Multiway Simple Cycle Separators and I/O-Efficient Algorithms for Planar Graphs

Lars Arge
MADALGO
Aarhus University
Denmark

Freek van Walderveen
MADALGO
Aarhus University
Denmark

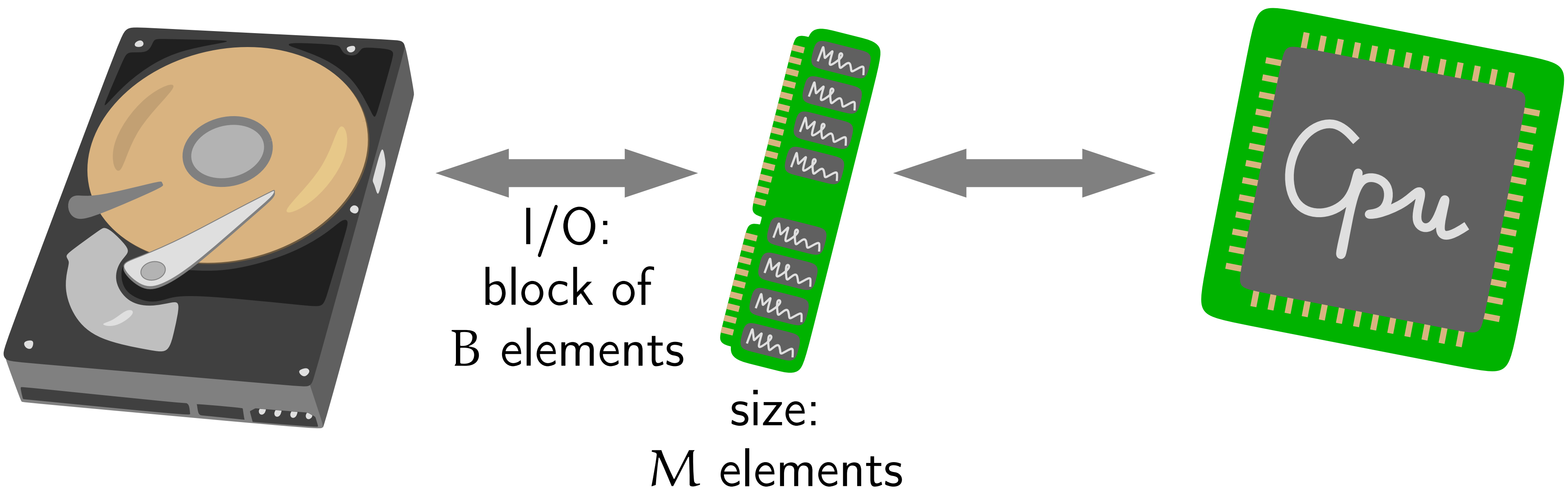
Norbert Zeh
Dalhousie University
Canada

Danmarks Grundforskningsfond
Danish National Research Foundation

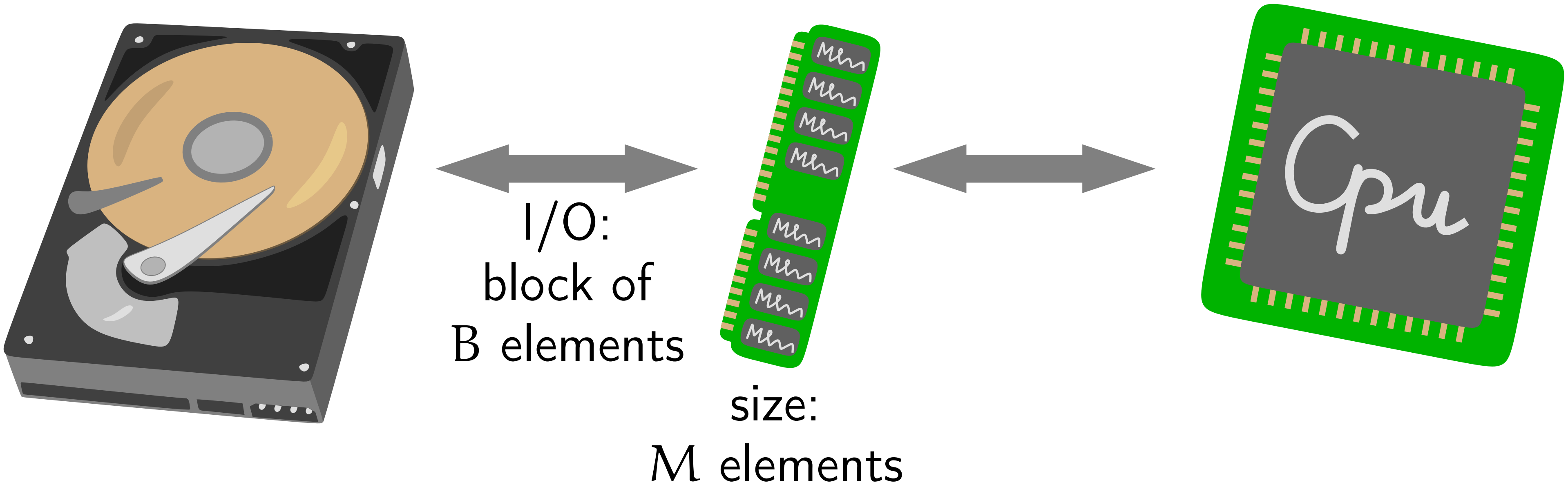
Outline

- I/O-efficient planar graph algorithms
- Definition multiway simple cycle separator
- Internal-memory construction
- Summary

I/O-efficient algorithms

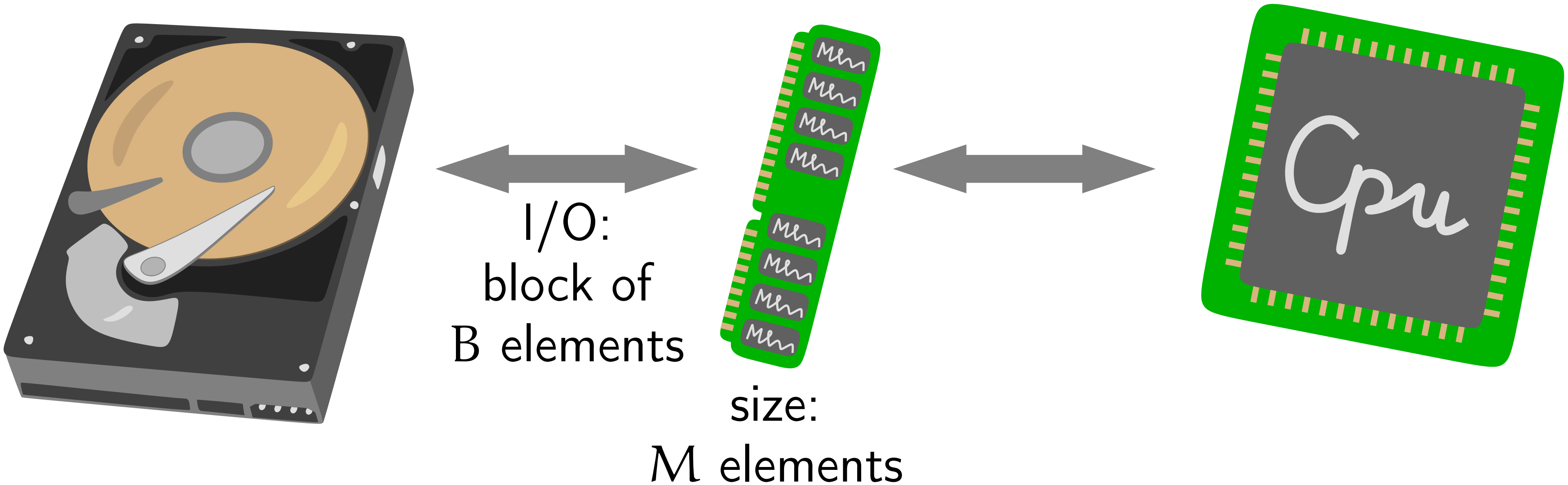


I/O-efficient algorithms



I/O model: analyze number of I/Os between internal and external memory

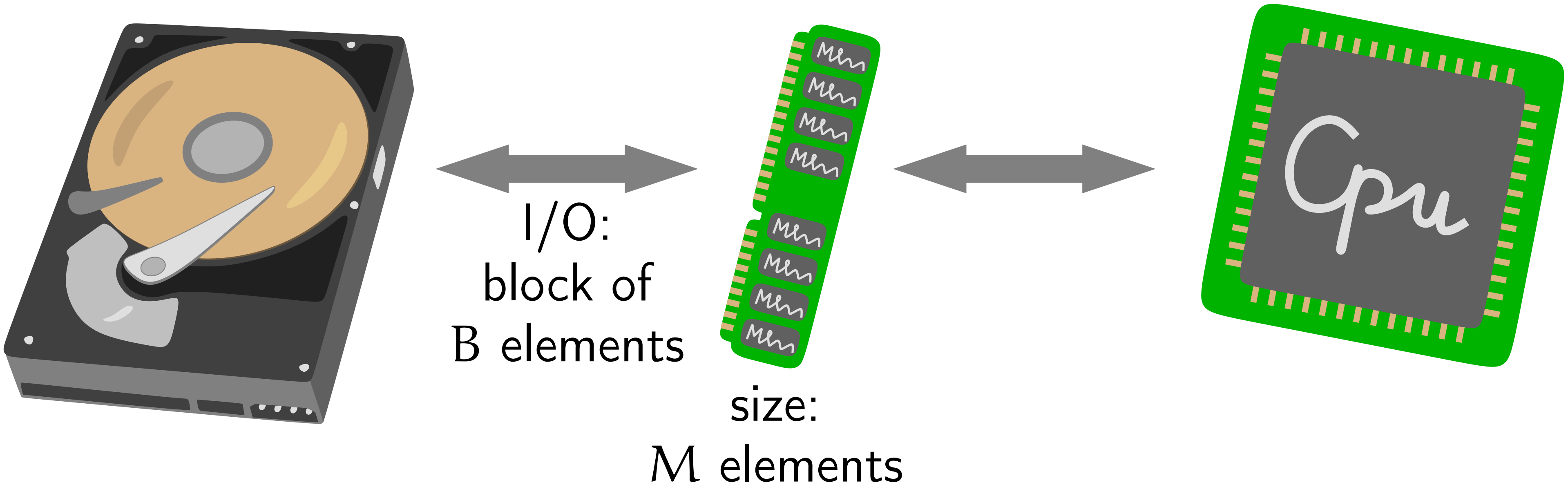
I/O-efficient algorithms



I/O model: analyze number of I/Os between internal and external memory

- Scanning N elements: $\Theta(N/B)$ I/Os

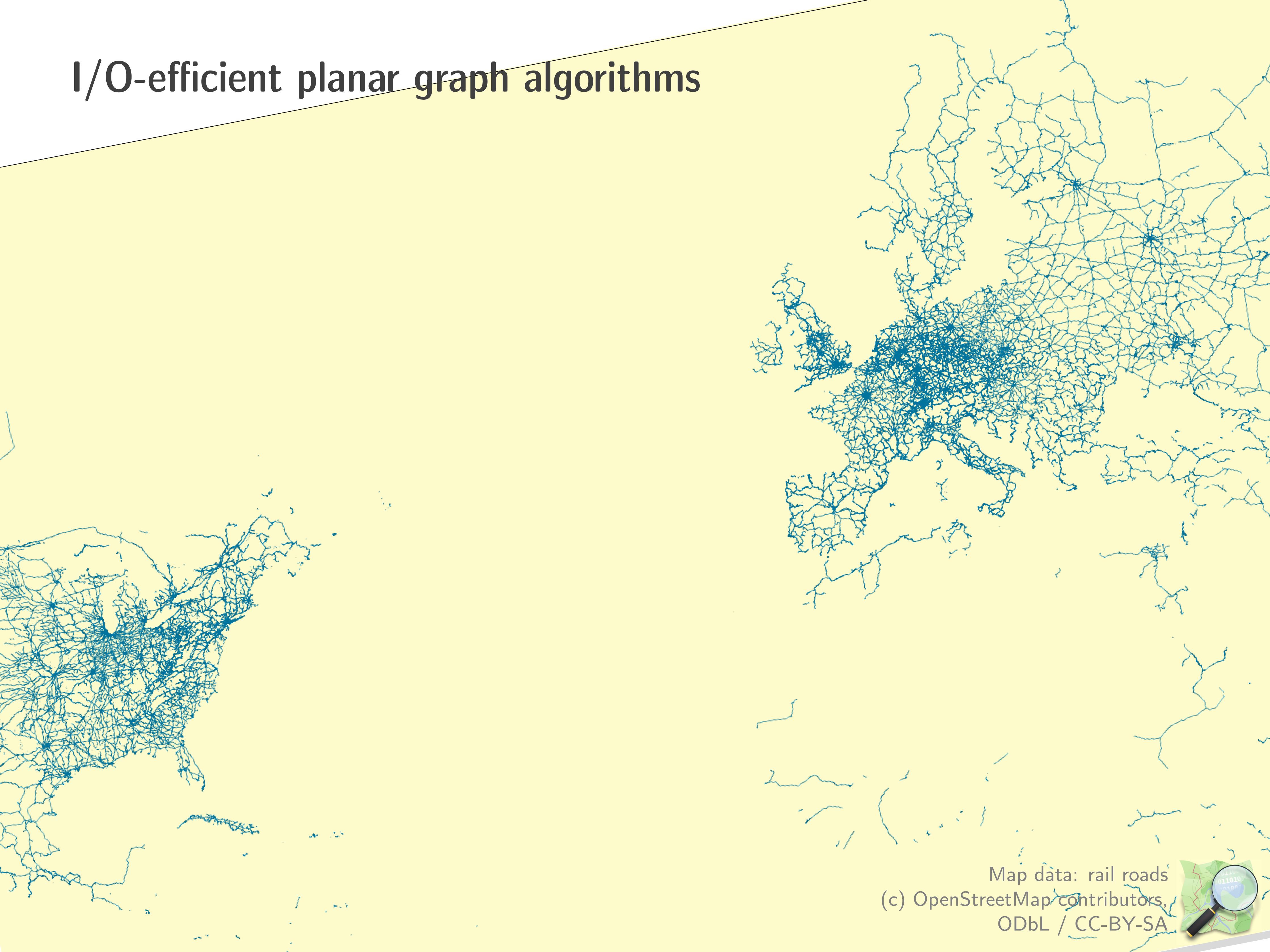
I/O-efficient algorithms



I/O model: analyze number of I/Os between internal and external memory

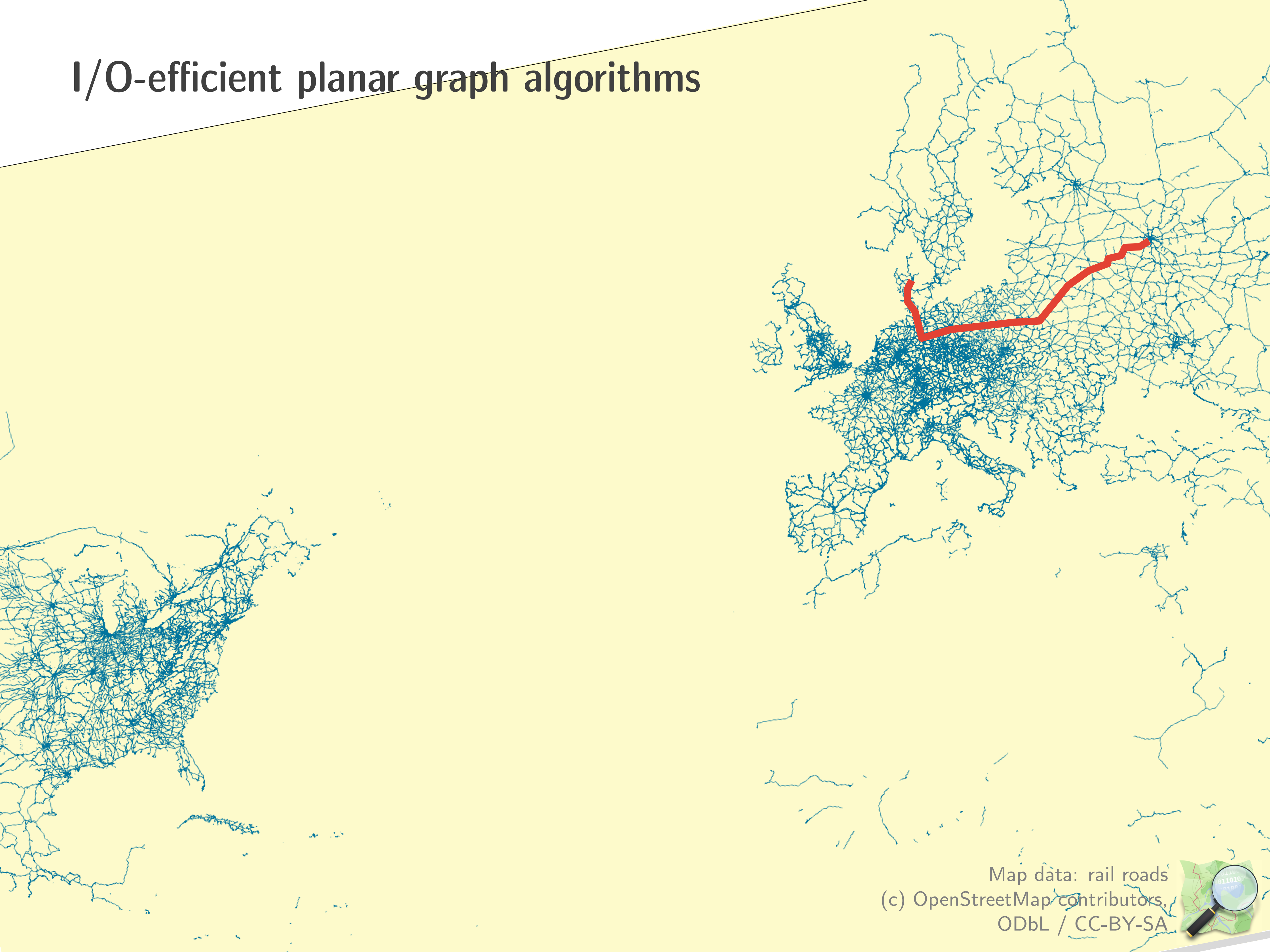
- Scanning N elements: $\Theta(N/B)$ I/Os
- Sorting N elements: $\Theta(\mathtt{sort}(N)) = \Theta(\frac{N}{B}\,\mathtt{log}_{M/B}\frac{N}{B})$ I/Os

I/O-efficient planar graph algorithms



Map data: rail roads
(c) OpenStreetMap contributors,
ODbL / CC-BY-SA

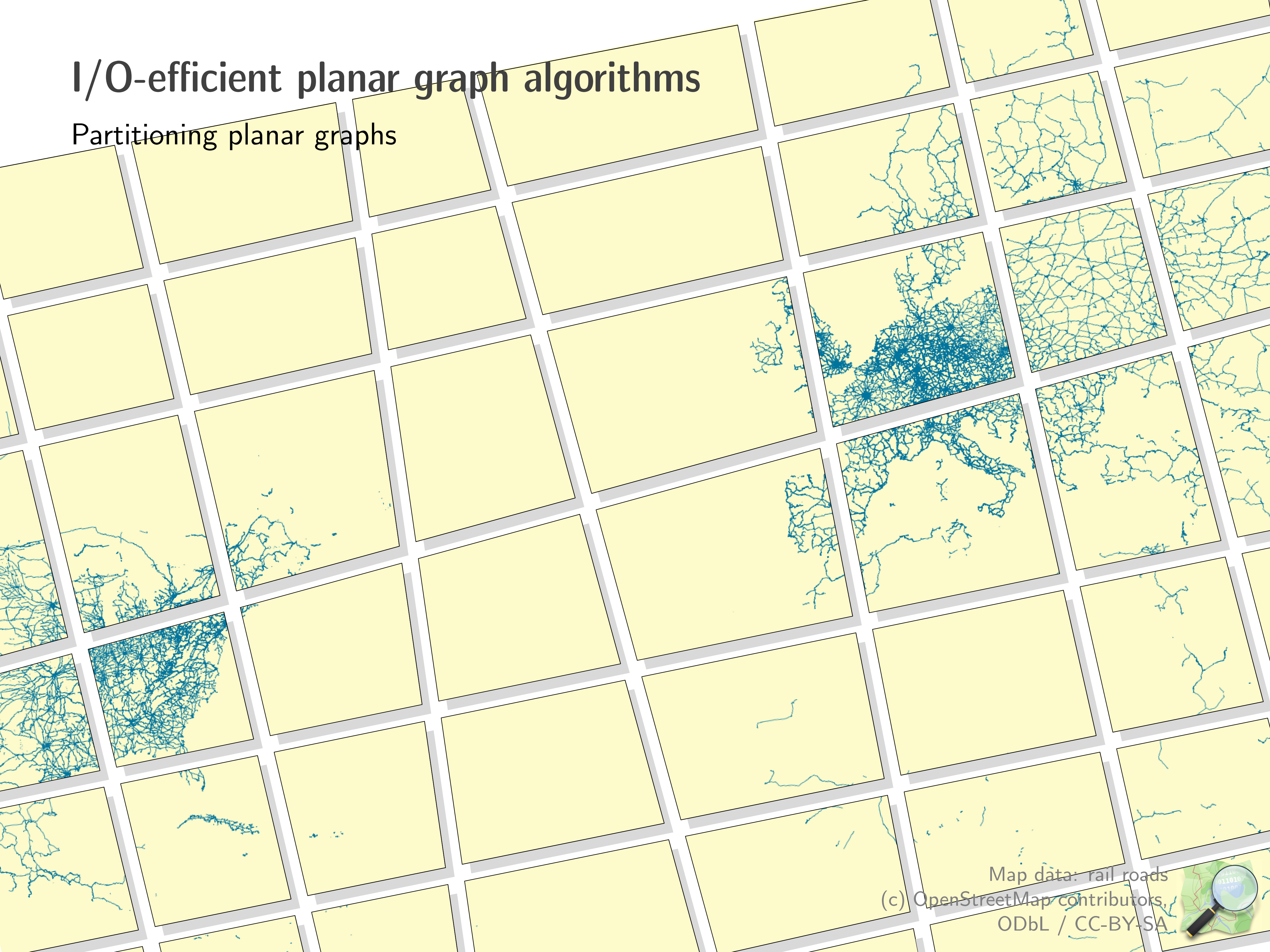
I/O-efficient planar graph algorithms



Map data: rail roads
(c) OpenStreetMap contributors,
ODbL / CC-BY-SA

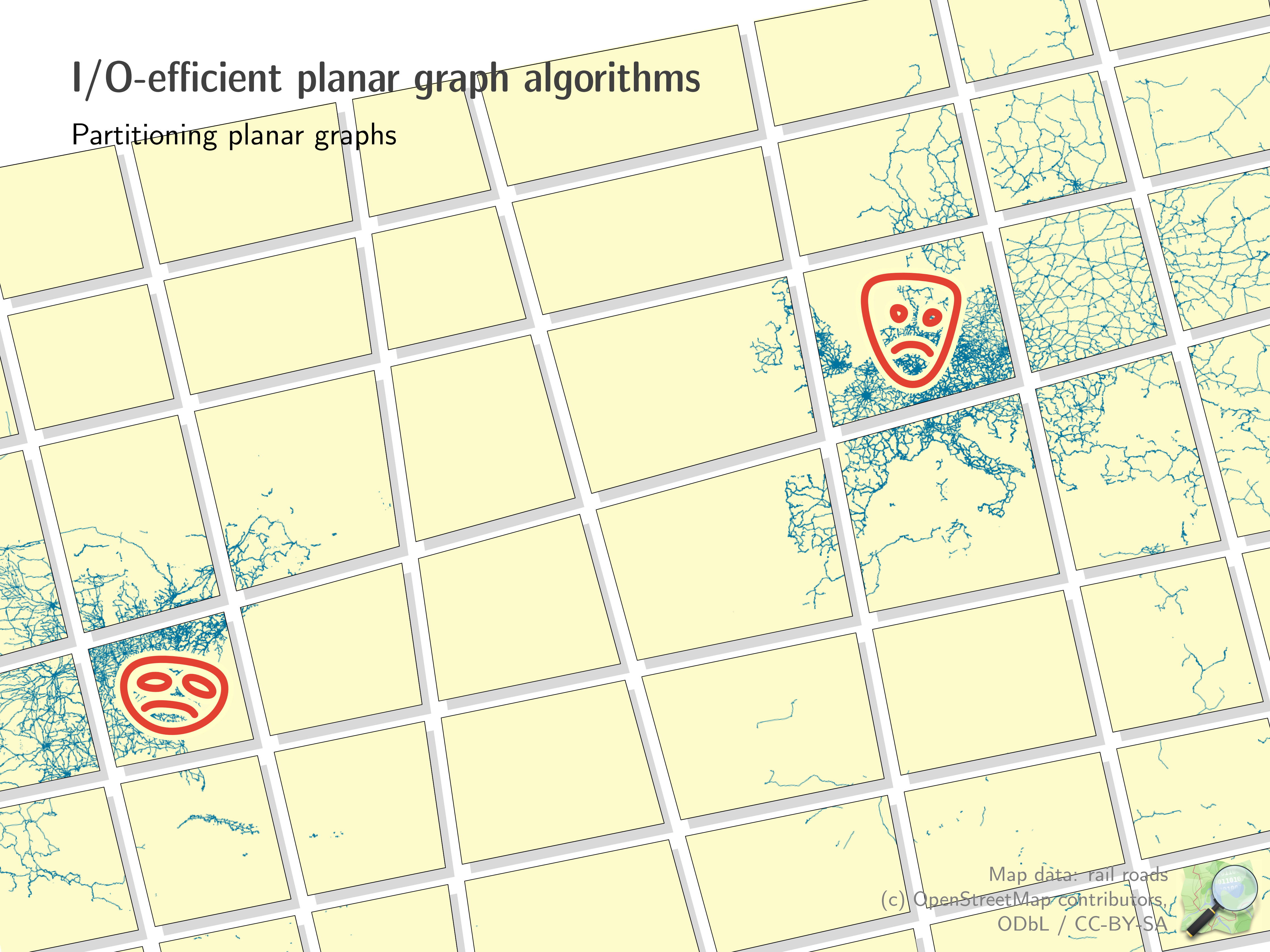
I/O-efficient planar graph algorithms

Partitioning planar graphs



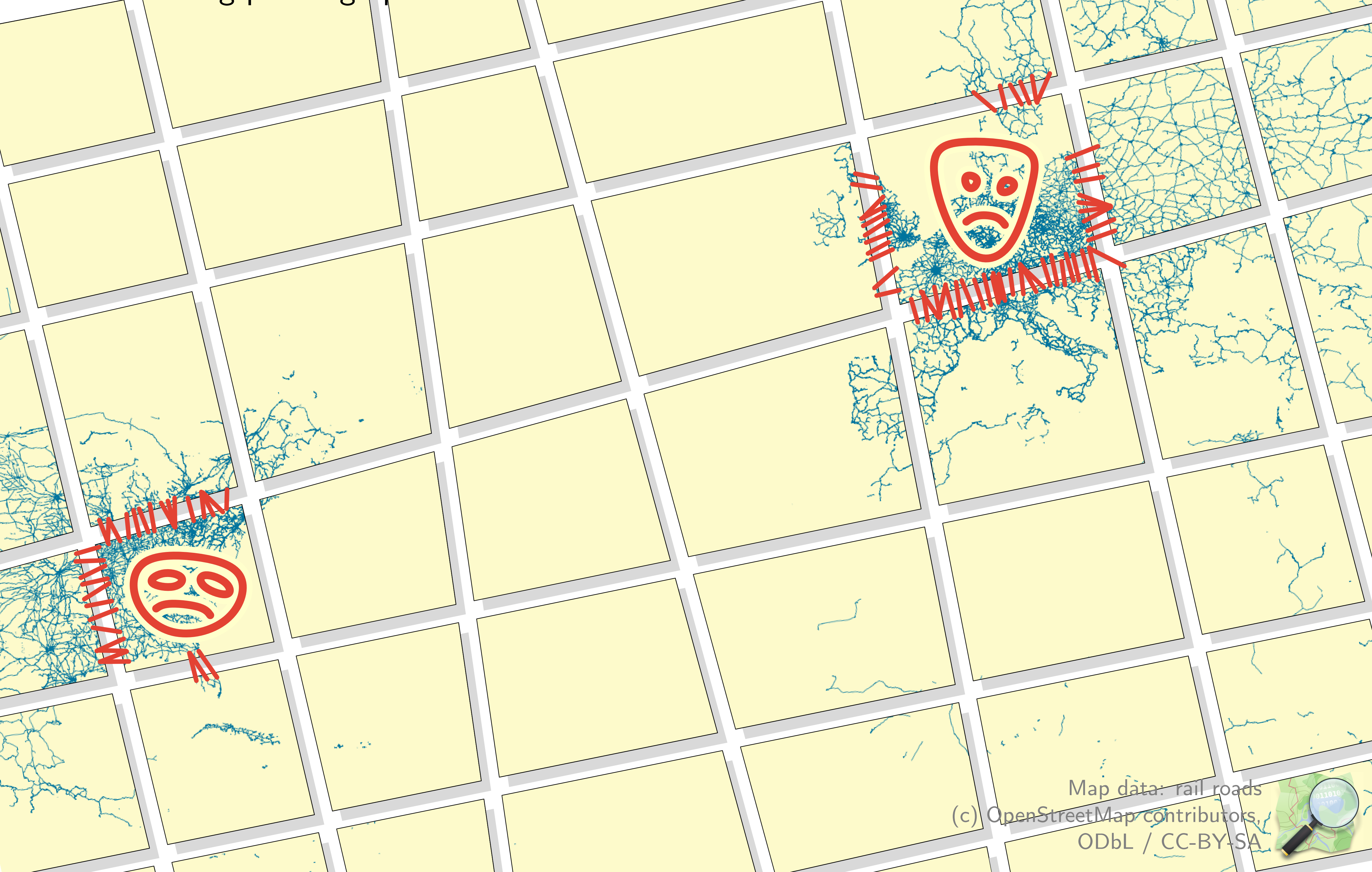
I/O-efficient planar graph algorithms

Partitioning planar graphs



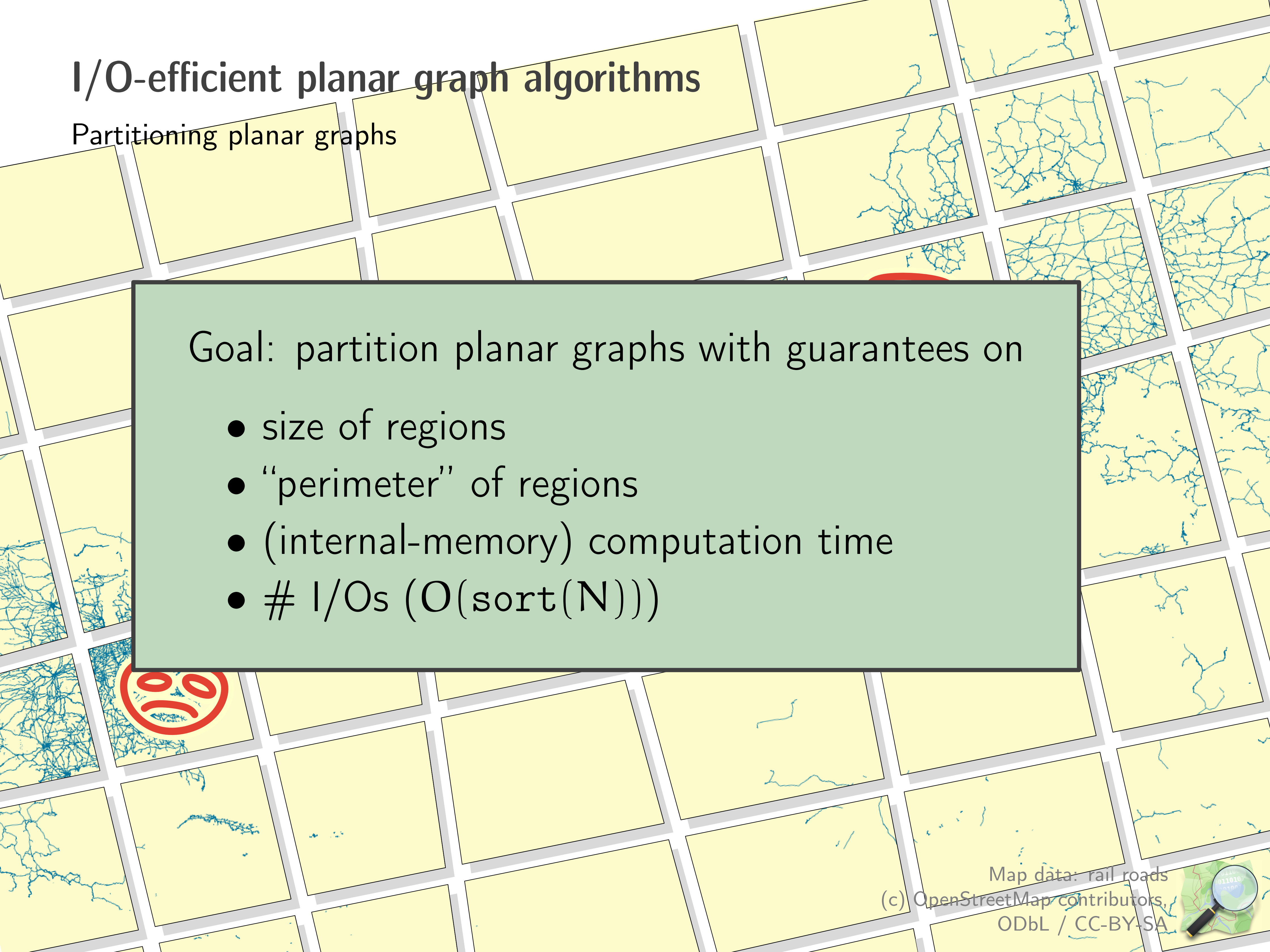
I/O-efficient planar graph algorithms

Partitioning planar graphs



I/O-efficient planar graph algorithms

Partitioning planar graphs

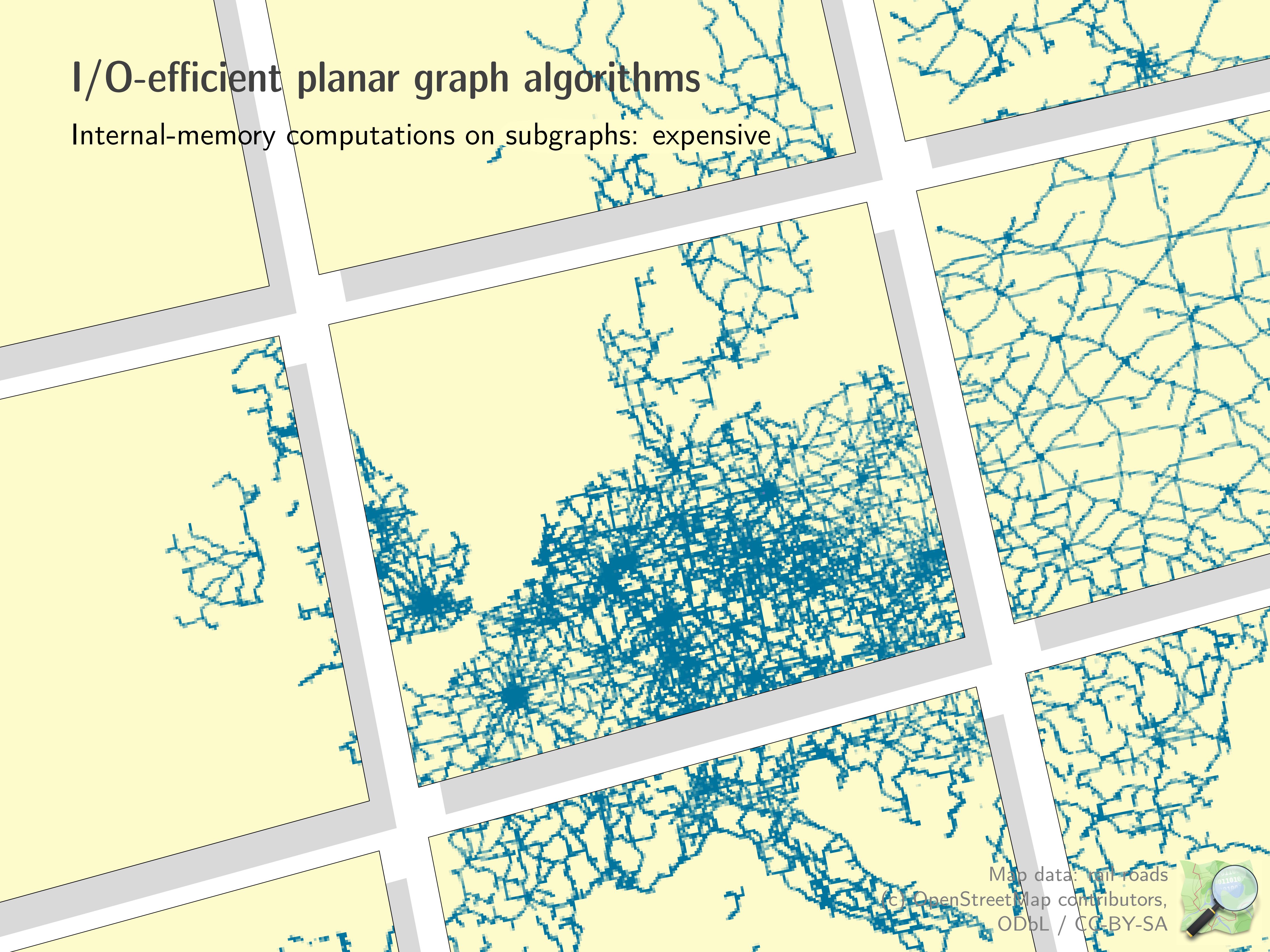


Goal: partition planar graphs with guarantees on

- size of regions
- "perimeter" of regions
- (internal-memory) computation time
- # I/Os ($O(\mathtt{sort}(N))$)

I/O-efficient planar graph algorithms

Internal-memory computations on subgraphs: expensive

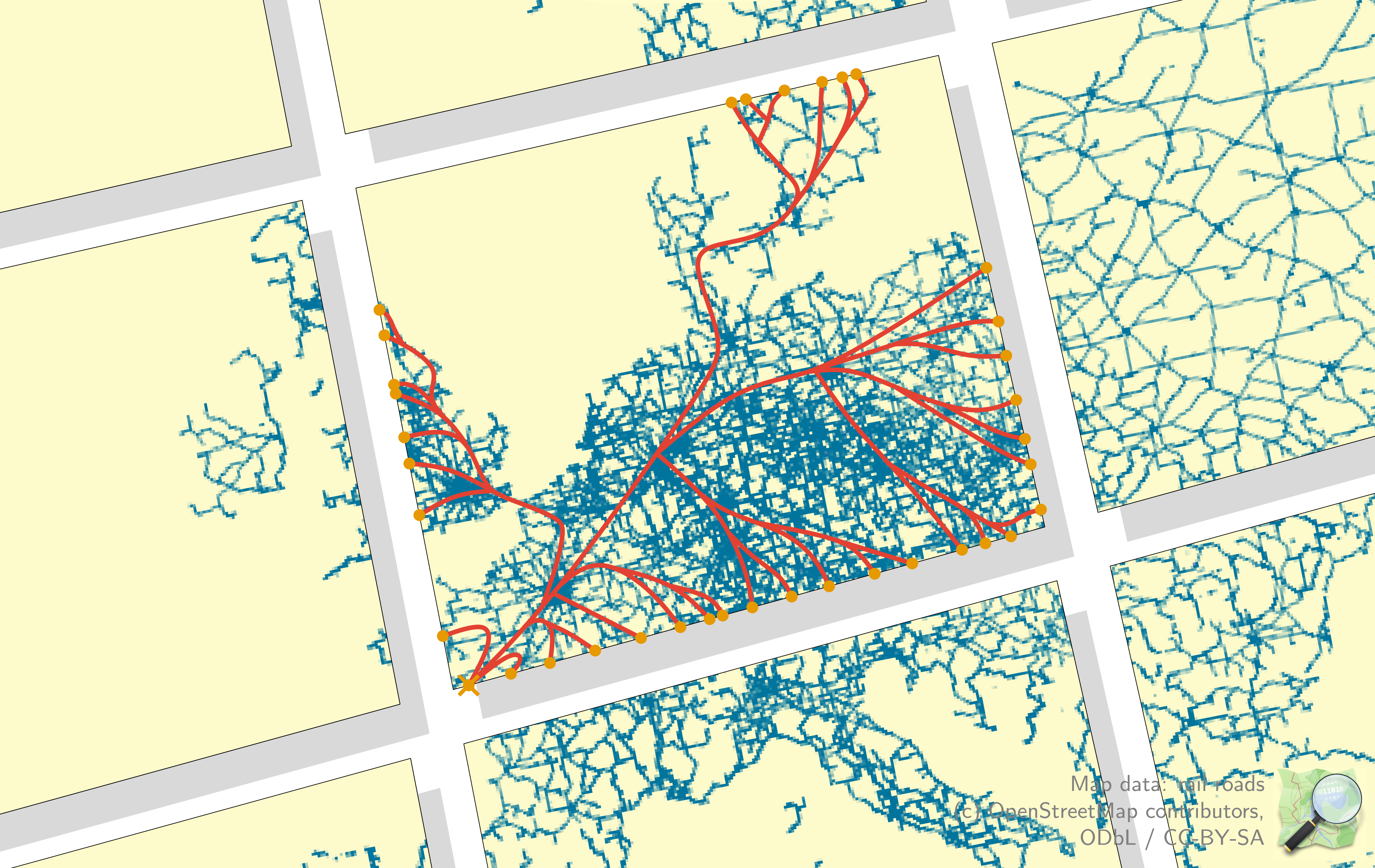


I/O-efficient planar graph algorithms

Internal-memory computations on subgraphs: expensive

I/O-efficient planar graph algorithms

Internal-memory computations on subgraphs: expensive

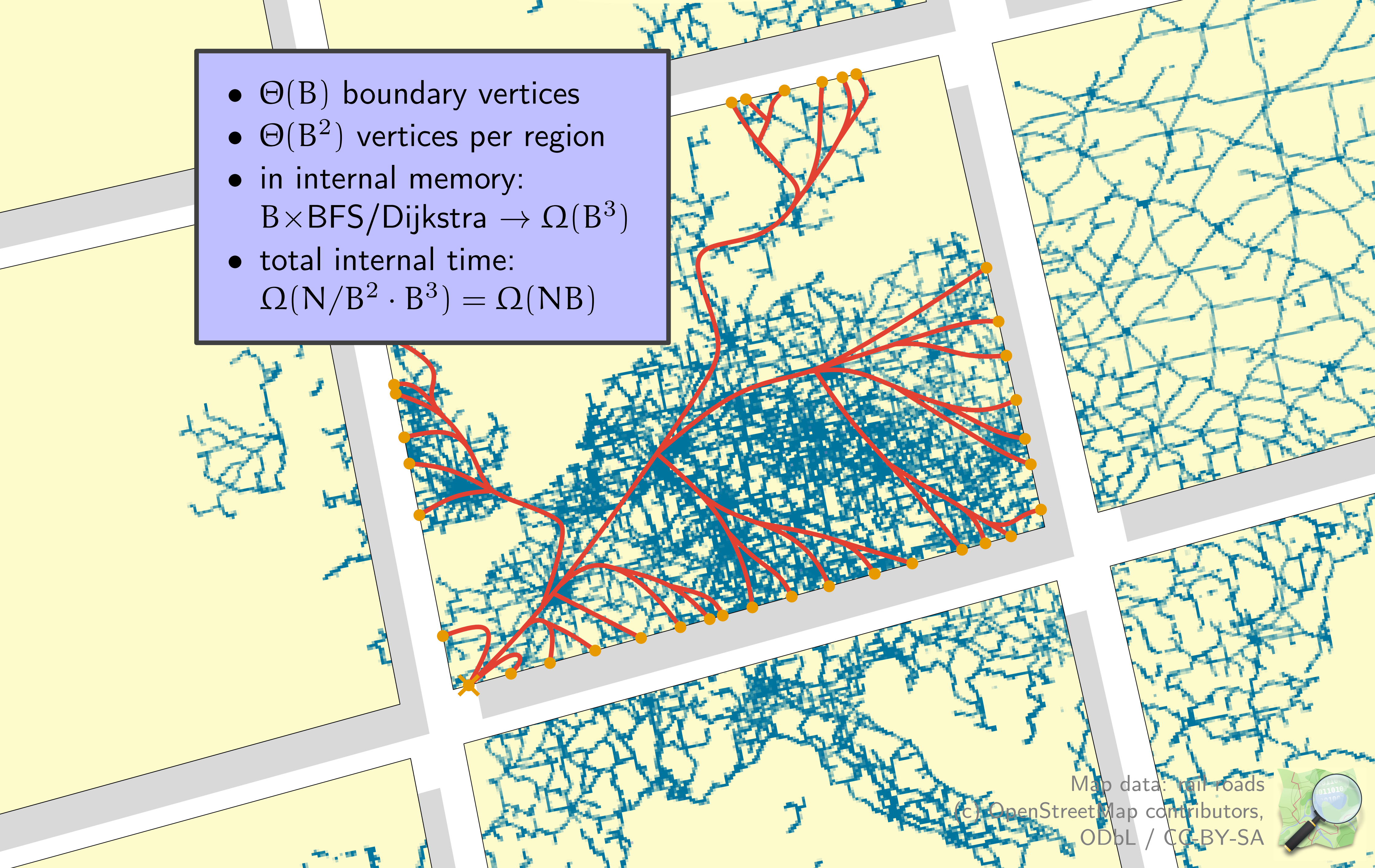


Map data: rail roads
(c) OpenStreetMap contributors,
ODbL / CC-BY-SA

I/O-efficient planar graph algorithms

Internal-memory computations on subgraphs: expensive

- $\Theta(B)$ boundary vertices
- $\Theta(B^2)$ vertices per region
- in internal memory:
  $B \times$BFS/Dijkstra $\rightarrow \Omega(B^3)$
- total internal time:
  $\Omega(N/B^2 \cdot B^3) = \Omega(NB)$

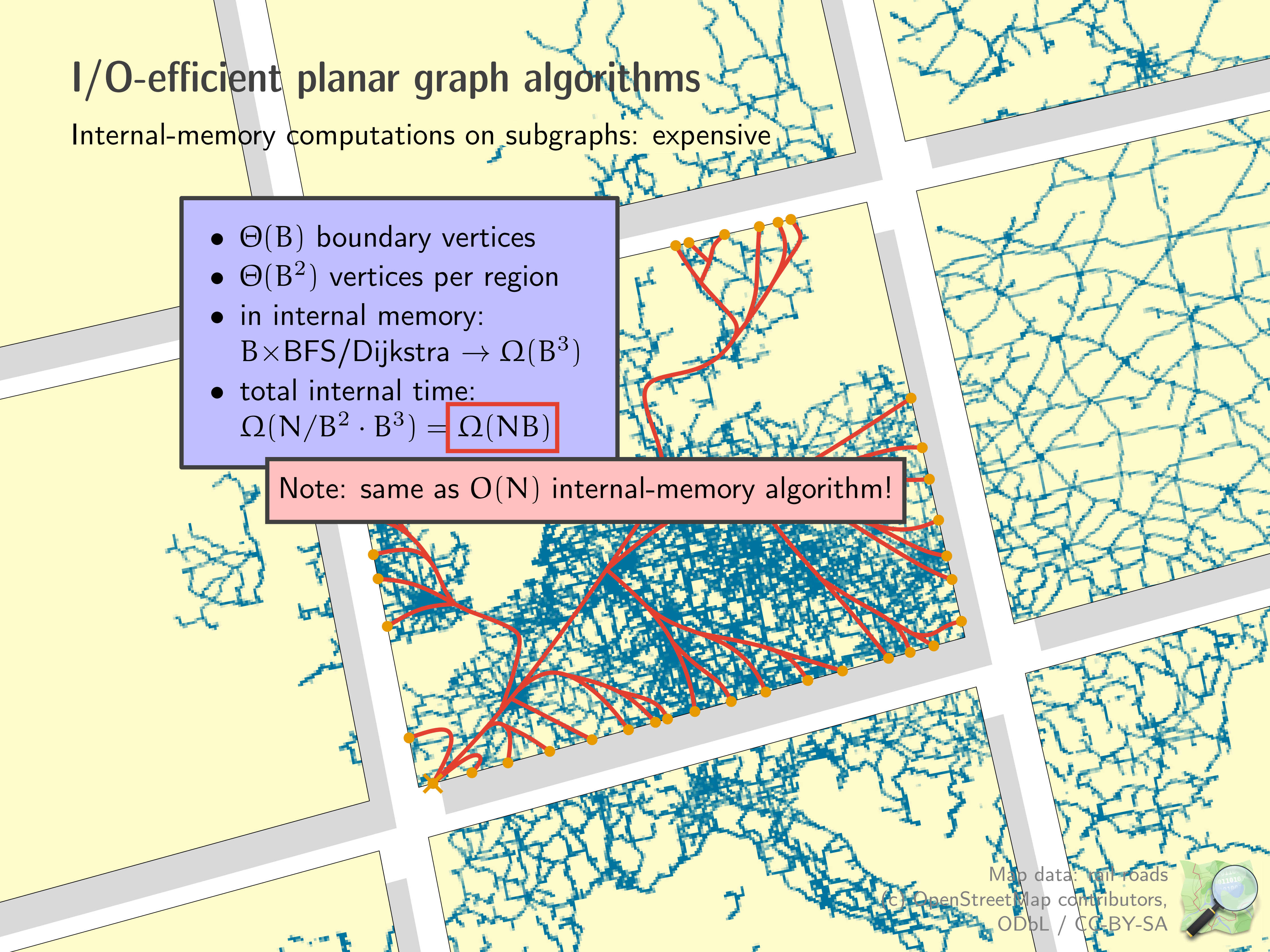


I/O-efficient planar graph algorithms

Internal-memory computations on subgraphs: expensive

- $\Theta(B)$ boundary vertices
- $\Theta(B^2)$ vertices per region
- in internal memory:
  $B \times$BFS/Dijkstra $\rightarrow \Omega(B^3)$
- total internal time:
  $\Omega(N/B^2 \cdot B^3) = \Omega(NB)$

Note: same as $O(N)$ internal-memory algorithm!



I/O-efficient planar graph algorithms

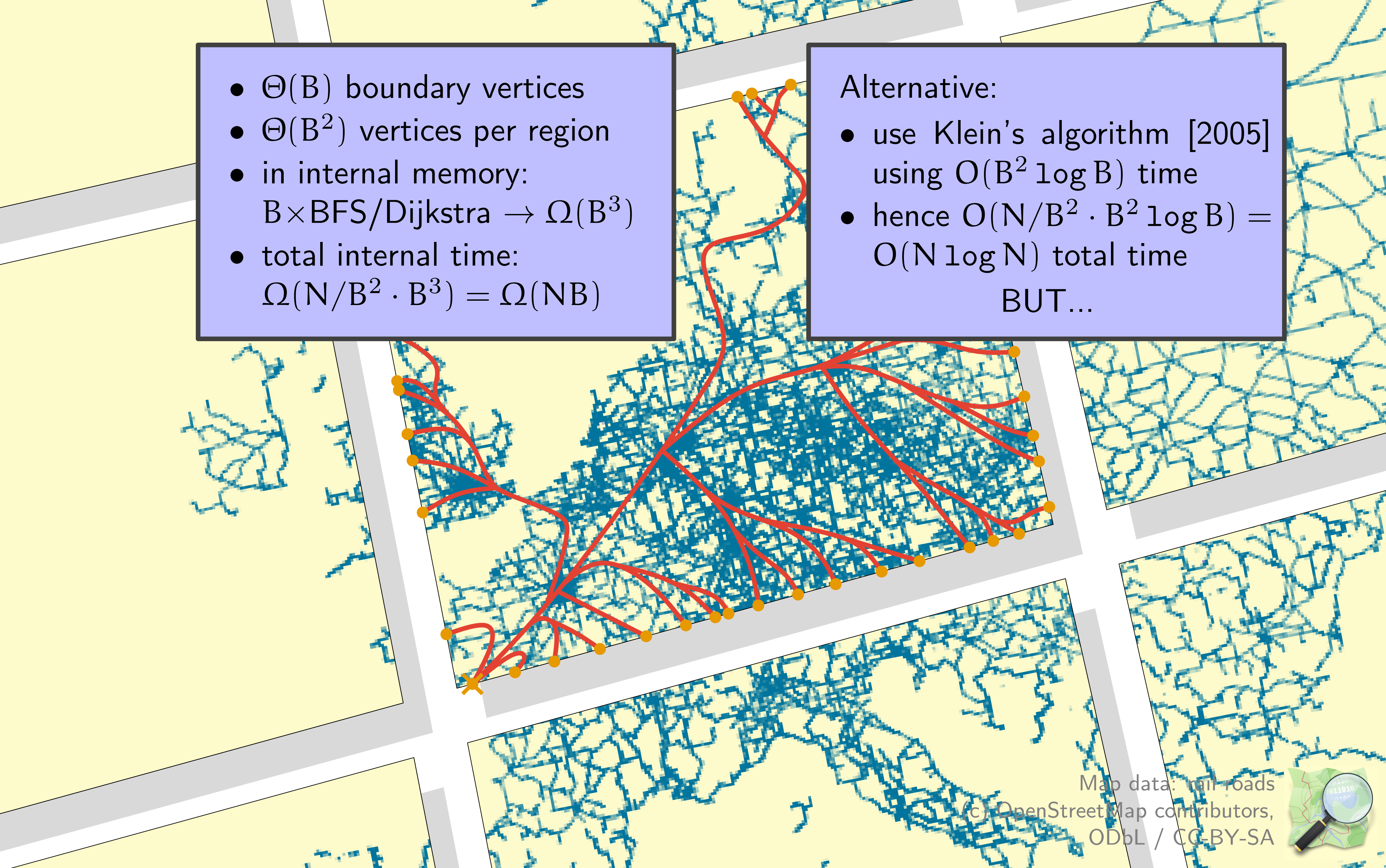
Internal-memory computations on subgraphs: expensive

- $\Theta(B)$ boundary vertices
- $\Theta(B^2)$ vertices per region
- in internal memory: $B \times$BFS/Dijkstra $\rightarrow \Omega(B^3)$
- total internal time: $\Omega(N/B^2 \cdot B^3) = \Omega(NB)$

Alternative:

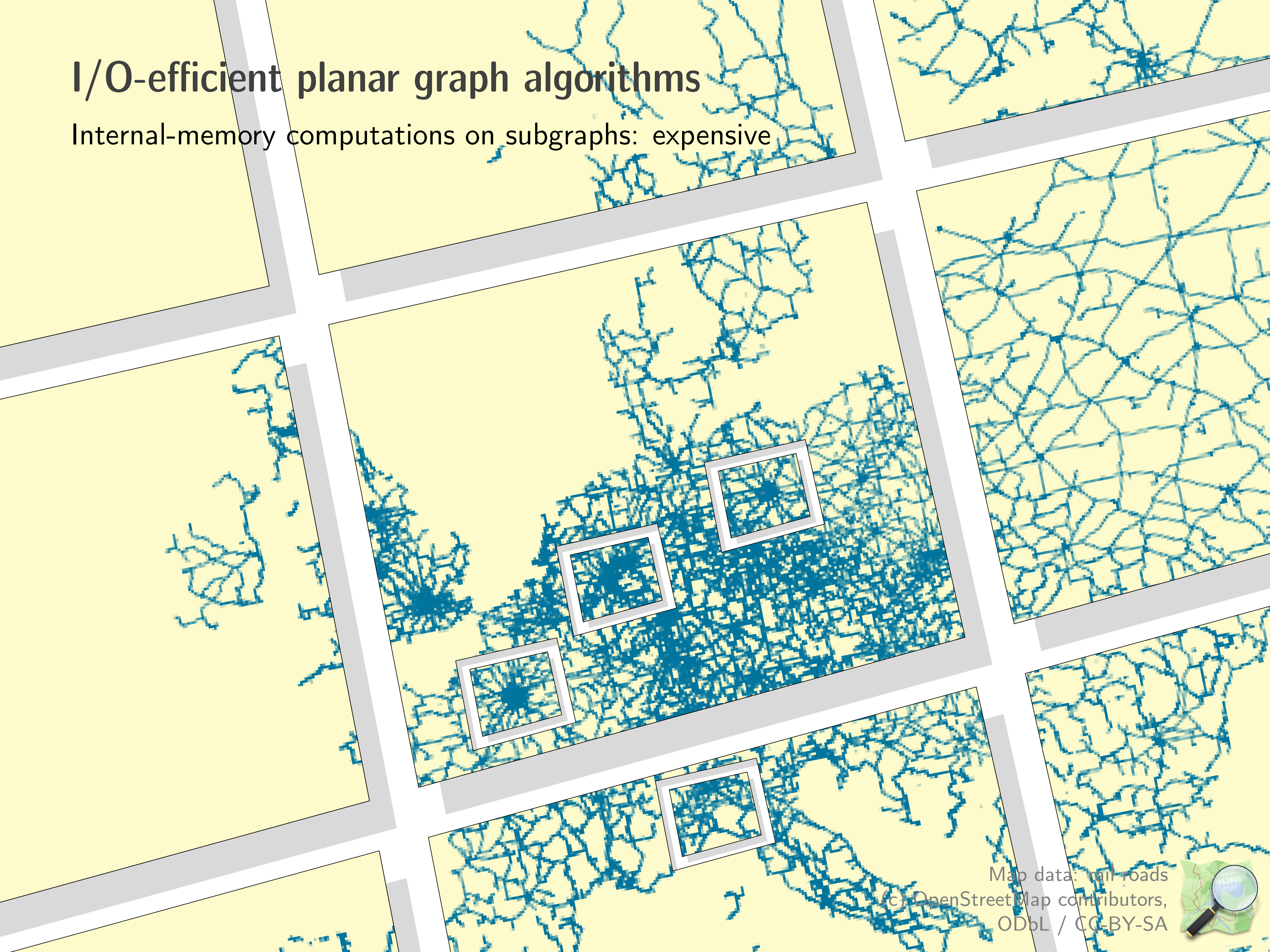
- use Klein's algorithm [2005] using $O(B^2 \log B)$ time
- hence $O(N/B^2 \cdot B^2 \log B) = O(N \log N)$ total time

BUT...



I/O-efficient planar graph algorithms

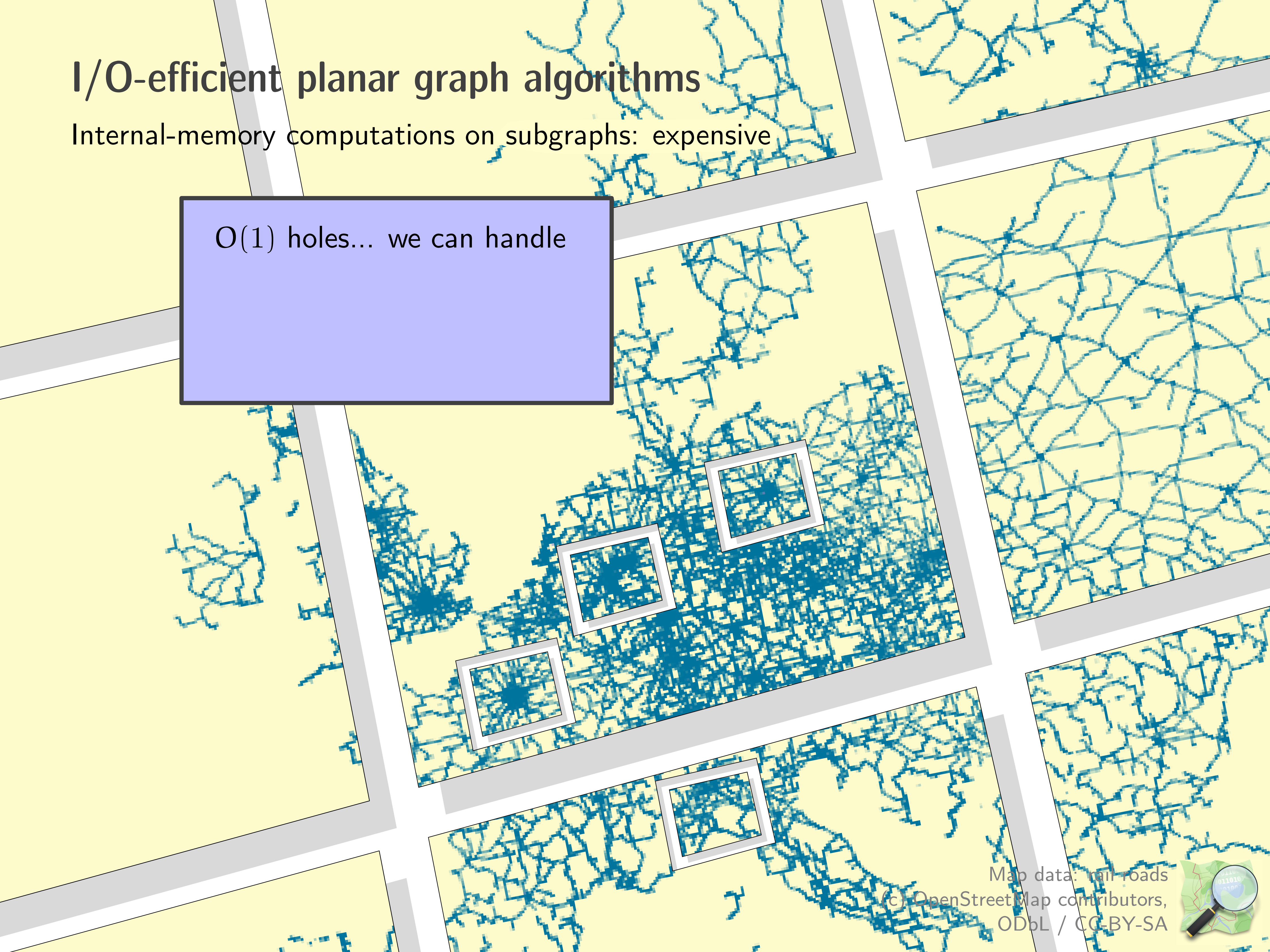
Internal-memory computations on subgraphs: expensive



I/O-efficient planar graph algorithms

Internal-memory computations on subgraphs: expensive

O(1) holes... we can handle

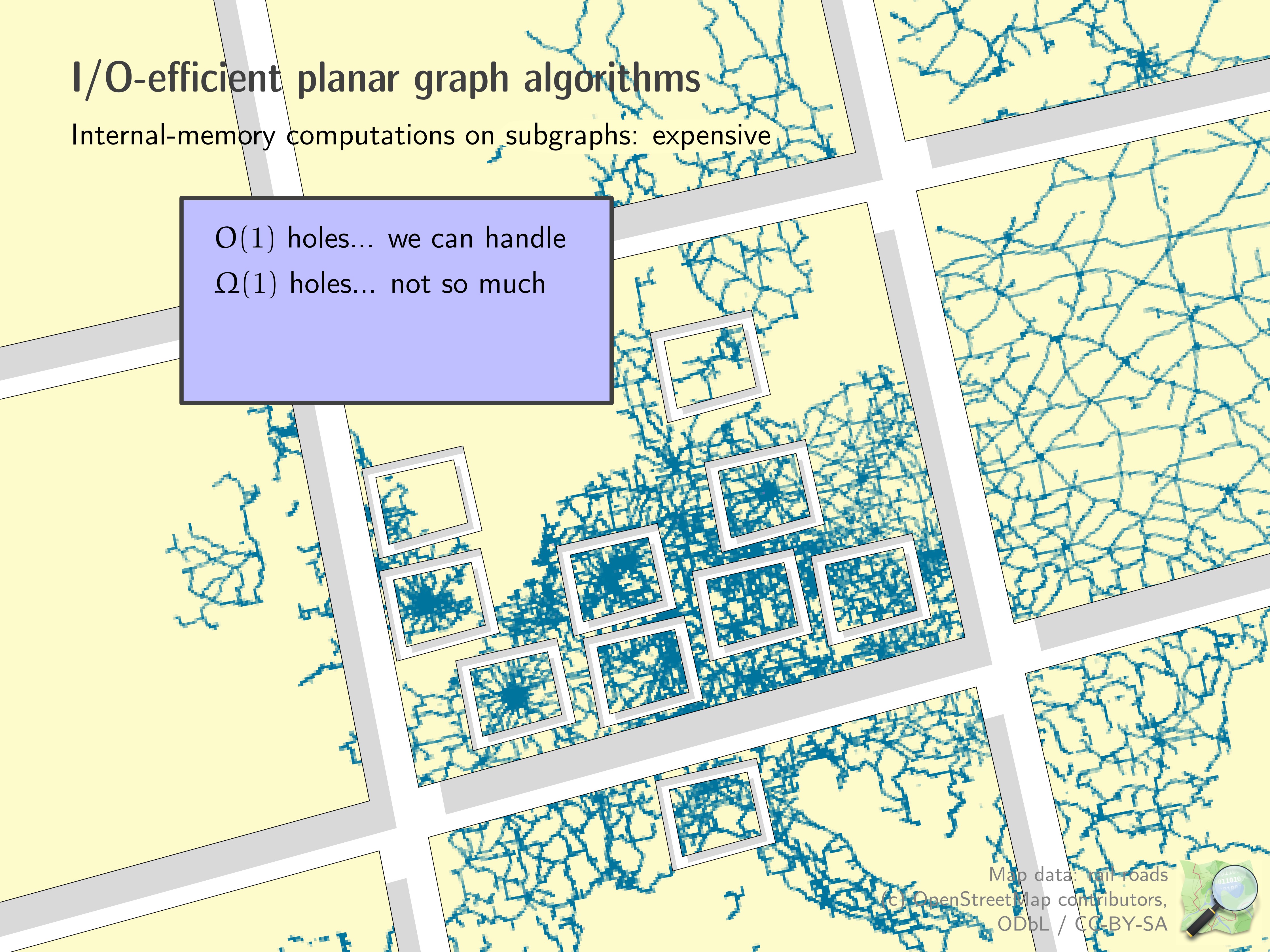


I/O-efficient planar graph algorithms

Internal-memory computations on subgraphs: expensive

$O(1)$ holes... we can handle

$\Omega(1)$ holes... not so much



I/O-efficient planar graph algorithms

Internal-memory computations on subgraphs: expensive

$O(1)$ holes... we can handle

$\Omega(1)$ holes... not so much

$\Rightarrow$ need separator with $O(1)$ holes per region

Multiway simple cycle separators: definition and previous work

Given parameter ε ($0 < \varepsilon < 1$):
multiway simple cycle separator of triangulated planar graph G of N vertices partitions G into (not necessarily connected) regions, such that:

- Number of regions $= O(1/\varepsilon)$
- Region size $= O(\varepsilon N)$
- Region boundary size $= O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Multiway simple cycle separators: definition and previous work

Given parameter ε $(0 < \varepsilon < 1)$:
multiway simple cycle separator of triangulated planar graph G of N vertices partitions G into (not necessarily connected) regions, such that:

- Number of regions $= O(1/\varepsilon)$
- Region size $= O(\varepsilon N)$
- Region boundary size $= O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Previous work:

- Italiano, Nussbaum, Sankowski, Wulff-Nilsen: Improved algorithms for min cut and max flow in undirected planar graphs, STOC'11.
  $O(N \log(\varepsilon N) + \sqrt{N/\varepsilon} \log N)$

Concurrent work:

- Klein, Mozes, Sommer: Structured recursive separator decompositions for planar graphs in linear time, arXiv:1208.2223.

Multiway cycle separators: construction

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$

Multiway cycle separators: construction

First: design $O(N)$ time internal-memory algorithm

Overview of internal-memory algorithm:

Step 1. Partition into small or low-diameter regions

Step 2. Split big (low-diameter) regions

Step 3. Limit #regions, boundary sizes, and #holes per region

Second: make I/O-efficient, i.e. $O(\texttt{sort}(N))$ I/Os, $O(N \log N)$ time
(Use bootstrapping with SSSP.)

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$

Multiway cycle separators: construction

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$

First: design $O(N)$ time internal-memory algorithm

Overview of internal-memory algorithm:

Step 1. Partition into small or low-diameter regions

Step 2. Split big (low-diameter) regions

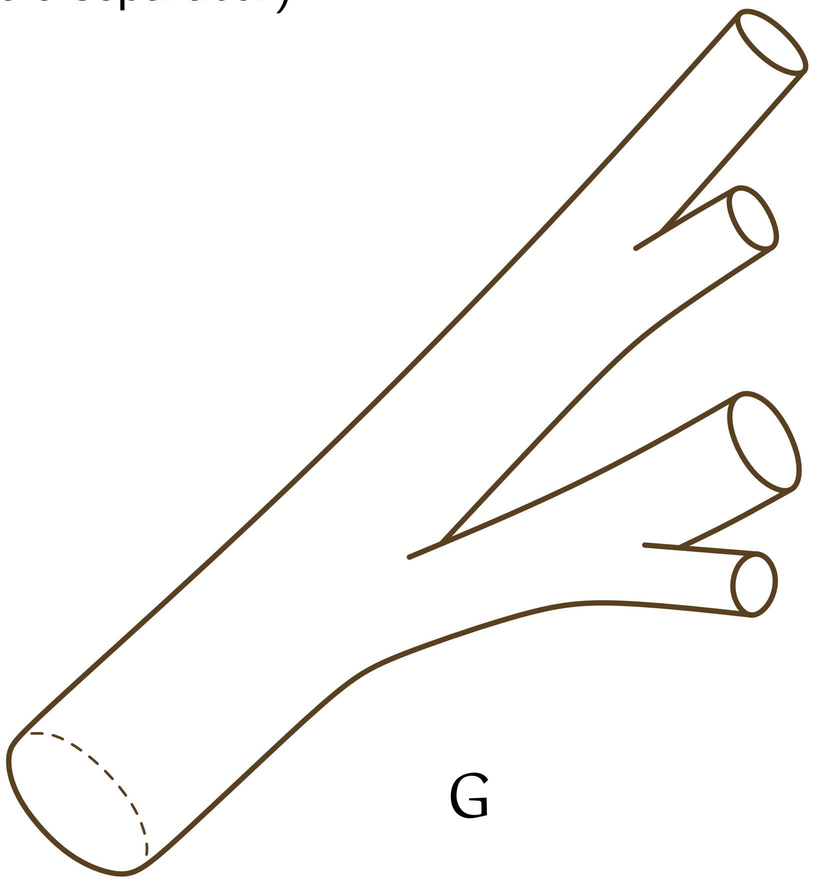
Step 3. Limit #regions, boundary sizes, and #holes per region

Second: make I/O-efficient, i.e. $O(\texttt{sort}(N))$ I/Os, $O(N \log N)$ time
(Use bootstrapping with SSSP.)

Multiway cycle separators: construction

Step 1. Partition into small or low-diameter regions

BFS on face-incidence graph (like Miller's two-way simple cycle separator)



Requirements:

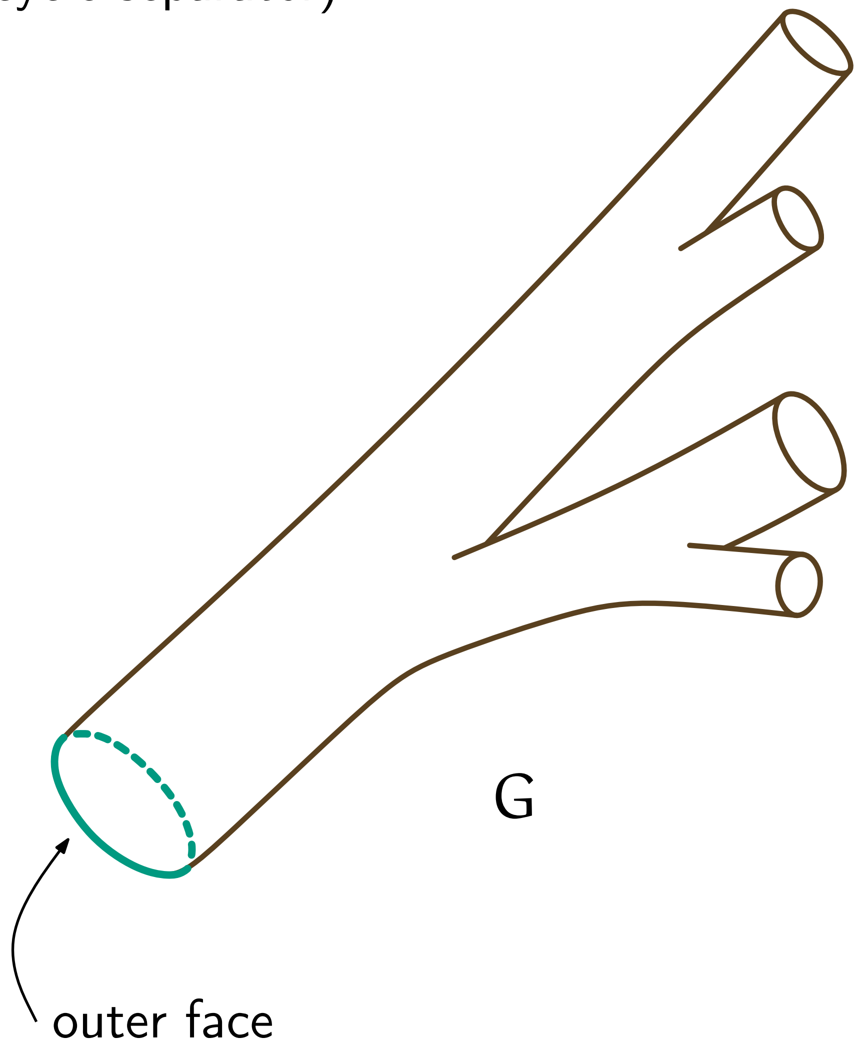
- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$

Multiway cycle separators: construction

Step 1. Partition into small or low-diameter regions

BFS on face-incidence graph (like Miller's two-way simple cycle separator)



Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

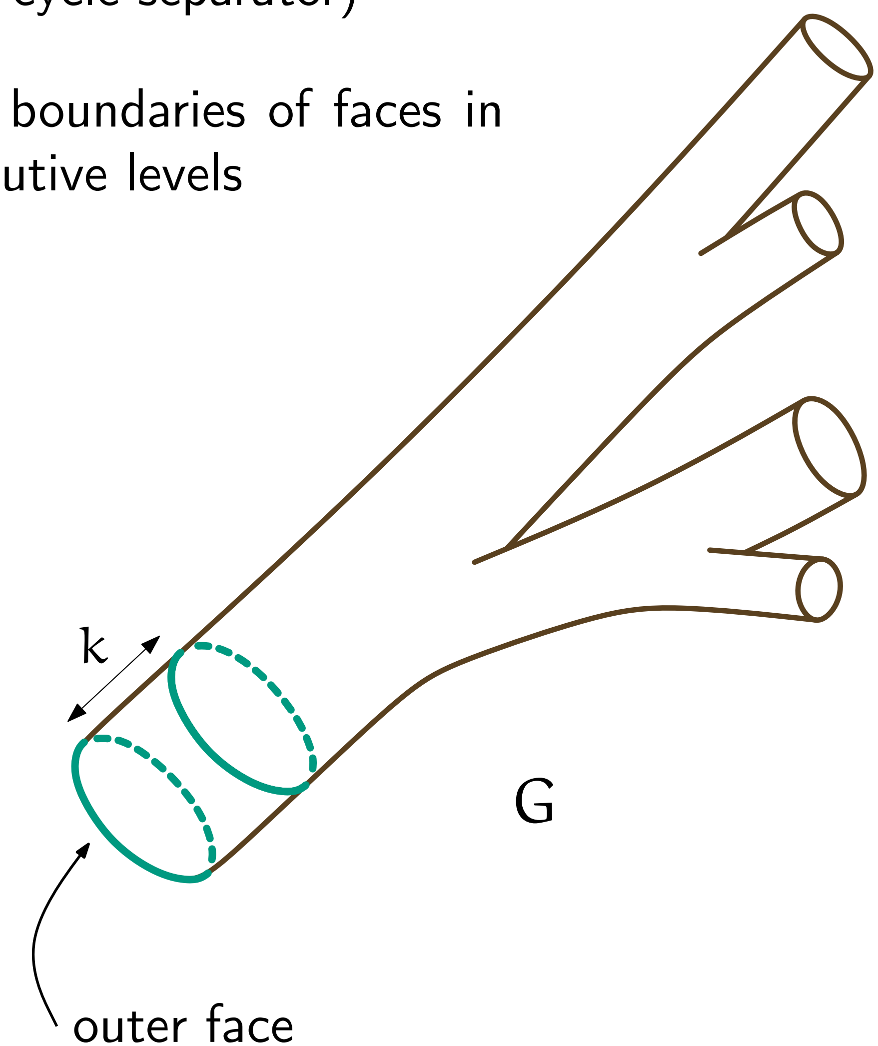
Big regions: size $> \varepsilon N$

Multiway cycle separators: construction

Step 1. Partition into small or low-diameter regions

BFS on face-incidence graph (like Miller's two-way simple cycle separator)

Select boundaries of faces in consecutive levels



Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

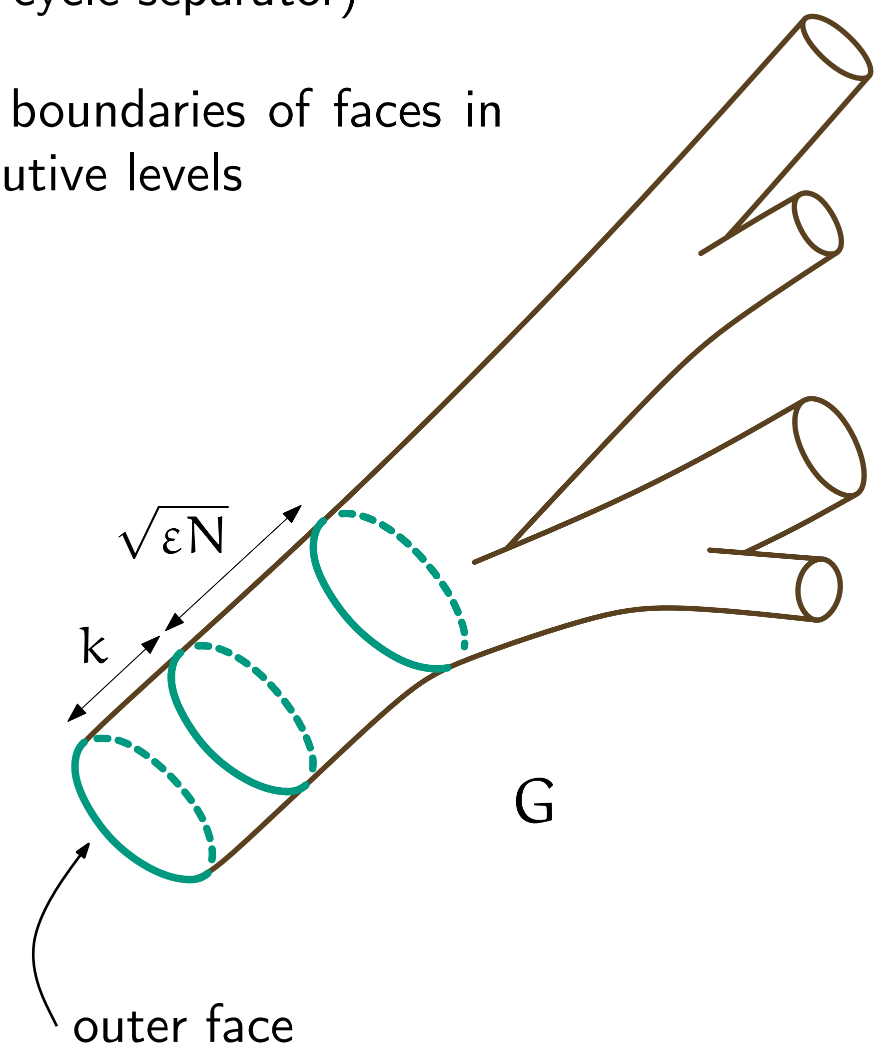
Big regions: size $> \varepsilon N$

Multiway cycle separators: construction

Step 1. Partition into small or low-diameter regions

BFS on face-incidence graph (like Miller's two-way simple cycle separator)

Select boundaries of faces in consecutive levels



Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

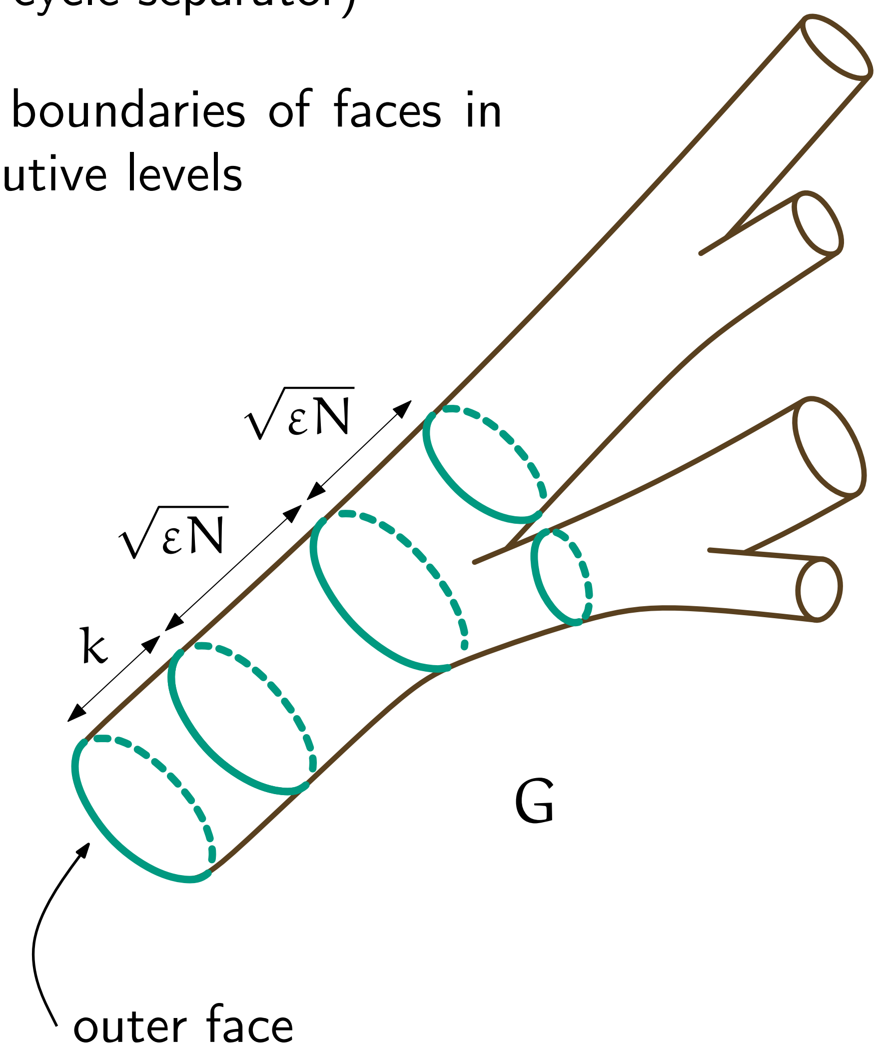
Big regions: size $> \varepsilon N$

Multiway cycle separators: construction

Step 1. Partition into small or low-diameter regions

BFS on face-incidence graph (like Miller's two-way simple cycle separator)

Select boundaries of faces in consecutive levels



Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$

Multiway cycle separators: construction

Step 1. Partition into small or low-diameter regions

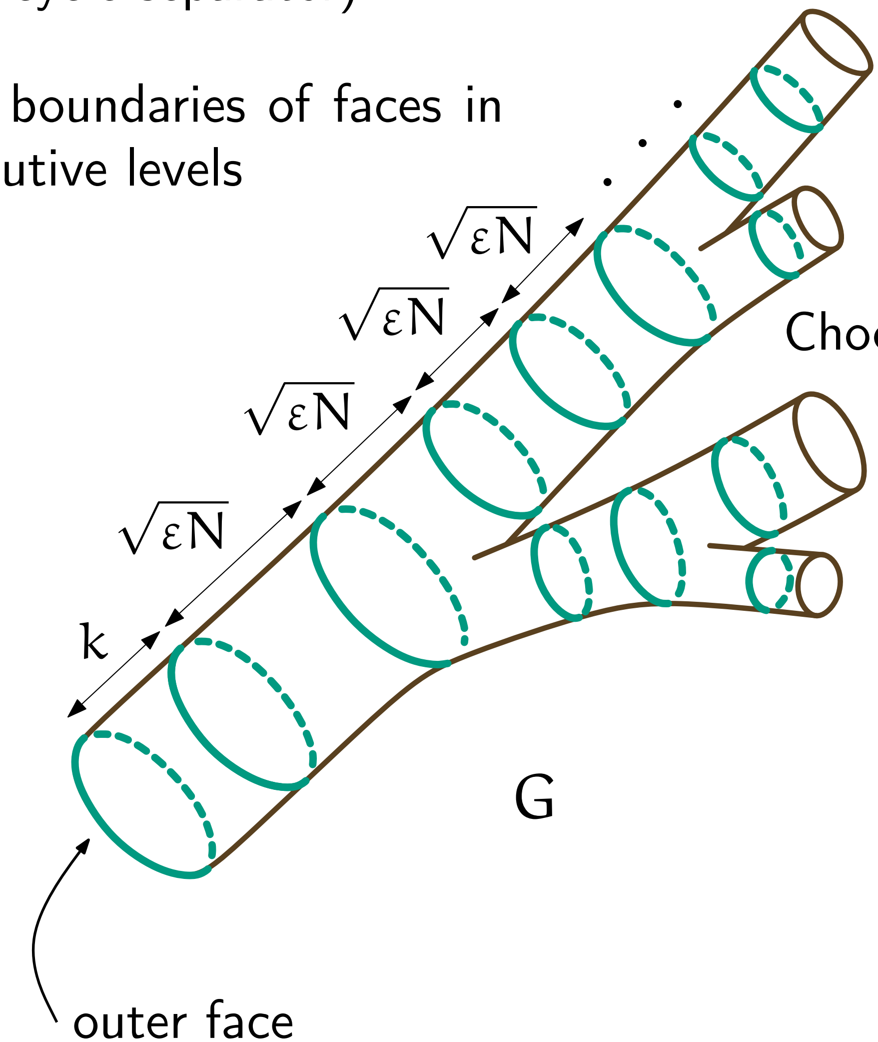
BFS on face-incidence graph (like Miller's two-way simple cycle separator)

Select boundaries of faces in consecutive levels

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$



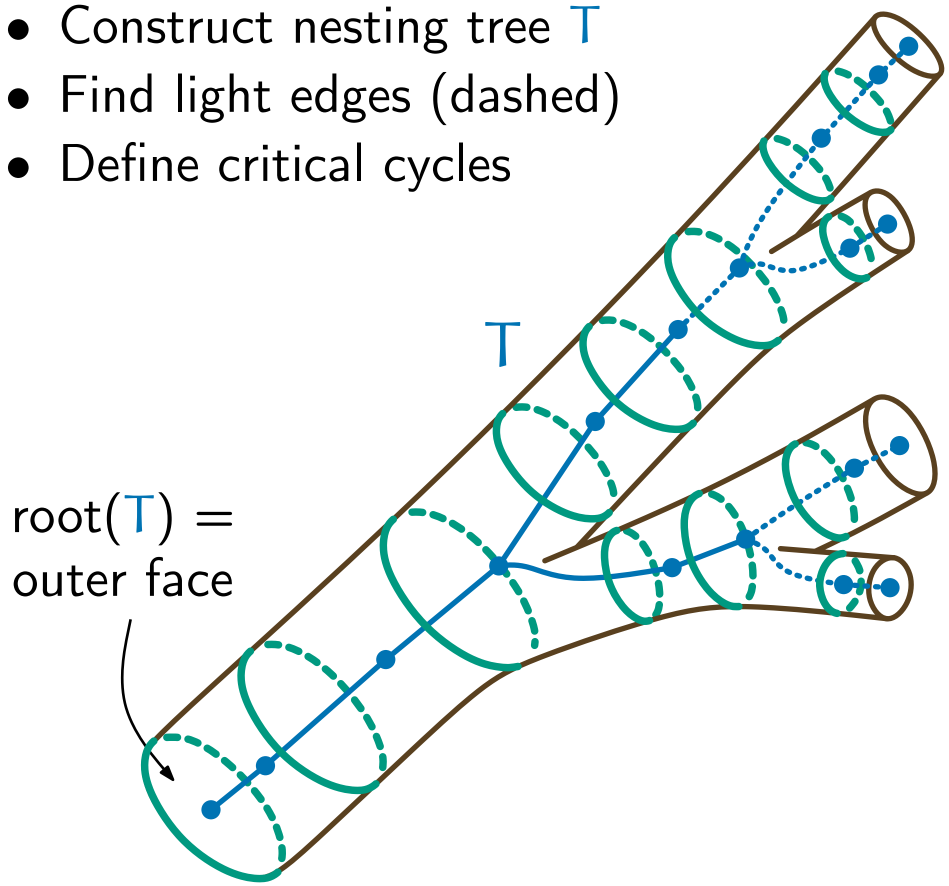
Choose k such that #selected edges $= O(\sqrt{N/\varepsilon})$

Multiway cycle separators: construction

Step 1. Partition into small or low-diameter regions

Reduce #boundary cycles:

- Construct nesting tree T
- Find light edges (dashed)
- Define critical cycles



Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$

Multiway cycle separators: construction

Step 1. Partition into small or low-diameter regions

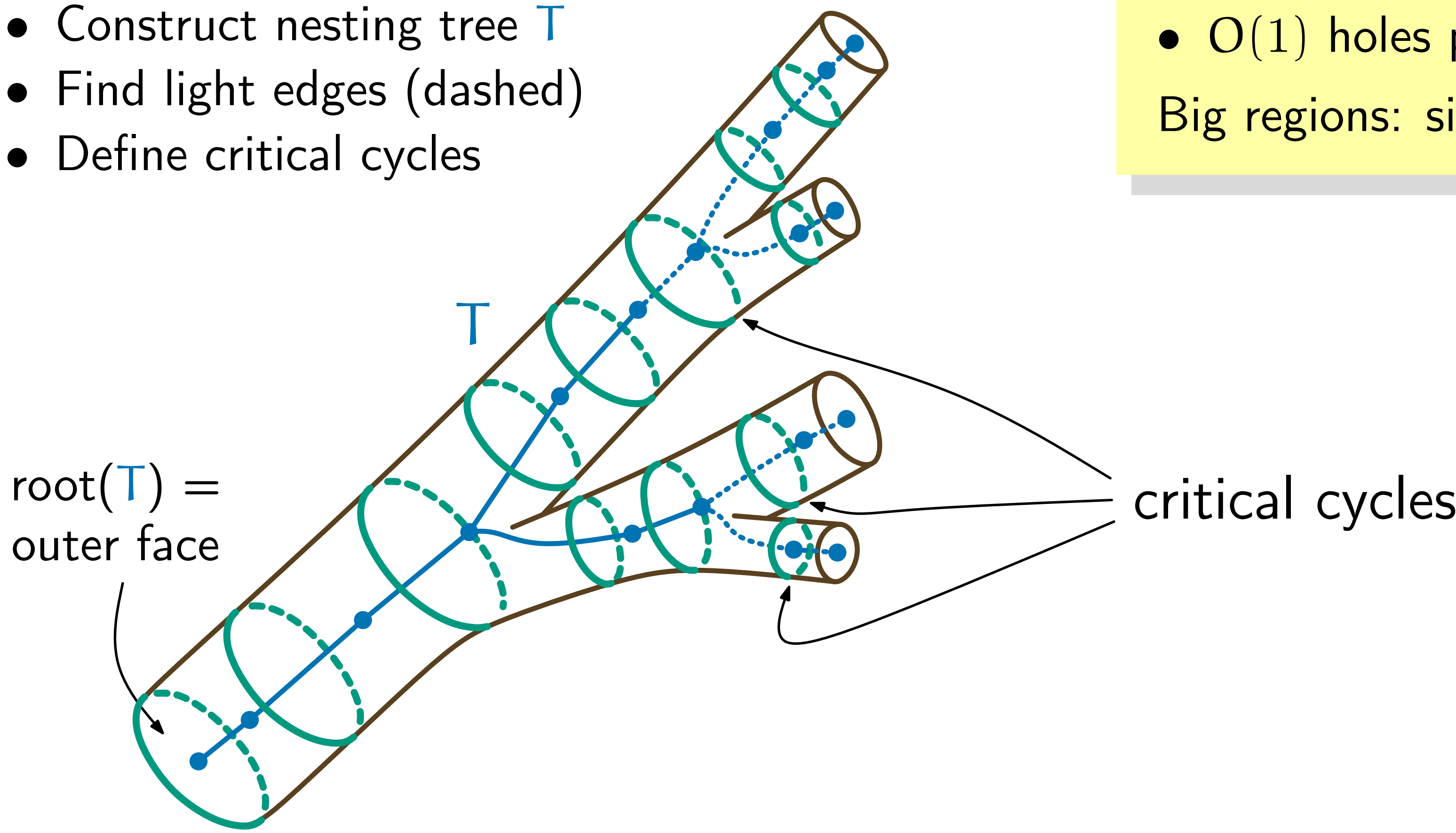
Reduce #boundary cycles:

- Construct nesting tree T
- Find light edges (dashed)
- Define critical cycles

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$



Multiway cycle separators: construction

Step 1. Partition into small or low-diameter regions

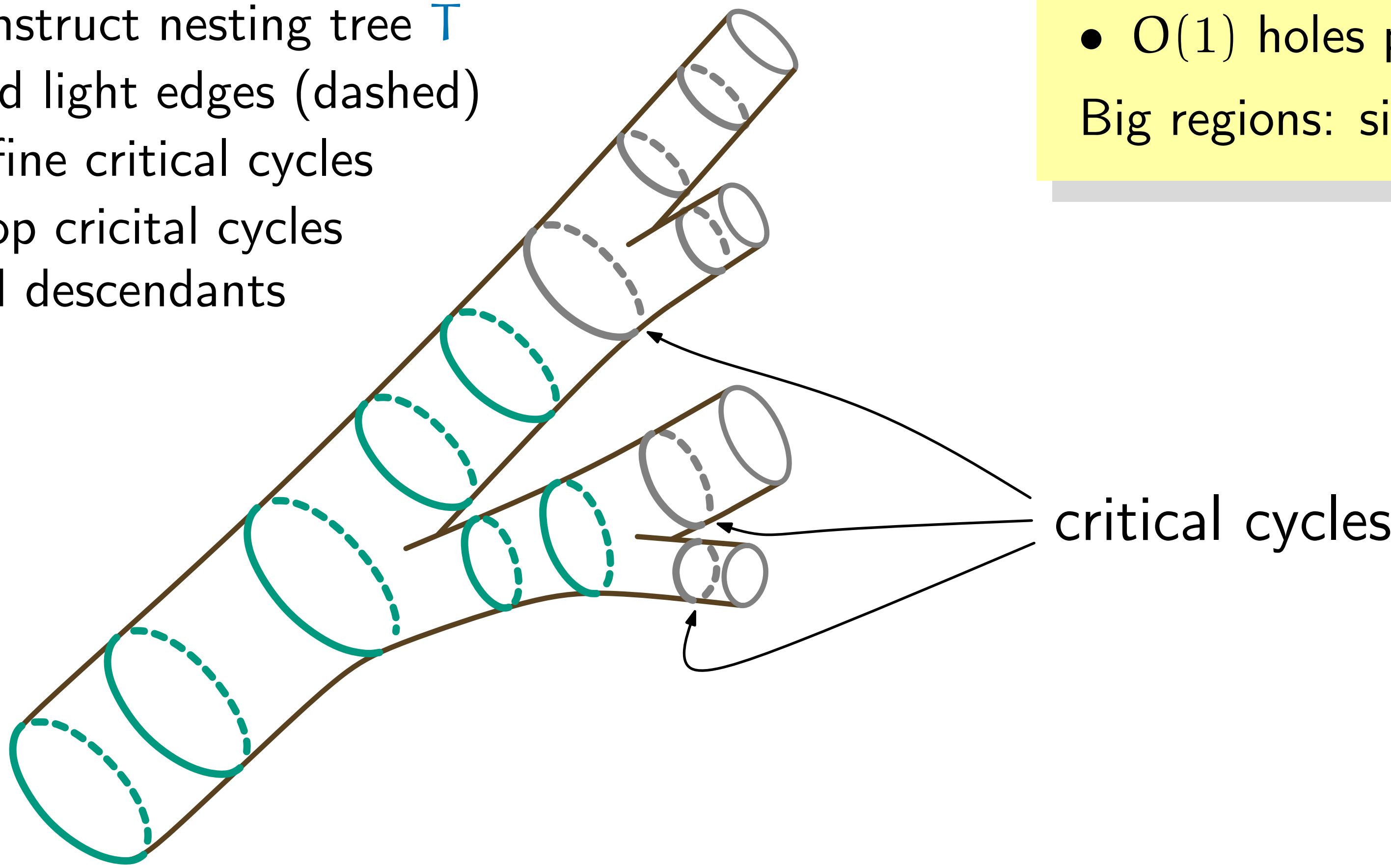
Reduce #boundary cycles:

- Construct nesting tree T
- Find light edges (dashed)
- Define critical cycles
- Drop cricital cycles and descendants

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$



Multiway cycle separators: construction

Step 1. Partition into small or low-diameter regions

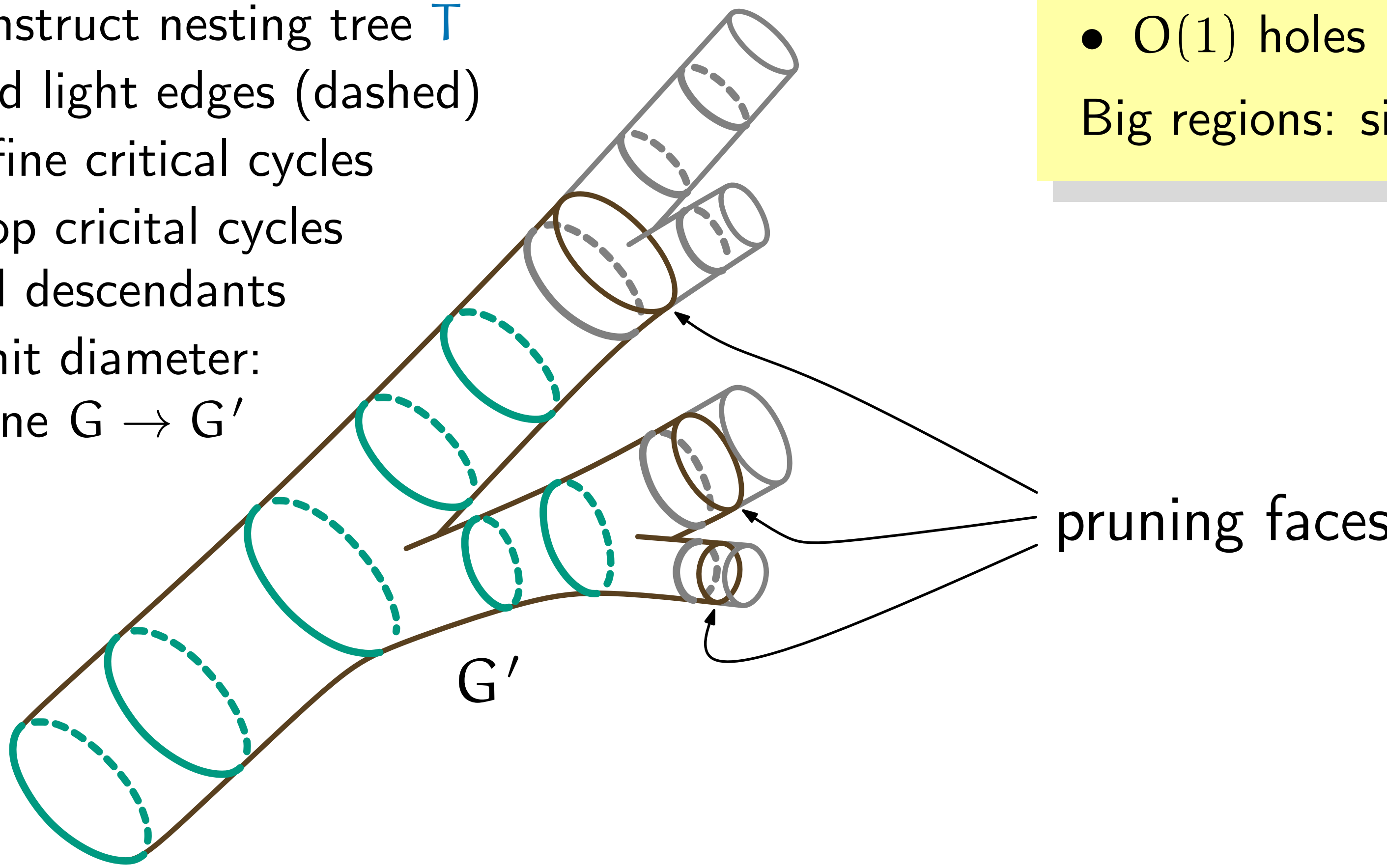
Reduce #boundary cycles:

- Construct nesting tree T
- Find light edges (dashed)
- Define critical cycles
- Drop cricital cycles and descendants
- Limit diameter: prune $G \to G'$

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$



Multiway cycle separators: construction

Step 1. Partition into small or low-diameter regions

Reduce #boundary cycles:

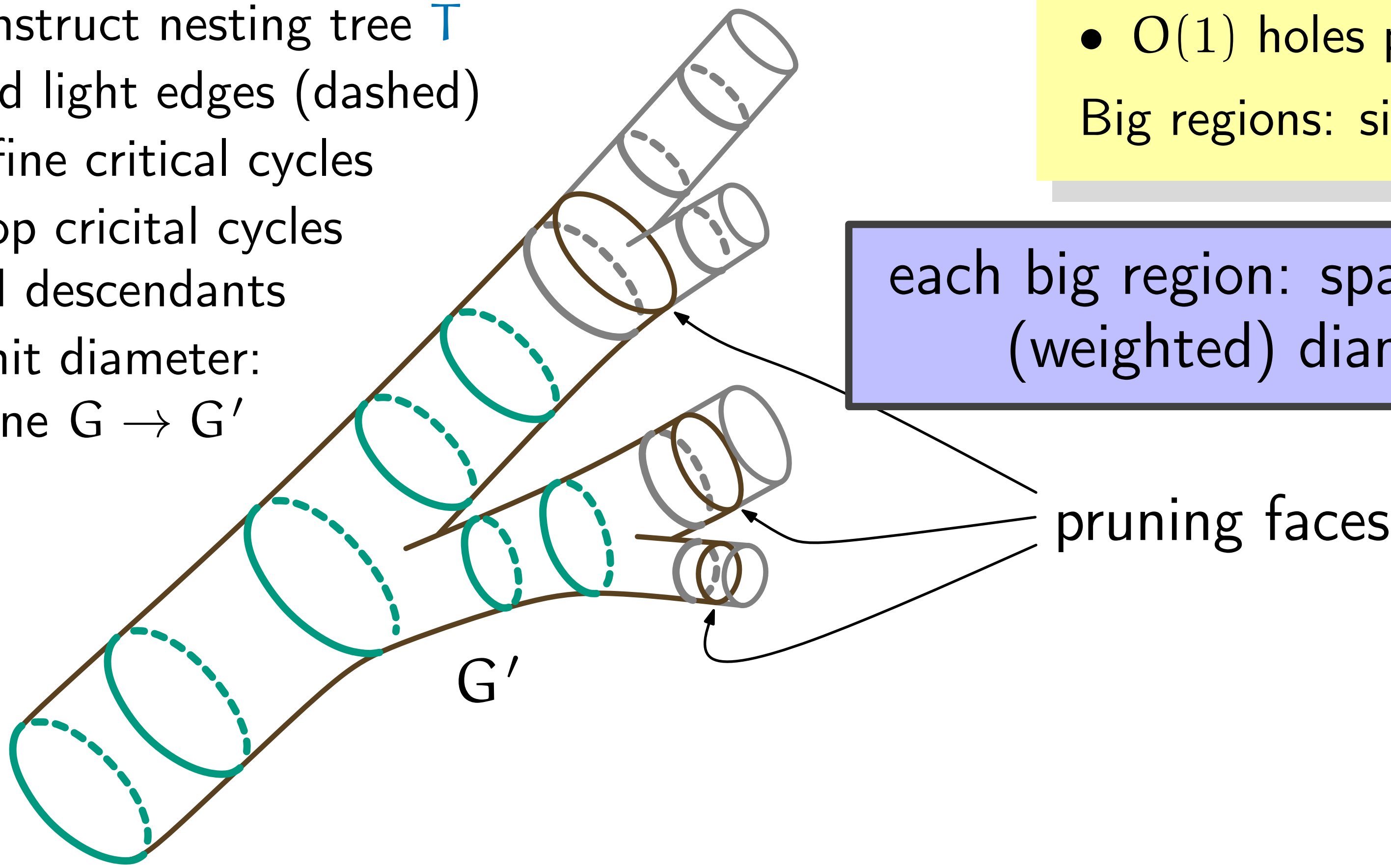
- Construct nesting tree T
- Find light edges (dashed)
- Define critical cycles
- Drop cricital cycles and descendants
- Limit diameter: prune $G \to G'$

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$

each big region: spanning tree with (weighted) diameter $\sqrt{\varepsilon N}$



Multiway cycle separators: construction

Step 1. Partition into small or low-diameter regions

Reduce #boundary cycles:

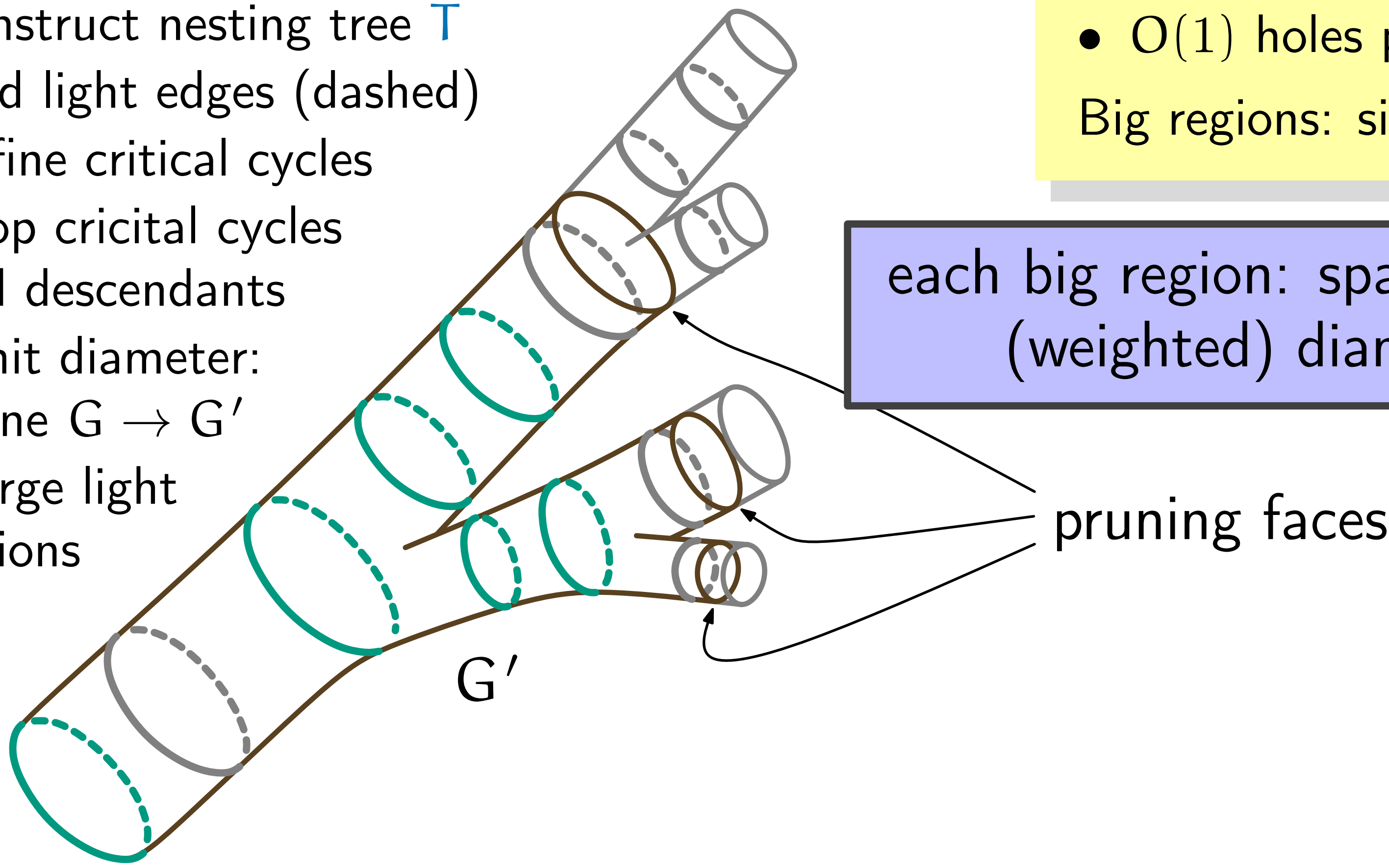
- Construct nesting tree T
- Find light edges (dashed)
- Define critical cycles
- Drop cricital cycles and descendants
- Limit diameter: prune $G \to G'$
- Merge light regions

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$

each big region: spanning tree with (weighted) diameter $\sqrt{\varepsilon N}$



Multiway cycle separators: construction

Step 2. Split big (low-diameter) regions

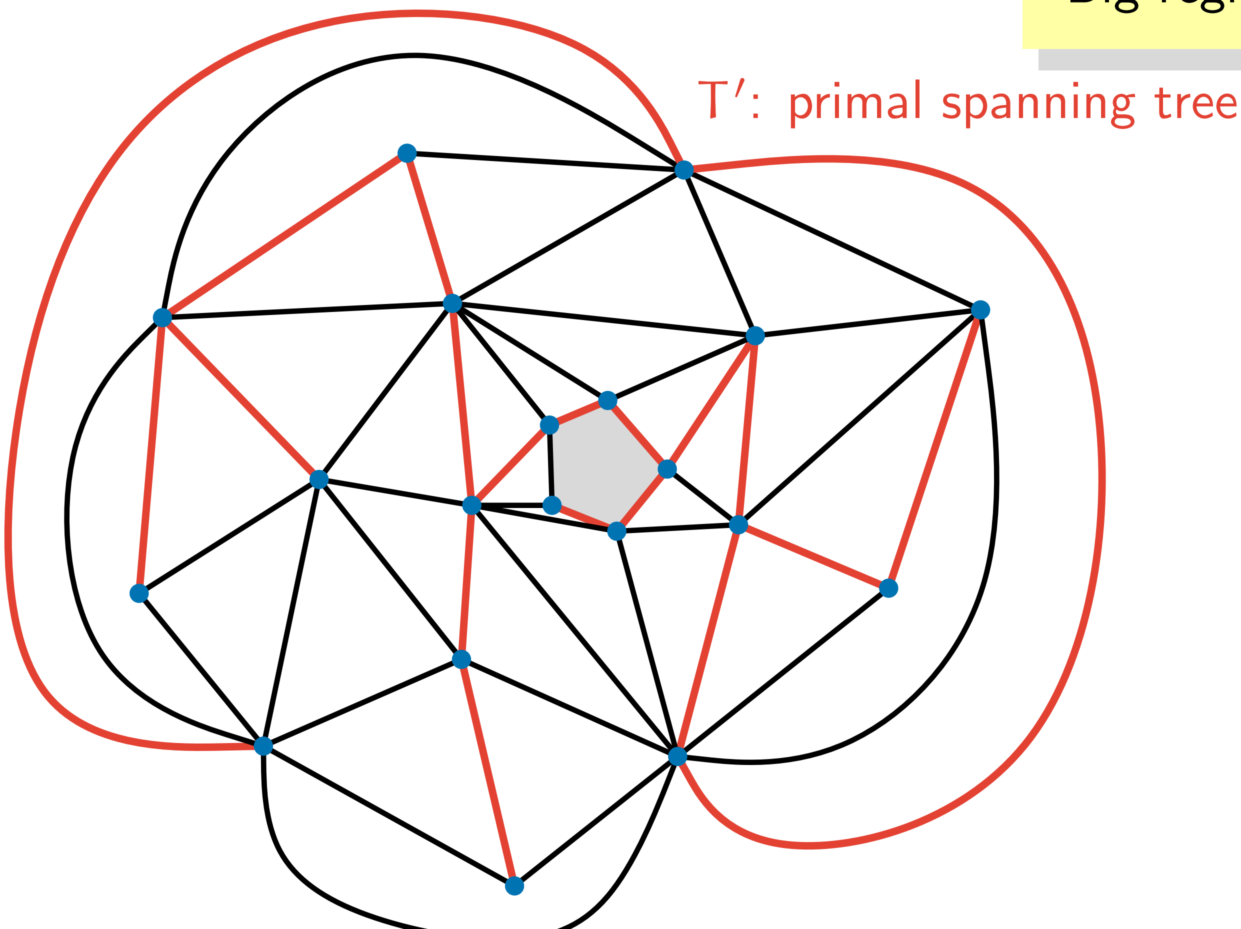
Goal: Partition G' (and G) into small regions

Step 2.1: Separation tree decomposition
Step 2.2: Nesting forest decomposition

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$



Multiway cycle separators: construction

Step 2. Split big (low-diameter) regions

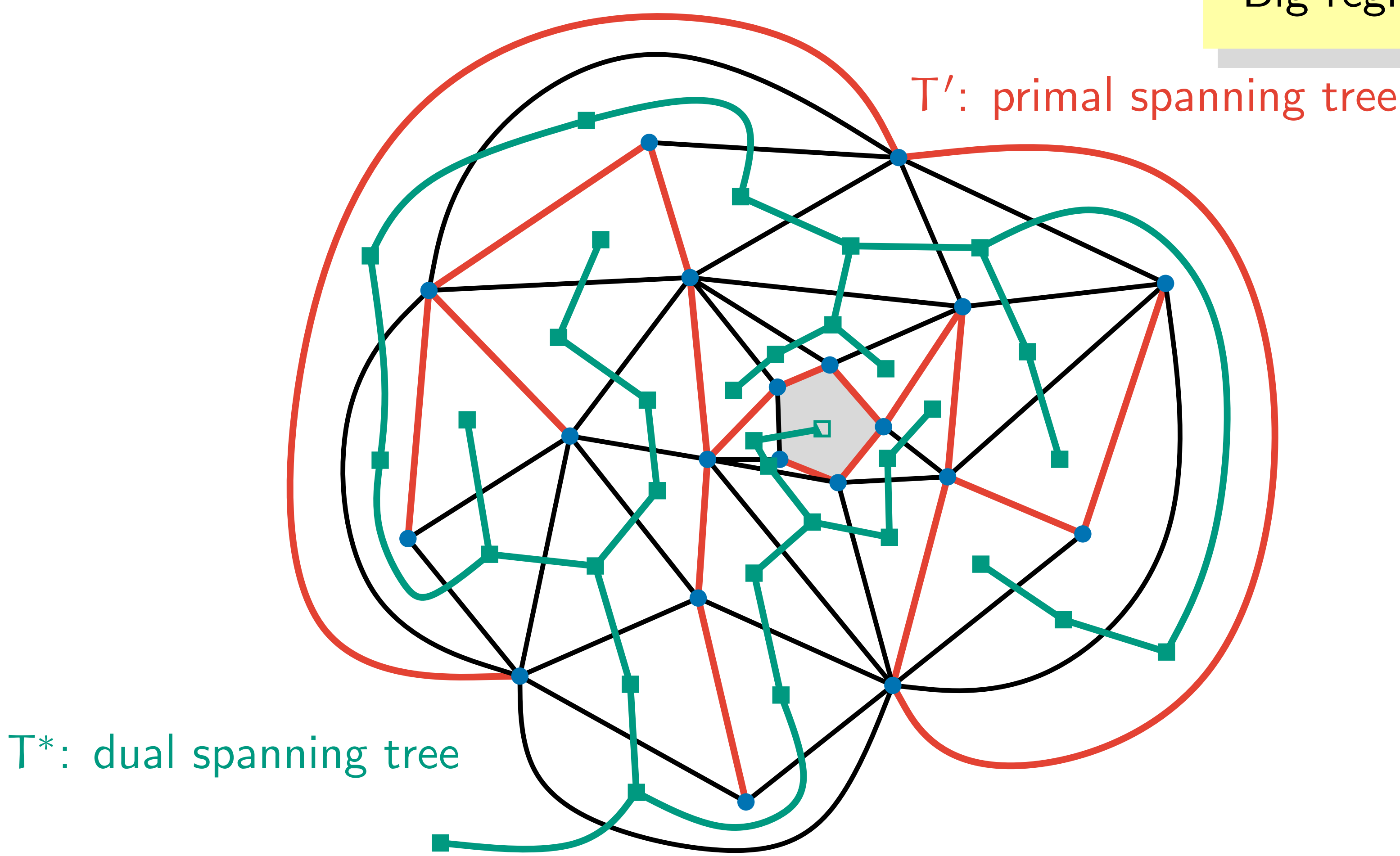
Goal: Partition G' (and G) into small regions

Step 2.1: Separation tree decomposition
Step 2.2: Nesting forest decomposition

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$



Multiway cycle separators: construction

Step 2. Split big (low-diameter) regions

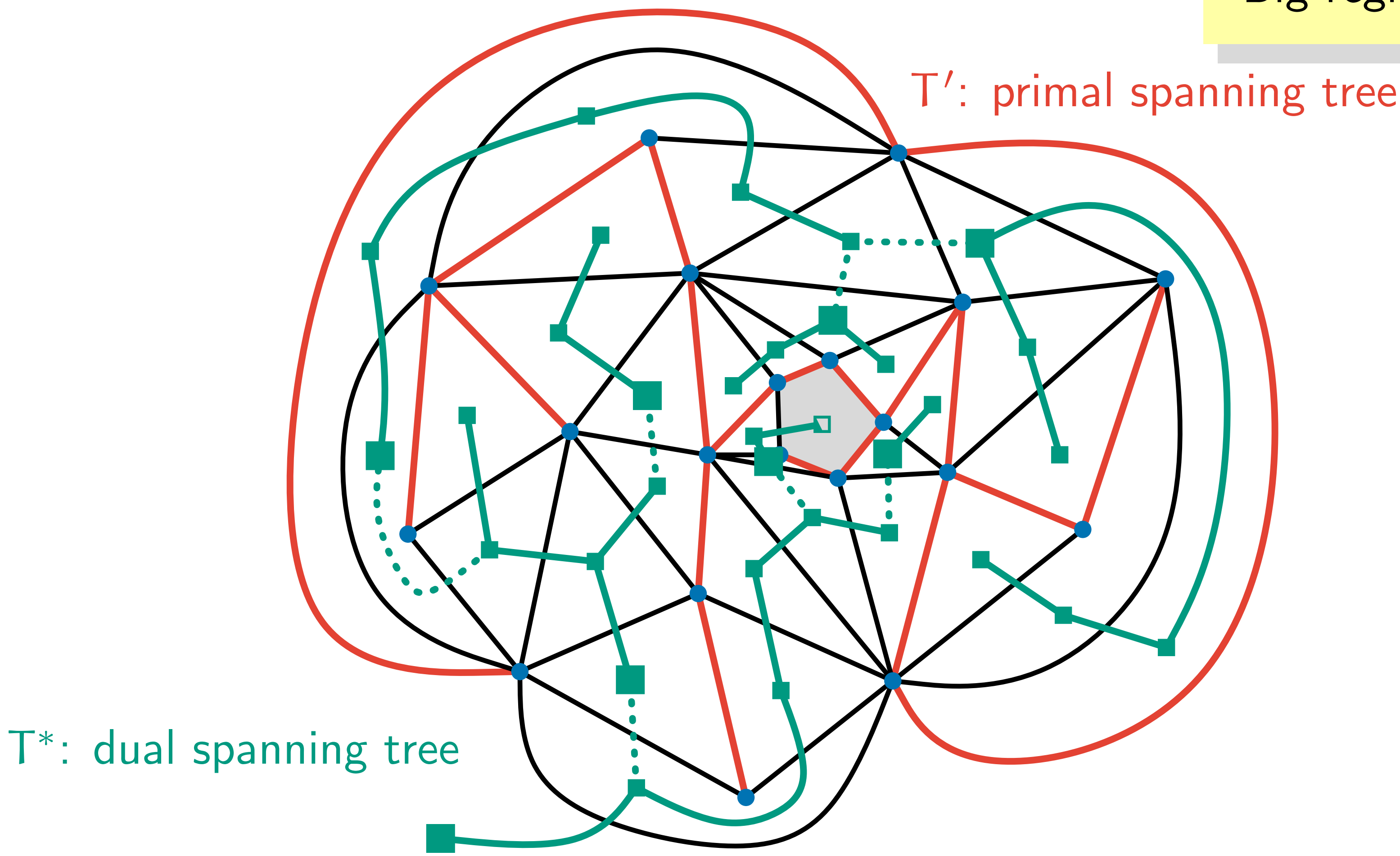
Goal: Partition G' (and G) into small regions

Step 2.1: Separation tree decomposition
Step 2.2: Nesting forest decomposition

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$



Multiway cycle separators: construction

Step 2. Split big (low-diameter) regions

Goal: Partition G' (and G) into small regions

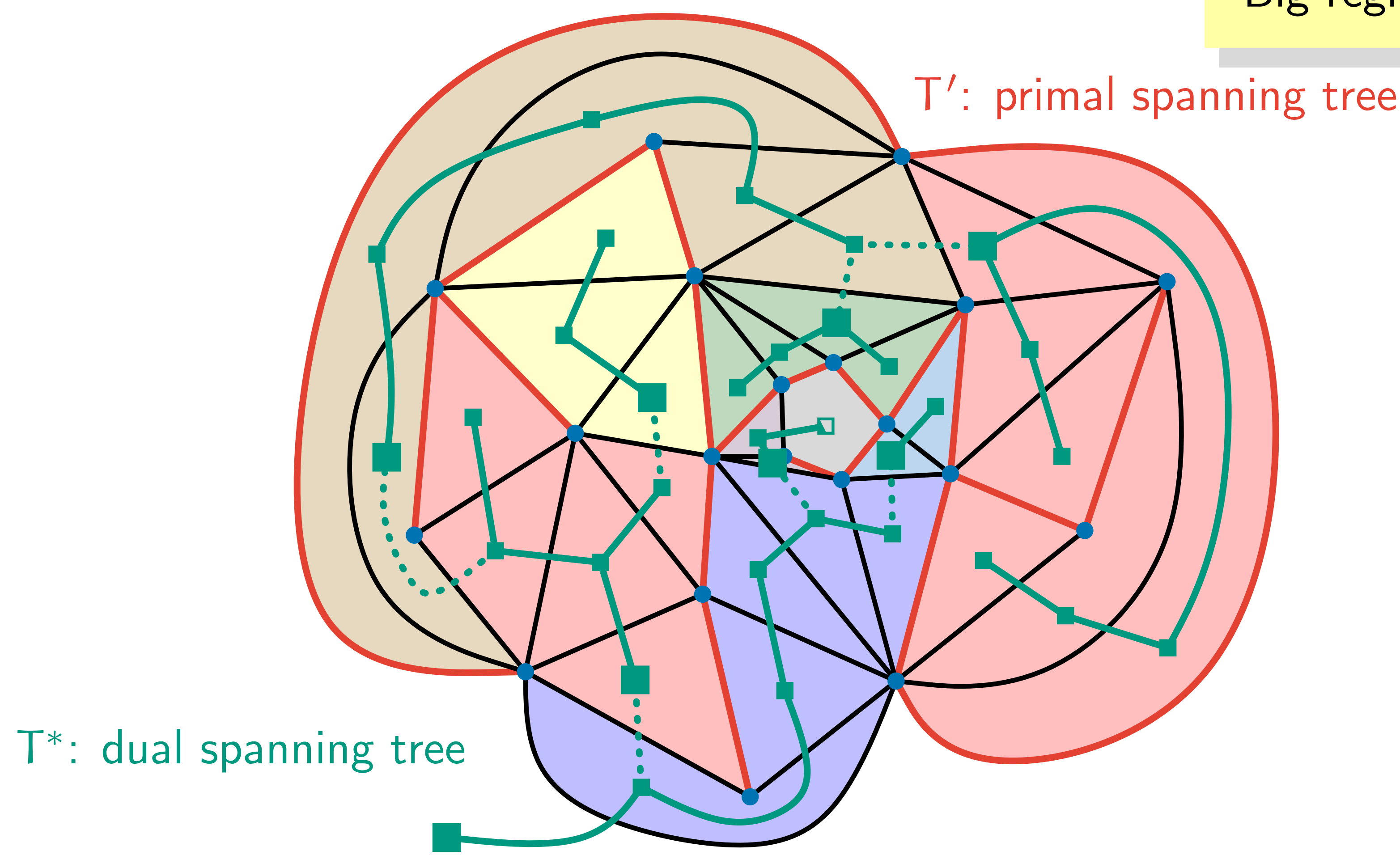
Step 2.1: Separation tree decomposition

Step 2.2: Nesting forest decomposition

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$



Multiway cycle separators: construction

Step 2. Split big (low-diameter) regions

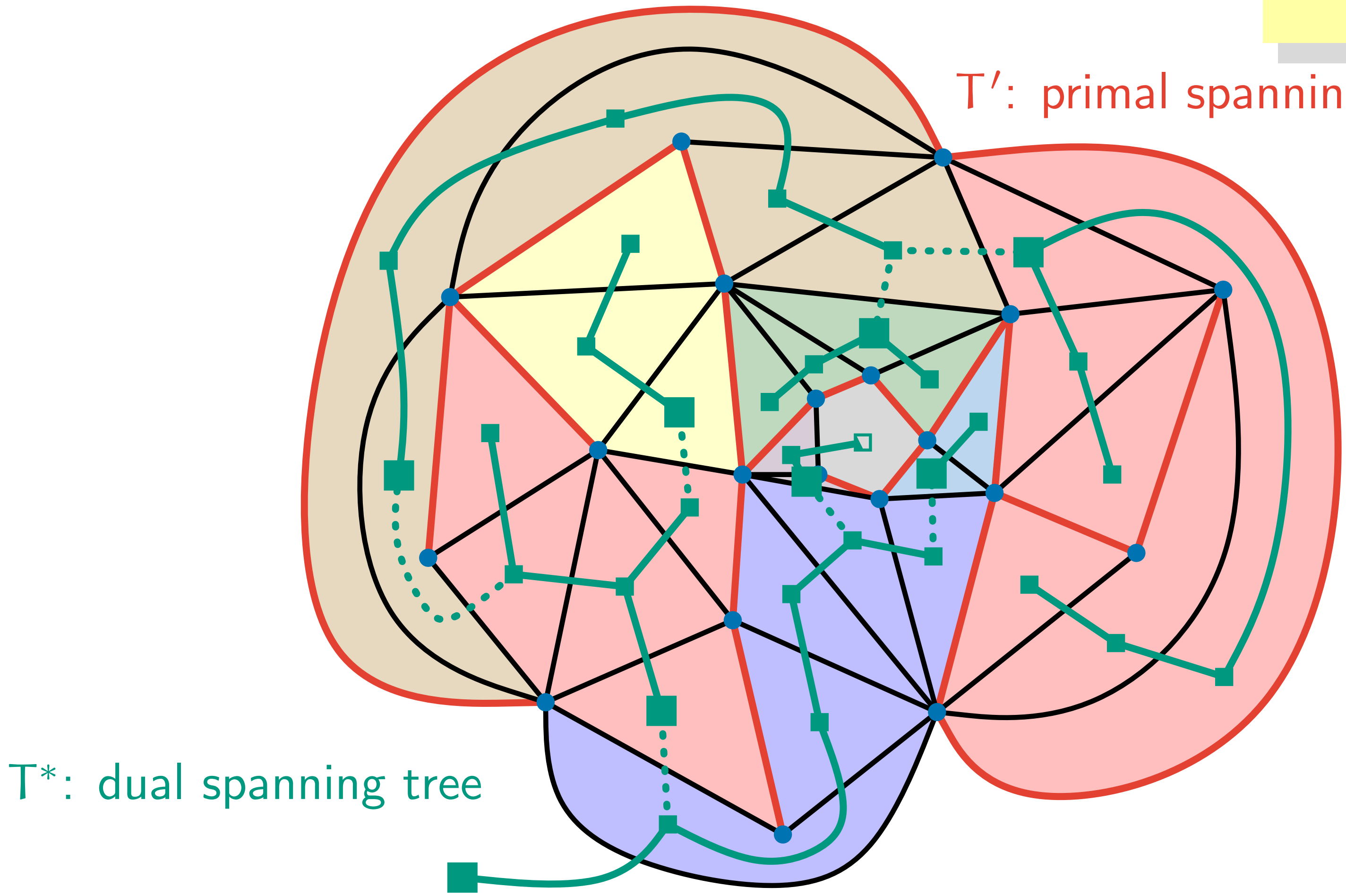
Goal: Partition G' (and G) into small regions

Step 2.1: Separation tree decomposition
Step 2.2: Nesting forest decomposition

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$



Output Step 2.1:
Big regions, such that regions hanging off region roots are small

Multiway simple cycle separators

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$

Summary

- $O(N)$ time internal-memory algorithm
- I/O-efficient algorithm using $O(\mathtt{sort}(N))$ I/Os and $O(N \log N)$ time
- Applications (same I/O and time bounds):
  - SSSP
  - Topsort DAGs
  - Strongly connected components

Bonus features, see paper

- Support vertex, edge, and face weights
- Support general 2-edge-connected graphs with max. face size s (boundary size $\to O(\sqrt{\varepsilon s N})$)

Multiway simple cycle separators

Requirements:

- $O(1/\varepsilon)$ regions
- Region size $O(\varepsilon N)$
- Region boundary $O(\sqrt{\varepsilon N})$
- $O(1)$ holes per region

Big regions: size $> \varepsilon N$

Summary

- $O(N)$ time internal-memory algorithm
- I/O-efficient algorithm using $O(\mathtt{sort}(N))$ I/Os and $O(N \log N)$ time
- Applications (same I/O and time bounds):
  - SSSP
  - Topsort DAGs
  - Strongly connected components

Thanks, that's it!

Bonus features, see paper

- Support vertex, edge, and face weights
- Support general 2-edge-connected graphs with max. face size s (boundary size $\rightarrow O(\sqrt{\varepsilon s N})$)