A Centralized Local Algorithm for the Sparse Spanning Graph Problem

Christoph Lenzen, Reut Levi

A Sublinear Tester for Outerplanarity (and Other Forbidden Minors) With One-Sided Error

Hendrik Fichtenberger, Reut Levi,
Yadu Vasudev, Maximilian Wötzel

A Centralized Local Algorithm for the Sparse Spanning Graph Problem

Christoph Lenzen, Reut Levi

Local Graph Algorithms

Map data © OpenStreetMap contributors

© BrokenSphere / Wikimedia Commons

classic / global algorithm

see whole input, $\Omega(n)$ time

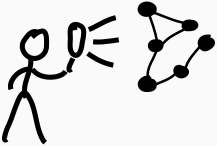
output solution

local algorithm

see only small parts, $o(n)$ time

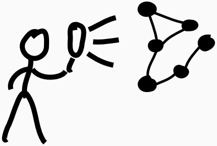
provide query access to solution

The Local Sparse Spanning Graph Problem (LSSG)



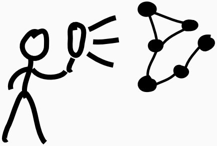
- bounded degree graph $G = (V, E)$ given, $V = [n]$
- LSSG algorithm provides query access to a spanning graph $G' = (V, E')$: "is $(7, 18) \in E'$?"

The Local Sparse Spanning Graph Problem (LSSG)



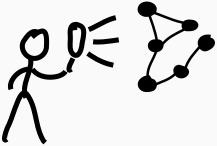
- bounded degree graph $G = (V, E)$ given, $V = [n]$
- LSSG algorithm provides query access to a spanning graph $G' = (V, E')$: "is $(7, 18) \in E'$?"
- answer is computed on demand, no preprocessing, all answers consistent with one G'

The Local Sparse Spanning Graph Problem (LSSG)



- bounded degree graph $G = (V, E)$ given, $V = [n]$
- LSSG algorithm provides query access to a spanning graph $G' = (V, E')$: "is $(7, 18) \in E'$?"
- answer is computed on demand, no preprocessing, all answers consistent with one G'
- local algorithm queries adjacency lists of input e. g., "what is the 2nd neighbor of vertex 14?"

The Local Sparse Spanning Graph Problem (LSSG)

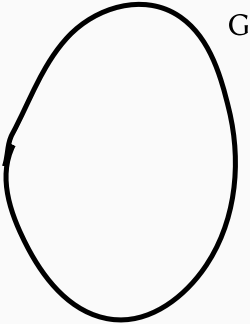


- bounded degree graph $G = (V, E)$ given, $V = [n]$
- LSSG algorithm provides query access to a spanning graph $G' = (V, E')$: "is $(7, 18) \in E'$?"
- answer is computed on demand, no preprocessing, all answers consistent with one G'
- local algorithm queries adjacency lists of input e. g., "what is the 2nd neighbor of vertex 14?"

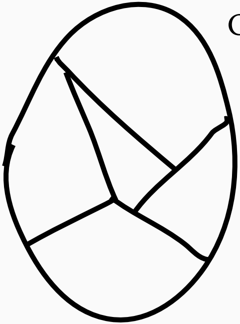
Main Result

An LSSG algorithm with query and time complexity $\tilde{O}(n^{2/3}) \cdot \mathrm{poly}(1/\epsilon)$ per query. It guarantees $|E'| \leq (1 + \epsilon)n$ w.h.p.

Graph Partitions



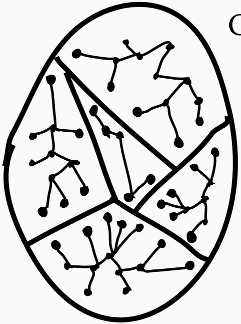
Given a graph G,



Given a graph G,

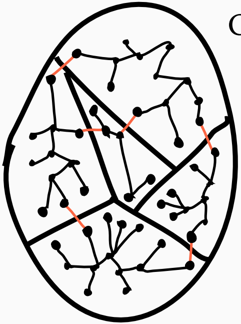
1. partition G into small parts

Graph Partitions



Given a graph G,

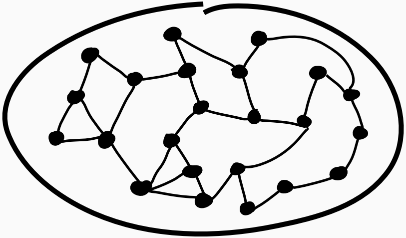
1. partition G into small parts
2. compute spanning tree inside of the parts



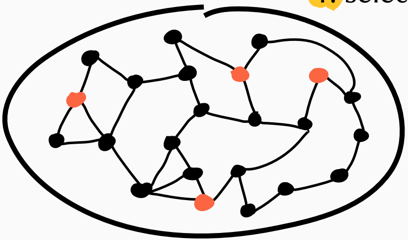
Given a graph G,

1. partition G into small parts
2. compute spanning tree inside of the parts
3. add εn edges between parts to make graph connected

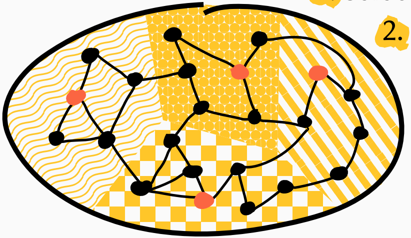
Voronoi Partitions



Voronoi Partitions

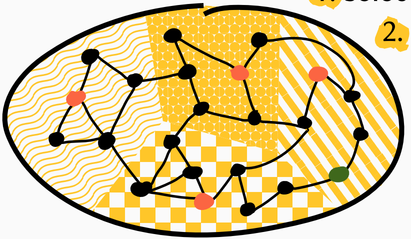


1. select $\Theta(n^{2/3})$ random centers



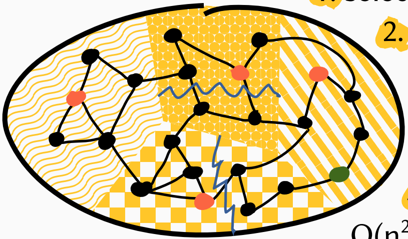
1. select $\Theta(n^{2/3})$ random centers
2. construct Voronoi cells according to path distance

Voronoi Partitions

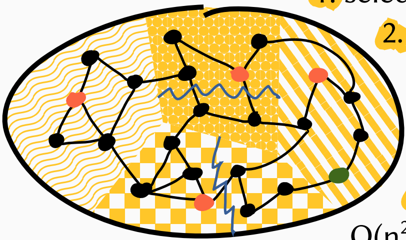


1. select $\Theta(n^{2/3})$ random centers
2. construct Voronoi cells according to path distance
3. sort out remote vertices: distance to center $\Omega(\log n)$

Voronoi Partitions



1. select $\Theta(n^{2/3})$ random centers
2. construct Voronoi cells according to path distance
3. sort out remote vertices: distance to center $\Omega(\log n)$
4. shatter Voronoi cells into $O(n^{2/3})$ *core* clusters of size $O(n^{1/3})$



1. select $\Theta(n^{2/3})$ random centers
2. construct Voronoi cells according to path distance
3. sort out remote vertices: distance to center $\Omega(\log n)$
4. shatter Voronoi cells into $O(n^{2/3})$ *core* clusters of size $O(n^{1/3})$

summary: each core cluster has

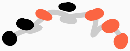
- a BFS spanning tree ✓
- diameter $O(\log n)$ ✓
- size $O(n^{1/3})$ ✓

Local Construction of Core Clusters

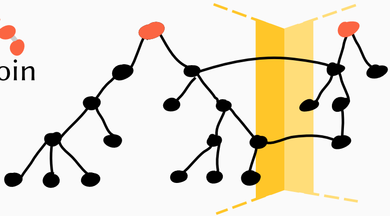
Local Construction of Core Clusters

1. each vertex flips a coin

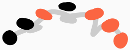
Local Construction of Core Clusters



1. each vertex flips a coin
2. BFS exploration

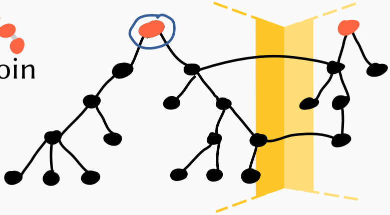


Local Construction of Core Clusters

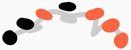


1. each vertex flips a coin
2. BFS exploration
3. cut heavy children

complexity: $O(n^{1/3})$

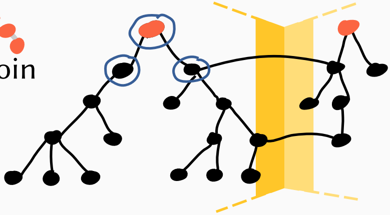


Local Construction of Core Clusters

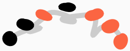


1. each vertex flips a coin
2. BFS exploration
3. cut heavy children

complexity: $O(n^{1/3})$

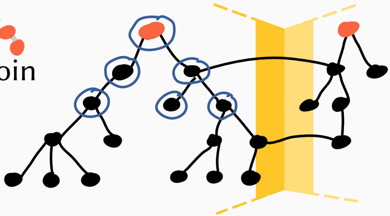


Local Construction of Core Clusters

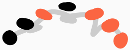


1. each vertex flips a coin
2. BFS exploration
3. cut heavy children

complexity: $O(n^{1/3})$

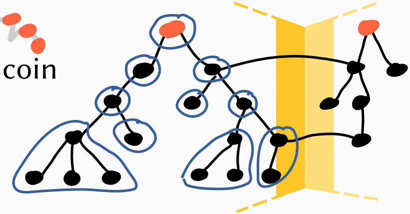


Local Construction of Core Clusters

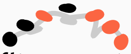


1. each vertex flips a coin
2. BFS exploration
3. cut heavy children

complexity: $O(n^{1/3})$

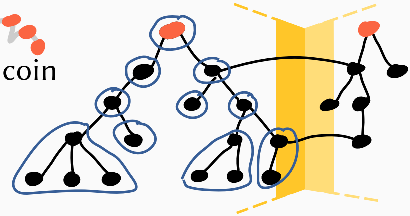


Local Construction of Core Clusters



1. each vertex flips a coin
2. BFS exploration
3. cut heavy children

complexity: $O(n^{1/3})$



Not in This Talk

- Partitioning of remote vertices into *remote clusters*
- Joining core clusters to reduce number of cluster pairs (≈ number of edges needed to connect clusters) to εn

A Sublinear Tester for Outerplanarity (& Other Forbidden Minors) With One-Sided Error

Hendrik Fichtenberger, Reut Levi,

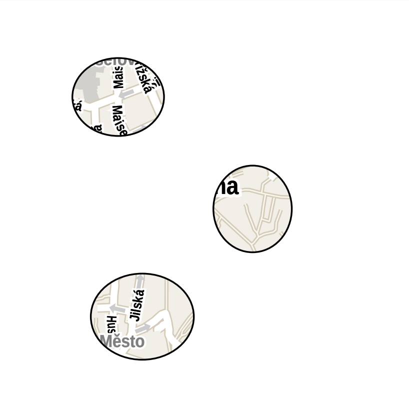
Yadu Vasudev, Maximilian Wötzel

Sublinear Graph Algorithms

classic / global algorithm

see everything, $\Omega(n)$ time

output solution

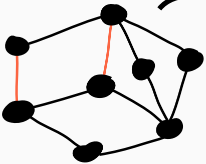


sublinear algorithm

see only small parts, $o(n)$ time

estimate solution's value

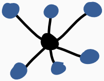
Testing Outerplanarity With One-Sided Error



$x = 0$: accept always

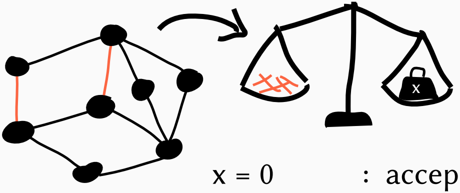
$0 < x \le \varepsilon dn$: don't care

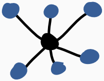
$\varepsilon dn < x$: reject w.p. 2/3



$\#$  $< d$

Testing Outerplanarity With One-Sided Error





● < d

$x = 0$: accept always

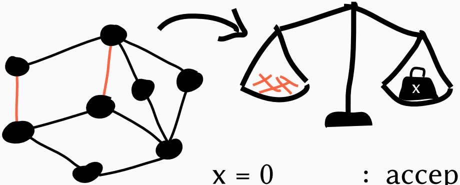
$0 < x \leq \varepsilon dn$: don't care

$\varepsilon dn < x$: reject w.p. 2/3

ε-far

ε-close

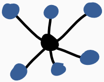
Testing Outerplanarity With One-Sided Error



x = 0 : accept always

0 < x ≤ εdn : don't care

εdn < x : reject w.p. 2/3



● < d

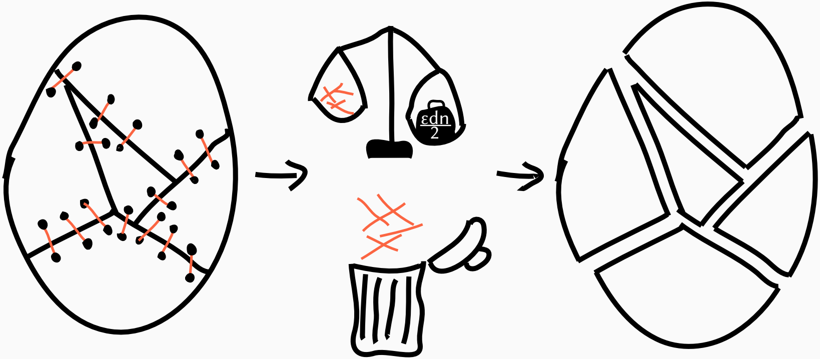
ε-far

ε-close

Main Result

An $\mathscr{F}$-minor freeness tester for every family $\mathscr{F}$ of forbidden minors that contains either the $K_{2,k}$, $(k \times 2)$-grid or k-circus graph with query complexity / running time $\tilde{O}(n^{2/3}/\epsilon^5)$

Partitioning Revisited

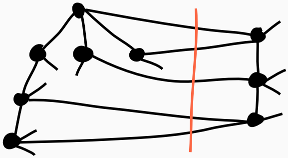


How about the cut in Voronoi partitions?

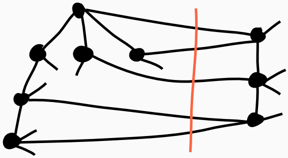
- number of cut edges involving a remote cluster is $\leq \epsilon dn/4$
- number of cut edges between core clusters might be $> \epsilon dn/4$

Construction of $K_{2,k}$

Theorem: cuts of size > f between clusters imply $K_{2,k}$-minors



Theorem: cuts of size > f between clusters imply $K_{2,k}$-minors

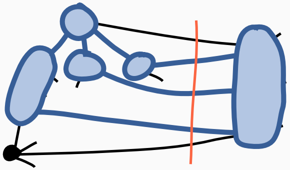


idea:
always have BFS tree,
enforce more structure
by large cut size

$f \approx \Theta(d\ k\ log(n))$

Construction of $K_{2,k}$

Theorem: cuts of size > f between clusters imply $K_{2,k}$-minors

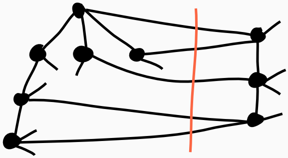


idea:
always have BFS tree,
enforce more structure
by large cut size

$f \approx \Theta(d\ k\ log(n))$

Construction of $K_{2,k}$

Theorem: cuts of size > f between clusters imply $K_{2,k}$-minors



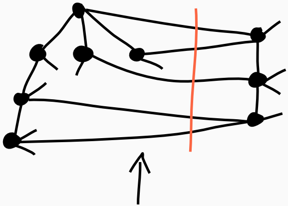
idea:
always have BFS tree,
enforce more structure
by large cut size

$$f \approx \Theta(d\ k\ log(n))$$

at most d incident
edges per vertex

Construction of $K_{2,k}$

Theorem: cuts of size > f between clusters imply $K_{2,k}$-minors



$(k+1) \cdot log(n)$ cut vertices on left side

idea:
always have BFS tree,
enforce more structure
by large cut size

$f \approx \Theta(d\ k\ log(n))$

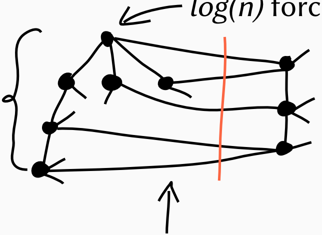
at most d incident
edges per vertex

Construction of $K_{2,k}$

Theorem: cuts of size > f between clusters imply $K_{2,k}$-minors

log(n) forces branching

$O(log\ n)$



(k+1) · *log(n)* cut vertices on left side

idea:
always have BFS tree,
enforce more structure
by large cut size

$f \approx \Theta(d\ k\ log(n))$

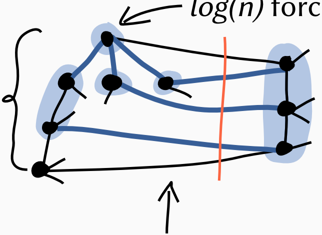
at most *d* incident
edges per vertex

Construction of $K_{2,k}$

Theorem: cuts of size > f between clusters imply $K_{2,k}$-minors

log(n) forces branching

$O(\log n)$



(k+1) · log(n) cut vertices on left side

idea:
always have BFS tree,
enforce more structure
by large cut size

$f \approx \Theta(d\, k \log(n))$

at most *d* incident
edges per vertex

Summary

Result: Local Spanning Graphs

An LSSG algorithm with query and time complexity $\tilde{O}(n^{2/3}) \cdot \text{poly}(1/\epsilon)$ per query. It guarantees $|E'| \leq (1+\epsilon)n$ w.h.p.

Result: Minor-Freeness Testing

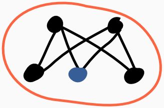
An $\mathcal{F}$-minor freeness tester for every family $\mathcal{F}$ of forbidden minors that contains either the $K_{2,k}$, $(k \times 2)$-grid or k-circus graph with query complexity / running time $\tilde{O}(n^{2/3}/\epsilon^5)$

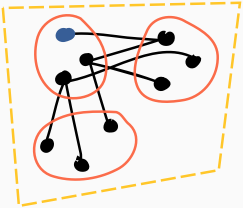
recent progress by Kumar et al. (2018) for arbitrary $\mathcal{F}$: $O(n^{1/2+o(1)})$

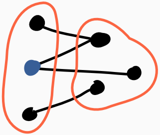
Additional Slides

Algorithm

1. sample O(f / ε) edges
2. for every sampled edge (u,v):
   i) explore cluster(s) of u,v
   ii) compute cut sizes between core cluster and remaining Voronoi cell of u,v
   iii) compute cut sizes between core / core cluster of u / v
3. reject iff minor found or some cut > f

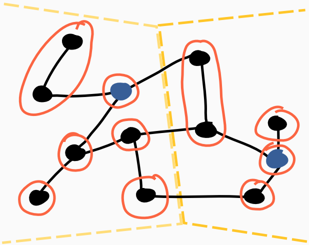






Super Clusters

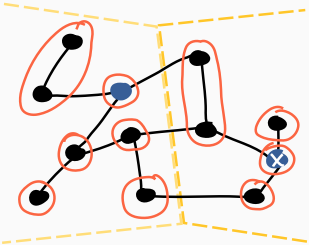
Problem: $f \cdot \#(\text{core clusters})^2 \notin O(\varepsilon dn)$



Super Clusters

Problem: $f \cdot \#(\text{core clusters})^2 \notin O(\varepsilon dn)$

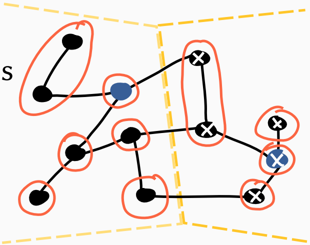
1. mark each Voronoi cell w.p. $1/n^{1/3}$



Super Clusters

Problem: $f \cdot \#(\text{core clusters})^2 \notin O(\varepsilon dn)$

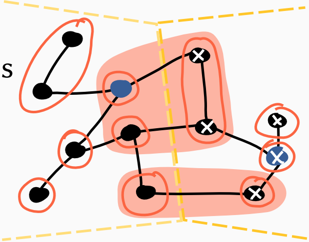
1. mark each Voronoi cell w.p. $1/n^{1/3}$
2. mark each core cluster of marked cells



Super Clusters

Problem: $f \cdot \#(\text{core clusters})^2 \notin O(\varepsilon d n)$

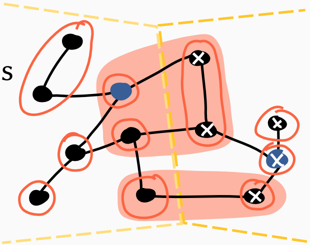
1. mark each Voronoi cell w.p. $1/n^{1/3}$
2. mark each core cluster of marked cells
3. join unmarked core clusters with marked neighboring core clusters



Super Clusters

Problem: $f \cdot \#(\text{core clusters})^2 \notin O(\varepsilon dn)$

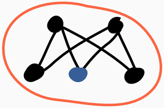
1. ~~mark~~ each Voronoi cell w.p. $1/n^{1/3}$
2. ~~mark~~ each core cluster of marked cells
3. join unmarked core clusters with marked neighboring core clusters

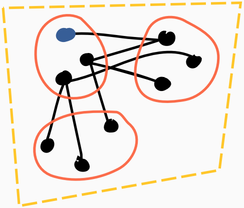


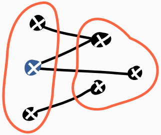
☑ ~~locally reconstructable~~
☑ local membership queries
☑ $f \cdot \#(\text{core clusters}) \cdot \#(\text{super clusters}) \in O(\varepsilon dn)$

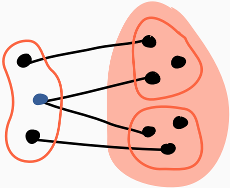
Tester With Super Clusters

1. sample O(f / ε) edges
2. for every sampled edge (u,v):
   - i) explore cluster(s) of u,v
   - ii) compute cut sizes between core cluster and remaining Voronoi cell of u,v
   - iii) compute cut sizes between core / core and core / super cluster of u / v
3. reject iff minor found or some cut > f

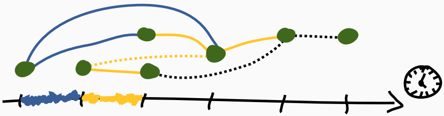








Remote Clusters [Elkin, Neiman, 2017]



1. each remote vertex picks random delay
2. after delay, start BFS: one level per time
3. construct *remote* clusters from BFS