Update

Andres

University of California, Riverside

November 19, 2018

Outline

Algorithms

- Split the problem in two stages:
  1. Find maximal disks at each timestamp (MaximalFinder) and
  2. Join maximal disks between adjancent timestamps (FlockFinder)
- Pseudocode for both algorithms available online: MaximalFinder[1] and FlockFinder[2].

[1] `https://tinyurl.com/y74lld5k`
[2] `https://tinyurl.com/yac26guf`

Maximal finder overall steps

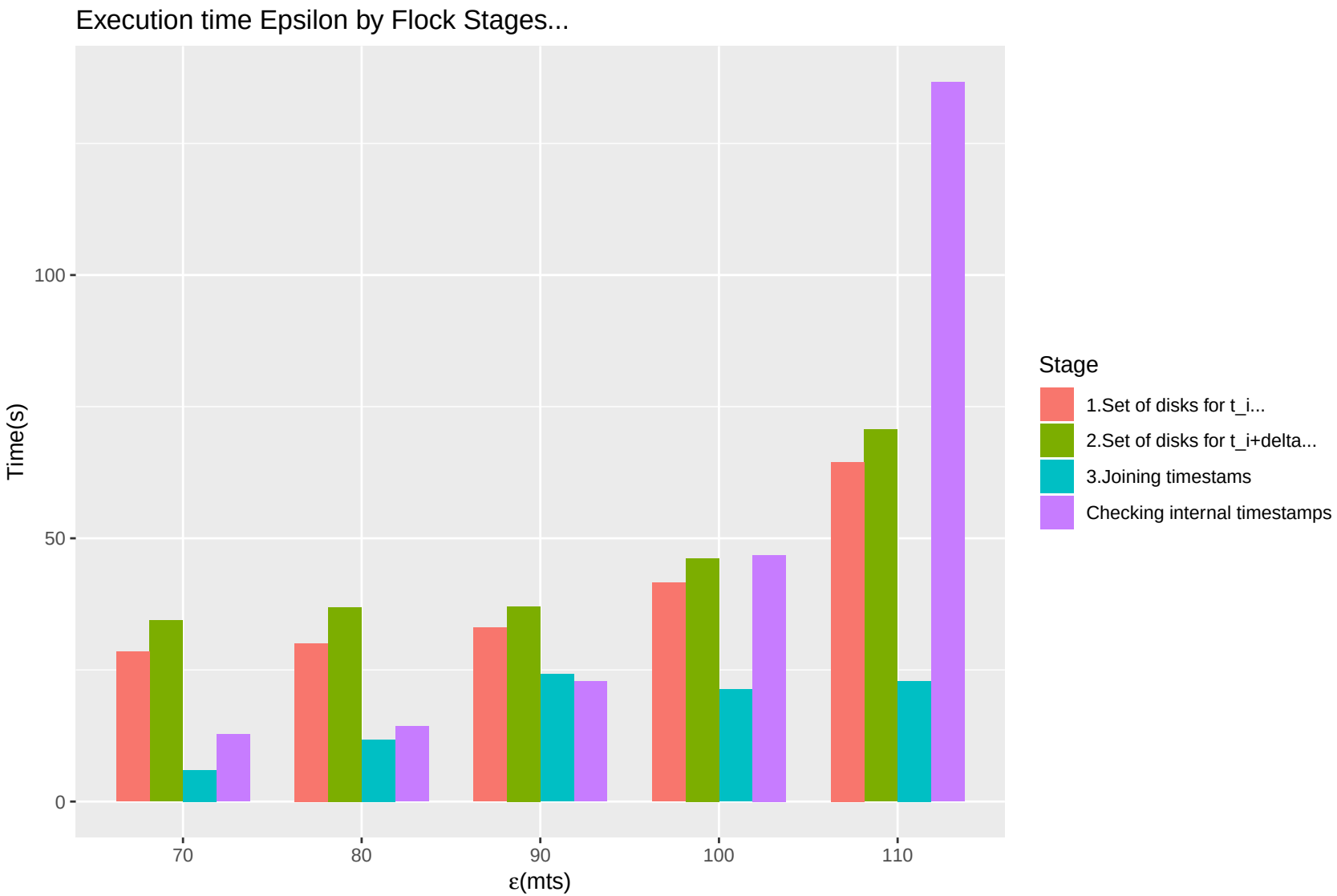
1. Indexing points...
2. Getting pairs...
3. Computing centers...
4. Indexing centers...
5. Getting disks...
6. Filtering disks $< \mu$...
7. Prunning duplicate candidates...
8. Indexing candidates...
9. Getting expansions...
10. Finding maximal disks.

Flock finder

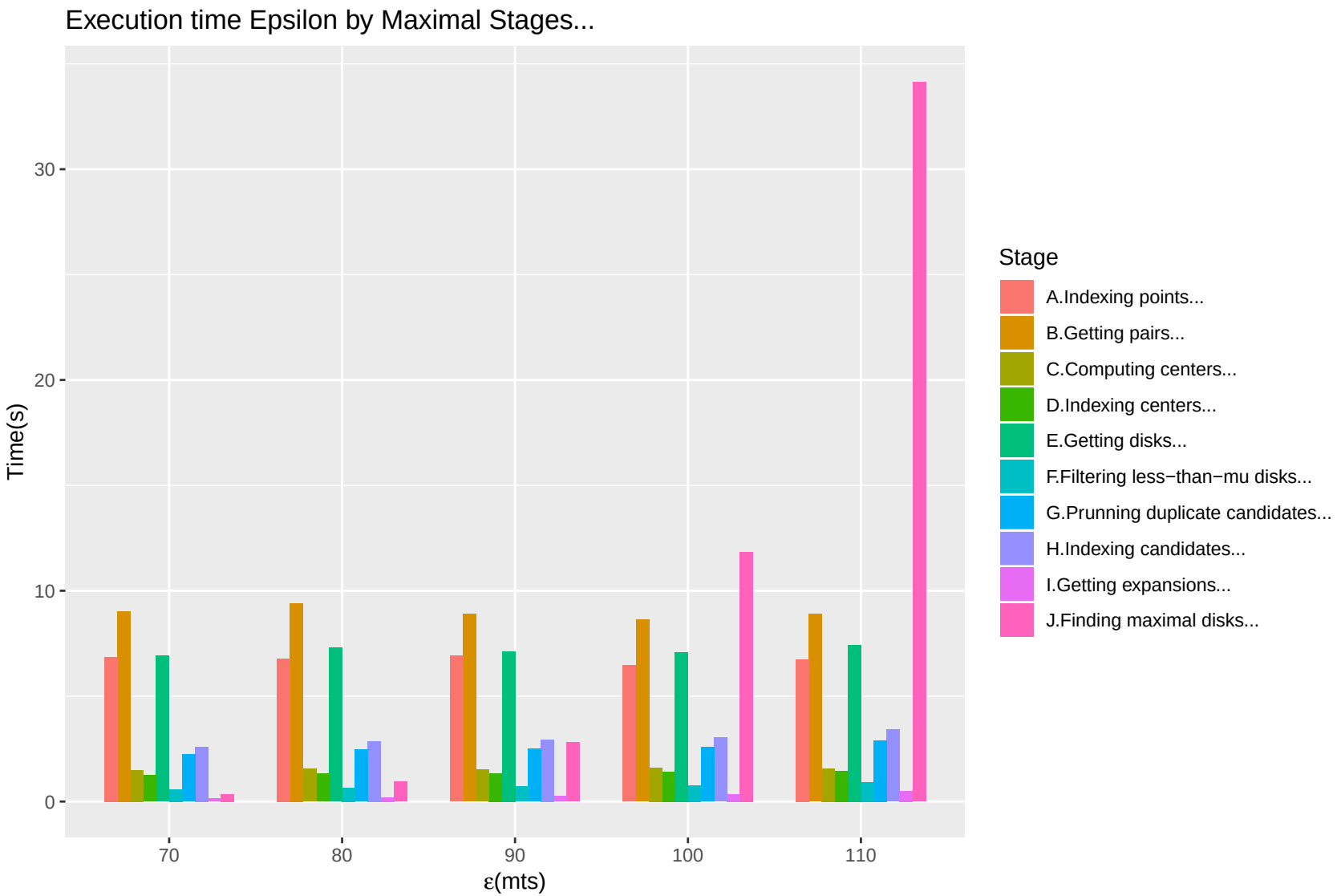
1. Set of disks for t_i...
2. Set of disks for $t_{i+\delta}$...
3. Joining timestams...
4. Checking internal timestamps.

Outline

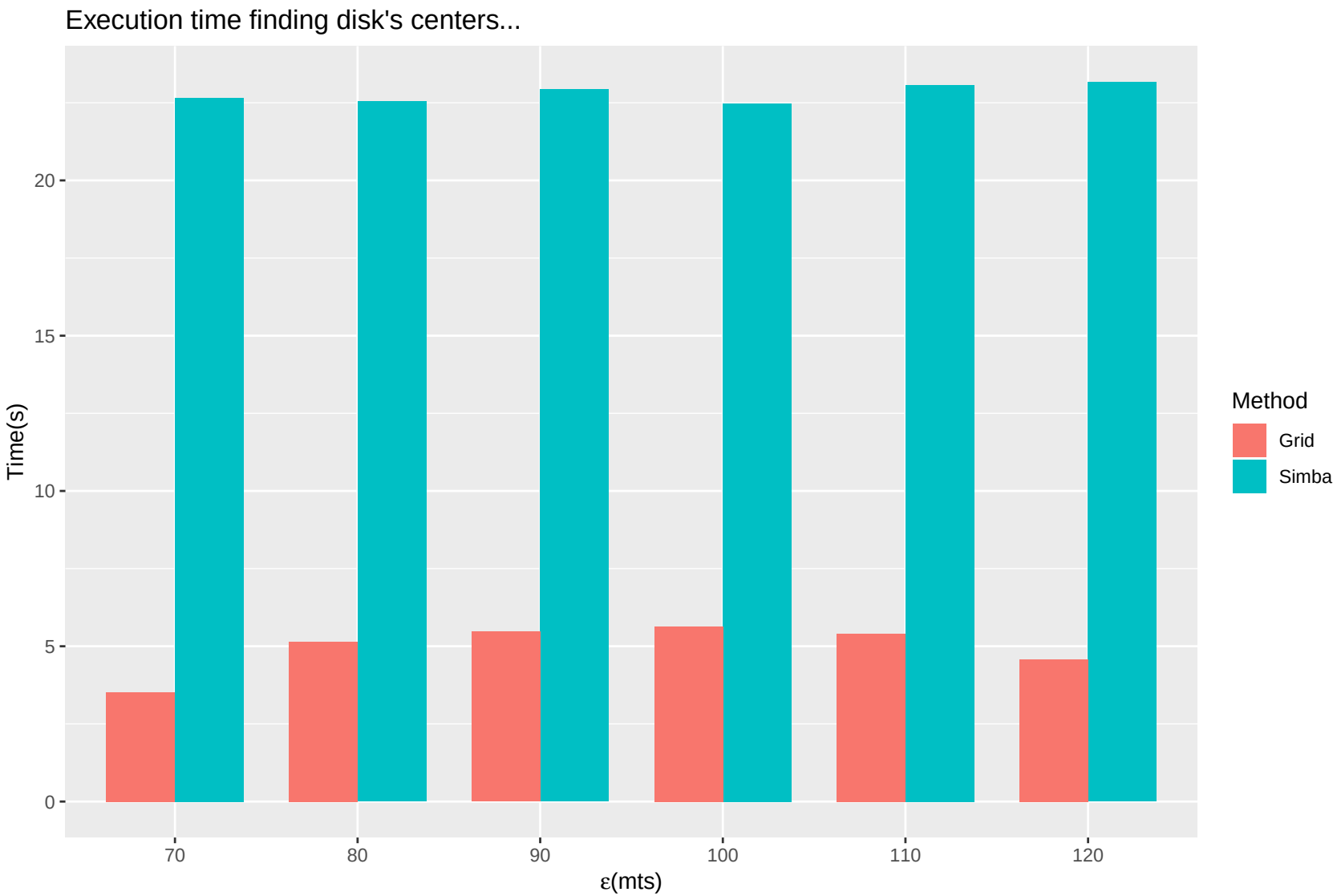
Performance



Performance



Performance



Bottlenecks

1. In flock finder:
   - Checking internal timestamps: When merge last approach prunes enough points it works as expected but large amount of intermediate points have huge impact.
2. In maximal finder:
   - Finding maximal disks: Even the new implementation is more stable, the most costly operation is removing duplicates and redundant disks.
3. Overall:
   - Online approach requires indexing at each timestamp.
   - Simba indexing is slow.

Outline

Possible solutions

1. Explore alternatives in Simba[3]
   - There are QuadTree and KDTree partitioners but they are not fully-integrated as indeces. (QuadTree only support 2D.)
   - For partitioning, RTree is already faster than QuadTree and KDTree in 2D and 3D datasets.
2. Grid indexing
   - Include the Grid partitioner in Simba and work on its index integration.
   - Implement distance join by my own.

[3] Already done.