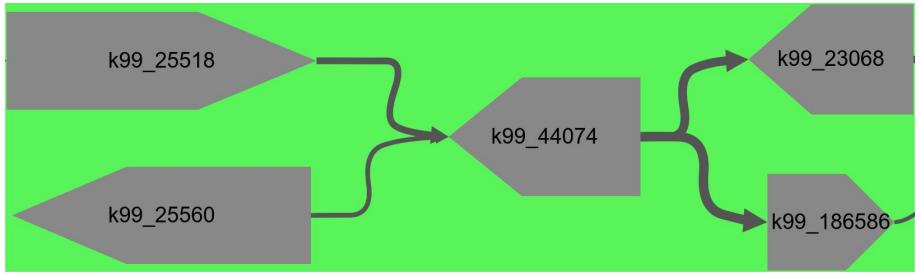


region of an assembly graph created from Human Microbiome Project (HMP) data, accession ID SRS049959

Exercise #2: Finding a frayed rope

https://gitlab.com/treangen/stamps2019

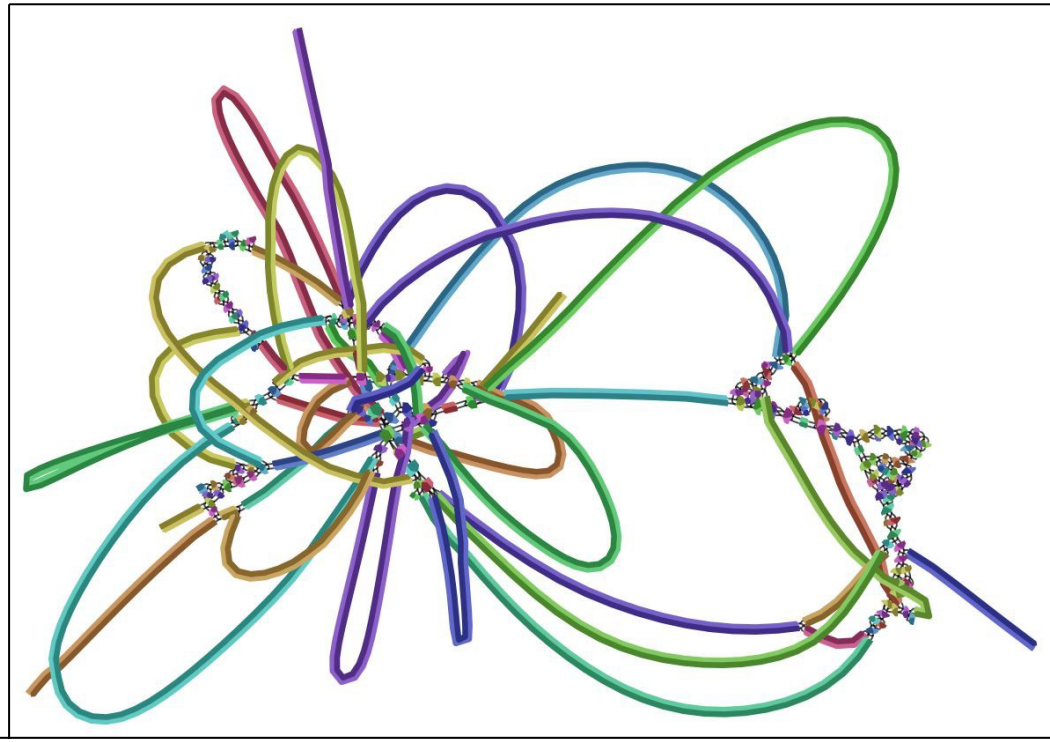
(this sort of thing)



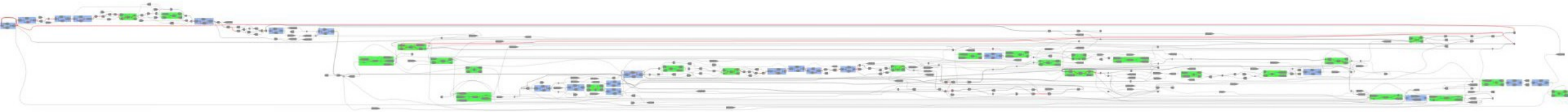
Hierarchical Graph Layout

Bandage, ABySS-Explorer, and Ray Cloud Browser all use **force-directed** layout algorithms

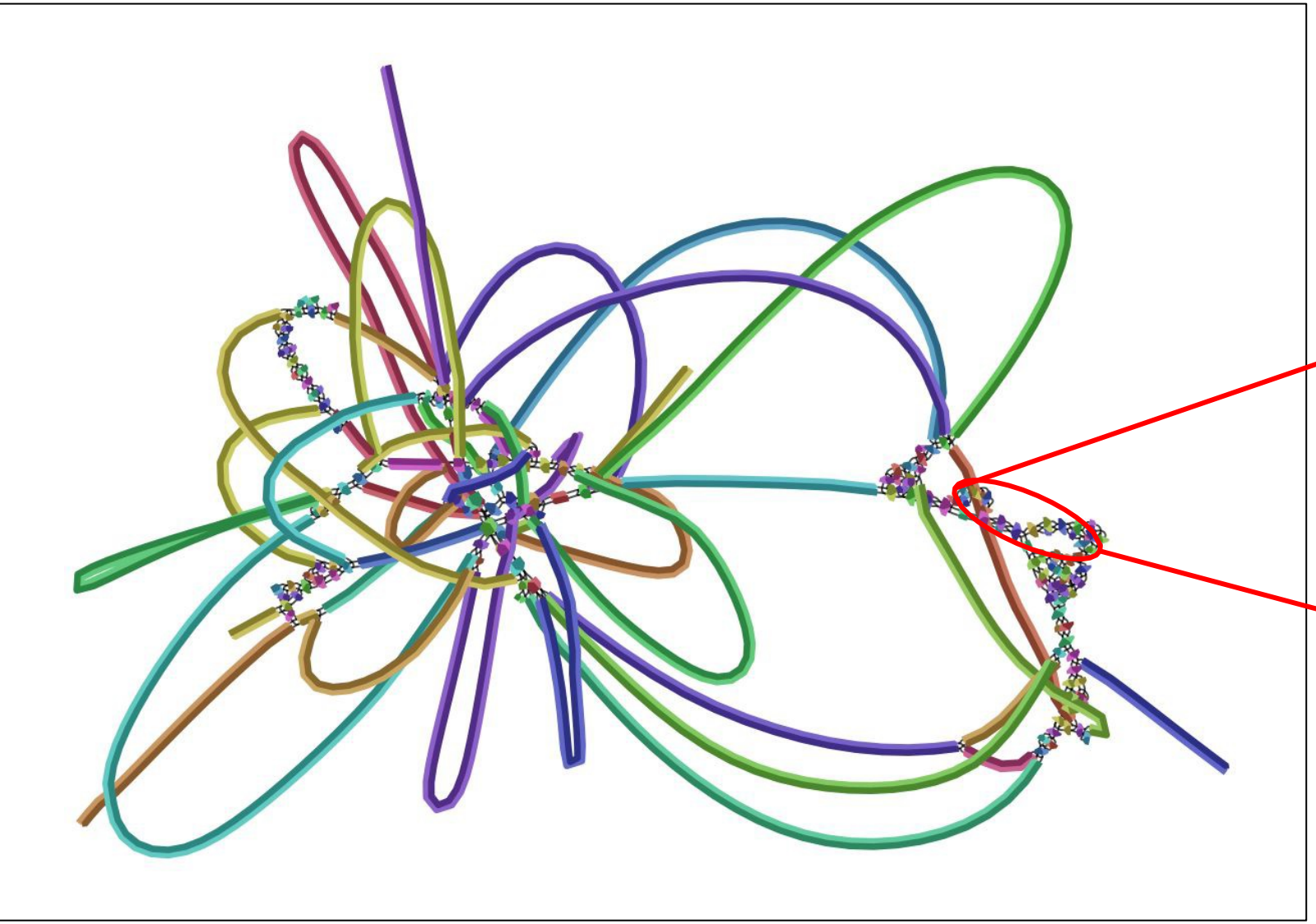
MetagenomeScope uses Graphviz' dot program to **hierarchically** lay out graphs

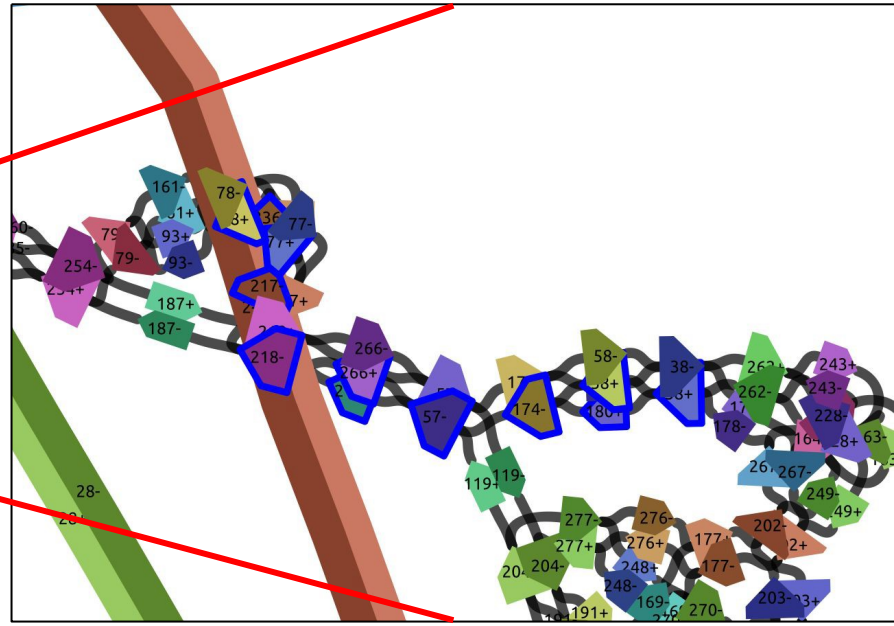


Largest connected component of an E. coli assembly graph produced with Velvet, visualized in Bandage (right) and MetagenomeScope (bottom)

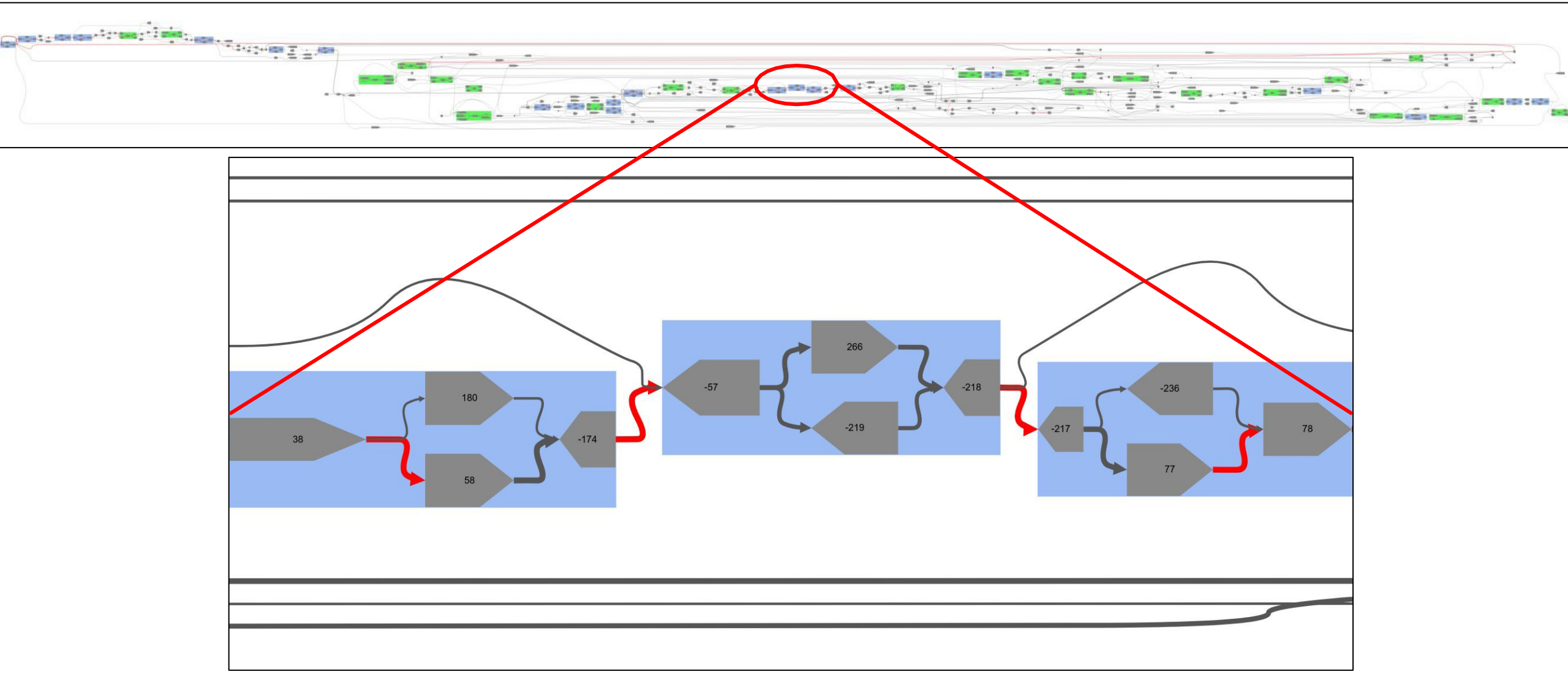


Reviewing that E. coli assembly graph (Bandage)



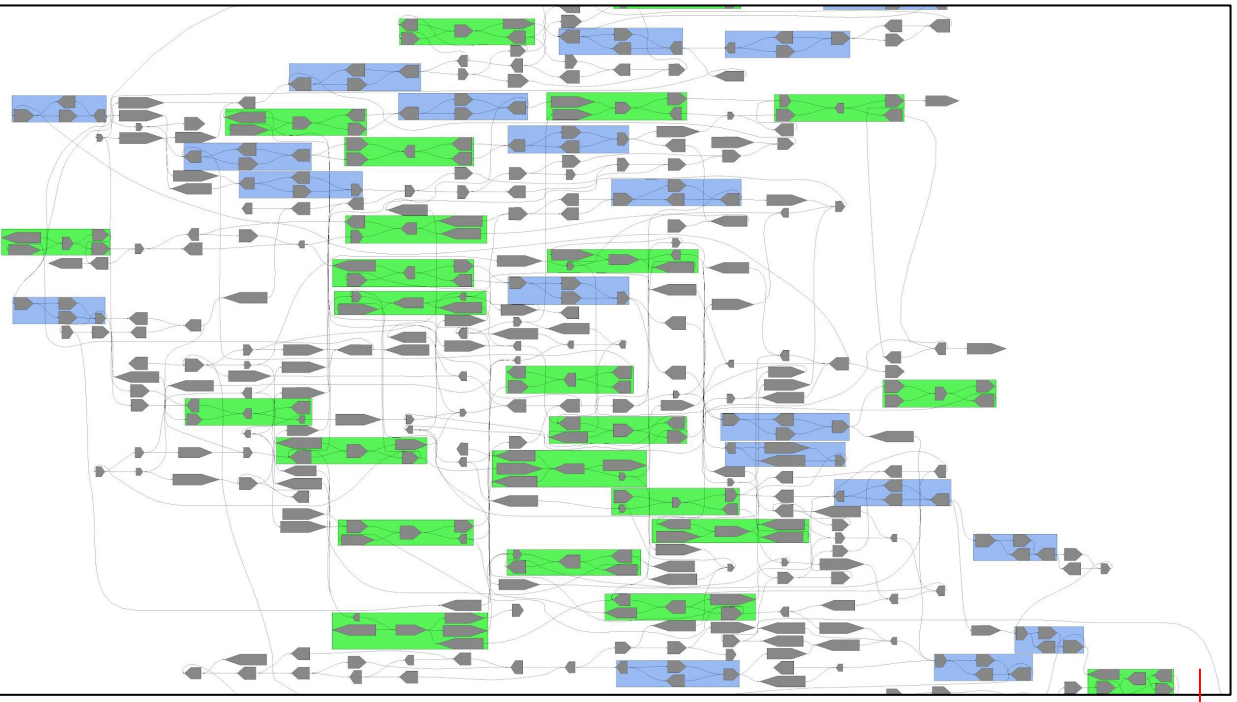


Reviewing that E. coli assembly graph (MetagenomeScope)



Lunch break

Structural Patterns and Layout Linearization



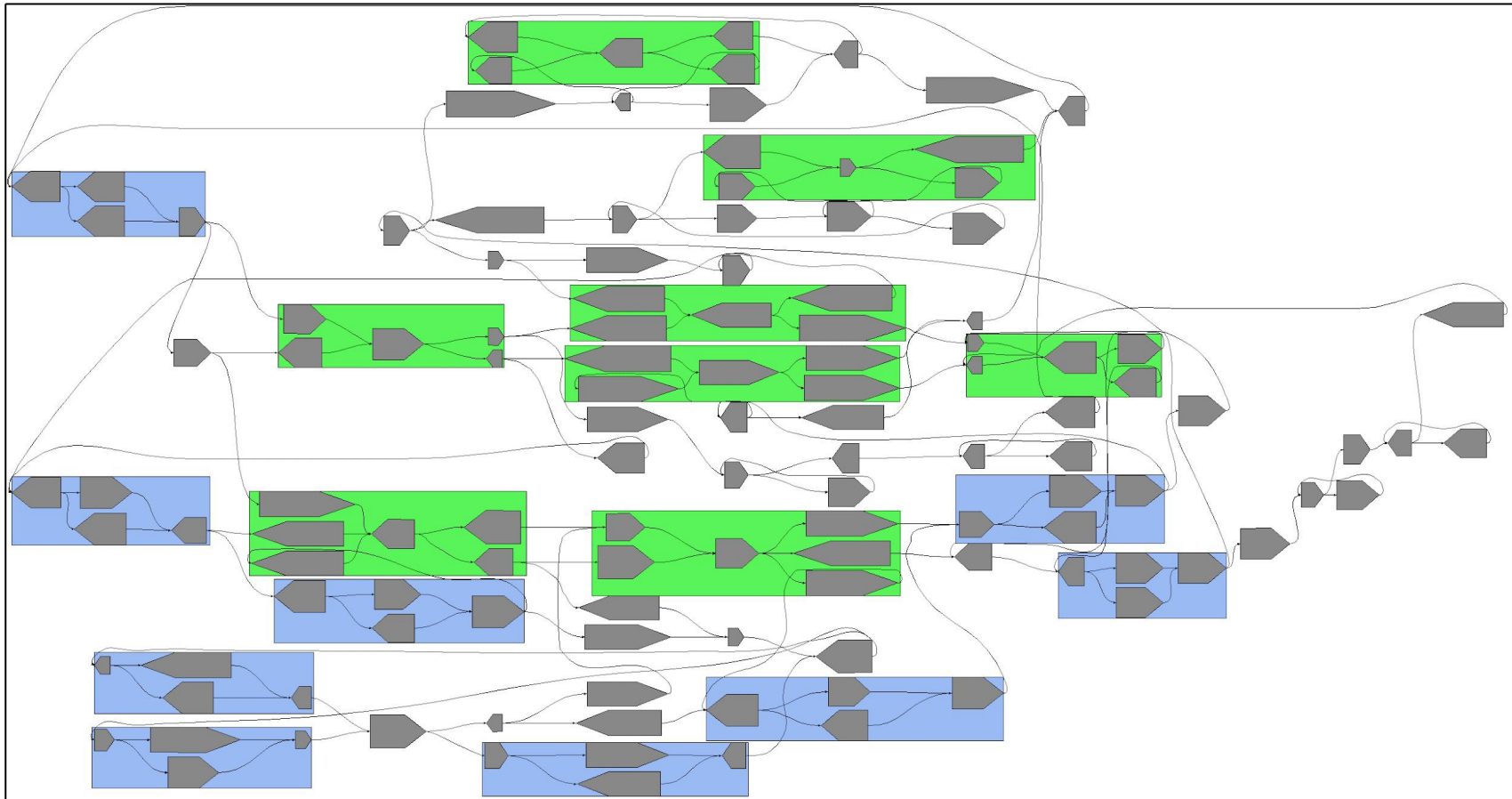
We modify our input to dot in order to linearize some graphs.

Structural Patterns and Layout Linearization

Initial Approach

Node groups (structural patterns) represented as “clusters” when laying out the graph.

Entire connected component is laid out at once.



Sample assembly graph provided with Bandage (Salmonella enterica)

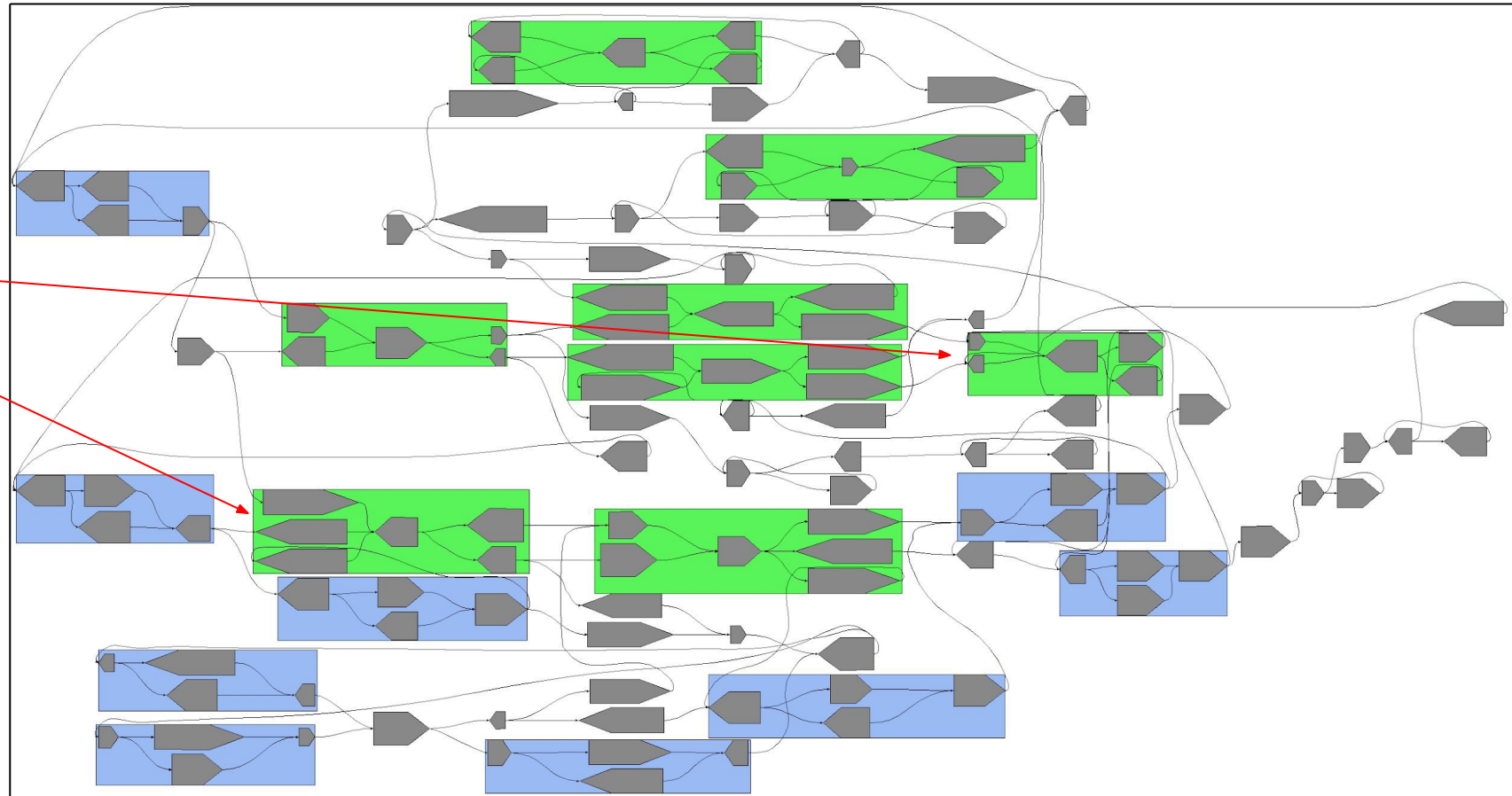
Structural Patterns and Layout Linearization

Initial Approach

Node groups (structural patterns) represented as “clusters” when laying out the graph.

Entire connected component is laid out at once.

Note that dot routes edges through clusters.



Sample assembly graph provided with Bandage (Salmonella enterica)

Structural Patterns and Layout Linearization

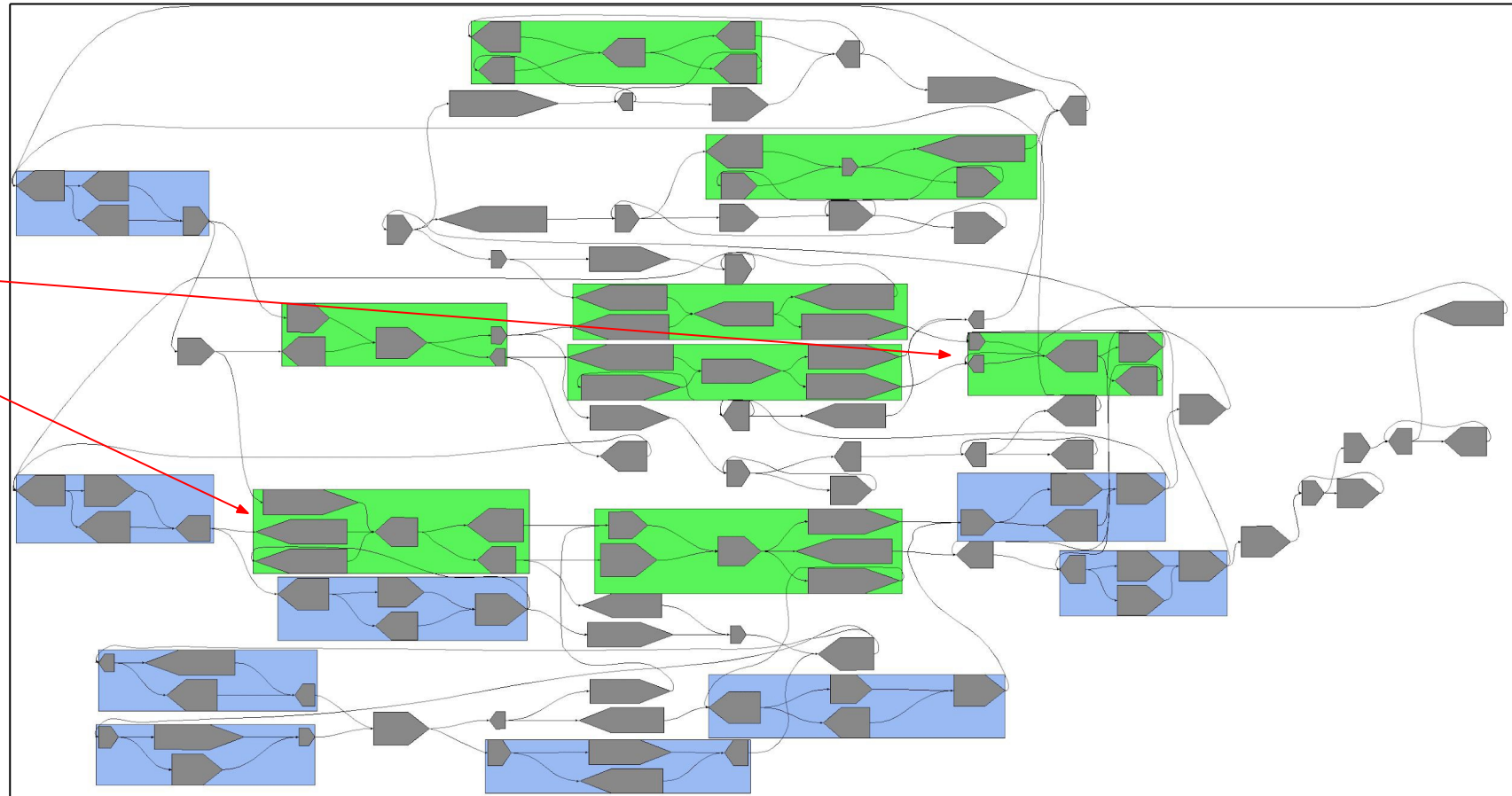
Initial Approach

Node groups (structural patterns) represented as “clusters” when laying out the graph.

Entire connected component is laid out at once.

Note that dot routes edges through clusters.

Can we avoid this?

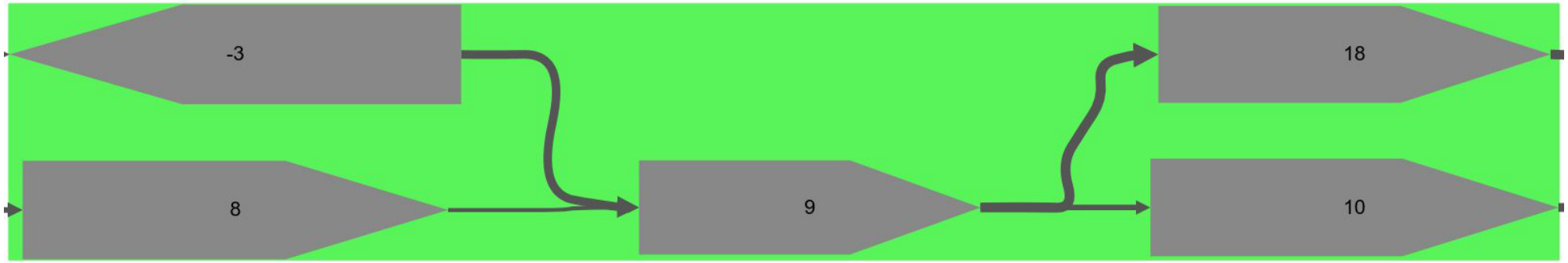


Sample assembly graph provided with Bandage (Salmonella enterica)

Structural Patterns and Layout Linearization

Revised Approach

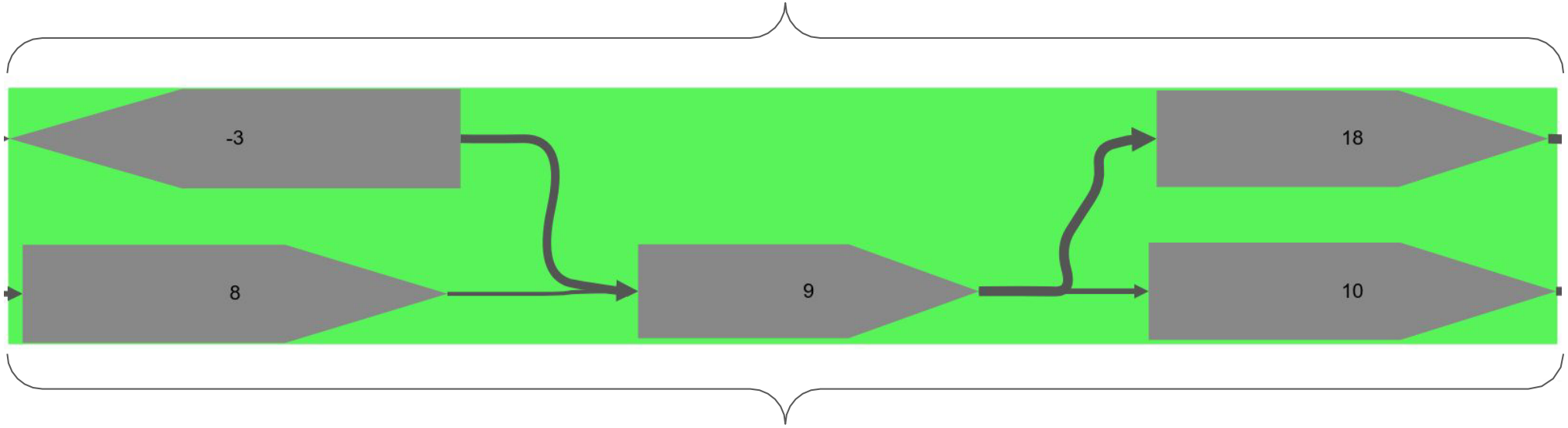
Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.



Structural Patterns and Layout Linearization

Revised Approach

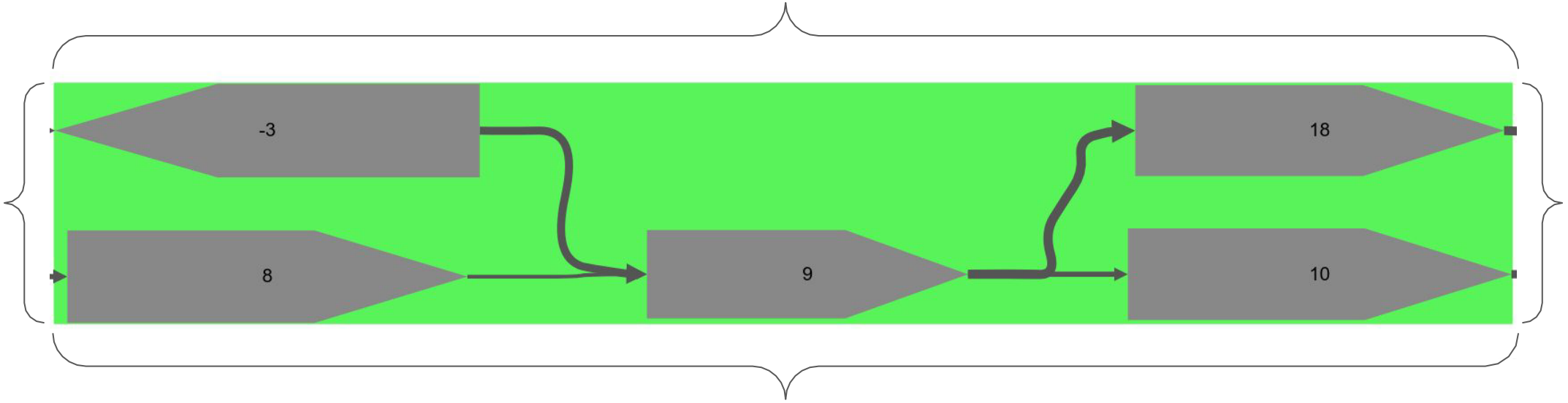
Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.



Structural Patterns and Layout Linearization

Revised Approach

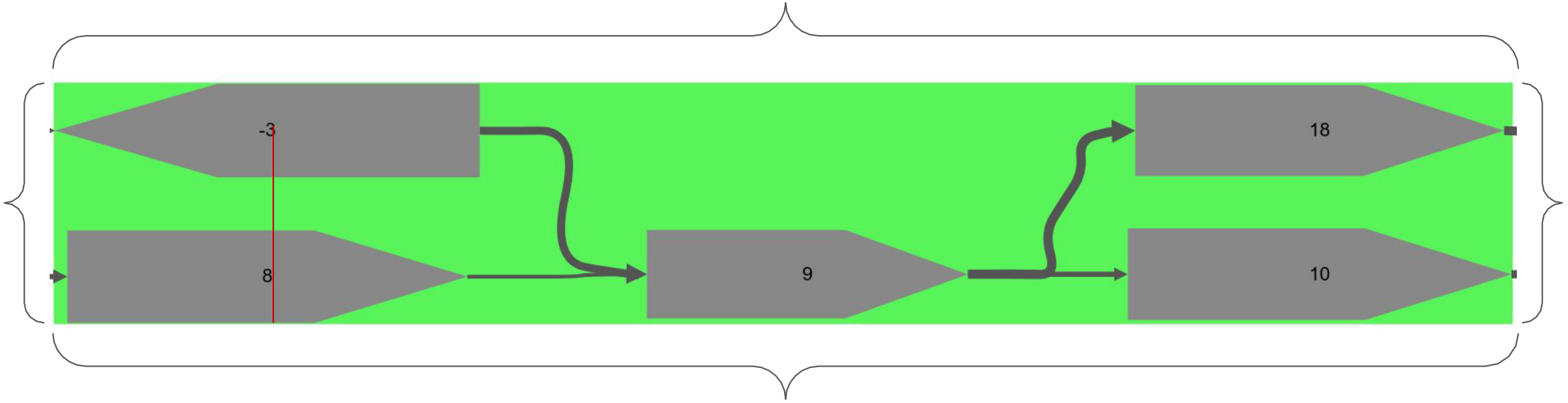
Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.



Structural Patterns and Layout Linearization

Revised Approach

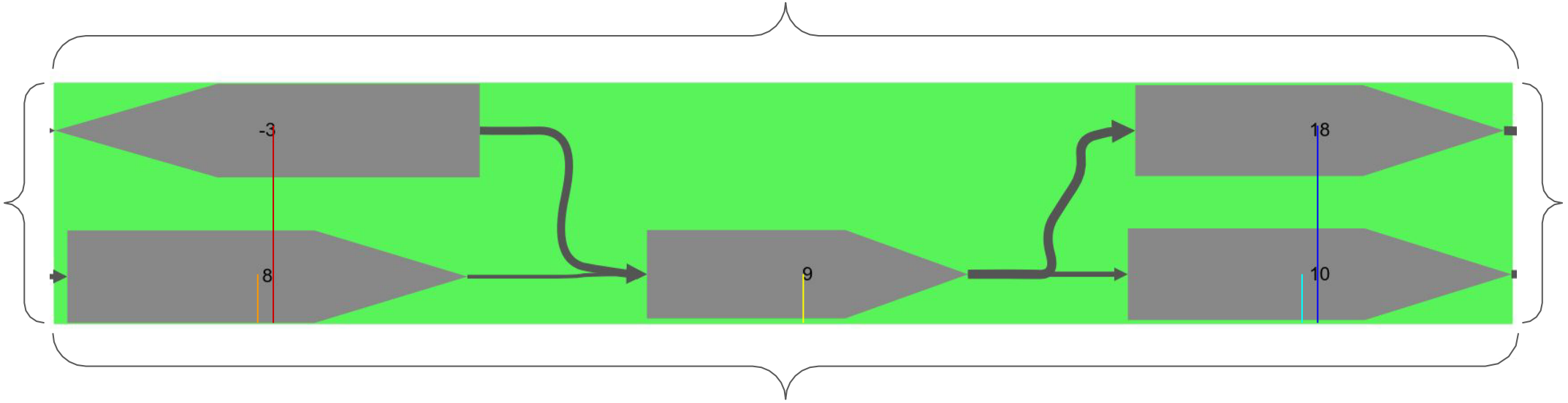
Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.



Structural Patterns and Layout Linearization

Revised Approach

Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.

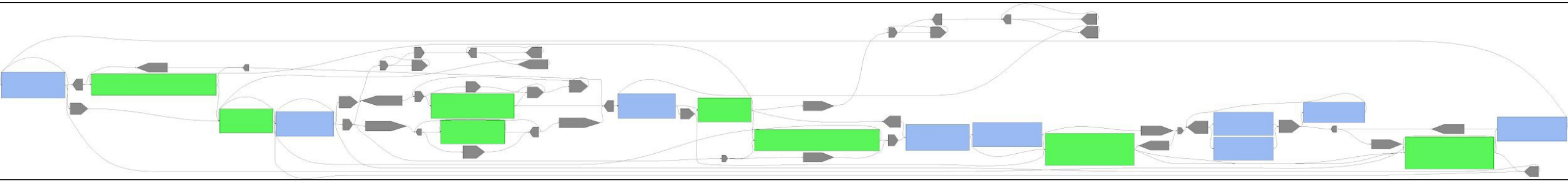


Structural Patterns and Layout Linearization

Revised Approach

Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.

Each node group is represented by a rectangular node of those dimensions during the component layout.



Sample assembly graph provided with Bandage (Salmonella enterica)

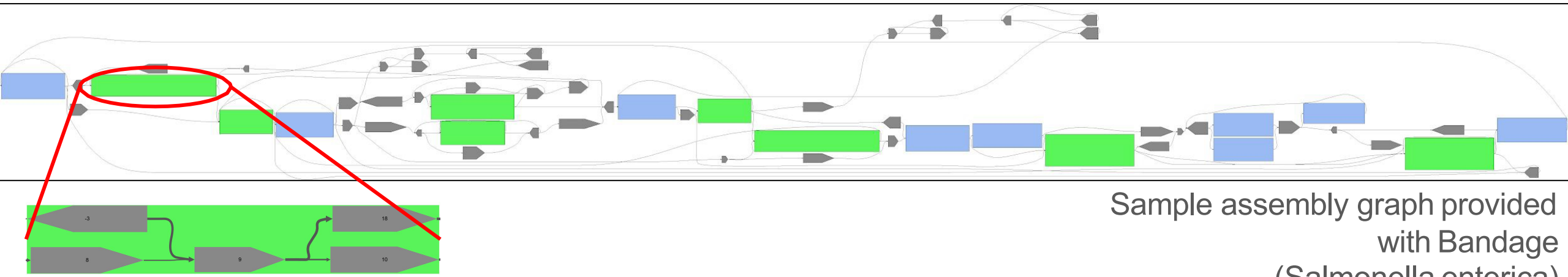
Structural Patterns and Layout Linearization

Revised Approach

Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.

Each node group is represented by a rectangular node of those dimensions during the component layout.

After the component layout process, “child” nodes and edges are backfilled into their parent node groups.



Sample assembly graph provided with Bandage (Salmonella enterica)

Structural Patterns and Layout Linearization

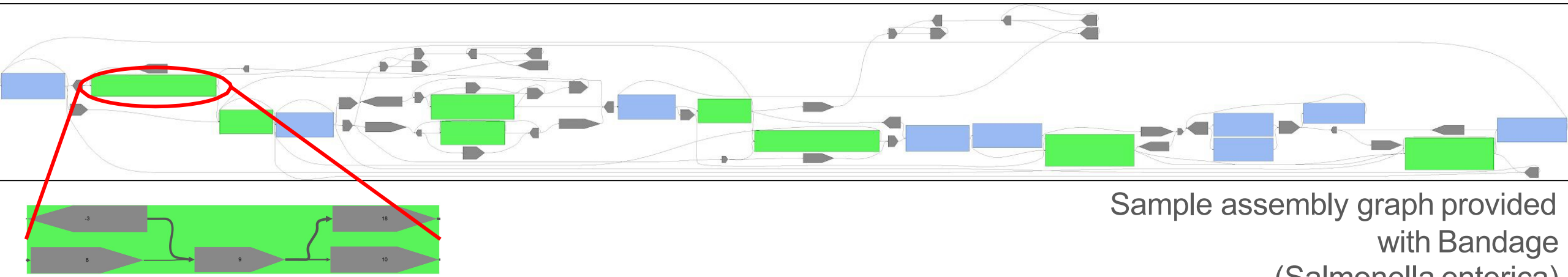
Revised Approach

Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.

Each node group is represented by a rectangular node of those dimensions during the component layout.

After the component layout process, “child” nodes and edges are backfilled into their parent node groups.

This prevents edges from being routed through node groups, linearizing the resulting layout.

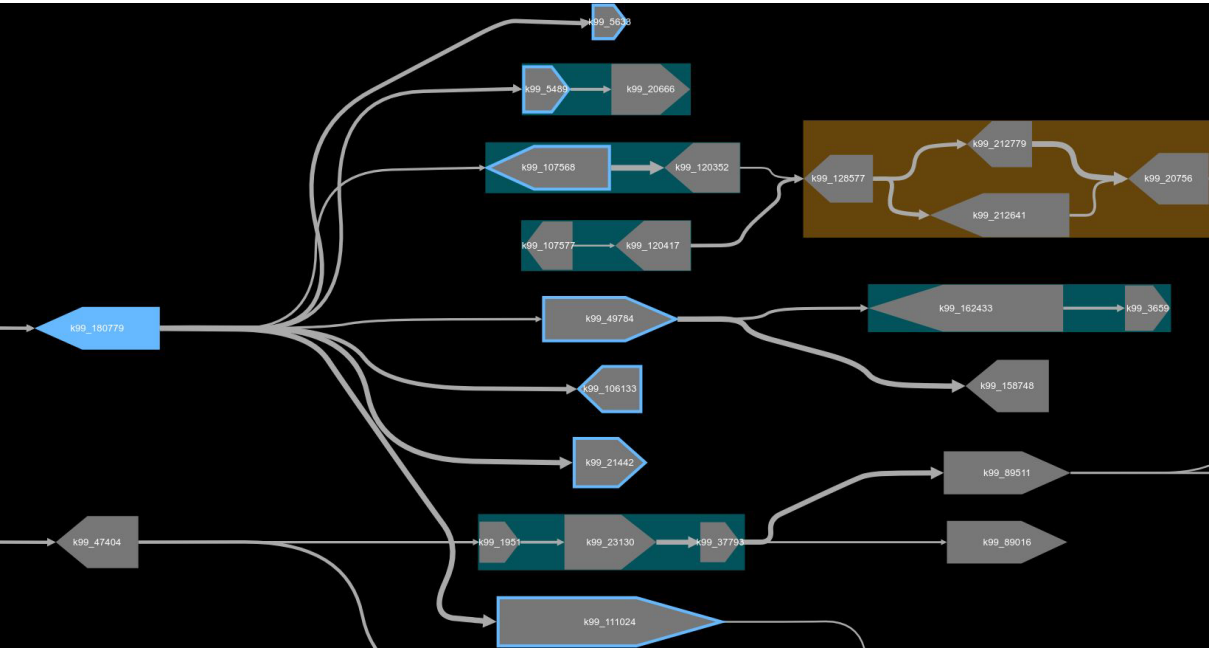


Sample assembly graph provided with Bandage (Salmonella enterica)

Path Construction and Visualization

MetagenomeScope includes tools that facilitate these tasks:

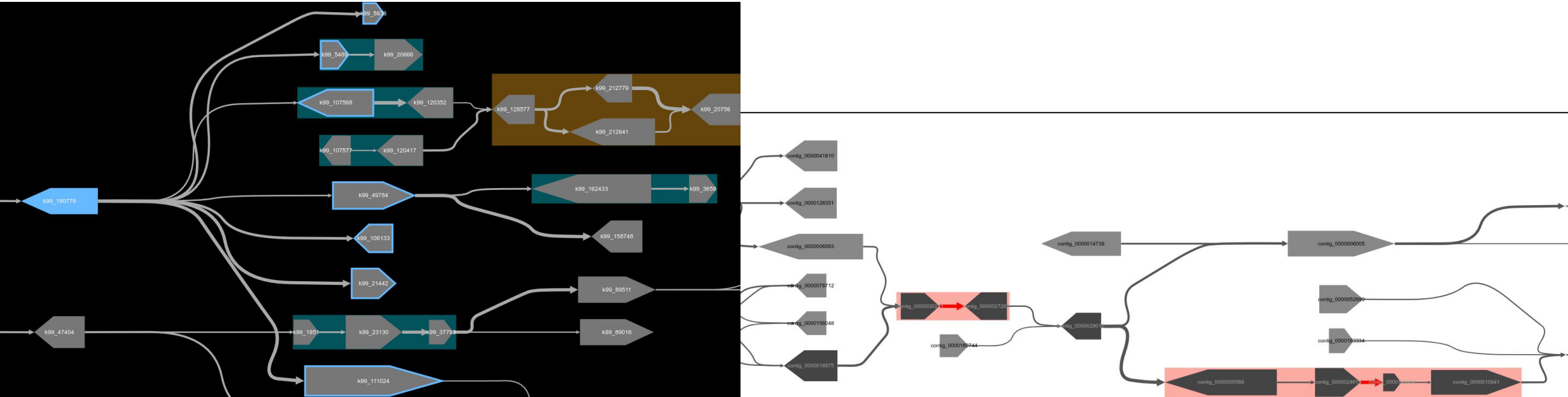
1. Select path(s) through an assembly graph



Path Construction and Visualization

MetagenomeScope includes tools that facilitate these tasks:

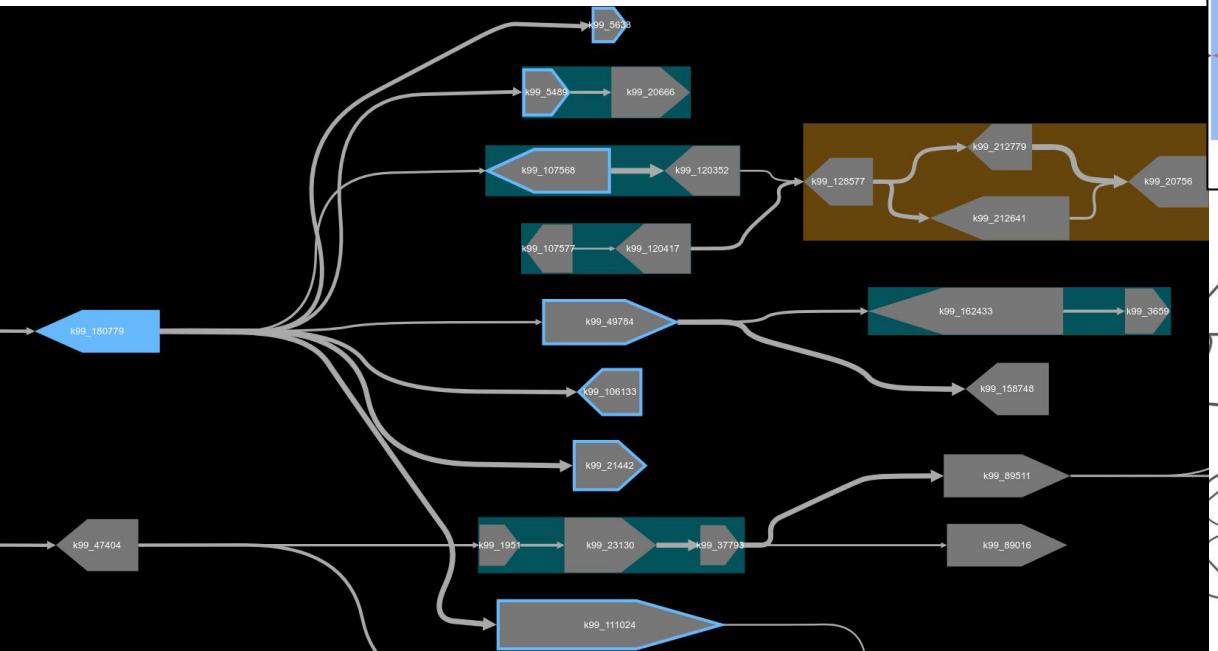
1. Select path(s) through an assembly graph
2. Visualize existing paths on top of an assembly graph

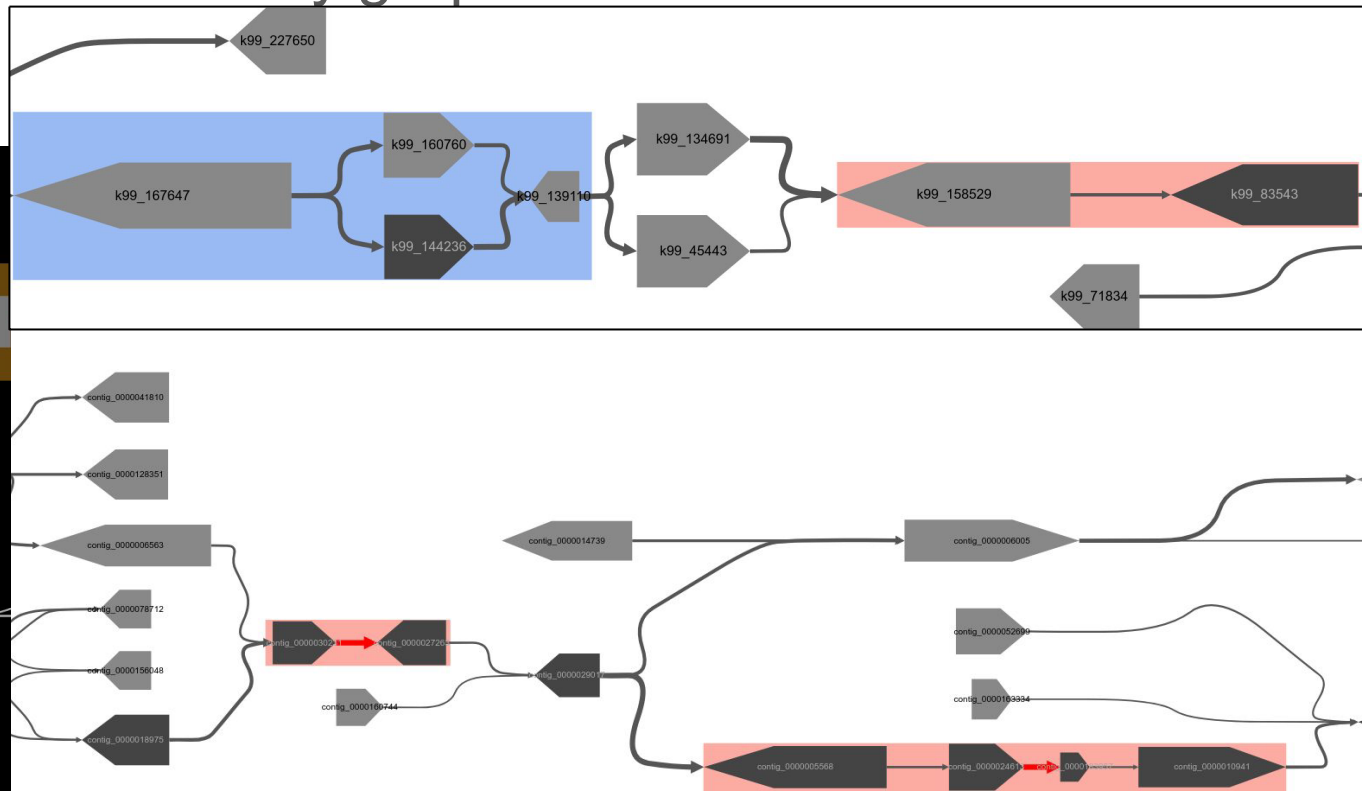


Path Construction and Visualization

MetagenomeScope includes tools that facilitate these tasks:

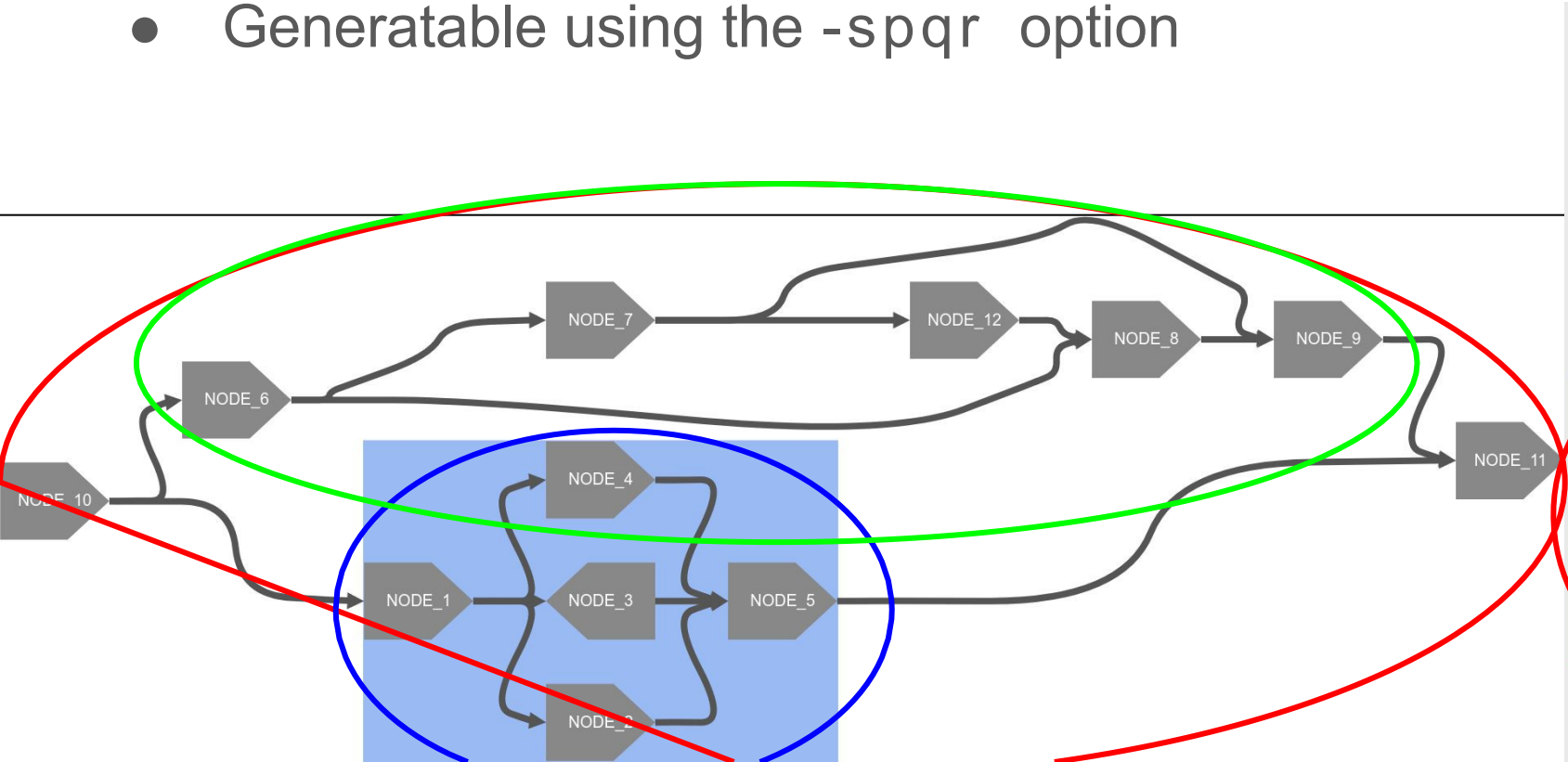
1. Select path(s) through an assembly graph
2. Visualize existing paths on top of an assembly graph
3. Identify errors in path generation

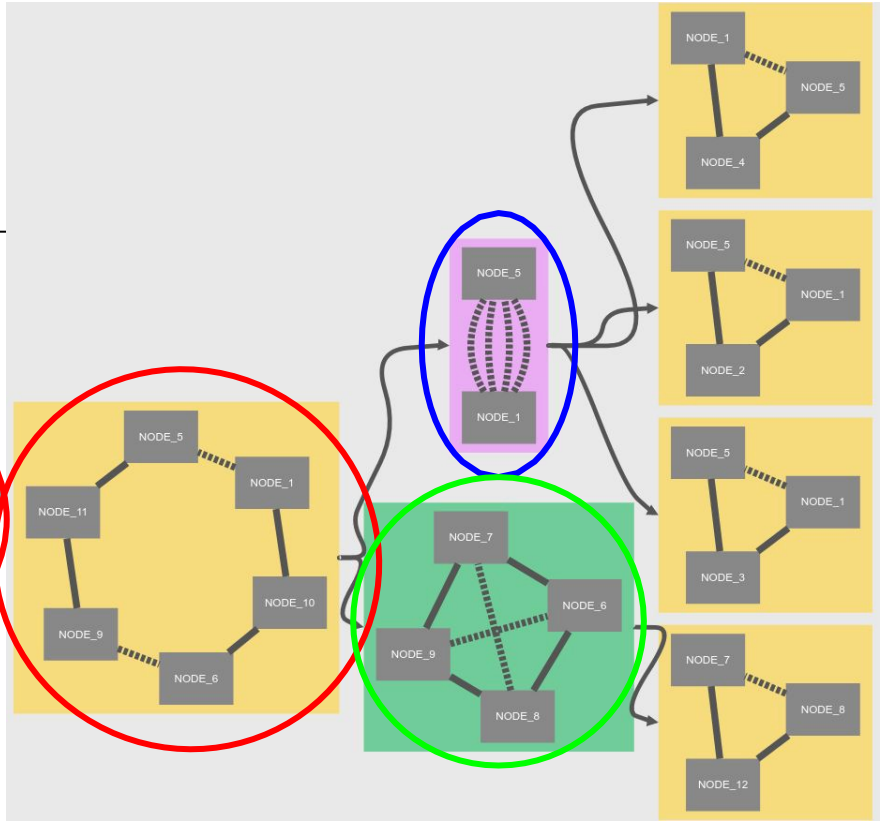


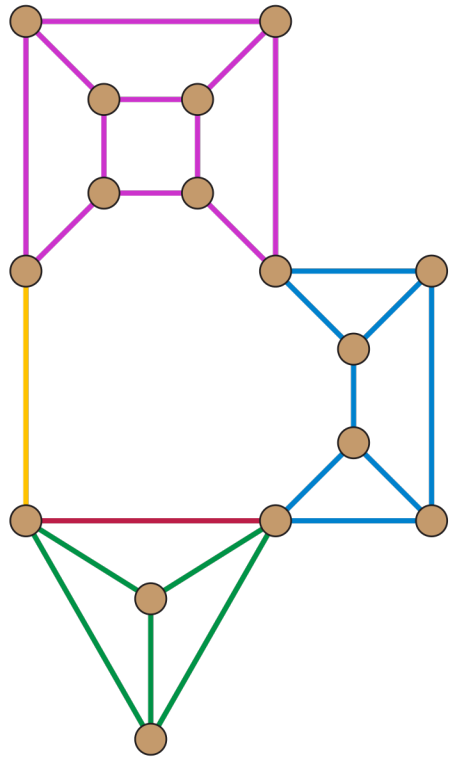


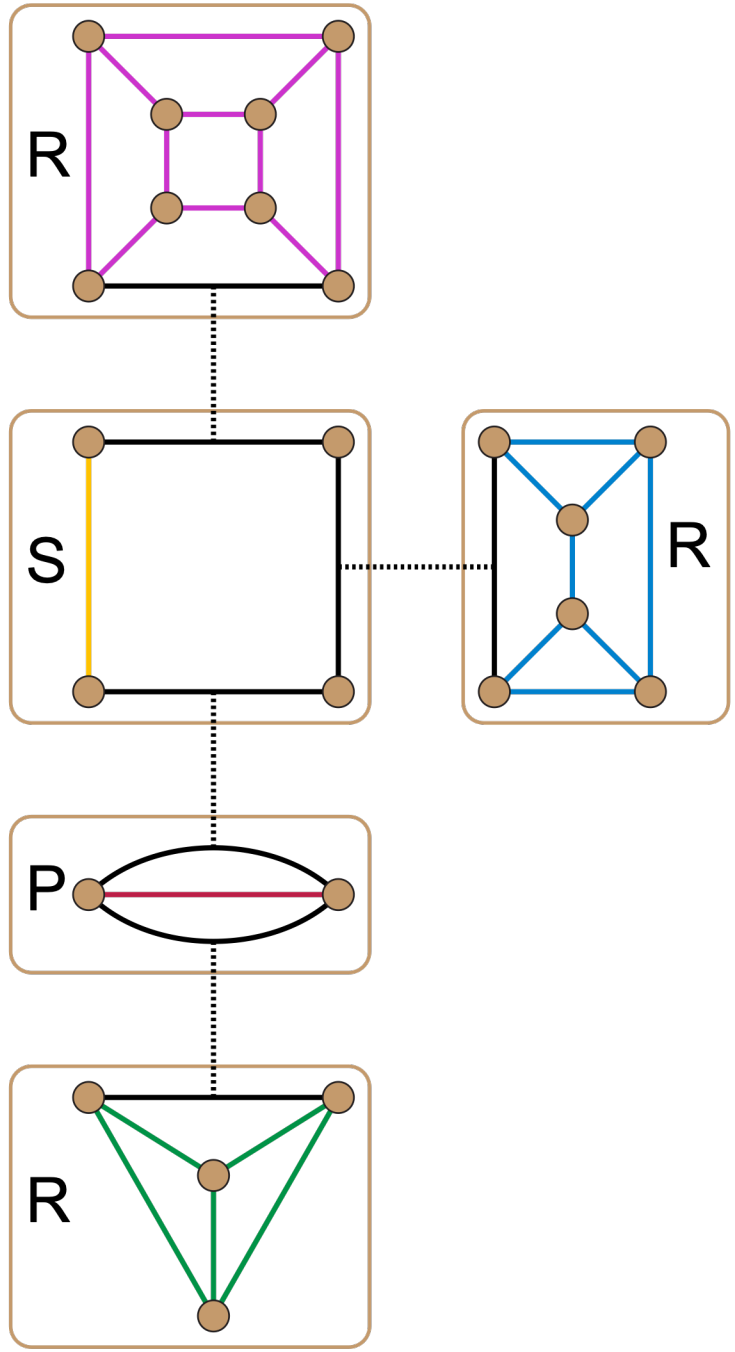
SPQR Tree Decomposition

- Decomposition of biconnected graph into triconnected components
- Data structure originally used for testing graph planarity
- Link between triconnected components and bubbles
- Generatable using the `-spqr` option





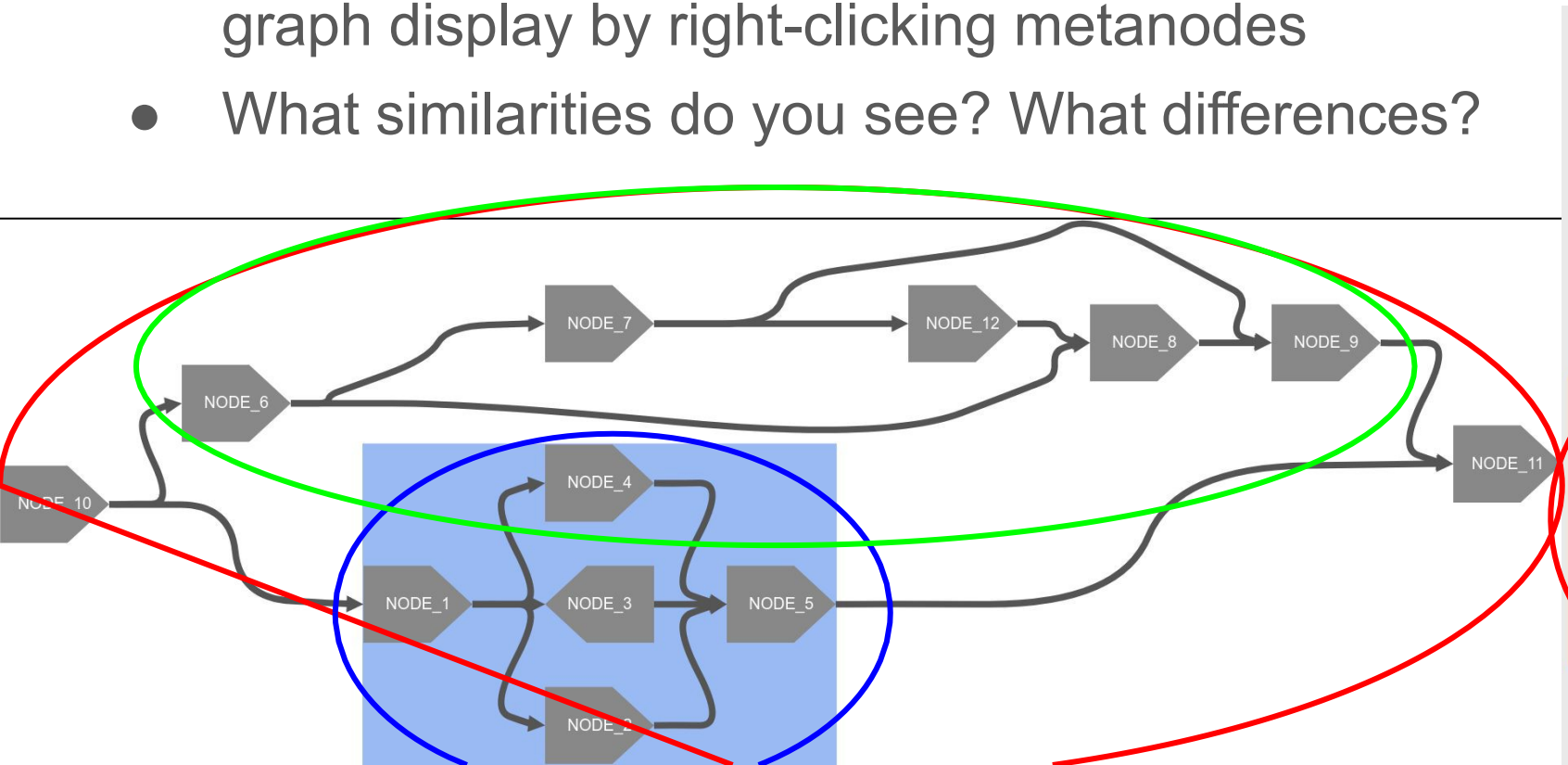


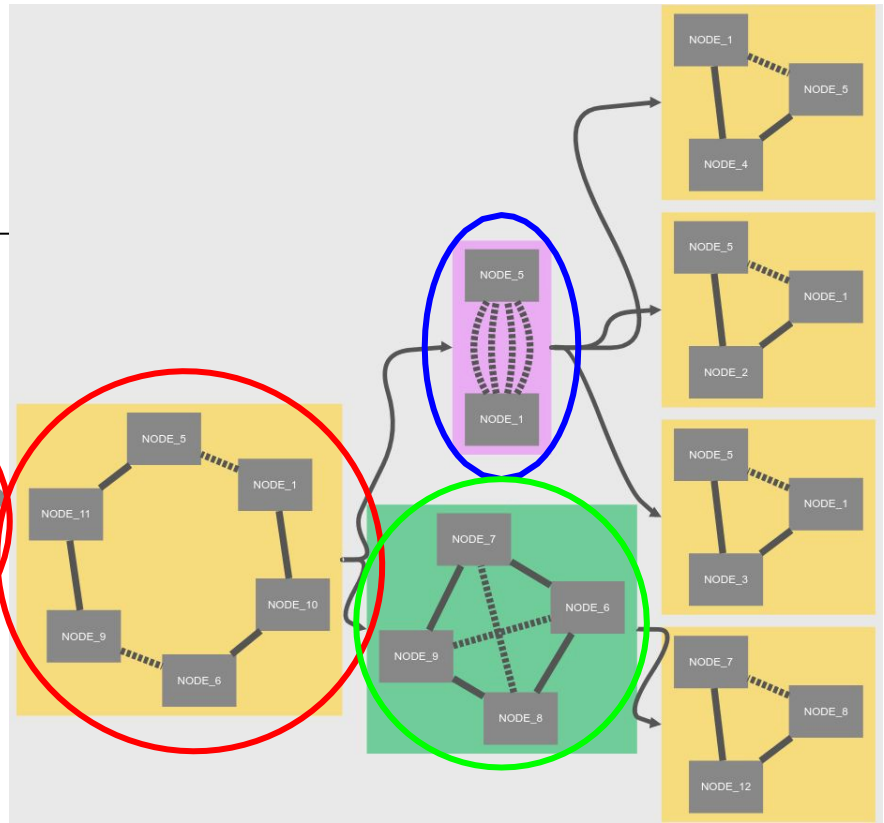


https://en.wikipedia.org/wiki/SPQR_tree

Exercise #3: Viewing an SPQR tree two ways

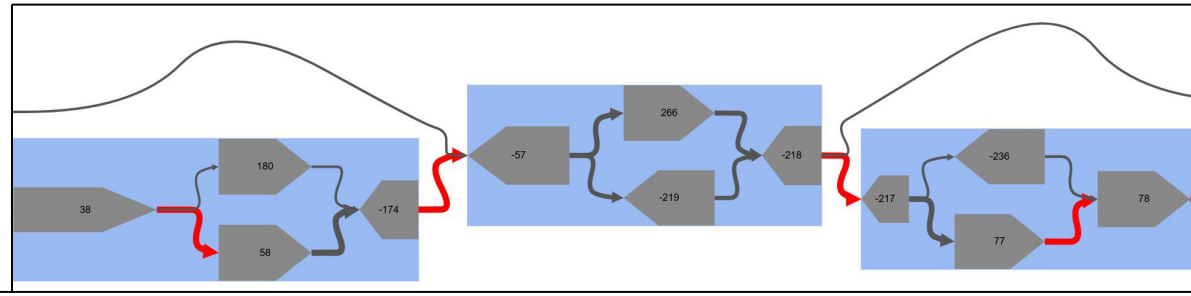
- Open the “MaryGold Paper Figure 2 graph” demo graph in MetagenomeScope’s viewer interface
- Draw the “explicit” and “implicit” SPQR tree graphs, and fully expand the graph display by right-clicking metanodes
- What similarities do you see? What differences?

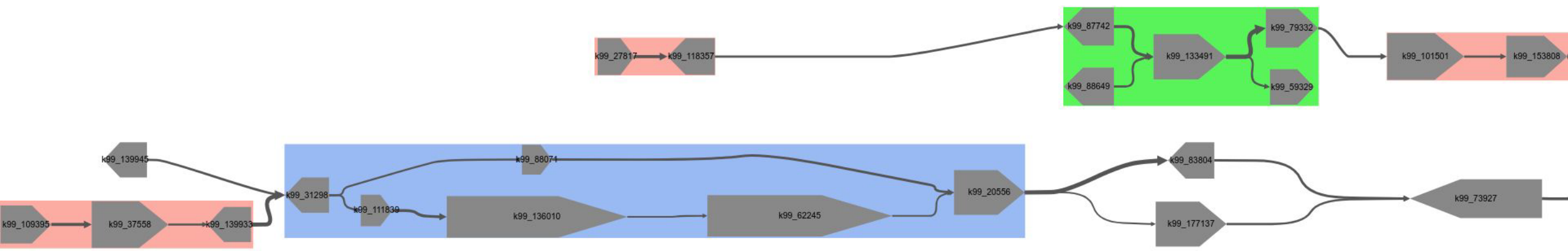




How MetagenomeScope Scales Nodes and Edges

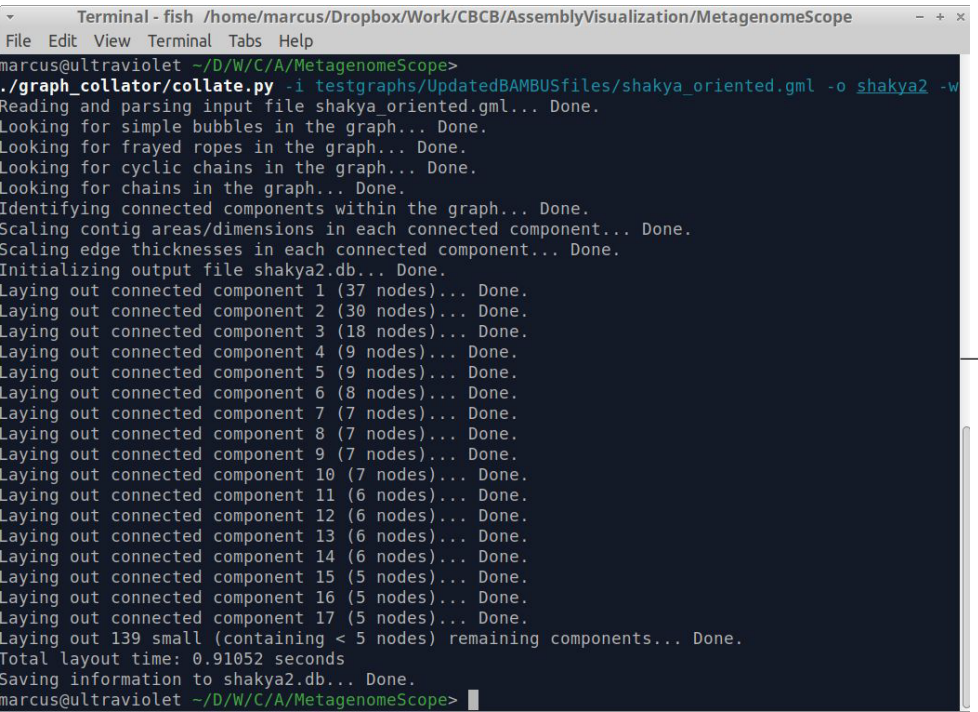
- Scaling nodes' area
  - Logarithmic → relative scaling of node area, based on length
  - Width/height proportions are altered based on percentiles: makes larger nodes look "longer"
- Scaling edges' thickness
  - Using Tukey fences to identify outlier edge weights
  - Flagging outliers in the display

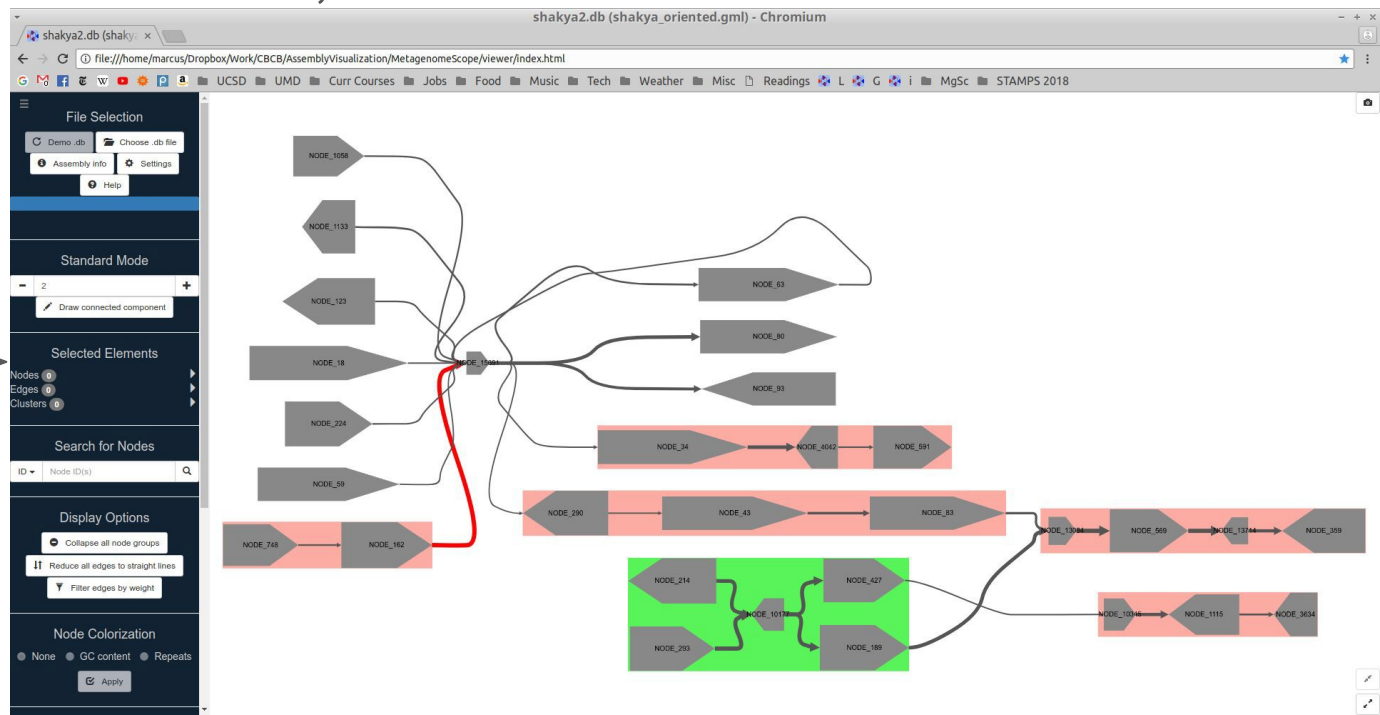




Implementation

- Command-line preprocessing script and web viewer interface
  - Script produces a SQLite3 database file, which is parsed on the client side using `sql.js`
  - Graph visualization is done on the client side (“serverlessly”) using Cytoscape.js
- Various tradeoffs in this approach (database filesize, layout time, …)
- Source code on GitHub under the GNU GPL, version 3





File Formats Accepted

- Velvet (.LastGraph)
- General (.gfa)
- MetaCarvel (.gml)
- SPAdes/MEGAHIT (.fastg) (sort of)

Exercise #4: Find an interesting pattern in the HMP (SRS049959) demo graph

- Human stool sample
- Sample processed by ATLAS in previous tutorial

Summary

- MetagenomeScope: new tool for visualizing/interacting with assembly graphs
- Uses a hierarchical layout algorithm
- Identifies structural patterns in the graph
  - Can be collapsed/uncollapsed
  - Influences the hierarchical layout process
- Supports various novel features that augment exploratory analysis of assembly graphs