# Approximation algorithms for Euler genus and related problems

Chandra Chekuri  Anastasios Sidiropoulos

February 3, 2014

*Slides based on a presentation of Tasos Sidiropoulos*

# Graphs and topology

# Graphs and topology

**Theorem (Kuratowski, 1930)**

*A graph is planar if and only if it does not contain $K_5$ and $K_{3,3}$ as a topological minor.*

# Graphs and topology

**Theorem (Kuratowski, 1930)**

*A graph is planar if and only if it does not contain $K_5$ and $K_{3,3}$ as a topological minor.*

**Theorem (Wagner, 1937)**

*A graph is planar if and only if it does not contain $K_5$ and $K_{3,3}$ as a minor.*

# Minors and Topological minors

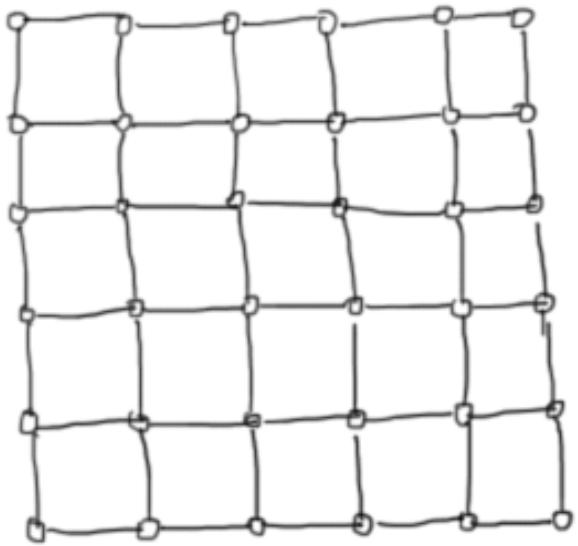
**Definition**

A graph $H$ is a minor of $G$ if $H$ is obtained from $G$ by a sequence of edge/vertex deletions and edge contractions.

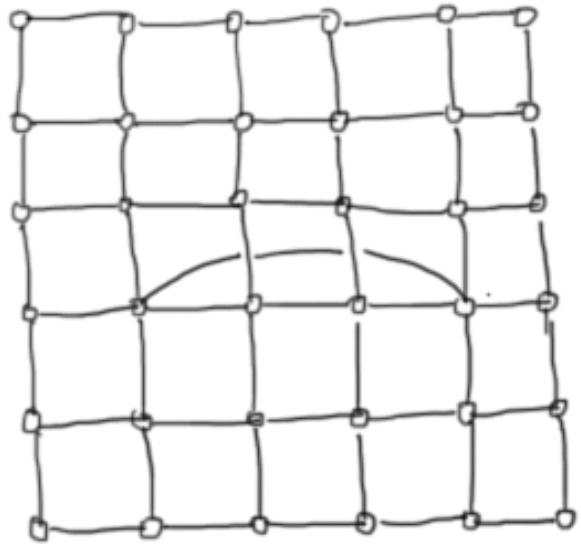
**Definition**

A graph $H$ is a topological minor of $G$ if a subdivision of $H$ is isomorphic to a subgraph of $G$.

# Planarity

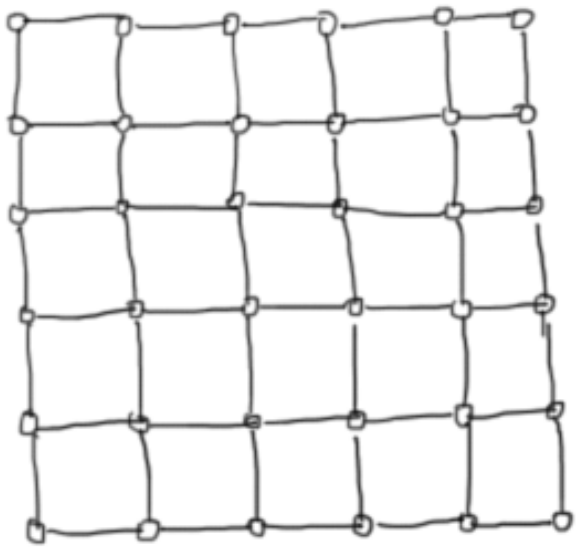


planar graph

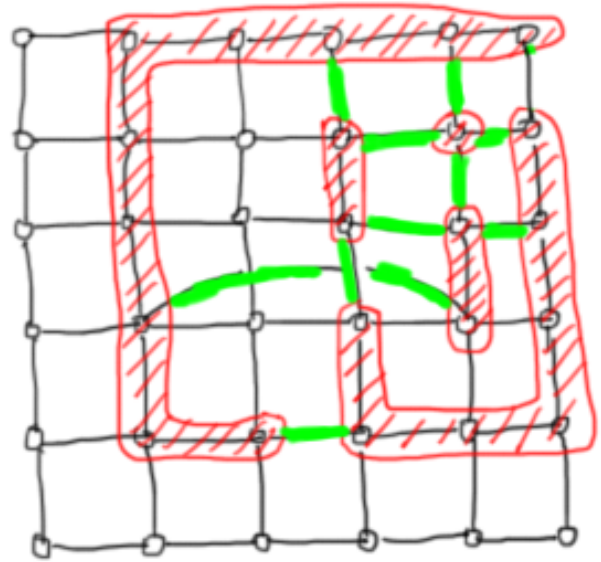


non-planar graph

# Planarity

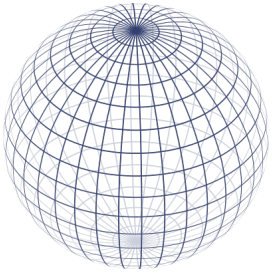


planar graph

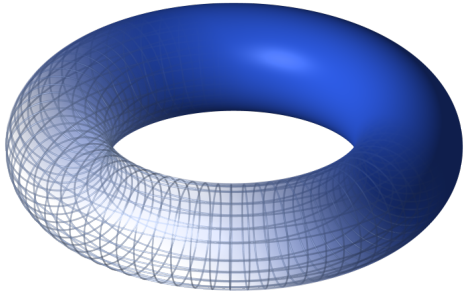


non-planar graph

# What about other surfaces?

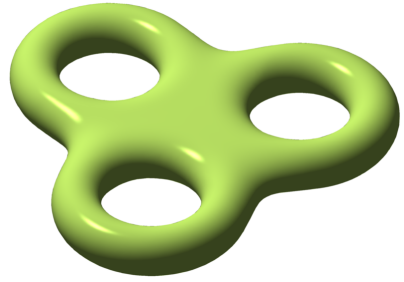


sphere

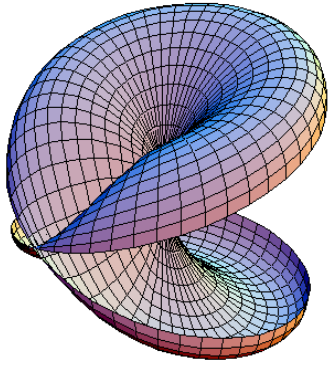


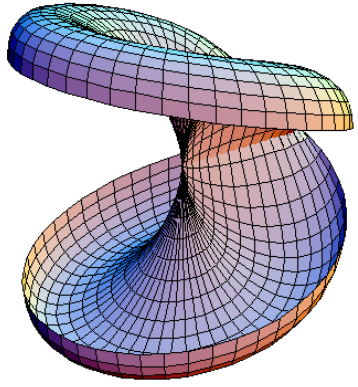
torus

double torus

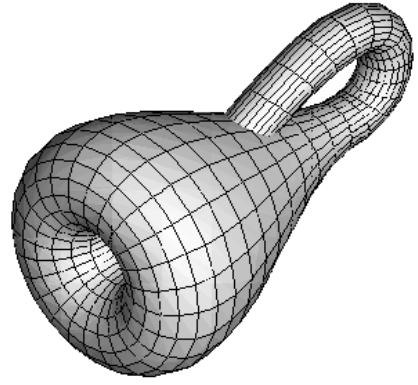


triple torus



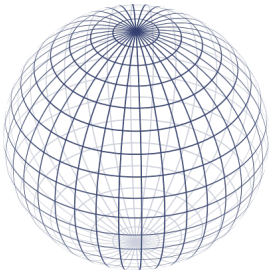


real projective plane

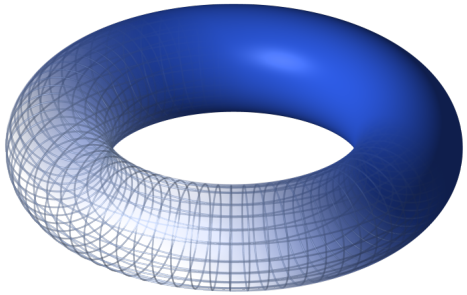


Klein bottle

# What about other surfaces?

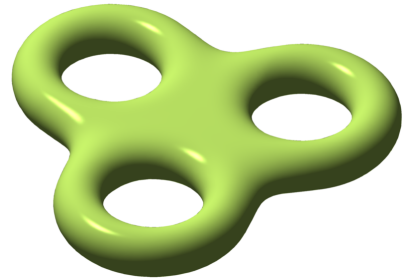


sphere
$g = 0$

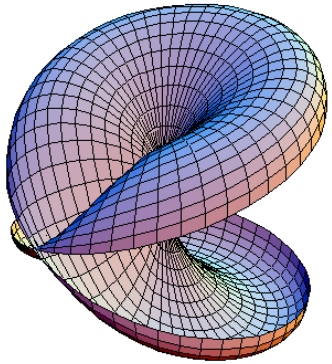


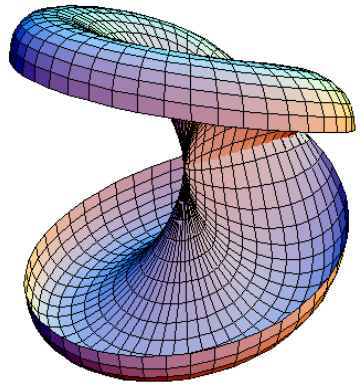
torus
$g = 1$

double torus
$g = 2$

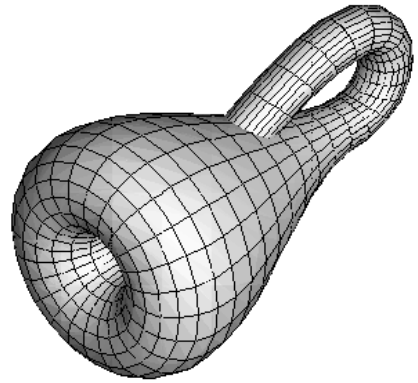


triple torus
$g = 3$





real projective plane
$k = 1$



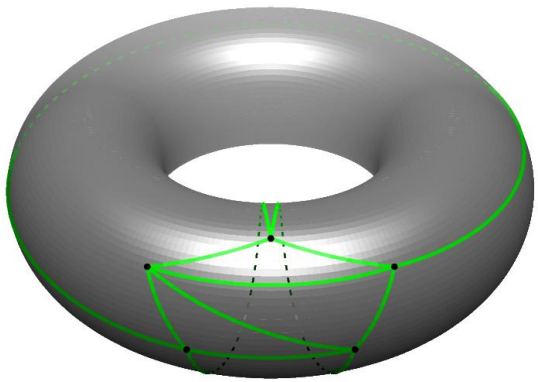
Klein bottle
$k = 2$

# Genus of graphs

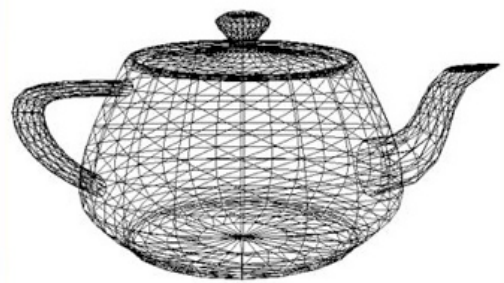
## Definition

The *orientable* (reps. *non-orientable*) *genus* of a graph $G$ is the minimum $k$, such that $G$ admits an embedding into a surface of orientable (resp. non-orientable) genus $k$.

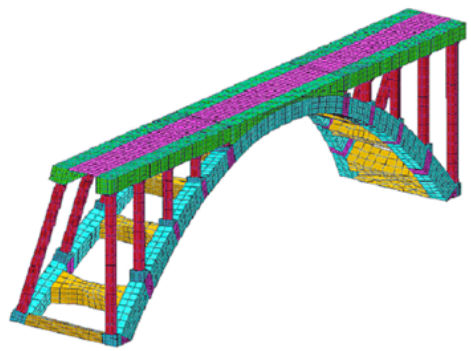
- Graph genus of a graph quantifies "closeness" to planarity.



$\text{genus}(K_5) = 1$



$\text{genus}(\text{teapot}) \leq 1$



$\text{genus}(\text{bridge}) \leq 24$

# Euler genus

For any surface $\mathcal{S}$,

$$\text{eg}(\mathcal{S}) := \min\{2 \cdot \text{genus}(\mathcal{S}), \text{nonorientable-genus}(\mathcal{S})\}$$

# Euler genus

For any surface $\mathcal{S}$,

$$\text{eg}(\mathcal{S}) := \min\{2 \cdot \text{genus}(\mathcal{S}), \text{nonorientable-genus}(\mathcal{S})\}$$

Euler characteristic: For any triangulation of $\mathcal{S}$, with $v$ vertices, $e$ edges, and $f$ faces, we have

$$\chi(\mathcal{S}) := v - e + f = 2 - \text{eg}(\mathcal{S})$$

# Euler genus

For any surface $\mathcal{S}$,

$$\text{eg}(\mathcal{S}) := \min\{2 \cdot \text{genus}(\mathcal{S}), \text{nonorientable-genus}(\mathcal{S})\}$$

Euler characteristic: For any triangulation of $\mathcal{S}$, with $v$ vertices, $e$ edges, and $f$ faces, we have

$$\chi(\mathcal{S}) := v - e + f = 2 - \text{eg}(\mathcal{S})$$

For any graph $G$,

$\text{eg}(G) := \min\{g \in \mathbb{N}_{\geq 0} : G \text{ embeds into a surface } \mathcal{S}, \text{ with } \text{eg}(\mathcal{S}) = g\}$

# Graph minors

**Definition**

A family of graphs $\mathcal{G}$ is minor-closed if for any $G \in \mathcal{G}$ all minors of $G$ are also in $\mathcal{G}$.

**Definition**

A property $P$ of graphs is minor-closed if all graphs satisfying $P$ form a minor-closed family.

# Minor-closed properties

- Planarity

# Minor-closed properties

- Planarity
- Orientable genus $g$, for some $g > 0$.

# Minor-closed properties

- Planarity
- Orientable genus $g$, for some $g > 0$.
- Nonorientable genus $g$, for some $g > 0$.

# Minor-closed properties

- Planarity
- Orientable genus $g$, for some $g > 0$.
- Nonorientable genus $g$, for some $g > 0$.
- Euler genus $g$, for some $g > 0$.

# Minor-closed properties

- Planarity
- Orientable genus $g$, for some $g > 0$.
- Nonorientable genus $g$, for some $g > 0$.
- Euler genus $g$, for some $g > 0$.
- $k$-apex, for some $k > 0$.

# Minor-closed properties

- Planarity
- Orientable genus $g$, for some $g > 0$.
- Nonorientable genus $g$, for some $g > 0$.
- Euler genus $g$, for some $g > 0$.
- $k$-apex, for some $k > 0$.
- Linkless embeddability in $\mathbb{R}^3$.

# Minor-closed properties

- Planarity
- Orientable genus $g$, for some $g > 0$.
- Nonorientable genus $g$, for some $g > 0$.
- Euler genus $g$, for some $g > 0$.
- $k$-apex, for some $k > 0$.
- Linkless embeddability in $\mathbb{R}^3$.
- Treewidth $k$, for some $k > 0$.

# Graph minors: central theorems

## Theorem (Robertson & Seymour, 2004)

*For any minor-closed property $P$, there exists a* **finite** *collection of graphs $\mathcal{F} = \mathcal{F}(P)$, such that a graph $G$ satisfies the property $P$ if and only if $G$ does not contain any of the graphs in $\mathcal{F}$ as a minor.*

# Graph minors: central theorems

**Theorem (Robertson & Seymour, 2004)**

*For any minor-closed property $P$, there exists a* **finite** *collection of graphs $\mathcal{F} = \mathcal{F}(P)$, such that a graph $G$ satisfies the property $P$ if and only if $G$ does not contain any of the graphs in $\mathcal{F}$ as a minor.*

**Theorem (Robertson & Seymour)**

*For any fixed graph $H$ there is polynomial time algorithm that given $G$ decides if $H$ is a minor of $G$. The running time is $O(|V(G)|^3)$.*

# Graph minors: central theorems

## Theorem (Robertson & Seymour, 2004)

*For any minor-closed property $P$, there exists a* **finite** *collection of graphs $\mathcal{F} = \mathcal{F}(P)$, such that a graph $G$ satisfies the property $P$ if and only if $G$ does not contain any of the graphs in $\mathcal{F}$ as a minor.*

## Theorem (Robertson & Seymour)

*For any fixed graph $H$ there is polynomial time algorithm that given $G$ decides if $H$ is a minor of $G$. The running time is $O(|V(G)|^3)$.*

- Implies *existence* of an efficient algorithm for testing *any* minor-closed property.

# Graph minors: central theorems

## Theorem (Robertson & Seymour, 2004)

*For any minor-closed property $P$, there exists a* **finite** *collection of graphs $\mathcal{F} = \mathcal{F}(P)$, such that a graph $G$ satisfies the property $P$ if and only if $G$ does not contain any of the graphs in $\mathcal{F}$ as a minor.*

## Theorem (Robertson & Seymour)

*For any fixed graph $H$ there is polynomial time algorithm that given $G$ decides if $H$ is a minor of $G$. The running time is $O(|V(G)|^3)$.*

- Implies *existence* of an efficient algorithm for testing *any* minor-closed property.

Running time improved to $O(|V(G)|^2)$ by [Kawarabayashi-Kobayashi-Reed '12].

# Computing the Euler genus of a graph

Given $n$-vertex graph $G$, decide whether $\mathrm{eg}(G) \leq g$, for some $g \geq 0$.

# Computing the Euler genus of a graph

Given $n$-vertex graph $G$, decide whether $\mathrm{eg}(G) \leq g$, for some $g \geq 0$.

- $O(n^3 \cdot f(g))$-time [Robertson & Seymour]

# Computing the Euler genus of a graph

Given $n$-vertex graph $G$, decide whether $\mathrm{eg}(G) \leq g$, for some $g \geq 0$.

- $O(n^3 \cdot f(g))$-time [Robertson & Seymour]
- $O(n \cdot f'(g))$-time [Mohar '99]

# Computing the Euler genus of a graph

Given $n$-vertex graph $G$, decide whether $\mathrm{eg}(G) \leq g$, for some $g \geq 0$.

- $O(n^3 \cdot f(g))$-time [Robertson & Seymour]
- $O(n \cdot f'(g))$-time [Mohar '99]
- $O(n \cdot 2^{O(g)})$-time [Kawarabayashi, Mohar & Reed '08]

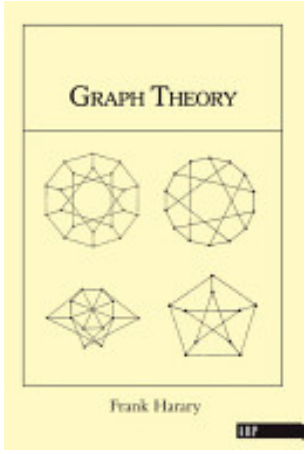
# Computing the Euler genus of a graph

Given $n$-vertex graph $G$, decide whether $\mathrm{eg}(G) \leq g$, for some $g \geq 0$.

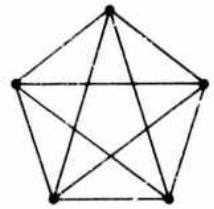
- $O(n^3 \cdot f(g))$-time [Robertson & Seymour]
- $O(n \cdot f'(g))$-time [Mohar '99]
- $O(n \cdot 2^{O(g)})$-time [Kawarabayashi, Mohar & Reed '08]
- NP-hard [Thomassen '89]

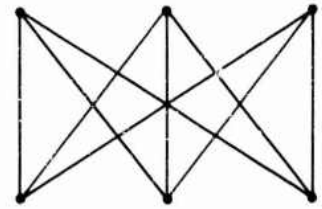
The exponential dependence on $g$ is necessary!

# Graphs and topology




*To KASIMIR KURATOWSKI*
*Who gave $K_5$ and $K_{3,3}$*
*To those who thought planarity*
*Was nothing but topology.*

$K_5$: 

$K_{3,3}$: 

# Approximating the Euler genus of a graph

**Question:** Given a graph $G$, how well can we *approximate* $\mathrm{eg}(G)$ in polynomial time?

An $\alpha$-approximation algorithm for $\mathrm{eg}(G)$, is a polynomial-time algorithm, that given $G$ outputs a drawing of $G$ into a surface $\mathcal{S}$ such that $\mathrm{eg}(\mathcal{S}) \leq \alpha \cdot \mathrm{eg}(G)$.

# Approximating the Euler genus of a graph

**Question:** Given a graph $G$, how well can we *approximate* $\mathrm{eg}(G)$ in polynomial time?

An $\alpha$-approximation algorithm for $\mathrm{eg}(G)$, is a polynomial-time algorithm, that given $G$ outputs a drawing of $G$ into a surface $\mathcal{S}$ such that $\mathrm{eg}(\mathcal{S}) \leq \alpha \cdot \mathrm{eg}(G)$.

- By Euler's characteristic,

$$|E(G)| \leq O(n) \cdot \mathrm{eg}(G) \leq O(n) \cdot |E(G)|$$

$\Rightarrow$ $O(n)$-approximation.

# Approximating the Euler genus of a graph

**Question:** Given a graph $G$, how well can we *approximate* $\mathrm{eg}(G)$ in polynomial time?

An $\alpha$-approximation algorithm for $\mathrm{eg}(G)$, is a polynomial-time algorithm, that given $G$ outputs a drawing of $G$ into a surface $\mathcal{S}$ such that $\mathrm{eg}(\mathcal{S}) \leq \alpha \cdot \mathrm{eg}(G)$.

- By Euler's characteristic,

$$|E(G)| \leq O(n) \cdot \mathrm{eg}(G) \leq O(n) \cdot |E(G)|$$

  $\Rightarrow$ $O(n)$-approximation.
- $O(1)$-approximation is not ruled out!

# Approximating the Euler genus of a graph

**Question:** Given a graph $G$, how well can we *approximate* $\mathrm{eg}(G)$ in polynomial time?

An $\alpha$-approximation algorithm for $\mathrm{eg}(G)$, is a polynomial-time algorithm, that given $G$ outputs a drawing of $G$ into a surface $\mathcal{S}$ such that $\mathrm{eg}(\mathcal{S}) \leq \alpha \cdot \mathrm{eg}(G)$.

- By Euler's characteristic,

$$|E(G)| \leq O(n) \cdot \mathrm{eg}(G) \leq O(n) \cdot |E(G)|$$

  $\Rightarrow$ $O(n)$-approximation.
- $O(1)$-approximation is not ruled out!

For bounded degree graphs:

- $O(n^{1/2})$-approximation [Chen, Kanchi & Kanevsky '97]

# A Pseudo-Approximation

**Theorem (Makarychev, Nayyeri & Sidiropoulos '12)**

*There is a polynomial-time algorithm, that given a Hamiltonial graph $G$ (along with a Hamilton path $P$) and integer $g$, either outputs a drawing of $G$ into a surface $\mathcal{S}$ such that* $\mathrm{eg}(\mathcal{S}) \leq O(g^{O(1)})$ *or correctly decides that* $\mathrm{eg}(G) > g$.

# Main Result

## Theorem (C, Sidiropolous '13)

*There is a polynomial-time algorithm, that given $G$ and integer $g$, either outputs a drawing of $G$ into a surface $\mathcal{S}$ such that* $\mathrm{eg}(\mathcal{S}) \leq O(\Delta^2 g^{12} \log^{19/2} n)$ *or correctly decides that* $\mathrm{eg}(G) > g$. *Here $\Delta$ is maximum degree and $n$ is number of nodes.*

# Main Result

## Theorem (C, Sidiropolous '13)

*There is a polynomial-time algorithm, that given $G$ and integer $g$, either outputs a drawing of $G$ into a surface $\mathcal{S}$ such that* $\mathrm{eg}(\mathcal{S}) \leq O(\Delta^2 g^{12} \log^{19/2} n)$ *or correctly decides that* $\mathrm{eg}(G) > g$. *Here $\Delta$ is maximum degree and $n$ is number of nodes.*

## Corollary

*An $O(n^{1/2-\delta})$-approximation for bounded-degree graphs.*

# Main Result

## Theorem (C, Sidiropolous '13)

*There is a polynomial-time algorithm, that given $G$ and integer $g$, either outputs a drawing of $G$ into a surface $\mathcal{S}$ such that* $\mathrm{eg}(\mathcal{S}) \leq O(\Delta^2 g^{12} \log^{19/2} n)$ *or correctly decides that* $\mathrm{eg}(G) > g$. *Here $\Delta$ is maximum degree and $n$ is number of nodes.*

## Corollary

*An $O(n^{1/2-\delta})$-approximation for bounded-degree graphs.*

# Beyond Euler genus

Further applications for bounded degree graphs.

- $g^{O(1)}$-approximation for orientable genus.

# Beyond Euler genus

Further applications for bounded degree graphs.

- $g^{O(1)}$-approximation for orientable genus.
- $k^{O(1)}$-approximation for crossing number.

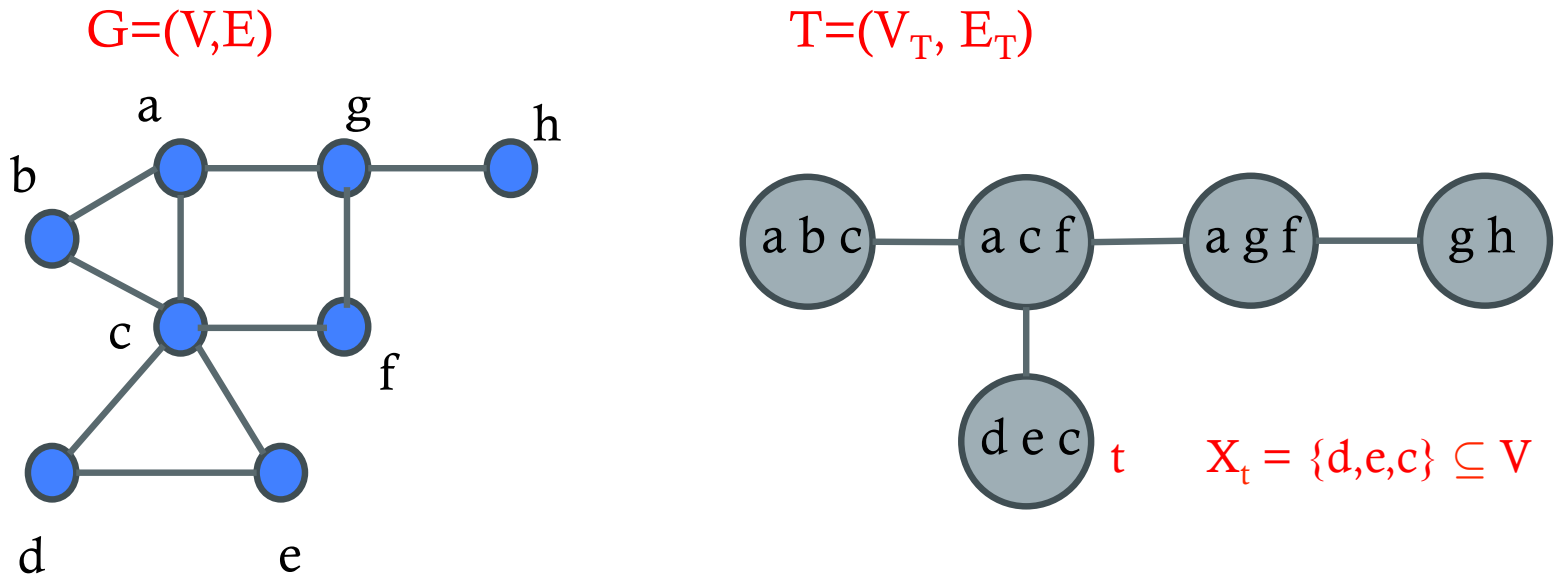
# Beyond Euler genus

Further applications for bounded degree graphs.

- $g^{O(1)}$-approximation for orientable genus.
- $k^{O(1)}$-approximation for crossing number.
- $k^{O(1)}$-approximation minimum edge/vertex planarization.

# Some details of the algorithm

# Tree decompositions and Treewidth




A *tree decomposition* of $G$ is a tree($T = (V_T, E_T)$ and collection of sets/bags $\{X_t \subseteq V(G) : t \in V_T\}$ such that

- $\cup_{t \in V_T} X_t = V(G)$.
- For every $\{u, v\} \in E(G)$, there is $t \in V_T$ s. t. $\{u, v\} \subseteq X_t$.
- For every $v \in V(G)$, the set $\{t \in V_T : v \in X_t\}$ forms a connected subtree of $T$.

The width of the decomposition is defined to be $\max_{t \in V_T} |X_t| - 1$.

# Tree decompositions and Treewidth

## Definition

The treewidth of a graph $G$, denoted by $\mathrm{tw}(G)$, is the minimum width of a tree decomposition for $G$.

# Tree decompositions and Treewidth

## Definition

The treewidth of a graph $G$, denoted by $\mathrm{tw}(G)$, is the minimum width of a tree decomposition for $G$.

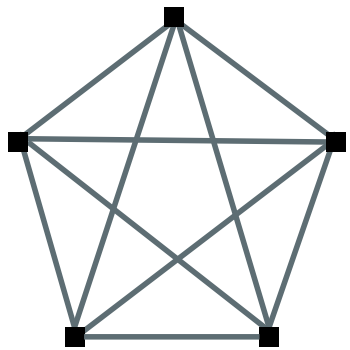
**Examples:**

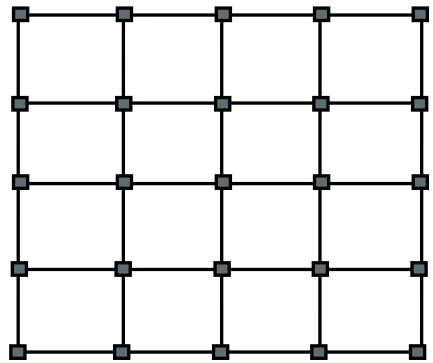
- For any tree $T$, $\mathrm{tw}(T) = 1$.
- For any cycle $C$, $\mathrm{tw}(C) = 2$.
- For a complete graph $K_n$, $\mathrm{tw}(K_n) = n - 1$.
- For any $(r \times r)$-grid $G$, $\mathrm{tw}(G) = \Theta(r)$. Thus planar graphs can have "large" treewidth.

# Treewidth / grid-minor duality

## Theorem (Robertson & Seymour '86)

*There is a function $f$ such that any graph $G$ with $\mathrm{tw}(G) \geq f(k)$ contains a clique of size $k$ or the $k \times k$ grid as a minor.*

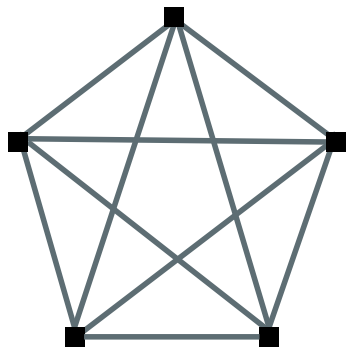


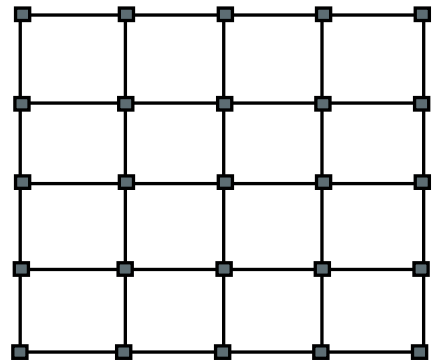


# Treewidth / grid-minor duality

## Theorem (Robertson & Seymour '86)

*There is a function $f$ such that any graph $G$ with* $\mathrm{tw}(G) \geq f(k)$ *contains a clique of size $k$ or the $k \times k$ grid as a minor.*





## Corollary (Grid-minor theorem)

$\mathrm{tw}(G) \geq f(k)$ *implies $G$ contains a $\sqrt{k} \times \sqrt{k}$ grid as a minor.*

The grid is a *canonical* obstruction for small treewidth.

# Treewidth / grid-minor duality: quantitative bounds

**Theorem (Robertson, Semour, Thomas '94)**

*Let $G$ be a graph such that $\mathrm{tw}(G) \geq 2^{O(k^5)}$ then $G$ contains a grid minor of size $k$.*
*If $G$ is a planar graph then $G$ contains a grid-minor of size $\mathrm{tw}(G)/6$.*

**Theorem (Demaine et al. '05)**

*Let $G$ be a graph of Euler genus $g$, then $G$ contains a "flat" grid minor of size $\Omega(\mathrm{tw}(G)/(g+1))$.*

**Theorem (C, Chuzhoy '13)**

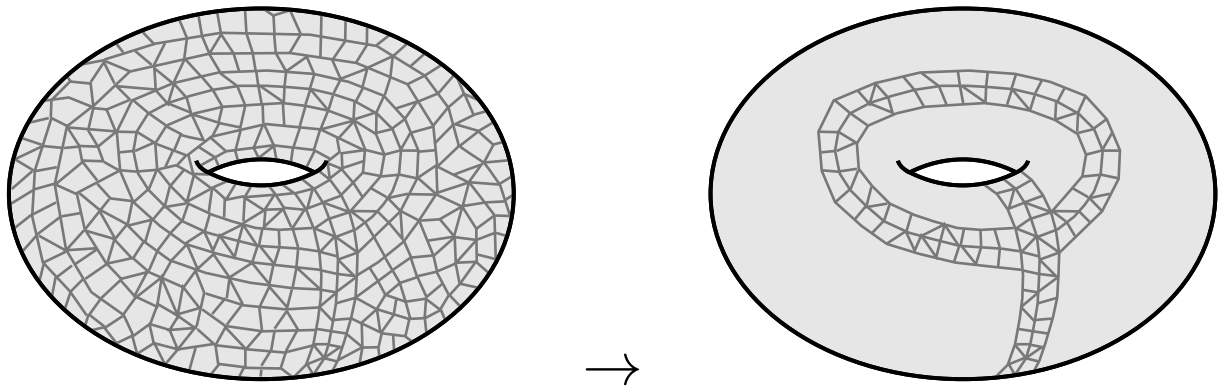
*Let $G$ be a graph such that $\mathrm{tw}(G) \geq k^{100}$ then $G$ contains a grid minor of size $k$.*

# Treewidth paradigm in algorithms

- If $G$ has "small" (constant) treewidth, solve problem efficiently or approximately. Running time typically depends exponentially on $\mathrm{tw}(G)$.
- If $G$ has "large" treewidth use obstruction given by structures such as grids

# High-level overview

Overview of exact algorithms.



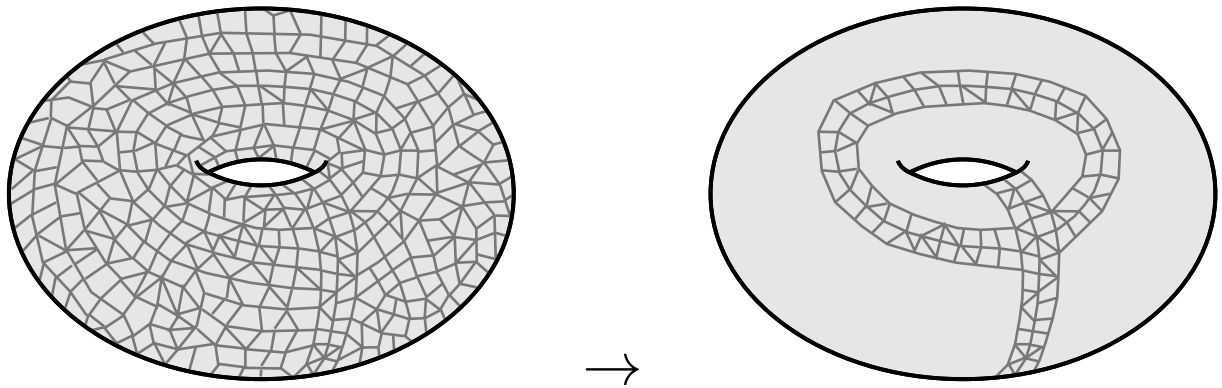
$G$

Given graph $G$ of high treewidth.

- Repeatedly remove "flat" parts.

# High-level overview

Overview of exact algorithms.



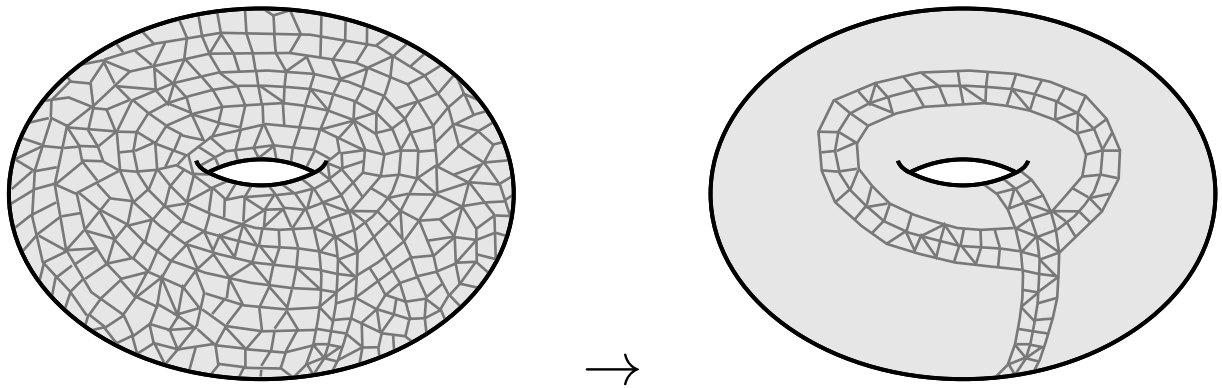
$G$ skeleton of $G$

Given graph $G$ of high treewidth.

- Repeatedly remove "flat" parts.
- Obtain "skeleton" $G'$ of "small" treewidth.

# High-level overview

Overview of exact algorithms.



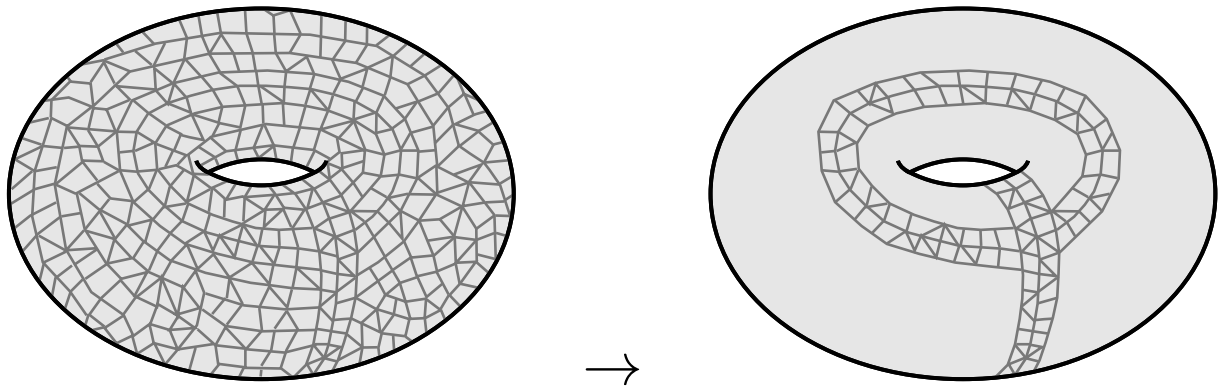
$G$ skeleton of $G$

Given graph $G$ of high treewidth.

- Repeatedly remove "flat" parts.
- Obtain "skeleton" $G'$ of "small" treewidth.
- Compute a drawing for the skeleton $G'$ exactly.

# High-level overview

Overview of exact algorithms.



$G$ skeleton of $G$

Given graph $G$ of high treewidth.

- Repeatedly remove "flat" parts.
- Obtain "skeleton" $G'$ of "small" treewidth.
- Compute a drawing for the skeleton $G'$ exactly.
- Extend the drawing to $G$.

# Flatness and embeddings

## Definition

Let $G$ be a graph, and let $H$ be a planar subgraph of $G$. We say that $H$ is *flat* if there exists a planar drawing of $H$, such that all edges $\{u, v\} \in E(G)$, with $u \in V(H)$, $v \in V(G) \setminus V(H)$, the vertex $u$ is on the outer face of $H$.

# Flatness and embeddings

## Definition

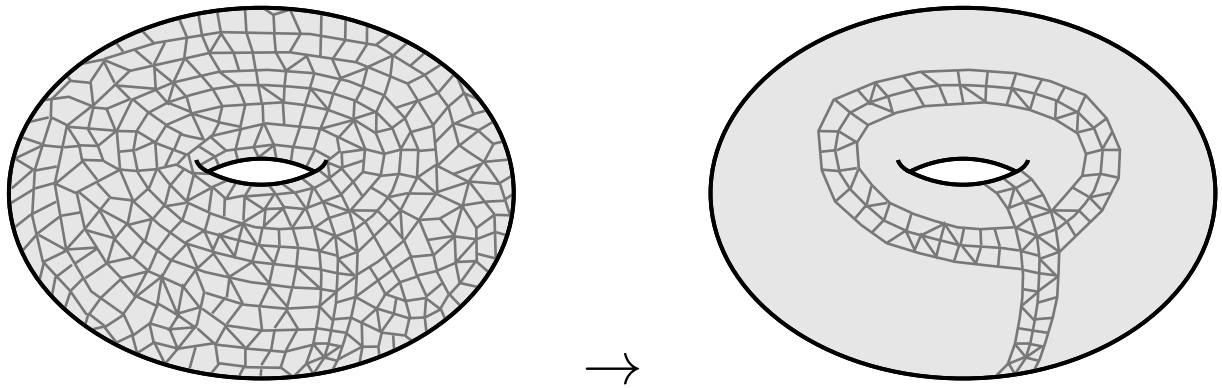
Let $G$ be a graph, and let $H$ be a planar subgraph of $G$. We say that $H$ is *flat* if there exists a planar drawing of $H$, such that all edges $\{u, v\} \in E(G)$, with $u \in V(H)$, $v \in V(G) \setminus V(H)$, the vertex $u$ is on the outer face of $H$.

## Theorem (Mohar '92)

*Let $G$ be a graph of Euler genus $g$, and let $H$ be a flat $(r \times r)$-grid minor $H$ in $G$, for some $r > 10 \cdot g$. Then, in any drawing of $G$ into a surface of genus $g$, the "central part" of $H$ is embedded inside a disk.*

# High-level overview

Overview of exact algorithms.



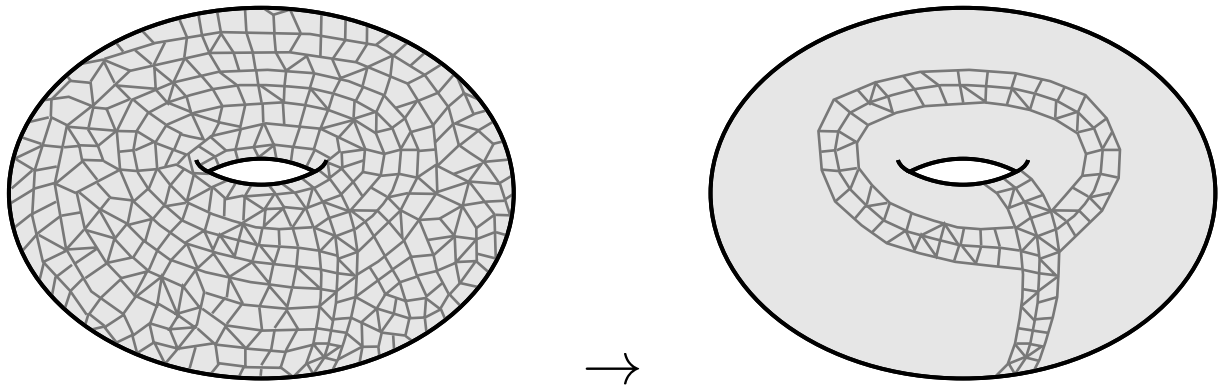
$G$ skeleton of $G$

Given graph $G$ of treewidth $\Omega(g^2)$

- Repeatedly remove "flat" parts: find flat grid of size $10g \times 10g$. Remove "middle" vertex $v$ of grid to obtain $G'$.

# High-level overview

Overview of exact algorithms.



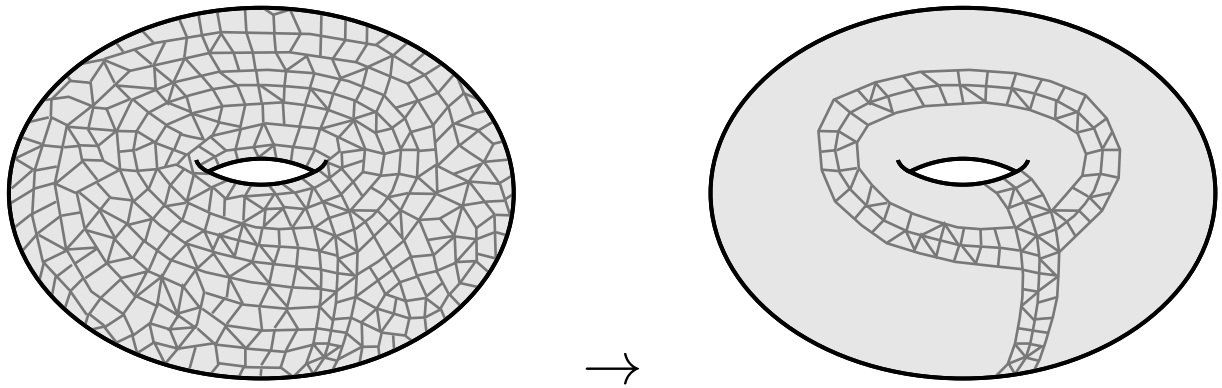
$G$ skeleton of $G$

Given graph $G$ of treewidth $\Omega(g^2)$

- Repeatedly remove "flat" parts: find flat grid of size $10g \times 10g$. Remove "middle" vertex $v$ of grid to obtain $G'$.
- Recursively find an embedding of $G'$.

# High-level overview

Overview of exact algorithms.



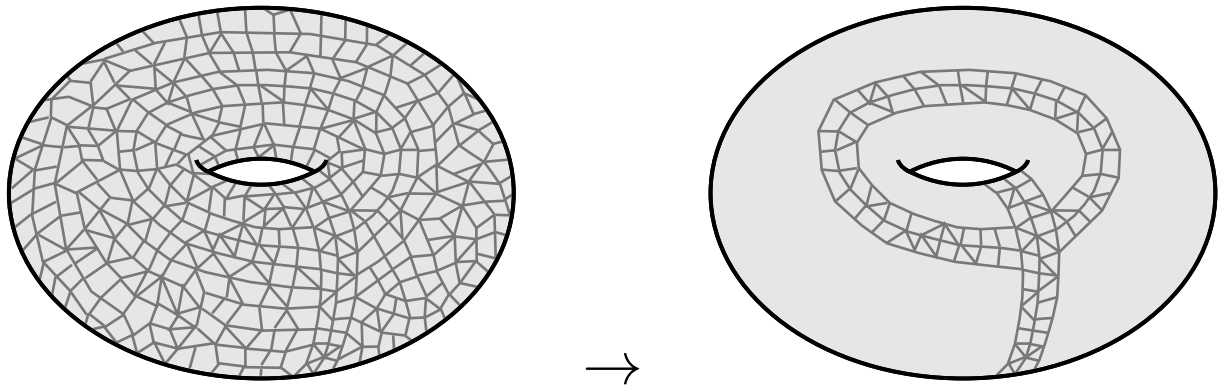
$G$ skeleton of $G$

Given graph $G$ of treewidth $\Omega(g^2)$

- Repeatedly remove "flat" parts: find flat grid of size $10g \times 10g$. Remove "middle" vertex $v$ of grid to obtain $G'$.
- Recursively find an embedding of $G'$.
- Extend the drawing to $G$ by placing $v$ in the middle of grid.

# High-level overview

Overview of exact algorithms.

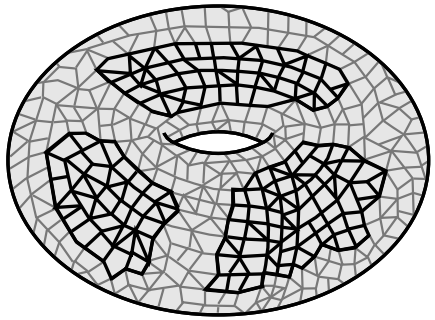


$G$ skeleton of $G$

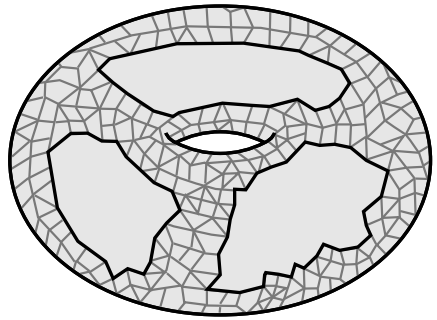
Given graph $G$ of treewidth $\Omega(g^2)$

- Repeatedly remove "flat" parts: find flat grid of size $10g \times 10g$. Remove "middle" vertex $v$ of grid to obtain $G'$.
- Recursively find an embedding of $G'$.
- Extend the drawing to $G$ by placing $v$ in the middle of grid.
- Recursion stops when $\mathrm{tw}(G)$ is "small", $\mathrm{tw}(G) = O(g^2)$. Compute a drawing exactly in time $f(g)\mathrm{poly}(n)$.

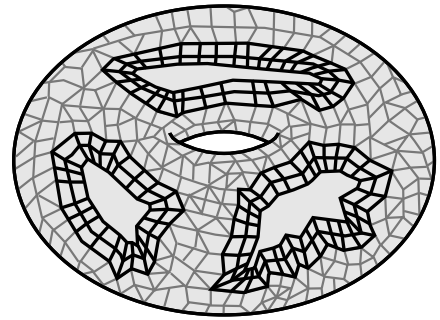
# High-level overview or our algorithm



$G$



approximate skeleton

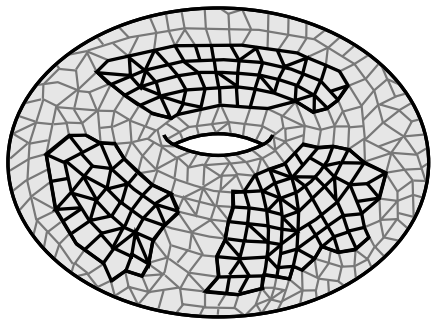


rigid skeleton

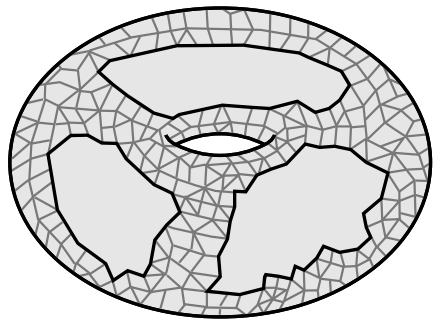
Given graph $G$ of high treewidth.

- Repeatedly remove "flat" parts.

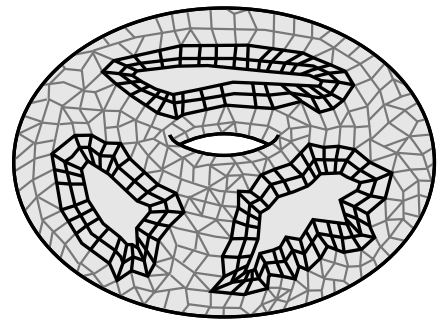
# High-level overview or our algorithm



$G$



approximate skeleton

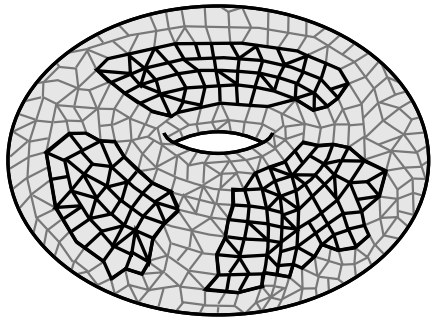


rigid skeleton

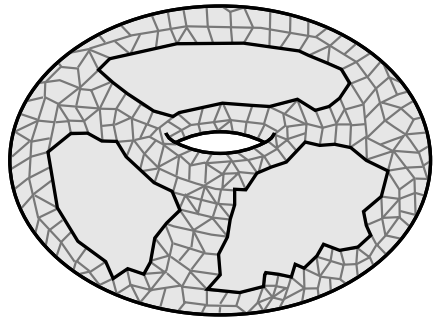
Given graph $G$ of high treewidth.

- Repeatedly remove "flat" parts.
- Obtain "skeleton" $G'$ of **small** treewidth.

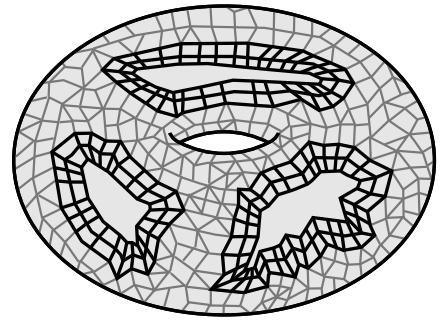
# High-level overview or our algorithm



$G$



approximate skeleton

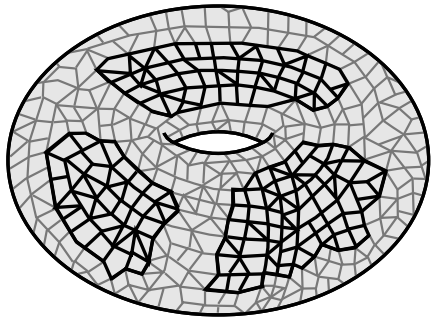


rigid skeleton

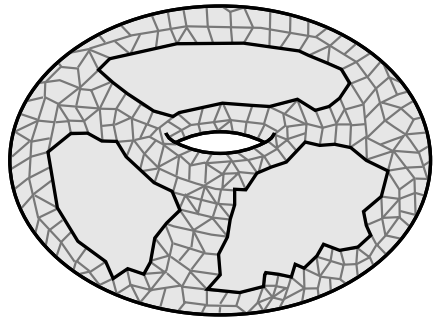
Given graph $G$ of high treewidth.

- Repeatedly remove "flat" parts.
- Obtain "skeleton" $G'$ of **small** treewidth.
- Compute a "rigid skeleton" $G''$.

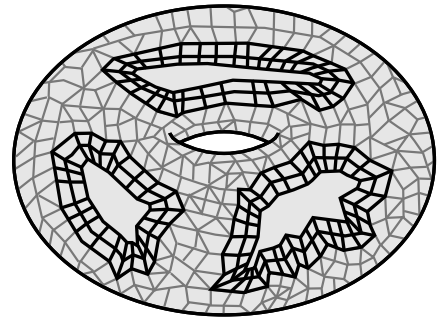
# High-level overview or our algorithm



$G$



approximate skeleton

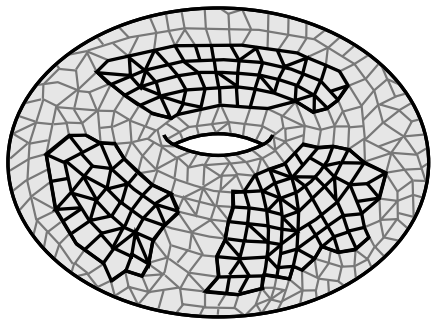


rigid skeleton

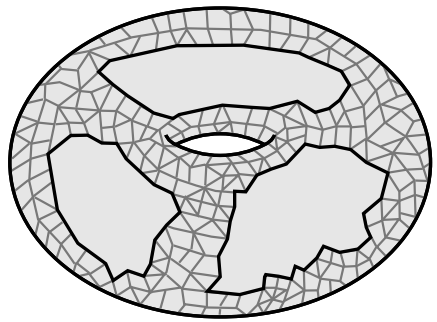
Given graph $G$ of high treewidth.

- Repeatedly remove "flat" parts.
- Obtain "skeleton" $G'$ of **small** treewidth.
- Compute a "rigid skeleton" $G''$.
- Compute a drawing for $G''$ **approximately**.

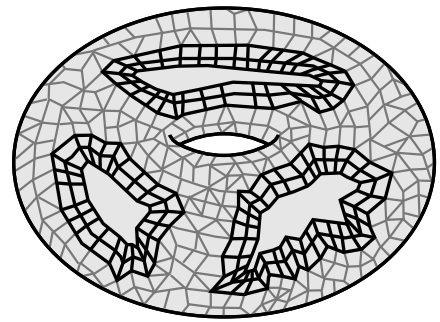
# High-level overview or our algorithm



$G$



approximate skeleton



rigid skeleton

Given graph $G$ of high treewidth.

- Repeatedly remove "flat" parts.
- Obtain "skeleton" $G'$ of **small** treewidth.
- Compute a "rigid skeleton" $G''$.
- Compute a drawing for $G''$ **approximately**.
- Modify the drawing to obtain a drawing for $G$.

# Planarizing graphs of small treewidth, and small genus

## Lemma

*There exists a polynomial time algorithm which given a graph $G$ of treewidth $t$, and an integer $g \geq 0$, either correctly decides that* $\mathrm{eg}(G) > g$, *or it outputs a set $X \subseteq V(G)$, such that*

- $|X| = O(gt \log^{3/2} n)$.
- $G \setminus X$ *is planar.*

# Planarizing graphs of small treewidth, and small genus

Decompose $G$ recursively as follows:

- If $G$ is planar stop.
- Else, $\mathrm{tw}(G) \leq t$ implies *balanced separator* $S \subset V(G)$ of size $\leq t$. Find $S$ using approximation algorithm. $G - S$ leaves components $G_1, G_2, \ldots, G_h$ where $|V(G_i)| \leq 2n/3$ for each $i$.
- Decompose each $G_i$.

# Planarizing graphs of small treewidth, and small genus

Decompose $G$ recursively as follows:

- If $G$ is planar stop.
- Else, $\mathrm{tw}(G) \leq t$ implies *balanced separator* $S \subset V(G)$ of size $\leq t$. Find $S$ using approximation algorithm. $G - S$ leaves components $G_1, G_2, \ldots, G_h$ where $|V(G_i)| \leq 2n/3$ for each $i$.
- Decompose each $G_i$.

Bounding number of nodes removed:

- Depth of recursion is $O(\log n)$
- Number of internal nodes in recursion tree at any level $i$ is at most $g$. Otherwise $G$ has more than $g$ disjoint subgraphs that are not planar! Implies $\mathrm{eg}(G) > g$.
- Total number of nodes removed as separators is $O(gt \log n)$. Lose extra factors for approximation.

# Handling "small" treewidth case

## Lemma

*There exists a polynomial time algorithm which given a graph $G$ of treewidth $t$, and an integer $g \geq 0$, either correctly decides that* $\mathrm{eg}(G) > g$, *or it outputs a set $X \subseteq V(G)$, such that*

- $|X| = O(gt \log^{3/2} n)$.
- $G \setminus X$ *is planar.*

Algorithm for "small" treewidth: $\mathrm{tw}(G) = g^{O(1)}$

- Embed $G \setminus X$ in plane.
- Add a handle for each edge incident to $|X|$. Number of edges incident to $X$ is at most $\Delta|X|$.
- Thus, embedding into a surface of orientable genus $O(\Delta g^{O(1)} \mathrm{polylog}(n))$.

# Handling "small" treewidth case

## Lemma

*There exists a polynomial time algorithm which given a graph $G$ of treewidth $t$, and an integer $g \geq 0$, either correctly decides that* $\mathrm{eg}(G) > g$, *or it outputs a set $X \subseteq V(G)$, such that*

- $|X| = O(gt \log^{3/2} n)$.
- $G \setminus X$ *is planar.*

Algorithm for "small" treewidth: $\mathrm{tw}(G) = g^{O(1)}$

- Embed $G \setminus X$ in plane.
- Add a handle for each edge incident to $|X|$. Number of edges incident to $X$ is at most $\Delta|X|$.
- Thus, embedding into a surface of orientable genus $O(\Delta g^{O(1)} \mathrm{polylog}(n))$.

Caveat: we cannot use this directly

# Handling "large" treewidth

$\mathrm{tw}(G) = t > g^{10}$: want to find "flat" grid minor in $\mathrm{poly}(g, n)$ time.

# Handling "large" treewidth

$\mathrm{tw}(G) = t > g^{10}$: want to find "flat" grid minor in $\mathrm{poly}(g, n)$ time.

Idea:

- $G$ has a grid minor $H$ of size $\Omega(t/g) \times \Omega(t/g)$.
- From lemma can remove $\tilde{O}(gt)$ nodes $X$ such that $G' = G - X$ is planar.
- $G'$ will have large treewidth if we can show that $H - X$ has a large grid minor.
- Recover grid minor from $G'$.

# Persistence of grid minors

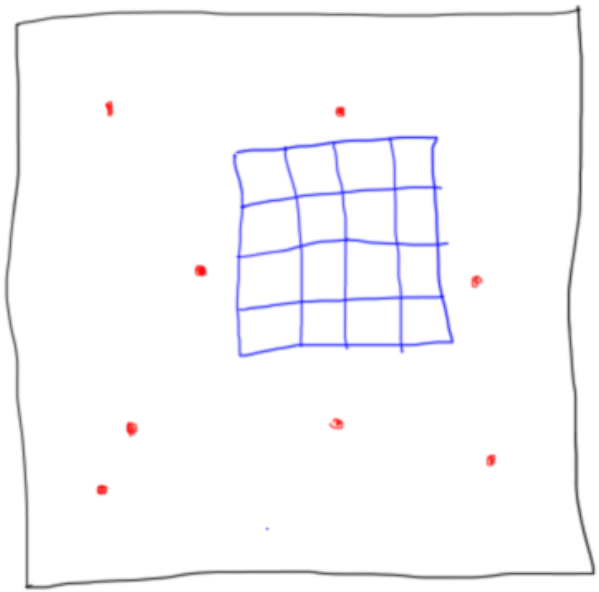
## Lemma (Eppstein '13)

*Let $r, f \geq 1$. Let $G$ be the $(r \times r)$-grid, and $X \subset V(G)$, with $|X| = f$. Then, $G \setminus X$ contains the $(r' \times r')$-grid as a minor, where $r' = \Theta(\min\{r, r^2/f\})$.*

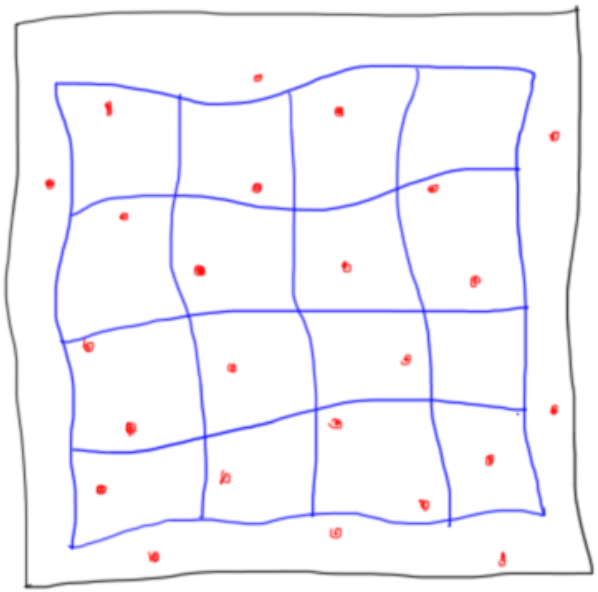
# Persistence of grid minors

## Lemma (Eppstein '13)

*Let $r, f \geq 1$. Let $G$ be the $(r \times r)$-grid, and $X \subset V(G)$, with $|X| = f$. Then, $G \setminus X$ contains the $(r' \times r')$-grid as a minor, where $r' = \Theta(\min\{r, r^2/f\})$.*



$f = O(r)$



$f = \Omega(r)$

# Grid minors and planarization

## Corollary (Chekuri, S '13)

*Let $G$ be a graph of Euler genus $g \geq 1$, and treewidth $t \geq 1$. There is a polynomial time algorithm to compute a set $X \subseteq V(G)$, with $|X| = (gt \log^{5/2} n)$, and a planar connected component of $G \setminus X$ containing the $(r' \times r')$-grid as a minor, with $r' = \Omega\left(\frac{t}{g^3 \log^{5/2} n}\right)$.*

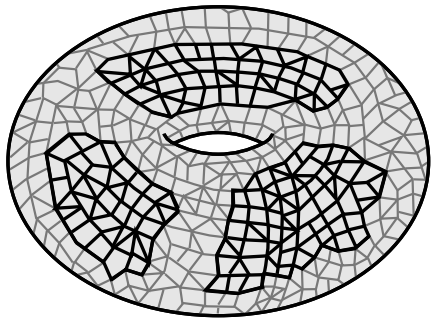
# Grid minors and planarization

## Corollary (Chekuri, S '13)

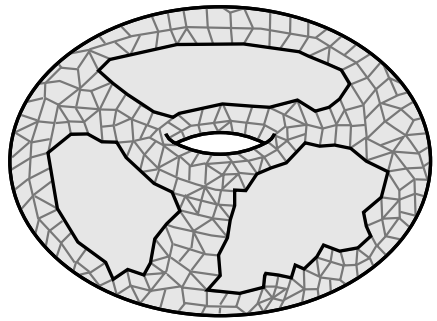
*Let $G$ be a graph of Euler genus $g \geq 1$, and treewidth $t \geq 1$. There is a polynomial time algorithm to compute a set $X \subseteq V(G)$, with $|X| = (gt \log^{5/2} n)$, and a planar connected component of $G \setminus X$ containing the $(r' \times r')$-grid as a minor, with $r' = \Omega\left(\frac{t}{g^3 \log^{5/2} n}\right)$.*

Thus if $t > g^{10}$ can find a flat grid minor of size $\tilde{\Omega}(g^7)$. This grid has to be rigidly embedded inside a disk in *any* embedding of $G$ into a surface of genus $\leq g$.

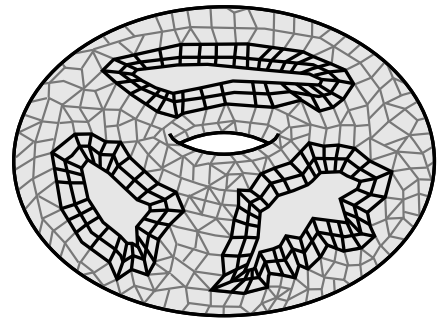
# Flat grid minors to Skeleton



$G$



approximate skeleton



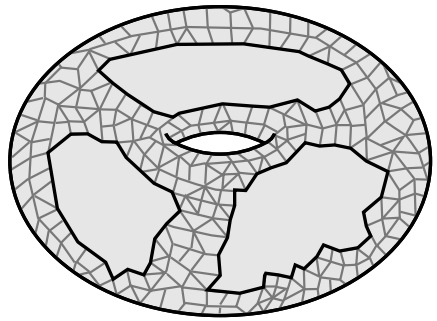
rigid skeleton

Given graph $G$ of high treewidth.

- Repeatedly remove "flat" grid minors.
- Obtain "skeleton" $G'$ of **small** treewidth.

To extend drawing of skeleton need to define it carefully.
Need to "merge" the multiple flat grid minors properly.

# Skeleton



- Removed parts form "patches" $(C_1, X_1), (C_2, X_2), \ldots, (C_k, X_k)$.
- Each patch $(C_i, X_i)$ consists of a set of nodes $X_i \subset V$ and a cycle $C_i \subset X_i$
- The patches are disjoint
- In *any* drawing of $G$ into a surface of Euler genus $g$, each patch $(C_i, X_i)$ has to be drawn in a disk with $C_i$ as its boundary.
- Skeleton that remains has "small" treewidth: $\tilde{O}(g^{O(1)})$.

# Making Skeleton rigid

- Embed skeleton $G'$ using the small treewidth algorithm
- Insert patches into the embedding of the skeleton

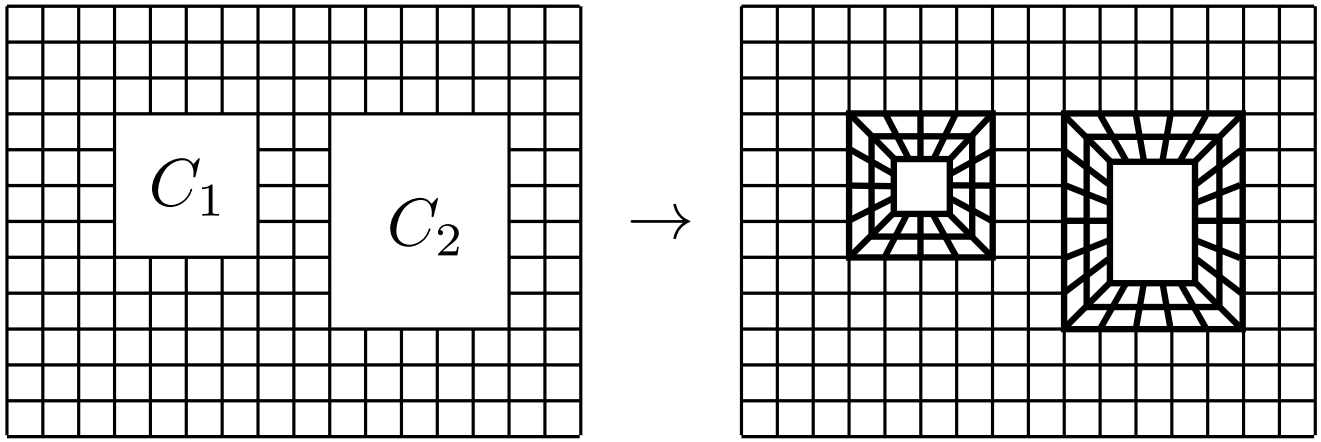
# Making Skeleton rigid

- Embed skeleton $G'$ using the small treewidth algorithm
- Insert patches into the embedding of the skeleton
- Problem: The cycles $C_i$ for the pathches may not be enclose a disk in the embedding of $G'$ since embedding is not into a genus $g$ surface.

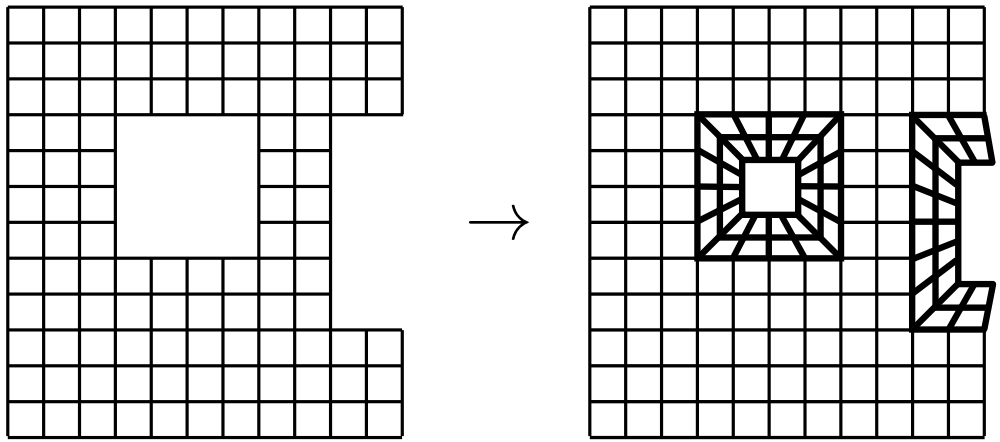
# Making Skeleton rigid

- Embed skeleton $G'$ using the small treewidth algorithm
- Insert patches into the embedding of the skeleton
- Problem: The cycles $C_i$ for the pathches may not be enclose a disk in the embedding of $G'$ since embedding is not into a genus $g$ surface.
- Our fix: Framing

# Framing




(a) $\{C_1, C_2\}$-Framing of a graph.



(b) $\{C_1, C_2\}$-Framing of a subgraph.

# Planarizing Skeleton

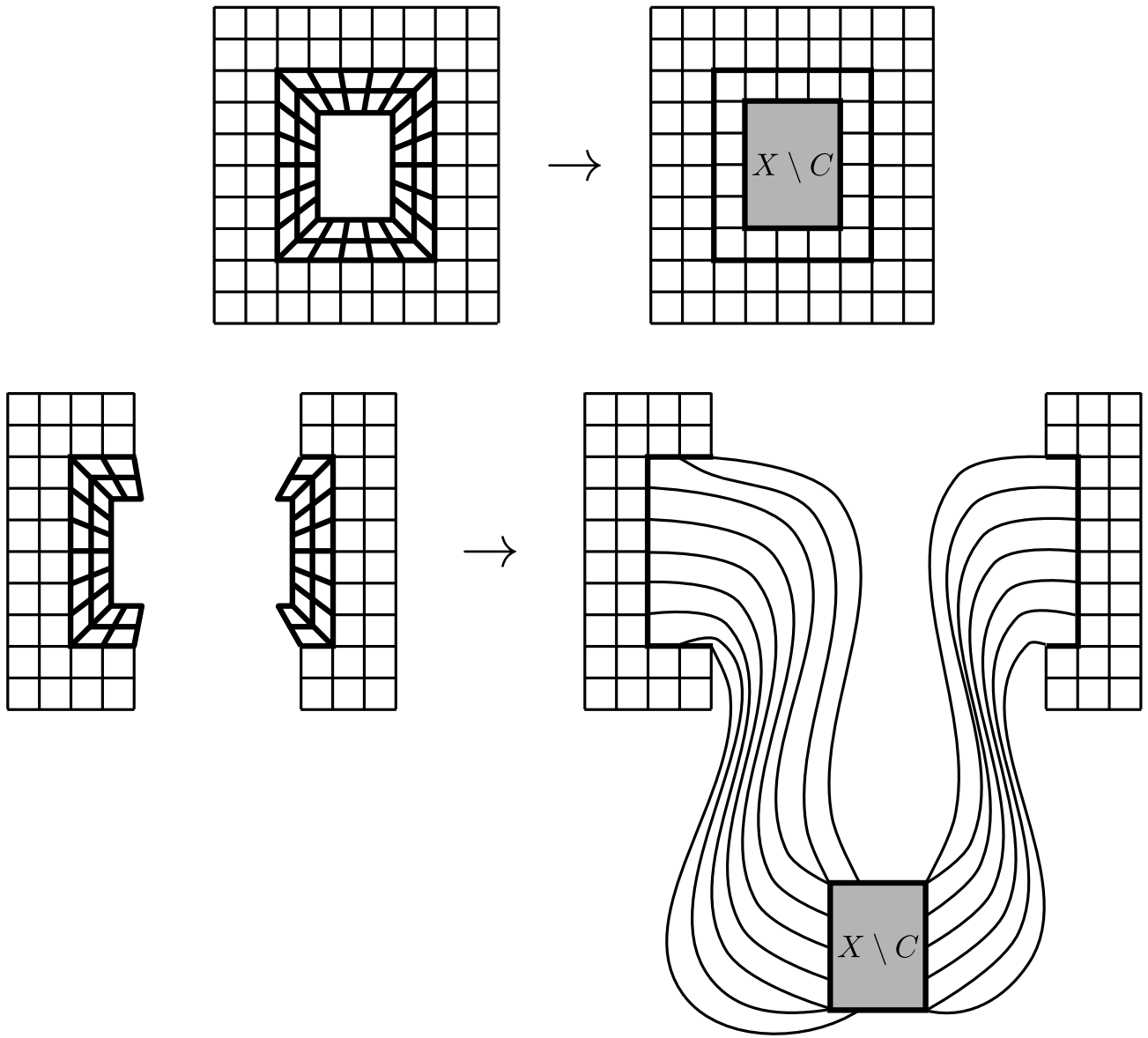
Skeleton has treewidth $\tilde{O}(g^{O(1)})$.

## Lemma

*Let $G'$ be skeleton of $G$ with patches $(X_1, C_1), \ldots, (X_r, C_r)$. The algorithm either correctly decides that $\mathrm{eg}(G) > g$, or outputs a set $X \subseteq V(G)$ s.t.*

- $|X| = O(\Delta g^{12} \log^{19/2} n)$.
- *For every connected component $H$ of $G' \setminus X$, the framing of $H$ is planar.*

# Extending embedding of Skeleton via Frames and Patches



# Beyond Euler genus

Further applications for bounded degree graphs.

- $g^{O(1)}$-approximation for orientable genus.
- $k^{O(1)}$-approximation for crossing number.
- $k^{O(1)}$-approximation minimum edge/vertex planarization.

Via a common framework that exploits the embedding given by the algorithm to approximate the Euler genus

# Representativity of face width of an embedding

### Definition

Let $\phi$ be an embedding of $G$ into a surface $\mathcal{S}$. A *noose* is a loop in $\mathcal{S}$ that only intersects $\phi(G)$ at $\phi(V(G))$. The length of a noose is the number of vertices it intersects.

### Definition

The *representativity* of $\phi$ is defined to be the smallest length of all noncontractible nooses in $\phi$.

# Approximating Orientable genus

Given $G$ and $g > 0$ is genus$(G) \leq g$?

# Approximating Orientable genus

Given $G$ and $g > 0$ is genus$(G) \leq g$?

- If genus$(G) \leq g$ then eg$(G) \leq 2g$
- Use approximation algorithm to embed $G$ into surface $\mathcal{S}$ such that eg$(\mathcal{S}) = \tilde{O}(g^{O(1)})$

# Approximating Orientable genus

Given $G$ and $g > 0$ is genus$(G) \leq g$?

- If genus$(G) \leq g$ then eg$(G) \leq 2g$
- Use approximation algorithm to embed $G$ into surface $\mathcal{S}$ such that eg$(\mathcal{S}) = \tilde{O}(g^{O(1)})$
- If $\mathcal{S}$ is orientable, done! Can check efficiently.

# Approximating Orientable genus

Given $G$ and $g > 0$ is genus$(G) \leq g$?

- If genus$(G) \leq g$ then eg$(G) \leq 2g$
- Use approximation algorithm to embed $G$ into surface $\mathcal{S}$ such that eg$(\mathcal{S}) = \tilde{O}(g^{O(1)})$
- If $\mathcal{S}$ is orientable, done! Can check efficiently.
- If $\mathcal{S}$ is non-orientable then there is an orientation reversing noose — find a shortest one, say of length $\ell$.

# Approximating Orientable genus

Given $G$ and $g > 0$ is genus$(G) \leq g$?

- If genus$(G) \leq g$ then eg$(G) \leq 2g$
- Use approximation algorithm to embed $G$ into surface $\mathcal{S}$ such that eg$(\mathcal{S}) = \tilde{O}(g^{O(1)})$
- If $\mathcal{S}$ is orientable, done! Can check efficiently.
- If $\mathcal{S}$ is non-orientable then there is an orientation reversing noose — find a shortest one, say of length $\ell$.
- If $\ell = \Omega(g^2)$ then *representativity* of the embedding into $\mathcal{S}$ is large. Implies that genus$(S) > g$.

# Approximating Orientable genus

Given $G$ and $g > 0$ is genus$(G) \leq g$?

- If genus$(G) \leq g$ then eg$(G) \leq 2g$
- Use approximation algorithm to embed $G$ into surface $\mathcal{S}$ such that eg$(\mathcal{S}) = \tilde{O}(g^{O(1)})$
- If $\mathcal{S}$ is orientable, done! Can check efficiently.
- If $\mathcal{S}$ is non-orientable then there is an orientation reversing noose — find a shortest one, say of length $\ell$.
- If $\ell = \Omega(g^2)$ then *representativity* of the embedding into $\mathcal{S}$ is large. Implies that genus$(S) > g$.
- Else $\ell < O(g^2)$, remove vertices of noose and $< \Delta\ell$ edges. Reduces genus by at least 1. Repeatedly do this until we have an orientable surface or no short noose. Total number of edges removed is $\tilde{O}(g^{O(1)})$. Add one handle for each edge.

# Some open problems

- Is there a $O(1)$-approximation for Euler genus?

# Some open problems

- Is there a $O(1)$-approximation for Euler genus?
- Can we remove the bounded-degree assumption in our algorithm?
- Improve the dependence on $g$?
- Which other minor-closed properties admit similar approximation algorithms? Can we approximate the largest clique-minor size?

Thank you!