The Parameterized Complexity of Graph Cyclability

Spyridon Maniatis

National and Kapodistrian
University of Athens

AGTAC, Koper, June 2015

Joint work with *Petr A. Golovach, Marcin Kamiński, and Dimitrios M. Thilikos*

Introduction

- We study (from the algorithmic point of view) a connectivity related parameter, namely cyclability [V. Chvátal, 1973].
- Can be thought as of a quantitive measure of Hamiltonicity (or a way to unify connectivity and Hamiltonicity):

Cyclability

A graph G is k -cyclable if every k vertices of $V(G)$ lie in a common cycle. The cyclability of G is the maximum integer k for which G is k -cyclable.

Introduction

- We study (from the algorithmic point of view) a connectivity related parameter, namely cyclability [V. Chvátal, 1973].
- Can be thought as of a quantitive measure of Hamiltonicity (or a way to unify connectivity and Hamiltonicity):

Cyclability

A graph G is k -cyclable if every k vertices of $V(G)$ lie in a common cycle. The cyclability of G is the maximum integer k for which G is k -cyclable.

Introduction

- We study (from the algorithmic point of view) a connectivity related parameter, namely cyclability [V. Chvátal, 1973].
- Can be thought as of a quantitive measure of Hamiltonicity (or a way to unify connectivity and Hamiltonicity):

Cyclability

A graph G is k -cyclable if every k vertices of $V(G)$ lie in a common cycle. The cyclability of G is the maximum integer k for which G is k -cyclable.

Introduction

- **Natural question**: Is there an efficient (polynomial?) algorithm computing the cyclability of a graph?
- NO, because HAMILTONIAN CYCLE is **NP**-hard (even for cubic planar graphs).
- From the parameterized complexity point of view?

Introduction

- **Natural question**: Is there an efficient (polynomial?) algorithm computing the cyclability of a graph?
- NO, because HAMILTONIAN CYCLE is **NP**-hard (even for cubic planar graphs).
- From the parameterized complexity point of view?

Introduction

- **Natural question**: Is there an efficient (polynomial?) algorithm computing the cyclability of a graph?
- NO, because HAMILTONIAN CYCLE is **NP**-hard (even for cubic planar graphs).
- From the parameterized complexity point of view?

The Parameterized Problem

p-Cyclability.

Input: A graph G and a positive integer k .
Parameter: k.
Question: Is G k -cyclable?

We actually consider, for technical reasons, the more general, annotated version of the problem:

The Annotated Version

p-Annotated Cyclability.

Input: A graph G, a set $R \subseteq V(G)$ and a positive integer k .
Parameter: k.
Question: Is it true that for every $S \subseteq R$ with $|S| \leq k$, there exists a cycle of G that meets all the vertices of S?

Of course, when $R = V(G)$ we have an instance of the initial problem.

The Annotated Version

p-Annotated Cyclability.

Input: A graph G, a set $R \subseteq V(G)$ and a positive integer k .
Parameter: k.
Question: Is it true that for every $S \subseteq R$ with $|S| \leq k$, there exists a cycle of G that meets all the vertices of S?

Of course, when $R = V(G)$ we have an instance of the initial problem.

Our results

We study the parameterized complexity of Cyclability. Our results are:

1. Cyclability is **co-W[1]-hard** (even for split-graphs), when parameterized by k .
2. The problem is in **FPT** when restricted to the class of planar graphs.
3. No polynomial kernel unless **NP** $\subseteq$ **co-NP/poly**, when restricted to cubic planar graphs.

Our results

We study the parameterized complexity of Cyclability. Our results are:

1. Cyclability is **co-W[1]**-hard (even for split-graphs), when parameterized by k .
2. The problem is in **FPT** when restricted to the class of planar graphs.
3. No polynomial kernel unless **NP** $\subseteq$ **co-NP/poly**, when restricted to cubic planar graphs.

Our results

We study the parameterized complexity of Cyclability. Our results are:

1. Cyclability is **co-W[1]**-hard (even for split-graphs), when parameterized by k .
2. The problem is in **FPT** when restricted to the class of planar graphs.
3. No polynomial kernel unless **NP** $\subseteq$ **co-NP/poly**, when restricted to cubic planar graphs.

Our results

We study the parameterized complexity of Cyclability. Our results are:

1. Cyclability is **co-W[1]**-hard (even for split-graphs), when parameterized by k .
2. The problem is in **FPT** when restricted to the class of planar graphs.
3. No polynomial kernel unless **NP** $\subseteq$ **co-NP/poly**, when restricted to cubic planar graphs.

Theorem 1

Theorem 1

The p-Cyclability problem is **co-W[1]**-hard. This also holds if the inputs are restricted to be split graphs.

Reduction of the k -Clique problem to:

p-Cyclability complement.

Input: A split graph G and a positive integer k .
Parameter: k.
Question: Is there an $S \subseteq V(G)$, $|S| \leq k$ s.t. there is no cycle of G that contains all vertices of S?

Theorem 1

Theorem 1

The p-Cyclability problem is **co-W[1]**-hard. This also holds if the inputs are restricted to be split graphs.

Reduction of the k -Clique problem to:

p-Cyclability complement.

Input: A split graph G and a positive integer k .
Parameter: k.
Question: Is there an $S \subseteq V(G)$, $|S| \leq k$ s.t. there is no cycle of G that contains all vertices of S?

Theorem 1

Theorem 1

The p-CYCLABILITY problem is **co-W[1]**-hard. This also holds if the inputs are restricted to be split graphs.

Reduction of the k -CLIQUE problem to:

p-CYCLABILITY COMPLEMENT.

Input: A split graph G and a positive integer k .
Parameter: k.
Question: Is there an $S \subseteq V(G)$, $|S| \leq k$ s.t. there is no cycle of G that contains all vertices of S?

Theorem 2

Theorem 2

The p-CYCLABILITY problem, when parameterized by k , is in **FPT** when its inputs are restricted to be planar graphs. Moreover, the corresponding **FPT**-algorithm runs in $2^{2^{O(k^2 \log k)}} \cdot n^2$ steps.

Irrelevant vertex technique

- We refer to p-Annotated Cyclability, restricted to planar graphs, as problem Π.
- Main idea of our algorithm: Application of the irrelevant vertex technique (introduced by Robertson, Seymour, GM XXII, 2012).

For our purposes, we actually consider two kinds of irrelevant vertex.

Irrelevant vertex technique

- We refer to p-Annotated Cyclability, restricted to planar graphs, as problem Π.
- Main idea of our algorithm: Application of the irrelevant vertex technique (introduced by Robertson, Seymour, GM XXII, 2012).

For our purposes, we actually consider two kinds of irrelevant vertex.

Irrelevant vertices

Problem-irrelevant vertex

Let (G, R, k) be an instance for Π. Then vertex $v \in V(G)$ is called problem-irrelevant for Π, if $(G, R, k) \in \Pi \Leftrightarrow (G \setminus v, R, k) \in \Pi$.

Color-irrelevant vertex

Let (G, R, k) be an instance for Π. Then vertex $v \in R$ is called color-irrelevant for Π, if $(G, R, k) \in \Pi \Leftrightarrow (G, R \setminus v, k) \in \Pi$.

Irrelevant vertices

Problem-irrelevant vertex

Let (G, R, k) be an instance for Π. Then vertex $v \in V(G)$ is called problem-irrelevant for Π, if $(G, R, k) \in \Pi \Leftrightarrow (G \setminus v, R, k) \in \Pi$.

Color-irrelevant vertex

Let (G, R, k) be an instance for Π. Then vertex $v \in R$ is called color-irrelevant for Π, if $(G, R, k) \in \Pi \Leftrightarrow (G, R \setminus v, k) \in \Pi$.

Irrelevant vertices

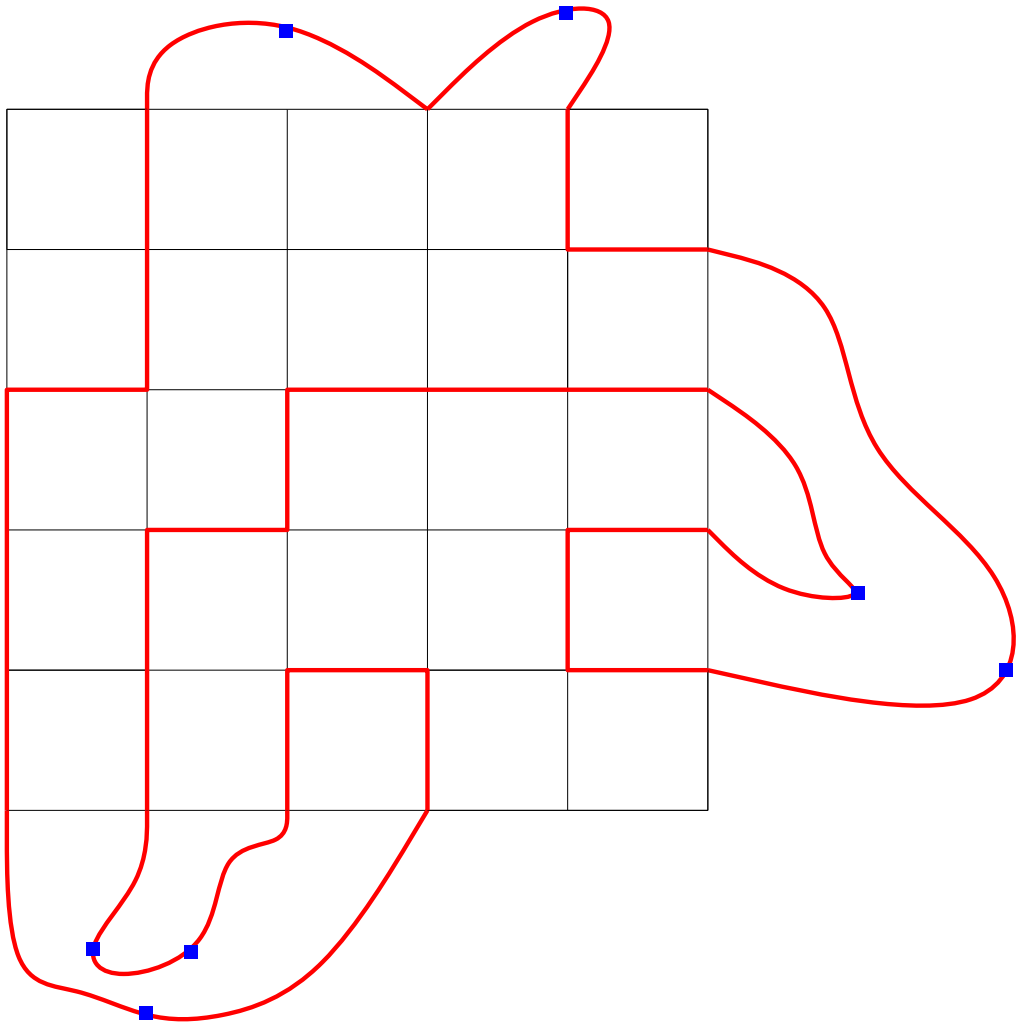
Problem-irrelevant vertex

Let (G, R, k) be an instance for Π. Then vertex $v \in V(G)$ is called problem-irrelevant for Π, if $(G, R, k) \in \Pi \Leftrightarrow (G \setminus v, R, k) \in \Pi$.

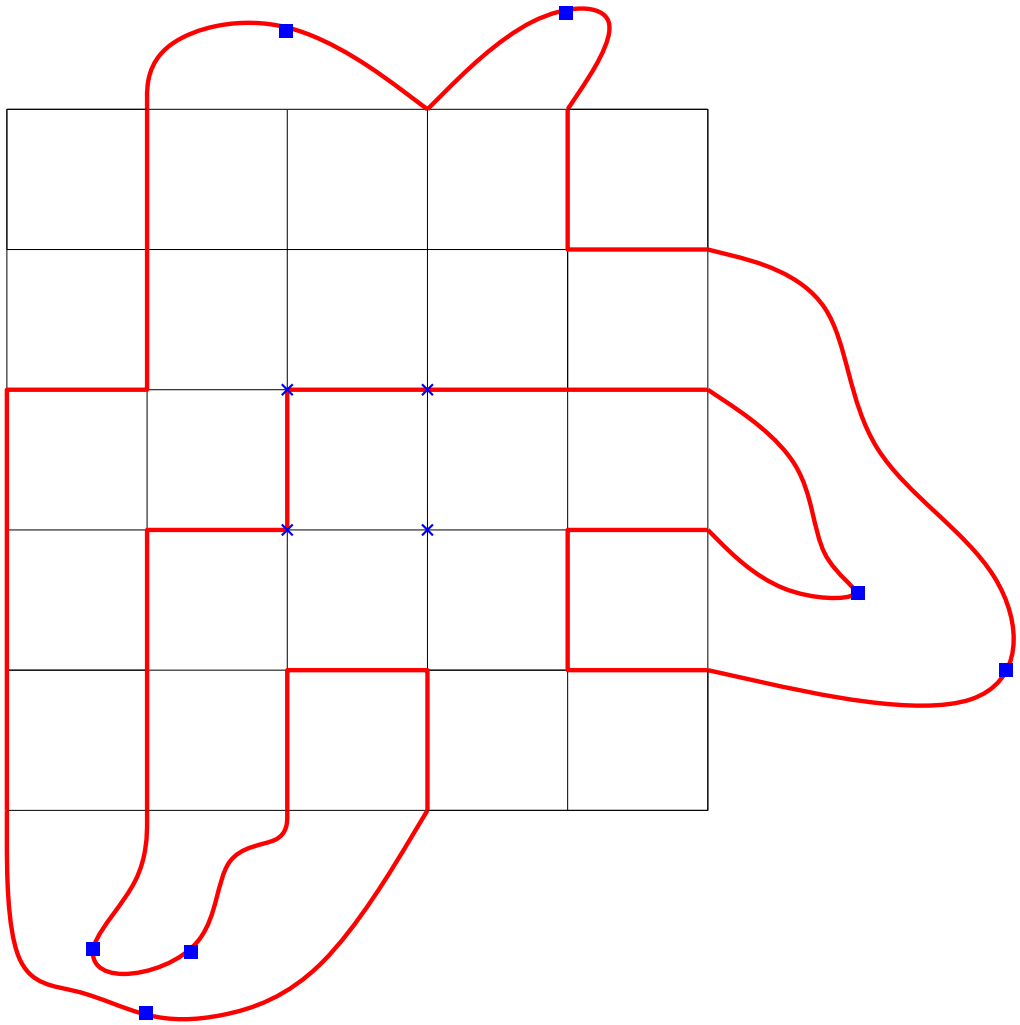
Color-irrelevant vertex

Let (G, R, k) be an instance for Π. Then vertex $v \in R$ is called color-irrelevant for Π, if $(G, R, k) \in \Pi \Leftrightarrow (G, R \setminus v, k) \in \Pi$.

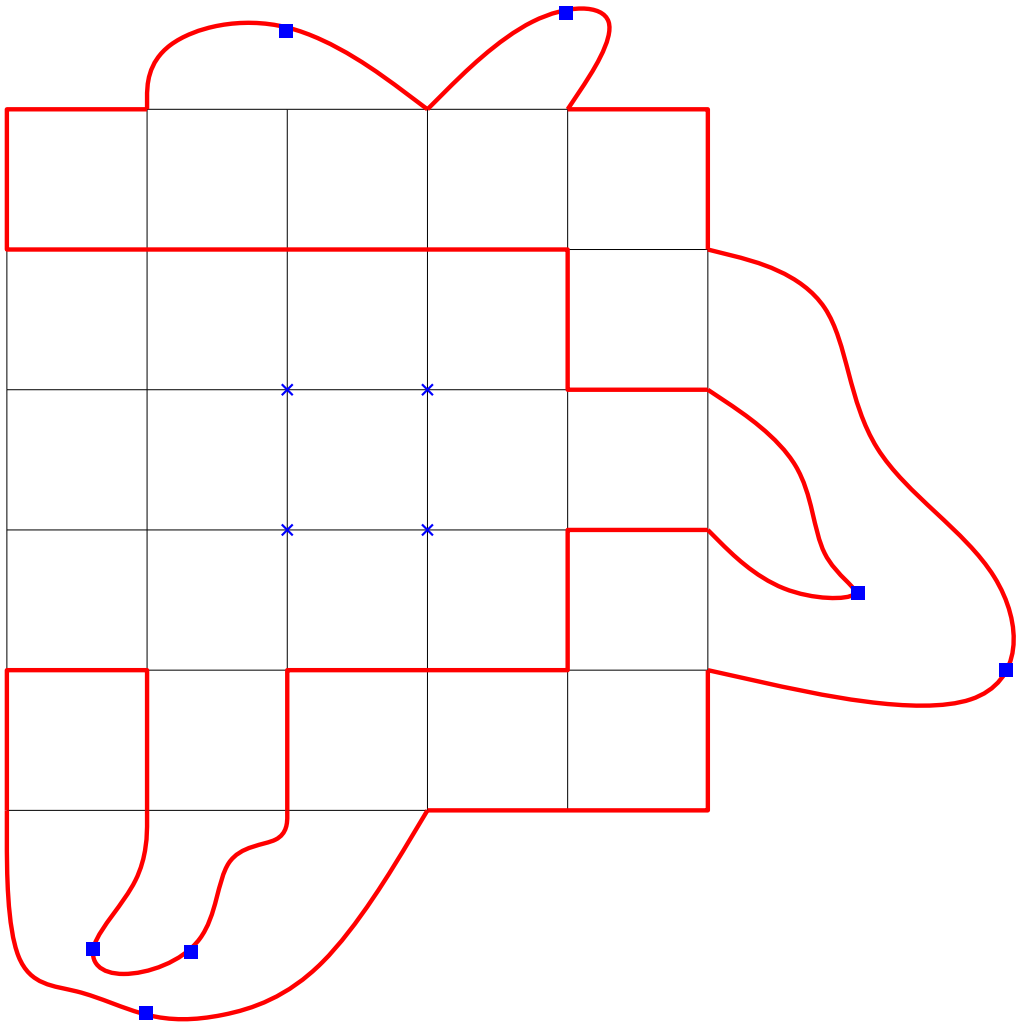
Problem-irrelevant vertices



Problem-irrelevant vertices



Problem-irrelevant vertices



The Algorithm (First step)

- Check if **tw**(G) is upper bounded by an (appropriate) function of k . If YES, solve using dynamic programming.
- Else, we show that there exists a cycle of the plane embedding that contains a "large" subdivided wall H as a subgraph and the part of G that is surrounded by the perimeter of H has bounded treewidth.
- Find in H a sequence $\mathcal{C}$ of "many" concentric cycles that are all traversed by "many" disjoint paths of H. We call such a structure a railed annulus.

‘

The Algorithm (First step)

- Check if **tw**(G) is upper bounded by an (appropriate) function of k . If YES, solve using dynamic programming.
- Else, we show that there exists a cycle of the plane embedding that contains a "large" subdivided wall H as a subgraph and the part of G that is surrounded by the perimeter of H has bounded treewidth.
- Find in H a sequence $\mathcal{C}$ of "many" concentric cycles that are all traversed by "many" disjoint paths of H. We call such a structure a railed annulus.

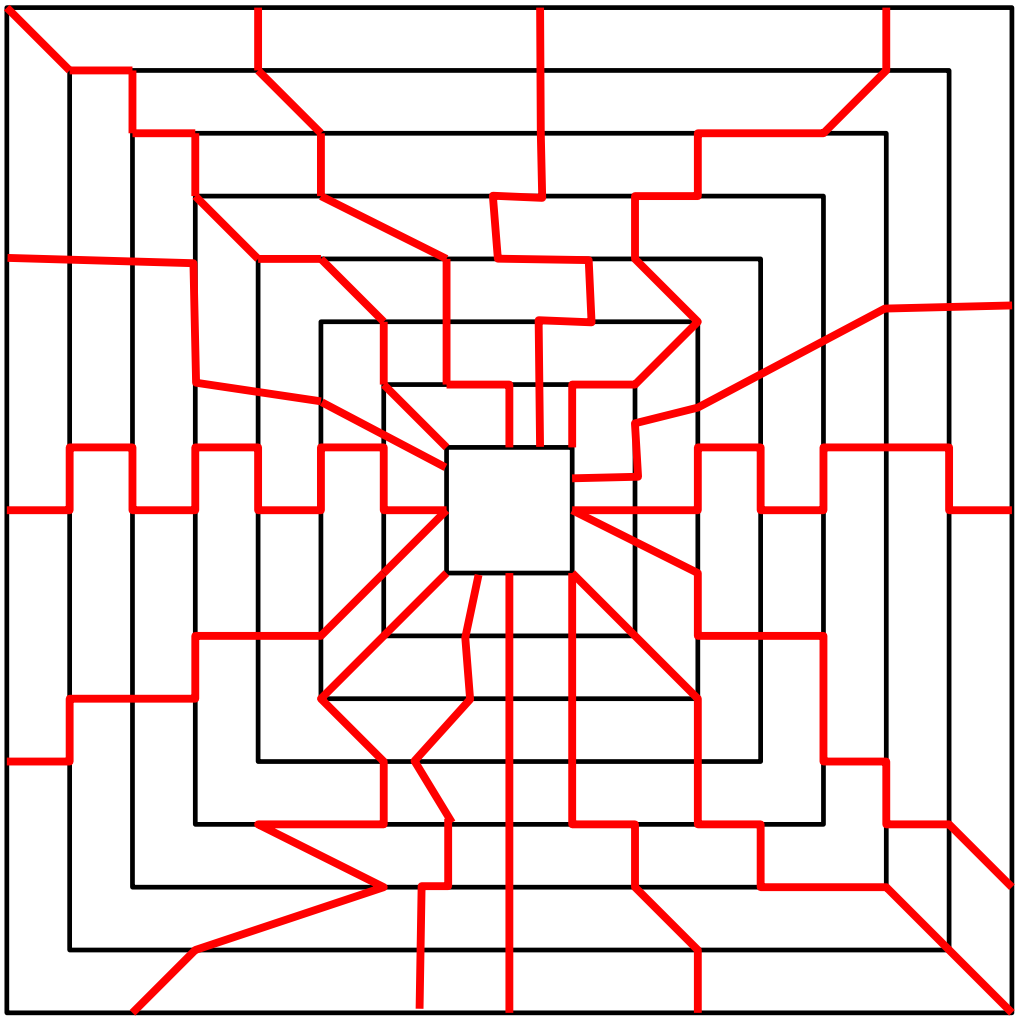
‘

The Algorithm (First step)

- Check if **tw**(G) is upper bounded by an (appropriate) function of k . If YES, solve using dynamic programming.
- Else, we show that there exists a cycle of the plane embedding that contains a "large" subdivided wall H as a subgraph and the part of G that is surrounded by the perimeter of H has bounded treewidth.
- Find in H a sequence $\mathcal{C}$ of "many" concentric cycles that are all traversed by "many" disjoint paths of H. We call such a structure a railed annulus.

‘

Railed annulus



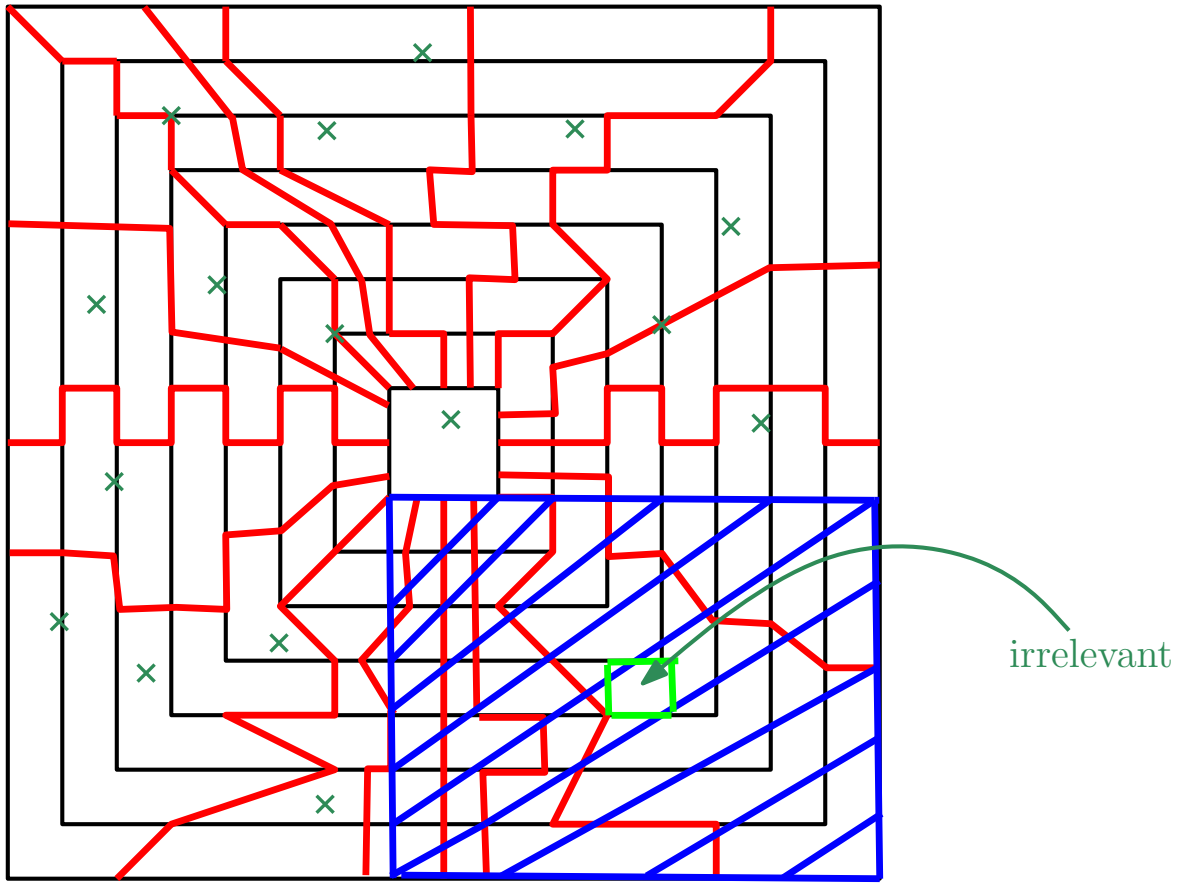
The Algorithm (Second step)

- Check whether in the railed annulus there exists a "large" ("bidimensional") part (function of k^2) not containing any colored vertices.
- If YES, pick a problem-irrelevant vertex (we prove that it exists) and produce a smaller equivalent instance.

The Algorithm (Second step)

- Check whether in the railed annulus there exists a "large" ("bidimensional") part (function of k^2) not containing any colored vertices.
- If YES, pick a problem-irrelevant vertex (we prove that it exists) and produce a smaller equivalent instance.

Big uncolored part



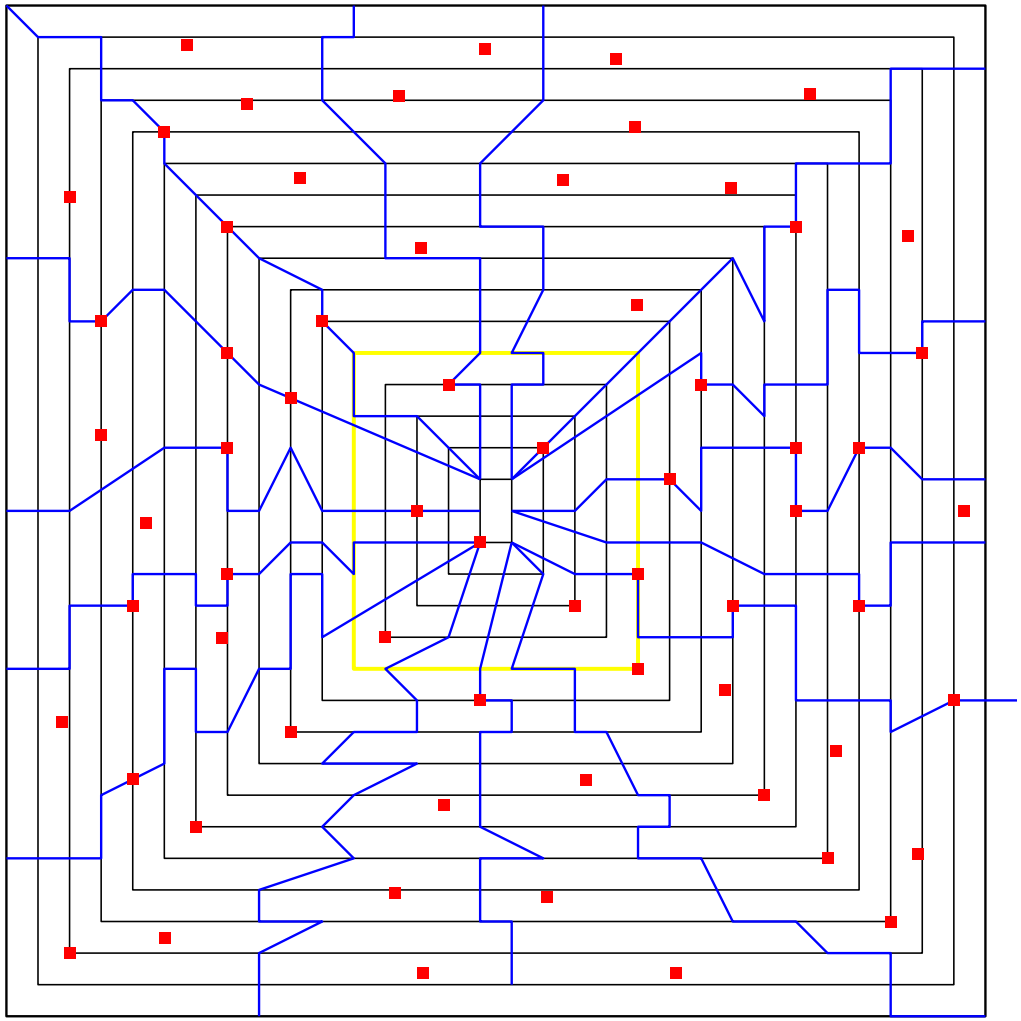
The Algorithm (Third step)

- Else, we know that the annotated vertices are "uniformly" distributed in the railed annulus.
- There exists an annotated vertex $w \in R$ in the "centre" of the annulus.
- We set up a sequence of instances of Π "around" w, each corresponding to the graph "cropped" by the interior of some cycles of $\mathcal{C}$.
- We show that in each of them there exists a "sufficiently large" (function of k) railed annulus.

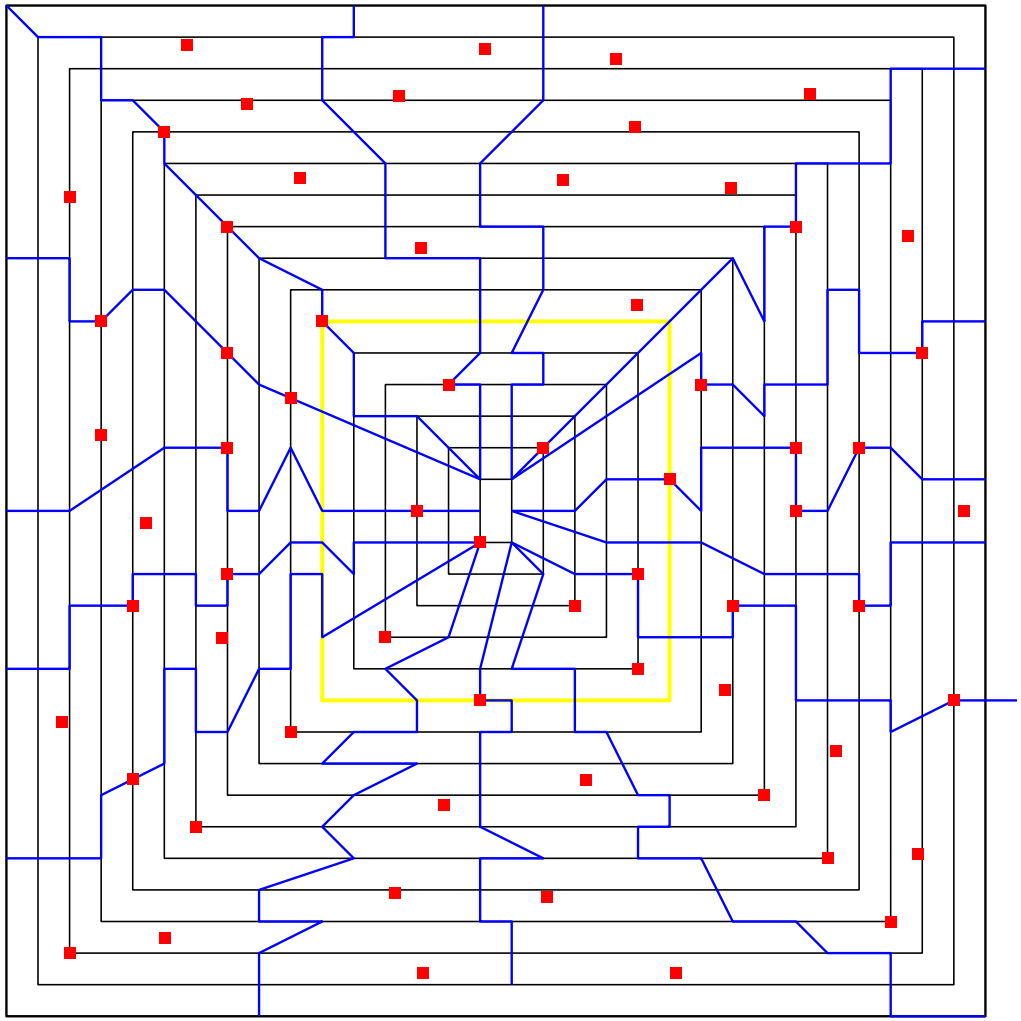
The Algorithm (Third step)

- Else, we know that the annotated vertices are "uniformly" distributed in the railed annulus.
- There exists an annotated vertex $w \in R$ in the "centre" of the annulus.
- We set up a sequence of instances of Π "around" w, each corresponding to the graph "cropped" by the interior of some cycles of $\mathcal{C}$.
- We show that in each of them there exists a "sufficiently large" (function of k) railed annulus.

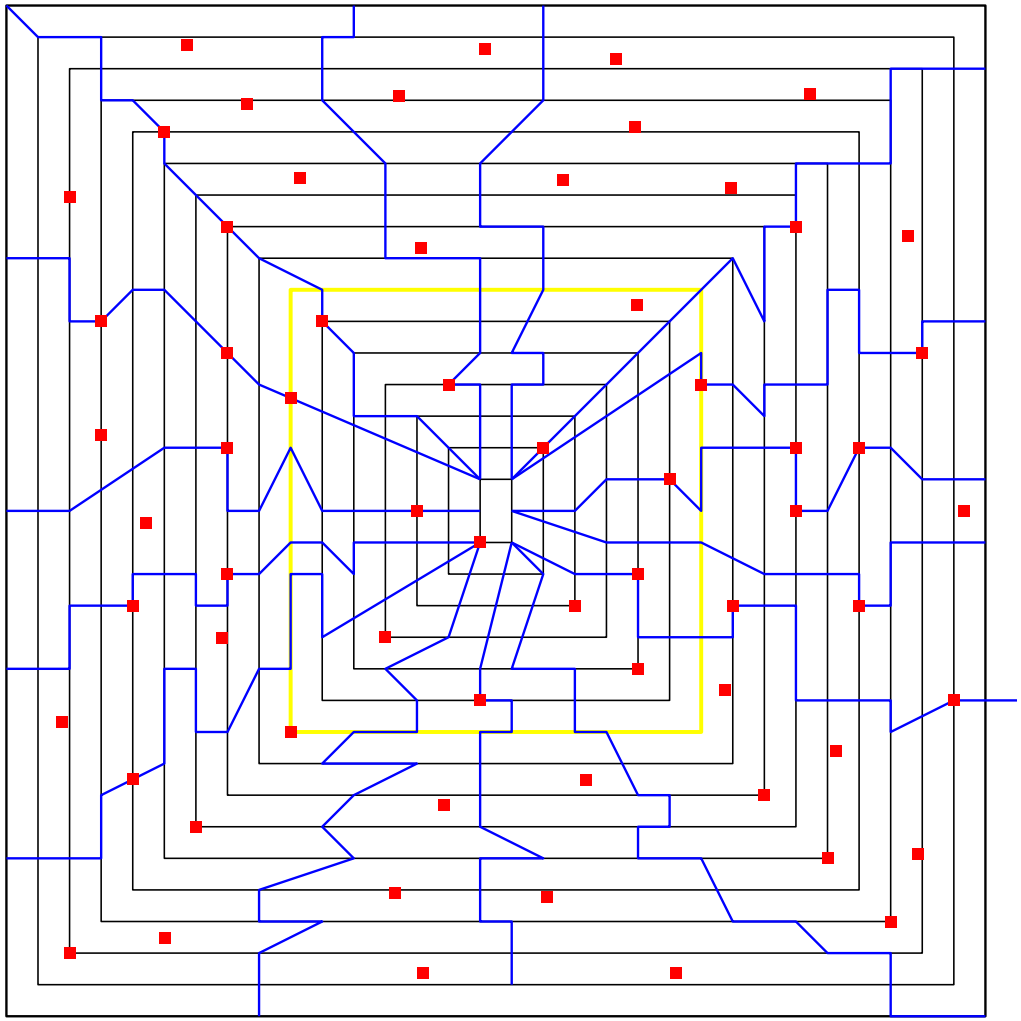
Sequence of instances



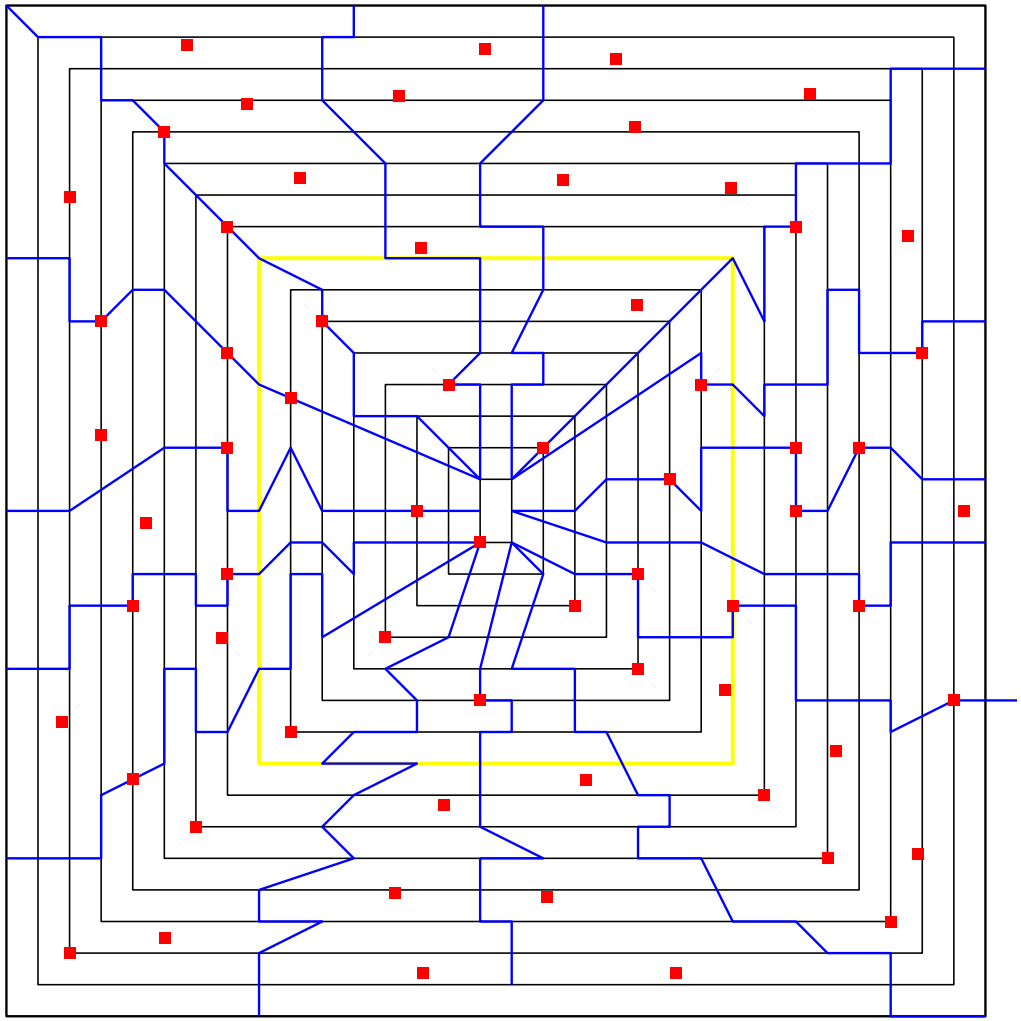
Sequence of instances



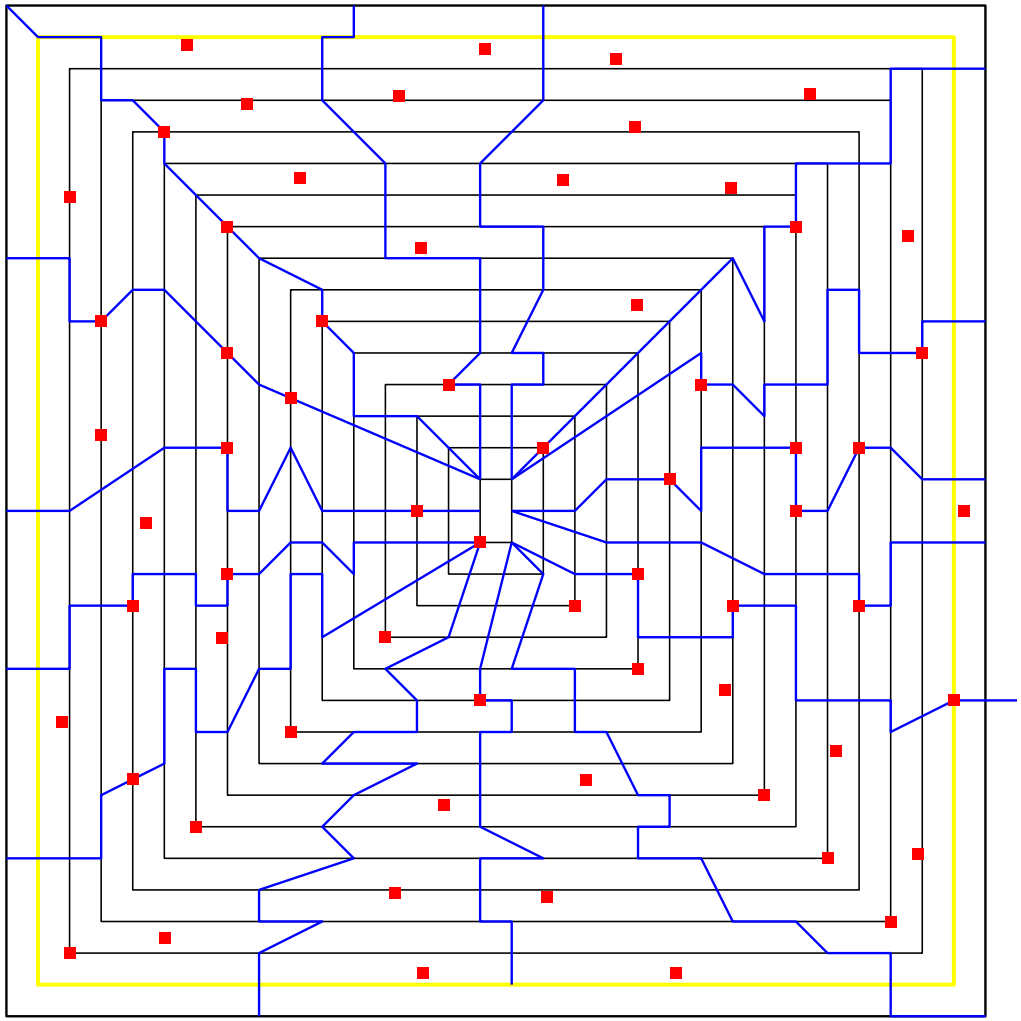
Sequence of instances



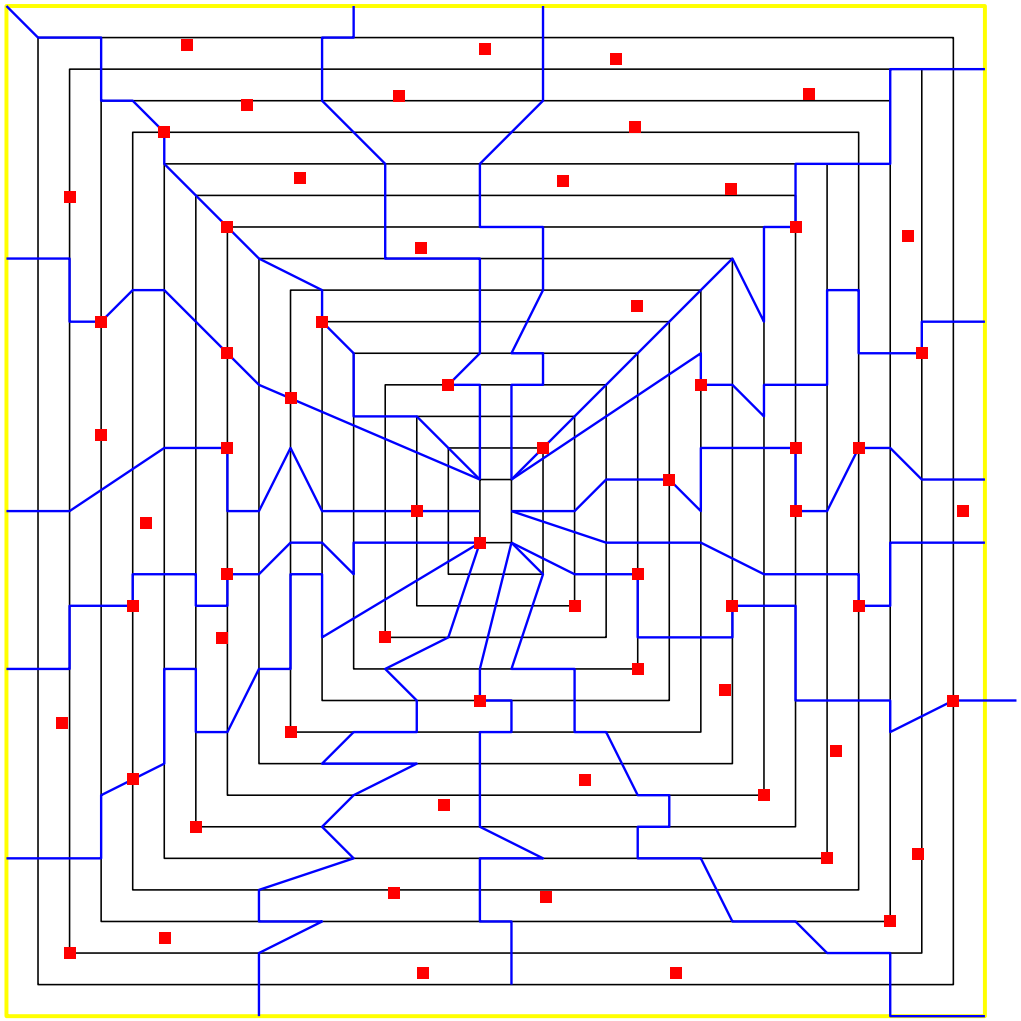
Sequence of instances



Sequence of instances



Sequence of instances



The Algorithm (Fourth step)

- Obtain an answer for every instance, produced in the second step, by a sequence of dynamic programming calls.
- If there exists a NO-instance report that the initial instance is a NO-instance and stop.
- Otherwise we prove that the annotated "central" vertex that we fixed earlier is color irrelevant.

The Algorithm (Fourth step)

- Obtain an answer for every instance, produced in the second step, by a sequence of dynamic programming calls.
- If there exists a NO-instance report that the initial instance is a NO-instance and stop.
- Otherwise we prove that the annotated "central" vertex that we fixed earlier is color irrelevant.

The Algorithm (Fourth step)

- Obtain an answer for every instance, produced in the second step, by a sequence of dynamic programming calls.
- If there exists a NO-instance report that the initial instance is a NO-instance and stop.
- Otherwise we prove that the annotated "central" vertex that we fixed earlier is color irrelevant.

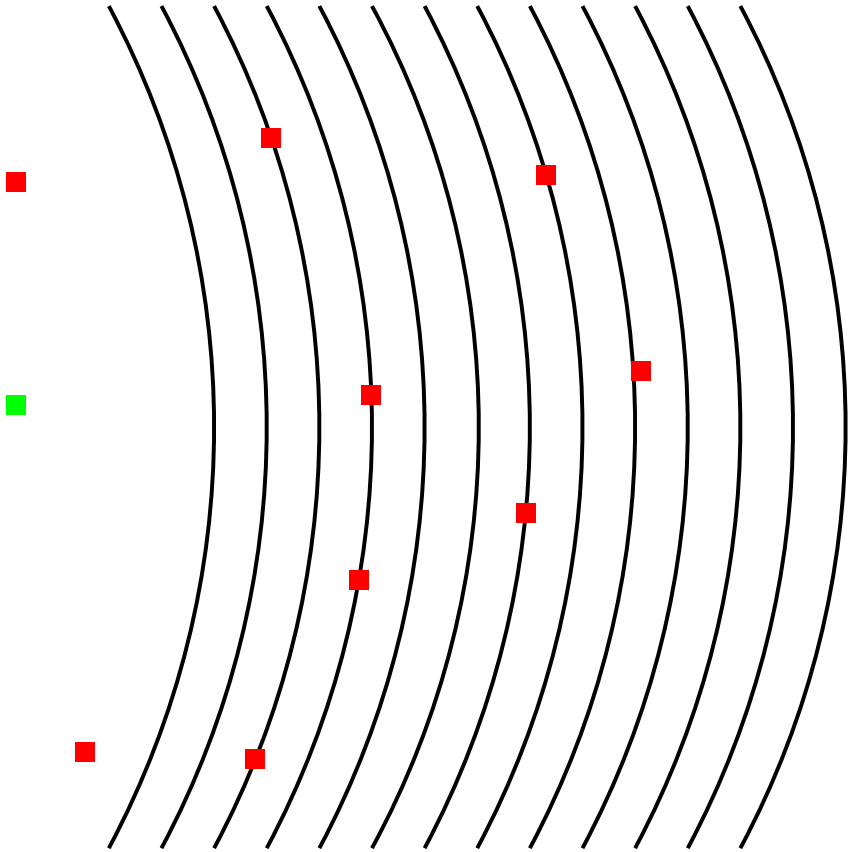
Color-irrelevant vertex

- $(G, R, k) \in \Pi \Rightarrow (G, R \setminus w, k) \in \Pi$: **Trivial**.
- $(G, R \setminus w, k) \in \Pi \Rightarrow (G, R, k) \in \Pi$?

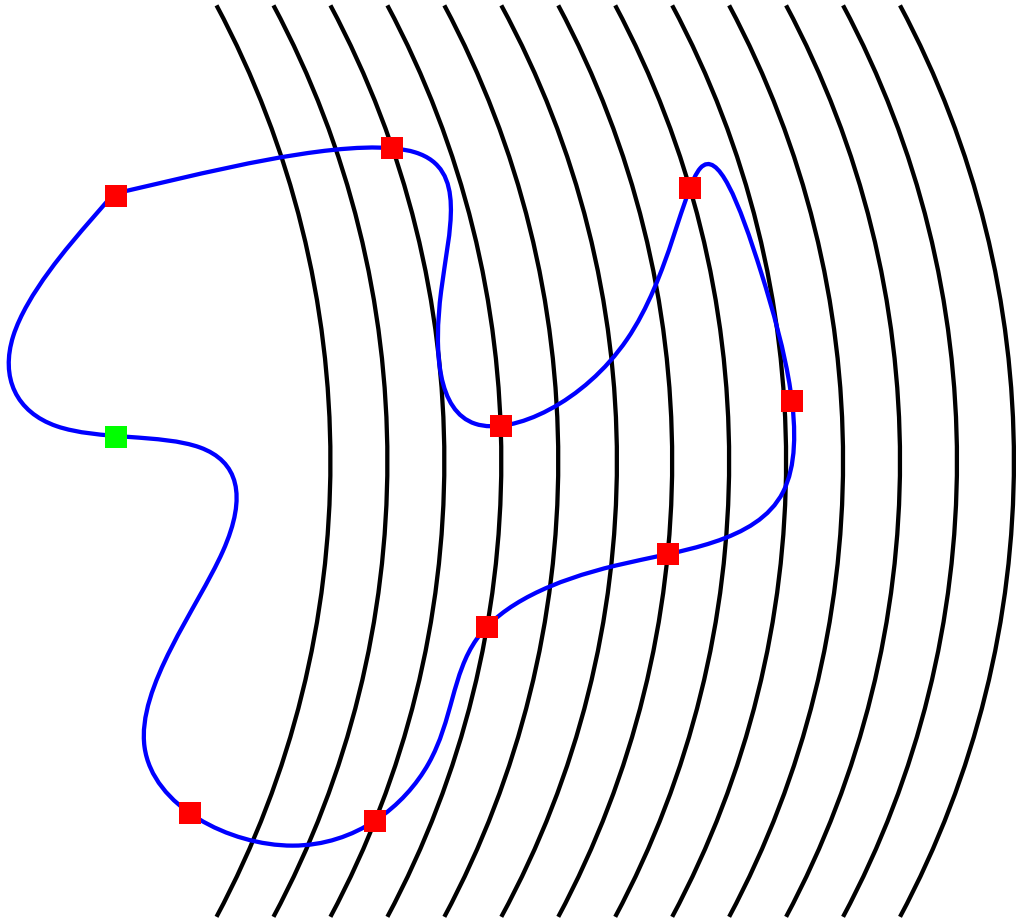
Color-irrelevant vertex

- $(G, R, k) \in \Pi \Rightarrow (G, R \setminus w, k) \in \Pi$: **Trivial**.
- $(G, R \setminus w, k) \in \Pi \Rightarrow (G, R, k) \in \Pi$?

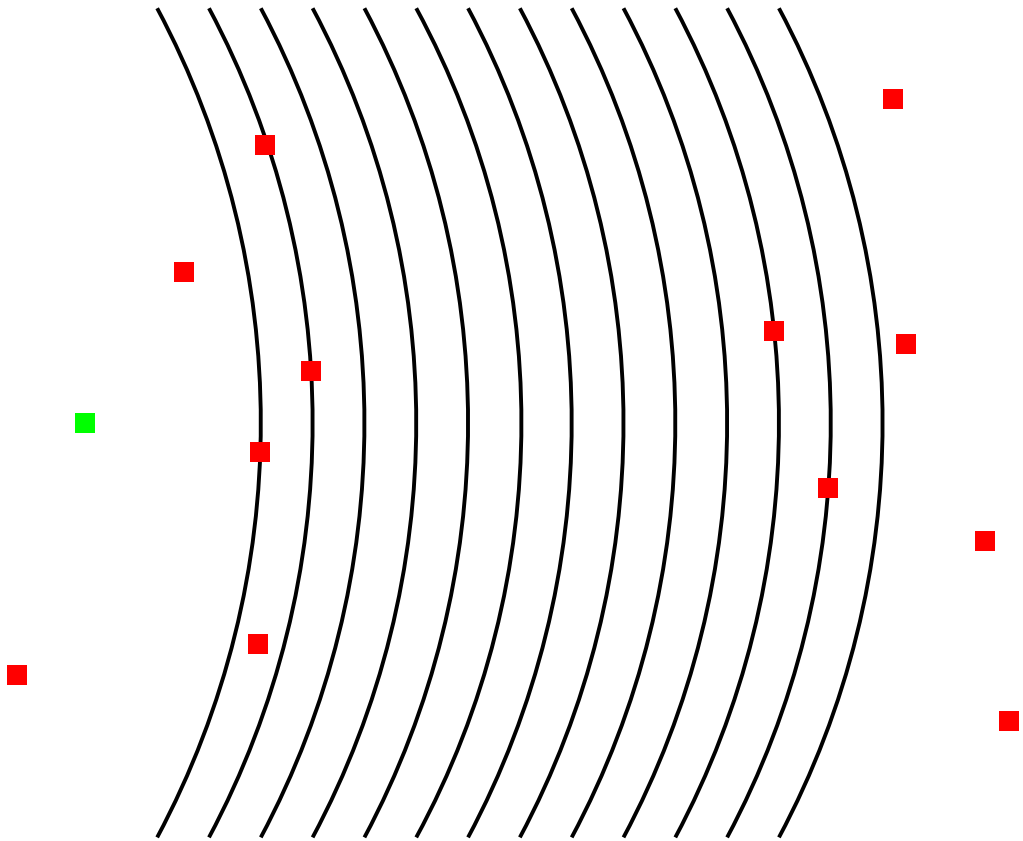
Color-irrelevant vertex



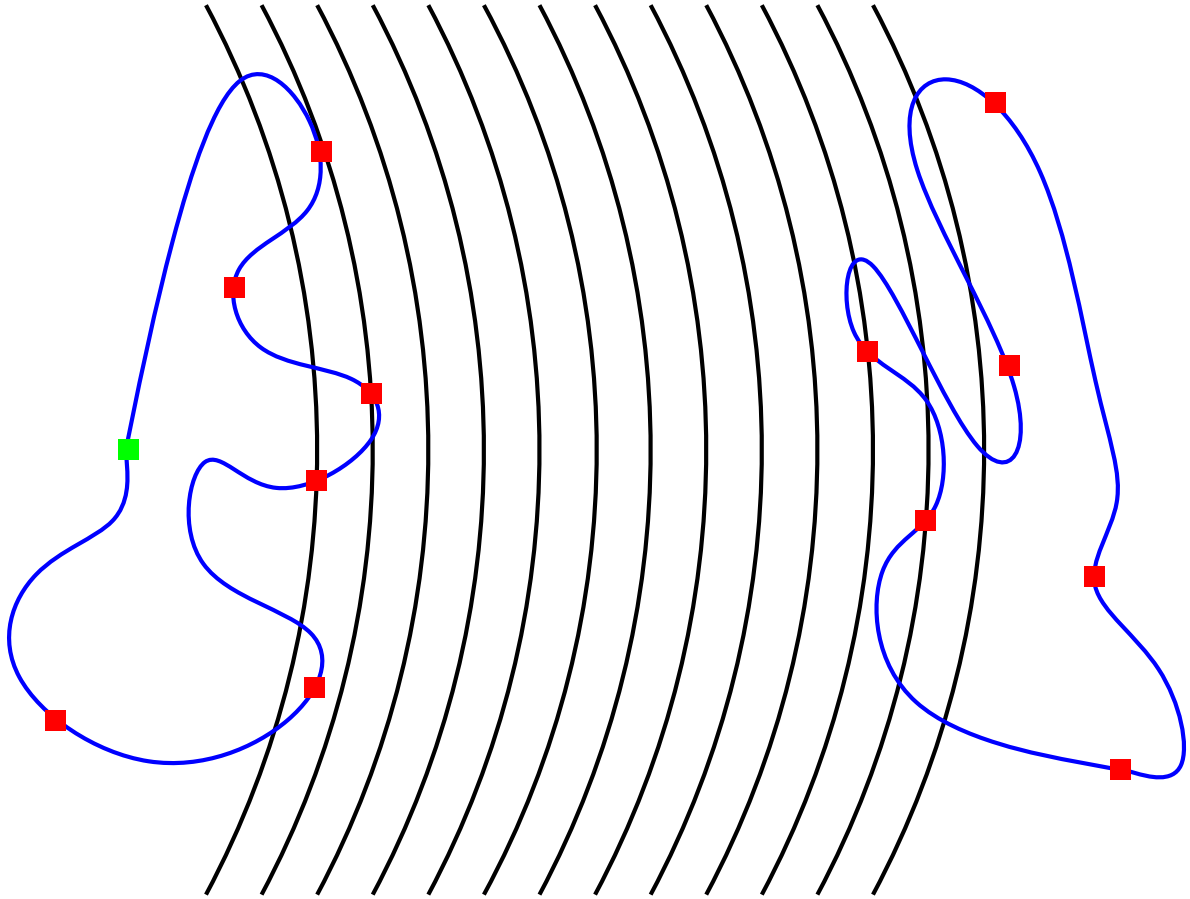
Color-irrelevant vertex



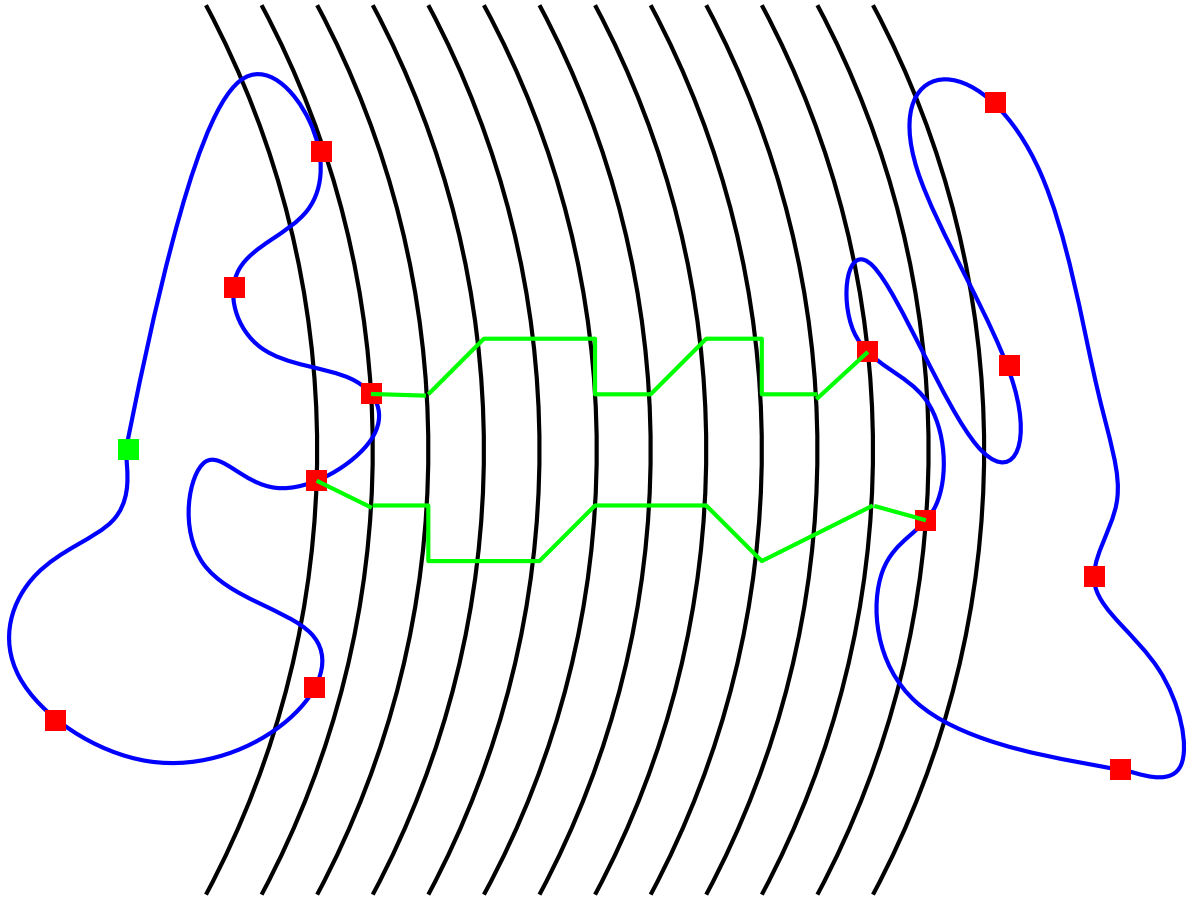
Color-irrelevant vertex



Color-irrelevant vertex



Color-irrelevant vertex



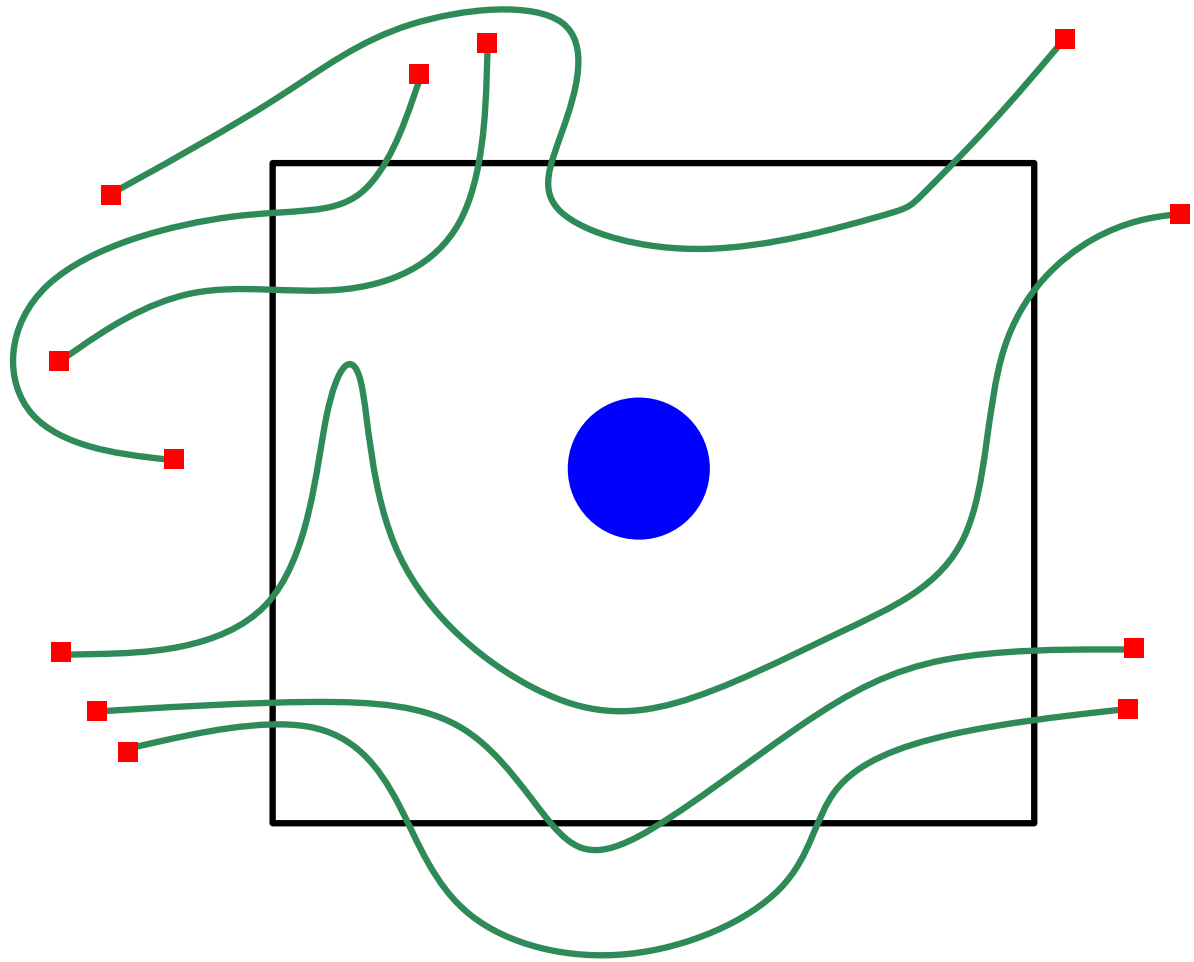
To sum up

After linear number of executions of the procedure:

- Input rejected or
- Treewidth is small $\rightarrow$ Dynamic programming

Something of the above will occur after $O(n)$ steps because at each iteration we reject the input, we "lose" a vertex or we uncolor a vertex.

Irrelevant vertices for the PDPP [Adler et al.]



Main combinatorial statement

The following theorem enables us to find problem-irrelevant vertices:

Theorem

Let G be a graph embedded on the sphere $\mathbb{S}_0$, that is the union of $r \geq 2$ concentric cycles $\mathcal{C} = \{C_1, \ldots, C_r\}$ and one more cycle C of G. Assume that C is tight in G, $T \cap V(\hat{C}_r) = \emptyset$ and the cyclic linkage $\mathcal{L} = (C, T)$ is strongly vital in G. Then $r \leq 16 \cdot |T| - 1$.

Intuition: If there exists a cycle that meets $S \subseteq R$, then there also exists one that meets S **and** does not "go deep" in a bidimensional graph that does not contain any vertices of S.

Main combinatorial statement

The following theorem enables us to find problem-irrelevant vertices:

Theorem

Let G be a graph embedded on the sphere $\mathbb{S}_0$, that is the union of $r \geq 2$ concentric cycles $\mathcal{C} = \{C_1, \ldots, C_r\}$ and one more cycle C of G. Assume that C is tight in G, $T \cap V(\hat{C}_r) = \emptyset$ and the cyclic linkage $\mathcal{L} = (C, T)$ is strongly vital in G. Then $r \leq 16 \cdot |T| - 1$.

Intuition: If there exists a cycle that meets $S \subseteq R$, then there also exists one that meets S **and** does not "go deep" in a bidimensional graph that does not contain any vertices of S.

Main combinatorial statement

The following theorem enables us to find problem-irrelevant vertices:

Theorem

Let G be a graph embedded on the sphere $\mathbb{S}_0$, that is the union of $r \geq 2$ concentric cycles $\mathcal{C} = \{C_1, \ldots, C_r\}$ and one more cycle C of G. Assume that C is tight in G, $T \cap V(\hat{C}_r) = \emptyset$ and the cyclic linkage $\mathcal{L} = (C, T)$ is strongly vital in G. Then $r \leq 16 \cdot |T| - 1$.

Intuition: If there exists a cycle that meets $S \subseteq R$, then there also exists one that meets S **and** does not "go deep" in a bidimensional graph that does not contain any vertices of S.

Dynamic Programming

Some more about the DP for Cyclability:

- Non-trivial DP algorithm ($2^{2^{\mathbf{tw} \cdot \log \mathbf{tw}}}$).
- Causes the double exponential dependance on $k^2 \log k$.
- DP improvement $\rightarrow$ Overall improvement of the algorithm.

Dynamic Programming

Some more about the DP for CYCLABILITY:

- Non-trivial DP algorithm ($2^{2^{\mathbf{tw} \cdot \log \mathbf{tw}}}$).
- Causes the double exponential dependance on $k^2 \log k$.
- DP improvement $\rightarrow$ Overall improvement of the algorithm.

Dynamic Programming

Some more about the DP for Cyclability:

- Non-trivial DP algorithm ($2^{2^{\mathbf{tw} \cdot \log \mathbf{tw}}}$).
- Causes the double exponential dependance on $k^2 \log k$.
- DP improvement $\rightarrow$ Overall improvement of the algorithm.

Kernelization lower bound

Our results suggest that CYCLABILITY, parameterized by k, is unlikely to admit a polynomial kernel, when restricted to planar graphs:

Theorem 3

CYCLABILITY, parameterized by k, has no polynomial kernel unless **NP** $\subseteq$ **co-NP/poly**, when restricted to cubic planar graphs.

Kernelization lower bound

Our results suggest that CYCLABILITY, parameterized by k, is unlikely to admit a polynomial kernel, when restricted to planar graphs:

Theorem 3

CYCLABILITY, parameterized by k, has no polynomial kernel unless **NP** $\subseteq$ **co-NP/poly**, when restricted to cubic planar graphs.

Kernelization lower bound

Let $L \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem.

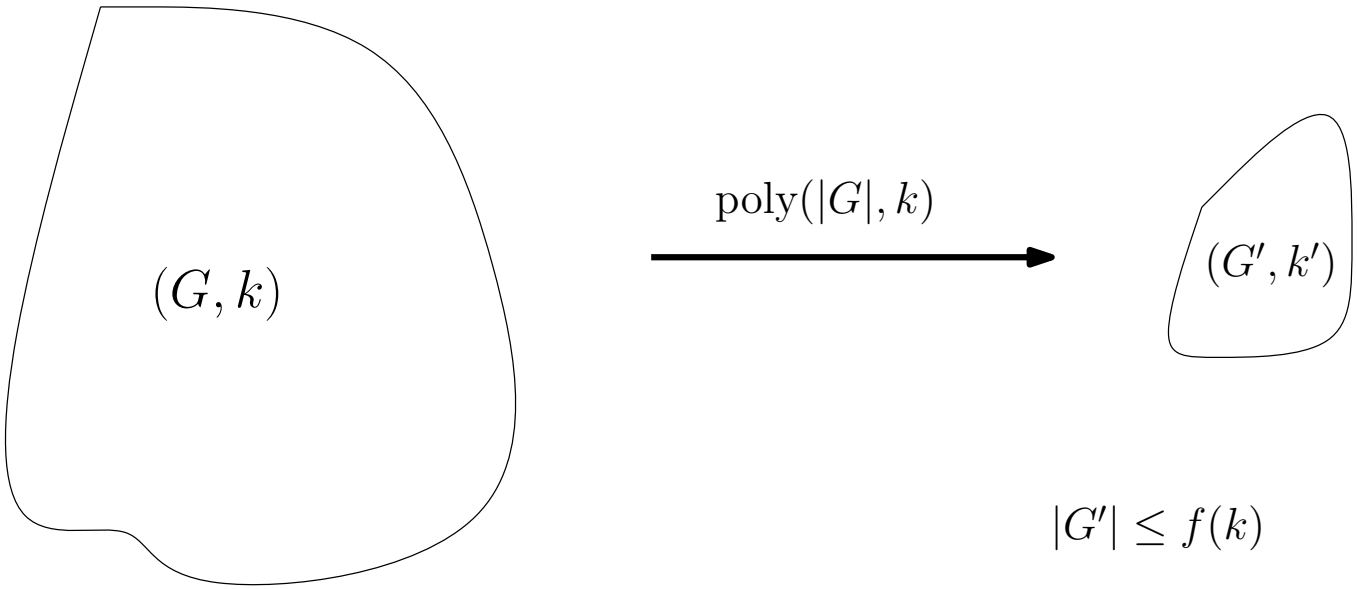
Kernelization for L

A kernelization for problem L is an algorithm that takes as an instance (x, k) of L and maps it, in polynomial time, to an instance (x', k') such that

1. $(x, k) \in L$ iff $(x', k') \in L$
2. $|x'| \leq f(k)$
3. $|k'| \leq g(k)$

where f and g are computable functions. Function f is the size of the kernel and a kernel is polynomial if the corresponding function f is polynomial.

Kernel



$$|G'| \leq f(k)$$

$$k' \leq g(k)$$

Kernelization lower bound

The proof uses the cross-composition technique (introduced by Bodlaender, Jansen and Kratsch):

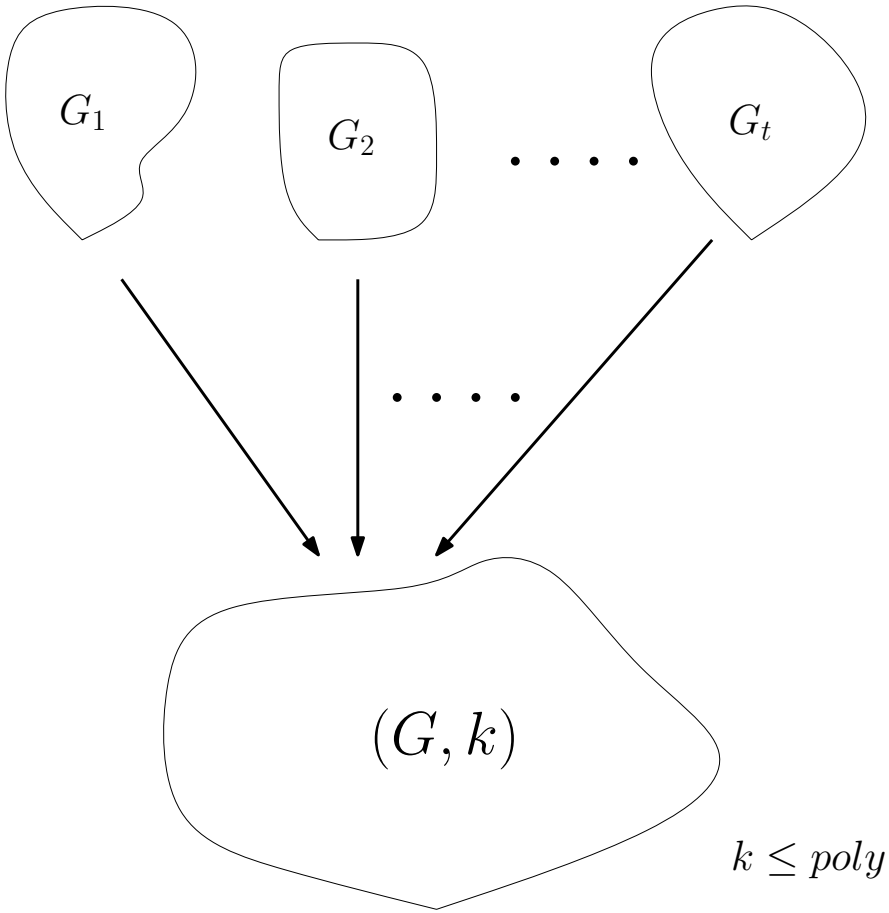
AND-cross-composition

An AND-cross-composition of $L \subseteq \Sigma^*$ into $Q \in \Sigma^* \times \mathbb{N}$ (w.r.t. a polynomial equivalence relation R), is an algorithm that, given t instances $x_1, \ldots, x_t \in \Sigma^*$ of L belonging to the same equivalence class of R, takes polynomial time in $\sum_{i=1}^{t} |x_i|$ and outputs an instance $(y, k) \in \Sigma^* \times \mathbb{N}$ such that:

- the parameter value k is polynomially bounded in $\max\{|x_1|, \ldots, |x_t|\} + \log t$
- (y, k) is a YES-instance for Q iff each instance x_i is a YES-instance for L for $i \in \{1, \ldots, t\}$

We say that L AND-cross-composes into Q if a cross-composition algorithm exists for a suitable relation R.

AND-cross-composition



$$k \leq poly(max\{|x_1|, \ldots |x_t|\} + logt)$$

Kernelization lower bound

Theorem

Assume that an **NP**-hard language L AND-cross-composes to a parameterized language Q. Then Q does not admit a polynomial kernel, unless **NP** $\subseteq$ **co-NP/poly**.

Hamiltonicity with a Given Edge

Input: A graph G and $e \in E(G)$.
Question: Does G have a hamiltonian cycle C s.t. $e \in E(C)$?

- Hamiltonicity with a Given Edge is **NP**-complete for cubic planar graphs.
- Hamiltonicity with a Given Edge AND-cross-composes into p-Cyclability.

Kernelization lower bound

Theorem

Assume that an **NP**-hard language L AND-cross-composes to a parameterized language Q. Then Q does not admit a polynomial kernel, unless **NP** $\subseteq$ **co-NP/poly**.

Hamiltonicity with a Given Edge

Input: A graph G and $e \in E(G)$.
Question: Does G have a hamiltonian cycle C s.t. $e \in E(C)$?

- Hamiltonicity with a Given Edge is **NP**-complete for cubic planar graphs.
- Hamiltonicity with a Given Edge AND-cross-composes into p-Cyclability.

Further research

1. Improve (if possible) the DP algorithm for Cyclability.
2. Prove completeness of Cyclability for some level of the polynomial hierrarchy.
3. Prove completeness of p-Cyclability for some level of the W-hierrarchy.
4. Apply the irrelevant vertex technique to more (connectivity-related) problems.

Further research

1. Improve (if possible) the DP algorithm for CYCLABILITY.
2. Prove completeness of CYCLABILITY for some level of the polynomial hierrarchy.
3. Prove completeness of p-CYCLABILITY for some level of the W-hierrarchy.
4. Apply the irrelevant vertex technique to more (connectivity-related) problems.

Further research

1. Improve (if possible) the DP algorithm for Cyclability.
2. Prove completeness of Cyclability for some level of the polynomial hierrarchy.
3. Prove completeness of p-Cyclability for some level of the W-hierrarchy.
4. Apply the irrelevant vertex technique to more (connectivity-related) problems.

Further research

1. Improve (if possible) the DP algorithm for Cyclability.
2. Prove completeness of Cyclability for some level of the polynomial hierrarchy.
3. Prove completeness of p-Cyclability for some level of the W-hierrarchy.
4. Apply the irrelevant vertex technique to more (connectivity-related) problems.

Thank you