Bandage: Next Generation

Anton Korobeynikov

About myself

- Technical leader of SPAdes genome assembler
  - Contributed to some other genome assembly & analysis tools out there as well
- Advocate of OSS, code sharing & reuse
- Contributor to some other open-source projects (not even in bioinformatics)
- Feel free to contact me with questions via email or GitHub issues (preferable)

What is the ideal result of a (meta-)genome assembly?

- In an ideal world:
  - Each of the species assembled separately into a set of contigs (one per each "chromosome")
- What's about strains?
  - Strain variations carefully phased out

Back to reality!

- Contigs / scaffolds?
  - Lots of useful contiguity information is lost
- Strain variants?
  - Strain contigs (shorter, represent individual strains)
  - Consensus contigs (longer, represent backbone structure)

One size does not fill all!

Motivating example

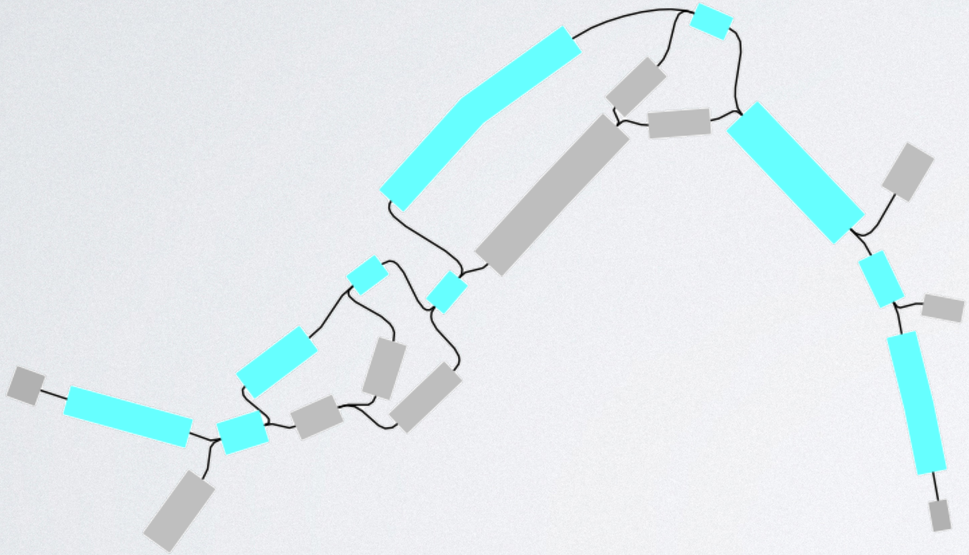
Hospital wastewater discharge metagenome (Ng et al, *Front Microbiol* 2017):

- Read-based estimation of abundances of antibiotic resistance genes
- Nine families of beta-lactamases were reported to be present with different variations
- Study uses read mapping to assembled gene sequences to estimate the abundance and perform further typing

Typical beta-lactamase gene length is 220–280 aa: unlikely to reside on a single read!

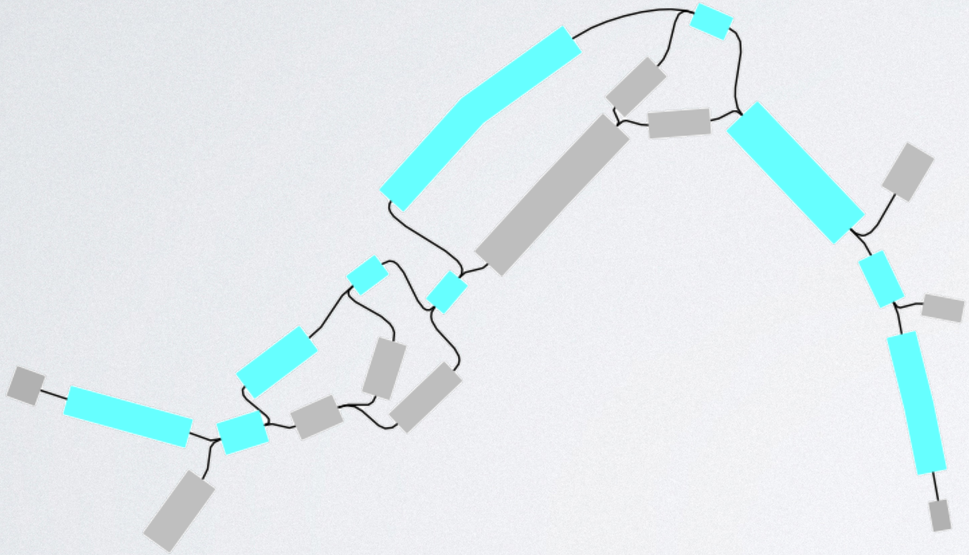
bla_{IMP-4} INSIDE ASSEMBLY GRAPH

Assembly graph:



bla_{IMP-4} INSIDE ASSEMBLY GRAPH

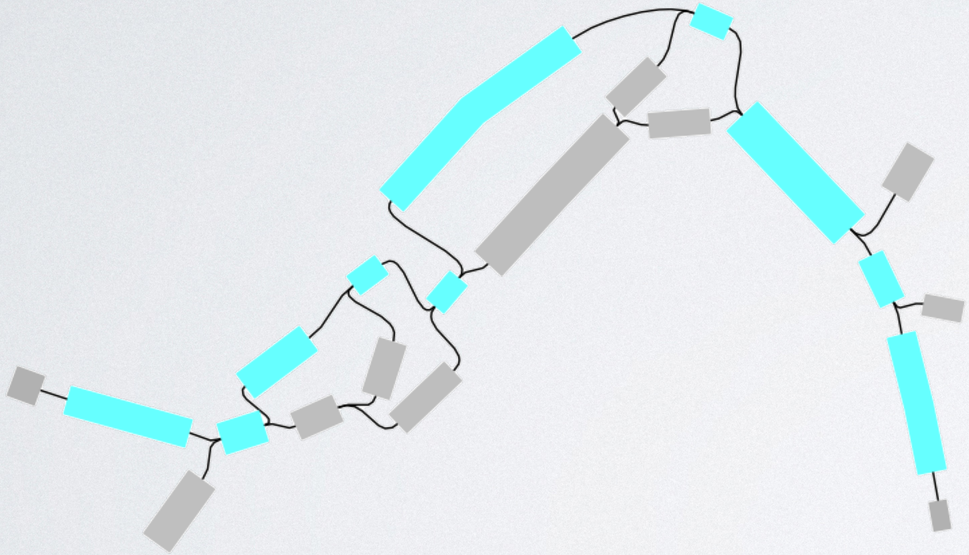
Assembly graph:



Yes, it is here!

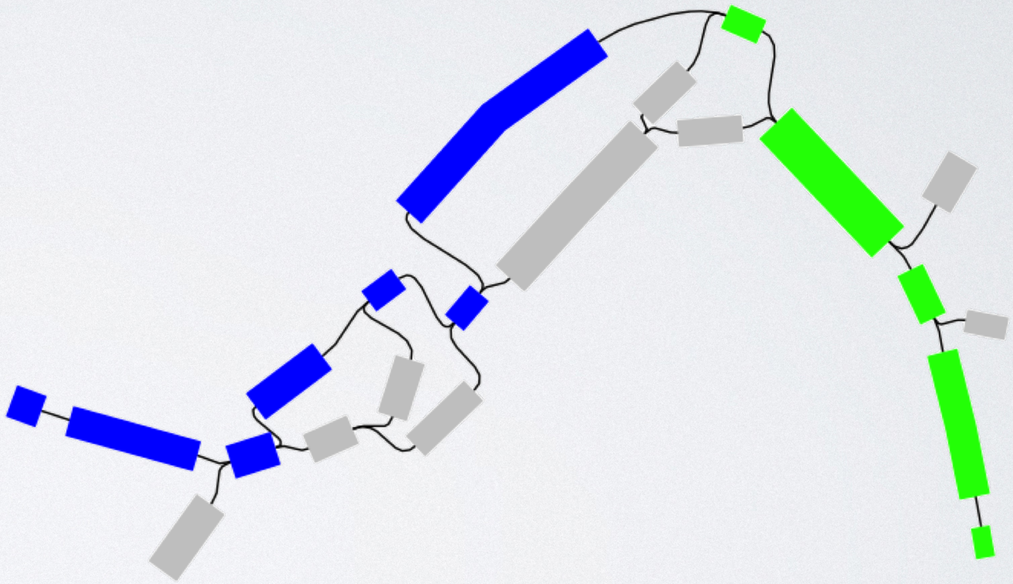
bla_{IMP-4} INSIDE ASSEMBLY GRAPH

Assembly graph:



Yes, it is here!

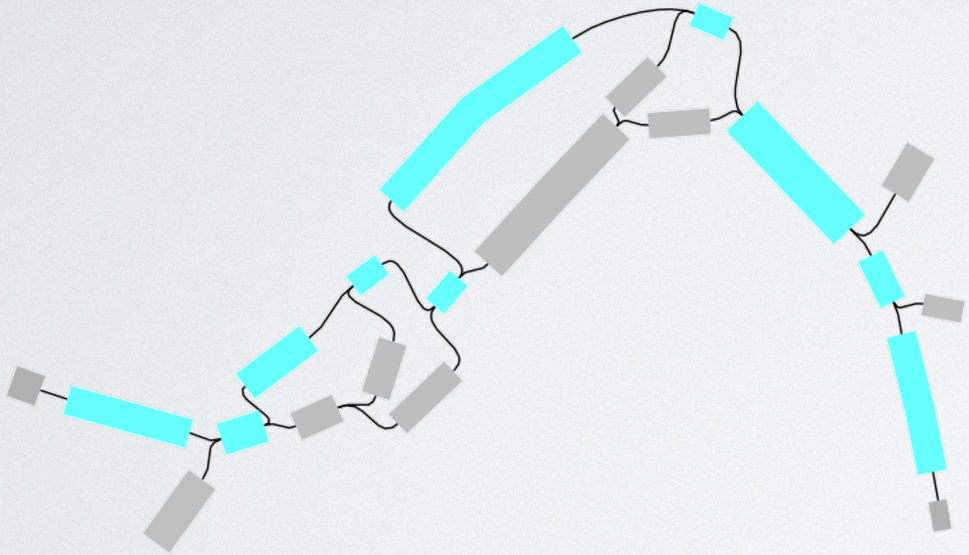
Scaffolds:



Just not assembled into a single scaffold

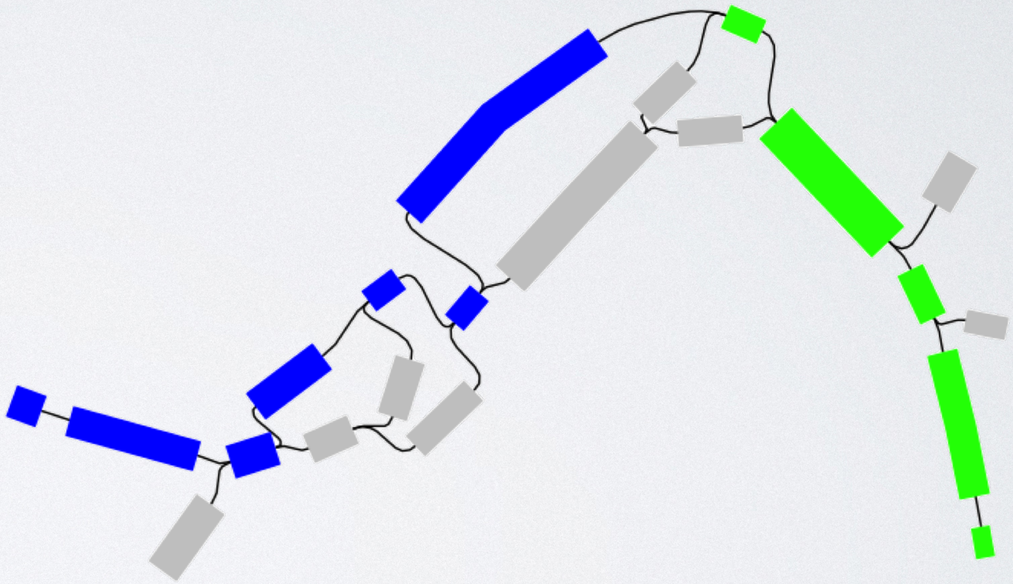
bla_{IMP-4} INSIDE ASSEMBLY GRAPH

Assembly graph:



Yes, it is here!

Scaffolds:



Just not assembled into a single scaffold

Reported abundance in (Ng et al, 2017): 0

ASSEMBLY GRAPHS

- Fully represent the outcome of an assembly process
- Include important contiguity, coverage, etc. information
- Contigs, scaffolds: "annotations" of certain assembly graph "features"
- Could and should be used for downstream analysis!

GFA format

- Text format used to represent assembly graphs
  - Could be optionally compressed (many tools support gzip'ed files)
- Two versions: GFAv1 and GFAv2 (our focus on GFAv1 only)
  - GFAv1.2 is the latest version
- Specification: https://gfa-spec.github.io/GFA-spec/GFA1.html

COMMON

- Lines: records
- Within a single line: tab separated fields
- First field: record type, always a single character
- Other fields: depending on record type
- Each record might have a set of optional typed tags

```
S	v1	CTGAA	SN:Z:chr1	SS:i:0	SR:i:0
S	v2	ACG	SN:Z:chr1	SS:i:5	SR:i:0
S	v3	TGGC	SN:Z:chr1	SS:i:8	SR:i:0
S	v4	TGTGA	SN:Z:chr1	SS:i:12	SR:i:0
S	v5	TTTC	SN:Z:foo	SS:i:8	SR:i:1
S	v6	CTGA	SN:Z:foo	SS:i:12	SR:i:1
S	v7	GTTAC	SN:Z:bar	SS:i:5	SR:i:2
L	v1	+	v2	+	0M	FO:Z:Foo
L	v2	+	v3	+	0M
L	v3	+	v4	+	0M
L	v2	+	v5	+	0M
L	v5	+	v6	+	0M
L	v6	+	v4	+	0M
L	v1	+	v7	+	0M
L	v7	+	v6	+	0M
P	foo	v1+,v2+,v5+,v6+,v4+	*
```

COMMON

- Lines: records
- Within a single line: tab separated fields
- First field: record type, always a single character
- Other fields: depending on record type
- Each record might have a set of optional typed tags

```
S	v1	CTGAA	SN:Z:chr1	SS:i:0	SR:i:0
S	v2	ACG	SN:Z:chr1	SS:i:5	SR:i:0
S	v3	TGGC	SN:Z:chr1	SS:i:8	SR:i:0
S	v4	TGTGA	SN:Z:chr1	SS:i:12	SR:i:0
S	v5	TTTC	SN:Z:foo	SS:i:8	SR:i:1
S	v6	CTGA	SN:Z:foo	SS:i:12	SR:i:1
S	v7	GTTAC	SN:Z:bar	SS:i:5	SR:i:2
L	v1	+	v2	+	0M	FO:Z:Foo
L	v2	+	v3	+	0M
L	v3	+	v4	+	0M
L	v2	+	v5	+	0M
L	v5	+	v6	+	0M
L	v6	+	v4	+	0M
L	v1	+	v7	+	0M
L	v7	+	v6	+	0M
P	foo	v1+,v2+,v5+,v6+,v4+	*
```

COMMON

- Lines: records
- Within a single line: tab separated fields
- First field: record type, always a single character
- Other fields: depending on record type
- Each record might have a set of optional typed tags

```
S       v1      CTGAA   SN:Z:chr1       SS:i:0  SR:i:0
S       v2      ACG     SN:Z:chr1       SS:i:5  SR:i:0
S       v3      TGGC    SN:Z:chr1       SS:i:8  SR:i:0
S       v4      TGTGA   SN:Z:chr1       SS:i:12 SR:i:0
S       v5      TTTC    SN:Z:foo        SS:i:8  SR:i:1
S       v6      CTGA    SN:Z:foo        SS:i:12 SR:i:1
S       v7      GTTAC   SN:Z:bar        SS:i:5  SR:i:2
L       v1      +       v2      +       0M      FO:Z:Foo
L       v2      +       v3      +       0M
L       v3      +       v4      +       0M
L       v2      +       v5      +       0M
L       v5      +       v6      +       0M
L       v6      +       v4      +       0M
L       v1      +       v7      +       0M
L       v7      +       v6      +       0M
P       foo     v1+,v2+,v5+,v6+,v4+     *
```

COMMON

- Lines: records
- Within a single line: tab separated fields
- First field: record type, always a single character
- Other fields: depending on record type
- Each record might have a set of optional typed tags

```
S    v1    CTGAA    SN:Z:chr1    SS:i:0   SR:i:0
S    v2    ACG      SN:Z:chr1    SS:i:5   SR:i:0
S    v3    TGGC     SN:Z:chr1    SS:i:8   SR:i:0
S    v4    TGTGA    SN:Z:chr1    SS:i:12  SR:i:0
S    v5    TTTC     SN:Z:foo     SS:i:8   SR:i:1
S    v6    CTGA     SN:Z:foo     SS:i:12  SR:i:1
S    v7    GTTAC    SN:Z:bar     SS:i:5   SR:i:2
L    v1    +        v2      +    0M       FO:Z:Foo
L    v2    +        v3      +    0M
L    v3    +        v4      +    0M
L    v2    +        v5      +    0M
L    v5    +        v6      +    0M
L    v6    +        v4      +    0M
L    v1    +        v7      +    0M
L    v7    +        v6      +    0M
P    foo   v1+,v2+,v5+,v6+,v4+    *
```

COMMON

- Lines: records
- Within a single line: tab separated fields
- First field: record type, always a single character
- Other fields: depending on record type
- Each record might have a set of optional typed tags

```
S    v1    CTGAA    SN:Z:chr1    SS:i:0   SR:i:0
S    v2    ACG      SN:Z:chr1    SS:i:5   SR:i:0
S    v3    TGGC     SN:Z:chr1    SS:i:8   SR:i:0
S    v4    TGTGA    SN:Z:chr1    SS:i:12  SR:i:0
S    v5    TTTC     SN:Z:foo     SS:i:8   SR:i:1
S    v6    CTGA     SN:Z:foo     SS:i:12  SR:i:1
S    v7    GTTAC    SN:Z:bar     SS:i:5   SR:i:2
L    v1    +        v2      +    0M       FO:Z:Foo
L    v2    +        v3      +    0M
L    v3    +        v4      +    0M
L    v2    +        v5      +    0M
L    v5    +        v6      +    0M
L    v6    +        v4      +    0M
L    v1    +        v7      +    0M
L    v7    +        v6      +    0M
P    foo   v1+,v2+,v5+,v6+,v4+    *
```

COMMON

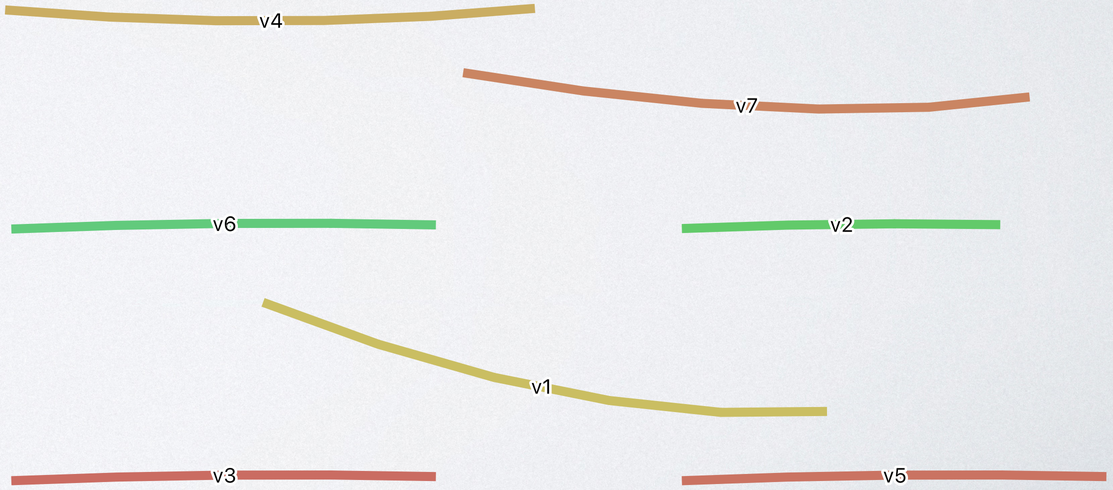
- Lines: records
- Within a single line: tab separated fields
- First field: record type, always a single character
- Other fields: depending on record type
- Each record might have a set of optional typed tags

```
S    v1    CTGAA    SN:Z:chr1    SS:i:0   SR:i:0
S    v2    ACG      SN:Z:chr1    SS:i:5   SR:i:0
S    v3    TGGC     SN:Z:chr1    SS:i:8   SR:i:0
S    v4    TGTGA    SN:Z:chr1    SS:i:12  SR:i:0
S    v5    TTTC     SN:Z:foo     SS:i:8   SR:i:1
S    v6    CTGA     SN:Z:foo     SS:i:12  SR:i:1
S    v7    GTTAC    SN:Z:bar     SS:i:5   SR:i:2
L    v1    +        v2      +    0M       FO:Z:Foo
L    v2    +        v3      +    0M
L    v3    +        v4      +    0M
L    v2    +        v5      +    0M
L    v5    +        v6      +    0M
L    v6    +        v4      +    0M
L    v1    +        v7      +    0M
L    v7    +        v6      +    0M
P    foo   v1+,v2+,v5+,v6+,v4+    *
```

Segments

- '**S**' record type
- Represents a continuous sequence or subsequence
- Fields:
  - Name
  - Sequence (might be absent, use '*' to indicate this)
  - Set of optional tags
- If sequence is absent, then **LN** tag must be present to indicate length

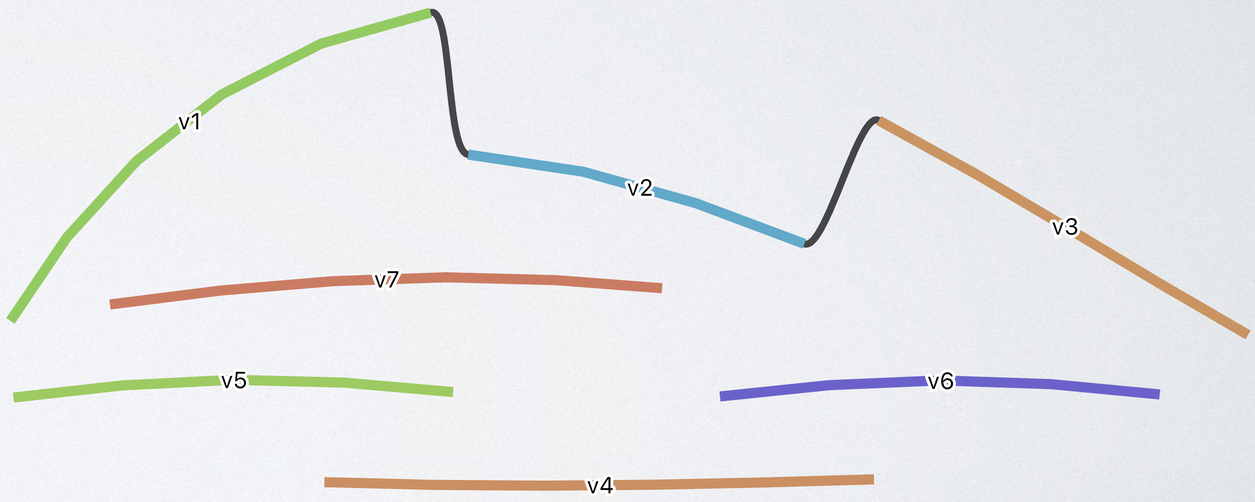
```
S       v1      CTGAA   SN:Z:chr1       SS:i:0  SR:i:0
S       v2      ACG     SN:Z:chr1       SS:i:5  SR:i:0
S       v3      TGGC    SN:Z:chr1       SS:i:8  SR:i:0
S       v4      TGTGA   SN:Z:chr1       SS:i:12 SR:i:0
S       v5      TTTC    SN:Z:foo        SS:i:8  SR:i:1
S       v6      CTGA    SN:Z:foo        SS:i:12 SR:i:1
S       v7      *       LN:i:42
```



Links

- 'L' record type
- An overlap between two segments
- Implies direct adjacency of segments in the genome
- Link is from the end of one segment to the beginning of another segment
- Stores the orientation of each segment and the CIGAR of overlap

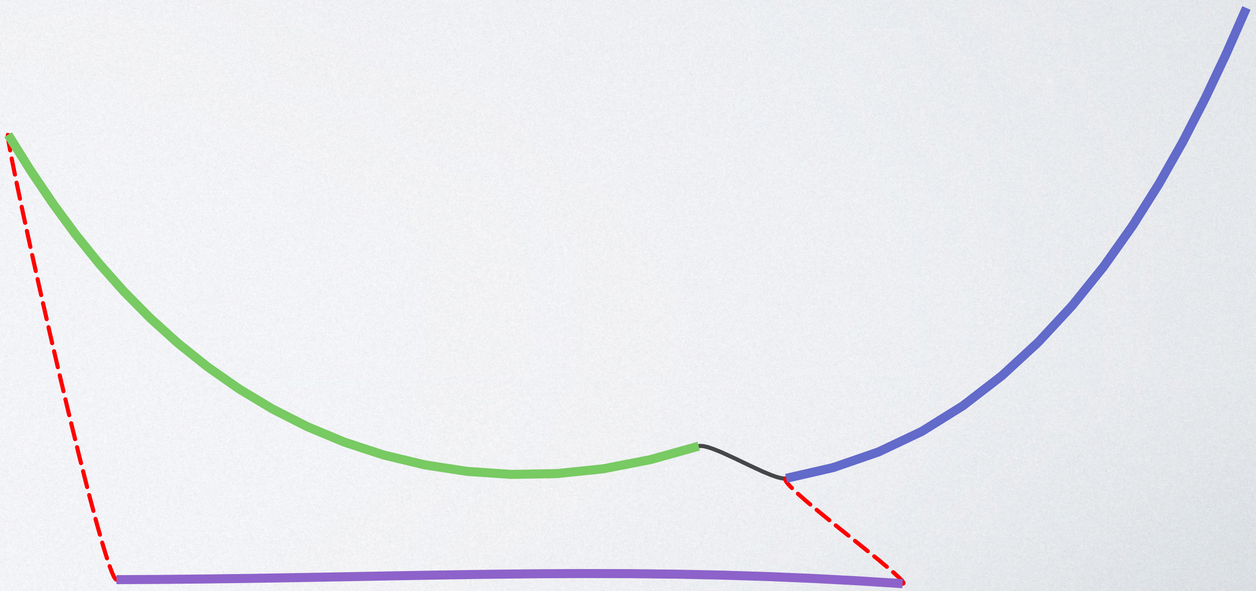
```
L	v1	+	v2	+	0M	FO:Z:Foo
L	v2	+	v3	-	55M
```



Jumps (since GFA v1.2)

- '`J`' record type
- Connection between two segments
- Similar to links, but does not imply a direct genome adjacency
- May provide estimated distance between segments (or represent a *shortcut*)
- Main usecase: represent segment relations across assembly gaps (e.g. in scaffolds)

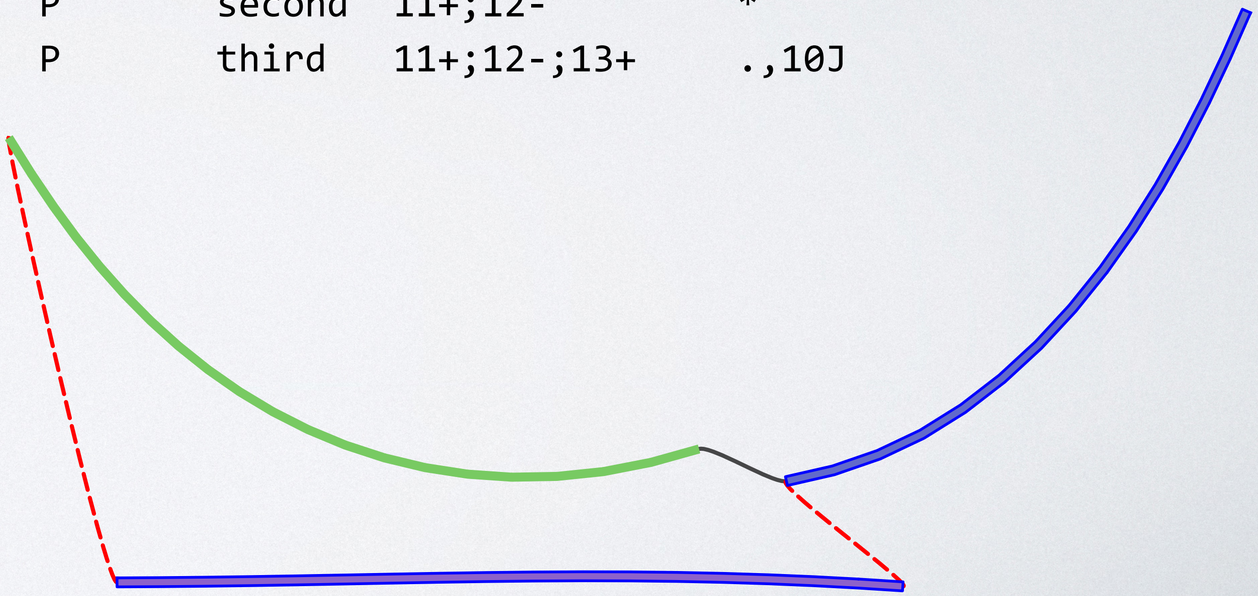
```
S	11	ACCTT
S	12	TCAAGG
S	13	CTTGATT
L	11	+	13	-	4M
J	11	+	12	-	*	SC:i:1
J	12	-	13	+	10
```



Paths

- 'P' record type
- Ordered list of oriented segments
- Each consecutive pair of oriented segments is supported by a link or a jump record
- May show segment overlaps or estimated gap size (in case of jump connection)
- Can represent scaffolds or trace of some sequence across the graph

```
S       11      ACCTT
S       12      TCAAGG
S       13      CTTGATT
L       11      +       13      -       4M
J       11      +       12      -       *       SC:i:1
J       12      -       13      +       10
P       first   11+,13-         *
P       second  11+;12-         *
P       third   11+;12-;13+     .,10J
```



Walks (since GFA v1.1)

- 'W' record type
- Ordered list of oriented segments
- Intended for pangenome use cases
- Only for graphs without segment overlaps
- Fields:
  - Sample identifier
  - Haplotype index
  - Sequence identifier (in a sample)
  - Start / end positions (on a sequence)
  - Ordered walk

```
S       s11     ACCTT
S       s12     TC
S       s13     GATT
L       s11     +       s12     -       0M
L       s12     -       s13     +       0M
L       s11     +       s13     +       0M
W       NA12878 1       chr1    0   11  >s11<s12>s13
```

GFA TAGS

- Optional fields of records
- Use `TAG:TYPE:VALUE` format
- `TAG`:
  - Two characters, normally uppercase
  - Lower case names are reserved for end users
- `TYPE`:
  - `A` – single printable character
  - `i` – signed integer
  - `f` – floating point value
  - `Z` – printable string (including space)
  - `J` – JSON
  - `H` – byte array in hex format
  - `B` – array of integers or floats

```
S    v1    CTGAA    SN:Z:chr1   SS:i:0   SR:i:0
S    v2    ACG      SN:Z:chr1   SS:i:5   SR:i:0
S    v3    TGGC     SN:Z:chr1   SS:i:8   SR:i:0
S    v4    TGTGA    SN:Z:chr1   SS:i:12  SR:i:0
S    v5    TTTC     SN:Z:foo    SS:i:8   SR:i:1
S    v6    CTGA     SN:Z:foo    SS:i:12  SR:i:1
S    v7    GTTAC    SN:Z:bar    SS:i:5   SR:i:2
L    v1    +        v2     +    0M       FO:Z:Foo
L    v2    +        v3     +    0M       CB:Z:red
L    v3    +        v4     +    0M       WD:i:3
L    v2    +        v5     +    0M       WD:f:7.5
L    v5    +        v6     +    0M       aa:A:f
L    v6    +        v4     +    0M       bb:i:7
L    v1    +        v7     +    0M       CB:Z:#ffeeaa
L    v7    +        v6     +    0M       PS:i:4
P    foo   v1+,v2+,v5+,v6+,v4+ *         WT:f:1e3
```

BANDAGE

Bioinformatics

Application for

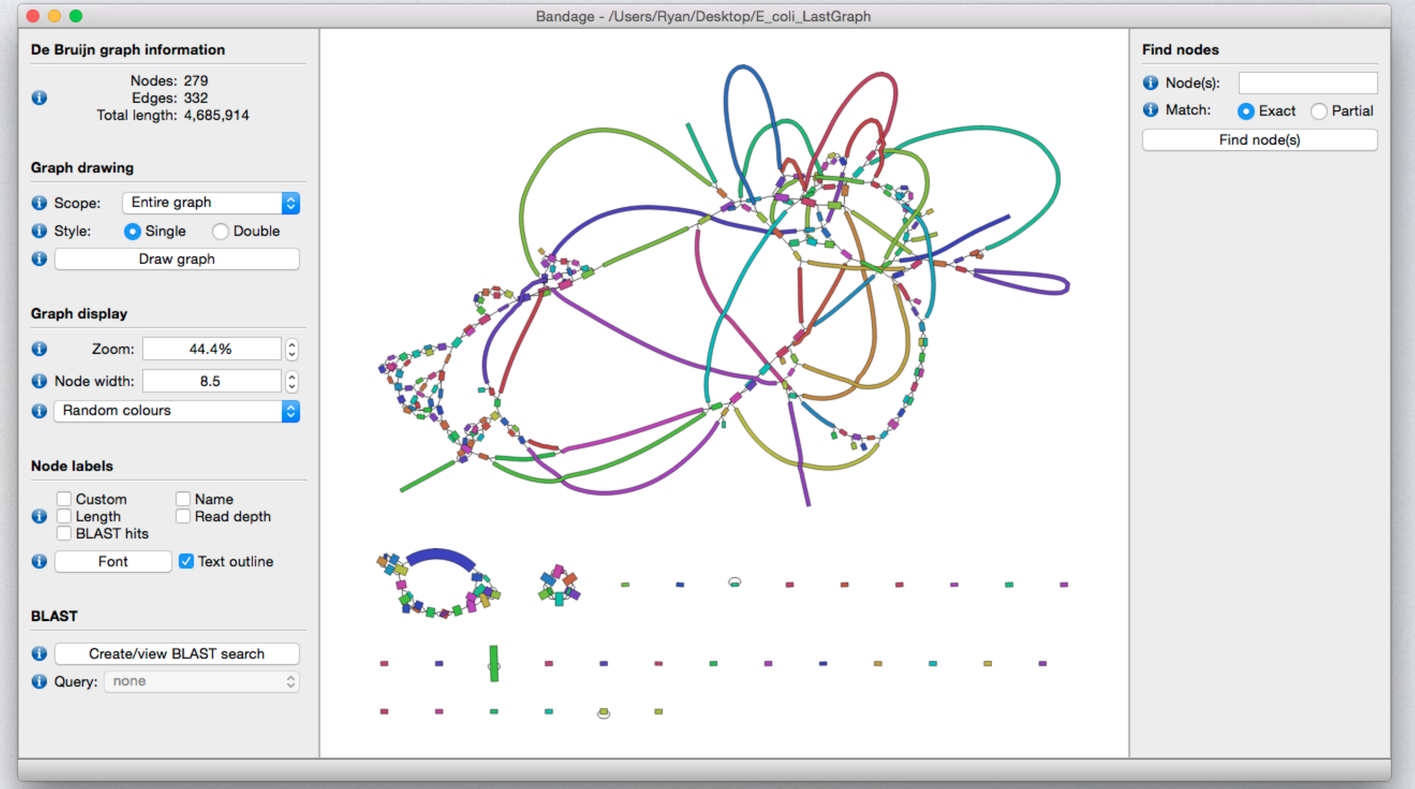
Navigating

De novo

Assembly

Graphs

Easily



http://rrwick.github.io/Bandage/
https://github.com/rrwick/Bandage/

Wick R.R., Schultz M.B., Zobel J. & Holt K.E. (2015). Bandage: interactive visualisation of de novo genome assemblies. Bioinformatics, 31(20), 3350-3352.

(Some) Bandage Issues

- Development effectively stopped several years ago
  - No future updates
  - Use of outdated frameworks
- Slow loading of somewhat larger graphs
  - Do not try to visualize metagenome or something similar on your laptop!
- Huge memory consumption
  - Requires lots of hacks & preprocessing
- No support for standard GFA features like paths or walks
- No ways to save graph layout
- Only BLAST for sequence-to-graph alignment
- …

Are you guys still working on this? #102

Open **Unaimend** opened this issue on 27 May 2021 · 1 comment

2022 update

Bandage is now about seven years old! Despite its popularity, I sadly don't have the time to give it the attention it deserves. Also, its codebase is too large and my C++/Qt skills are too rusty for me to make anything more than trivial changes.

Bandage-NG

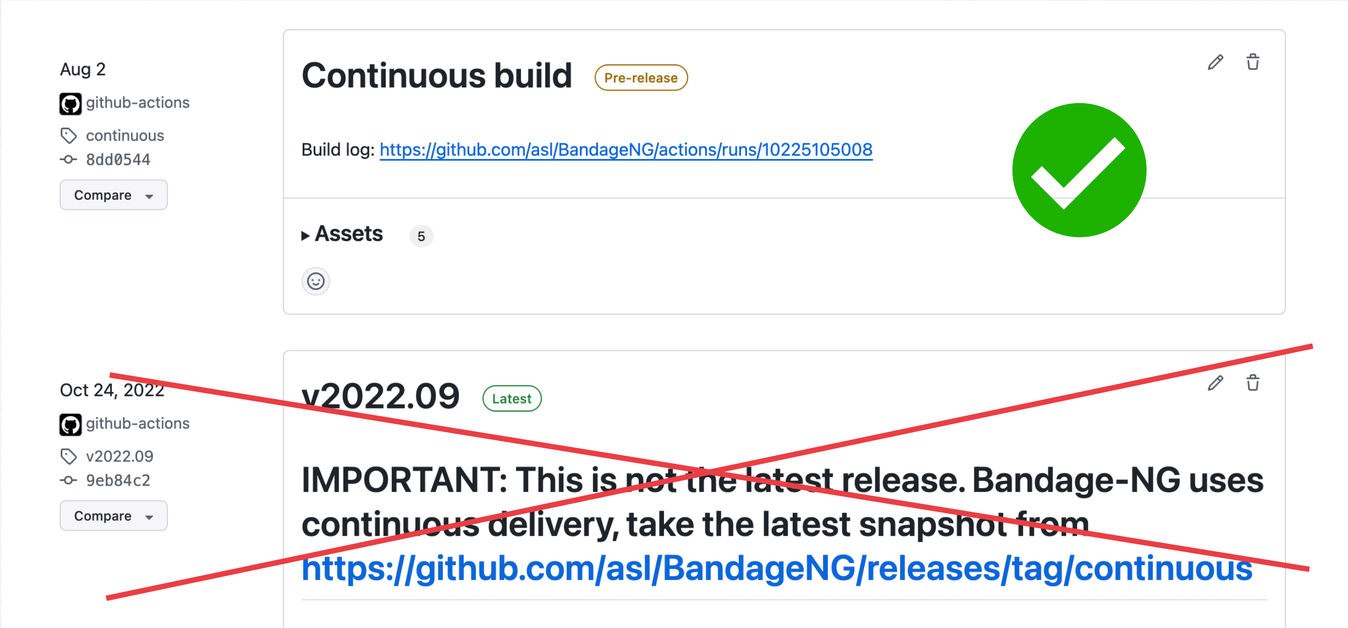
- Fork of original Bandage
- Contains patches that I maintained locally
- Some new development & improvements
  - New features
  - Speedups
- Tries to utilize better and latest approaches for certain things
  - Release management, CI, etc.

	File Size, Gb	Time to load, s	Speedup	RAM, Gb	Improvement
SRR103663.gfa	0.47	10.12	6.8x	1.05	3.16x
SRR6201737.gfa	0.81	6.97	13.7x	1.5	3.7x
sheepB.a_ctg.gfa	2.2	12.58	2.7x	0.68	11.67x
sheepB.p_ctg.gfa	5.4	23.7	4.2x	1.5	11.1x

Bandage vs Bandage-NG 0.9.5 (even better now!)

Downloading & Installing

- Get it from https://github.com/asl/BandageNG/releases/tag/continuous
- We use `continuous` builds: any change after testing triggers a new binary snapshot build
- Binaries for:
  - x86-64/Linux (via AppImage)
  - MacOS X (both x86-64 and ARM64)
  - Windows (experimental)



Command Line Interface

- Some (but not all) functionality is available via CLI interface
- Subcommands:
  - `load` – load graph into GUI
  - `image` – produce image of a graph
  - `info` – display graph information
  - `reduce` – extract subgraph
  - `querypaths` – emit paths of some BLAST queries
  - `layout` – layout the graph and save the layout
- Additional options to customize subcommand output

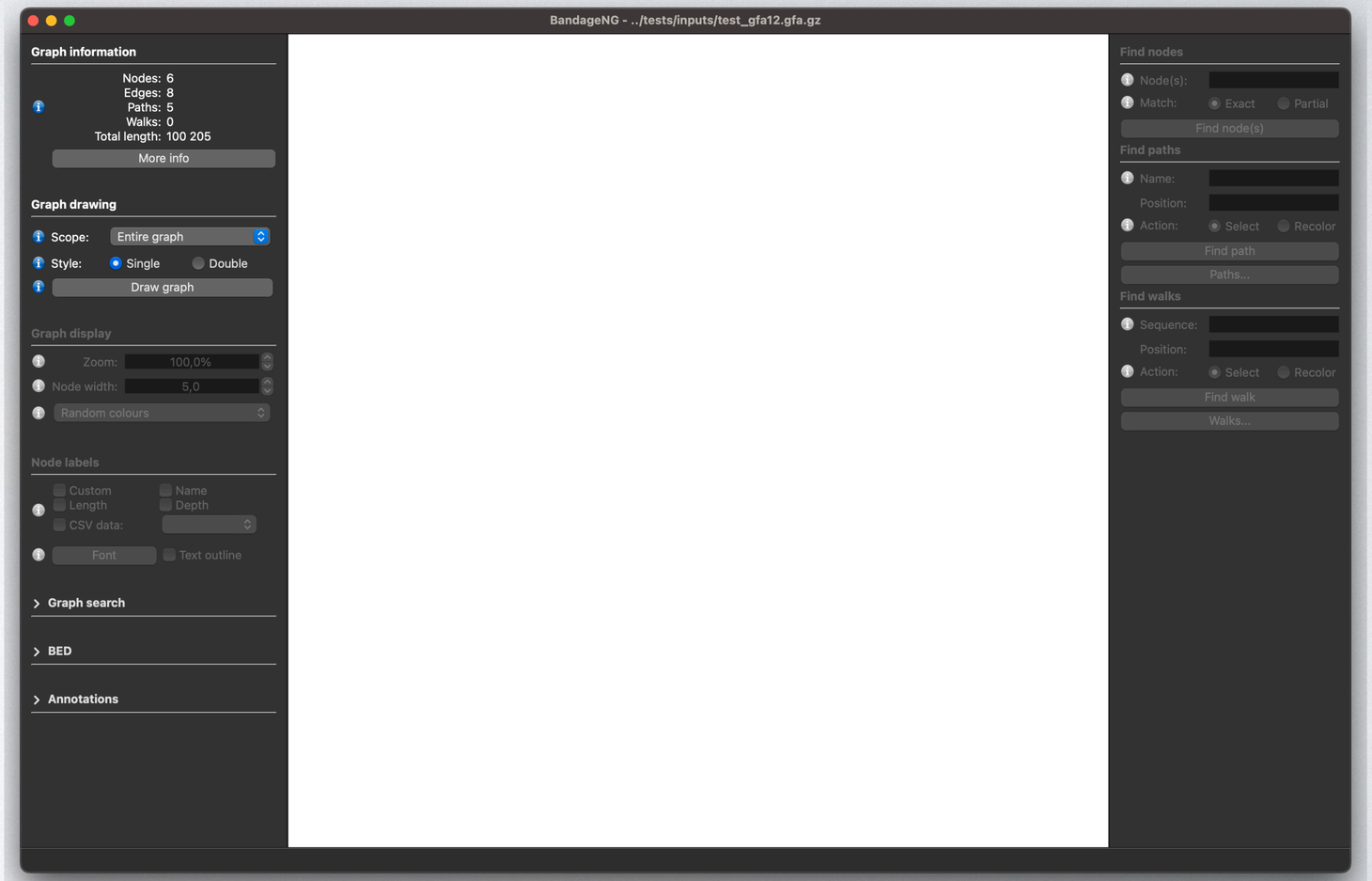
```
  ____                  _
 |  _ \                | |
 | |_) | __ _ _ __   __| | __ _  __ _  __
 |  _ < / _` | '_ \ / _` |/ _` |/ _` |/ _ \
 | |_) | (_| | | | | (_| | (_| | (_| |  __/
 |____/ \__,_|_| |_|\__,_|\__,_|\__, |\___|
                                  __/ |
                                 |___/
Version: 2024.07-dev

Usage: ./BandageNG.app/Contents/MacOS/BandageNG [OPTIONS] [SUBCOMMAND]

Options:
  -h,--help                   Print this help message and exit
  --version                   Display program version information and exit
  --helpall
[Option Group: Graph scope]
  These settings control the graph scope. If the aroundnodes scope is used, then the
  Options:
    --scope SCOPE:value in {entire->0,aroundnodes->1,aroundblast->4,depthrange->5} OR
                                Graph scope, from one of the following options: entir
    --exact,--partial{false}    Choose between exact or partial node name matching (d
    --distance INT:INT in [0 - 100] [0]
                                The number of node steps away to draw for the aroundn
    --mindepth FLOAT:FLOAT in [0 - 1e+06] [10]
                                The minimum allowed depth for the depthrange scope
    --maxdepth FLOAT:FLOAT in [0 - 1e+06] [100]
                                The maximum allowed depth for the depthrange scope
    --query TEXT:FILE           A FASTA file of either nucleotide or protein sequence
    --nodes TEXT [0]            A comma-separated list of starting nodes for the arou
[Option Group: Graph size]
  Options:
```

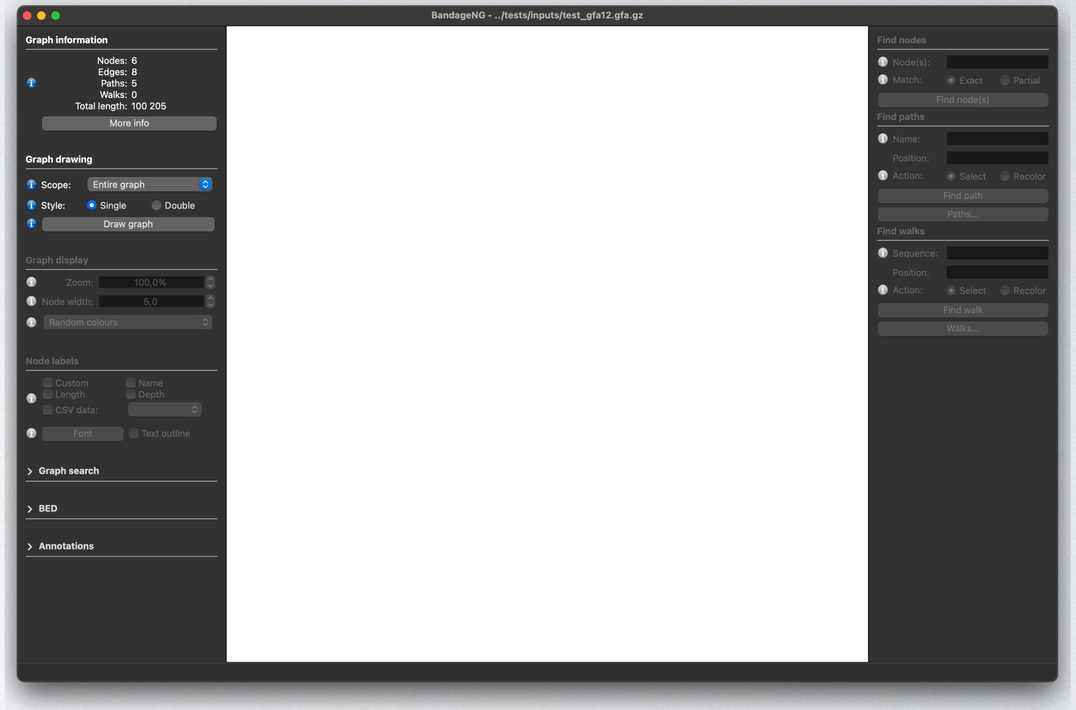
Loading Graphs

- Use load CLI command
- Or just go via File/Load graph



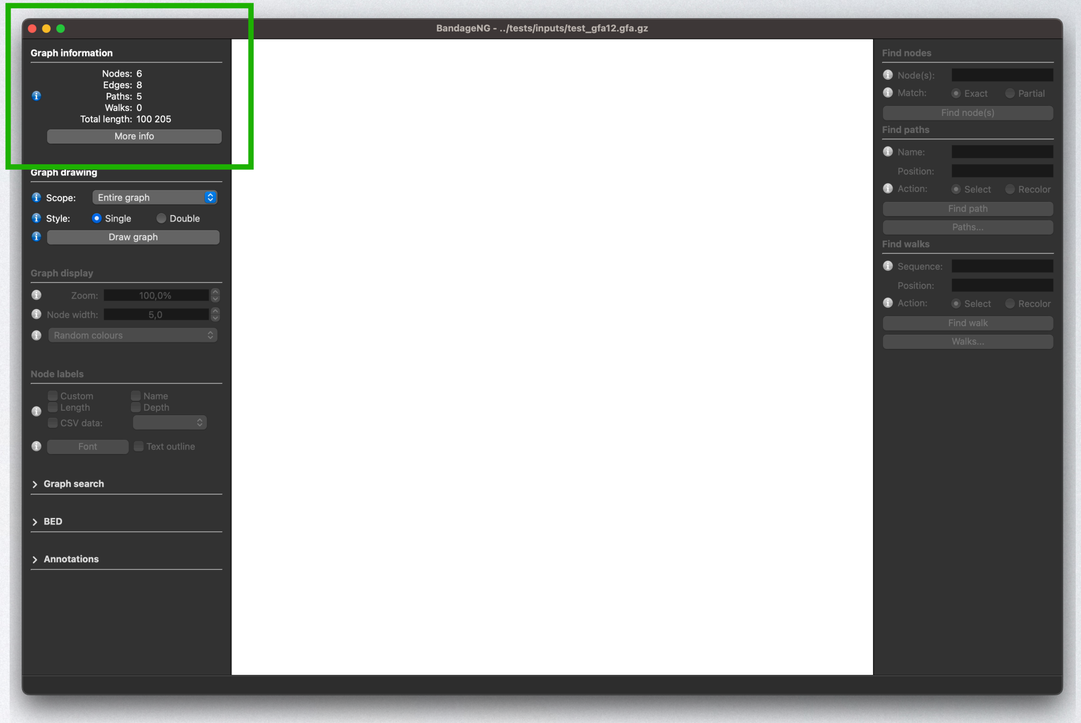
Graph Information

- Basic information about graph:
  - Number of nodes, edges, paths, walks, …
- More detailed information:
  - Connectivity
  - Coverage depth
  - Node lengths (N50, etc.)



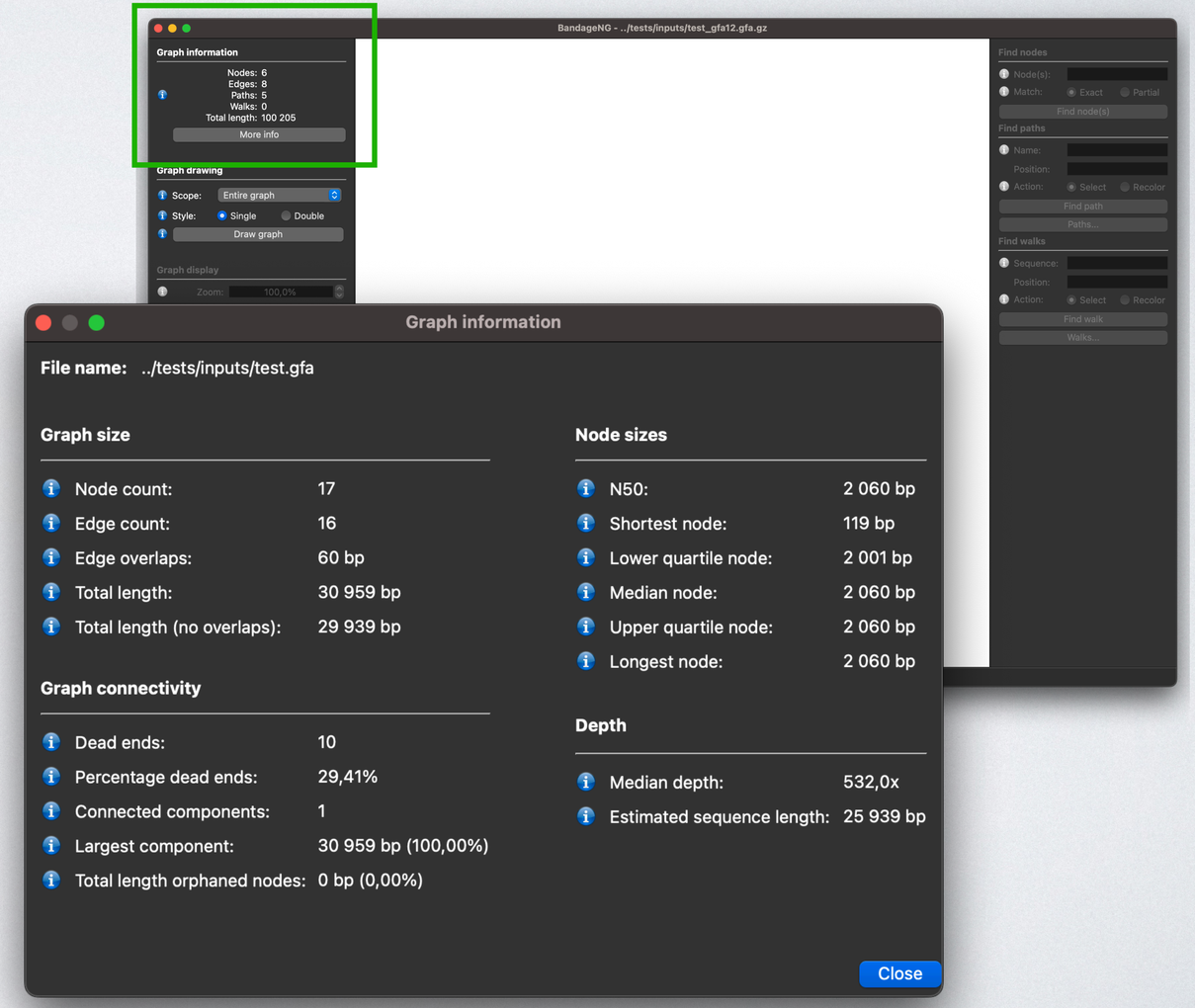
Graph Information

- Basic information about graph:
  - Number of nodes, edges, paths, walks, …
- More detailed information:
  - Connectivity
  - Coverage depth
  - Node lengths (N50, etc.)



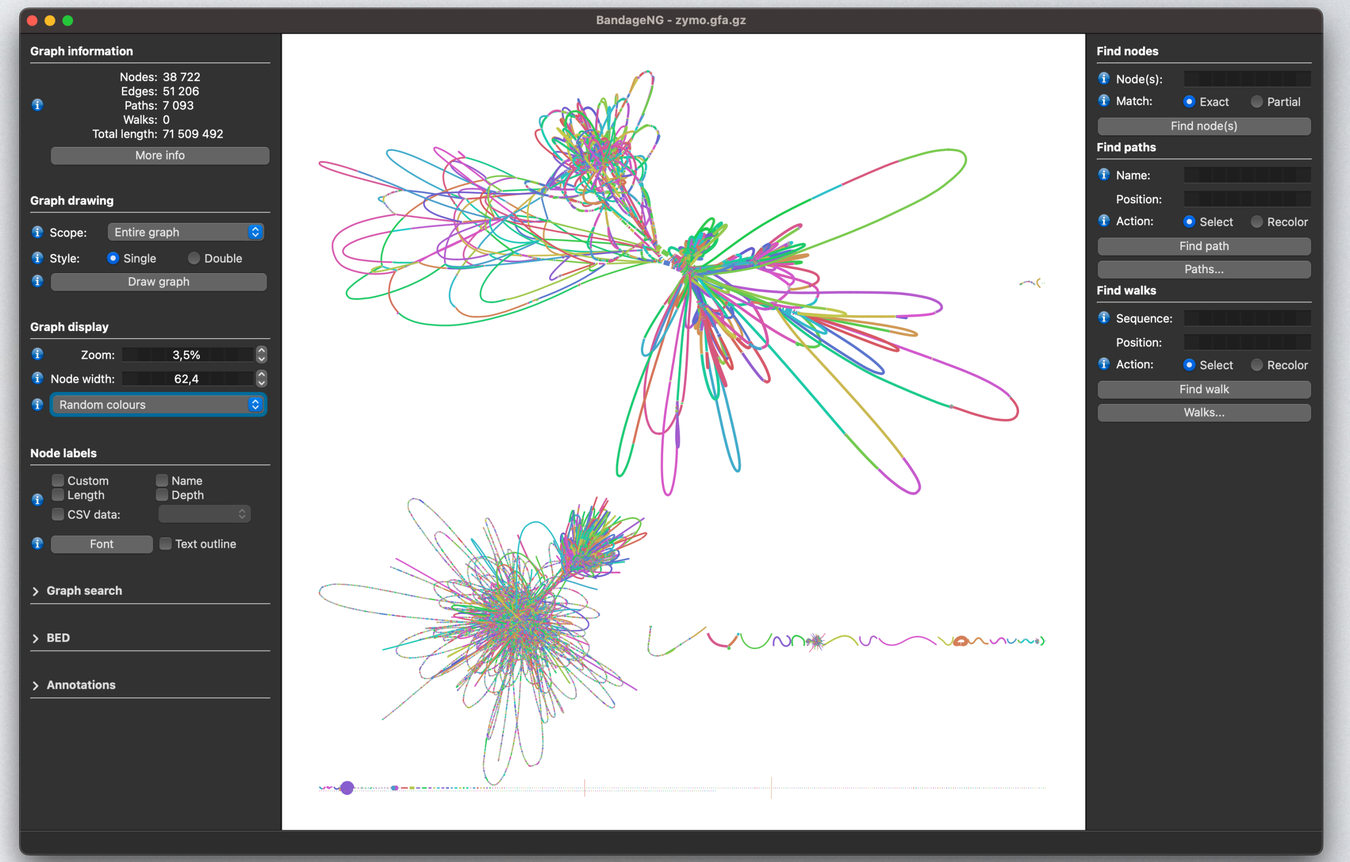
Graph Information

- Basic information about graph:
  - Number of nodes, edges, paths, walks, …
- More detailed information:
  - Connectivity
  - Coverage depth
  - Node lengths (N50, etc.)



Graph Layout

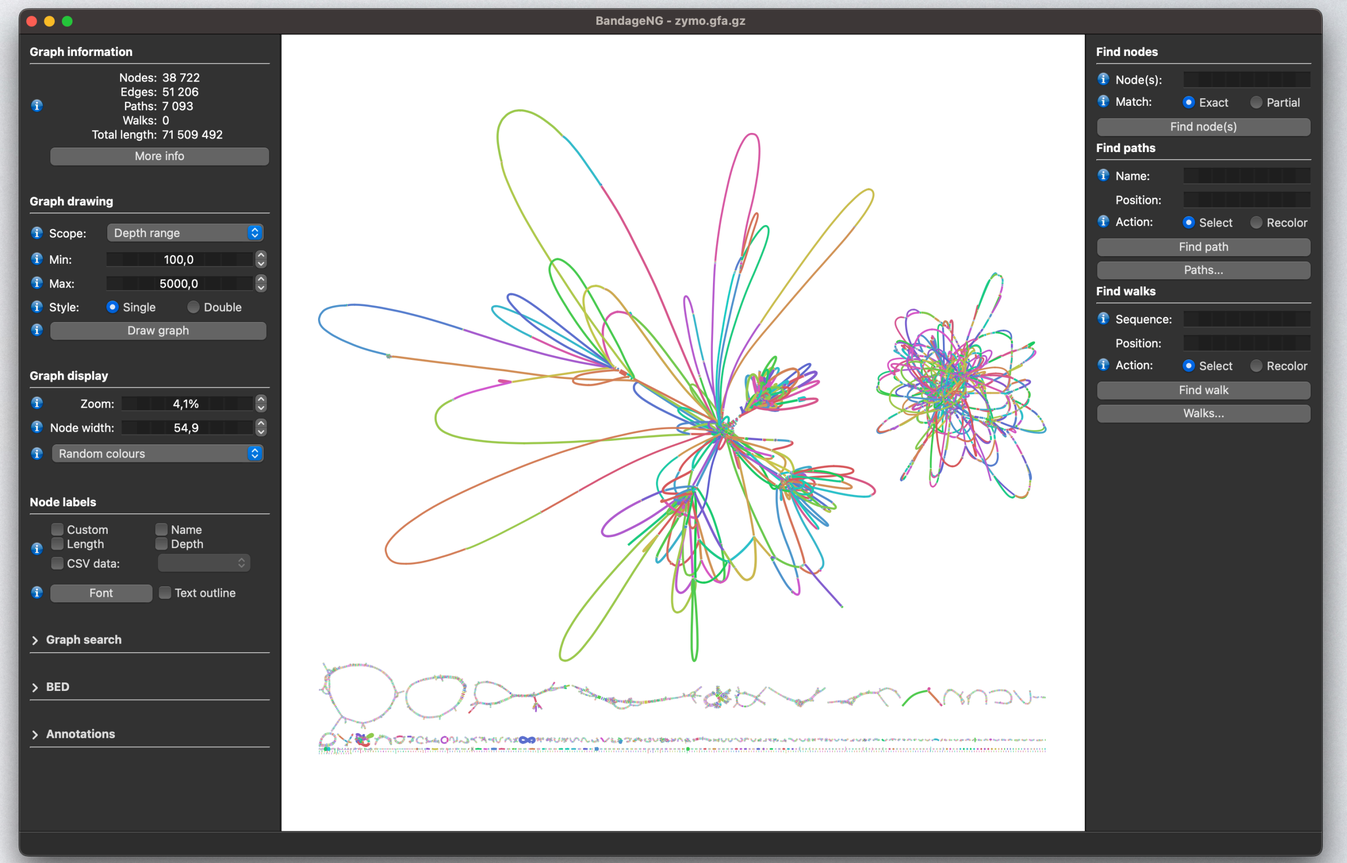
- Visualization of graph on a plane
- Not all graphs could be drawn without self-intersections
- Graphs are large: computational intensive task
- Uses state-of-the-art OGDF library (fast multipole multilevel layout algorithm)
- Layout in parallel:
  - Splits graph into connectivity components
  - Layout each independently in separate threads
  - And then reassemble into a complete picture



Stefan Hachul, Michael Jünger: *Drawing Large Graphs with a Potential-Field-Based Multilevel Algorithm*. GD '04, LNCS 3383, pp. 285-295, 2004

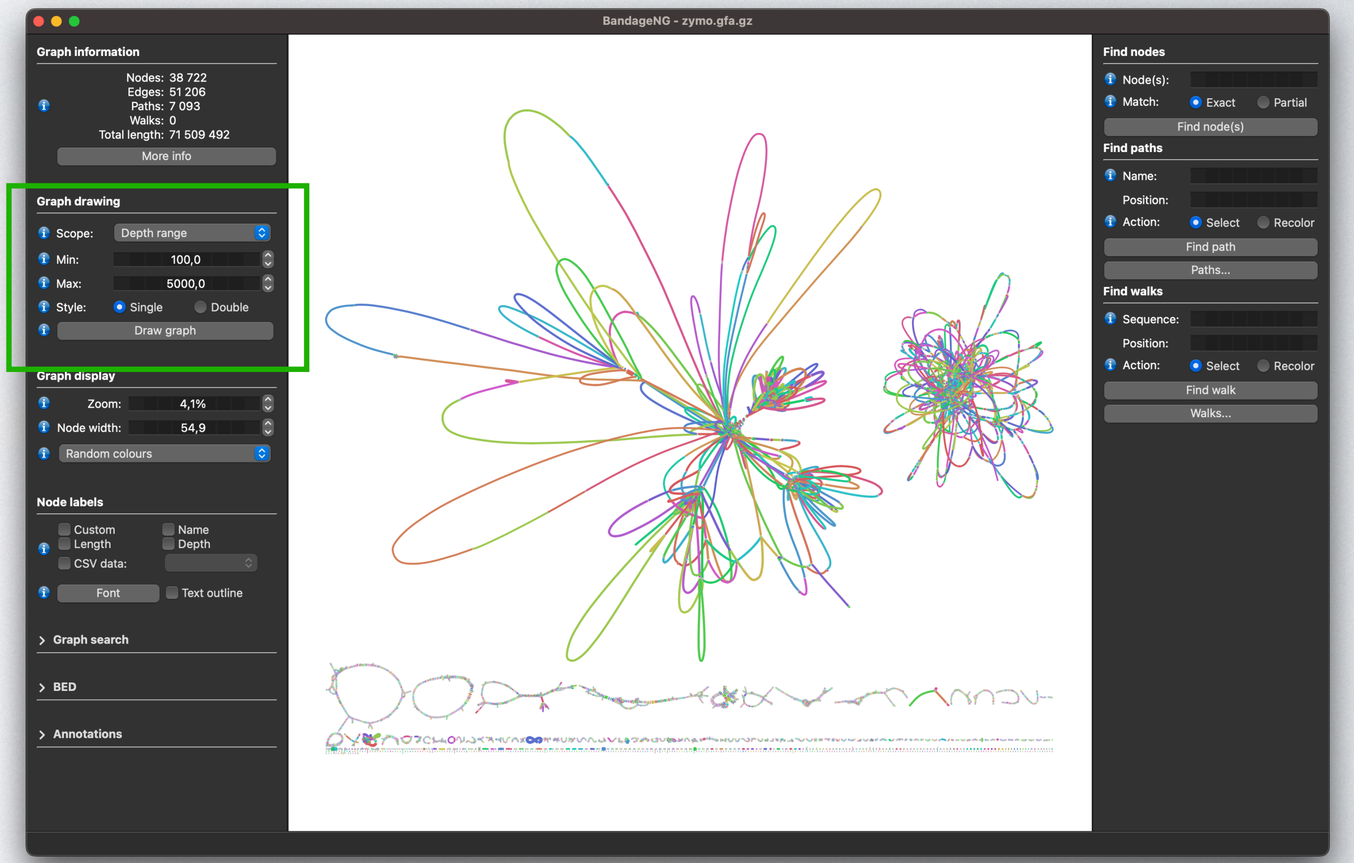
Drawing Scopes

- Limit the layout & drawing to subgraph
  - Entire graph
  - Around nodes up to certain distance
  - Around paths / walks up to certain distance
  - Around query paths up to certain distance
  - Depth range



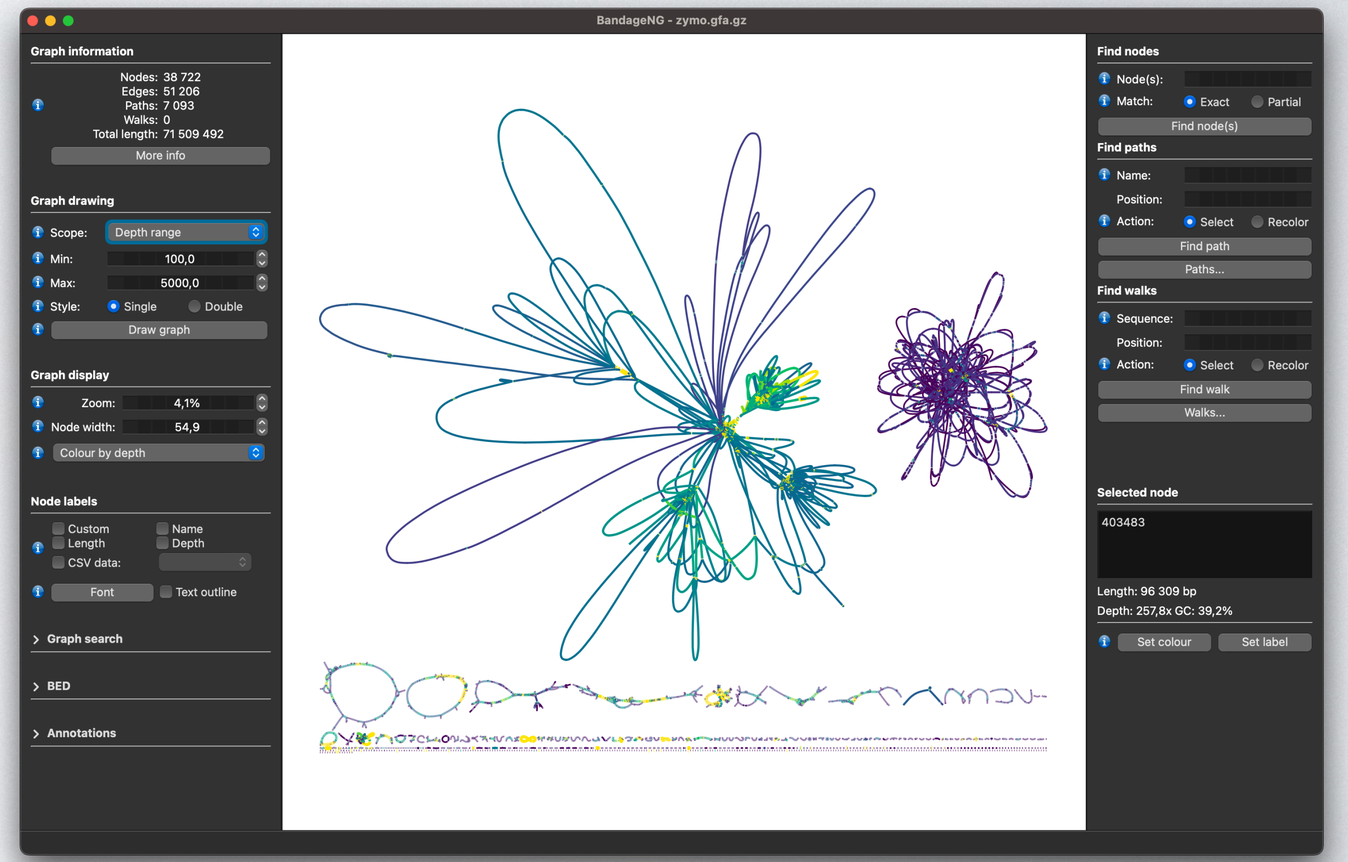
Drawing Scopes

- Limit the layout & drawing to subgraph
  - Entire graph
  - Around nodes up to certain distance
  - Around paths / walks up to certain distance
  - Around query paths up to certain distance
  - Depth range



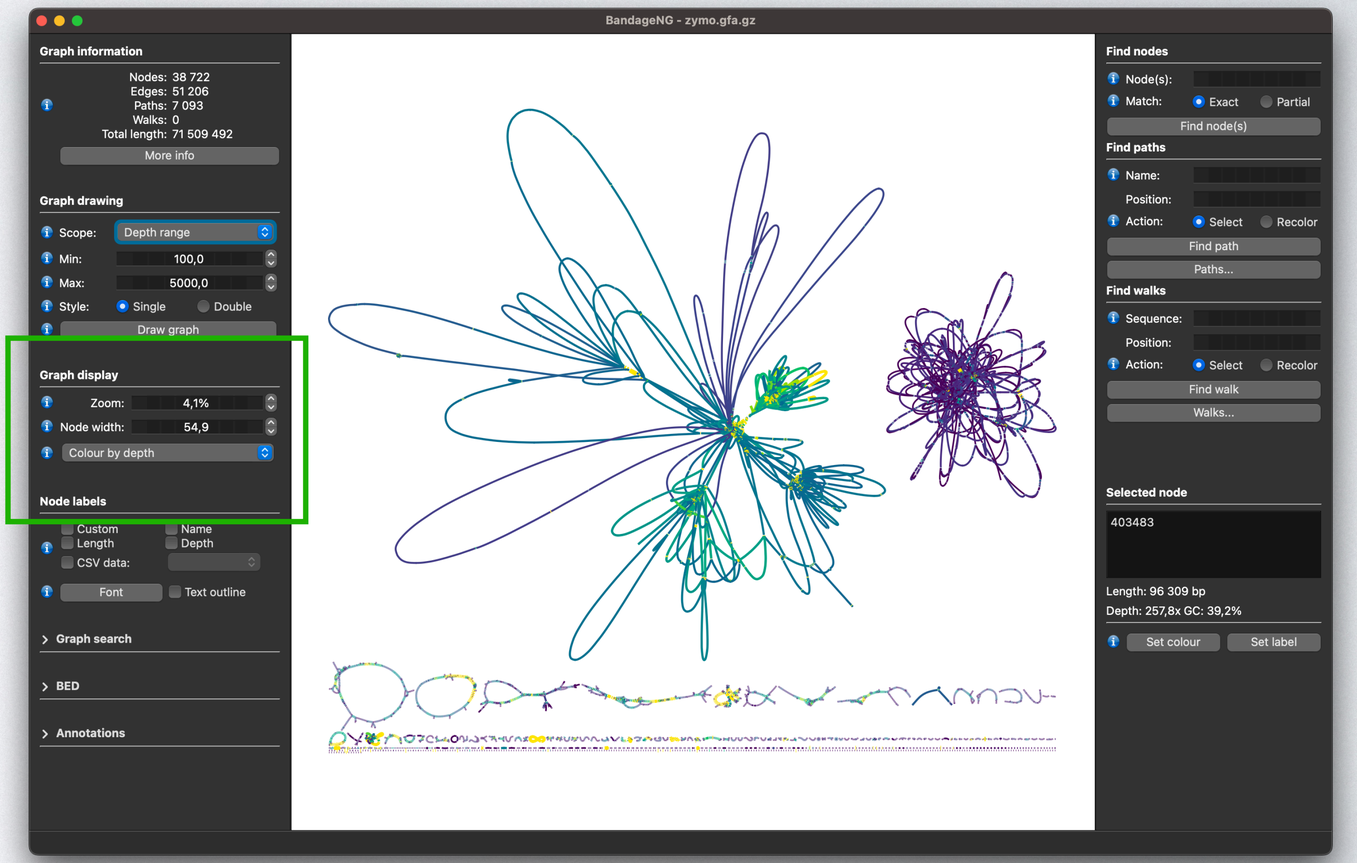
Node Colors

- Different ways to color a single node:
  - Gray
  - Random
  - Uniform
  - By depth
  - By contiguity
  - GC content
  - By GFA tag
  - By CSV column



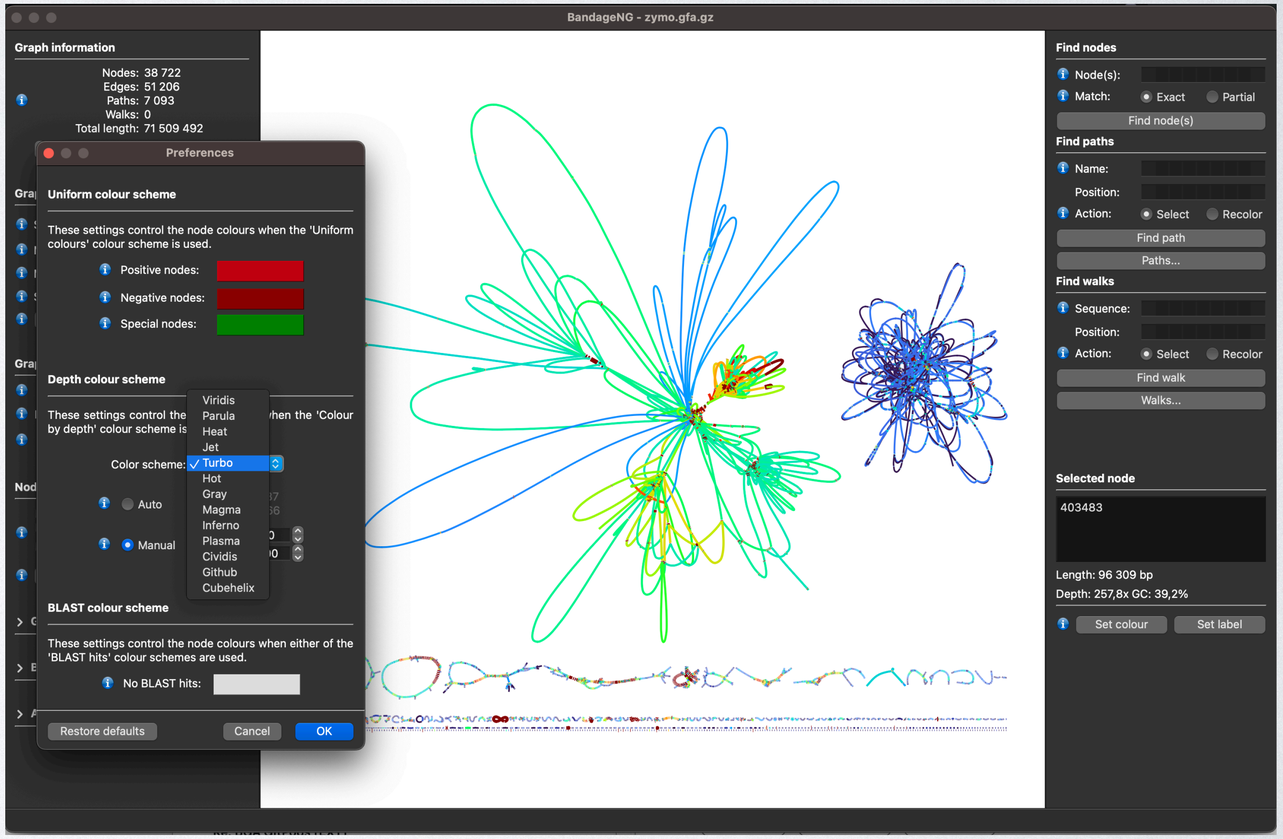
Node Colors

- Different ways to color a single node:
  - Gray
  - Random
  - Uniform
  - By depth
  - By contiguity
  - GC content
  - By GFA tag
  - By CSV column



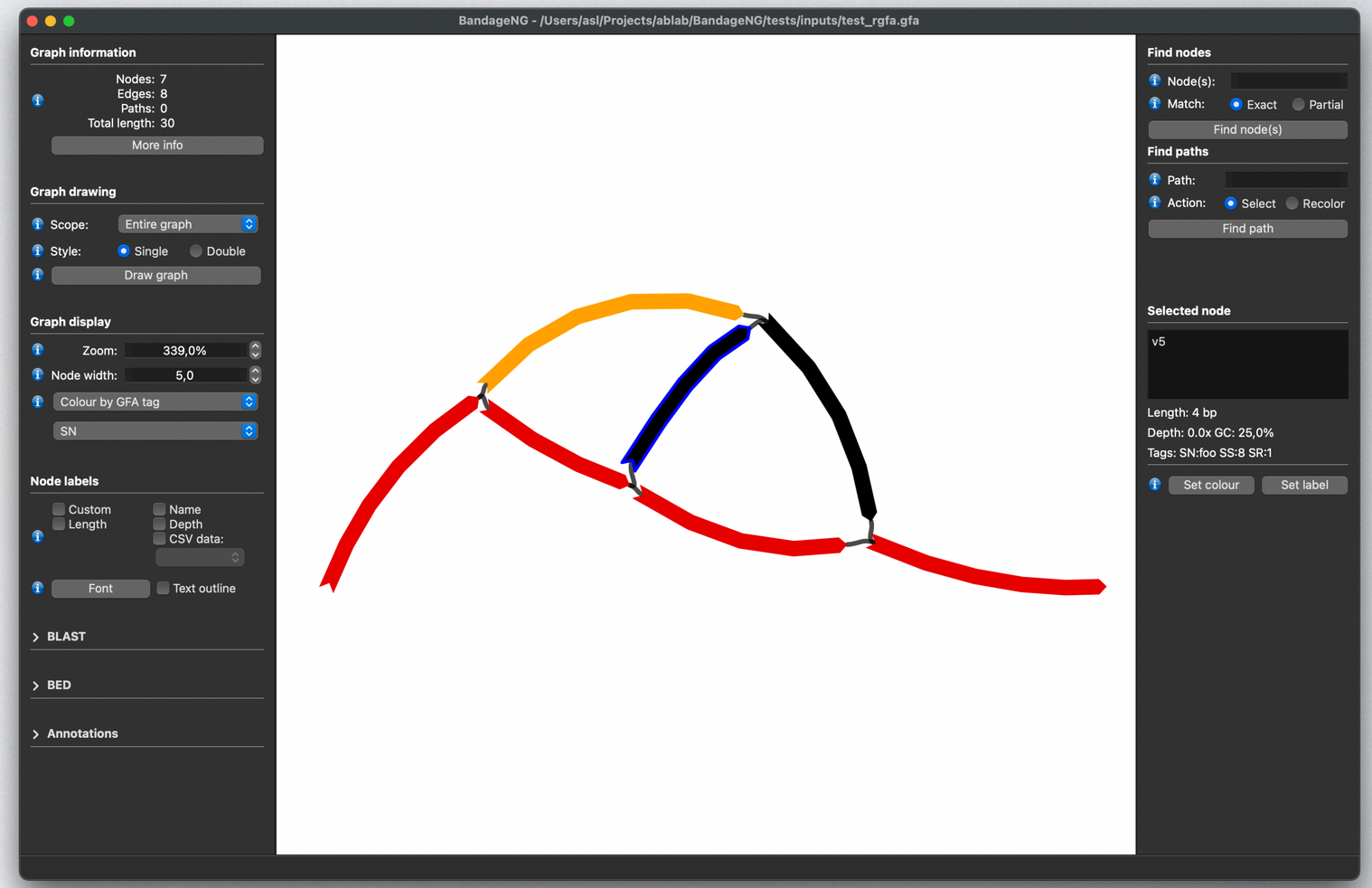
Color Schemes

- Used when nodes are colored by depth or GC content
- Set of some top-quality colormaps
- Including colorblind-friendly ones



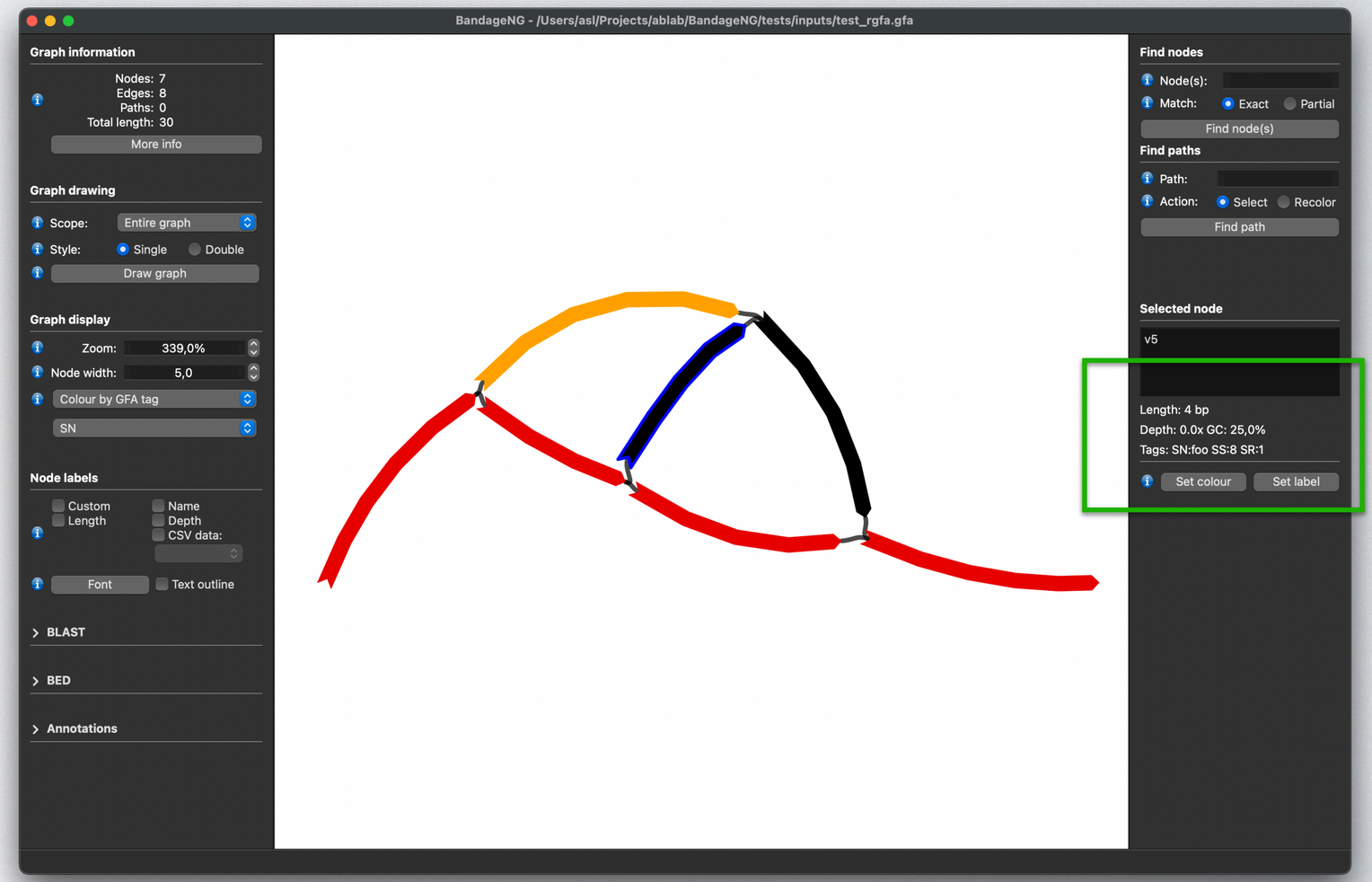
Color By Tag / rGFA support

- Show optional tags
- Color by tag value



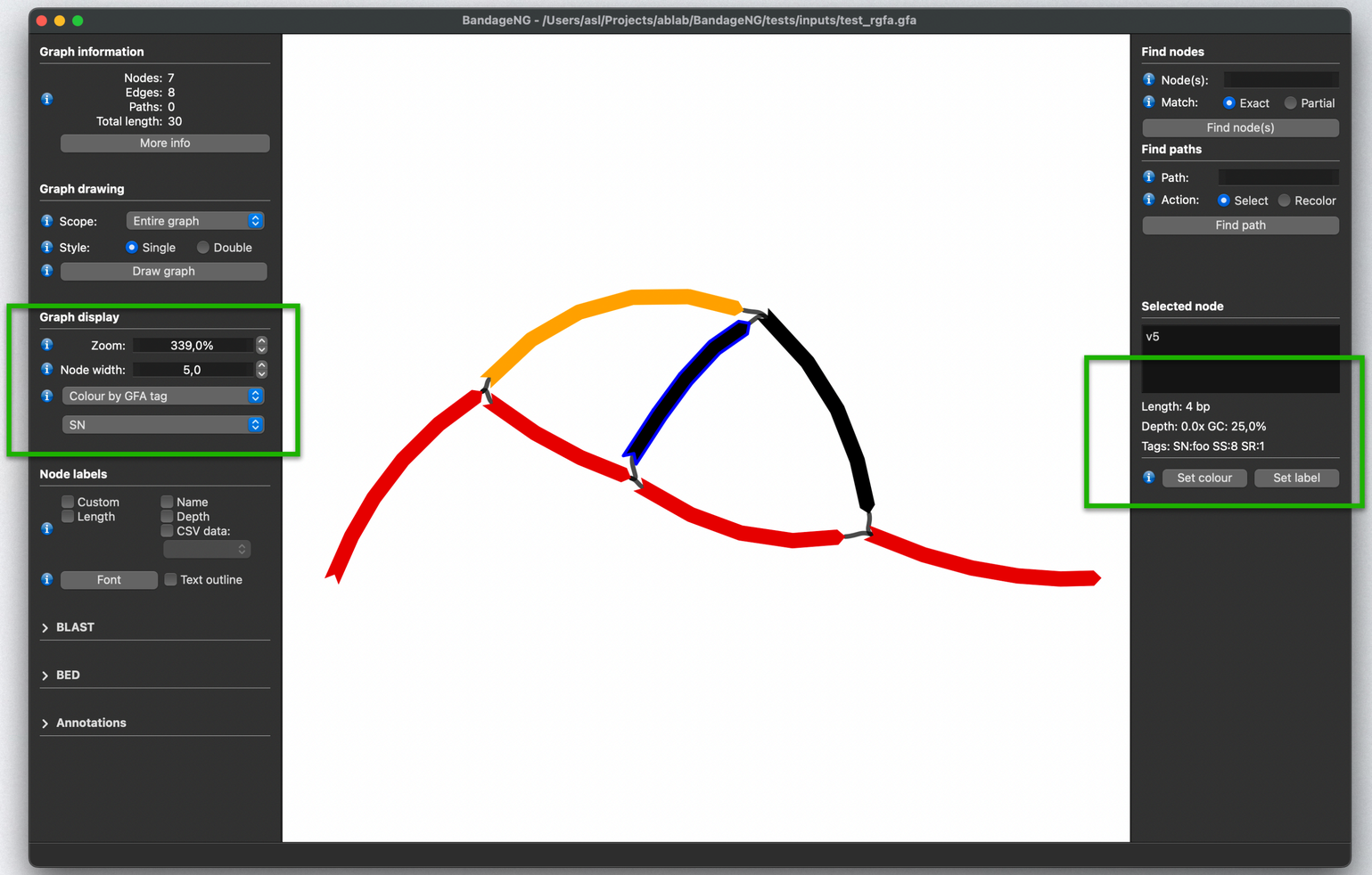
Color By Tag / rGFA support

- Show optional tags
- Color by tag value



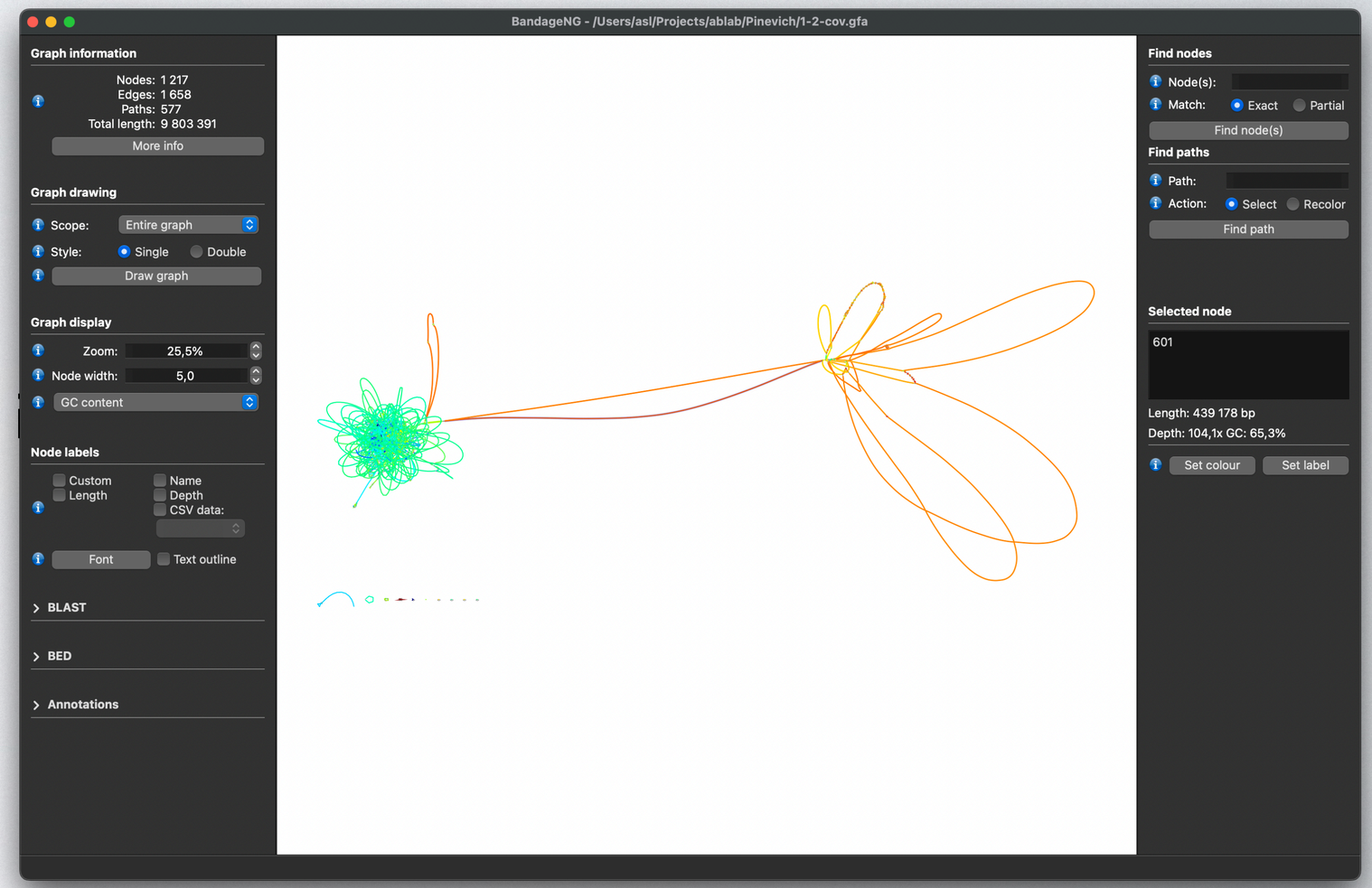
Color By Tag / rGFA support

- Show optional tags
- Color by tag value



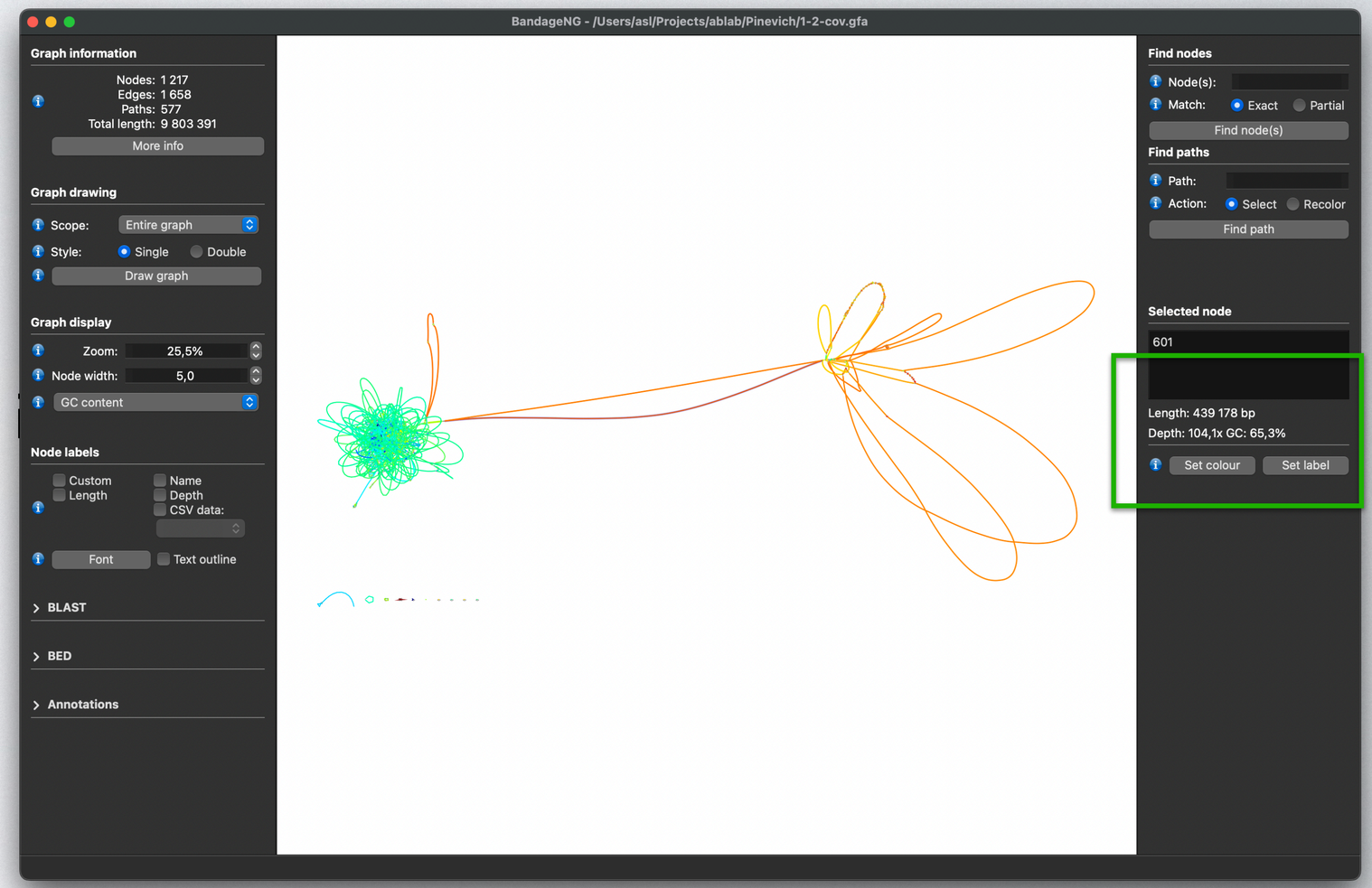
Color by GC content

- Quickly spot e.g. contamination i
assemblies



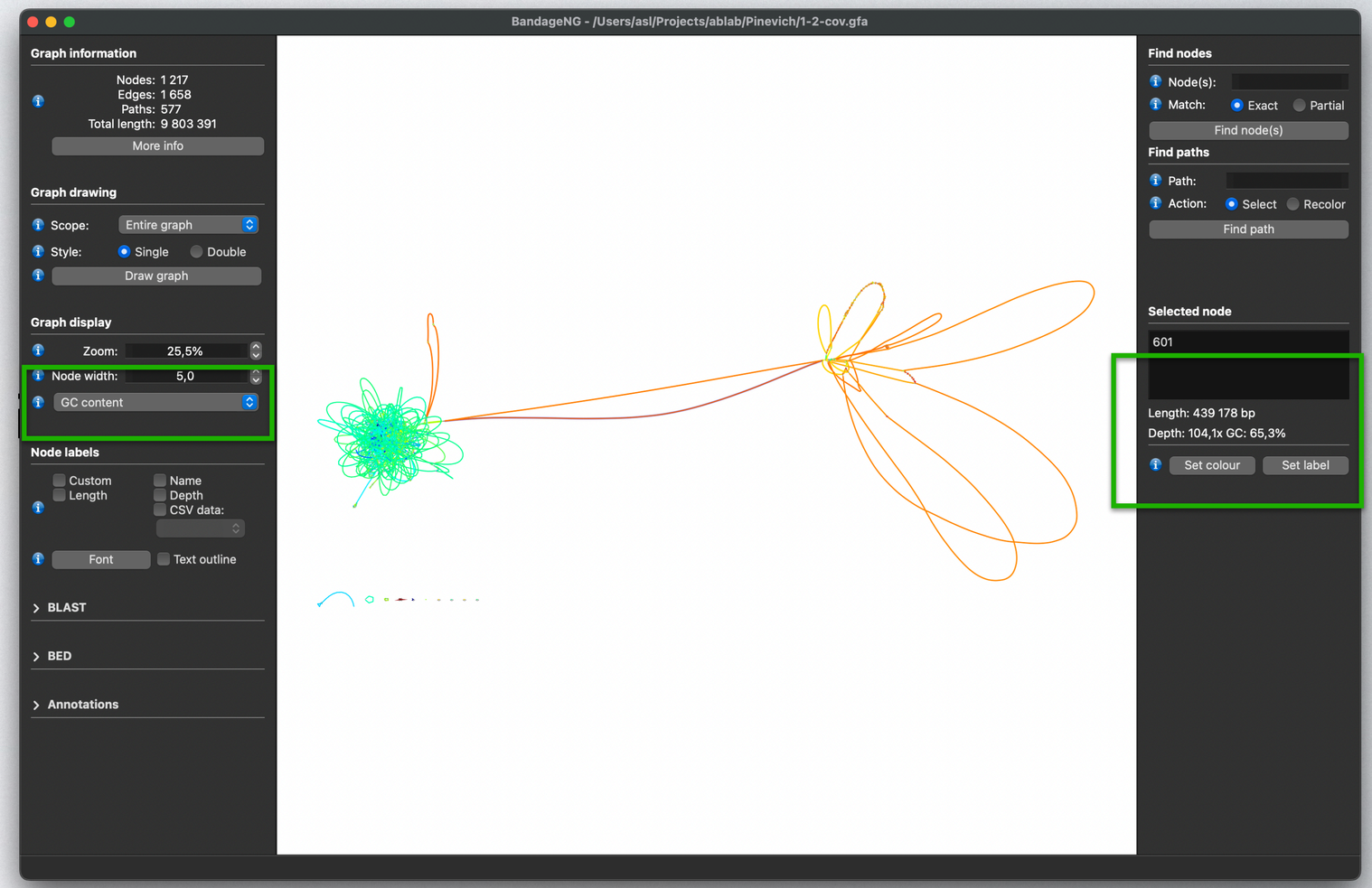
Color by GC content

- Quickly spot e.g. contamination i
assemblies



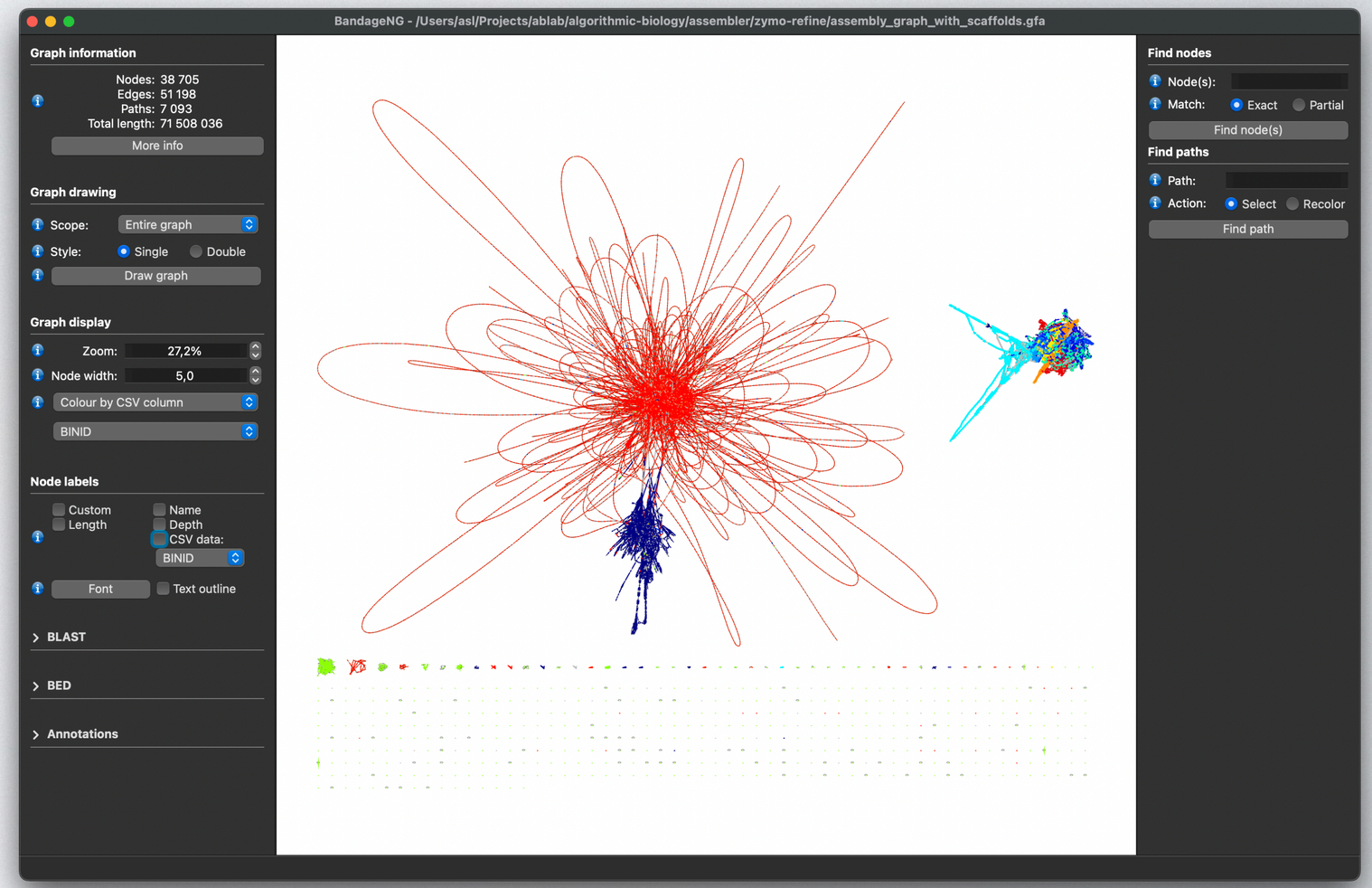
Color by GC content

- Quickly spot e.g. contamination in assemblies



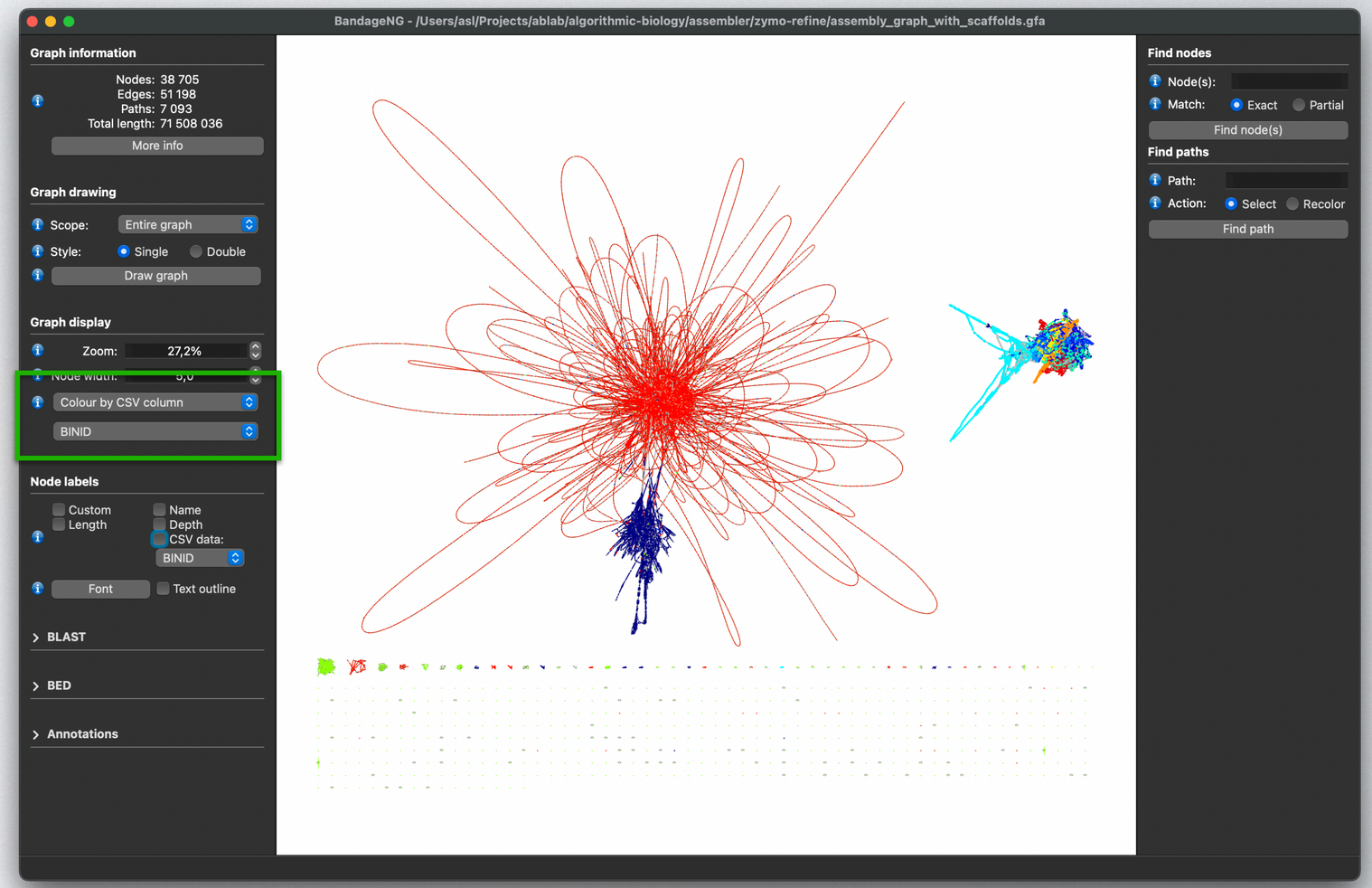
Use CSV colors

- Can use arbitrary columns
- And switch between them on-fly
- Visualize MAGs easily!



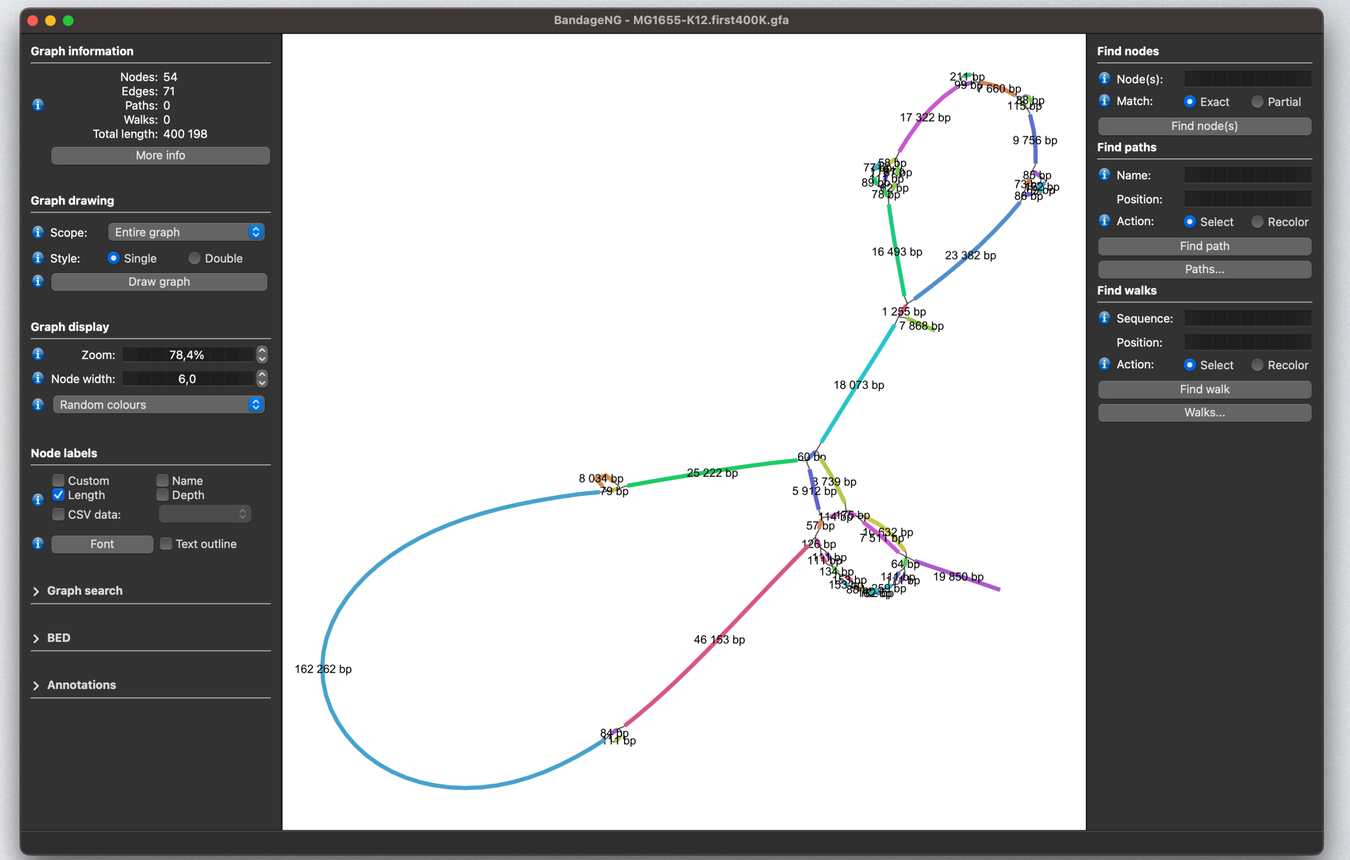
Use CSV colors

- Can use arbitrary columns
- And switch between them on-fly
- Visualize MAGs easily!



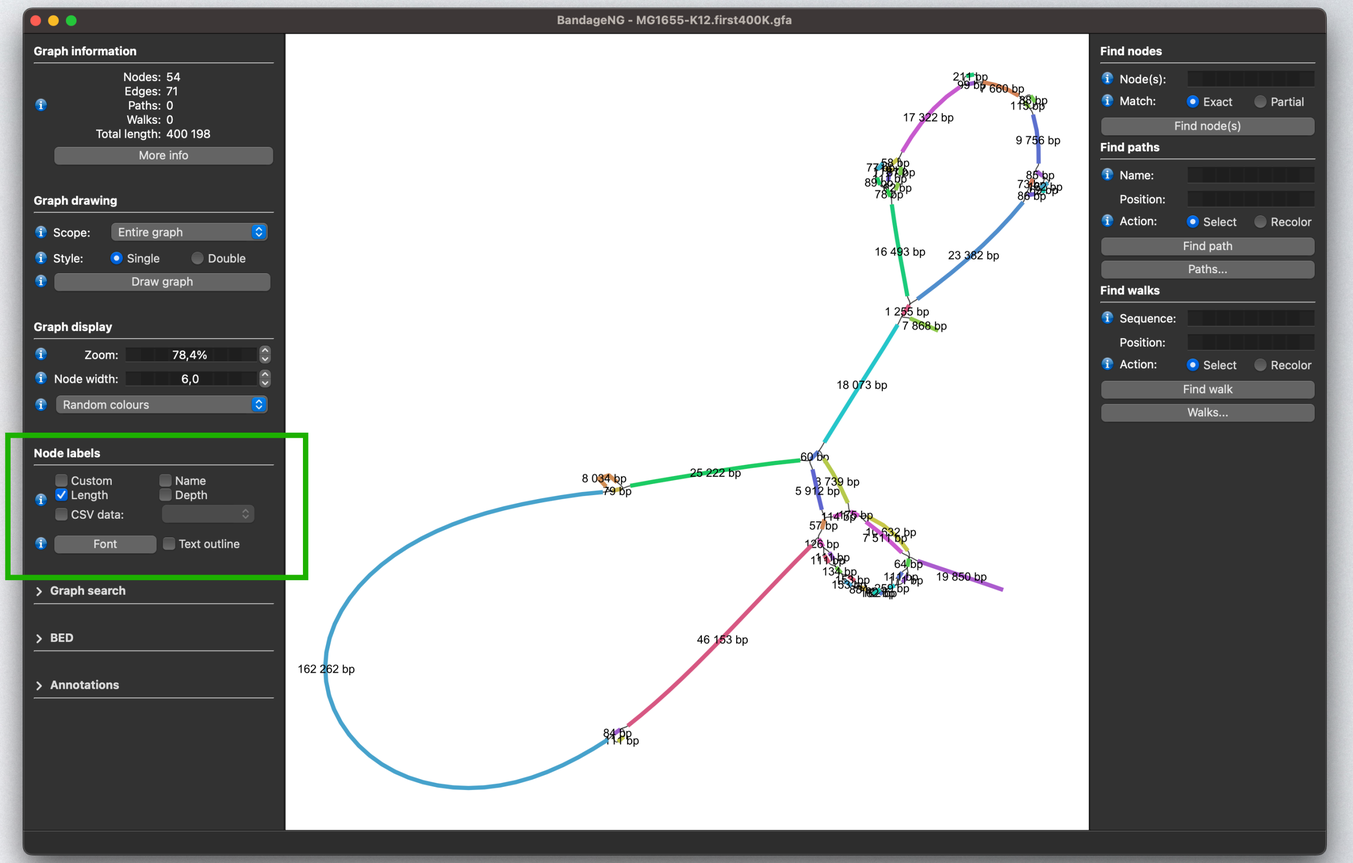
Node Labels

- By default nodes are unlabelled to show topology of the graph
- Additionally could show:
  - Length
  - Name
  - Depth
  - CSV column



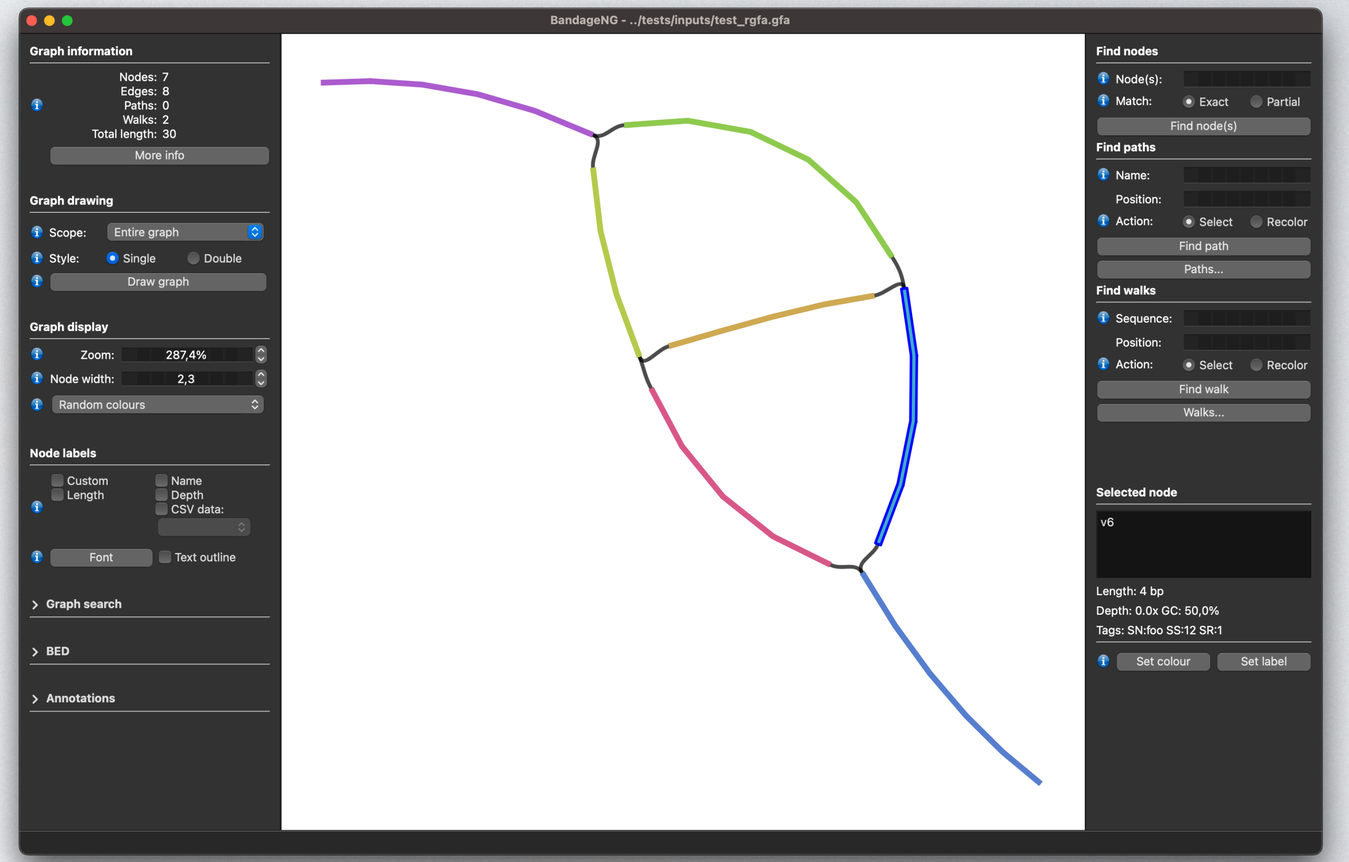
Node Labels

- By default nodes are unlabelled to show topology of the graph
- Additionally could show:
  - Length
  - Name
  - Depth
  - CSV column



Node Display

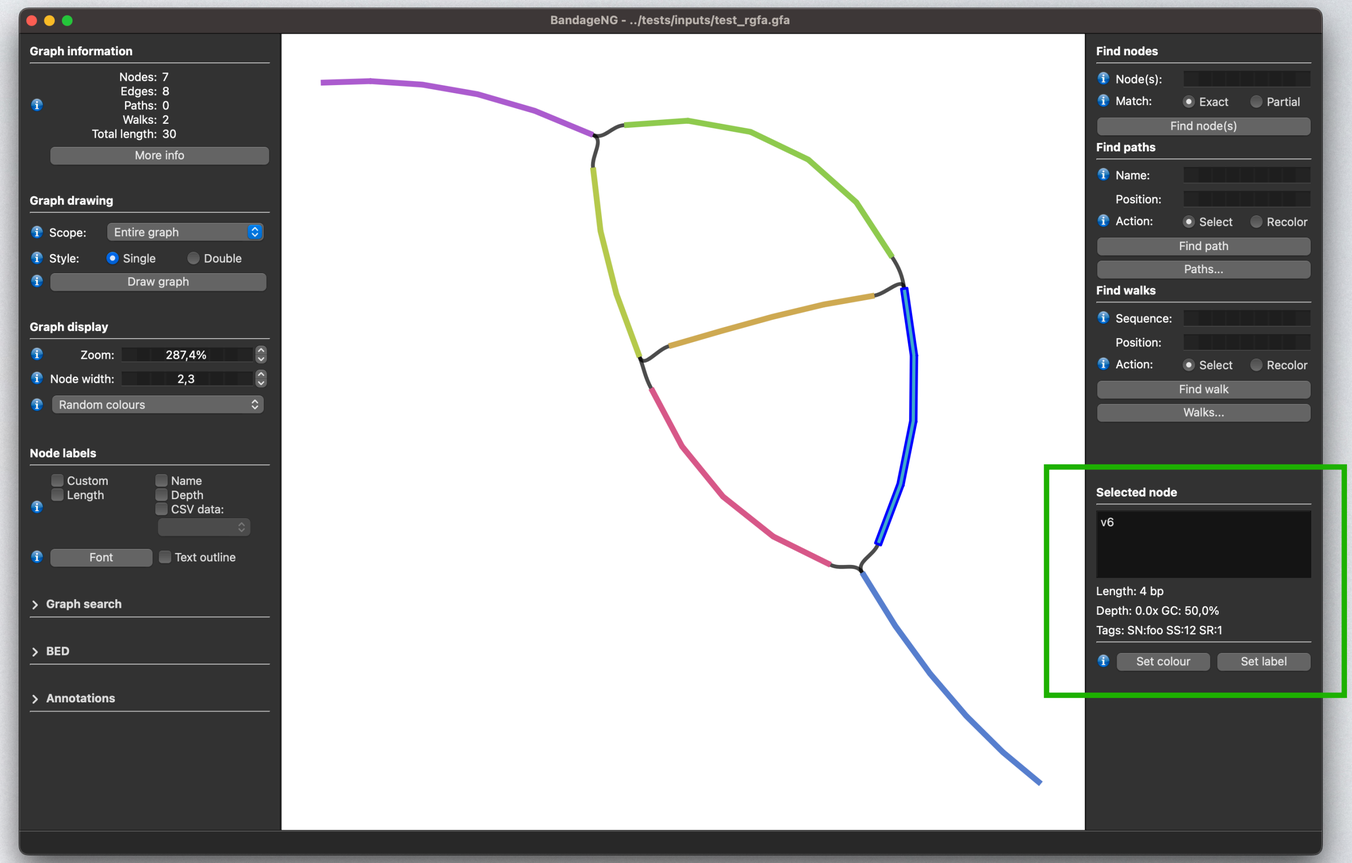
- Length
- Depth
- GC contents
- Tags (if any)
- Can change color / label



Multiple nodes could be selected

Node Display

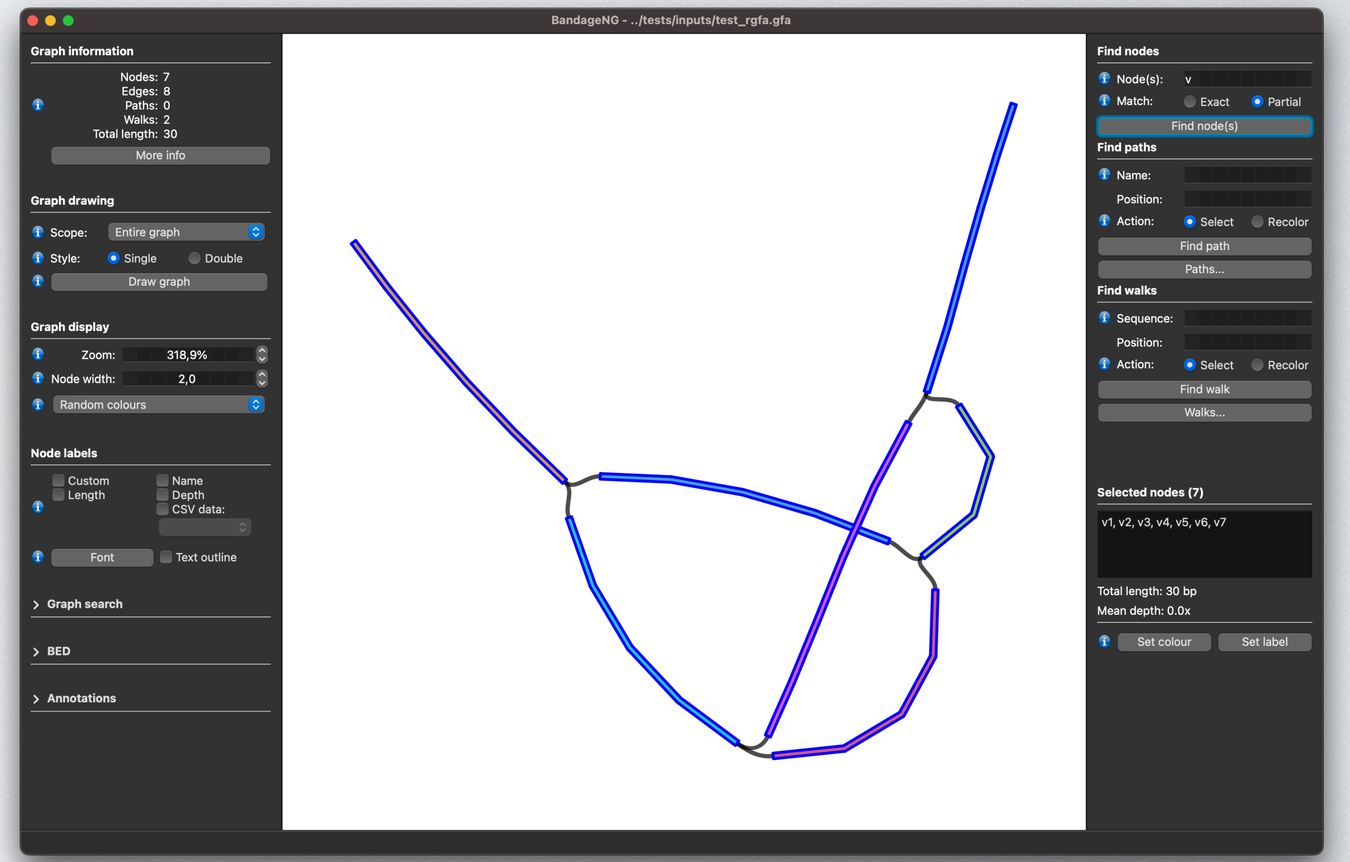
- Length
- Depth
- GC contents
- Tags (if any)
- Can change color / label



Multiple nodes could be selected

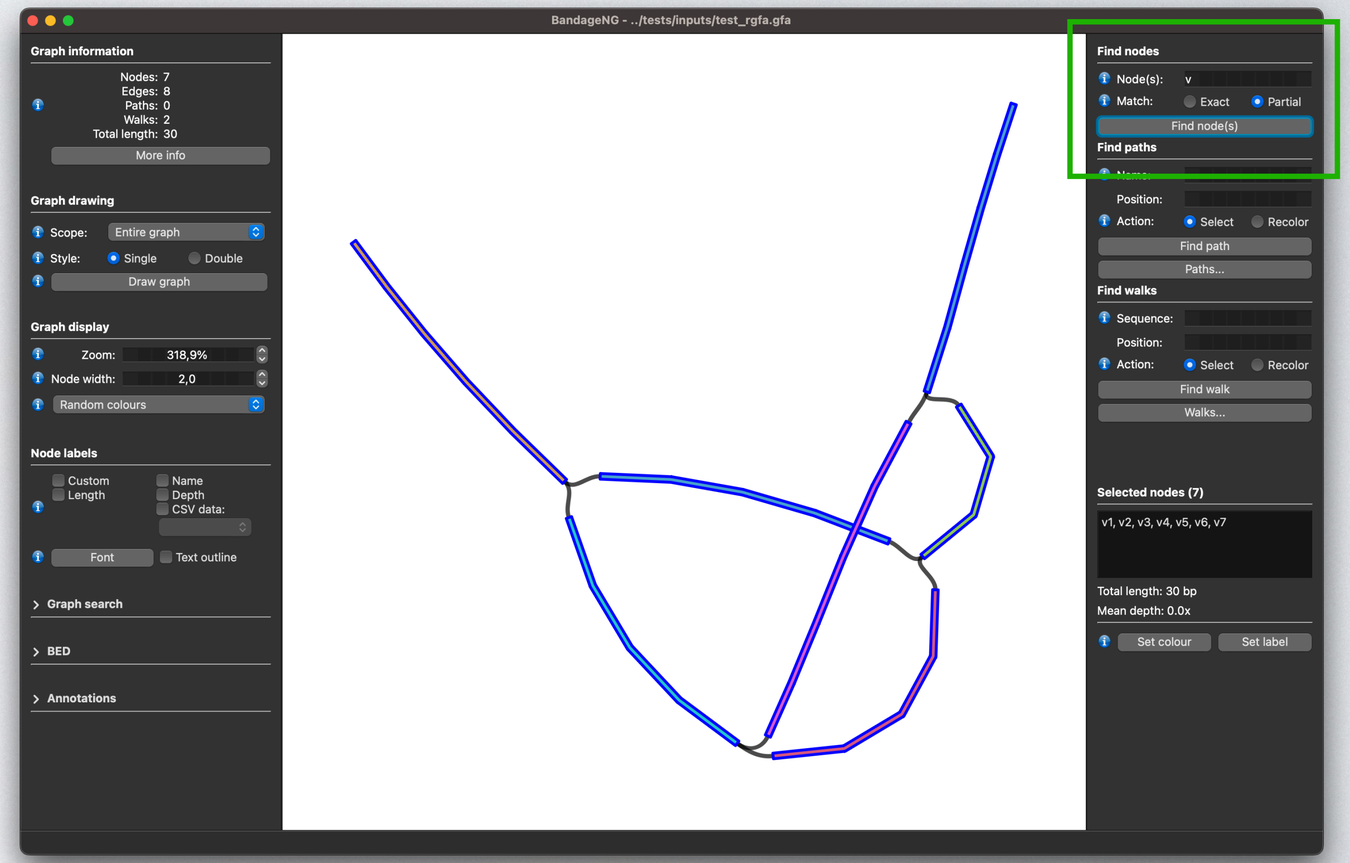
Node Lookup

- Node names:
  - Segment name + orientation (default)
  - Other name if renamed
  - In `single` mode the orientation could be omitted
- Lookup:
  - Select visible node(s) by name
  - Allows full and partial name match



Node Lookup

- Node names:
  - Segment name + orientation (default)
  - Other name if renamed
  - In `single` mode the orientation could be omitted
- Lookup:
  - Select visible node(s) by name
  - Allows full and partial name match



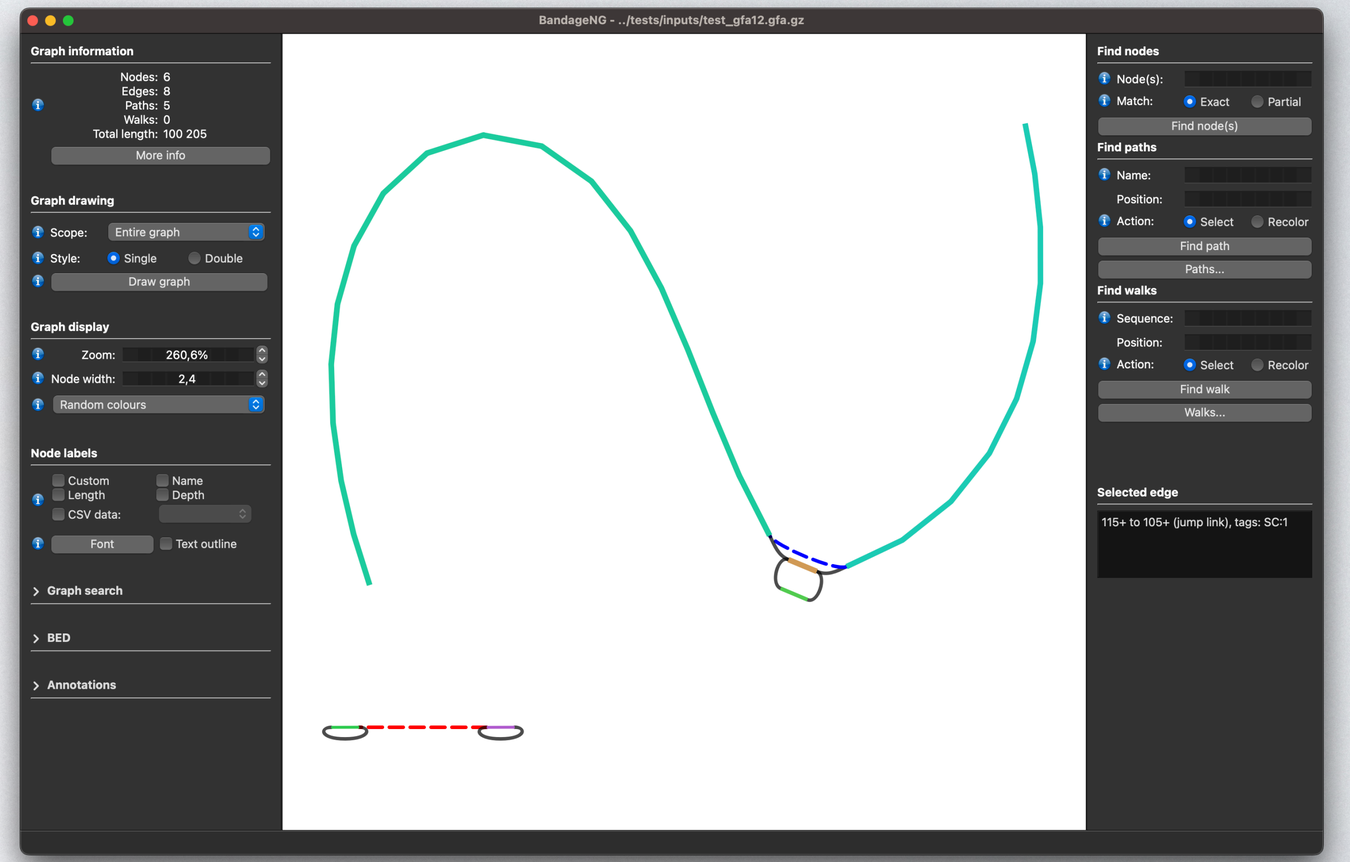
Depth Tags

- Multiple ways to specify depth for a node via GFA tags (ordered by priority):
  - `DP` or `dp`: average depth of the node
  - `KC`: k-mer count, node depth is calculated as `KC / length`, approximate
  - `RC`: read count, node depth is calculated as `RC / length`
  - `FC`: fragment count, node depth is calculated as `FC / length`

Prefer `DP` tags if possible!

Edge Display

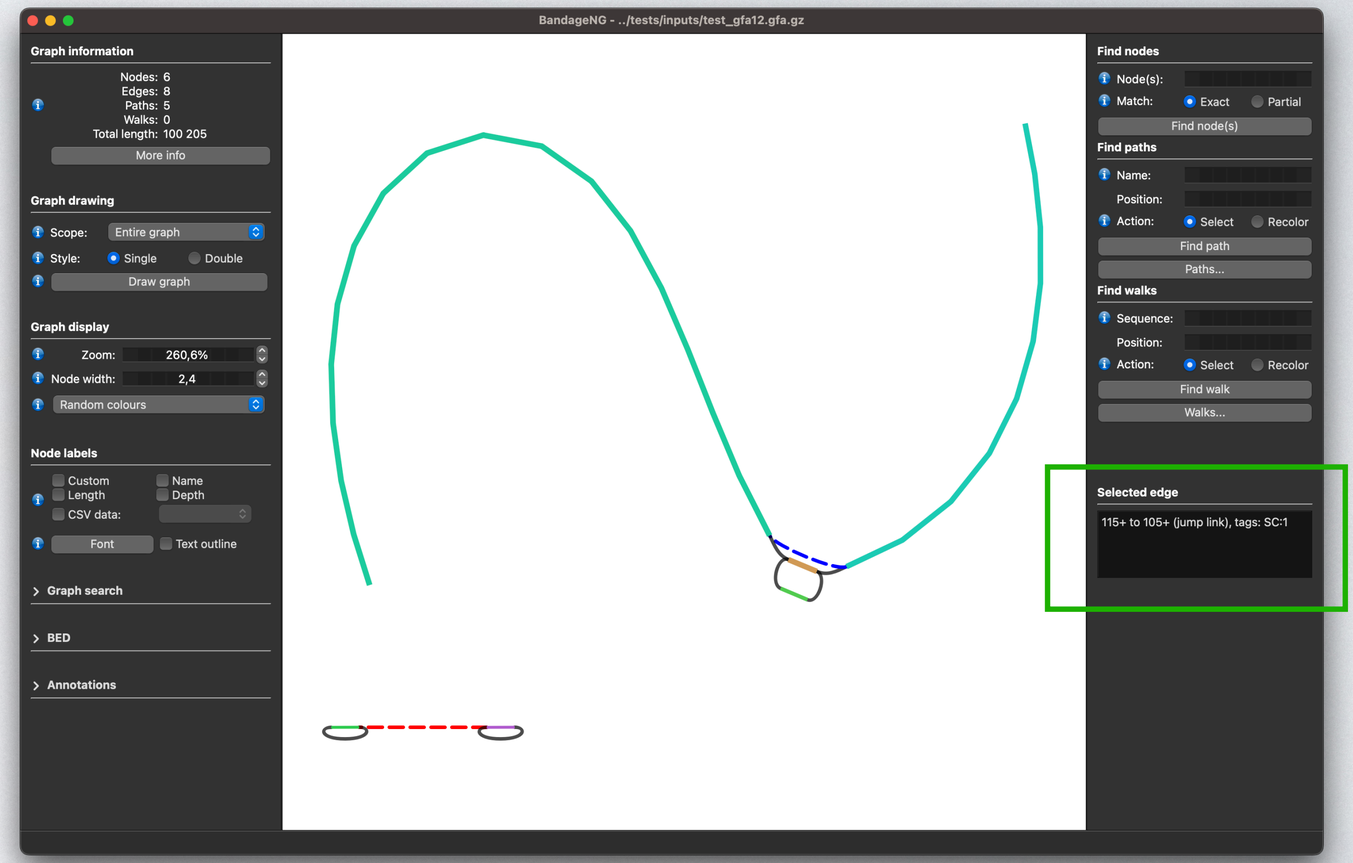
- Start node name
- End node name
- Length of overlap
- Type of edge: jump link, link
- Any tags attached to edge



Multiple edges could be selected

Edge Display

- Start node name
- End node name
- Length of overlap
- Type of edge: jump link, link
- Any tags attached to edge



Multiple edges could be selected

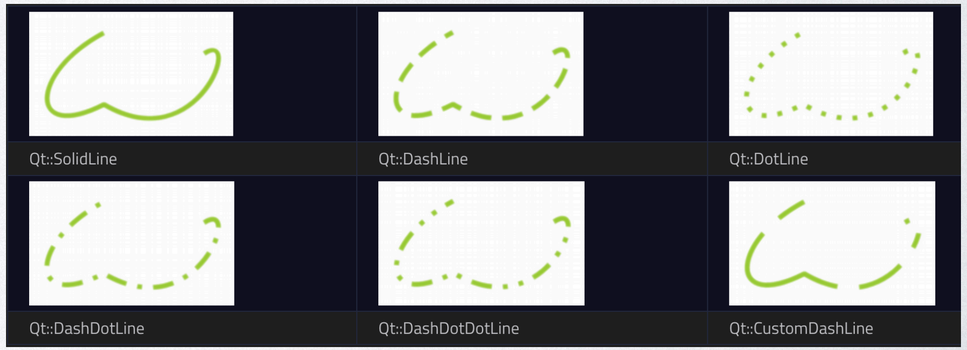
Customization

- Manual: point & click, change color, etc.
- Via Bandage-NG GFA tags
  - `LB/L2:Z`: custom label for node (reverse-complementary for `L2`)
  - `CB/C2:Z`: custom color for node / edge (reverse-complementary for `C2`)
    - Use `#RGB` notion
    - Or any of SVG standard color names
  - `WD:f`: custom width for edge
  - `PS:i`: pen style to draw edge
- Via CSV annotations (more later)

4.4 Recognized color keyword names

The following is the list of recognized color keywords that can be used as a keyword value for data type <color>:

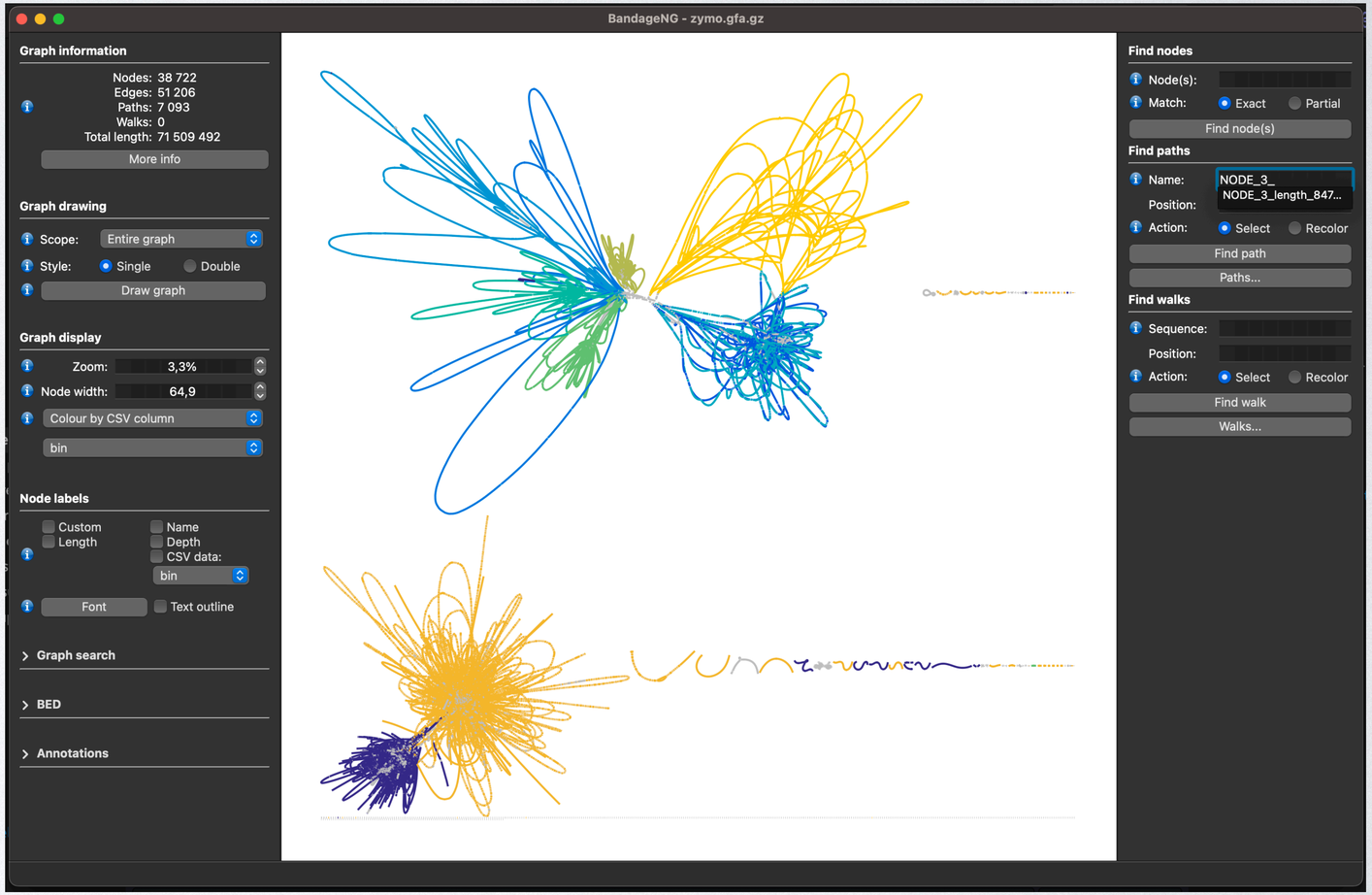
aliceblue	rgb(240, 248, 255)	lightpink	rgb(255, 182, 193)
antiquewhite	rgb(250, 235, 215)	lightsalmon	rgb(255, 160, 122)
aqua	rgb(0, 255, 255)	lightseagreen	rgb(32, 178, 170)
aquamarine	rgb(127, 255, 212)	lightskyblue	rgb(135, 206, 250)
azure	rgb(240, 255, 255)	lightslategray	rgb(119, 136, 153)
beige	rgb(245, 245, 220)	lightslategrey	rgb(119, 136, 153)
bisque	rgb(255, 228, 196)	lightsteelblue	rgb(176, 196, 222)
black	rgb(0, 0, 0)	lightyellow	rgb(255, 255, 224)
blanchedalmond	rgb(255, 235, 205)	lime	rgb(0, 255, 0)



More info on tags: https://github.com/asl/BandageNG/wiki/Custom-GFA-tags

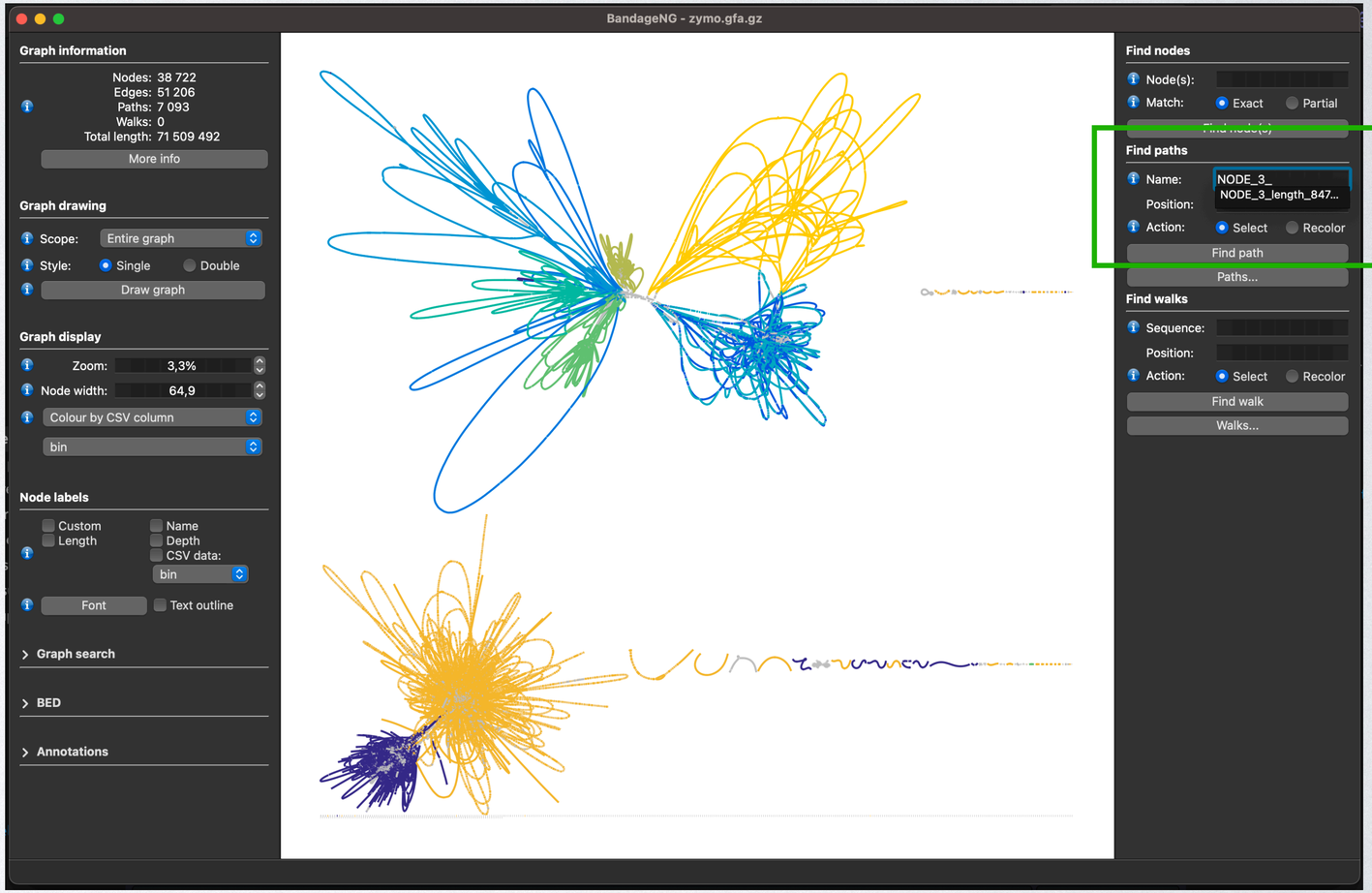
Path / Walk Lookup

- Find all paths by name
- Match by unique prefix
- Zoom in to show all nodes of a path
- Two possible actions:
  - Select
  - Recolor all nodes of a path



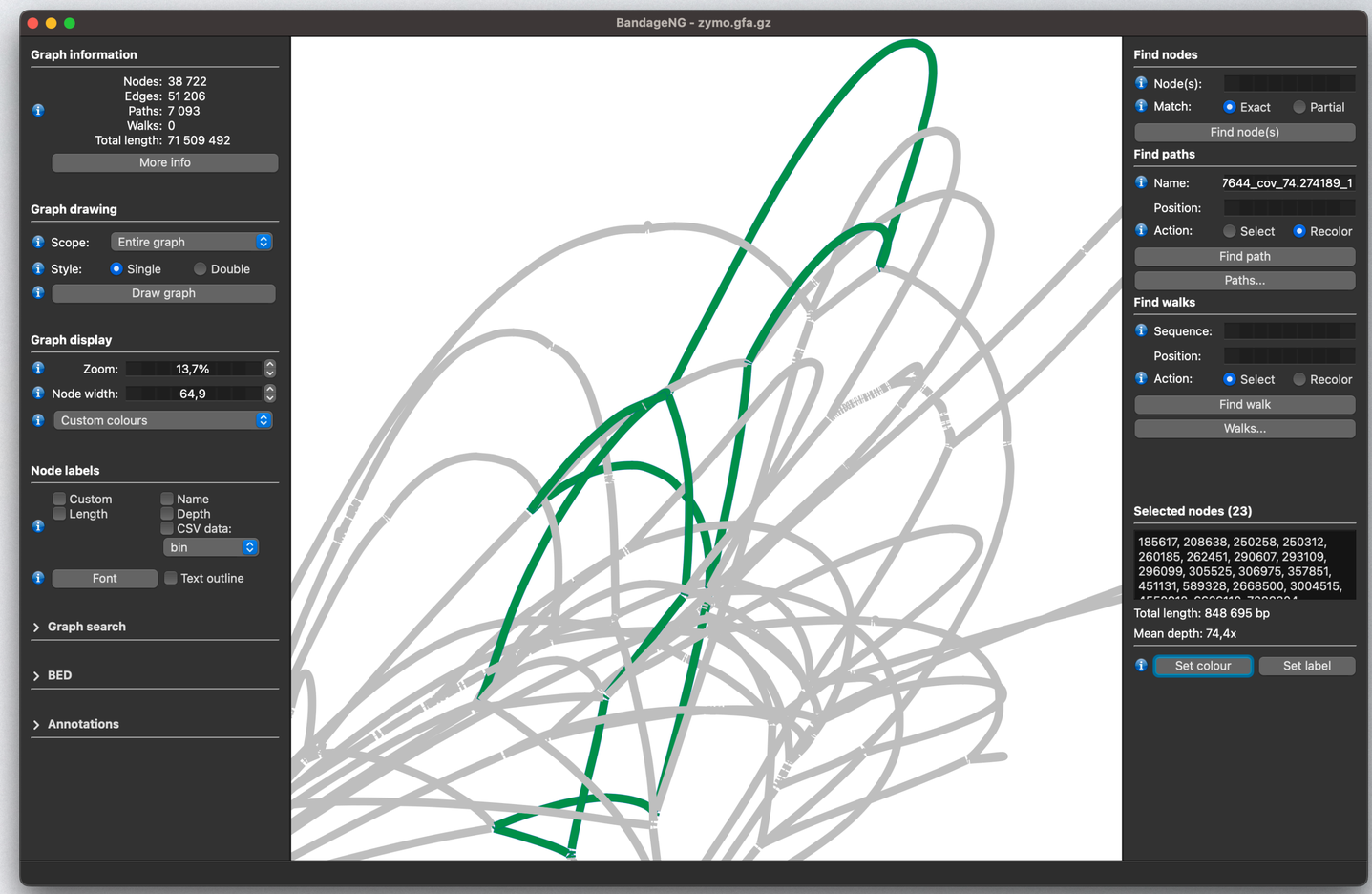
Path / Walk Lookup

- Find all paths by name
- Match by unique prefix
- Zoom in to show all nodes of a path
- Two possible actions:
  - Select
  - Recolor all nodes of a path



Path / Walk Lookup

- Find all paths by name
- Match by unique prefix
- Zoom in to show all nodes of a path
- Two possible actions:
  - Select
  - Recolor all nodes of a path

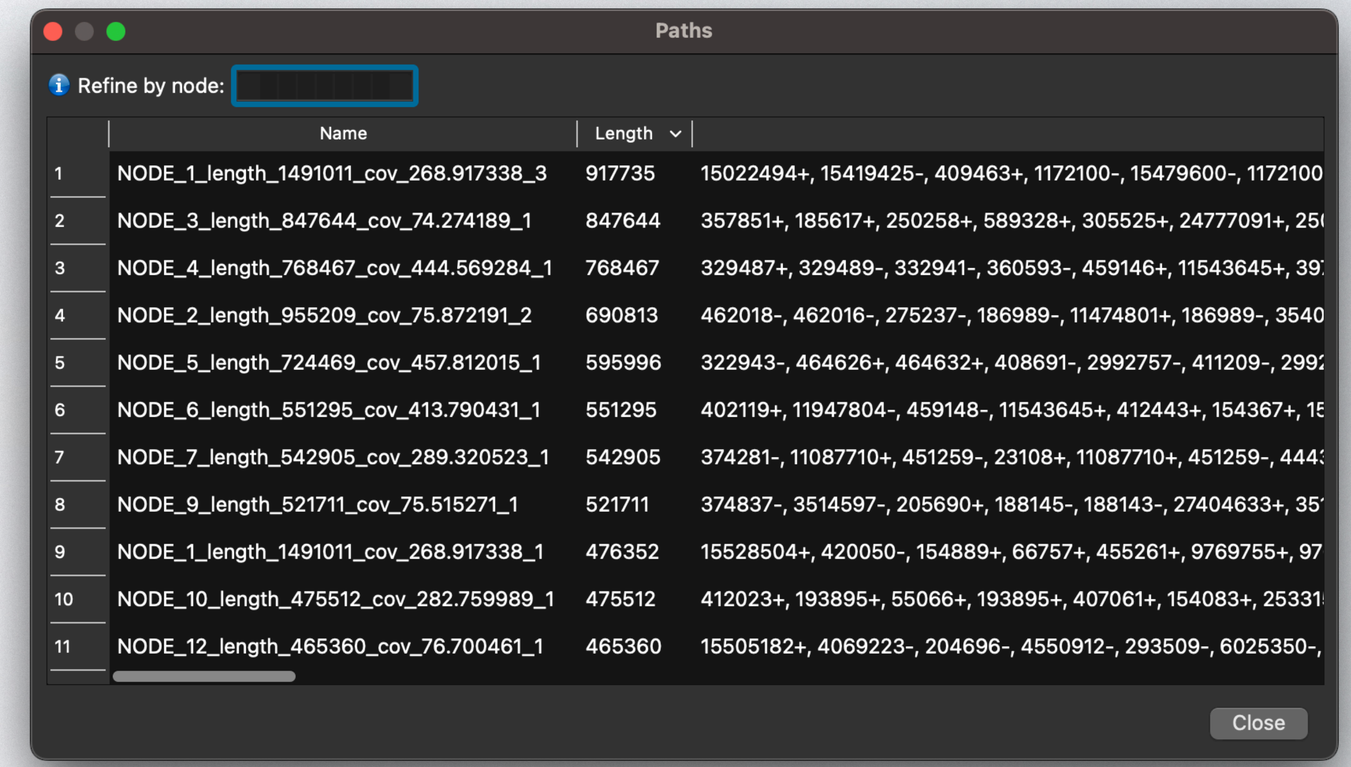


List of all Paths

- List of all paths
- Path lengths
- Nodes of the path
- Refine by node(s):
  - Show all paths containing nodes
  - For single node show its position on the containing path

List of all Paths

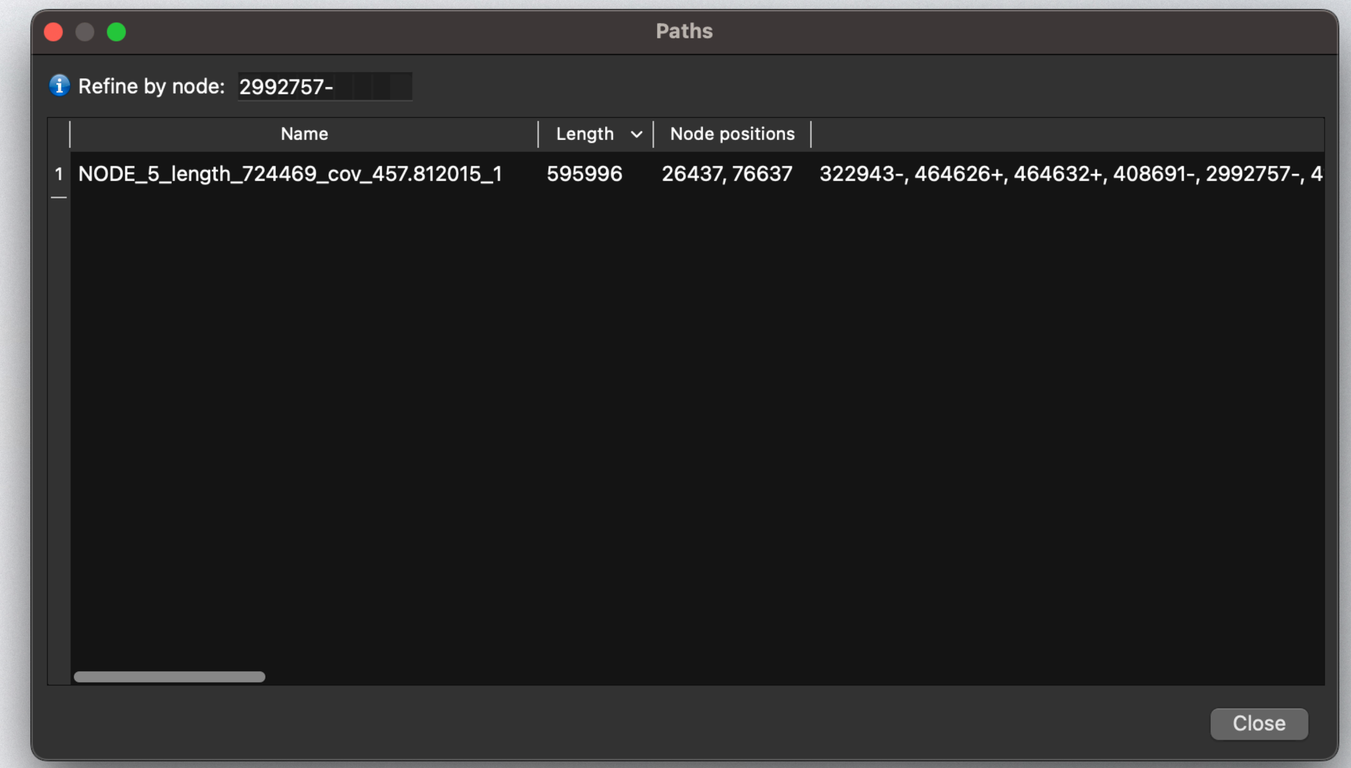
- List of all paths
- Path lengths
- Nodes of the path
- Refine by node(s):
  - Show all paths containing nodes
  - For single node show its position on the containing path



	Name	Length	
1	NODE_1_length_1491011_cov_268.917338_3	917735	15022494+, 15419425-, 409463+, 1172100-, 15479600-, 1172100
2	NODE_3_length_847644_cov_74.274189_1	847644	357851+, 185617+, 250258+, 589328+, 305525+, 24777091+, 25
3	NODE_4_length_768467_cov_444.569284_1	768467	329487+, 329489-, 332941-, 360593-, 459146+, 11543645+, 39
4	NODE_2_length_955209_cov_75.872191_2	690813	462018-, 462016-, 275237-, 186989-, 11474801+, 186989-, 3540
5	NODE_5_length_724469_cov_457.812015_1	595996	322943-, 464626+, 464632+, 408691-, 2992757-, 411209-, 299
6	NODE_6_length_551295_cov_413.790431_1	551295	402119+, 11947804-, 459148-, 11543645+, 412443+, 154367+, 15
7	NODE_7_length_542905_cov_289.320523_1	542905	374281-, 11087710+, 451259-, 23108+, 11087710+, 451259-, 444
8	NODE_9_length_521711_cov_75.515271_1	521711	374837-, 3514597-, 205690+, 188145-, 188143-, 27404633+, 35
9	NODE_1_length_1491011_cov_268.917338_1	476352	15528504+, 420050-, 154889+, 66757+, 455261+, 9769755+, 97
10	NODE_10_length_475512_cov_282.759989_1	475512	412023+, 193895+, 55066+, 193895+, 407061+, 154083+, 25331
11	NODE_12_length_465360_cov_76.700461_1	465360	15505182+, 4069223-, 204696-, 4550912-, 293509-, 6025350-,

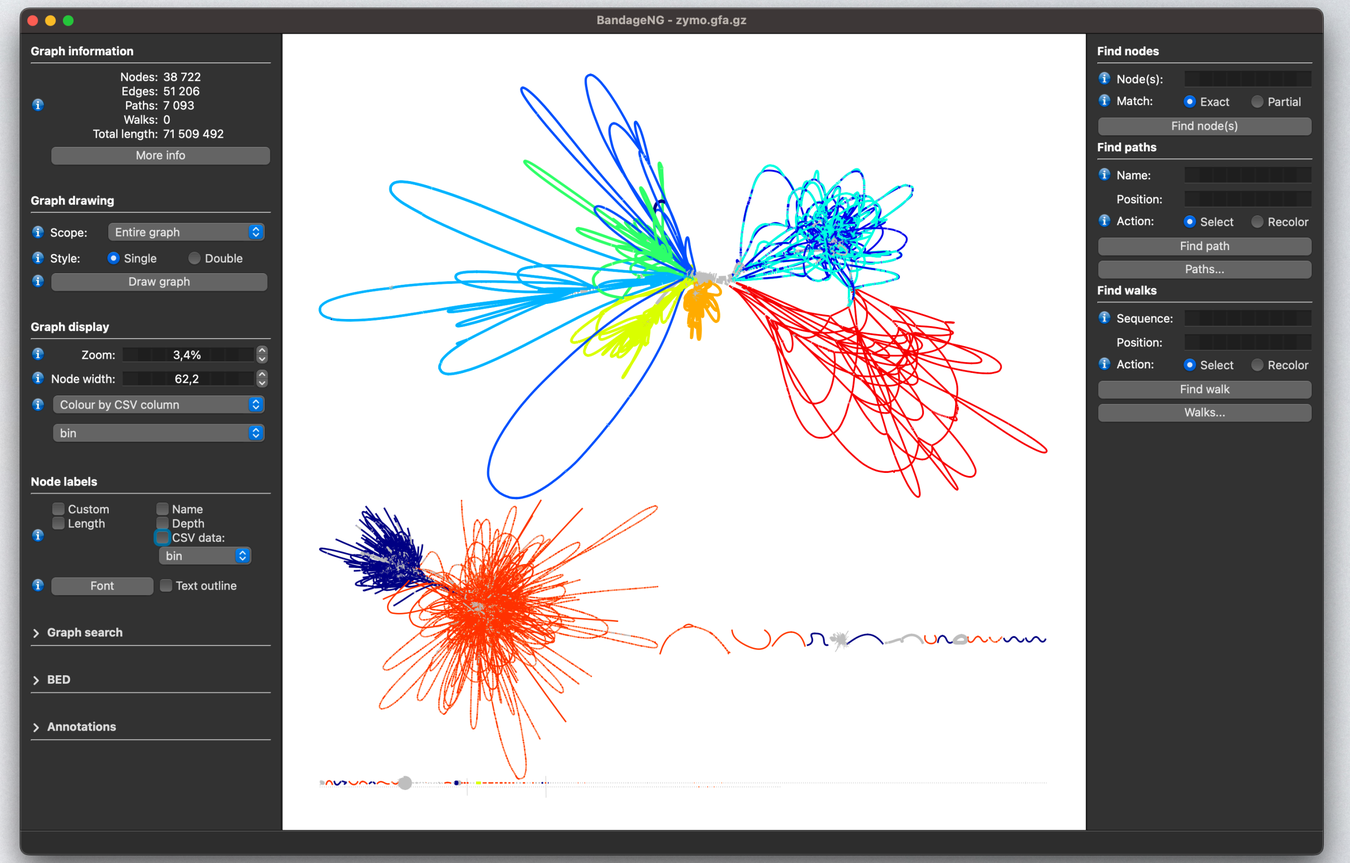
List of all Paths

- List of all paths
- Path lengths
- Nodes of the path
- Refine by node(s):
  - Show all paths containing nodes
  - For single node show its position on the containing path



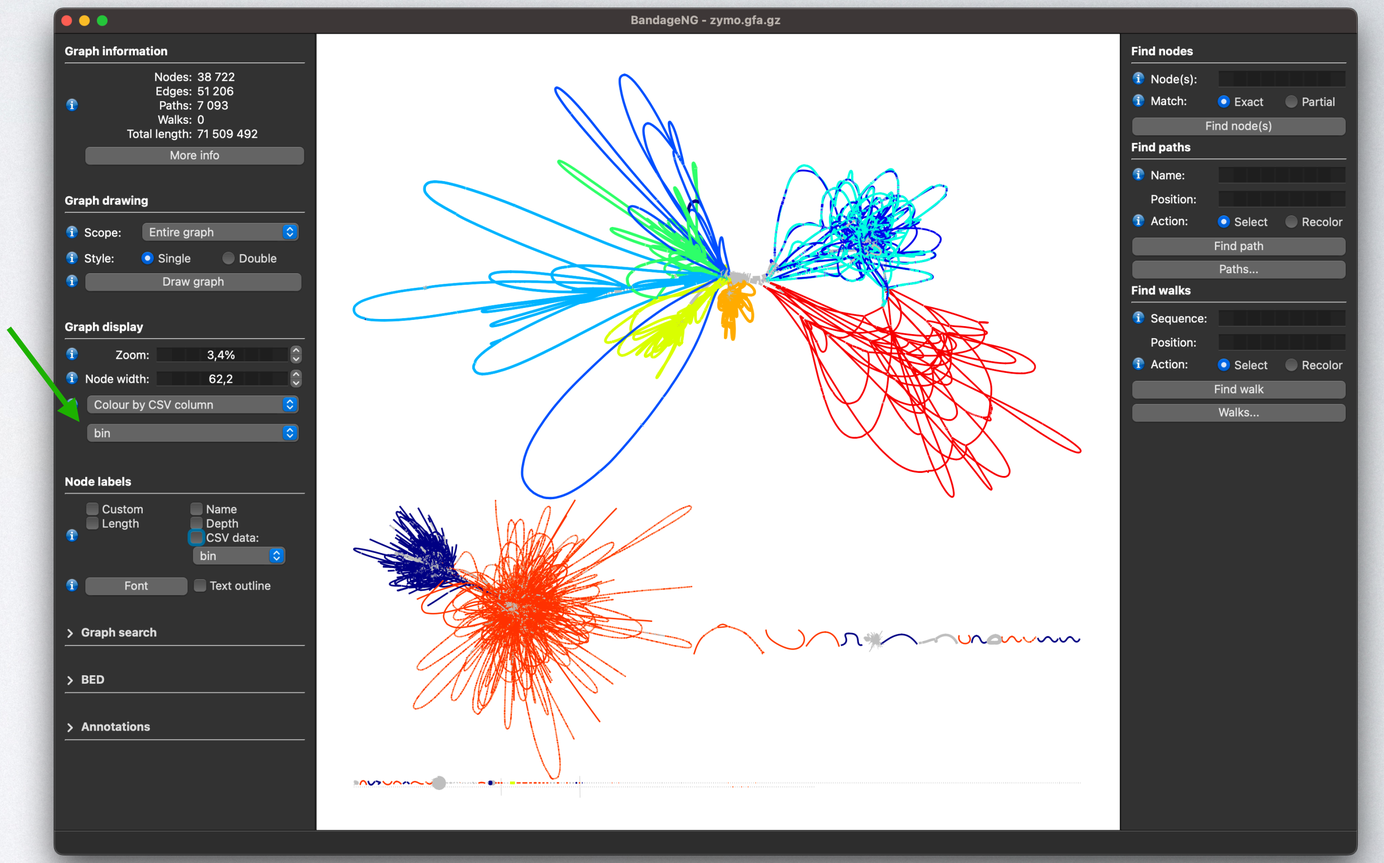
Augmentation

- Can load additional annotation for each node given from CSV / TSV file
- Currently only colors and additional labels are there
  - Colors are loaded automatically from columns named `color` or `colour`
  - Or can chose column name for colors instead
- Can use path names instead of nodes
  - Compare e.g. MAG binnings



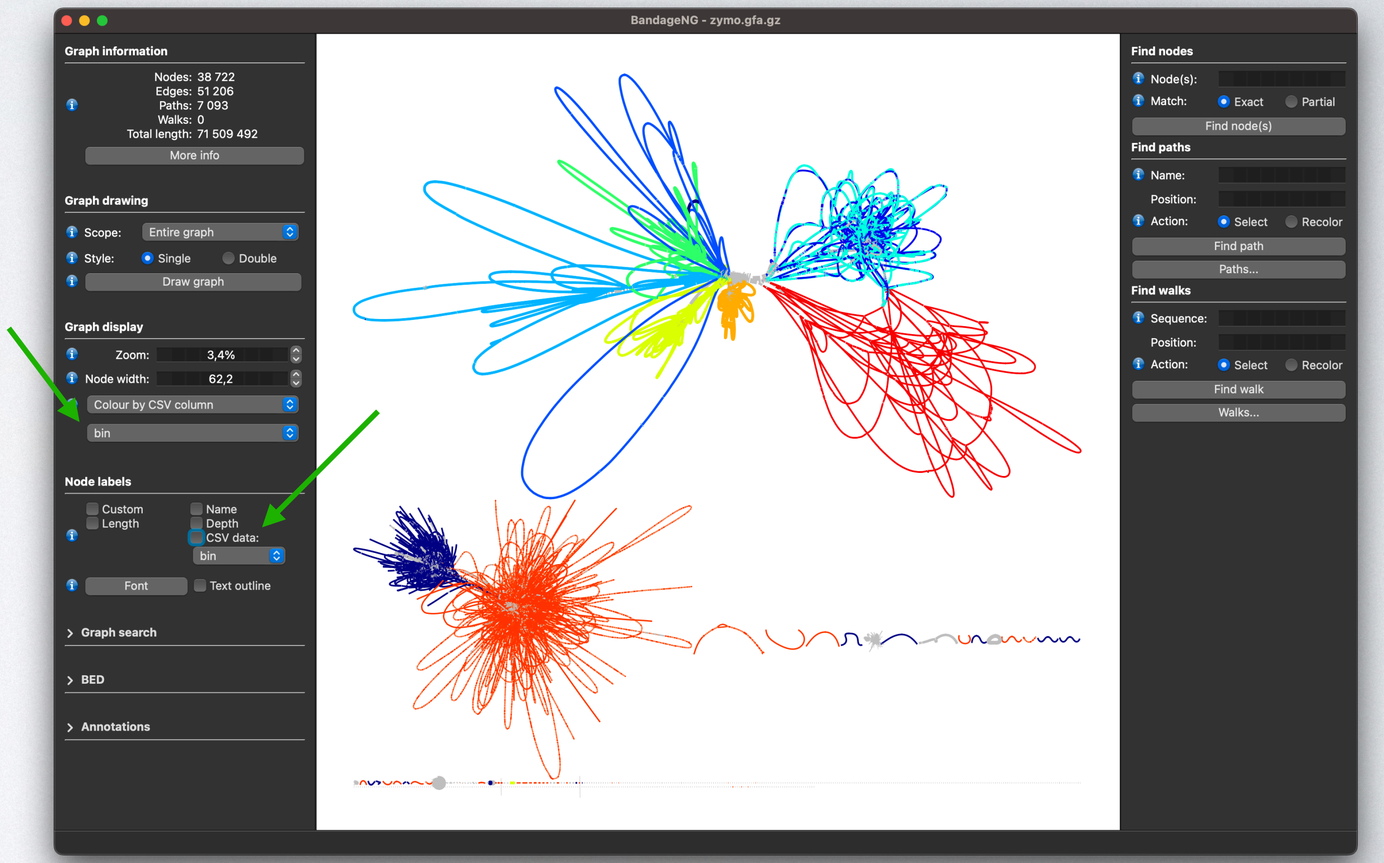
Augmentation

- Can load additional annotation for each node given from CSV / TSV file
- Currently only colors and additional labels are there
  - Colors are loaded automatically from columns named `color` or `colour`
  - Or can chose column name for colors instead
- Can use path names instead of nodes
  - Compare e.g. MAG binnings



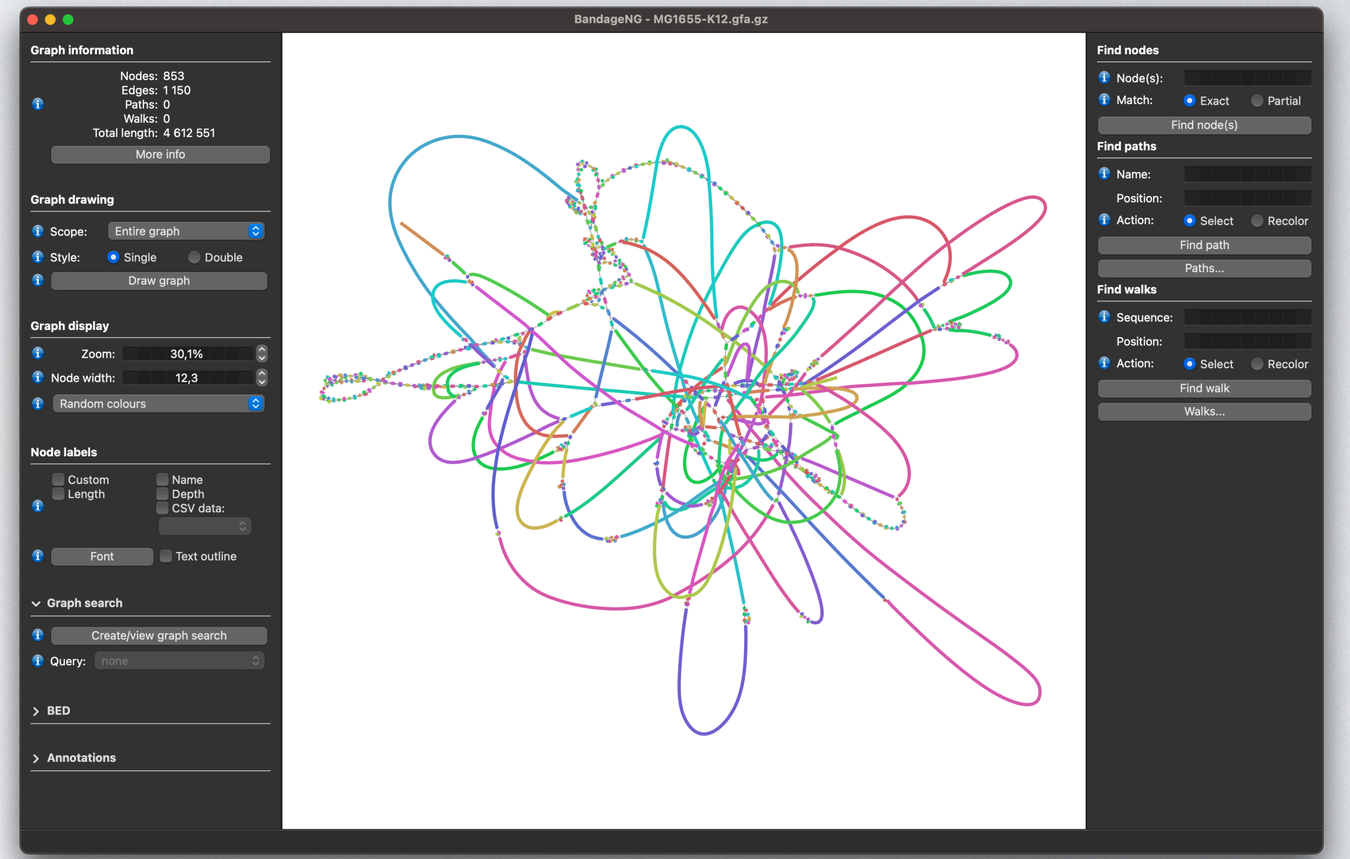
Augmentation

- Can load additional annotation for each node given from CSV / TSV file
- Currently only colors and additional labels are there
  - Colors are loaded automatically from columns named `color` or `colour`
  - Or can chose column name for colors instead
- Can use path names instead of nodes
  - Compare e.g. MAG binnings



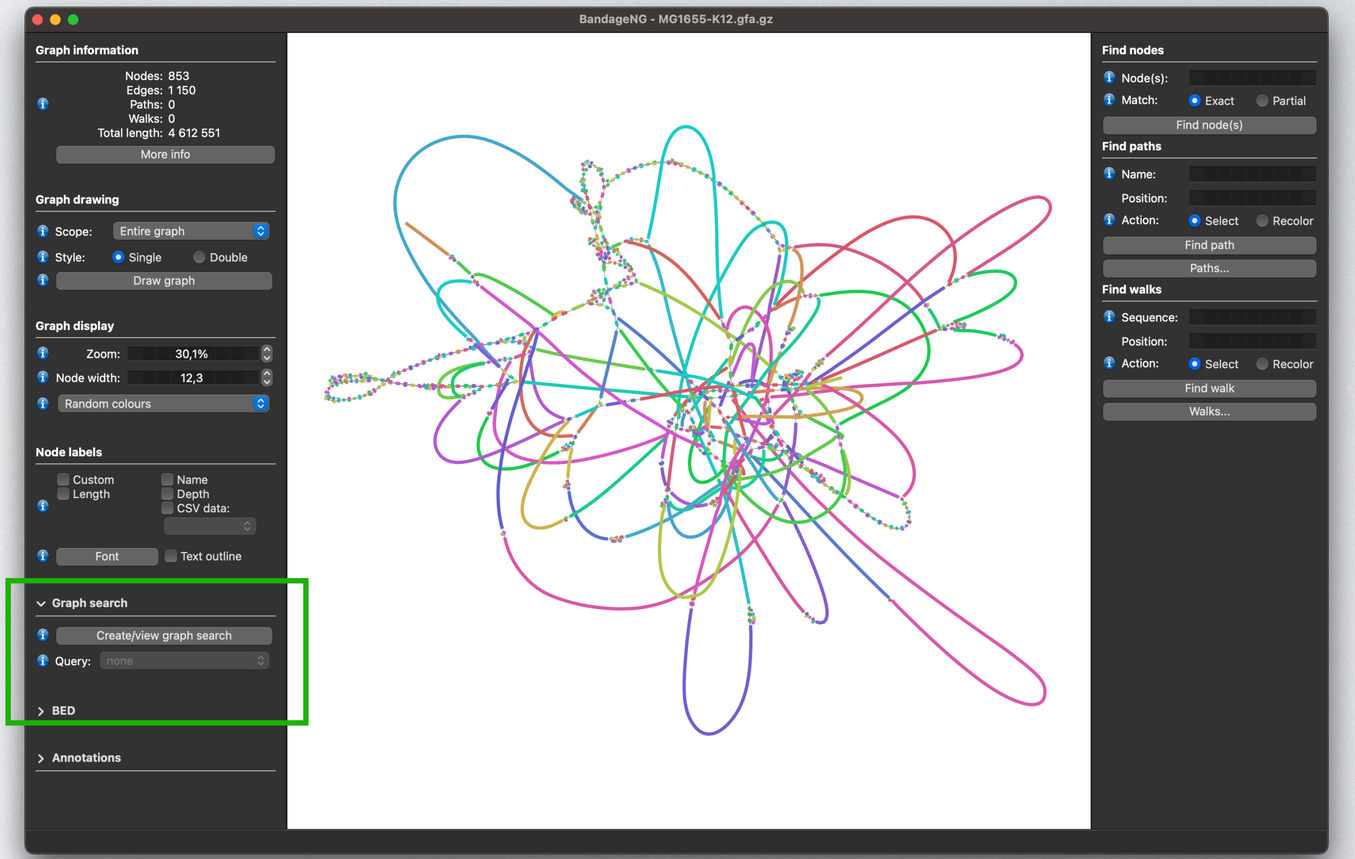
Graph Search: Common

- See paths of queries on assembly graph
  - Graph nodes: DB
- Poor-man graph alignment:
  - Perform alignment to graph node sequences,
  - Try to reconstruct paths from individual node alignments
  - Prefer proper tools if possible: GraphAligner, SPAligner, PathRacer, etc.
- Can use graph paths in addition to node sequences to increase specificity
  - Usually: align to scaffolds as well



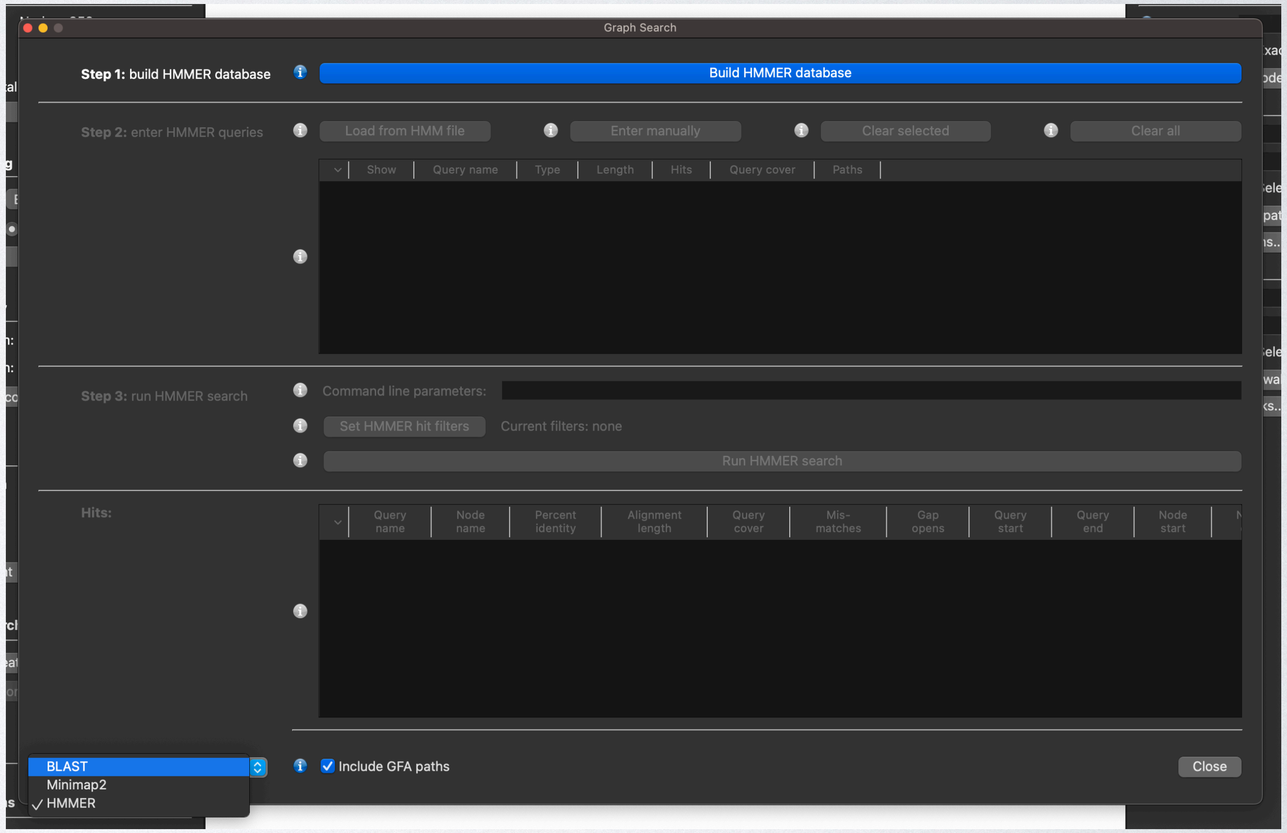
Graph Search: Common

- See paths of queries on assembly graph
  - Graph nodes: DB
- Poor-man graph alignment:
  - Perform alignment to graph node sequences,
  - Try to reconstruct paths from individual node alignments
  - Prefer proper tools if possible: GraphAligner, SPAligner, PathRacer, etc.
- Can use graph paths in addition to node sequences to increase specificity
  - Usually: align to scaffolds as well



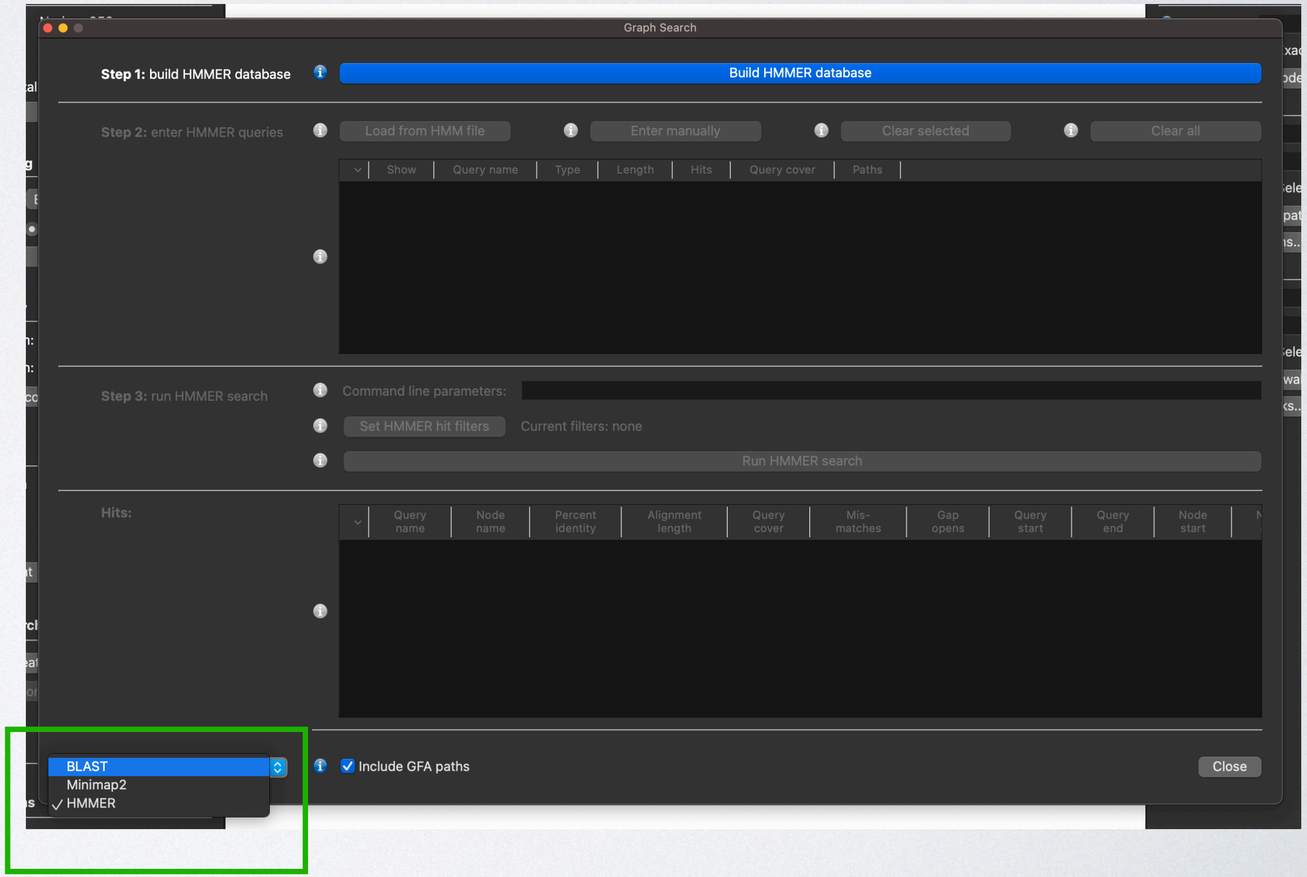
BLAST / Minimap2 / HMMER

- Different aligners are supported
  - BLAST mode supports both nucleotide and AA queries
  - HMMER mode supports both nucleotide and AA HMMs
  - Bandage-NG does necessary 6-frame translation and coordinate conversion on-fly



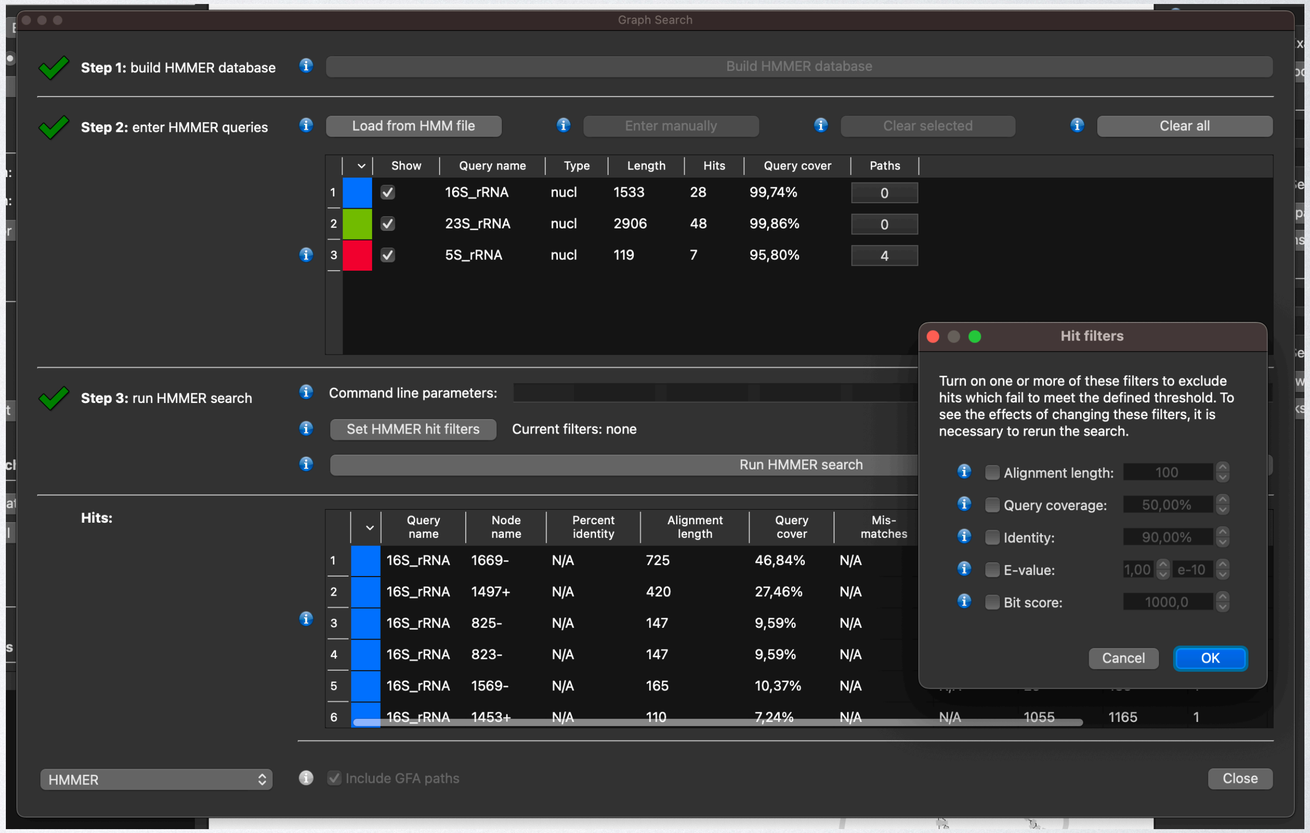
BLAST / Minimap2 / HMMER

- Different aligners are supported
  - BLAST mode supports both nucleotide and AA queries
  - HMMER mode supports both nucleotide and AA HMMs
  - Bandage-NG does necessary 6-frame translation and coordinate conversion on-fly



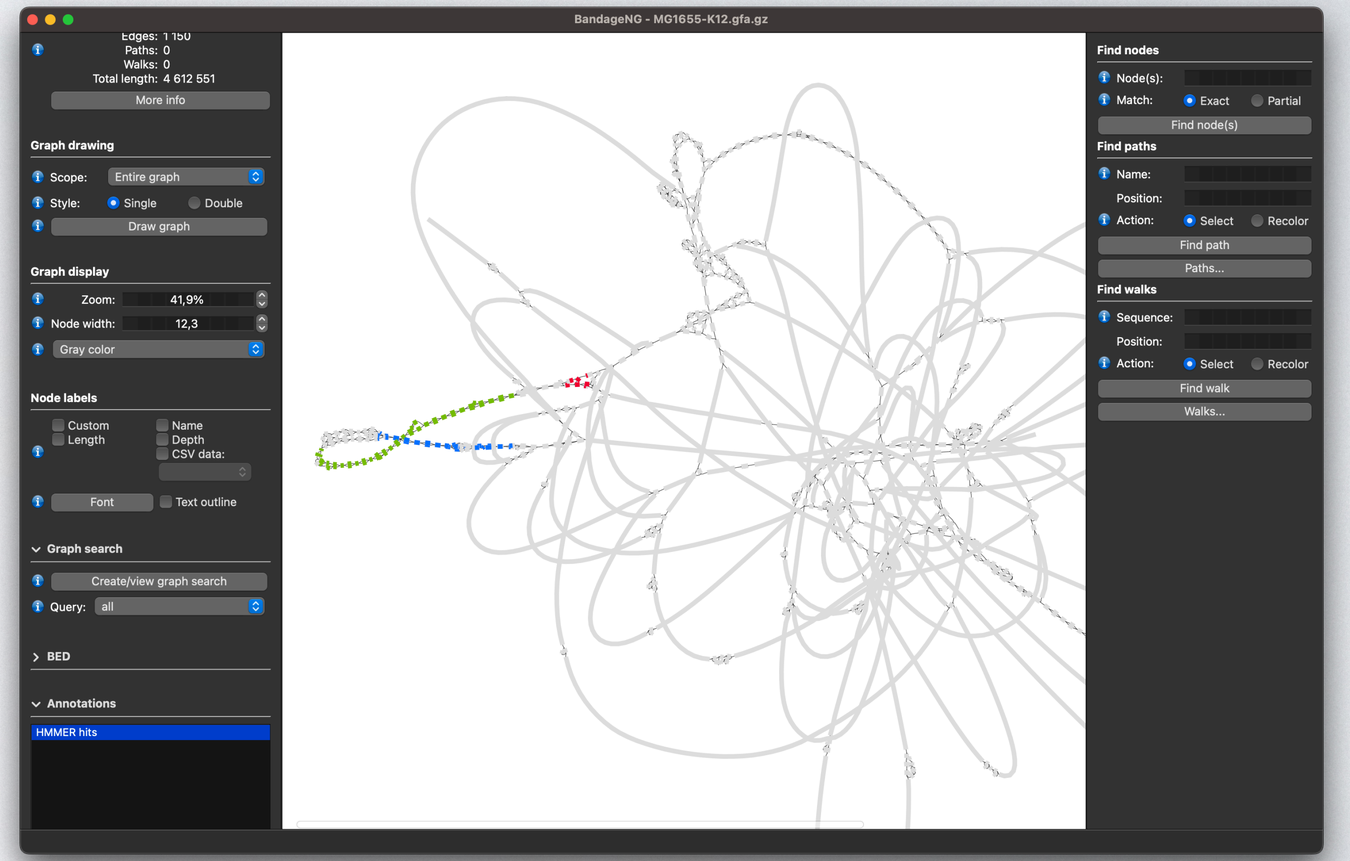
Graph Search Workflow

- Build database
- Load queries
- Setup hit filters / add cmdline parameters (e.g. # of threads)
- Run search
- Check hits & paths



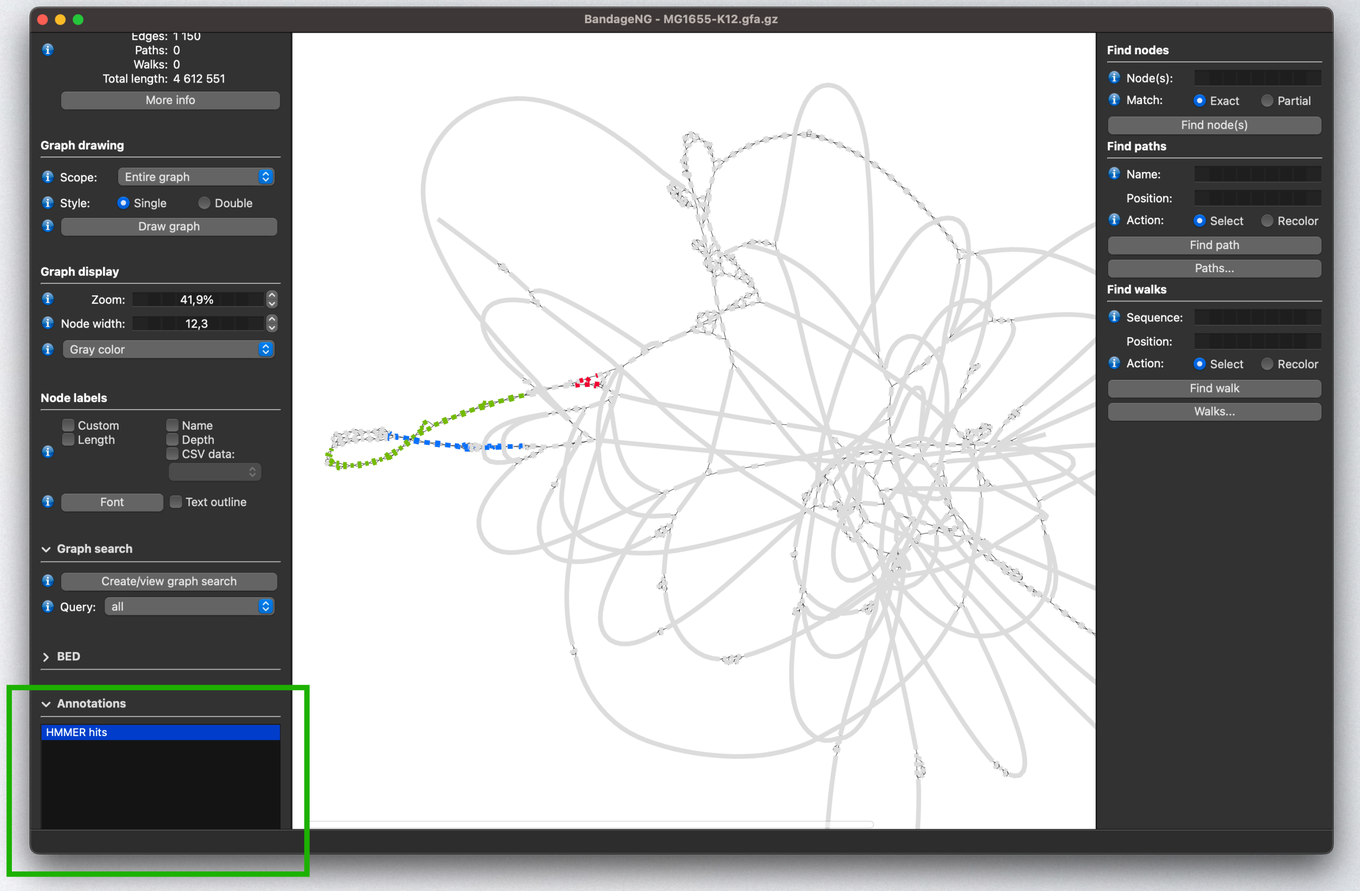
Search Annotations

- Shows query paths on assembly graph
- Can have multiple types of annotations at the same time (HMMER hits, BLAST hits, BED, …)
- Better to switch to grey colors for nodes
- Different color schemes for hits:
  - Solid: colored by query
  - Rainbow: colored by query coordinates



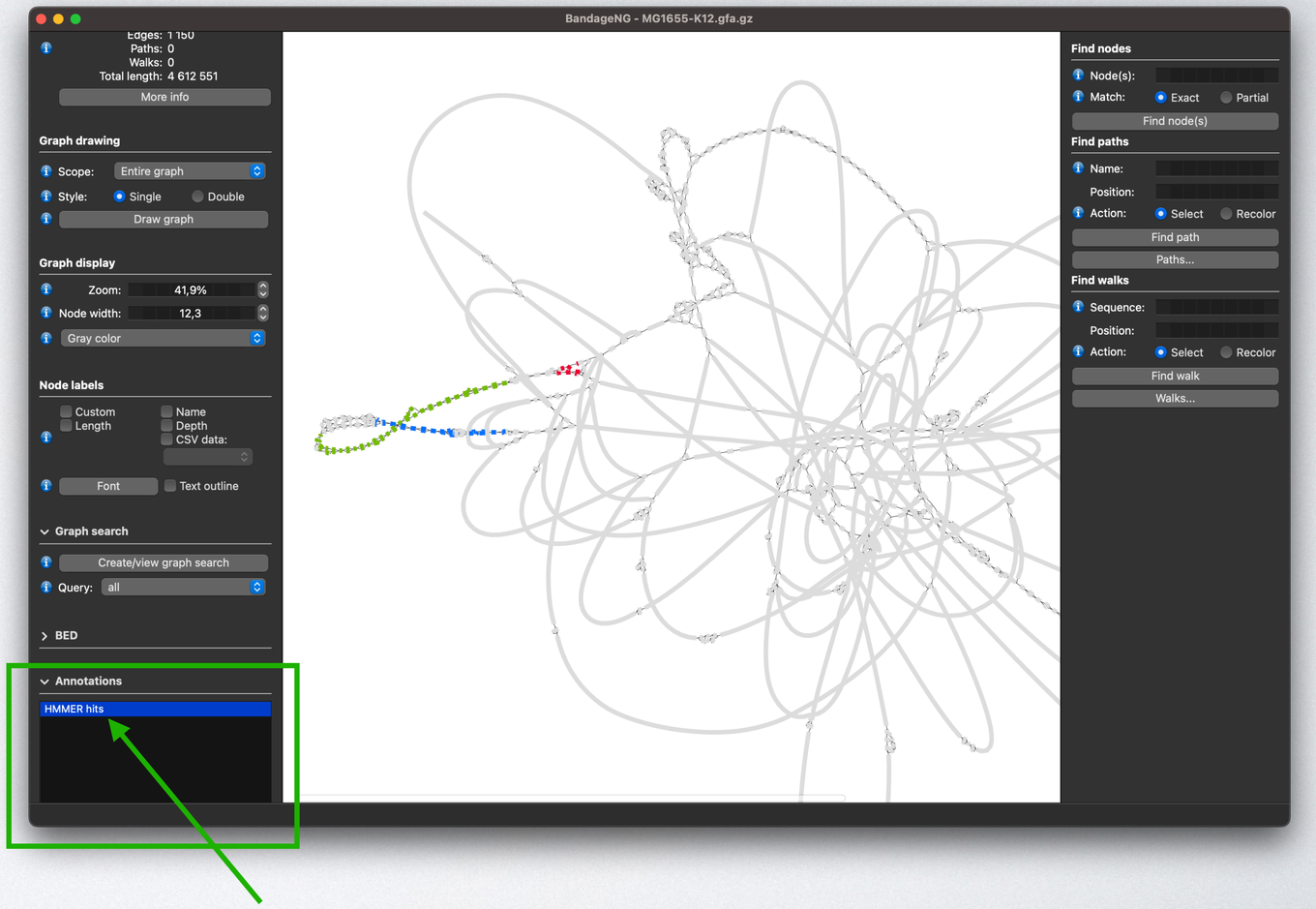
Search Annotations

- Shows query paths on assembly graph
- Can have multiple types of annotations at the same time (HMMER hits, BLAST hits, BED, …)
- Better to switch to grey colors for nodes
- Different color schemes for hits:
  - Solid: colored by query
  - Rainbow: colored by query coordinates



Search Annotations

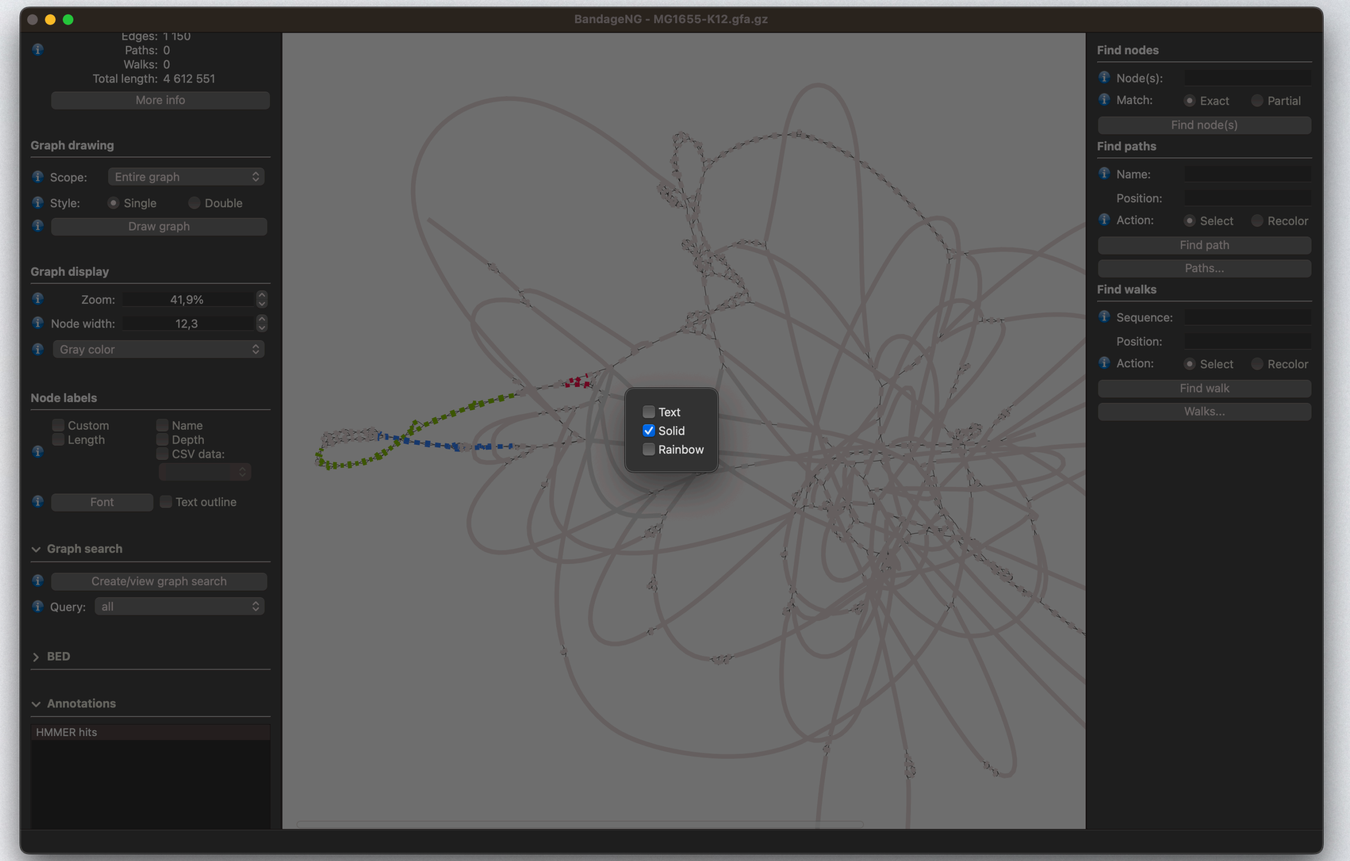
- Shows query paths on assembly graph
- Can have multiple types of annotations at the same time (HMMER hits, BLAST hits, BED, …)
- Better to switch to grey colors for nodes
- Different color schemes for hits:
  - Solid: colored by query
  - Rainbow: colored by query coordinates



Double click here

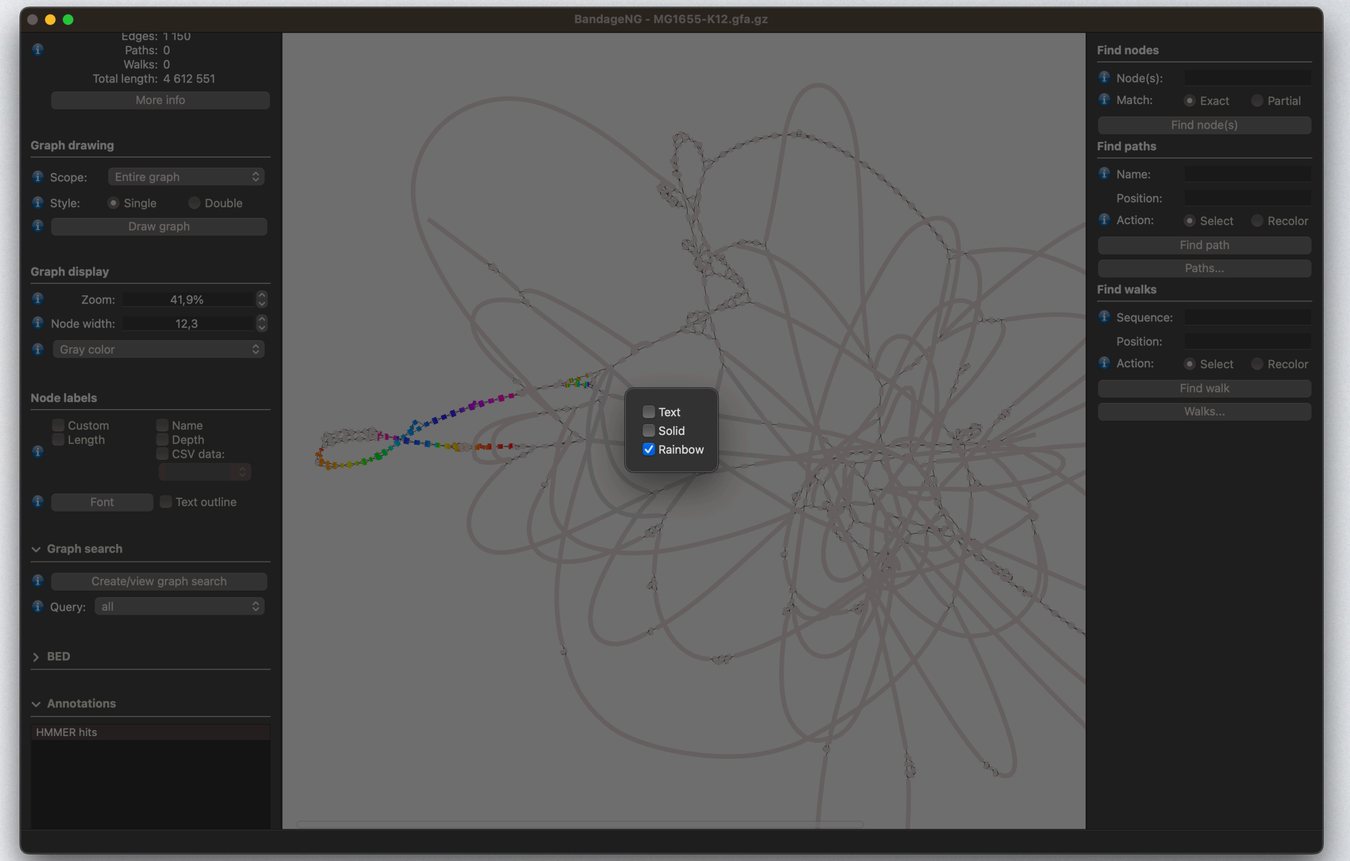
Search Annotations

- Shows query paths on assembly graph
- Can have multiple types of annotations at the same time (HMMER hits, BLAST hits, BED, …)
- Better to switch to grey colors for nodes
- Different color schemes for hits:
  - Solid: colored by query
  - Rainbow: colored by query coordinates



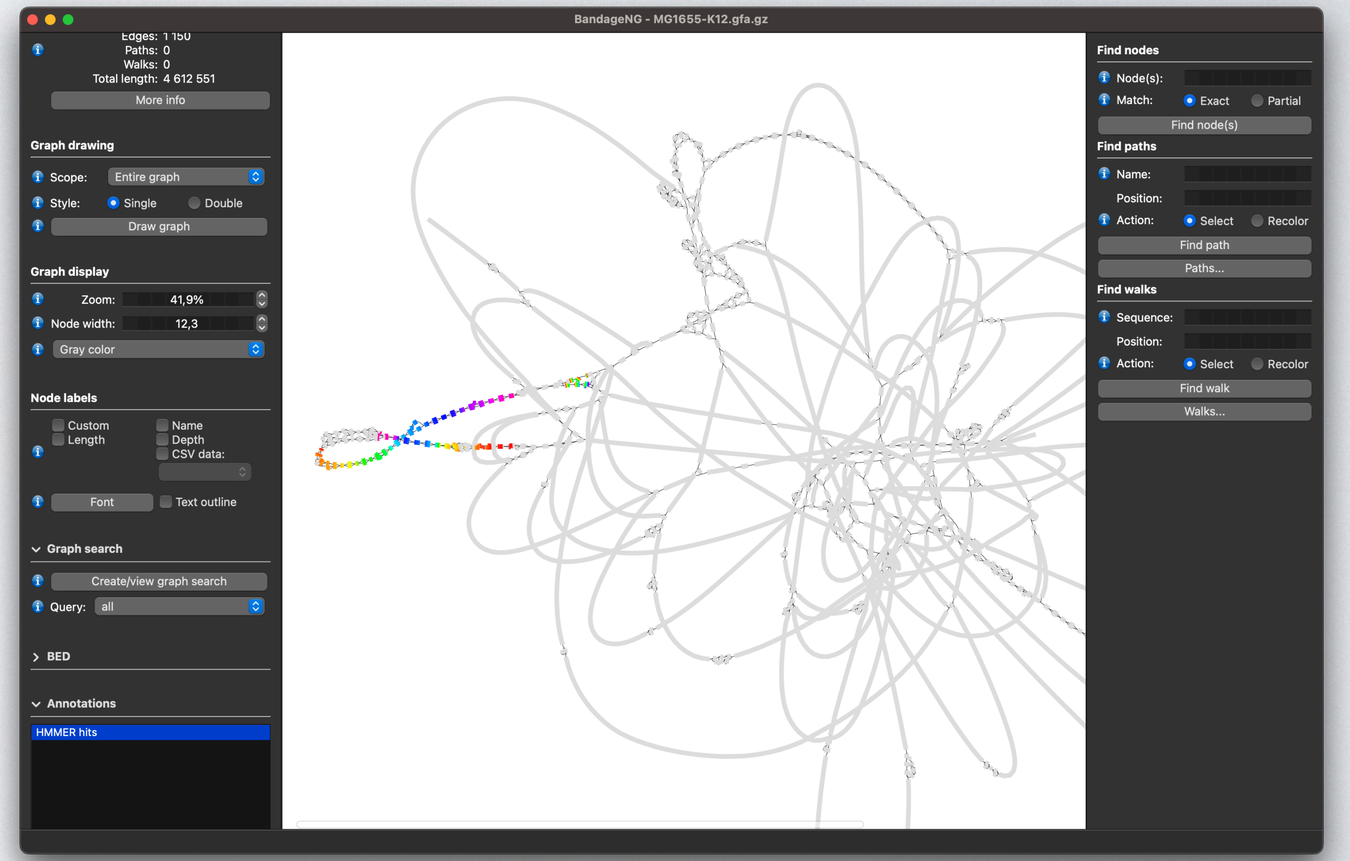
Search Annotations

- Shows query paths on assembly graph
- Can have multiple types of annotations at the same time (HMMER hits, BLAST hits, BED, …)
- Better to switch to grey colors for nodes
- Different color schemes for hits:
  - Solid: colored by query
  - Rainbow: colored by query coordinates



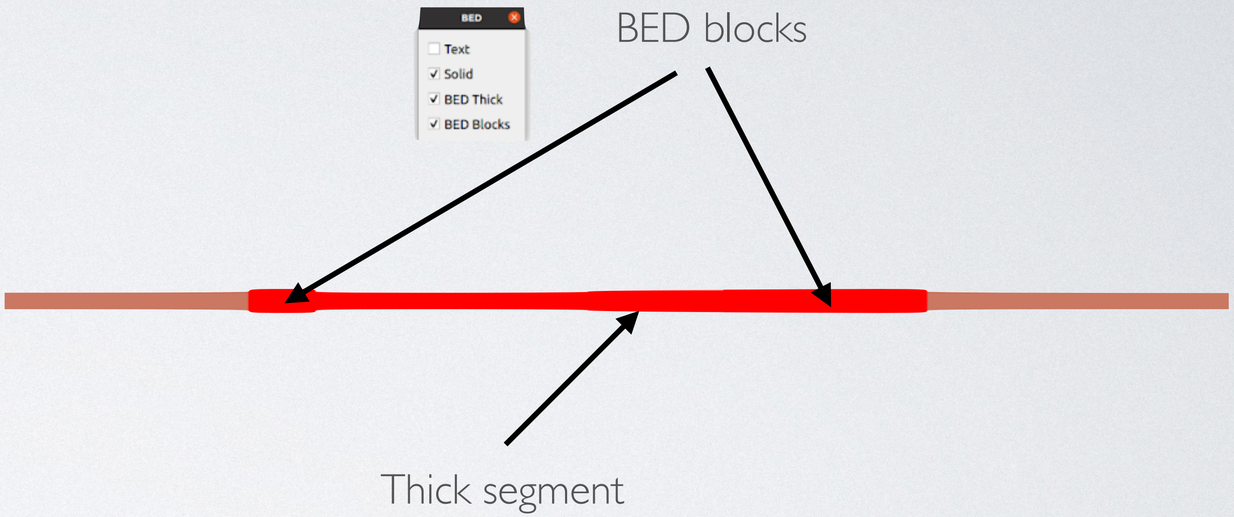
Search Annotations

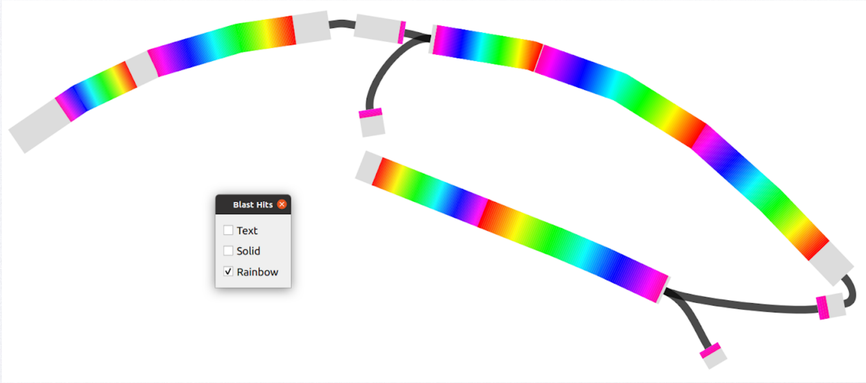
- Shows query paths on assembly graph
- Can have multiple types of annotations at the same time (HMMER hits, BLAST hits, BED, …)
- Better to switch to grey colors for nodes
- Different color schemes for hits:
  - Solid: colored by query
  - Rainbow: colored by query coordinates



BED Annotations

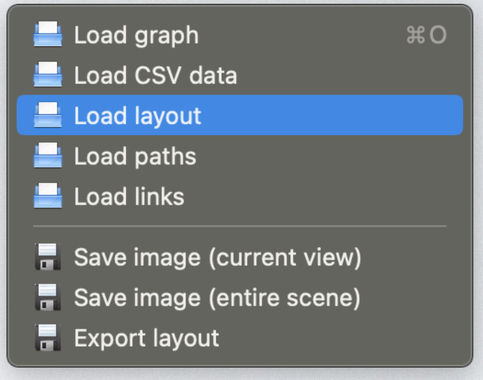
- Support all standard BED things: blocks, thick parts, etc.
- One can use existing converters for other formats: paf2bed, gff2bed, etc





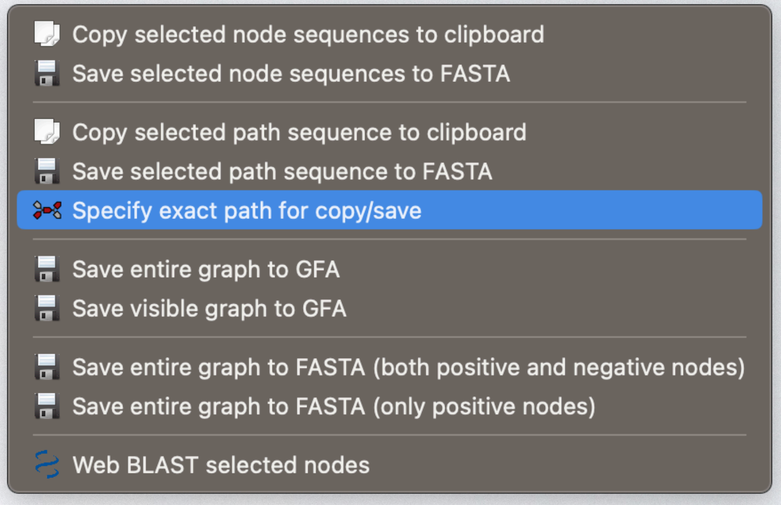
Exporting / Importing Layout

- Two formats:
  - Fine-grained Bandage format (with node segments)
  - Common format with coordinates of node segments
- Allows to save & reload graph layout for later after manual edits, etc.



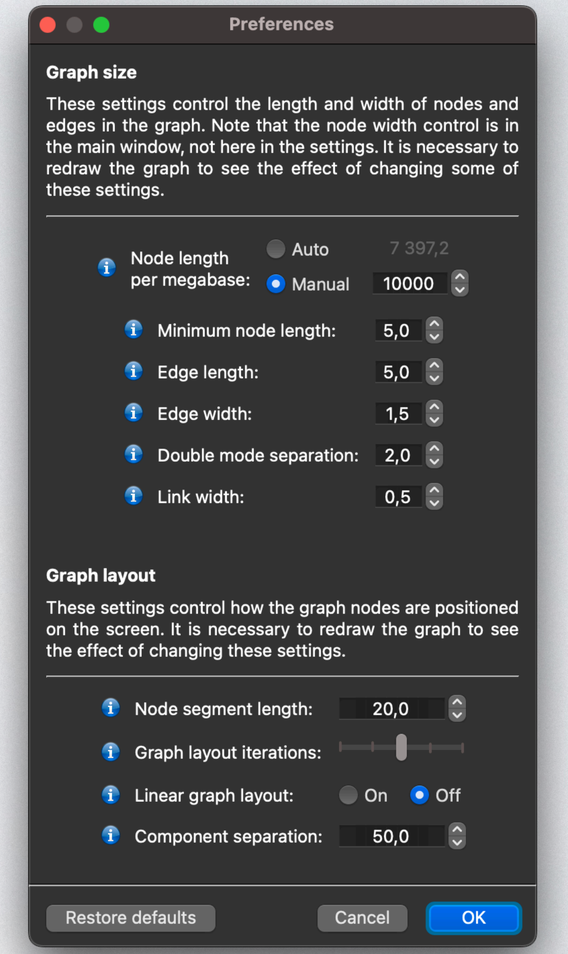
Output

- Copy / save selected node sequence
- Copy / save selected path sequence
- Save graph / subgraph to GFA
- Save graph / subgraph to FASTA
- BLAST selected nodes on NCBI server



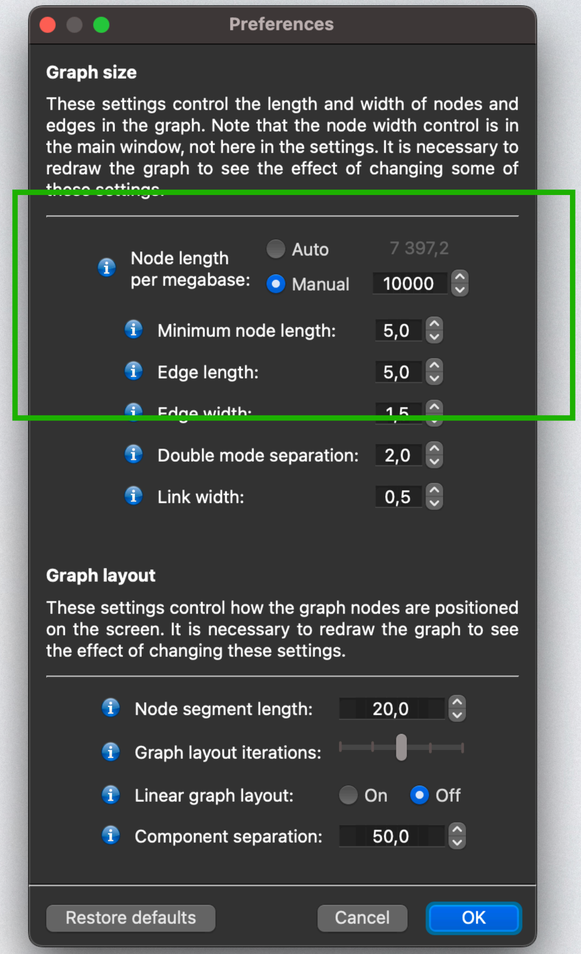
Settings: Graph Size

- Node length per megabase:
  - Larger value: longer nodes
  - Might need to tune if autodetection fails
  - Node length cannot be less than minimum



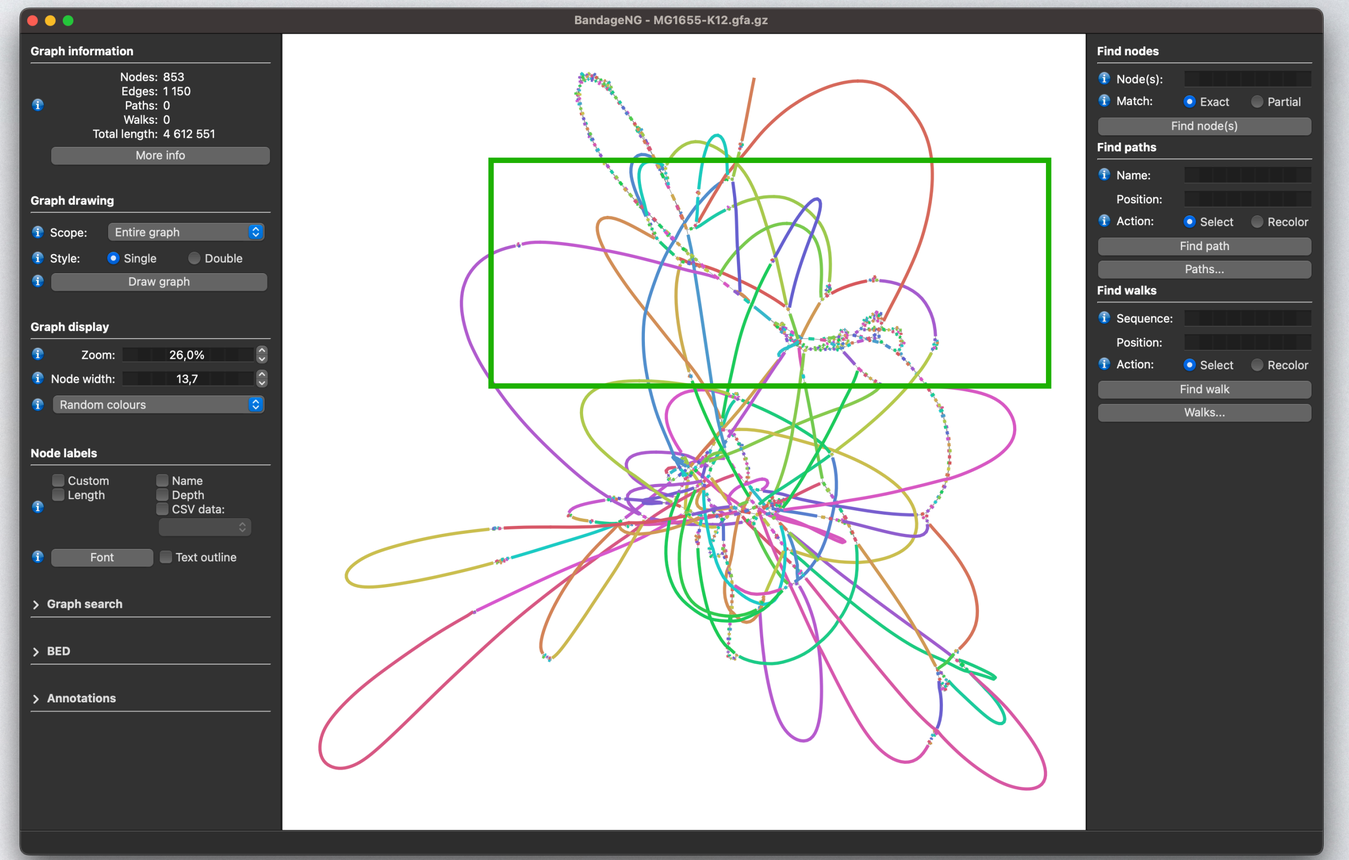
Settings: Graph Size

- Node length per megabase:
  - Larger value: longer nodes
  - Might need to tune if autodetection fails
  - Node length cannot be less than minimum



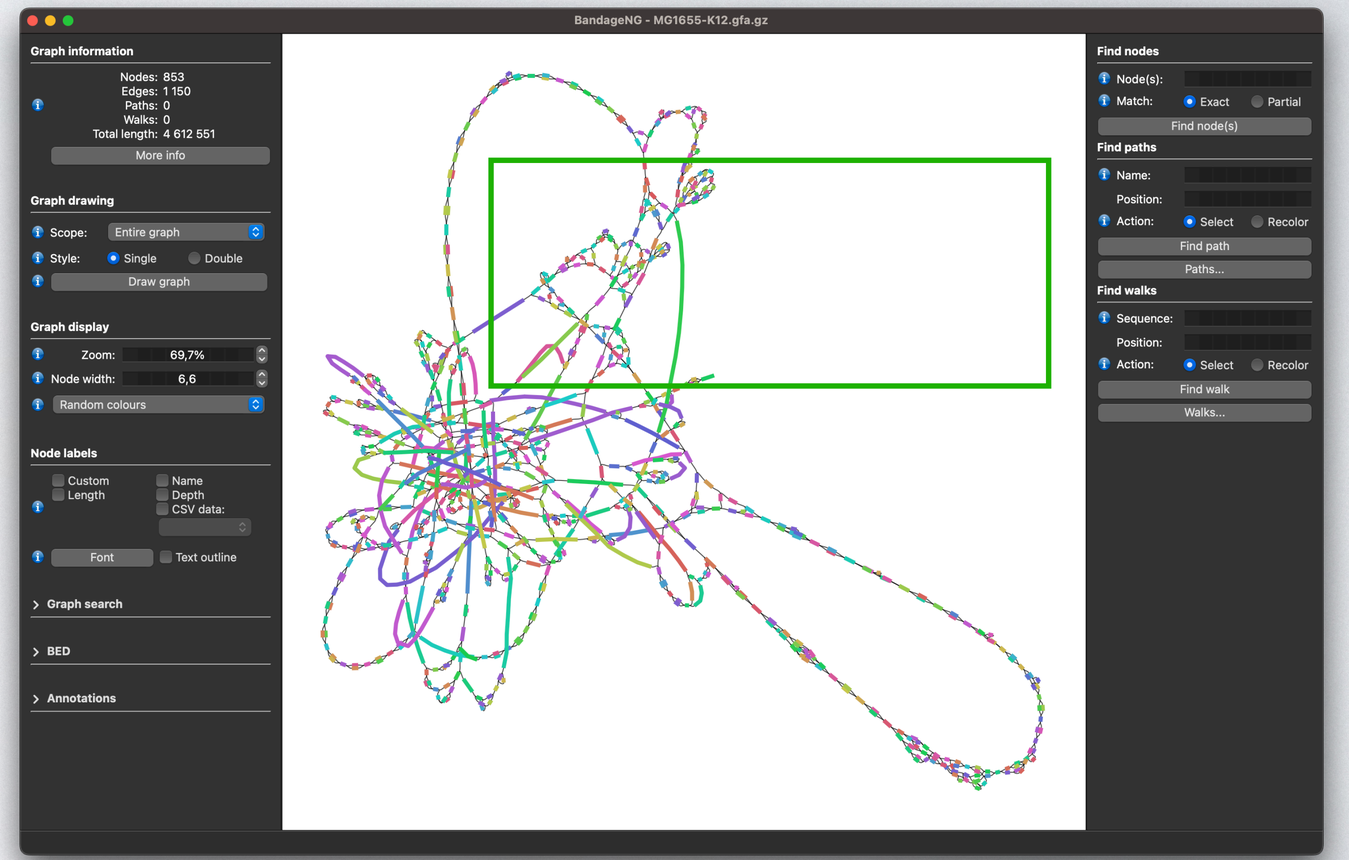
Settings: Graph Size

- Node length per megabase:
  - Larger value: longer nodes
  - Might need to tune if autodetection fails
  - Node length cannot be less than minimum



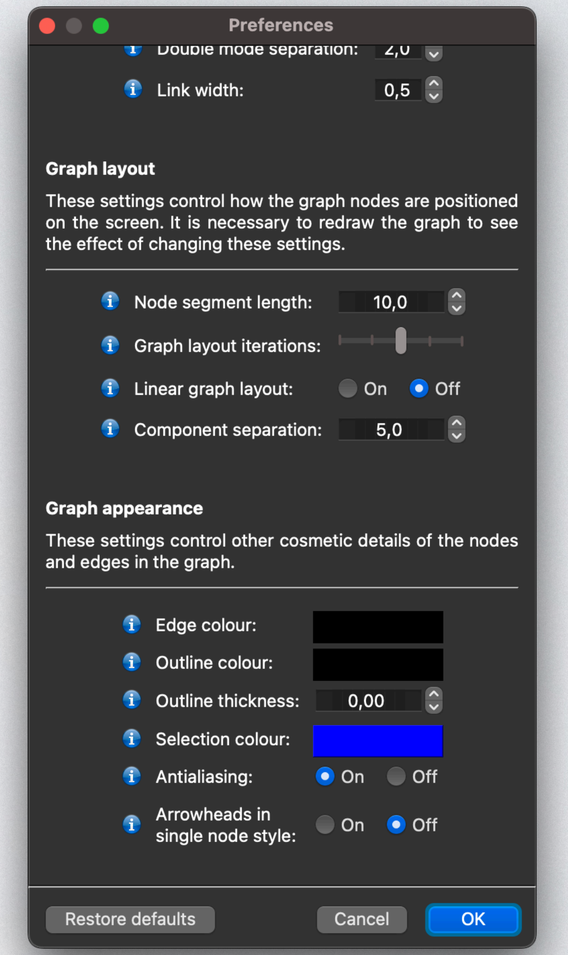
Settings: Graph Size

- Node length per megabase:
  - Larger value: longer nodes
  - Might need to tune if autodetection fails
  - Node length cannot be less than minimum



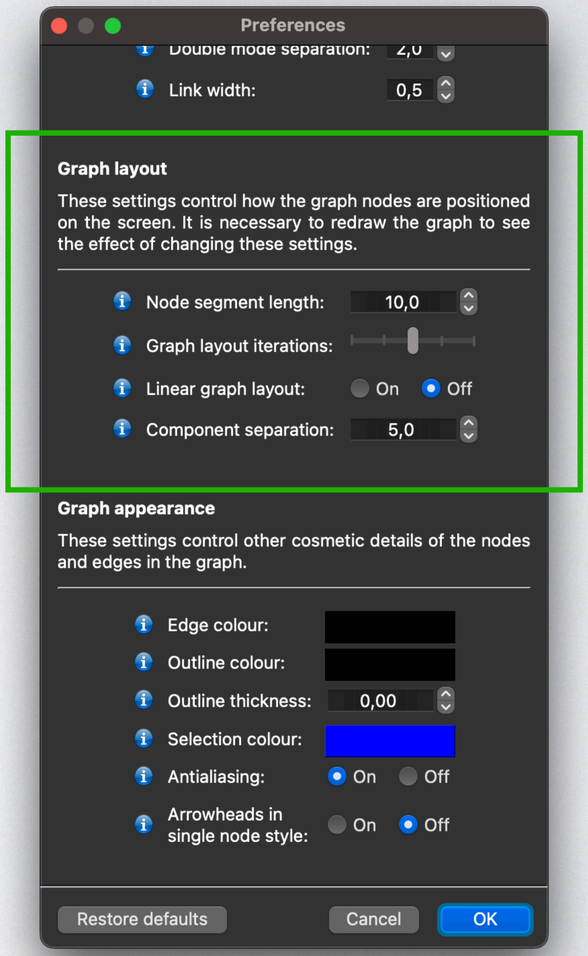
Settings: Layout

- Node segment length
  - Nodes are represented as multiple segments
  - Smaller segments: smoother nodes, but longer layout
- Graph layout iterations
  - Better layout for more iterations
- Linear graph layout
  - Mostly for pangenomes
- Component separation
  - Space between multiple connected components



Settings: Layout

- Node segment length
  - Nodes are represented as multiple segments
  - Smaller segments: smoother nodes, but longer layout
- Graph layout iterations
  - Better layout for more iterations
- Linear graph layout
  - Mostly for pangenomes
- Component separation
  - Space between multiple connected components



WIP: Spoilers

- Support for node links
  - E.g. Hi-C connections
  - More ideas are needed to make this useful
- Some UI / UX improvements, better error reporting
- Graph layout improvements?
- You ideas are always welcome!
- Better yet, patches are even more welcome ;)

Q & A?