Scaling Spatial Overlay Operations and Flock Pattern Discovery

Andres Calderon · acald013@ucr.edu

University of California, Riverside

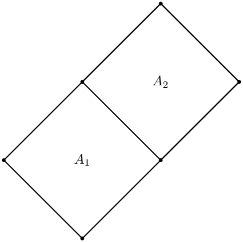
November 5, 2024

Plan

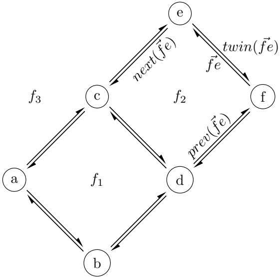
1. Scalable overlay operations over DCEL polygons layers
2. Scaling DCEL Overlay Operations to Support Dangle and Cut Edges
3. Scalable Processing of Moving Flock Patterns

What is a DCEL?

- **Doubly Connected Edge List** - DCEL.
- A spatial data structure to **represent planar subdivisions** of surfaces.
- Represent topological and geometric information as vertices, edges, and faces.
- **Applications**: polygon overlays, polygon triangulation and their applications in surveillance, robot motion planing, circuit board printing, etc.

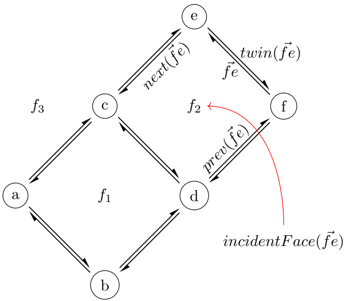


Planar subdivision



DCEL representation

- DCEL uses three tables: Vertices, Faces and Half-edges.



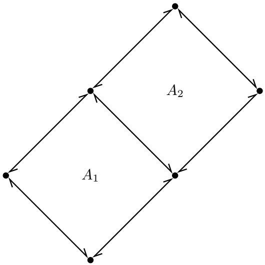
vertex	coordinates	incident edge
a	(0,2)	$\vec{ba}$
b	(2,0)	$\vec{db}$
c	(2,4)	$\vec{dc}$
⋮	⋮	⋮

face	boundary edge	hole list
f_1	$\vec{ab}$	*nil*
f_2	$\vec{fe}$	*nil*
f_3	*nil*	*nil*

half-edge	origin	face	twin	next	prev
$\vec{fe}$	f	f_2	$\vec{ef}$	$\vec{ec}$	$\vec{df}$
$\vec{ca}$	c	f_1	$\vec{ac}$	$\vec{ab}$	$\vec{dc}$
$\vec{db}$	d	f_3	$\vec{bd}$	$\vec{ba}$	$\vec{fd}$
⋮	⋮	⋮	⋮	⋮	⋮

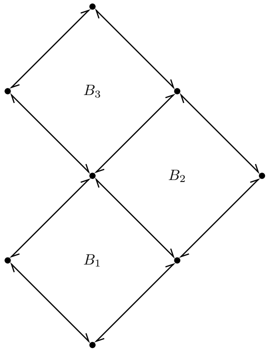
DCEL Advantages

- **Efficiency**: very efficient for computation of *overlay operators*.
- **Re-usability**: allows multiple operations over the same DCEL.
- **Pipelining**: the output of a DCEL operator can be input to another DCEL operator.



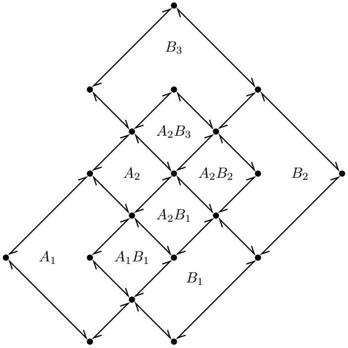
DCEL Advantages

- **Efficiency**: very efficient for computation of *overlay operators*.
- **Re-usability**: allows multiple operations over the same DCEL.
- **Pipelining**: the output of a DCEL operator can be input to another DCEL operator.



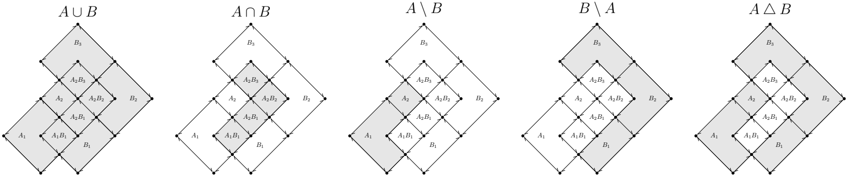
DCEL Advantages

- **Efficiency**: very efficient for computation of *overlay operators*.
- **Re-usability**: allows multiple operations over the same DCEL.
- **Pipelining**: the output of a DCEL operator can be input to another DCEL operator.



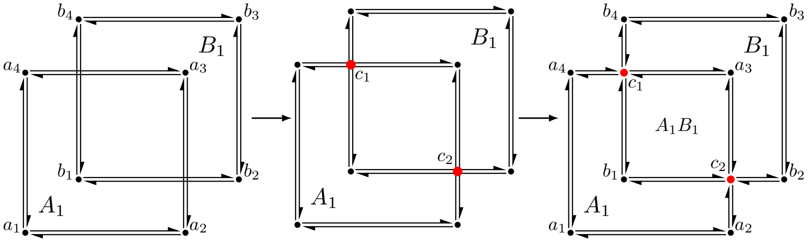
DCEL Advantages

- **Efficiency**: very efficient for computation of *overlay operators*.
- **Re-usability**: allows multiple operations over the same DCEL.
- **Pipelining**: the output of a DCEL operator can be input to another DCEL operator.

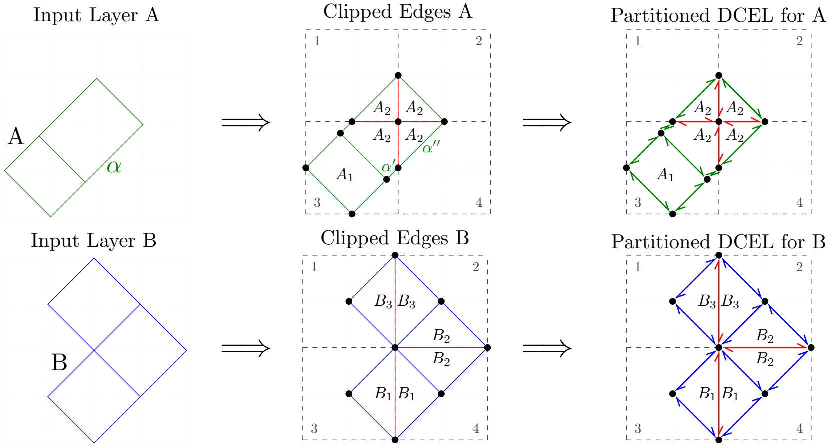


- Currently only sequential DCEL implementations exist.
- Unable to deal with large datasets (i.e. US Census tracks at national level).
- We propose a *scalable distributed* approach to compute the overlay of two polygon layers using DCELs.
- Distribution enables scalability, but introduces challenges: the ***orphan-cell*** problem and the ***orphan-hole*** problem.

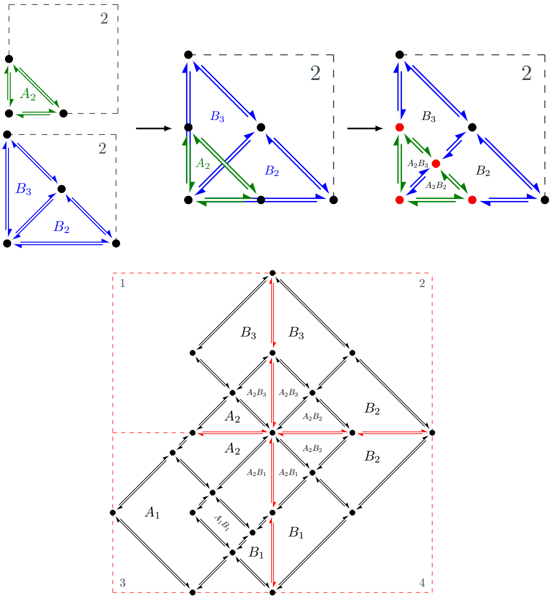
- Consider two (simple) input DCELs A_1 and B_1. The sequential algorithm first finds the intersections of half-edges.
- Then, new vertices (e.g. c_1, c_2) are created, half-edges are updated, new faces are added and labeled (e.g. A_1B_1).



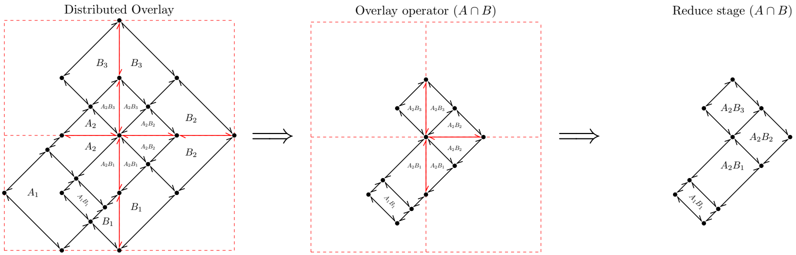
- Two phases: (1) Distributed DCEL construction, and (2) Distributed overlay evaluation.
- Distribution is based on a spatial index (e.g. quadtree)
- Each input DCEL layer (e.g. A, B) is partitioned using the same index
- Each index cell should contain all information needed so that it can compute the overlay DCEL locally
- For each cell to be independent, we need to create "artificial" edges and vertices


Input Layer A
A
α
Clipped Edges A
1
2
A2 A2
A2 A2
α''
α'
A1
3
4
Partitioned DCEL for A
1
2
A2 A2
A2 A2
A1
3
4
Input Layer B
B
Clipped Edges B
1
2
B3 B3
B2
B2
B1 B1
3
4
Partitioned DCEL for B
1
2
B3 B3
B2
B2
B1 B1
3
4

Distributed DCEL construction

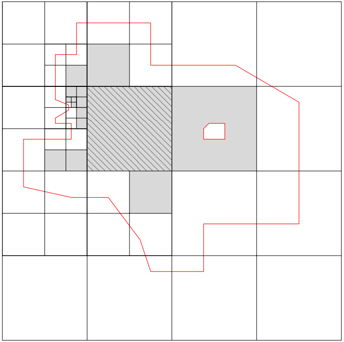


- Answering global overlay queries...
  - To compute a particular overlay operator, we query local DCELs.
  - This work is done independently at each cell (node).
  - SDCEL then collects back all local DCEL answers and computes the final answer (by removing artificial edges and concatenating the resulting faces).



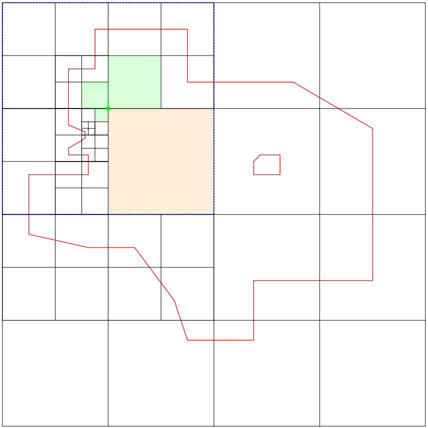
(a) (b) (c)

- We next discuss the **orphan cell** problem (orphan holes are handled similarly).
- A large face (e.g. the red polygon in the figure) can contain cells that do not intersect with any of the face's boundary edges (called *regular edges*).
- Such cells do not contain any label and thus we do not know which face they belong to.



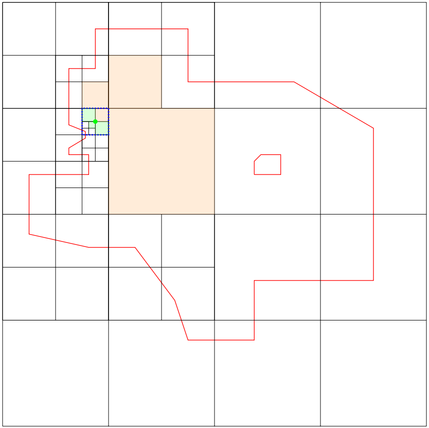
Labeling orphan cells and orphan holes

- We provide an algorithm to efficiently solve the orphan cell problem.

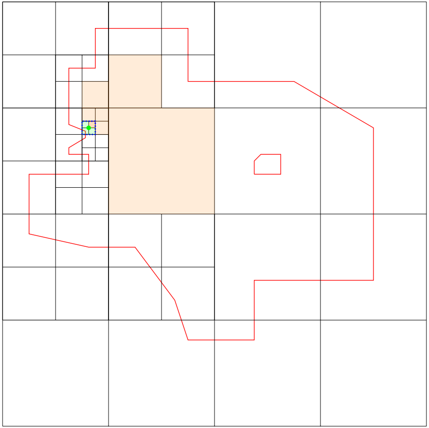


Labeling orphan cells and orphan holes

- We provide an algorithm to efficiently solve the orphan cell problem.



- We provide an algorithm to efficiently solve the orphan cell problem.



Overlay optimizations

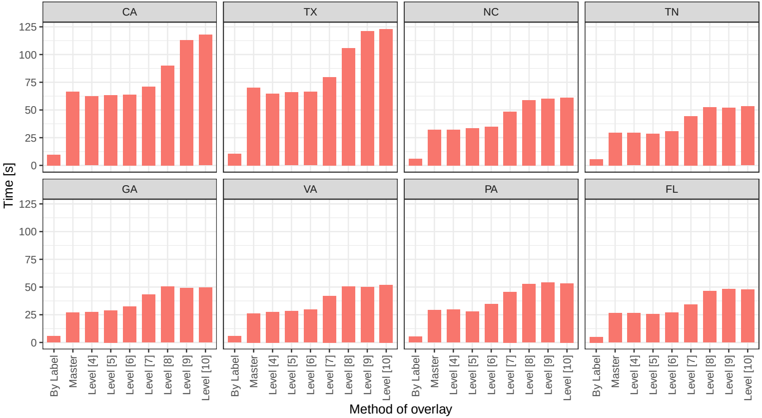
- Optimizing for faces overlapping many cells...
  - Naive approach sends all faces that overlap a cell to a master node (that will combine them).
  - We propose an intermediate reduce processing step.
    - The user provides a level in the quadtree structure and faces are evaluated at those intermediate reducers.
  - We also consider another approach that re-partitions such faces using their labels as the key.
    - It avoids the reduce phase but implies an additional shuffle.
    - However, as we show in the experiments this overhead is minimal.

- Optimizing for unbalanced layers...
  - Finding intersections is the most critical part of the overlay computation.
  - However, in many cases one of the layers has much more half-edges than the other.
  - Sweep-line algorithms to detect intersections run over all the edges.
  - Instead we scan the larger dataset only for the x-intervals where there are half-edges from the smaller dataset.

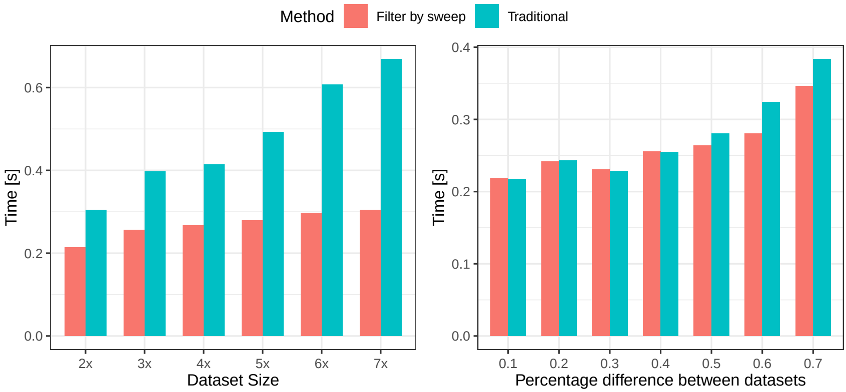
- Datasets.

Dataset	Layer	Number of polygons	Number of edges
MainUS	Polygons for 2000	64983	35417146
	Polygons for 2010	72521	36764043
GADM	Polygons for Level 2	160241	64598411
	Polygons for Level 3	223490	68779746
CCT	Polygons for 2000	7028	2711639
	Polygons for 2010	8047	2917450

- Evaluation of the overlapping faces optimization.

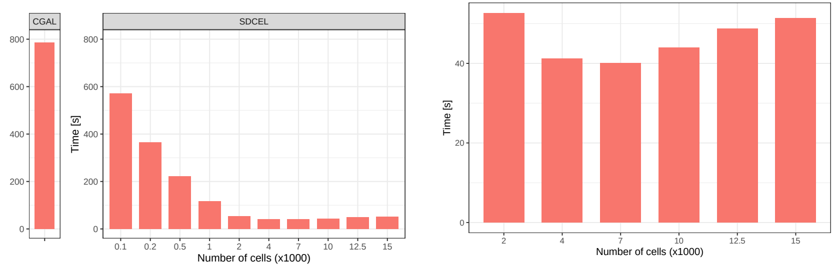


- Evaluation of the unbalanced layers optimization.

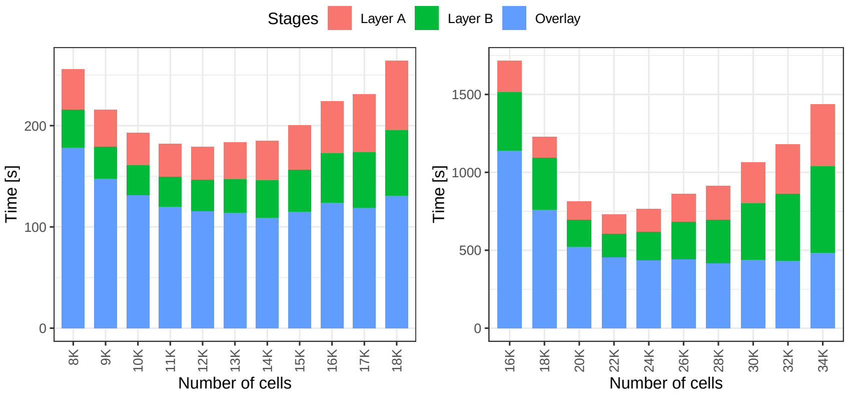


(a) (b)

- Performance varying number of partition cells (CCT dataset).



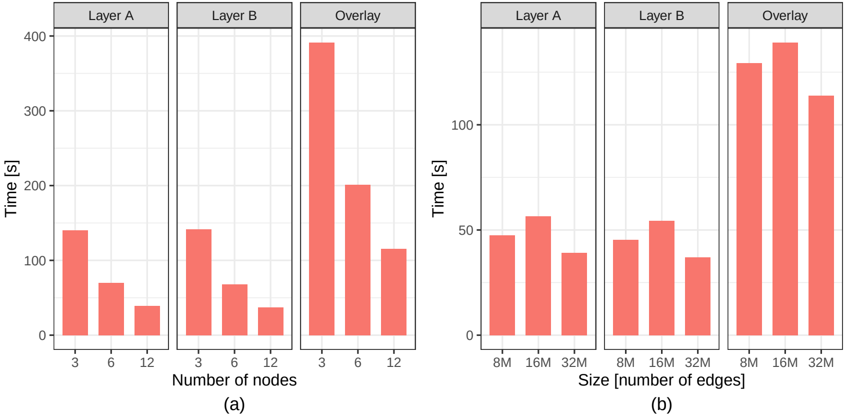
- Performance with MainUS and GADM datasets.



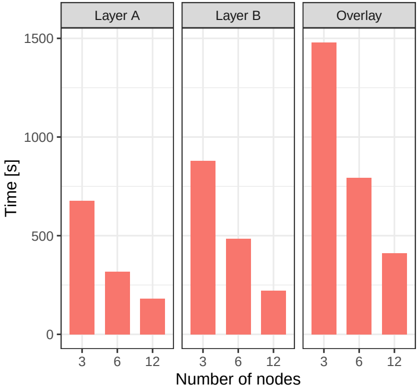
(a) (b)

Experimental evaluation

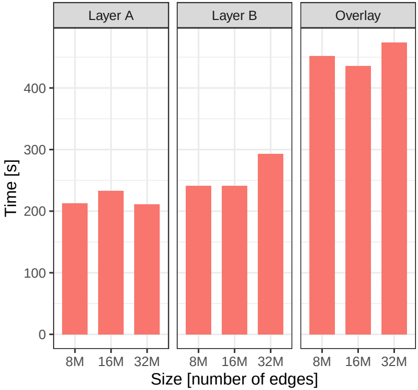
- Scale-up and Speed-up.



- Scale-up and Speed-up.



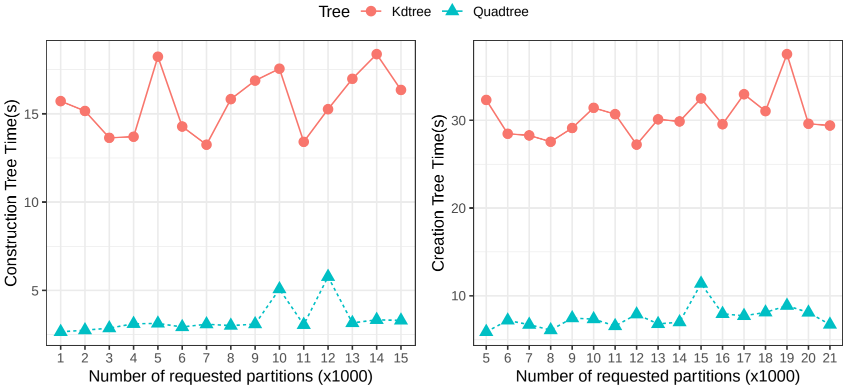
(a)



(b)

Experimental evaluation

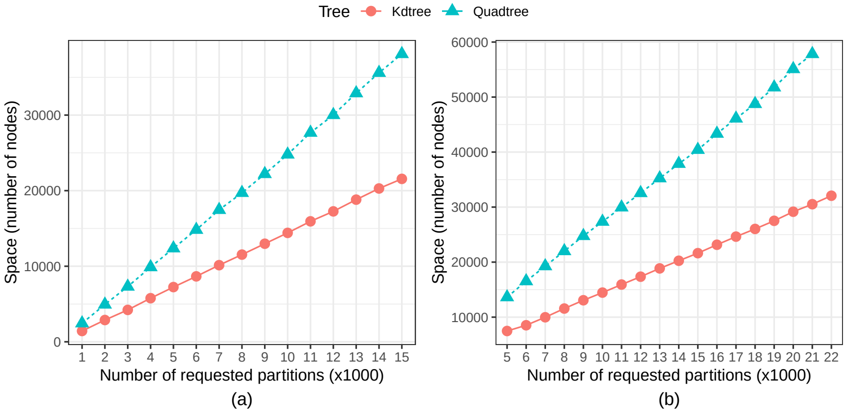
- Space-oriented vs Data-oriented partitioners.



(a) (b)

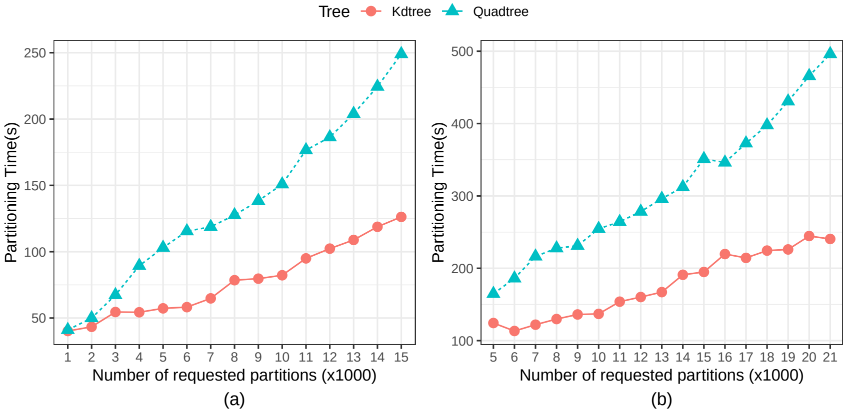
Experimental evaluation

- Space-oriented vs Data-oriented partitioners.

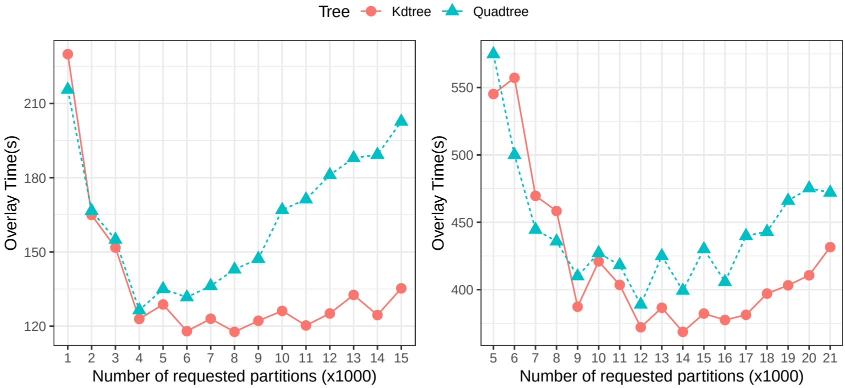


Experimental evaluation

- Space-oriented vs Data-oriented partitioners.

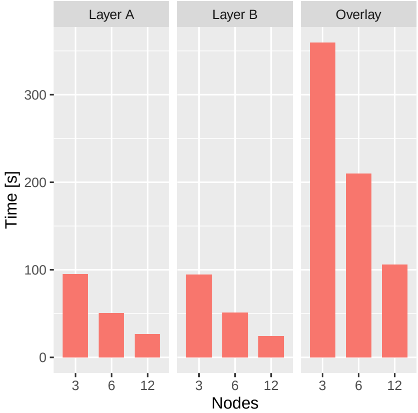


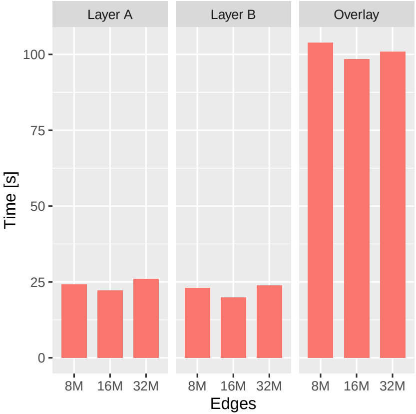
- Space-oriented vs Data-oriented partitioners.



(a) (b)

- Space-oriented vs Data-oriented partitioners.





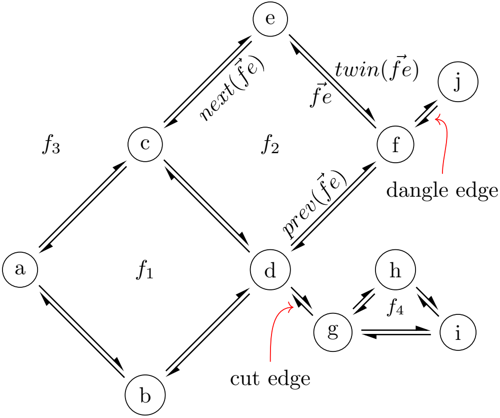
Plan

1. Scalable overlay operations over DCEL polygons layers

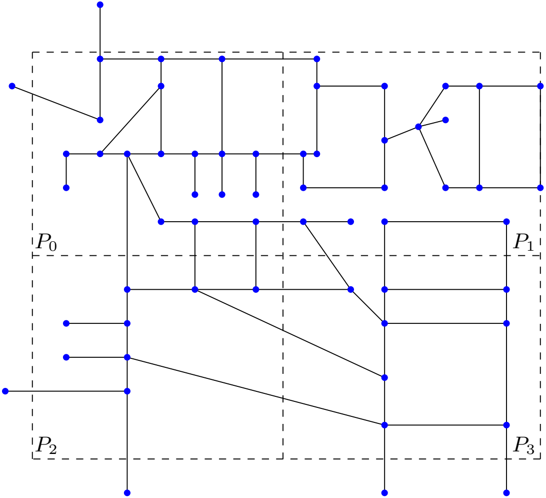
2. Scaling DCEL Overlay Operations to Support Dangle and Cut Edges

3. Scalable Processing of Moving Flock Patterns

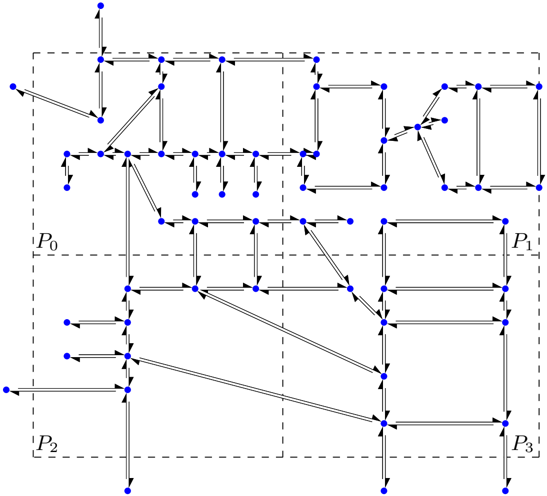
- We extend the overlay DCEL approach to accept scattered and noisy line segments as input, rather than being restricted to clean polygon data.


e
$next(\vec{fe})$
$twin(\vec{fe})$
$\vec{fe}$
j
f_3
c
f_2
f
$prev(\vec{fe})$
dangle edge
a
f_1
d
h
f_4
g
i
cut edge
b

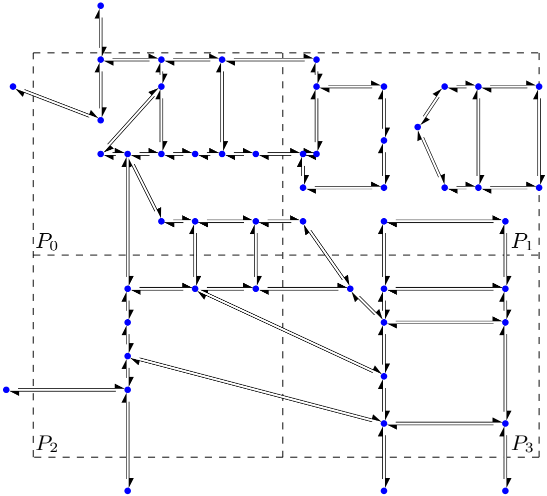
Dangle and cut edges

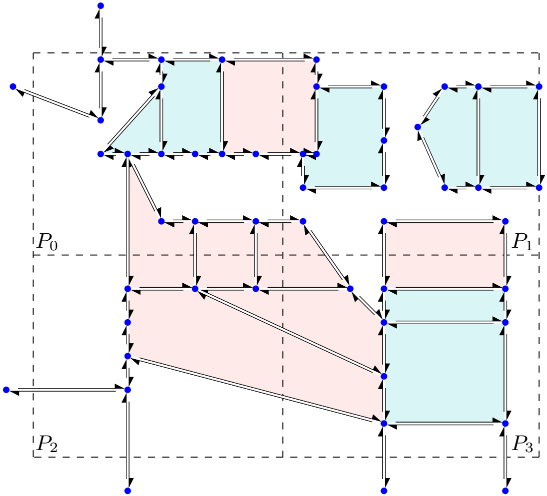


Dangle and cut edges

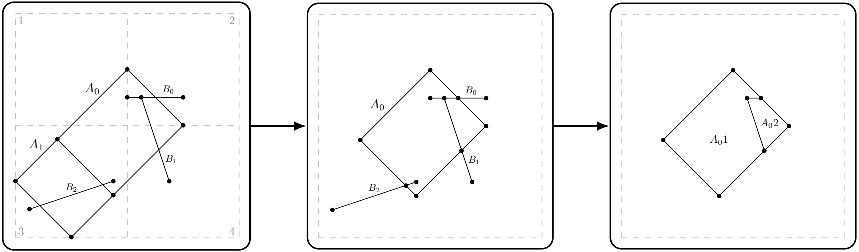


Dangle and cut edges




P_0
P_1
P_2
P_3

Dangle and cut edges

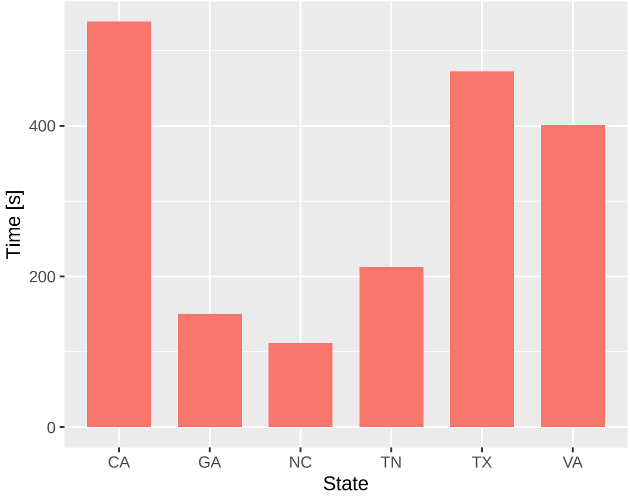


(a) (b) (c)

Dangle and cut edges

Dataset	Number of Polygons Layer A	Number of Edges Layer B	Result Polygons
TN	1,272	3,380,780	41,761
GA	1,633	4,647,171	49,125
NC	1,272	7,212,604	22,413
TX	4,399	8,682,950	98,635
VA	1,554	8,977,361	38,941
CA	7,038	9,103,610	96,916

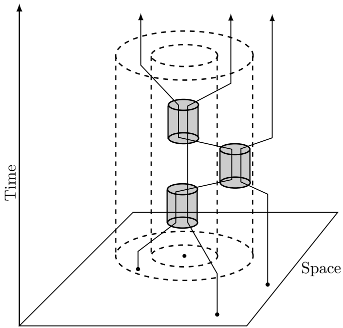
Dangle and cut edges



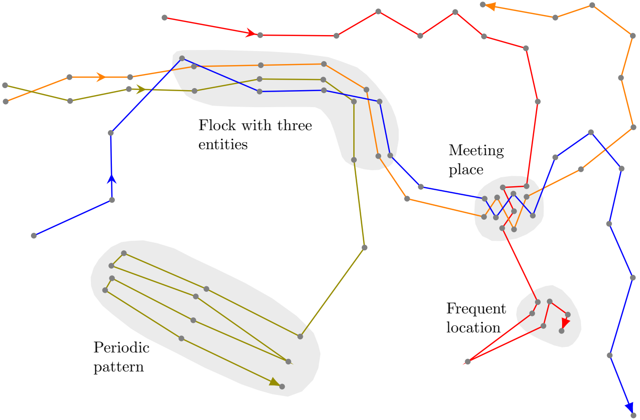
Plan

1. Scalable overlay operations over DCEL polygons layers
2. Scaling DCEL Overlay Operations to Support Dangle and Cut Edges
3. Scalable Processing of Moving Flock Patterns

- A spatial trajectory is a trace in time generated by a moving entity in a geographical space.
- i.e. $p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$
- A trajectory is stored as a time-ordered sequence of points, $p_i = (x, y, t)$ (spatial coordinate + time instant).



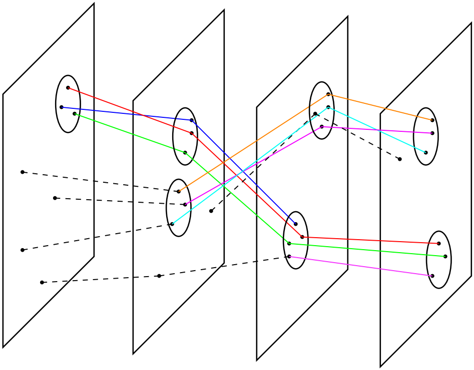
(Shoval, 2017)



(Gudmundsson, et al. 2008)

- i.e. convoys, moving clusters, swarms, gatherings, **flocks**, ...

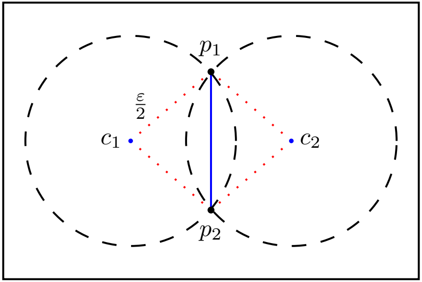
Flocks



- ε: Diameter of the circle which contains all the objects.
- μ: Minimum number of objects.
- δ: Minimum time interval the objects travel ‘together’.

Basic Flock Evaluation algorithm

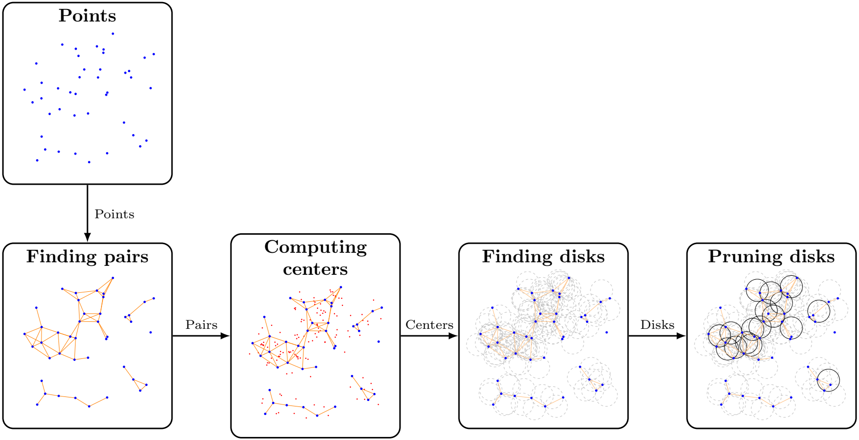
- Vieira, et al. 2009.
- The first polynomial-time solution for determining disk locations.
- Under fixed time duration it has polynomial time complexity $O(\delta|\tau|^{(2\delta)+1})$



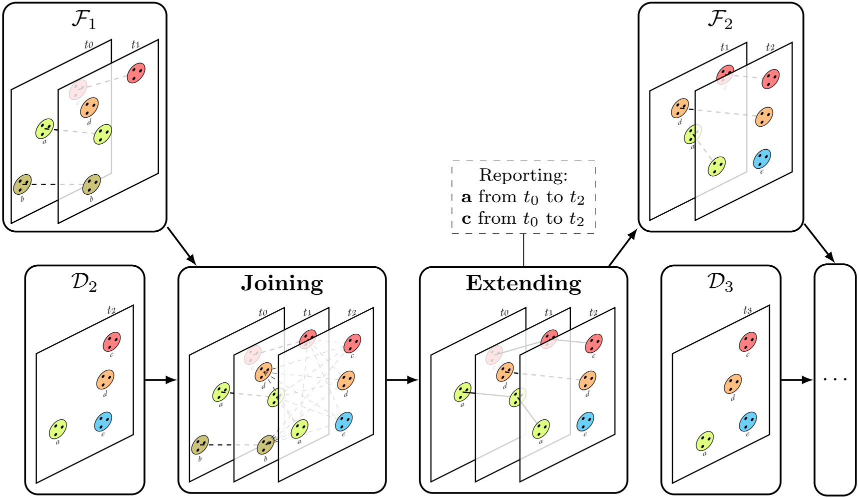
Basic Flock Evaluation algorithm

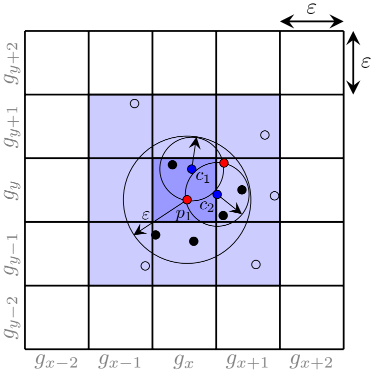
- Two main parts:
  - In the spatial domain it finds maximal disks at each time instant.
  - In the temporal domain it joins consecutive times to match set of maximal disks.

- BFE overview...

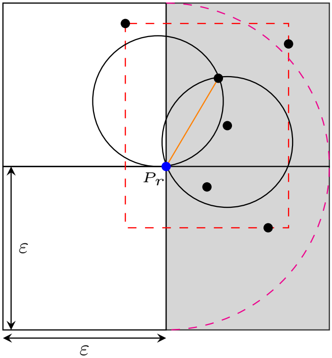


- BFE overview...





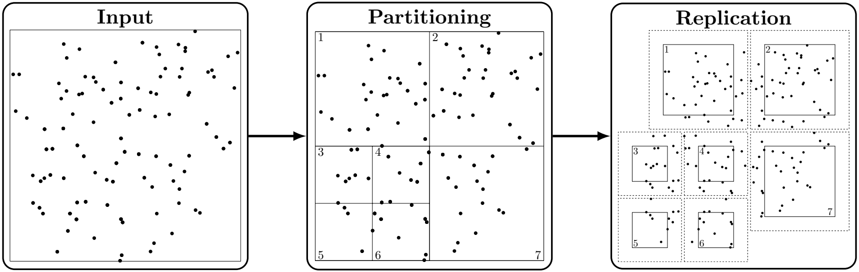
(Vieira, et al. 2009)



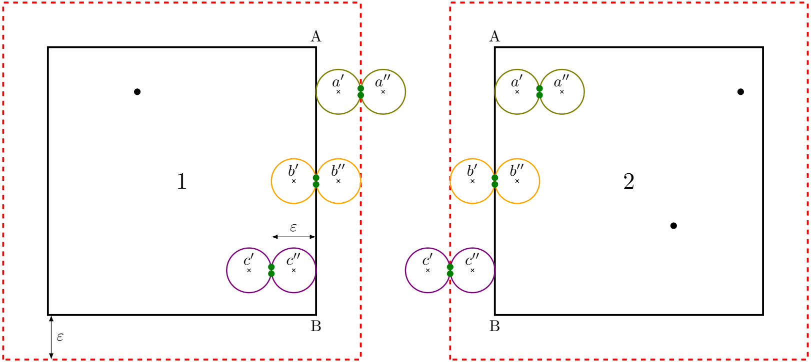
(Tanaka, et al. 2016)

- High complexity limits scalability.
- Large datasets with dense clusters of moving entities per time instant significantly impact performance.
- Specifically,
    - identifying maximal disks is hindered by the extensive number of candidates requiring pruning.
    - when parallelizing, we must address moving flocks that traverse contiguous partitions.
- We propose a parallel and scalable solution for both spatial and temporal domains.

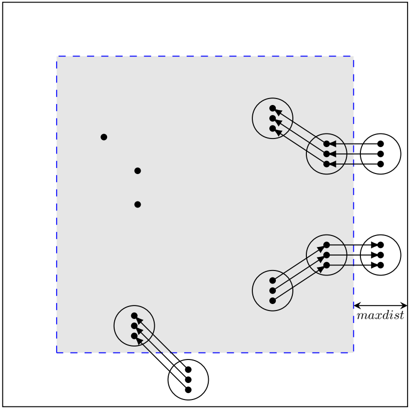
- Partitioning strategy...



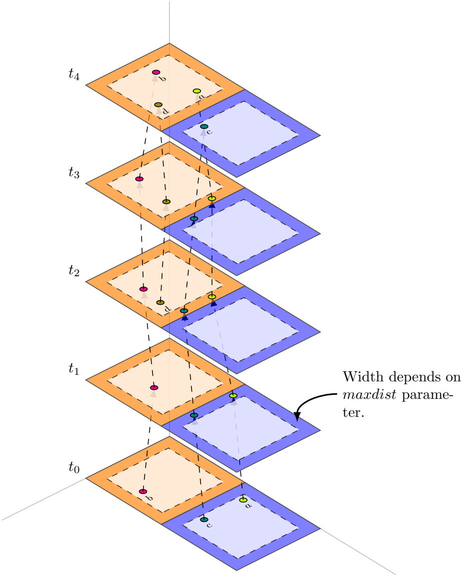
- Handling duplication...



- We introduce the *maxdist* parameter to define an area were we have to track **crossing partial flocks** (CPFs)...



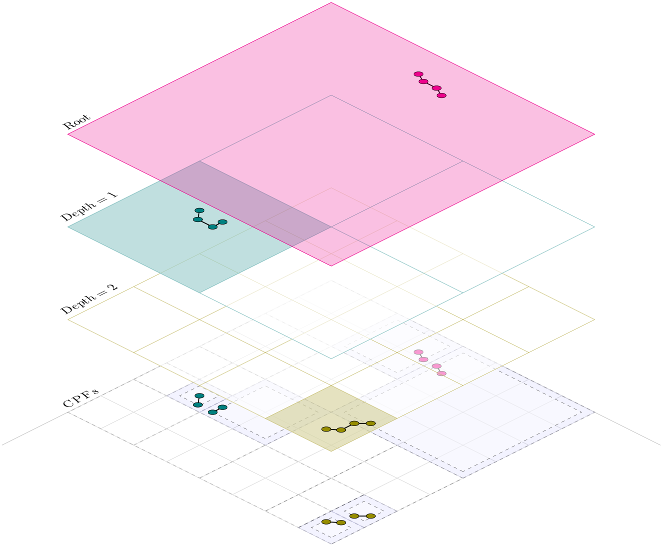
On the temporal domain

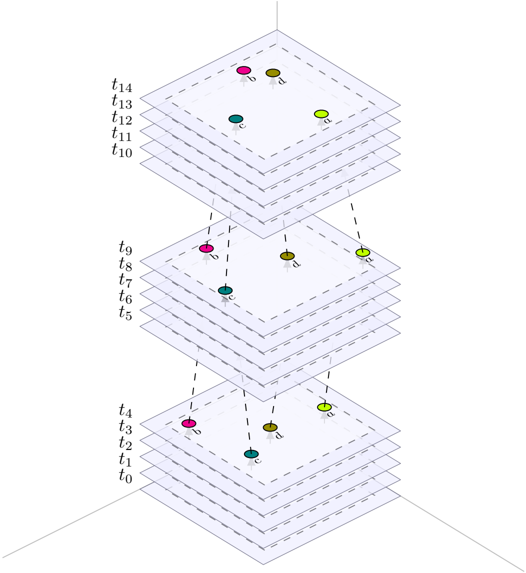


*a,b,c and d are flocks moving along time.

- Discovered flocks inside the safe area are ready to be reported.
- CPFs require post-processing. We propose four alternative:
  - Master
  - By-Level
  - Least Common Ancestor (LCA)
  - Cube-based

On the temporal domain



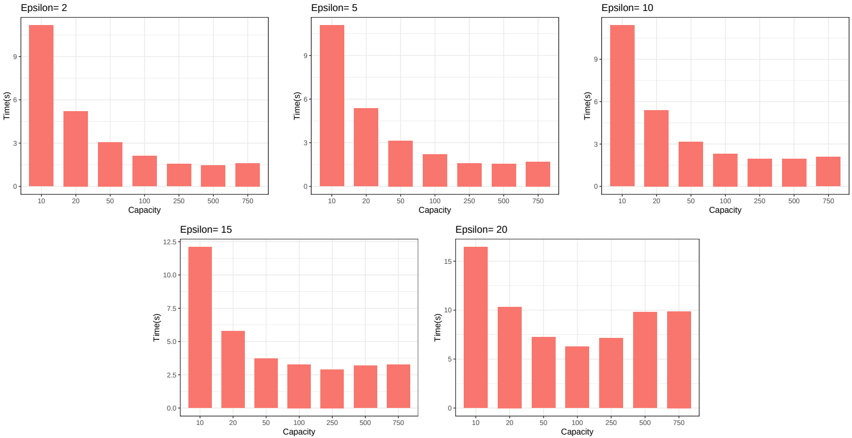


*a,b,c and d are flocks moving along time.

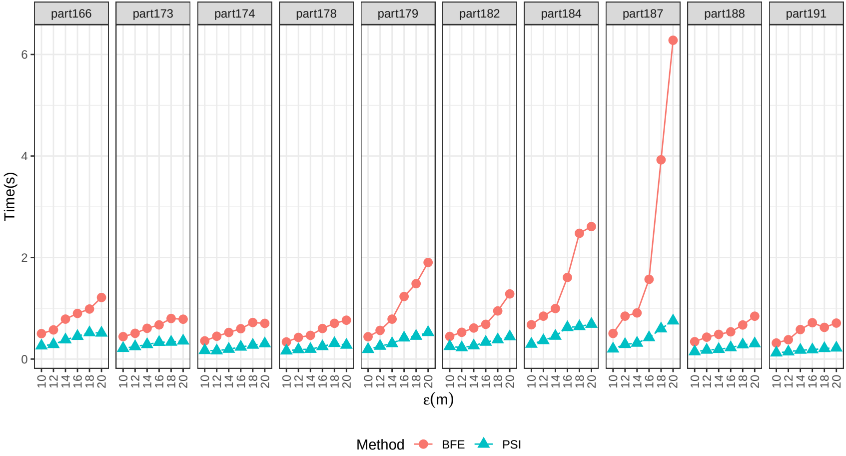
Dataset	Number of Trajectories	Total number of points	Maximum Duration (min)
Berlin10K	10000	97526	10
LA25K	25000	1495637	30
LA50K	50000	2993517	60

Experiments

- Optimizing the number of partitions for Phase 1.

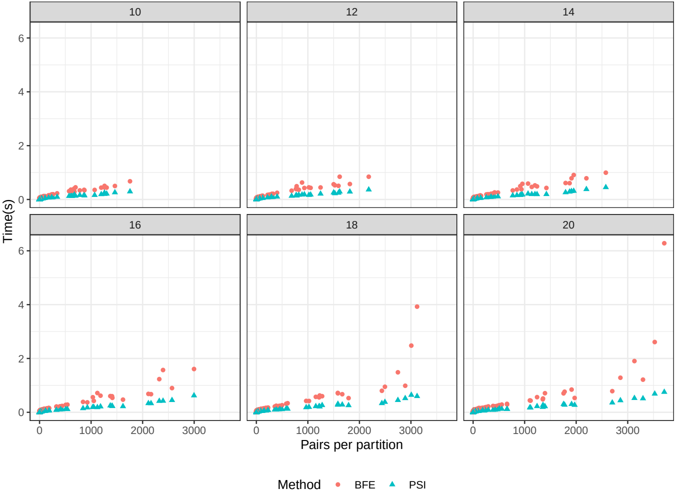


- Analyzing most costly partitions.
  - Top 10 most costly partitions.

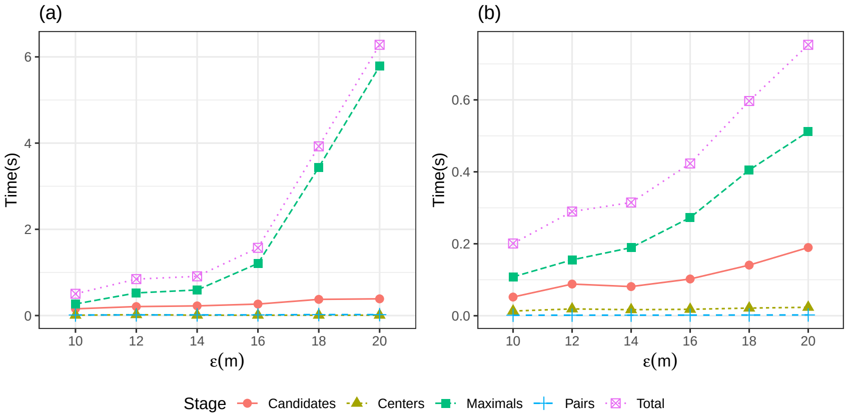


Experiments

- Analyzing most costly partitions.
  - By Pairs density..



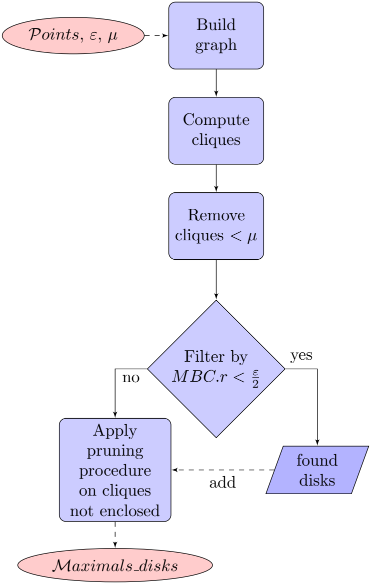
- Analyzing most costly partitions.
  - By Stages in the most costly partition [(a) BFE (b) PSI].



Can we reduce pruning time?

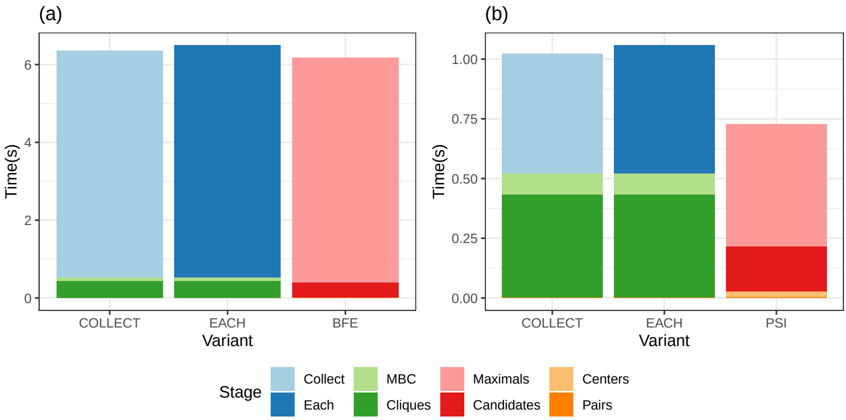
- **Maximal clique (MC)**: Given an undirected graph, a MC is a subset of vertices, each directly connected to every other in the subset, that cannot be expanded by adding additional vertices.
- **Minimum Bounding Circle (MBC)**: Given a set of points in Euclidean space, the MBC is the smallest circle that can enclose all the points.

Can we reduce pruning time?

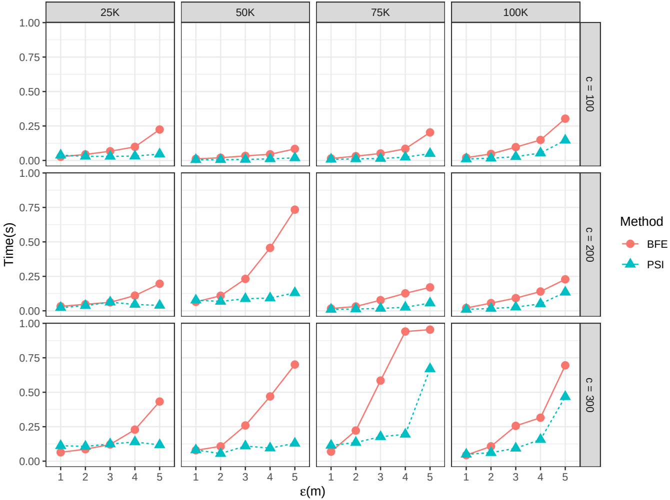


Can we reduce pruning time?

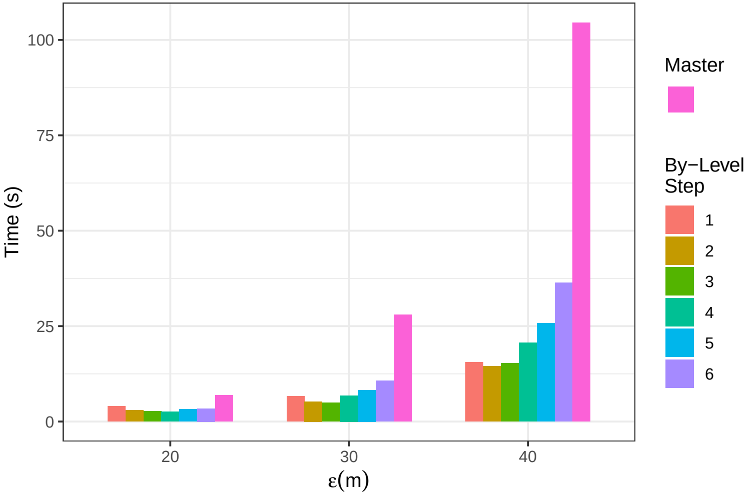
- Phase 1 variants performance [(a) vs BFE (b) vs PSI].



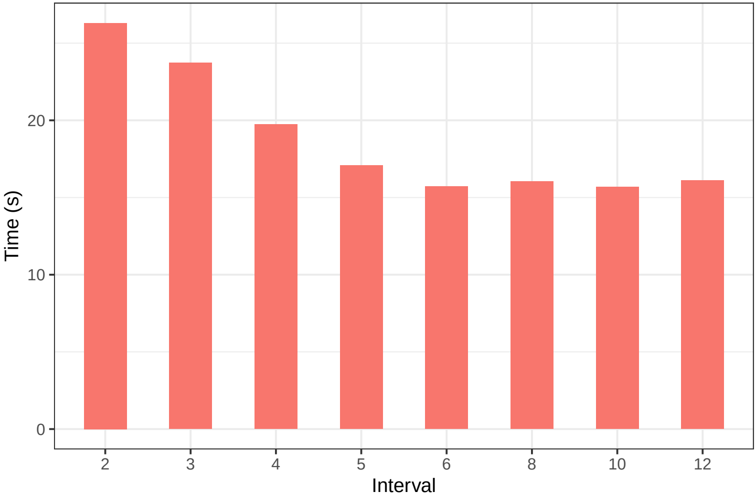
- Relative performance of Phase 1 using synthetic datasets.



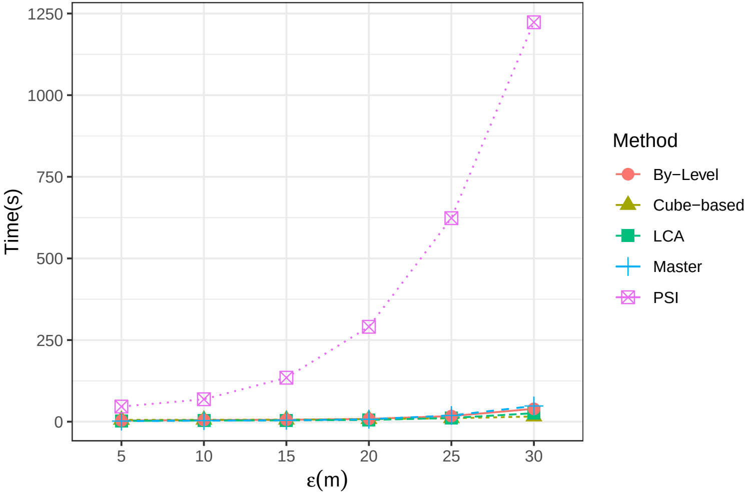
- Finding best *step* value for By-Level alternative.



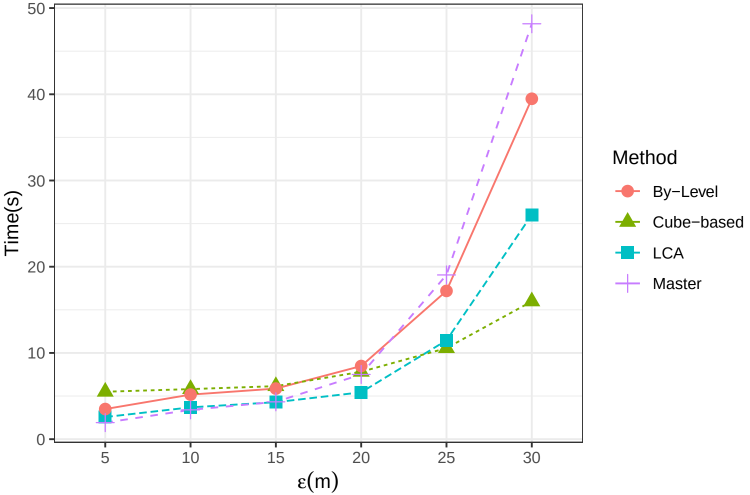
- Finding best *interval* value for Cube-based alternative.



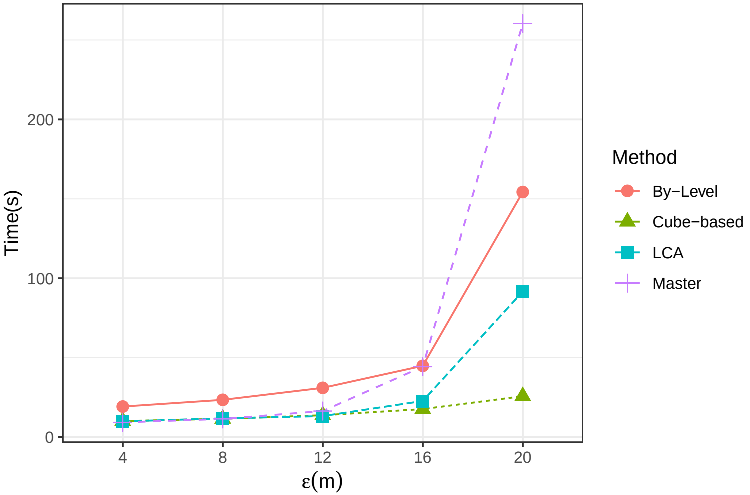
- Scalable alternatives vs standard PSI.



- Scalable alternatives in LA25K dataset.



- Scalable alternatives in LA50K dataset.



Thank you!