Graph Layout by Path-Guided Stochastic Gradient Descent

Simon Heumos[1*], Andrea Guarracino[2*], and Erik Garrison[3,4]

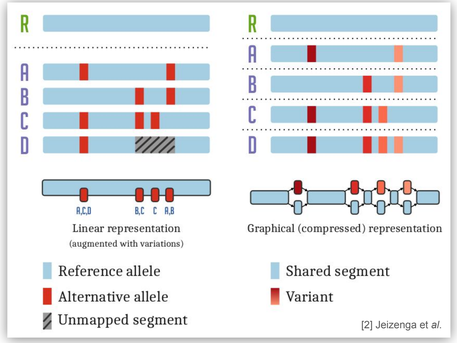
[1]*Quantitative Biology Center (QBiC) Tübingen, University of Tübingen, Tübingen, Germany,* [2]*University of Rome Tor Vergata, Via della Ricerca Scientifica 1, Rome, Italy,* [3]*Genomics Institute, University of California, Santa Cruz, Santa Cruz, CA, USA,* [4]*Biomolecular Engineering and Bioinformatics, University of California Santa Cruz, Santa Cruz, CA, USA*

**Contributed equally.*

Supported by: Federal Ministry for Economic Affairs and Energy, on the basis of a decision by the German Bundestag; ZIM Zentrales Innovationsprogramm Mittelstand

Pangenome graphs built from raw sets of alignments may have complex structures which can introduce difficulty in downstream analyses, visualization, mapping, and interpretation. Graph sorting aims to find the best node order for a 1D and 2D layout to simplify these complex regions. Pangenome graphs embed linear pangenomic sequences as paths in the graph, but to our knowledge, no algorithm takes into account this biological information in the sorting. Moreover, existing 2D layout methods struggle to deal with large graphs. We present a new layout algorithm to simplify a pangenome graph, by using path-guided stochastic gradient descent (SGD[3]) to move a single pair of nodes at a time. We exemplify how the 1D path-guided SGD implementation is a key step in general pangenome analyses such as pangenome graph linearization and simplification.

VARIATION GRAPHS ENCODE PANGENOMES

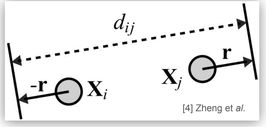


A pangenome[1] models the full set of genomic elements in a given species or clade. It can efficiently be encoded[2] in the form of a variation graph, which embeds the linear sequences of the pangenome as paths in the graphs themselves.

https://bit.ly/PangenomeGraph
https://bit.ly/OptimizedDynamicGraphImplementation

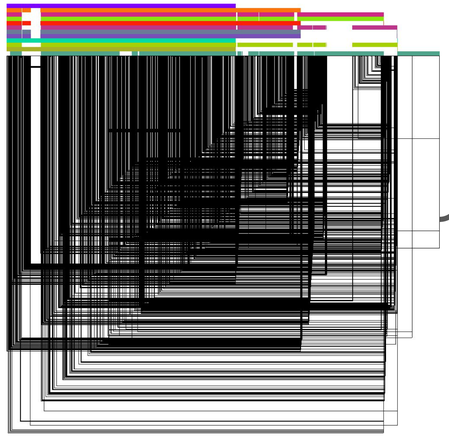
PATH-GUIDED STOCHASTIC GRADIENT DESCENT

Our algorithm moves a single pair of nodes at a time, optimizing the disparity between the layout distance of a node pair and the actual nucleotide distance of a path traversing these nodes.

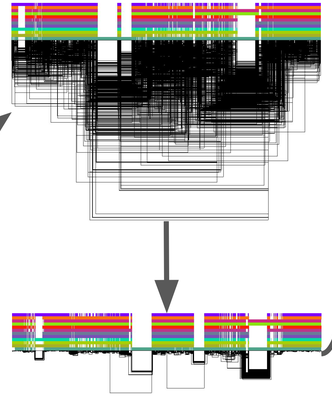


- The first node X_i of a pair is a uniform path step pick from all nodes.
- The second node X_j of a pair is sampled from the same path following a Zipfian distribution.
- The path nucleotide distance of the nodes in the pair guides the actual layout distance d_{ij} update of these nodes. The magnitude r of the update depends on the current learning rate of the SGD.

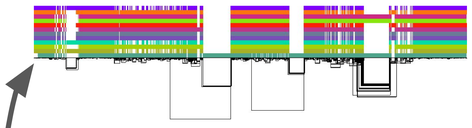
Unsorted graph in 1D



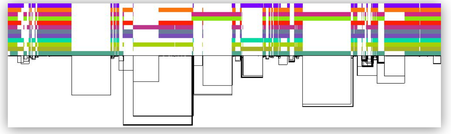
Intermediate snapshots in 1D



Sorted graph in 1D

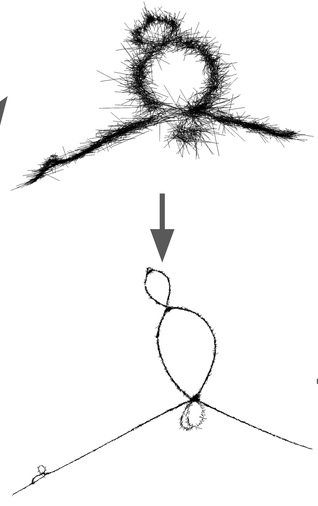


Smoothed graph in 1D

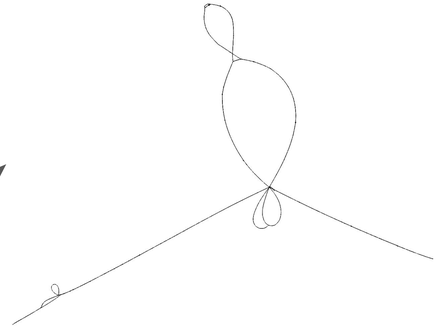


Unsorted graph in 2D

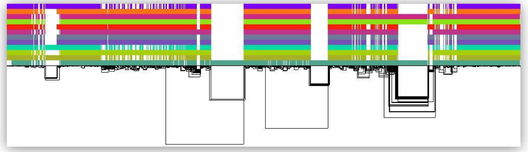
Intermediate snapshots in 2D



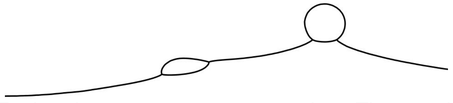
Sorted graph in 2D



GRAPH VISUALIZATIONS EXPLAINED



- The graph nodes' are arranged from left to right forming the pangenome's sequence.
- The colored bars represent the binned, linearized renderings of the embedded paths versus this pangenome sequence in a binary matrix.
- The black lines under the paths, so called links, represent the topology of the graph.



- Each dot represents a node. The node's x-coordinates are on the x-axis and the y-coordinates are on the y-axis, respectively.

GRAPH SIMPLIFICATION PIPELINE

- Smoothxg runs SPOA for each block of paths that are collinear within a seqwish induced variation graph. A prerequisite is that the graph nodes are sorted according to their occurrence in the graph's embedded paths. Our 1D path-guided SGD algorithm is designed to provide this kind of sort.

FUTURE WORK

- Explore the path-guided SGD parameter space
- Compare our proposed 2D graph layouting algorithm with existing pangenome graph visualization tools
- Enhance our 2D drawing method, draw paths in 2D
- Find out performance boundaries applying the algorithms up to gigabase-scale pangenome graphs.

References

1. Eizenga et al. (2020). Pangenome Graphs. *Annual Reviews of Genomics and Human Genetics*, 21, 1.
2. Eizenga et al. (2020). Efficient dynamic variation graphs. *Bioinformatics*, btaa640.
3. Zheng et al. (2019). Graph Drawing by Stochastic Gradient Descent. *IEEE Transactions on Visualization and Computer Graphics*. 25, 2738-2748.

Acknowledgements

We thank Vincenza Colonna for organizing the Crusco Summer Hackathon and the Forentum Ritrovato museum for hosting it.
We thank the deNBI cloud for providing computational resources.
S.H. acknowledges funding from the Central Innovation Programme (ZIM) for SMEs of the Federal Ministry for Economic Affairs and Energy of Germany.

de.NBI GERMAN NETWORK FOR BIOINFORMATICS INFRASTRUCTURE

Questions

- ***How/why*** *node sorting* ***improves*** *read mapping* ***speed***, ***memory***, *and* ***accuracy*** *against a pangenome graph?*

  → When the graph is well sorted, the mapping is easier because the nodes are already in their linear order, so we can build chains for mapping easier, improving seeding for the colinear chains during the mapping (minimap2).

  → Most pangenome variation graphs may have large scale structural variation globally, but, usually, they are locally linear or partially orderable. We can sort these graphs so that their colinear regions are represented contiguously within a sort order, and our algorithm is suited for that. This lets us use efficient collinear chaining methods to find target mappings. One can use SPOA to obtain a base-level alignment.
- *What about non-biological graphs? Is the algorithm suitable also for other types of graphs?*

  →Yes, it is. Our implementation only requires a graph represented as a variation graph (you can take a look at the GFAv1 format specifications). This is a type of sequence graph which embeds samples as paths in the graph itselfs, paths that our algorithm exploits in the sorting. However, you don't need to embed paths in your graph, because we have already implemented a tool (odgi cover) to cover the graph, generating paths in it.

Questions

- What about memory requirements of your sorting algorithm?
  → ~ 2-4 times the GFA input. Depends on the complexity of the graph. Instead of holding all node pairs distances in memory, we use a succinct path index to randomly sample the node pair we want to update. **Therefore**, we overcome the major limit of **Zheng** et al.: **Quadratic memory requirement!**
- Why is there visually much less sequence in the **smoothxg viz** compared to all the other 1D sorts?
  → Here SPOA is changing the alignments. “If the synteny requirement is too big then some of the sequences will stop mapping against each other around SVs.”
- Does there already exist a similar method on how you transformed the SGD into a multi-threaded version?
  → There is a work which describe formally what we are already doing: Hogwild!, which is a lock-free multi-threaded SGD, where results of other threads can be overwritten. In our case, the datastructure to work on for optimizing the sorting is sparse: this means that each gradient update only modify a small part of the decision variable. This is a condition for Hogwild!, and our method, to achieve a nearly optimal rate of convergence to a good solution, a good sorting in our case.
- Why do you use the Zipfian distribution for the sampling of the second node?
  → Focus on a short scale, occasionally samples larger values. **By experience.** Because we want to update node pairs which are close in path space more often than node pairs that are far away in path space in order to resolve the locally complex regions of the graph.
  → We can try a linear one.

Questions

- Why is a sorted odgi graph smaller on disk?
  → Edge space: relativistic encoding: small deltas use 1 byte, larger ones several bytes
  → bit width for paths, will dominate the effect for the edges
- Can you mathematically quantify the quality of a sort?
  → There are several possible way to do that:
  → You can define a stress function, calculating the distances of all possible pairs of nodes, trying to minimize it, avoiding node overlapping. We want to point out that this formulation is not doable in practise, because the complexity of its computation is quadratic respect to the number of node, and it can't be computed quickly on medium-big graphs.
  → However, we define specific metric which measure the quality of the sorting taking into account the path embedded in the graph. For example, in 1D, we go through the graph following each path, considering a penality each time an edge go back respect to the linear order. Moreover, this metric is computable in linear time respect the number of nodes, avoding the complexity problem.