Immersions and edge-deletion problems

Jean-Florent Raymond

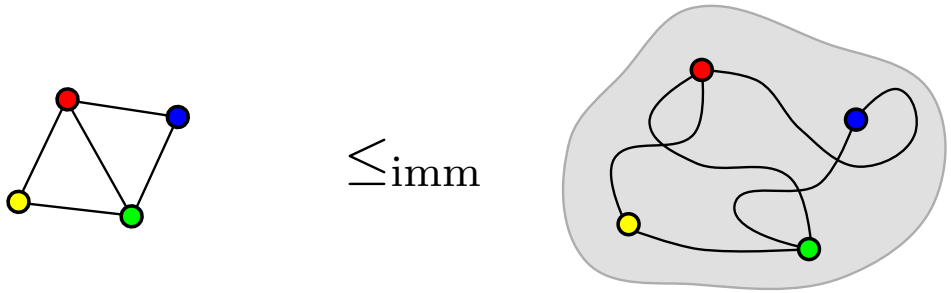
LIRMM and University of Warsaw

Joint work with Archontia Giannopoulou (TU Berlin), Michał Pilipczuk (University of Warsaw), Dimitrios M. Thilikos (LIRMM–CNRS), and Marcin Wrochna (University of Warsaw).

Immersions

H is an *immersion* of G if we can map

- vertices of H to distinct vertices of G;
- edges of H to edge-disjoint paths of G between the corresponding vertices.



What is an edge-deletion problem?

Question

Can I delete a few edges in my graph and get a nice property?

What is an edge-deletion problem?

Question

Can I delete a few edges in my graph and get a nice property?

The properties we consider: to be F-immersion-free

What is an edge-deletion problem?

Question

Can I delete a few edges in my graph and get a nice property?

The properties we consider: to be F-immersion-free

F-Immersion-Deletion

Input: a graph G and an integer k;
Question: can I remove k edges in G to get an F-immersion-free graph?

What is an edge-deletion problem?

Question

Can I delete a few edges in my graph and get a nice property?

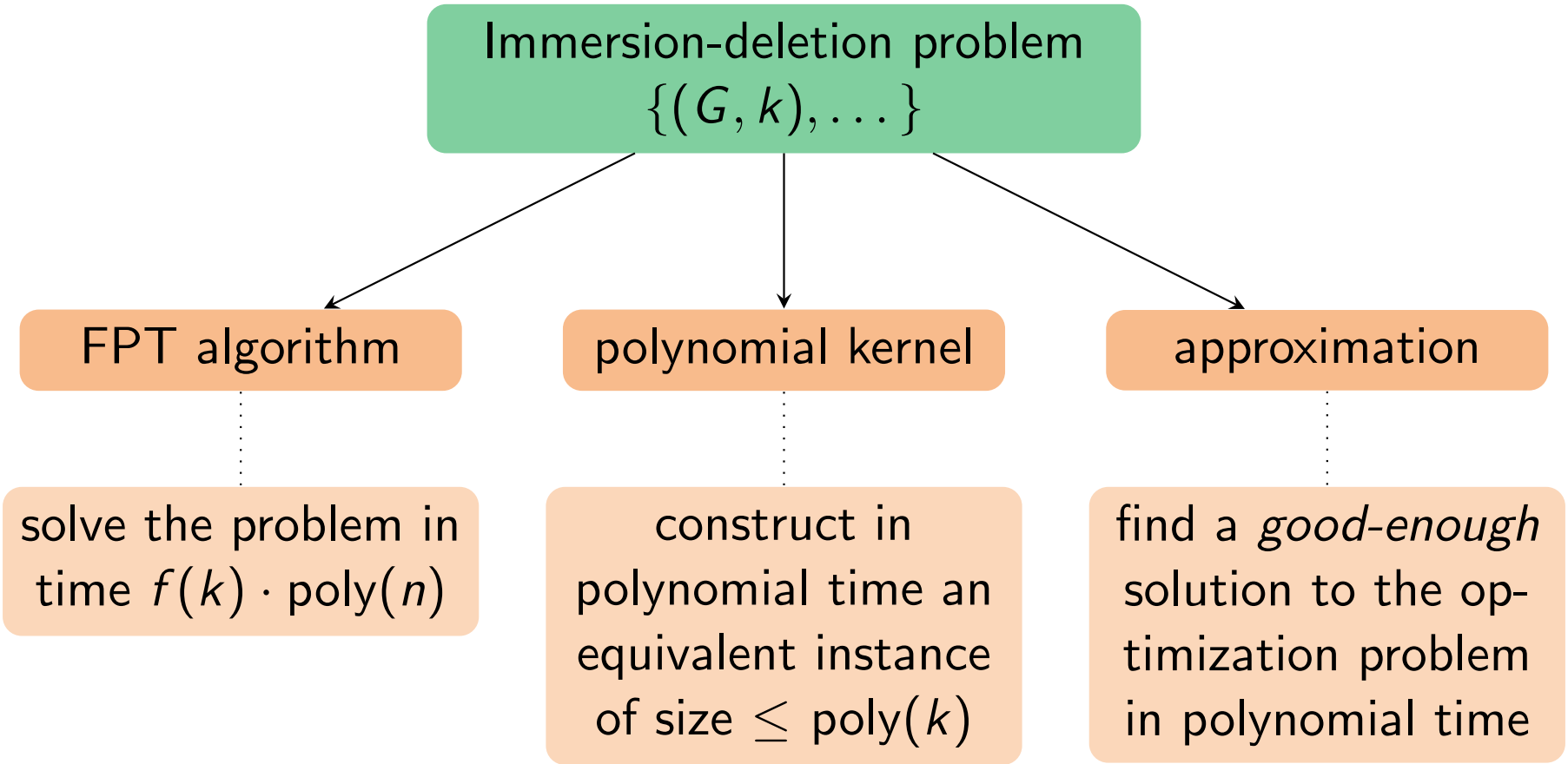
The properties we consider: to be F-immersion-free

F-Immersion-Deletion

Input: a graph G and an integer k;
Question: can I remove k edges in G to get an F-immersion-free graph?

Example: Feedback Arc Set.

The targets


Immersion-deletion problem
$\{(G, k), \ldots\}$
FPT algorithm
polynomial kernel
approximation
solve the problem in time $f(k) \cdot \text{poly}(n)$
construct in polynomial time an equivalent instance of size $\leq \text{poly}(k)$
find a *good-enough* solution to the optimization problem in polynomial time

A meta-theorem on immersion-deletion problems

F: connected planar subcubic graph (in all this talk)

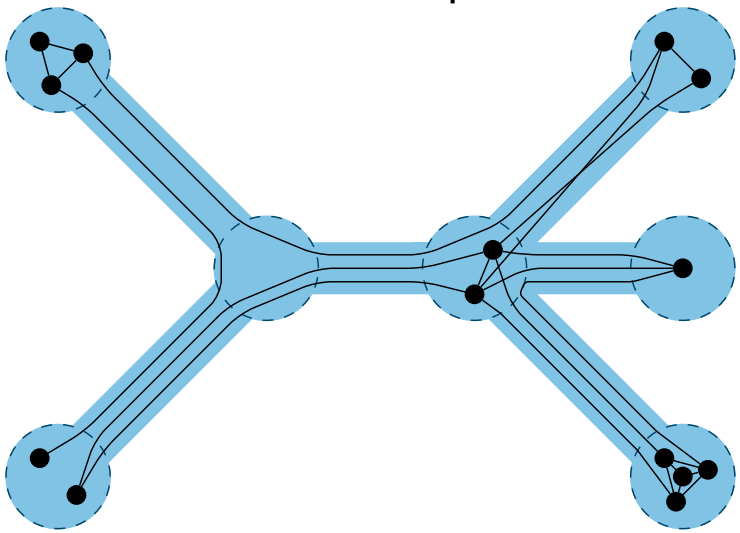
Theorem (Giannopoulou, Pilipczuk, Thilikos, R., Wrochna, 2016+)

F-Immersion-Deletion admits

- a constant factor approximation that runs in polynomial time;
- a $O(k)$-kernel
- (a single exponential FPT algorithm).

The tools for dealing with immersions

Tree-cut decomposition:



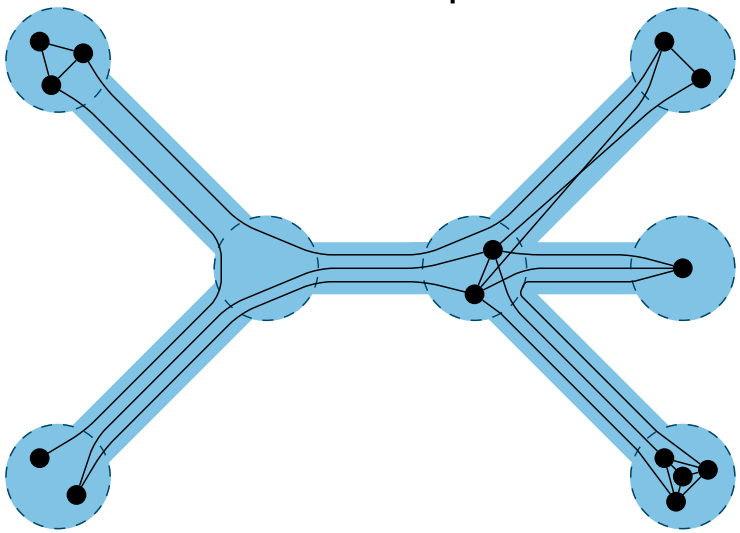
Associated parameter: **tcw**
(tree-cut width)

Small **tcw** implies:

- *small* bags;
- *thin* edges;
- small number of *thick* neighbors.

The tools for dealing with immersions

Tree-cut decomposition:



Associated parameter: **tcw**
(tree-cut width)

Small **tcw** implies:

- *small* bags;
- *thin* edges;
- small number of *thick* neighbors.

Theorem [Wollan, JCTB 2015] + [Chuzhoy STOC 2015]

For every planar subcubic graph F,

$$F \not\leq_{\text{imm}} G \quad \Rightarrow \quad \mathbf{tcw}(G) = O(|F|^{30}).$$

Approximation algorithm: structure of an instance

OPT: min size of a set of edges meeting all F-immersions (solution)

Approximation algorithm: structure of an instance

OPT: min size of a set of edges meeting all F-immersions (solution)

$G = F$-immersion-free graph $+$ OPT edges

Approximation algorithm: structure of an instance

OPT: min size of a set of edges meeting all F-immersions (solution)

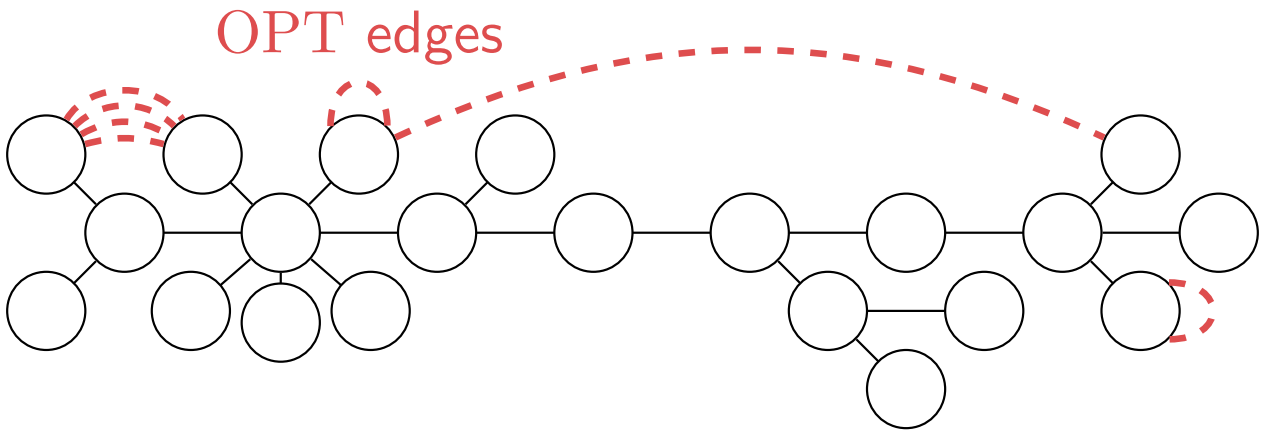
$$G = F\text{-immersion-free graph} + \mathrm{OPT}\text{ edges}$$
$$= \text{ graph of small } \mathbf{tcw} + \mathrm{OPT}\text{ edges}$$

Approximation algorithm: structure of an instance

OPT: min size of a set of edges meeting all F-immersions (solution)

G = F-immersion-free graph + OPT edges

= graph of small **tcw** + OPT edges

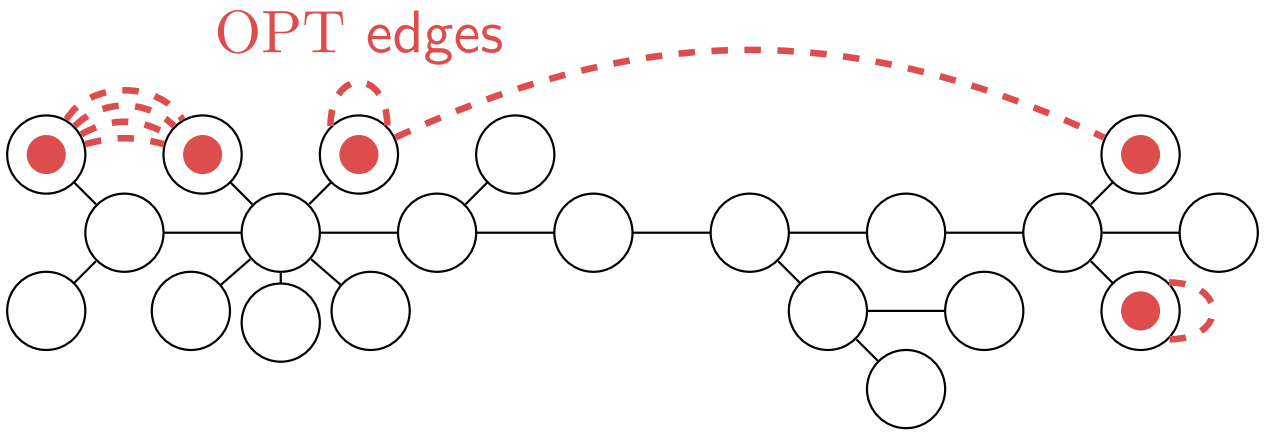


Approximation algorithm: structure of an instance

OPT: min size of a set of edges meeting all F-immersions (solution)

G = F-immersion-free graph + OPT edges

= graph of small **tcw** + OPT edges

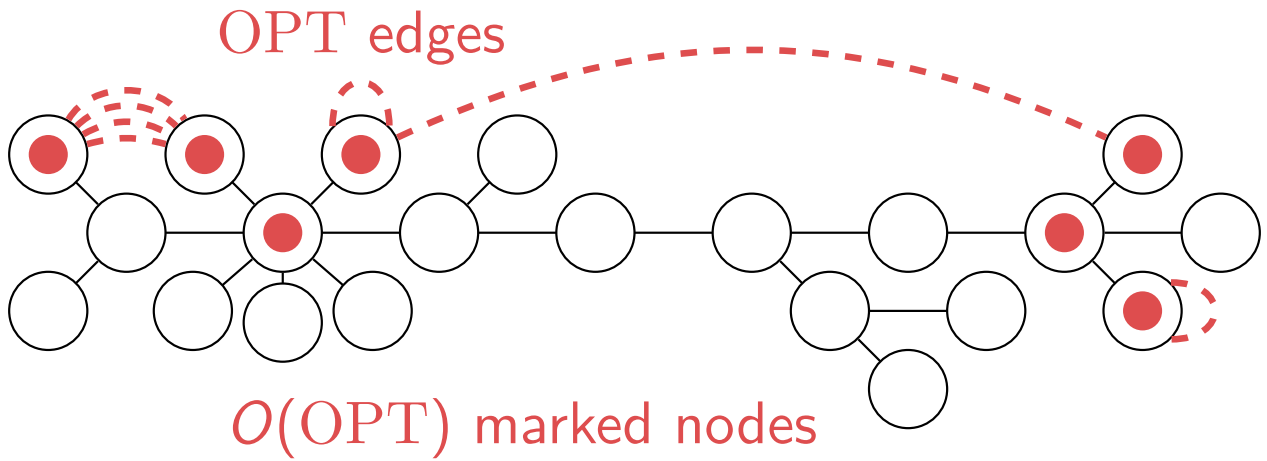


Approximation algorithm: structure of an instance

OPT: min size of a set of edges meeting all F-immersions (solution)

$G = F$-immersion-free graph + OPT edges

$=$ graph of small **tcw** + OPT edges

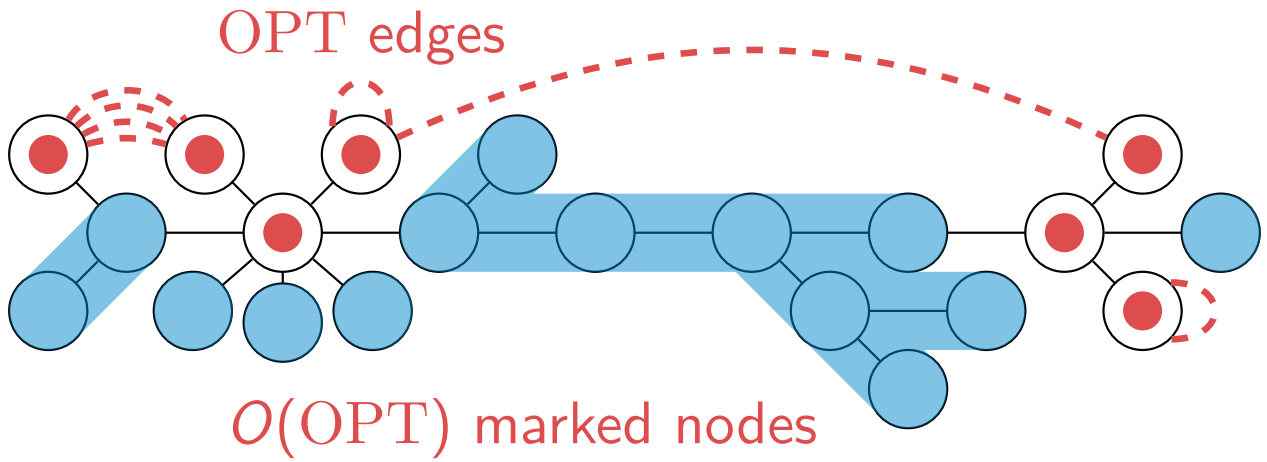


Approximation algorithm: structure of an instance

OPT: min size of a set of edges meeting all F-immersions (solution)

$G = F$-immersion-free graph + OPT edges

$=$ graph of small **tcw** + OPT edges

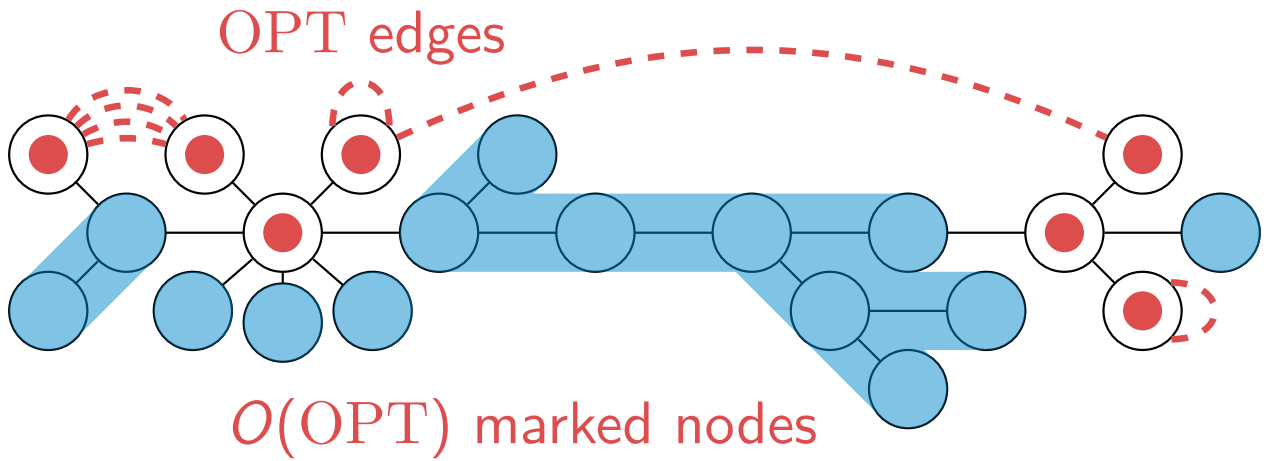


Approximation algorithm: structure of an instance

OPT: min size of a set of edges meeting all F-immersions (solution)

$G = F$-immersion-free graph + OPT edges

$=$ graph of small **tcw** + OPT edges



blue parts:

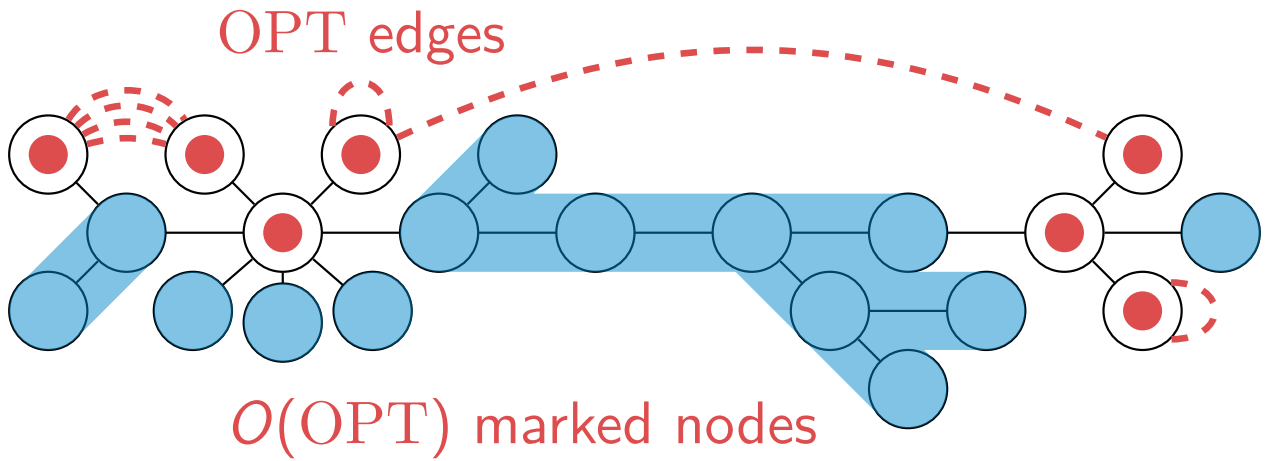
- small tree-cut width;

Approximation algorithm: structure of an instance

OPT: min size of a set of edges meeting all F-immersions (solution)

$G = F$-immersion-free graph + OPT edges

$=$ graph of small **tcw** + OPT edges



blue parts:

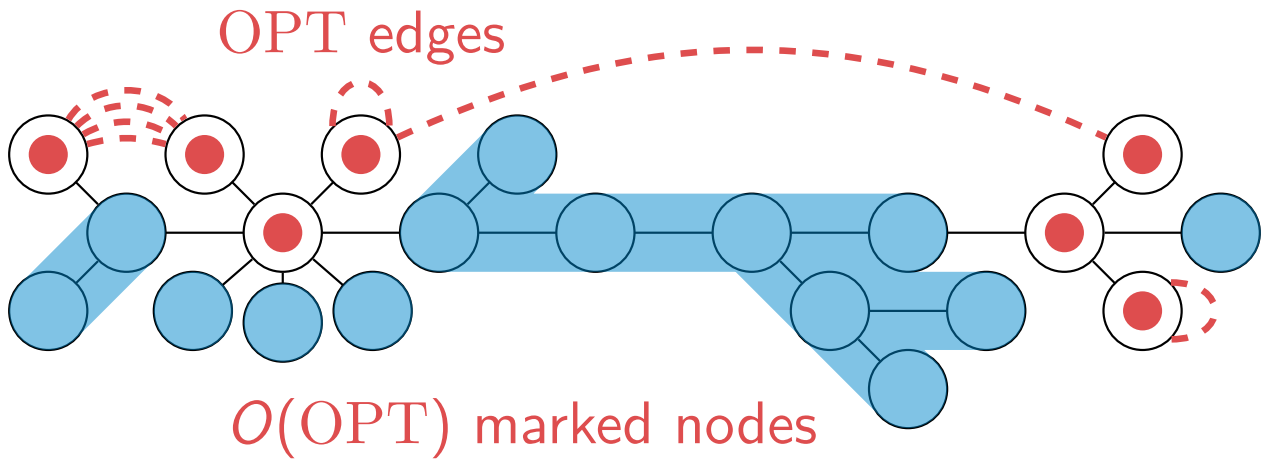
- small tree-cut width;
- small boundary;

Approximation algorithm: structure of an instance

OPT: min size of a set of edges meeting all F-immersions (solution)

$G = F$-immersion-free graph + OPT edges

$=$ graph of small **tcw** + OPT edges



blue parts:

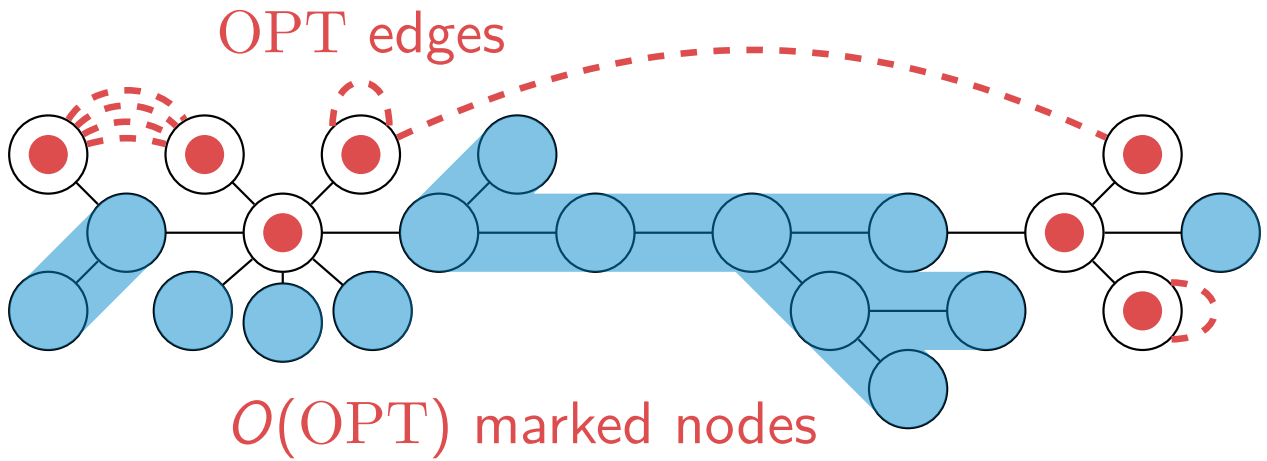
- small tree-cut width;
- small boundary;
- can be big;

Approximation algorithm: structure of an instance

OPT: min size of a set of edges meeting all F-immersions (solution)

$G = F$-immersion-free graph + OPT edges

$=$ graph of small **tcw** + OPT edges

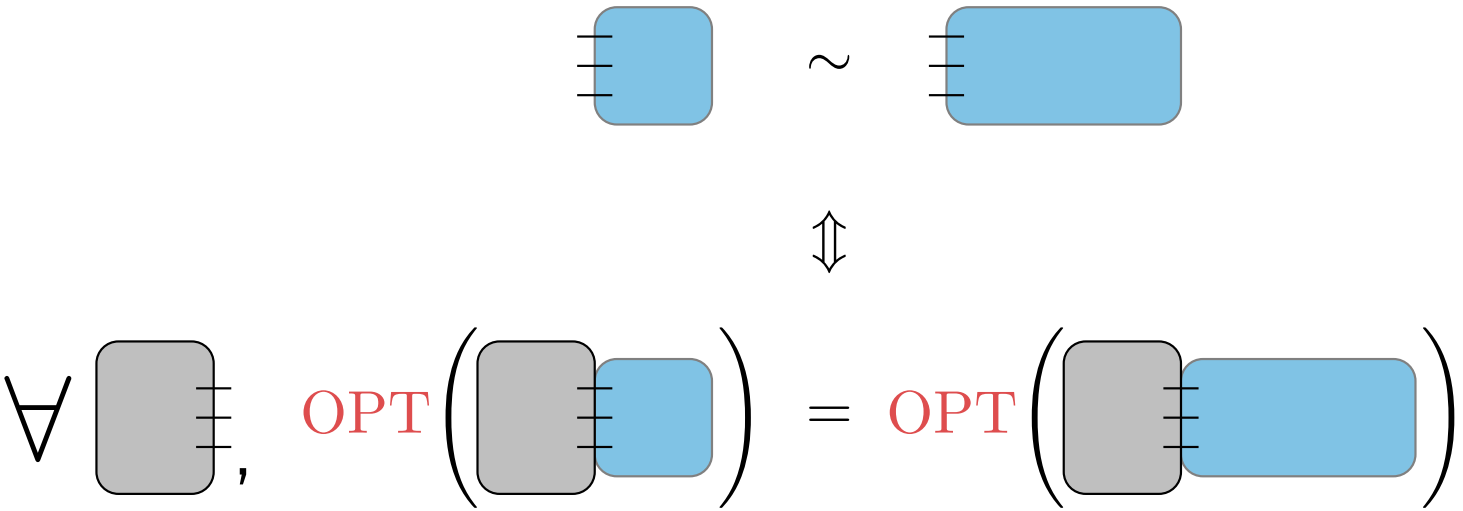


blue parts:

- small tree-cut width;
- small boundary;
- can be big;
- can be many.

Dealing with blue parts

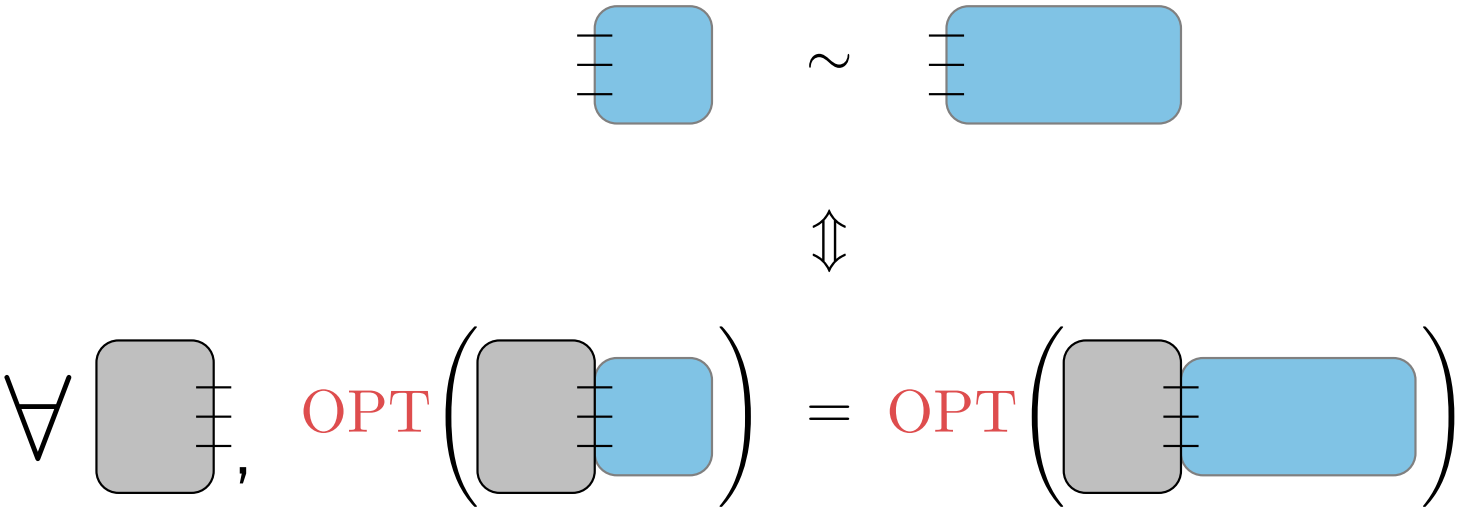
We define an equivalence relation on subgraphs with a small boundary:



Dealing with blue parts

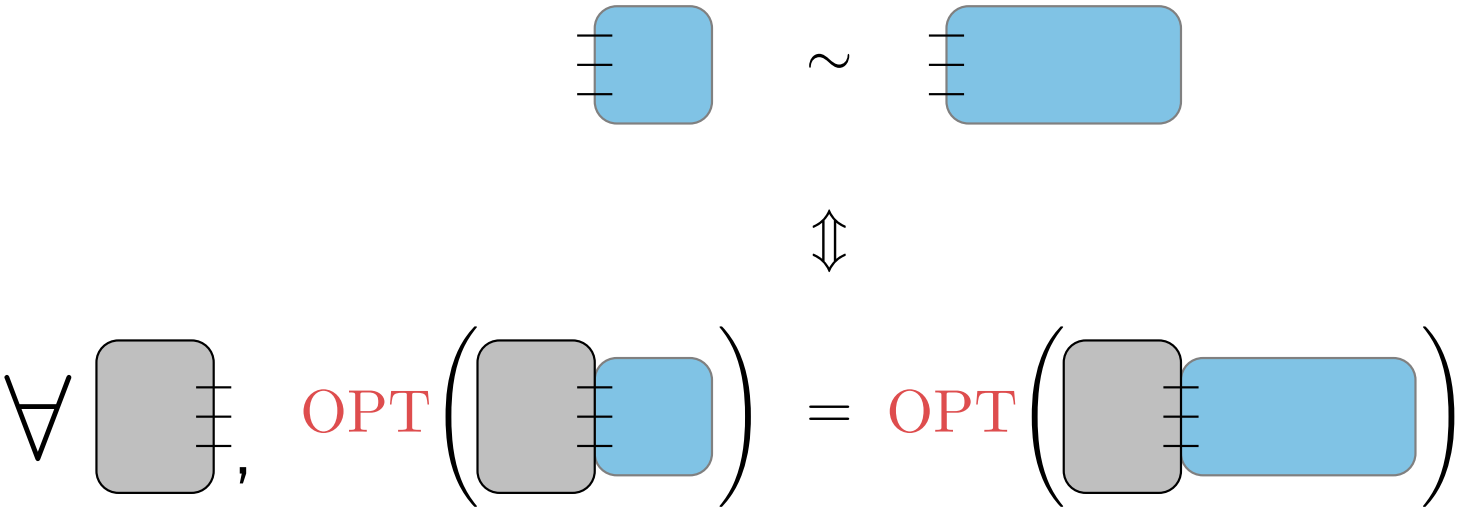
We define an equivalence relation on subgraphs with a small boundary:

(= protrusions)



Dealing with blue parts

We define an equivalence relation on subgraphs with a small boundary:
(= protrusions)

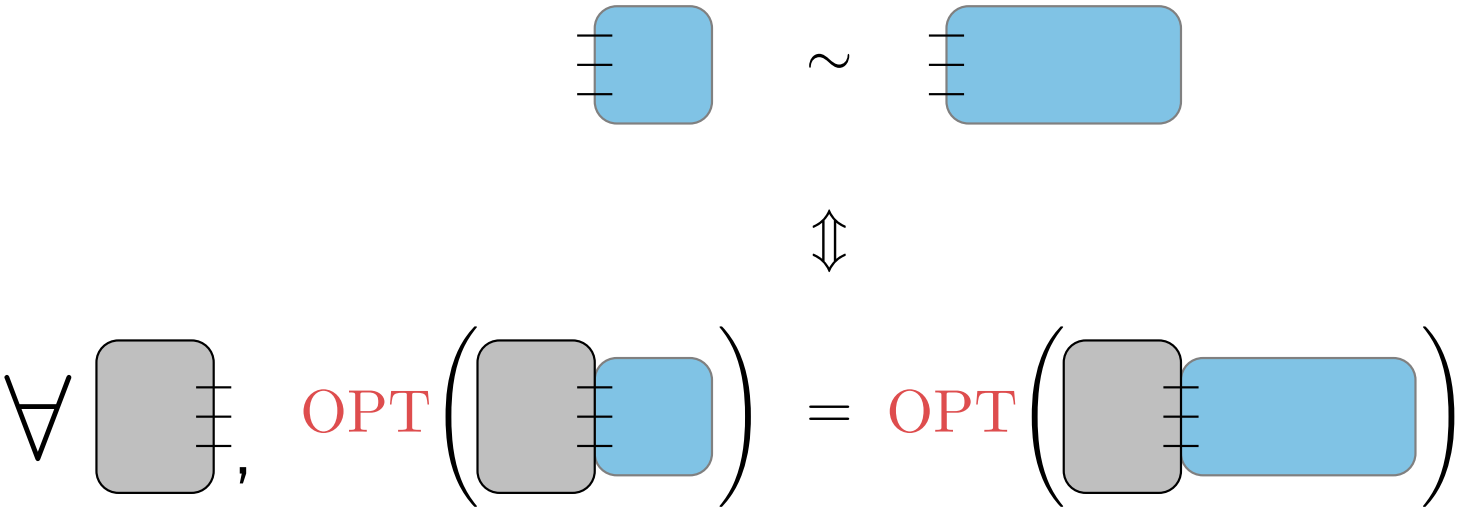


We show that, if we consider protrusions of small **tcw**:

- for each protrusion there is an equivalent protrusion of constant size;

Dealing with blue parts

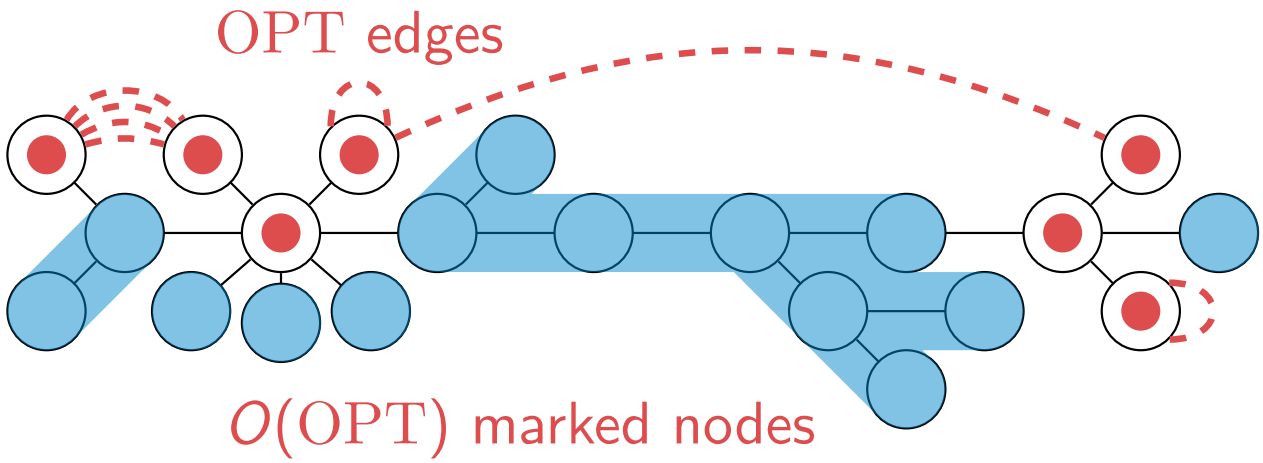
We define an equivalence relation on subgraphs with a small boundary:
(= protrusions)



We show that, if we consider protrusions of small **tcw**:

- for each protrusion there is an equivalent protrusion of constant size;
- we can find and *compress* protrusions in polynomial time.

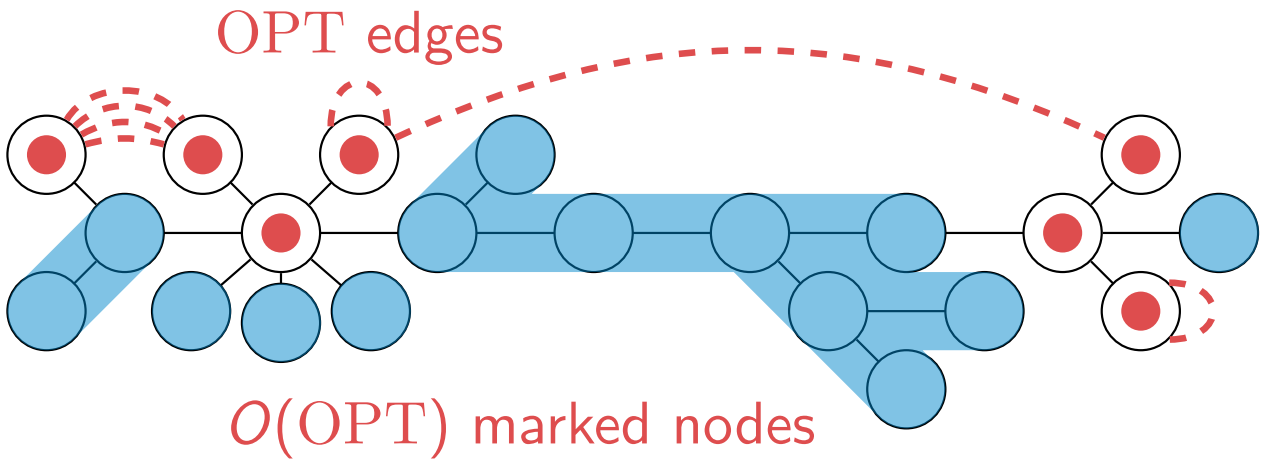
Back to our instance, after reduction



blue parts:

- small tree-cut width;
- small boundary;
- **constant size**;
- can be many.

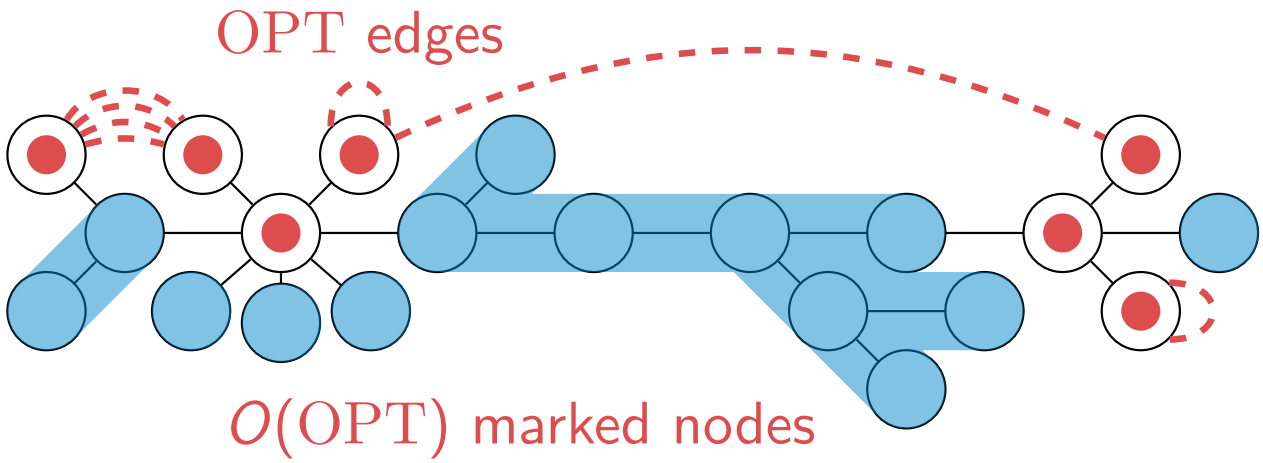
Back to our instance, after reduction



blue parts:

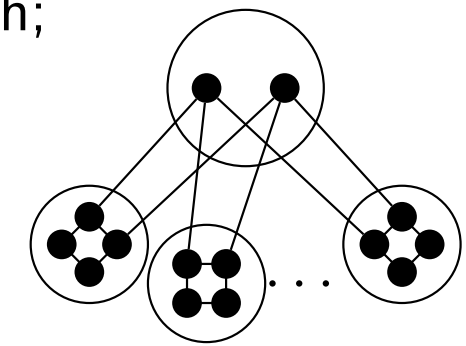
- small tree-cut width;
- small boundary;
- **constant size**;
- can be many.

Back to our instance, after reduction



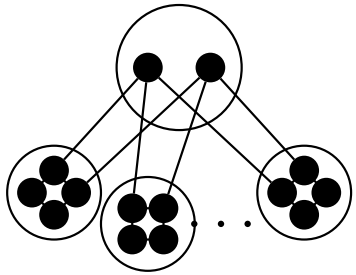
blue parts:

- small tree-cut width;
- small boundary;
- **constant size**;
- can be many.



Dealing with bouquets

a bouquet:



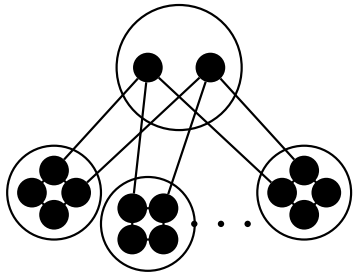
Lemma

We can find in polytime $Z \subseteq E(G)$ s.t., for some constant c,

$$\mathrm{OPT}(G \setminus Z) < \mathrm{OPT}(G) \qquad |Z| \leq c \cdot (\mathrm{OPT}(G) - \mathrm{OPT}(G \setminus Z)).$$

Dealing with bouquets

a bouquet:



Lemma

We can find in polytime $Z \subseteq E(G)$ s.t., for some constant c,

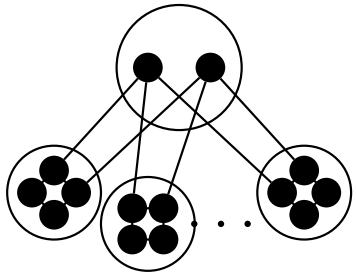
$$\mathrm{OPT}(G \setminus Z) < \mathrm{OPT}(G) \qquad |Z| \leq c \cdot (\mathrm{OPT}(G) - \mathrm{OPT}(G \setminus Z)).$$

How?

- delete all but a constant number of elements of each large bouquet;

Dealing with bouquets

a bouquet:



Lemma

We can find in polytime $Z \subseteq E(G)$ s.t., for some constant c,

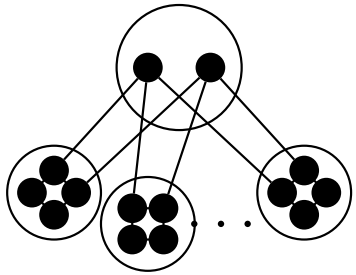
$$\mathrm{OPT}(G \setminus Z) < \mathrm{OPT}(G) \qquad |Z| \leq c \cdot (\mathrm{OPT}(G) - \mathrm{OPT}(G \setminus Z)).$$

How?

- delete all but a constant number of elements of each large bouquet;
- (this does not create large protrusions);

Dealing with bouquets

a bouquet:



Lemma

We can find in polytime $Z \subseteq E(G)$ s.t., for some constant c,

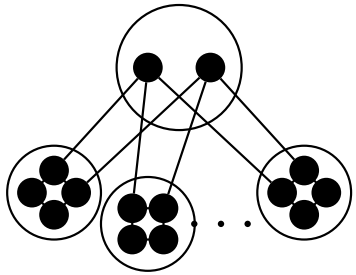
$$\mathrm{OPT}(G \setminus Z) < \mathrm{OPT}(G) \qquad |Z| \leq c \cdot (\mathrm{OPT}(G) - \mathrm{OPT}(G \setminus Z)).$$

How?

- delete all but a constant number of elements of each large bouquet;
- (this does not create large protrusions);
- Z is the set of edges of the resulting graph.

Dealing with bouquets

a bouquet:



Lemma

We can find in polytime $Z \subseteq E(G)$ s.t., for some constant c,

$$\mathrm{OPT}(G \setminus Z) < \mathrm{OPT}(G) \qquad |Z| \leq c \cdot (\mathrm{OPT}(G) - \mathrm{OPT}(G \setminus Z)).$$

How?

- delete all but a constant number of elements of each large bouquet;
- (this does not create large protrusions);
- Z is the set of edges of the resulting graph.

Iteratively removing this set of edges gives the desired approximation.

The approximation algorithm

Input: G

1. reduce all large protrusions;

The approximation algorithm

Input: G

1. reduce all large protrusions;
2. delete all but a constant number of elements of each large bouquet;

The approximation algorithm

Input: G

1. reduce all large protrusions;
2. delete all but a constant number of elements of each large bouquet;
3. find a $Z \subseteq E(G)$ as defined on previous slide;

The approximation algorithm

Input: G

1. reduce all large protrusions;
2. delete all but a constant number of elements of each large bouquet;
3. find a $Z \subseteq E(G)$ as defined on previous slide;
4. apply recursively on $G \setminus Z \rightsquigarrow$ approximate solution Z';

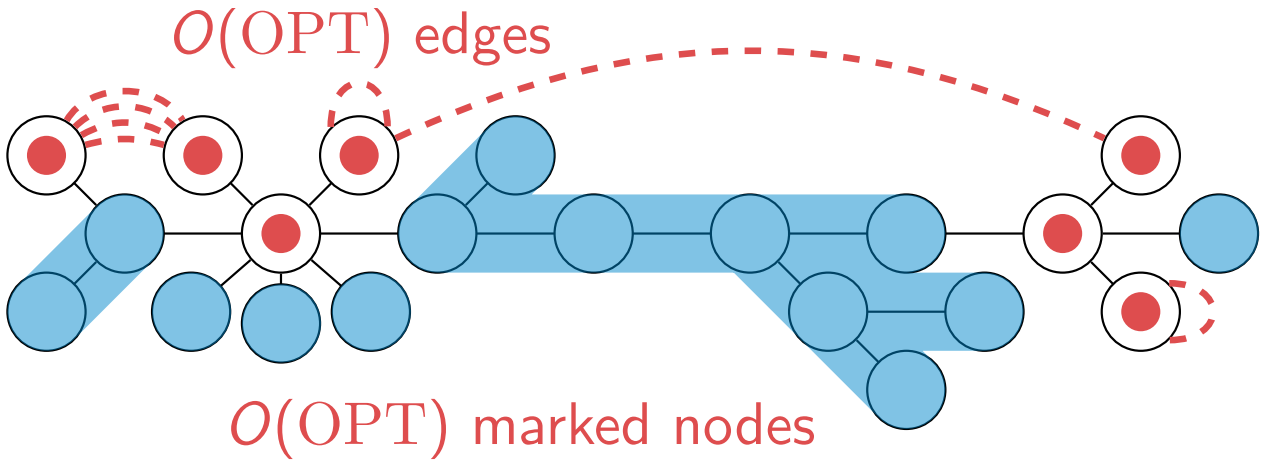
The approximation algorithm

Input: G

1. reduce all large protrusions;
2. delete all but a constant number of elements of each large bouquet;
3. find a $Z \subseteq E(G)$ as defined on previous slide;
4. apply recursively on $G \setminus Z \rightsquigarrow$ approximate solution Z';
5. output $Z \cup Z'$.

Kernelization

We consider an instance (G, k) with an approximate solution.

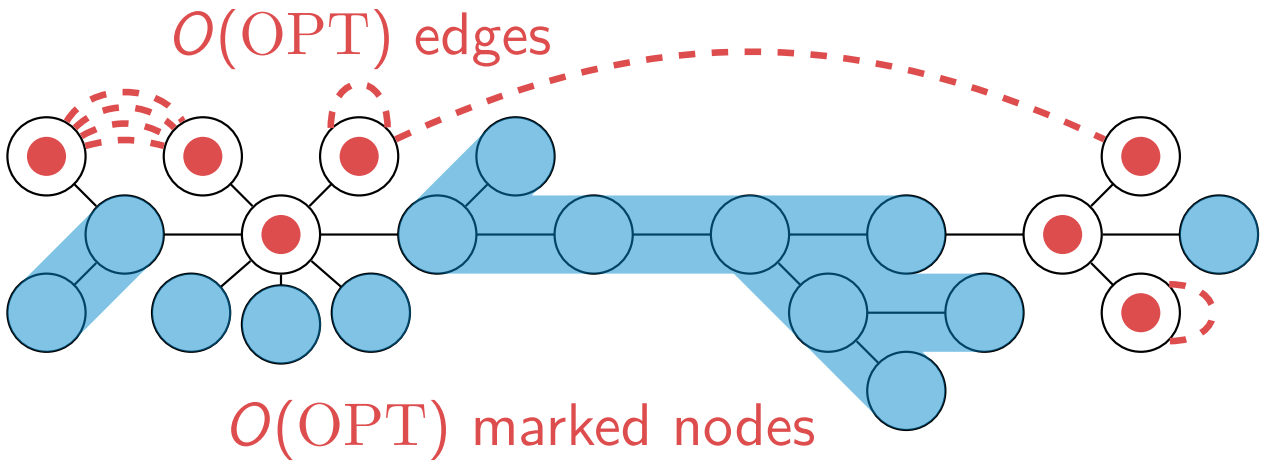


blue parts:

- small tree-cut width;
- small boundary;
- constant size;
- can be many.

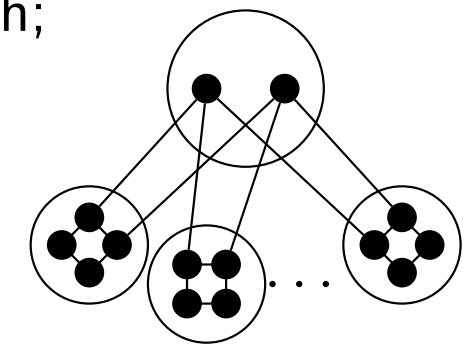
Kernelization

We consider an instance (G, k) with an approximate solution.

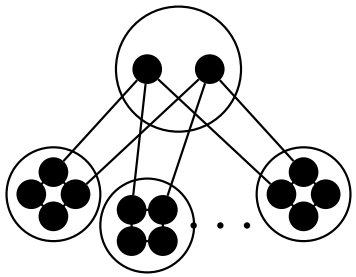


blue parts:

- small tree-cut width;
- small boundary;
- constant size;
- can be many.

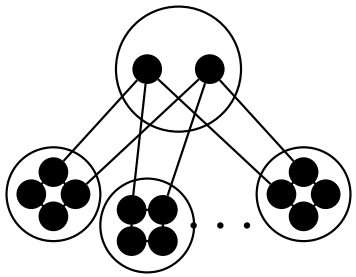


Dealing with bouquets (again)



$Z \subseteq E(G)$ *affects* a bouquet B if its removal disconnects B into F-free parts.

Dealing with bouquets (again)

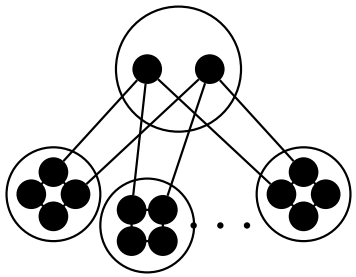


$Z \subseteq E(G)$ *affects* a bouquet B if its removal disconnects B into F-free parts.

Lemma

If Z affects B, then we can delete all but $O(|Z|)$ elements of B without changing OPT.

Dealing with bouquets (again)



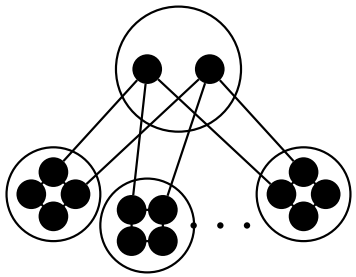
$Z \subseteq E(G)$ *affects* a bouquet B if its removal disconnects B into F-free parts.

Lemma

If Z affects B, then we can delete all but $O(|Z|)$ elements of B without changing OPT.

Note: the approximate solution affects every bouquet.

Dealing with bouquets (again)



$Z \subseteq E(G)$ *affects* a bouquet B if its removal disconnects B into F-free parts.

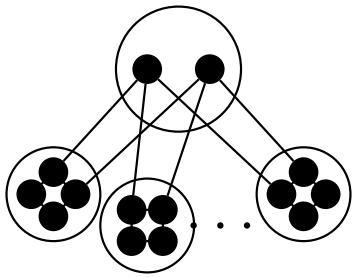
Lemma

If Z affects B, then we can delete all but $O(|Z|)$ elements of B without changing OPT.

Note: the approximate solution affects every bouquet.

Using the approximate solution, we can reduce the size of bouquets.

Dealing with bouquets (again)



$Z \subseteq E(G)$ *affects* a bouquet B if its removal disconnects B into F-free parts.

Lemma

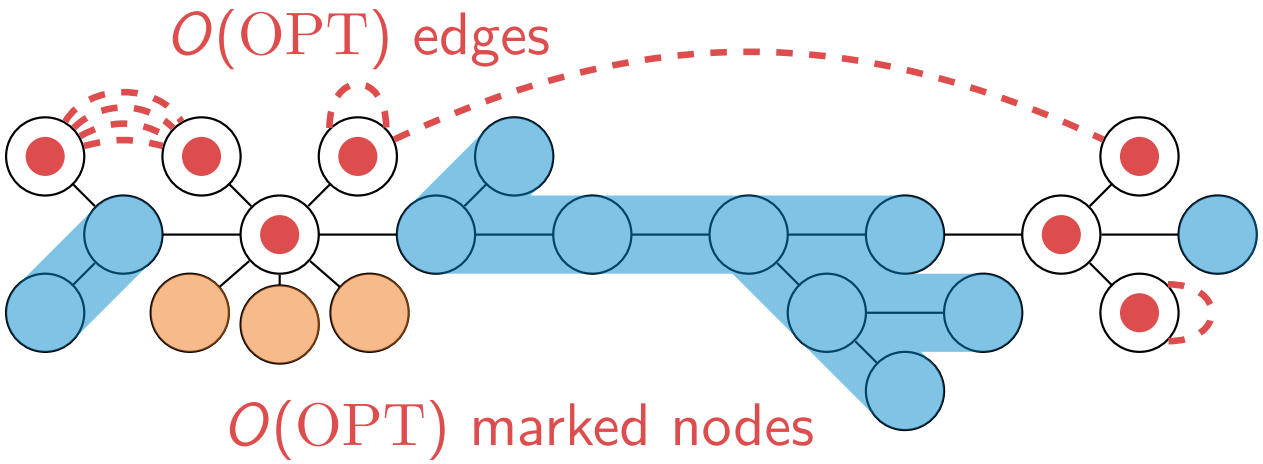
If Z affects B, then we can delete all but $O(|Z|)$ elements of B without changing OPT.

Note: the approximate solution affects every bouquet.

Using the approximate solution, we can reduce the size of bouquets.

$\rightsquigarrow$ ammortizes to a total bouquet size of $O(\mathrm{OPT})$

After reducing protrusions and cleaning bouquets



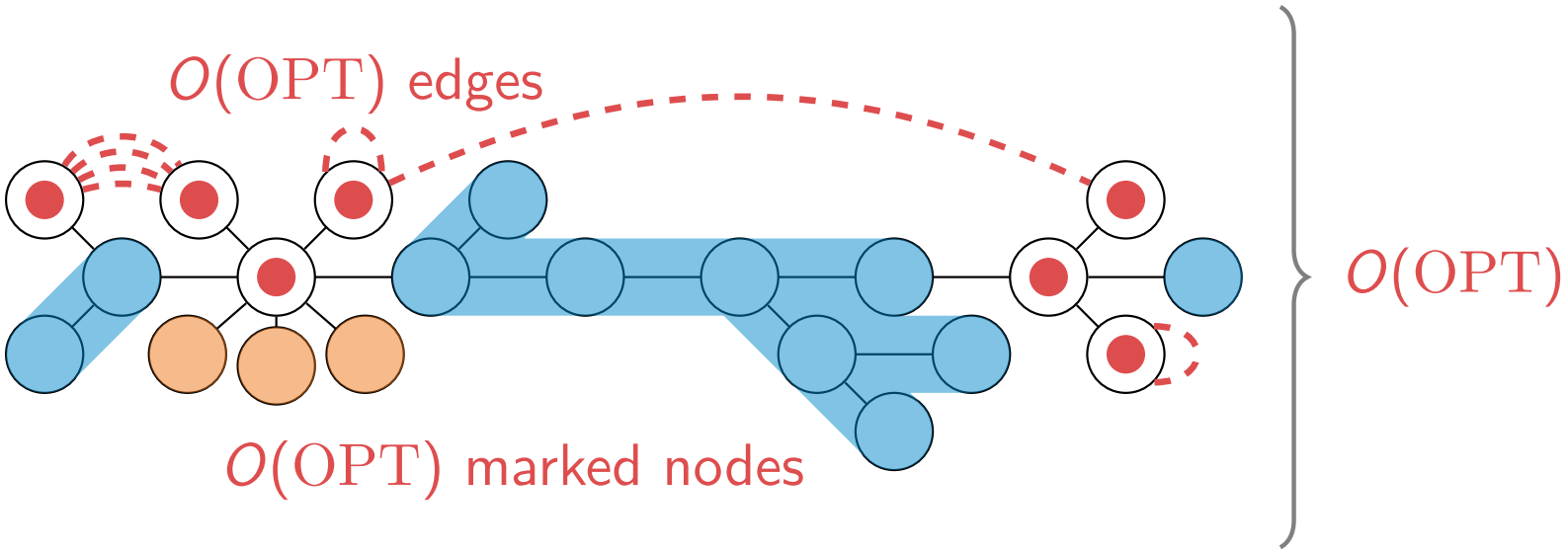
bouquets:

- constant size;
- $O(\mathrm{OPT})$ many.

blue parts:

- constant size;
- $O(\mathrm{OPT})$ many.

After reducing protrusions and cleaning bouquets



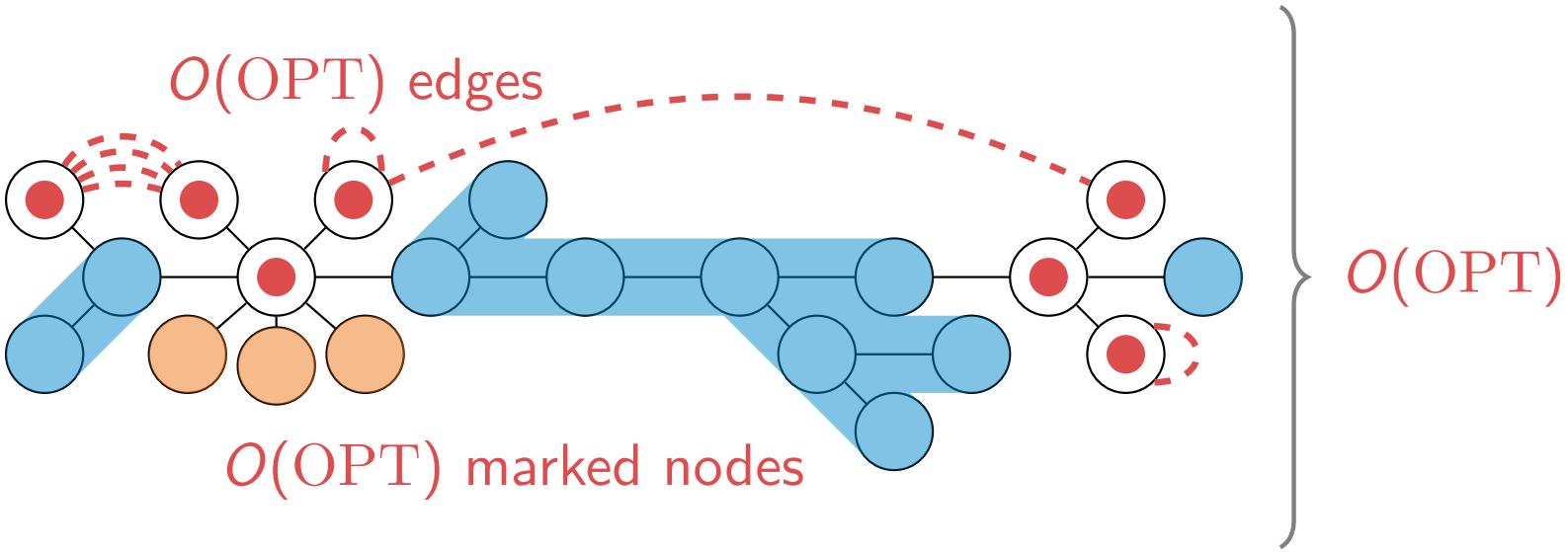
bouquets:

- constant size;
- $O(\mathrm{OPT})$ many.

blue parts:

- constant size;
- $O(\mathrm{OPT})$ many.

After reducing protrusions and cleaning bouquets



bouquets:

- constant size;
- $O(\mathrm{OPT})$ many.

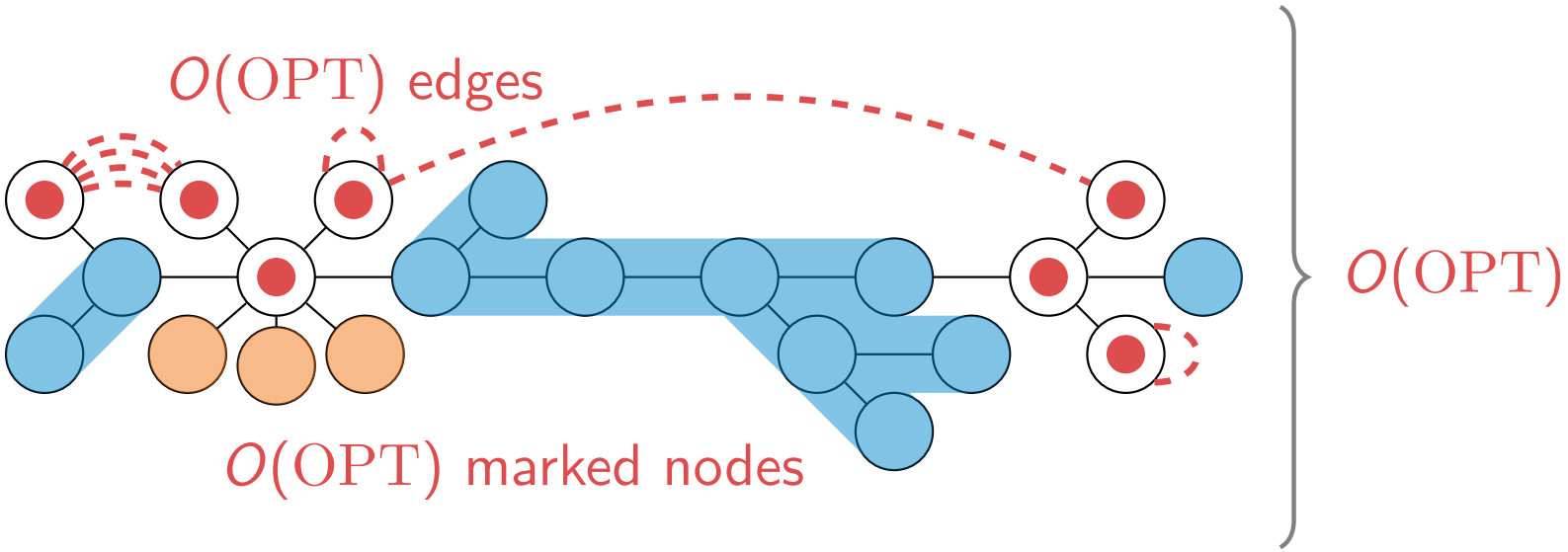
blue parts:

- constant size;
- $O(\mathrm{OPT})$ many.

The kernel:

1. reduce protrusions;

After reducing protrusions and cleaning bouquets



bouquets:

- constant size;
- $O(\mathrm{OPT})$ many.

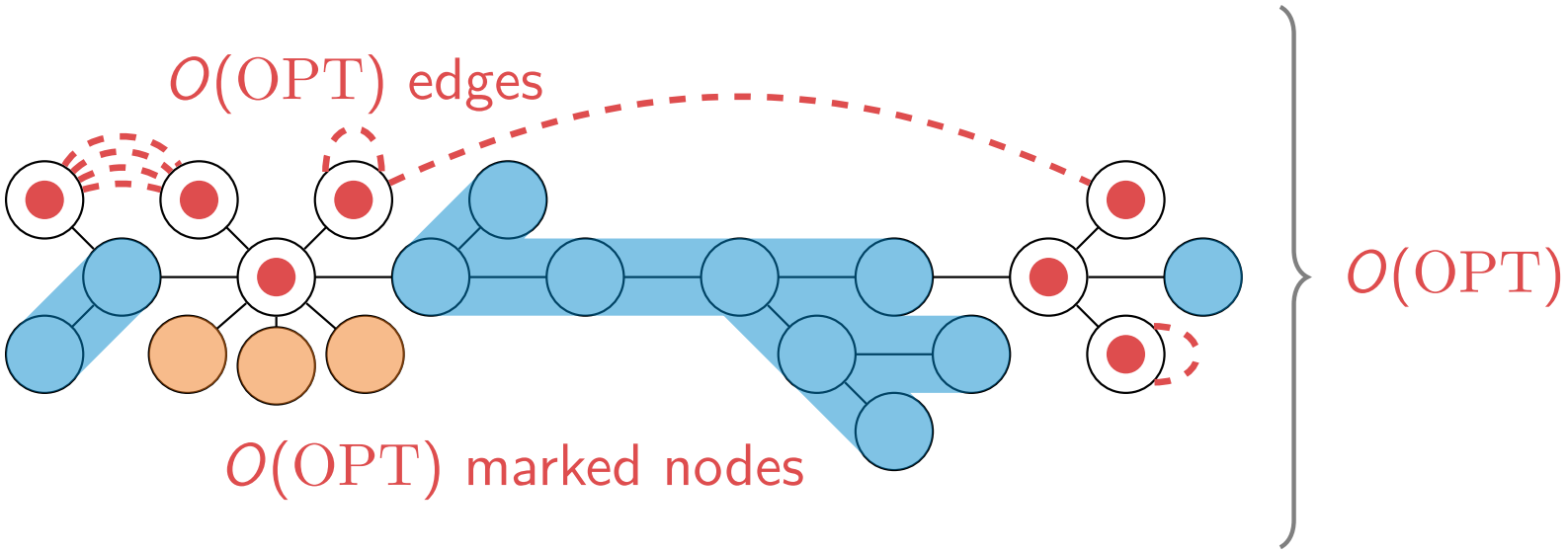
blue parts:

- constant size;
- $O(\mathrm{OPT})$ many.

The kernel:

1. reduce protrusions;
2. compute an approximate solution;

After reducing protrusions and cleaning bouquets



bouquets:

- constant size;
- $O(\mathrm{OPT})$ many.

blue parts:

- constant size;
- $O(\mathrm{OPT})$ many.

The kernel:

1. reduce protrusions;
2. compute an approximate solution;
3. use it to shrink large bouquets.

Application to obstructions

F: connected planar subcubic

Kernelization result (reminder)

Given an instance (G, k) of F-Immersion Deletion, we can compute in polynomial time an equivalent instance of size $O(k)$.

Application to obstructions

F: connected planar subcubic

Kernelization result (reminder)

Given an instance (G, k) of F-Immersion Deletion, we can compute in polynomial time an equivalent instance of size $O(k)$.

$\mathcal{G} = F$-immersion-free graphs

$\mathcal{G}_k =$ graphs at *edge-distance* $\leq k$ from $\mathcal{G}$

Application to obstructions

F: connected planar subcubic

Kernelization result (reminder)

Given an instance (G, k) of F-Immersion Deletion, we can compute in polynomial time an equivalent instance of size $O(k)$.

$\mathcal{G} = F$-immersion-free graphs

$\mathcal{G}_k$= graphs at *edge-distance* $\leq k$ from $\mathcal{G}$

Theorem (Giannopoulou, Pilipczuk, Thilikos, R., Wrochna, 2016+)

Every minimal graph that does not belong to $\mathcal{G}_k$ has $\leq c_F \cdot k$ edges.

Application to obstructions

F: connected planar subcubic

Kernelization result (reminder)

Given an instance (G, k) of F-Immersion Deletion, we can compute in polynomial time an equivalent instance of size $O(k)$.

$\mathcal{G} = F$-immersion-free graphs
$\mathcal{G}_k$= graphs at *edge-distance* $\leq k$ from $\mathcal{G}$

Theorem (Giannopoulou, Pilipczuk, Thilikos, R., Wrochna, 2016+)

Every minimal graph that does not belong to $\mathcal{G}_k$ has $\leq c_F \cdot k$ edges.

- G is an obstruction of $\mathcal{G}_k \Rightarrow (G, k)$ is a NO-instance;

Application to obstructions

F: connected planar subcubic

Kernelization result (reminder)

Given an instance (G, k) of F-Immersion Deletion, we can compute in polynomial time an equivalent instance of size $O(k)$.

$\mathcal{G} = F$-immersion-free graphs
$\mathcal{G}_k$= graphs at *edge-distance* $\leq k$ from $\mathcal{G}$

Theorem (Giannopoulou, Pilipczuk, Thilikos, R., Wrochna, 2016+)

Every minimal graph that does not belong to $\mathcal{G}_k$ has $\leq c_F \cdot k$ edges.

- G is an obstruction of $\mathcal{G}_k \Rightarrow (G, k)$ is a NO-instance;
- the kernelization produces a NO-instance (G', k) with $O(k)$ edges;

Application to obstructions

F: connected planar subcubic

Kernelization result (reminder)

Given an instance (G, k) of F-Immersion Deletion, we can compute in polynomial time an equivalent instance of size $O(k)$.

$\mathcal{G} = F$-immersion-free graphs
$\mathcal{G}_k$= graphs at *edge-distance* $\leq k$ from $\mathcal{G}$

Theorem (Giannopoulou, Pilipczuk, Thilikos, R., Wrochna, 2016+)

Every minimal graph that does not belong to $\mathcal{G}_k$ has $\leq c_F \cdot k$ edges.

- G is an obstruction of $\mathcal{G}_k \Rightarrow (G, k)$ is a NO-instance;
- the kernelization produces a NO-instance (G', k) with $O(k)$ edges;
- G' is an immersion of G (from our proof);

Application to obstructions

F: connected planar subcubic

Kernelization result (reminder)

Given an instance (G, k) of F-Immersion Deletion, we can compute in polynomial time an equivalent instance of size $O(k)$.

$\mathcal{G} = F$-immersion-free graphs
$\mathcal{G}_k$= graphs at *edge-distance* $\leq k$ from $\mathcal{G}$

Theorem (Giannopoulou, Pilipczuk, Thilikos, R., Wrochna, 2016+)

Every minimal graph that does not belong to $\mathcal{G}_k$ has $\leq c_F \cdot k$ edges.

- G is an obstruction of $\mathcal{G}_k \Rightarrow (G, k)$ is a NO-instance;
- the kernelization produces a NO-instance (G', k) with $O(k)$ edges;
- G' is an immersion of G (from our proof);
- as G is minimal, $G = G'$.

Application to immersion closed parameters

Let **p** be a parameter among cutwidth, carving-width, tree-cut width, etc..

Application to immersion closed parameters

Let **p** be a parameter among cutwidth, carving-width, tree-cut width, etc..

Theorem (Giannopoulou, Pilipczuk, Thilikos, R., Wrochna, 2016+)

For every $r \in \mathbb{N}$, the problem **p**-AT-MOST-r-EDGE-DELETION *admits:*

- *a constant factor approximation;*
- *a linear kernel.*

Application to immersion closed parameters

Let **p** be a parameter among cutwidth, carving-width, tree-cut width, etc..

Theorem (Giannopoulou, Pilipczuk, Thilikos, R., Wrochna, 2016+)

For every $r \in \mathbb{N}$, the problem **p**-AT-MOST-r-EDGE-DELETION *admits:*

- *a constant factor approximation;*
- *a linear kernel.*

Idea: graphs G with small $\mathbf{p}(G)$ exclude a planar subcubic graph.

Minors vs. Immersions

F-Minor-Deletion

Input: a graph G and an integer k;
Question: can I remove k vertices in G to get an F-minor-free graph?

Minors vs. Immersions

F-Minor-Deletion

Input: a graph G and an integer k;
Question: can I remove k vertices in G to get an F-minor-free graph?

F: connected planar graph.

Theorem (Fomin, Lokshtanov, Misra, and Saurabh), FOCS 2012)

F-Minor-Deletion admits

- a randomized constant factor approximation that runs in polytime;
- a $O(k^c)$-kernel
- a single exponential FPT algorithm.

Minors vs. Immersions

F-Minor-Deletion

Input: a graph G and an integer k;
Question: can I remove k vertices in G to get an F-minor-free graph?

F: connected planar graph.

Theorem (Fomin, Lokshtanov, Misra, and Saurabh), FOCS 2012)

F-Minor-Deletion admits

- a randomized constant factor approximation that runs in polytime;
- a $O(k^c)$-kernel
- a single exponential FPT algorithm.

Same approach, different issues.

Recap

When F is connected planar subcubic, F-Immersion Deletion has:

- constant factor approximation that runs in polynomial time;
- a linear kernel;
- (a single exponential FPT algorithm).

Recap

When F is connected planar subcubic, F-Immersion Deletion has:

- constant factor approximation that runs in polynomial time;
- a linear kernel;
- (a single exponential FPT algorithm).

Tree-cut width might be the right parameter to study immersions.

Recap

When F is connected planar subcubic, F-Immersion Deletion has:

- constant factor approximation that runs in polynomial time;
- a linear kernel;
- (a single exponential FPT algorithm).

Tree-cut width might be the right parameter to study immersions.

Further work: deal with graphs F that are not planar subcubic.

Recap

When F is connected planar subcubic, F-Immersion Deletion has:

- constant factor approximation that runs in polynomial time;
- a linear kernel;
- (a single exponential FPT algorithm).

Tree-cut width might be the right parameter to study immersions.

Further work: deal with graphs F that are not planar subcubic.

Thank you for your attention!