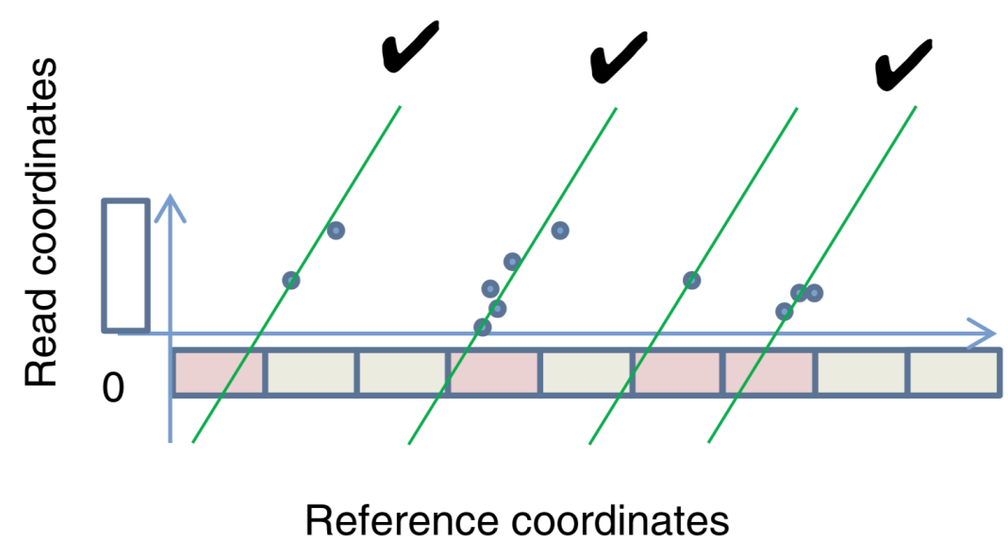
Distance Indexing and Seed Clustering in Sequence Graphs

Authors: Xian Chang, Jordan Eizenga, Adam M. Novak, Jouni Sirén, Benedict Paten

Presented by Nae-Chyun Chen, 2/12/2020

Distance between two positions

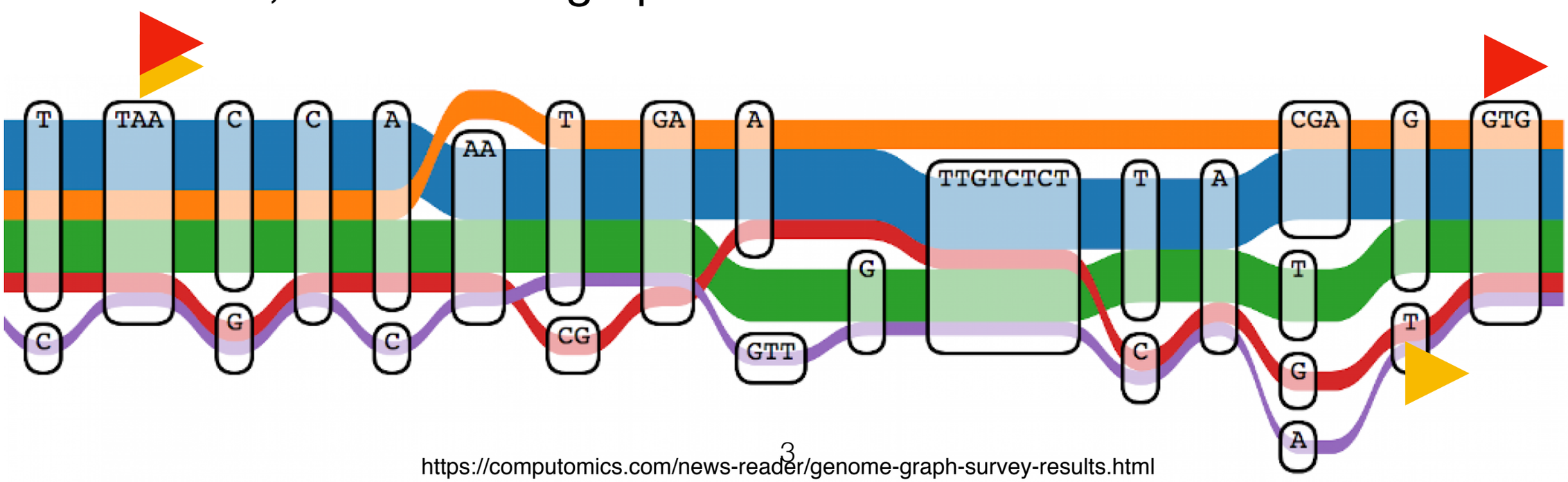
- Minimum distance: minimum oriented traversal distance between two positions
- Straightforward when the reference genome is linear
  - $offset_a - offset_b$
- Important in seed chaining problem



Sović, Ivan, et al. "Fast and sensitive mapping of nanopore sequencing reads with GraphMap." *Nature communications* 7.1 (2016): 1-11.

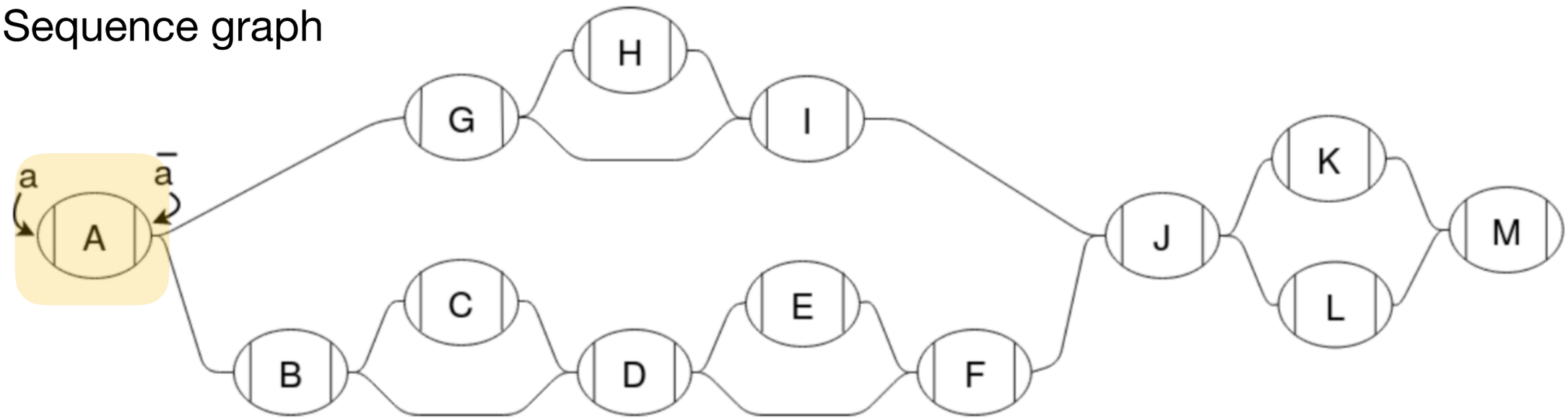
Distance in a graph genome?

- Not so straightforward
- `vg` approximated distance using a path-based method
  - If there's a shared path, treat as linear
  - If not, traverse the graph



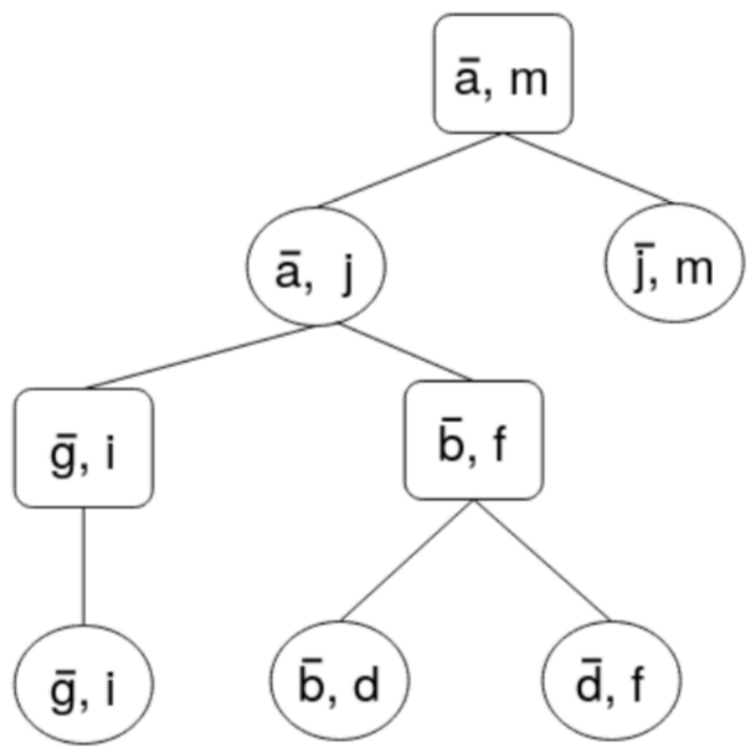
https://computomics.com/news-reader/genome-graph-survey-results.html

Sequence graph

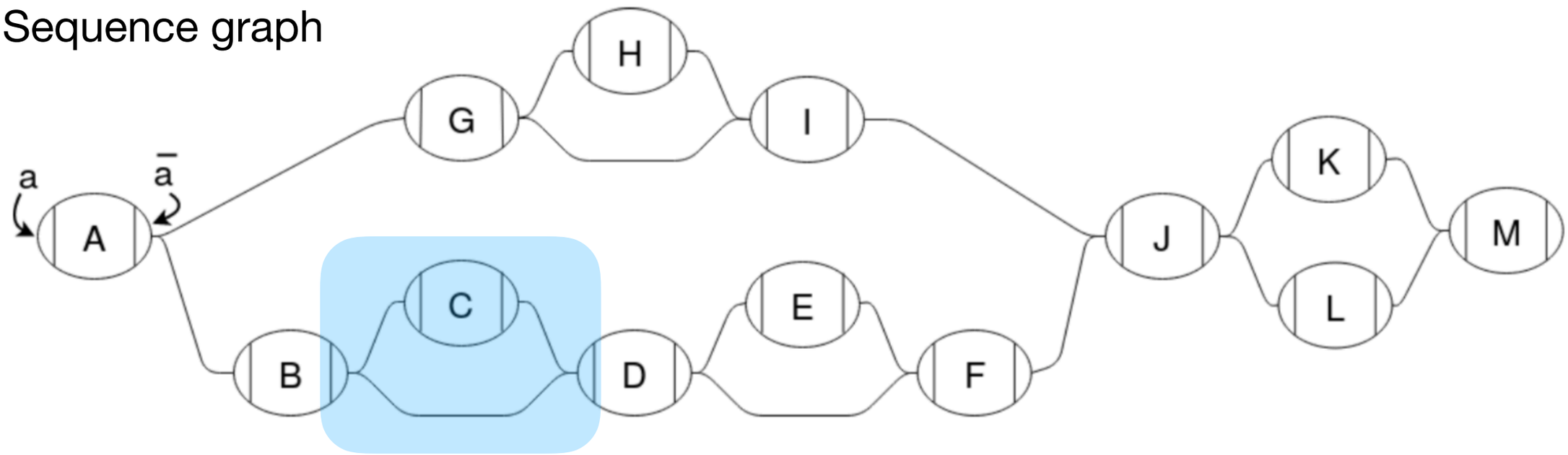


Node: a sequence of nucleotides, has two sides (oriented)

Snarl tree

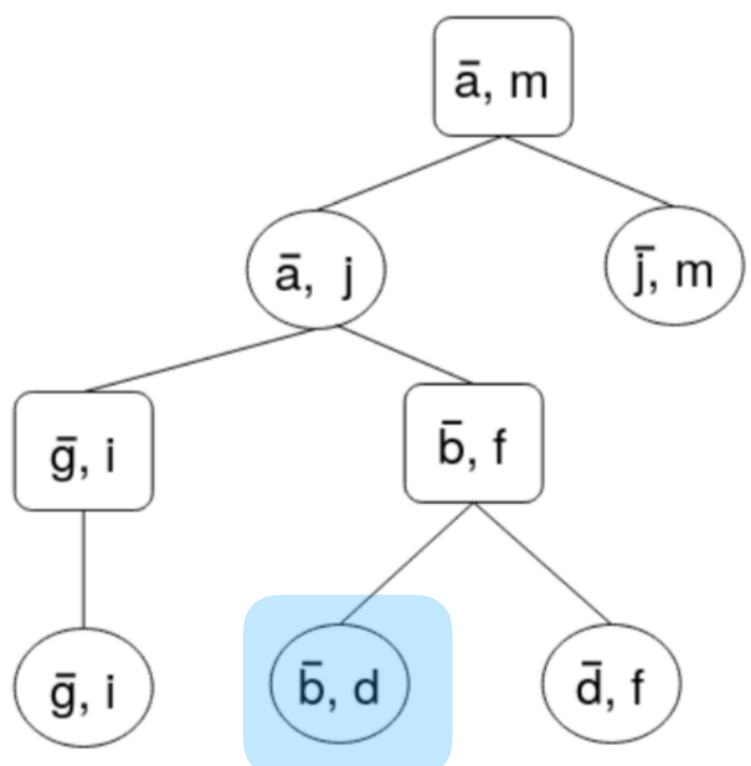


Sequence graph

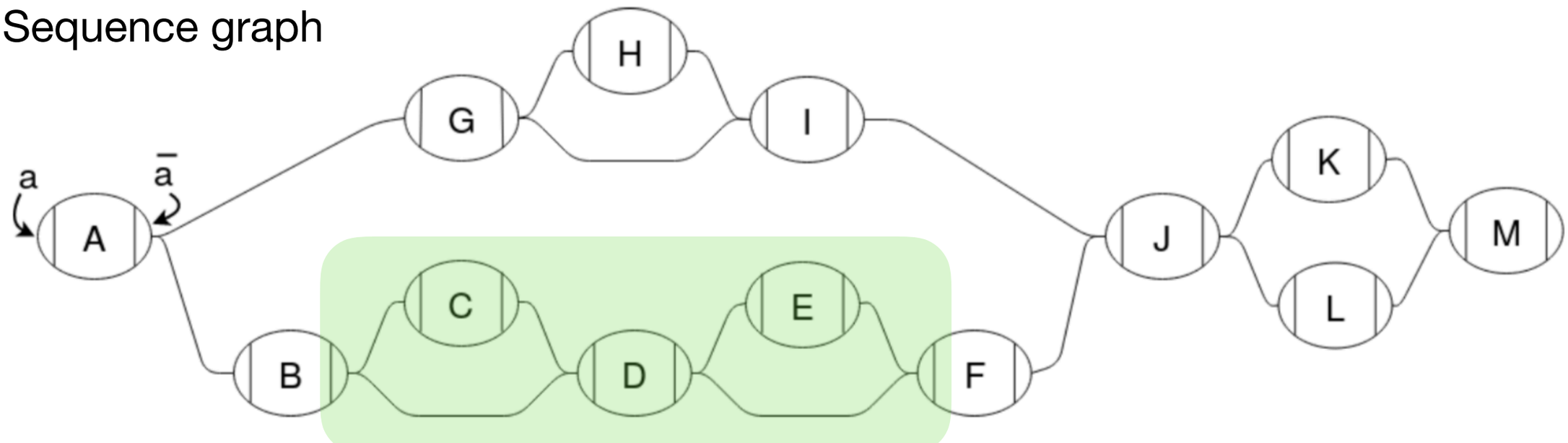


Snarl: a separable and minimal subgraph (oval nodes in a snarl tree)

Snarl tree



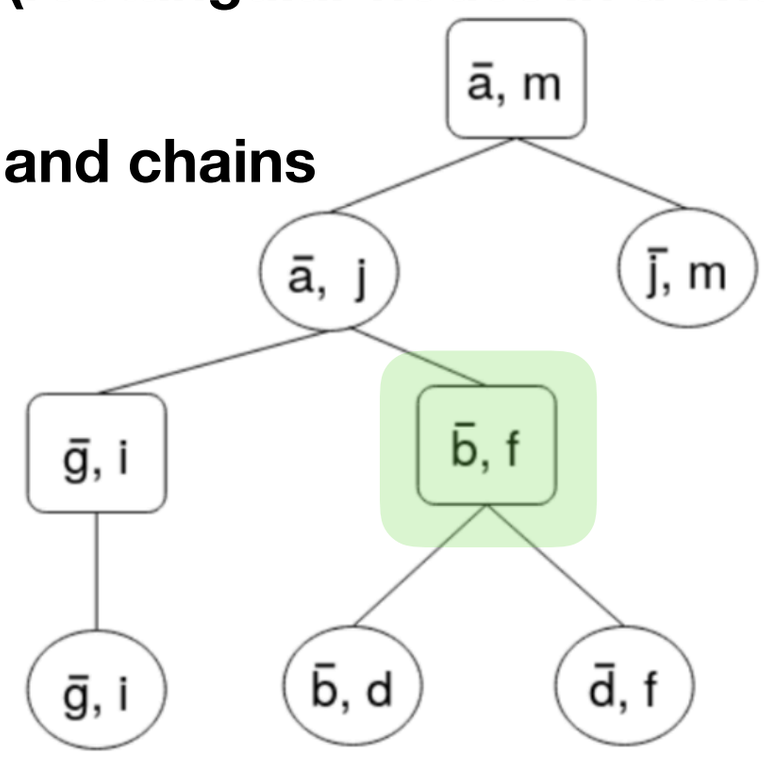
Sequence graph



Chain: a sequence of contiguous snarls (rectangular nodes in a snarl tree)

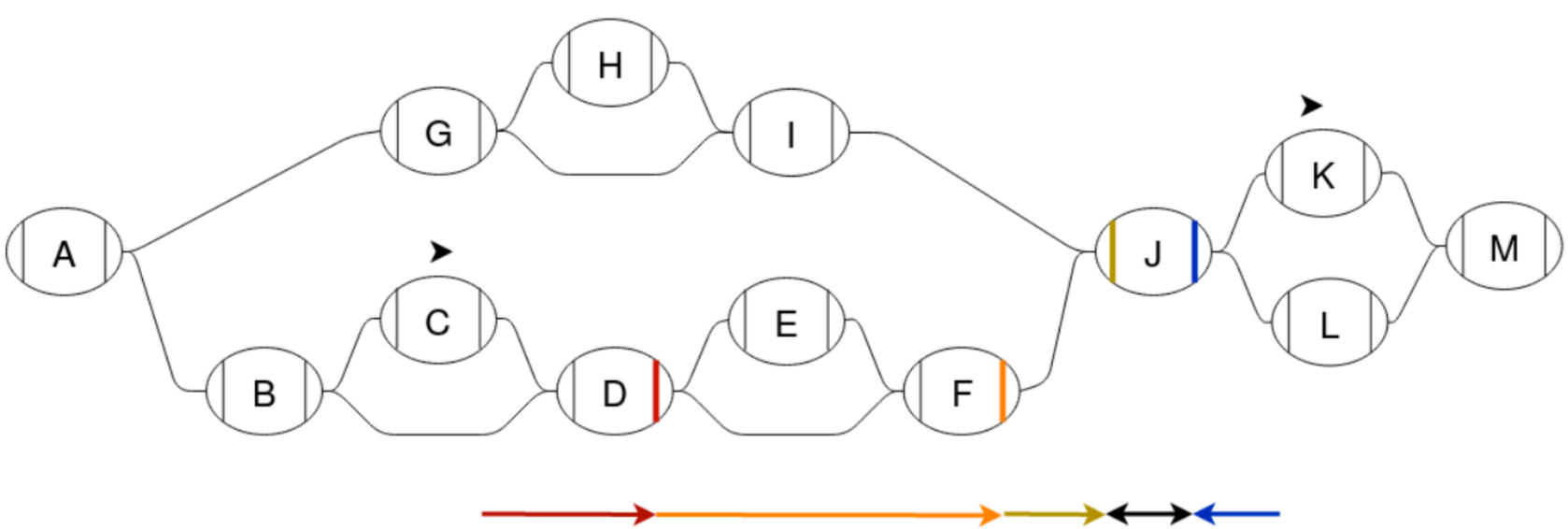
Snarl tree

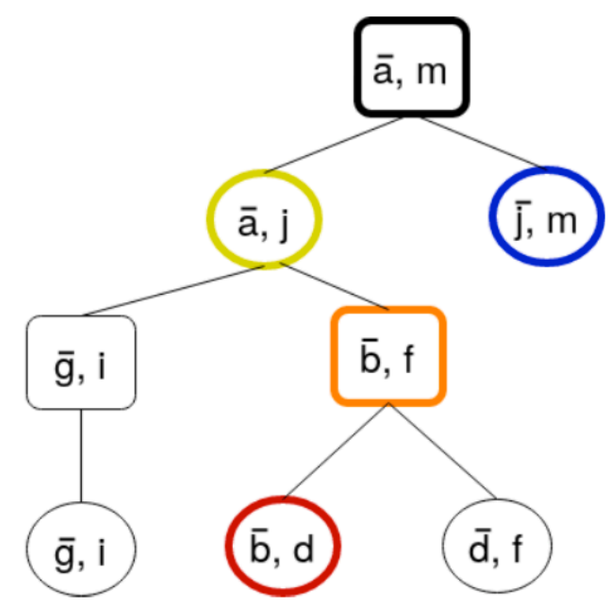
Snarl tree: alternating snarls and chains



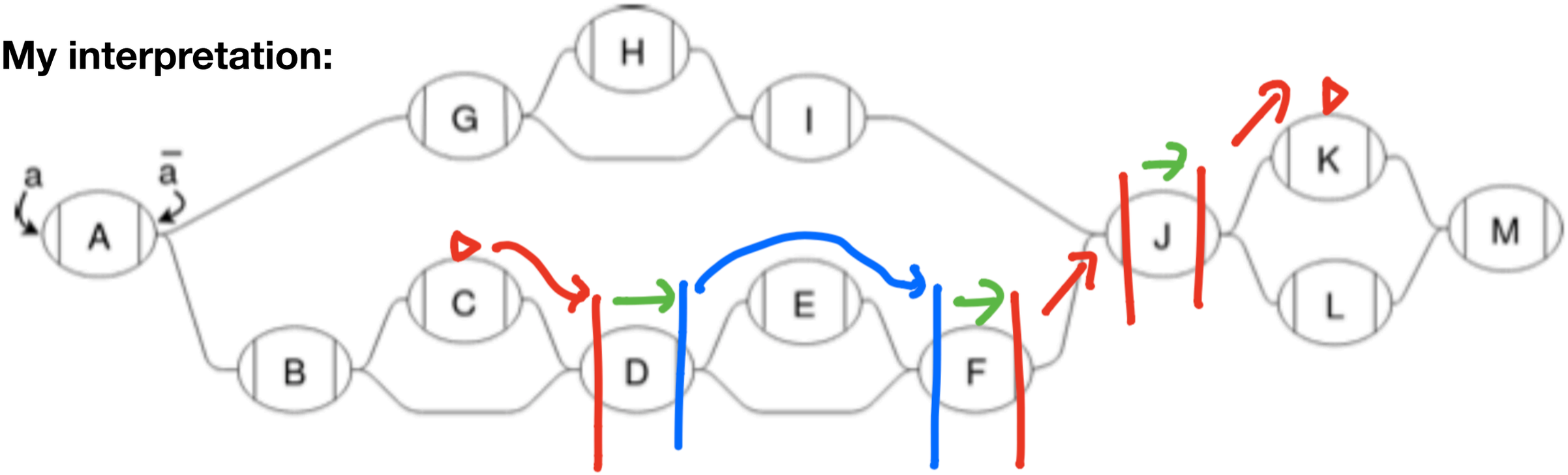
Split distance property

- Minimum distance can be broken up into:
  - From node/chain boundaries to the boundaries of their parent snarl
  - From snarl boundaries to their parent chain boundaries
  - Between sibling structures in their parent structure





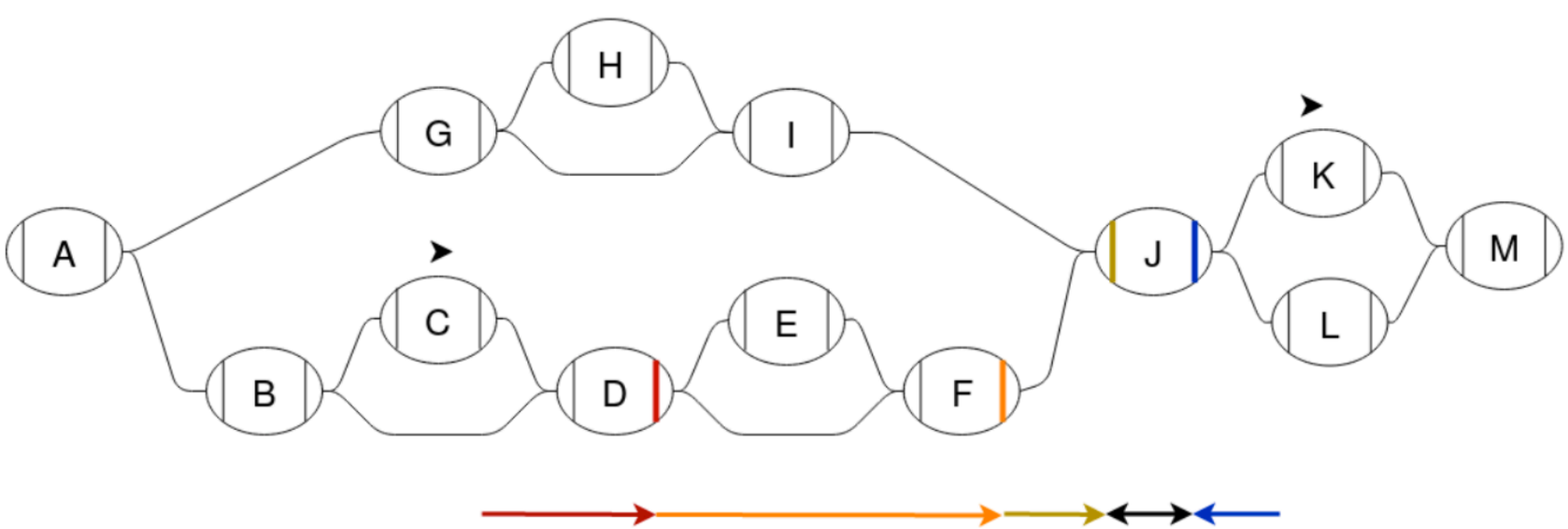
My interpretation:

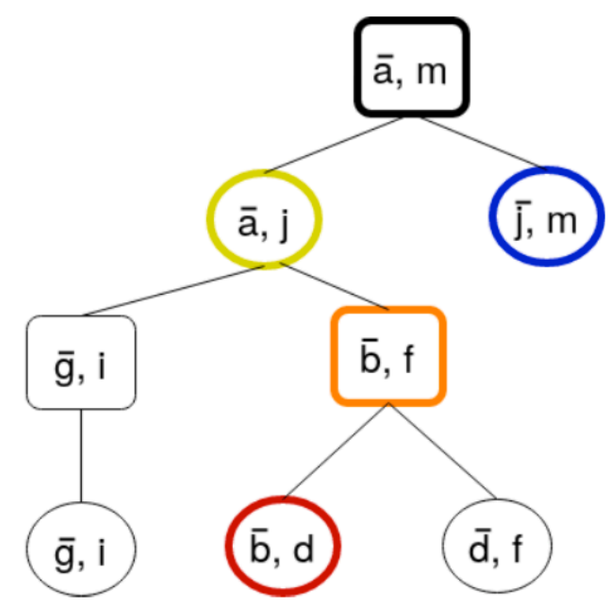


node 2 snarl

snarl 2 chain

struct 2 struct

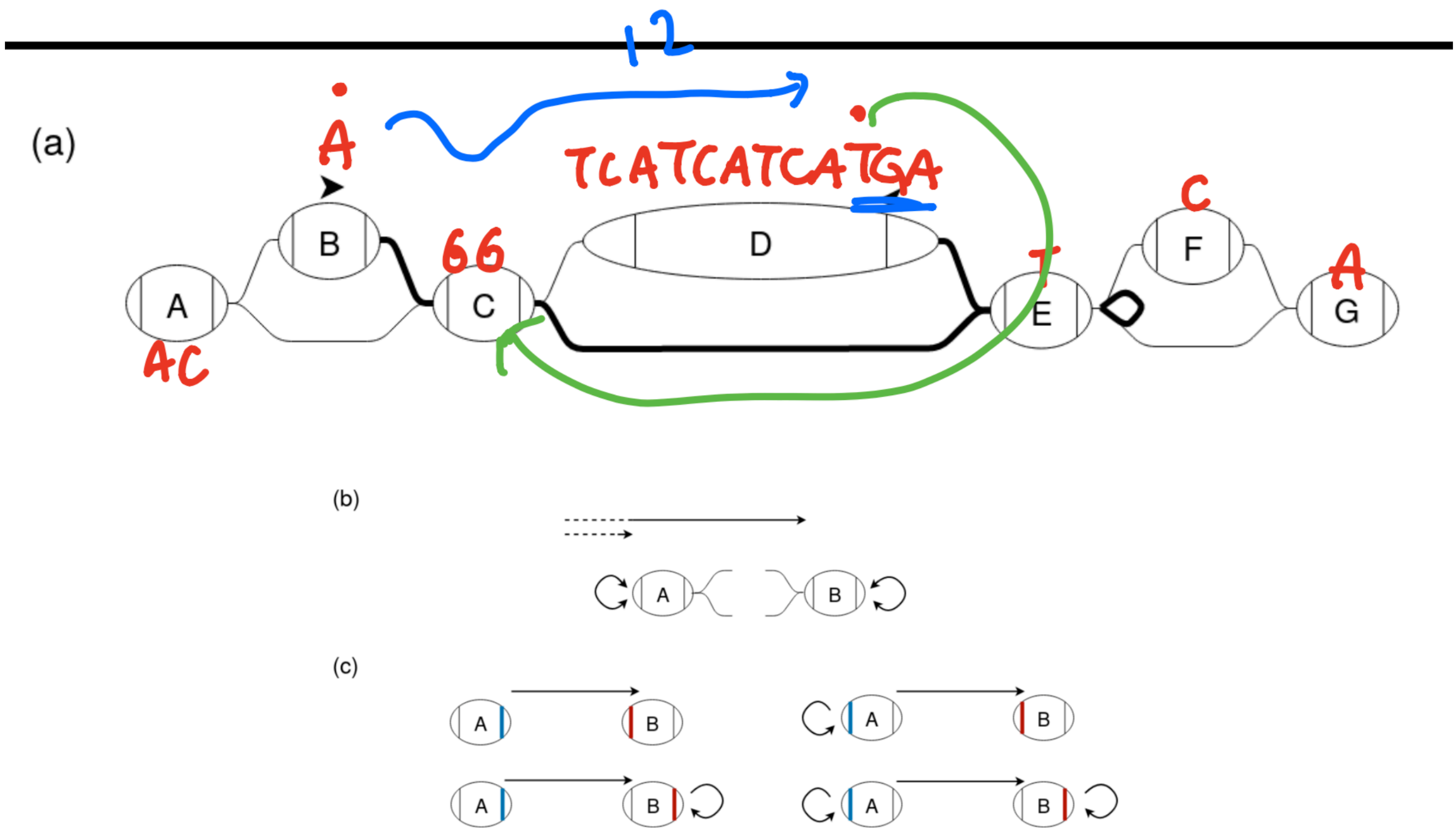




Minimum distance index

- Snarl index
  - Minimum distances between every pair of child structures
- Chain index
  - Minimum distances from chain boundaries to child snarls' boundaries
  - Loop indexes to store forward and backward distance

Reverse direction

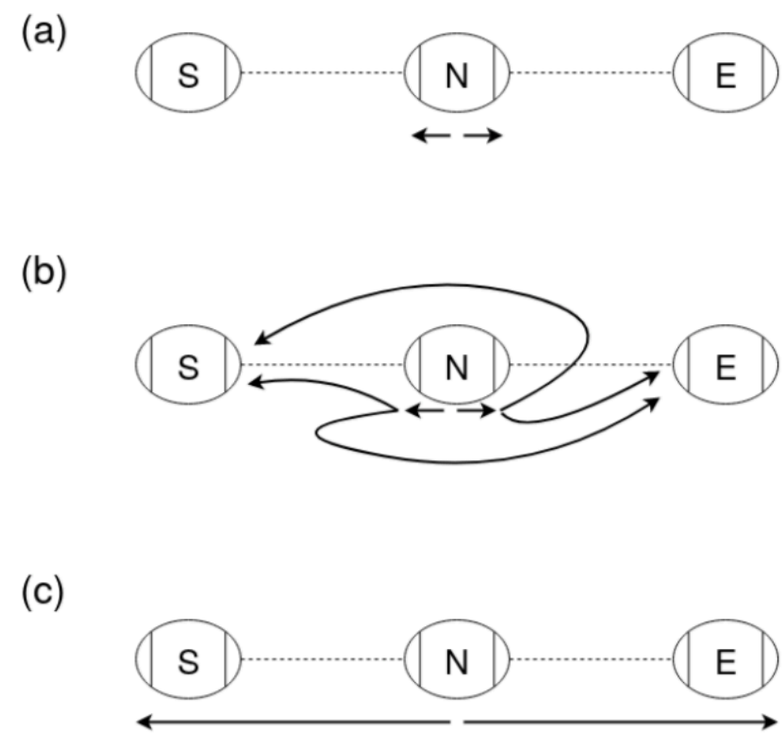


Minimum distance index

- Building:
  - Dijkstra to build snarl index;
    traversing snarls to build chain index
- Size: $O(\sum_{S} n_s^2 + \sum_{C} n_c)$
  - S=snarl, C=chain, n_s=#structs in snarl s, n_c=#structs in chain c
  - 12.2 GB on disk, 17.7GB in memory

minDist()

- Find the least common ancestor (LCA) for both positions: $O(d)$
- Calculate distances to both boundaries of the LCA: $O(d)$, see right figure
- Traverse all the way up to the root of the snarl tree: $O(d)$



Algorithm 1: distToAncestor(*position*, *ancestor*): Given a position and ancestor structure, return the minimum distance from the position to both sides of a child of the ancestor and the child

```
begin
    struct ⟵ parentOf(position)
    dist_l, dist_r ⟵ distances from position to ends of node, one is ∞
    while parentOf(struct) is not ancestor do
        /* Find the minimum distance from position to the boundaries of each ancestor */
        dist_l, dist_r ⟵ distToEndsOfParent(struct, dist_l, dist_r)
        struct ⟵ parentOf(struct)
    return dist_l, dist_r, struct
```

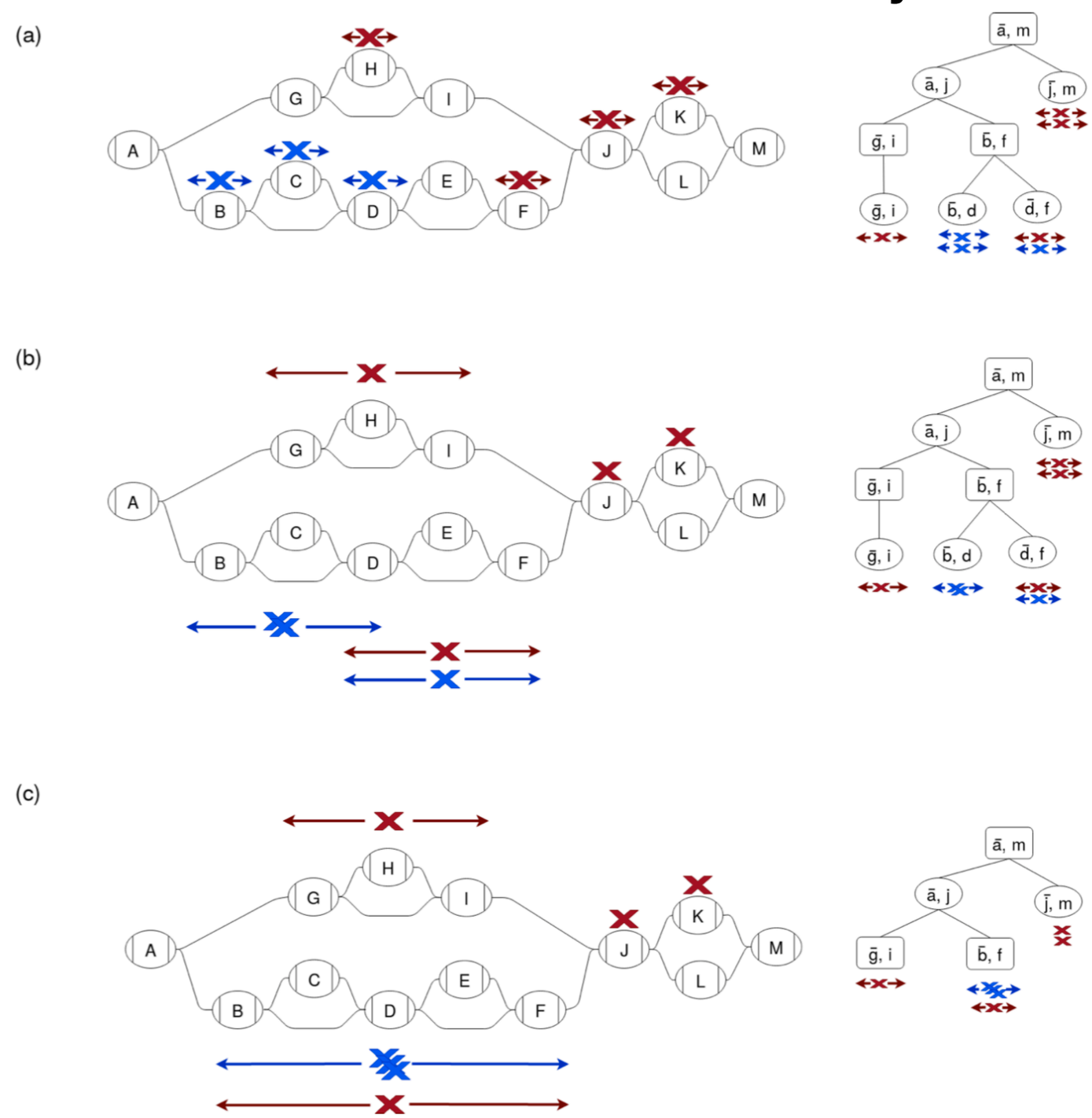
Algorithm 2: minDistance(*position_1*, *position_2*): Return the minimum distance from *position_1* to *position_2*, ∞ if no path between them exists

```
begin
    /* Get distances from each position to the ends of a child of the least common ancestor */
    ancestor ⟵ leastCommonAncestor(position_1, position_2)
    dist1_l, dist1_r, struct_1 ⟵ distToAncestor(position_1, ancestor)
    dist2_l, dist2_r, struct_2 ⟵ distToAncestor(position_2, ancestor)
    min_dist ⟵ ∞
    while ancestor is not root of snarl tree do
        /* Given the distance from each position to both sides of a child of ancestor, find the
           minimum distance between the two positions in ancestor */
        min_dist ⟵ min(min_dist,
            distWithinStructure(ancestor, struct_1, struct_2, dist1_l, dist1_r, dist2_l, dist2_r))
        dist1_l, dist1_r ⟵ distToEndsOfParent(struct_1, dist1_l, dist1_r)
        dist2_l, dist2_r ⟵ distToEndsOfParent(struct_2, dist2_l, dist2_r)
        struct_1 ⟵ ancestor, struct_2 ⟵ ancestor
        ancestor ⟵ parentOf(ancestor)
    return min_dist
```

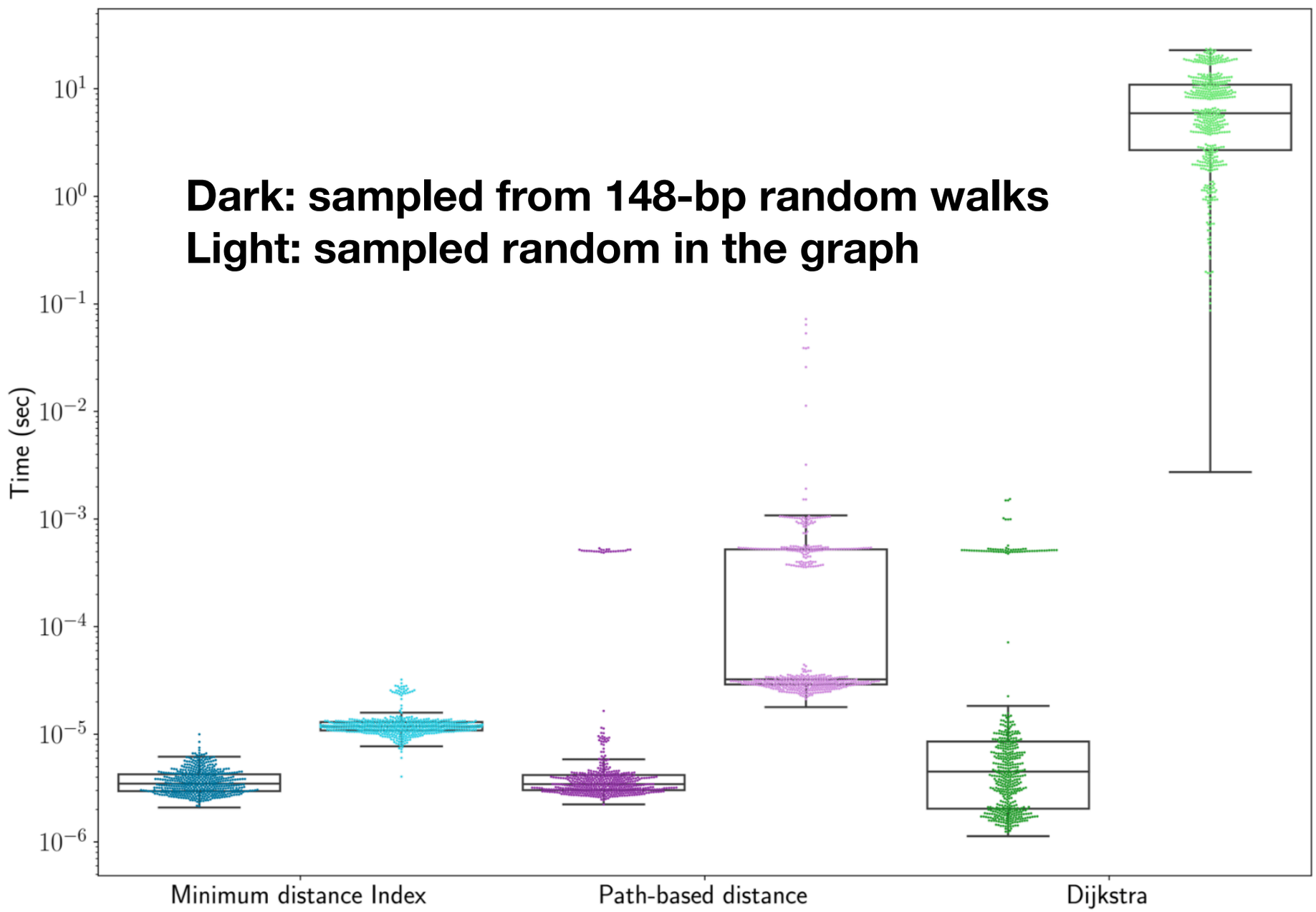
Seed clustering

- Progressively agglomerate clusters if they are close enough
- $O(dn^2)$ in time
- Faster in practice

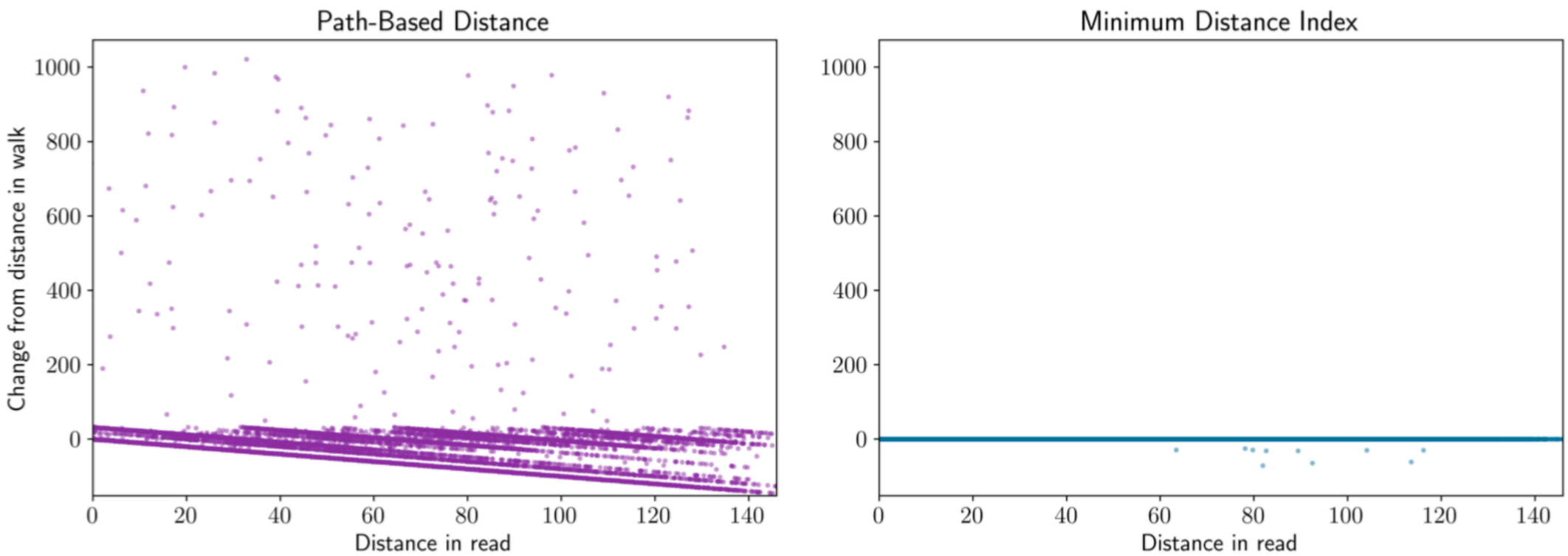
Positions colored by the final clusters



minDist() runtime

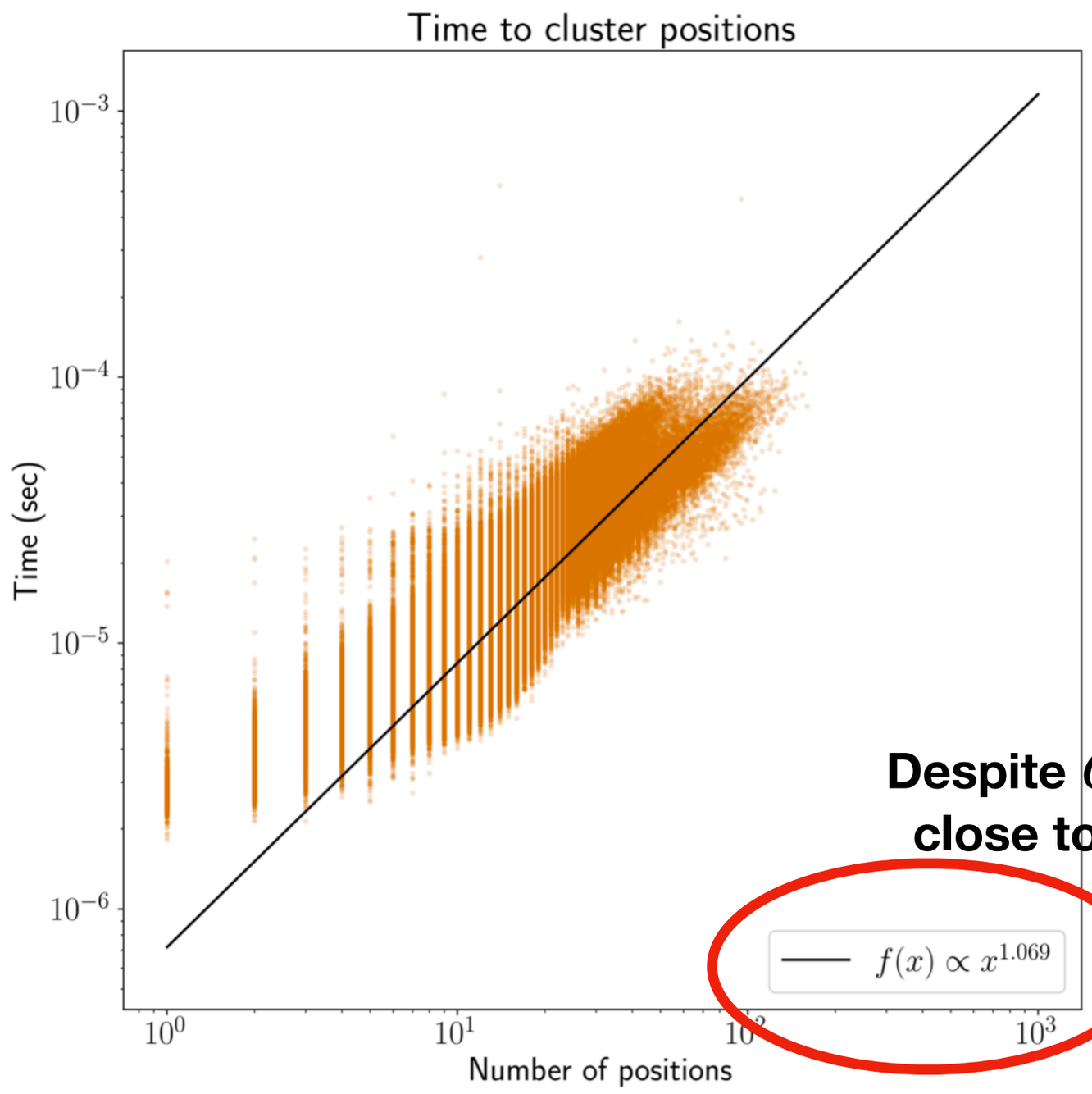


Accuracy



- Simulate a structural variant graph
- Sample positions from 148-bp random walks overlapping SVs

Clustering runtime



Despite $O(dn^2)$ worst case, close to linear in practice

Summary

- A simple and elegant distance calculation method for graph genome, and can be used for seed clustering
- Faster and more accurate than current approach, but high memory overhead
  - Is clustering the bottleneck of graph-based alignment?
  - See great potential for improvement, e.g. heuristics, early-termination mechanism, index compression
- Code is not available