Sparsification Upper and Lower Bounds for Graph Problems and Not-All-Equal SAT

Astrid Pieterse
Supervisor: Bart Jansen

October 1, 2015

Technische Universiteit Eindhoven
University of Technology

Contents

Introduction

Goal

Develop fast algorithms to solve problems

- Failed for some problems
- Many considered problems proven NP-hard

Solution

Run an efficient preprocessing algorithm to reduce input size

- How to analyze this preprocessing?
  - Preprocessing algorithm cannot guarantee $n' \leq n - 1$

Goal

Develop fast algorithms to solve problems

- Failed for some problems
- Many considered problems proven NP-hard

Solution

Run an efficient preprocessing algorithm to reduce input size

- How to analyze this preprocessing?
  - Preprocessing algorithm cannot guarantee $n \to n-1$

Introduction

Goal

Develop fast algorithms to solve problems

- Failed for some problems
- Many considered problems proven NP-hard

Solution

Run an efficient preprocessing algorithm to reduce input size

- How to analyze this preprocessing?
  - Preprocessing algorithm cannot guarantee

Introduction

Goal

Develop fast algorithms to solve problems

- Failed for some problems
- Many considered problems proven NP-hard

Solution

Run an efficient preprocessing algorithm to reduce input size

- How to analyze this preprocessing?
  - Preprocessing algorithm cannot guarantee $n \to n-1$

Introduction

Goal

Develop fast algorithms to solve problems

- Failed for some problems
- Many considered problems proven NP-hard

Solution

Run an efficient preprocessing algorithm to reduce input size

- How to analyze this preprocessing?
  - Preprocessing algorithm cannot guarantee $n \to n-1$

Introduction

Goal

Develop fast algorithms to solve problems

- Failed for some problems
- Many considered problems proven NP-hard

Solution

Run an efficient preprocessing algorithm to reduce input size

- How to analyze this preprocessing?
  - Preprocessing algorithm cannot guarantee $n \to n-1$

Fixed parameter tractability

Parameterized problem

Decision problem P with inputs of type (x, k), where k is the parameter

- Examples
  - Solution size k
  - The size of a vertex cover in an input graph
  - The number of vertices in an input graph
  - $\ldots$

P is *Fixed Parameter Tractable* if there is an algorithm with running time

$$O(f(k) \cdot \text{poly}(n))$$

- For NP-hard problems, we expect f to be exponential in k

Fixed parameter tractability

Parameterized problem

Decision problem P with inputs of type (x, k), where k is the parameter

- Examples
  - Solution size k
  - The size of a vertex cover in an input graph
  - The number of vertices in an input graph
  - $\ldots$

P is *Fixed Parameter Tractable* if there is an algorithm with running time

$$O(f(k) \cdot \text{poly}(n))$$

- For NP-hard problems, we expect f to be exponential in k

Fixed parameter tractability

Parameterized problem

Decision problem P with inputs of type (x, k), where k is the parameter

- Examples
  - Solution size k
  - The size of a vertex cover in an input graph
  - The number of vertices in an input graph
  - $\ldots$

P is *Fixed Parameter Tractable* if there is an algorithm with running time

$$O(f(k) \cdot \text{poly}(n))$$

- For NP-hard problems, we expect f to be exponential in k

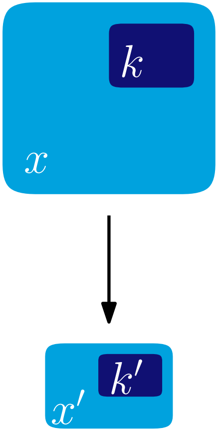
Kernel

Reduce the size of an input instance, before solving the problem.

Kernel

- Algorithm mapping $(x, k) \in P$ to $(x', k') \in P$
  - The running time is polynomial in $|x| + k$
  - $|x'|$ and k' are bounded by $f(k)$
  - (x', k') is a YES-instance for P if and only if (x, k) is a YES-instance for P
- $f(k)$ is the *size*

Any FPT problem has a kernel.

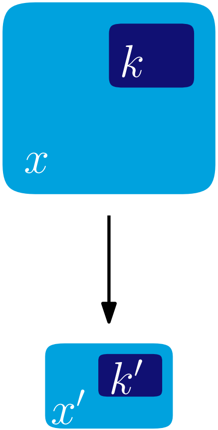


Kernel

Reduce the size of an input instance, before solving the problem.

Kernel

- Algorithm mapping $(x, k) \in P$ to $(x', k') \in P$
  - The running time is polynomial in $|x| + k$
  - $|x'|$ and k' are bounded by $f(k)$
  - (x', k') is a YES-instance for P if and only if (x, k) is a YES-instance for P
- $f(k)$ is the *size*



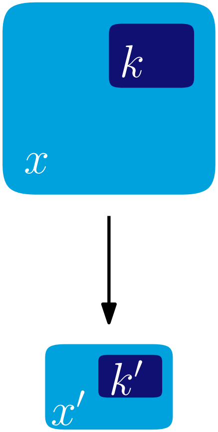
Any FPT problem has a kernel.

Generalized kernel

Reduce the size of an input instance, before solving the problem.

Generalized kernel

- Algorithm mapping $(x, k) \in P$ to $(x', k') \in P'$
  - The running time is polynomial in $|x| + k$
  - $|x'|$ and k' are bounded by $f(k)$
  - (x', k') is a YES-instance for P' if and only if (x, k) is a YES-instance for P
- $f(k)$ is the *size*

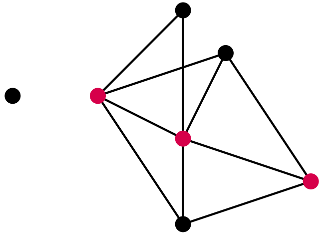


Any FPT problem has a kernel.

Kernel: Example

Vertex Cover

- Input: Graph G, integer k
- Question: Are there k vertices in G, such that for every edge, at least one of its endpoints is chosen?
- Parameter: k



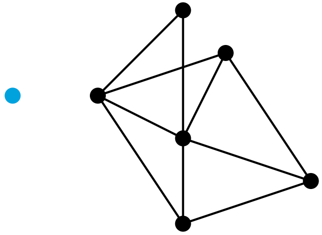
Vertex Cover

- Input: Graph G, integer k
- Question: Are there k vertices in G, such that for every edge, at least one of its endpoints is chosen?
- Parameter: k

Rule 1

If v is an isolated vertex

- Remove v.



Kernel: Example

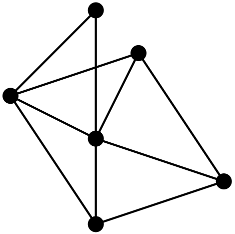
Vertex Cover

- Input: Graph G, integer k
- Question: Are there k vertices in G, such that for every edge, at least one of its endpoints is chosen?
- Parameter: k

Rule 1

If v is an isolated vertex

- Remove v.



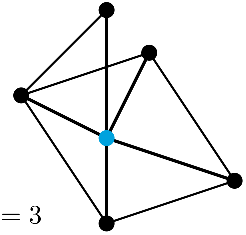
Vertex Cover

- Input: Graph G, integer k
- Question: Are there k vertices in G, such that for every edge, at least one of its endpoints is chosen?
- Parameter: k

Rule 2

If some vertex v has degree larger than k and $k > 0$

- Remove v and decrease k by one.



$k = 3$

Kernel: Example

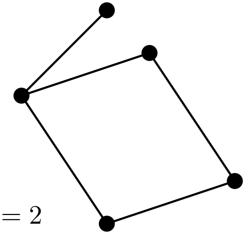
Vertex Cover

- Input: Graph G, integer k
- Question: Are there k vertices in G, such that for every edge, at least one of its endpoints is chosen?
- Parameter: k

Rule 2

If some vertex v has degree larger than k and $k > 0$

- Remove v and decrease k by one.



$k = 2$

Kernel: Example

Vertex Cover

- Input: Graph G, integer k
- Question: Are there k vertices in G, such that for every edge, at least one of its endpoints is chosen?
- Parameter: k

Rule 3

Apply rules 1 and 2 until no longer possible.

- Every vertex has degree at most k
- Every vertex can cover at most k edges
- If we have more than $k \cdot k = k^2$ edges remaining
- Output *NO*

Kernel: Example

Vertex Cover

- Input: Graph G, integer k
- Question: Are there k vertices in G, such that for every edge, at least one of its endpoints is chosen?
- Parameter: k

Rule 3

Apply rules 1 and 2 until no longer possible.

- Every vertex has degree at most k
- Every vertex can cover at most k edges
- If we have more than $k \cdot k = k^2$ edges remaining
- Output *NO*

Kernel: Example

Vertex Cover

- Input: Graph G, integer k
- Question: Are there k vertices in G, such that for every edge, at least one of its endpoints is chosen?
- Parameter: k

Rule 3

Apply rules 1 and 2 until no longer possible.

- Every vertex has degree at most k
- Every vertex can cover at most k edges
- If we have more than $k \cdot k = k^2$ edges remaining
- Output *NO*

Kernel: Example

Vertex Cover

- Input: Graph G, integer k
- Question: Are there k vertices in G, such that for every edge, at least one of its endpoints is chosen?
- Parameter: k

Rule 3

Apply rules 1 and 2 until no longer possible.

- Every vertex has degree at most k
- Every vertex can cover at most k edges
- If we have more than $k \cdot k = k^2$ edges remaining
- Output *NO*

Kernel: Example

Vertex Cover

- Input: Graph G, integer k
- Question: Are there k vertices in G, such that for every edge, at least one of its endpoints is chosen?
- Parameter: k

Resulting kernel

Apply rules 1 and 2 until no longer possible, then apply rule 3.

- Kernel with at most k^2 edges
- and at most $2k^2$ vertices.

Kernel: Example

Vertex Cover

- Input: Graph G, integer k
- Question: Are there k vertices in G, such that for every edge, at least one of its endpoints is chosen?
- Parameter: k

Resulting kernel

Apply rules 1 and 2 until no longer possible, then apply rule 3.

- Kernel with at most k^2 edges
- and at most $2k^2$ vertices.

My goal

Graph problems

- Chosen parameter is $|V|$.
- Represent as an adjacency matrix, $O(|V|^2)$ storage.
- Can we do better, can we reduce the number of edges to something sub-quadratic?

Logic problems

- Chosen parameter: Number of variables.
- Can we reduce the number of clauses?

$$\underbrace{(x \vee y \vee z)}_{\text{clause}} \wedge (x \vee \neg y \vee \neg z) \wedge (\neg x \vee \neg y \vee \neg z)$$

Proving lower bounds

Suppose we know that P does *not* have a small generalized kernel. How to use this result?

Linear parameter transformation

Let P,P' be decision problems. A linear parameter transformation from P to P' maps (x, k) from P to (x', k') from P' where

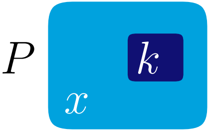
- The reduction takes polynomial time
- (x, k) is a yes-instance $\Leftrightarrow$ (x', k') is yes
- It is linear: $k' = O(k)$

Proving lower bounds

Theorem

If P' has a generalized kernel of size $O(k^d)$, this implies that P has a generalized kernel of $O(k^d)$.

- Let (x, k) be given for P
- Use the transformation to obtain (x', k') for P'
- Use the kernel of P' to compress (x', k').
- Resulting kernel size $O(k^d)$

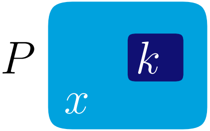


Proving lower bounds

Theorem

If P' has a generalized kernel of size $O(k^d)$, this implies that P has a generalized kernel of $O(k^d)$.

- Let (x, k) be given for P
- Use the transformation to obtain (x', k') for P'
- Use the kernel of P' to compress (x', k').
- Resulting kernel size $O(k^d)$

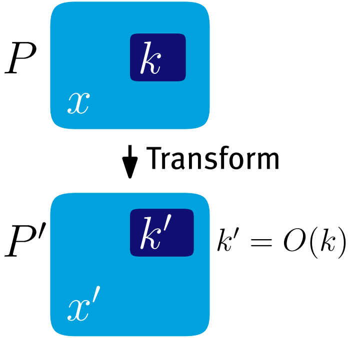


Proving lower bounds

Theorem

If P' has a generalized kernel of size $O(k^d)$, this implies that P has a generalized kernel of $O(k^d)$.

- Let (x, k) be given for P
- Use the transformation to obtain (x', k') for P'
- Use the kernel of P' to compress (x', k').
- Resulting kernel size $O(k^d)$

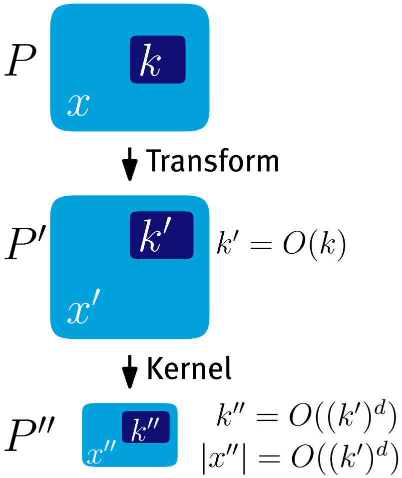


Proving lower bounds

Theorem

If P' has a generalized kernel of size $O(k^d)$, this implies that P has a generalized kernel of $O(k^d)$.

- Let (x, k) be given for P
- Use the transformation to obtain (x', k') for P'
- Use the kernel of P' to compress (x', k').
- Resulting kernel size $O(k^d)$



Proving lower bounds

Theorem

If P' has a generalized kernel of size $O(k^d)$, then P has a generalized kernel of $O(k^d)$.

Consequence

If P does *not* have a generalized kernel of size $O(k^{d-\varepsilon})$ for $\varepsilon > 0$, then P' does not have a generalized kernel of size $O(k^{d-\varepsilon})$.

d-CNF-SAT

- Input: A Boolean formula $\mathcal{F}$ in CNF form, where every clause contains at most d literals.

$$\mathcal{F} = \underbrace{(x \vee \neg y \vee \ldots \vee z)}_{\text{Clause}} \wedge (\neg x \vee \neg z \vee \ldots \vee \neg y) \wedge \ldots$$

- Parameter: The number of variables n.
- Question: Can we find a truth assignment such that $\mathcal{F}$ is *true*?

Satisfiability

- Important NP-hard problem
  - For $d \geqslant 3$

Claim

d-CNF-Sat does not have a generalized kernel of size $O(n^{d-\varepsilon})$, unless $NP \subseteq coNP/poly$ (Dell and Van Melkebeek).

$NP \not\subseteq coNP/poly$ will be used as an assumption, compare to $P \neq NP$.

Satisfiability

- Important NP-hard problem
  - For $d \geqslant 3$

Claim

d-CNF-Sat does not have a generalized kernel of size $O(n^{d-\varepsilon})$, unless $\mathrm{NP} \subseteq \mathrm{coNP}/\mathrm{poly}$ (Dell and Van Melkebeek).

$\mathrm{NP} \not\subseteq \mathrm{coNP}/\mathrm{poly}$ will be used as an assumption, compare to $\mathrm{P} \neq \mathrm{NP}$.

Satisfiability

- Important NP-hard problem
  - For $d \geqslant 3$

Claim

d-CNF-Sat does not have a generalized kernel of size $O(n^{d-\varepsilon})$, unless $NP \subseteq coNP/poly$ (Dell and Van Melkebeek).

$NP \not\subseteq coNP/poly$ will be used as an assumption, compare to $P \neq NP$.

Not-All-Equal Satisfiability

d-NAE-SAT

- Input: A Boolean formula in CNF form, where every clause contains at most d literals.

$$\underbrace{(x \vee \neg y \vee \ldots \vee z)}_{\text{Clause}} \wedge (\neg x \vee \neg z \vee \ldots \vee \neg y) \wedge \ldots$$

- Parameter: The number of variables n.
- Question: Can we find a truth assignment such that each clause contains at least one *true* and one *false* literal?

Compare to d-CNF-Sat, where we only require at least one *true* literal per clause.

Not-All-Equal Satisfiability

d-NAE-SAT

- Input: A Boolean formula in CNF form, where every clause contains at most d literals.

$$\underbrace{(x \vee \neg y \vee \ldots \vee z)}_{\text{Clause}} \wedge (\neg x \vee \neg z \vee \ldots \vee \neg y) \wedge \ldots$$

- Parameter: The number of variables n.
- Question: Can we find a truth assignment such that each clause contains at least one *true* and one *false* literal?

Compare to d-CNF-Sat, where we only require at least one *true* literal per clause.

Not-All-Equal Satisfiability: Lower bound

Prove that this problem does not have a generalized kernel of size $O(n^{d-1-\varepsilon})$, unless $NP \subseteq coNP/poly$.

- Use a linear parameter transformation from d-CNF-Sat to $(d+1)$-NAE-Sat.

Not-All-Equal Satisfiability: Lower bound

d-NAE-Sat does not have a generalized kernel of size $O(n^{d-1-\varepsilon})$, unless $NP \subseteq coNP/poly$.

Proof

- Given formula $\mathcal{F}$ for d-CNF-Sat

$$\mathcal{F} = (x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z) \wedge (\neg x \vee \neg y \vee \neg z).$$

- Transform to formula $\mathcal{G}$ for $(d+1)$-NAE-Sat, add variable b to each clause

$$\mathcal{G} = (x \vee y \vee z \vee b) \wedge (x \vee \neg y \vee \neg z \vee b) \wedge (\neg x \vee \neg y \vee \neg z \vee b).$$

- Show that $\mathcal{F}$ is satisfiable if and only if $\mathcal{G}$ is NAE-satisfiable.

Not-All-Equal Satisfiability: Lower bound

Show that $\mathcal{F}$ is satisfiable if and only if $\mathcal{G}$ is NAE-satisfiable.

$$\begin{aligned}
\mathcal{F} &= (x \lor y \lor z) && \land (x \lor \neg y \lor \neg z) && \land (\neg x \lor \neg y \lor \neg z). \\
\mathcal{G} &= (x \lor y \lor z \lor b) && \land (x \lor \neg y \lor \neg z \lor b) && \land (\neg x \lor \neg y \lor \neg z \lor b).
\end{aligned}$$

Proof

$(\Rightarrow)$ Suppose $\mathcal{F}$ is satisfiable

- Choose the same assignment together with $b =$ *false*.
- Every clause contains one *true* and one *false* literal.

Not-All-Equal Satisfiability: Lower bound

Show that $\mathcal{F}$ is satisfiable if and only if $\mathcal{G}$ is NAE-satisfiable.

$$\begin{aligned}
\mathcal{F} &= (x \vee y \vee z) && \wedge\, (x \vee \neg y \vee \neg z) && \wedge\, (\neg x \vee \neg y \vee \neg z). \\
\mathcal{G} &= (x \vee y \vee z \vee b) && \wedge\, (x \vee \neg y \vee \neg z \vee b) && \wedge\, (\neg x \vee \neg y \vee \neg z \vee b).
\end{aligned}$$

Proof

$(\Leftarrow)$ Suppose $\mathcal{G}$ is NAE-satisfiable

- Suppose $b =$ *false*, choose the same assignment for $\mathcal{F}$.
- If $b =$ *true*, choose the opposite assignment for $\mathcal{F}$.

Not-All-Equal Satisfiability: Upper bound

d-NAE-Sat does not have a generalized kernel of size $O(n^{d-1-\varepsilon})$, unless $\mathrm{NP} \subseteq \mathrm{coNP/poly}$.

- Is this a tight bound?
  - Not trivial.
- Provide a generalized kernel via d-*Hypergraph* 2-*Colorability*

$$\underbrace{(x, k)}_{d\text{-NAE-Sat}} \xrightarrow{\text{lpt}} \underbrace{(x', k')}_{d\text{-Hypergraph } 2\text{-Colorability}} \xrightarrow{\text{kernel}} \underbrace{(x'', k'')}_{d\text{-Hypergraph } 2\text{-Colorability}}$$

d-HYPERGRAPH 2-COLORABILITY

- Input: A hypergraph, where every edge contains at *at most* d vertices.
- Parameter: The number of vertices n.
- Question: Can we color each vertex with red/blue such that every edge contains at least one red and one blue vertex?

d-Hypergraph 2-Colorability

Lower bound

d-Hypergraph 2-colorability does not have a generalized kernel of size at most $O(n^{d-1-\varepsilon})$, unless $\mathrm{NP} \subseteq \mathrm{coNP/poly}$.

Kernel

d-Hypergraph 2-colorability has a kernel with $O(n^{d-1})$ edges.

d-Hypergraph 2-Colorability: Kernel

For simplicity, let every edge have *exactly* d vertices, let the edges be $e_1, \dots, e_m$. Enumerate all subsets of V that have size $d-1$ as $S_1, S_2, \dots, S_\ell$
Create the following matrix M.

$$\begin{array}{cc} & \begin{array}{ccccc} e_1 & e_2 & \dots & e_m \end{array} \\ \begin{array}{c} S_1 \\ S_2 \\ \dots \\ S_\ell \end{array} & \begin{pmatrix} S_1 \subseteq e_1 & S_1 \subseteq e_2 & \dots & S_1 \subseteq e_m \\ S_2 \subseteq e_1 & S_2 \subseteq e_2 & \dots & S_2 \subseteq e_m \\ \dots & \dots & \dots & \dots \\ S_\ell \subseteq e_1 & S_\ell \subseteq e_2 & \dots & S_\ell \subseteq e_m \end{pmatrix} \end{array}$$

Compute a base of the columns of this matrix. This results in a subset of the edges of G.

d-Hypergraph 2-Colorability: Kernel

Size

- Matrix M has at most $\binom{n}{d-1} \leqslant n^{d-1}$ rows.
- Any base of M contains at most n^{d-1} edges.
- Storage per edge: $O(d \log n)$ bits.
- Total number of bits: $O(n^{d-1} d \log n)$.

Correctness

- If all edges in the base are split
- all remaining edges are split.

d-NAE-Sat

Lower bound

No generalized kernel of size $O(n^{d-1-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathrm{NP} \subseteq \mathrm{coNP}/\mathrm{poly}$.

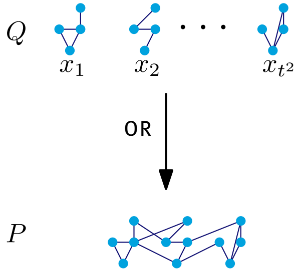
Generalized kernel

Generalized kernel of size $O(n^{d-1} \cdot d \cdot \log n)$

- Linear parameter transformation to d-Hypergraph 2-Colorability
- Kernel for d-Hypergraph 2-Colorability

Degree-2 cross-composition

- Think of an appropriate NP-hard problem Q
- Give a polynomial time algorithm
- Input: t^2 instances of Q: $x_1, \ldots, x_{t^2}$
  - We can assume they are *similar*
- Output: An instance (y, k) of P, where
  - $k = O(t \cdot \max |x_i|^c)$
  - (y, k) is a logical OR of the input
    - (y, k) is a YES-instance iff at least one x_i is



Proving lower bounds: Cross-composition

Theorem

If there exists a degree-2 cross-composition, P has no generalized kernel of size $O(k^{2-\varepsilon})$

- Assuming $NP \not\subseteq coNP/poly$.

Proof sketch

The combination of a degree-2 cross-composition and a generalized kernel of size $O(k^{2-\varepsilon})$ results in an algorithm we consider unlikely to exist.

- Suppose for contradiction that P has a generalized kernel of size $O(k^{1.9})$

Proving lower bounds: Cross-composition

n^{100} inputs
Size n each

x_1

x_2

$\cdots$

$x_{n^{100}}$

Q

Proving lower bounds: Cross-composition

n^{100} inputs
Size n each

x_1

x_2

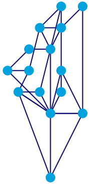
$\cdots$

$x_{n^{100}}$

Q

degree-2 cross-composition

1 input
$k = O(n^{50} \cdot \max |x_i|^c)$
Size: unknown



P

Proving lower bounds: Cross-composition

n^{100} inputs
Size n each

x_1

x_2

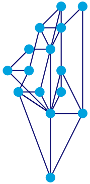
$\cdots$

$x_{n^{100}}$

Q

degree-2 cross-composition

1 input
$k = O(n^{51})$
Size: unknown



P

Proving lower bounds: Cross-composition

n^{100} inputs
Size n each

x_1

x_2

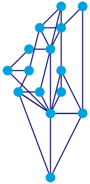
$\cdots$

$x_{n^{100}}$

Q

degree-2 cross-composition

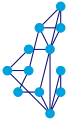
1 input
$k = O(n^{51})$
Size: unknown



P

generalized kernel

1 input
$k' = O((n^{51})^{1.9})$
Size = $O((n^{51})^{1.9})$



P'

Proving lower bounds: Cross-composition

n^{100} inputs
Size n each

x_1

x_2

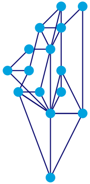
$\cdots$

$x_{n^{100}}$

Q

degree-2 cross-composition

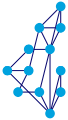
1 input
$k = O(n^{51})$
Size: unknown



P

generalized kernel

1 input
$k' = O(n^{97})$
Size = $O(n^{97})$

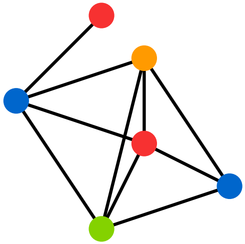


P'

4-Coloring

Goal

Prove that there is no kernel of size $O(|V|^{2-\varepsilon})$ for 4-COLORING.



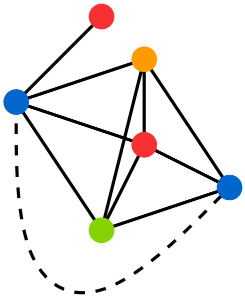
4-List Coloring

We use list coloring with 4 colors in total, instead of 4-coloring.

- Every vertex has a list of allowed colors
  - Subset of $\{r, g, b, o\}$
- Can be transformed back to 4-coloring using 4 vertices

Goal

Prove that there is no kernel of size $O(|V|^{2-\varepsilon})$ for 4-COLORING.



4-List Coloring

We use list coloring with 4 colors in total, instead of 4-coloring.

- Every vertex has a list of allowed colors
  - Subset of $\{r, g, b, o\}$
- Can be transformed back to 4-coloring using 4 vertices

4-Coloring

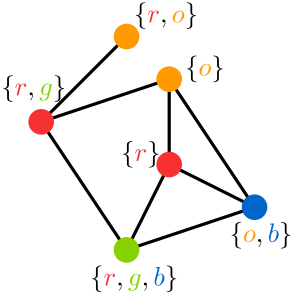
Goal

Prove that there is no kernel of size $O(|V|^{2-\varepsilon})$ for 4-COLORING.

4-List Coloring

We use list coloring with 4 colors in total, instead of 4-coloring.

- Every vertex has a list of allowed colors
  - Subset of $\{r, g, b, o\}$
- Can be transformed back to 4-coloring using 4 vertices



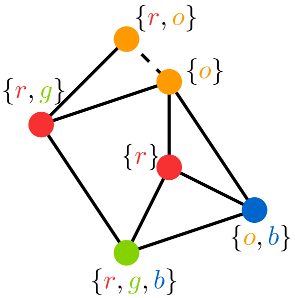
Goal

Prove that there is no kernel of size $O(|V|^{2-\varepsilon})$ for 4-COLORING.

4-List Coloring

We use list coloring with 4 colors in total, instead of 4-coloring.

- Every vertex has a list of allowed colors
  - Subset of $\{r, g, b, o\}$
- Can be transformed back to 4-coloring using 4 vertices

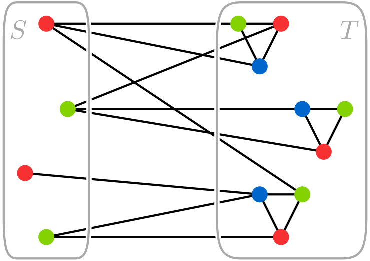


4-Coloring: Cross-composition

NP-hard starting problem

2-3-Coloring on Triangle-Split Graphs

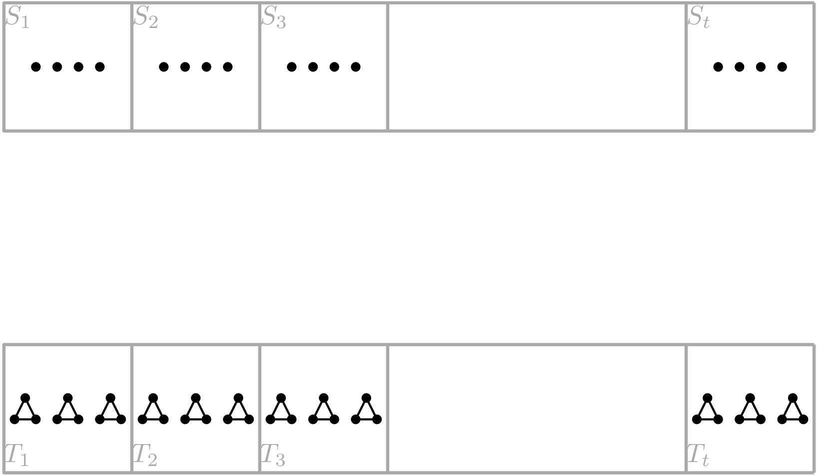
- Input: Graph $G = (S \cup T, E)$ where S is an independent set and T consists of disjoint triangles.
- Question: Does G have a proper 3-coloring, such that S is colored using only 2 colors?
  - We call this a 2-3-*coloring* of G.



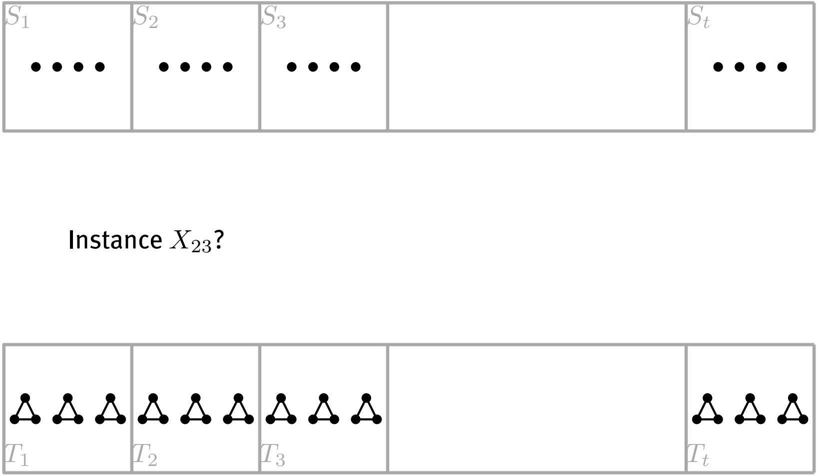
4-Coloring: Cross-composition

- Assume we have t^2 instances
- Let $|S| = n$ and $|T| = 3m$ for all instances
- Construct an instance G for 4-coloring
  - Polynomial time
  - At most $O(t \cdot (n + m))$ vertices
  - OR of all inputs
- We cannot use t^2 vertices $\rightarrow$ we cannot copy all vertices
  - But we can keep all edges!
- Enumerate instances as X_{ij}, where $i = 1, \ldots, t$

4-Coloring: First idea

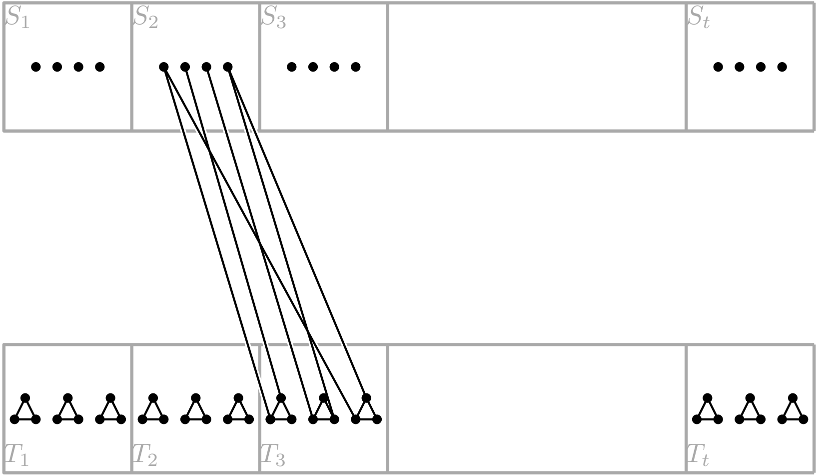


4-Coloring: First idea

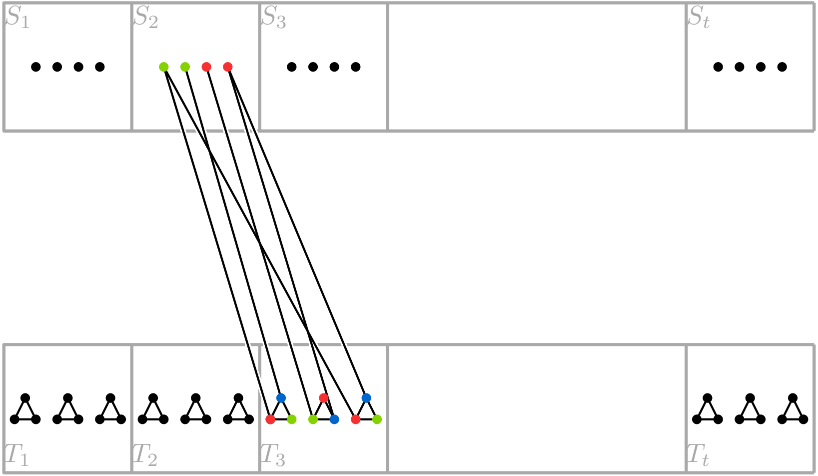


Instance X_{23}?

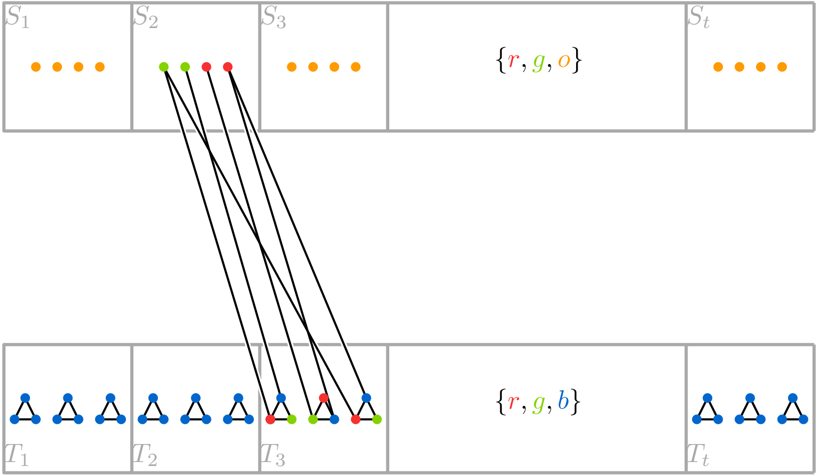
4-Coloring: First idea



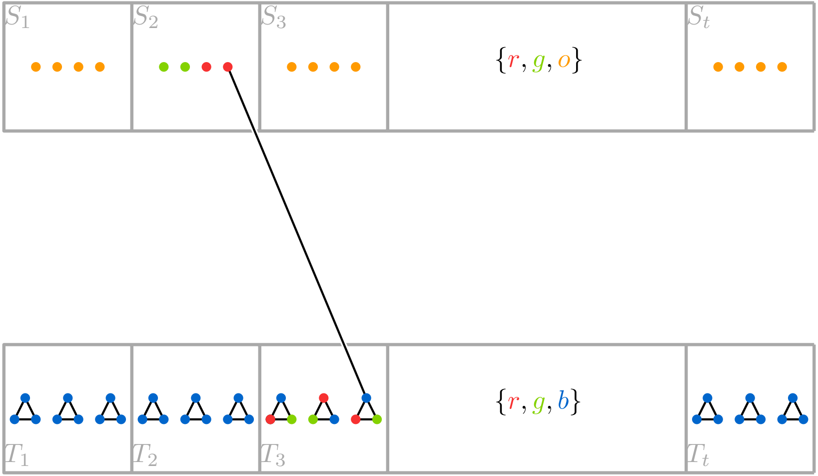
4-Coloring: First idea



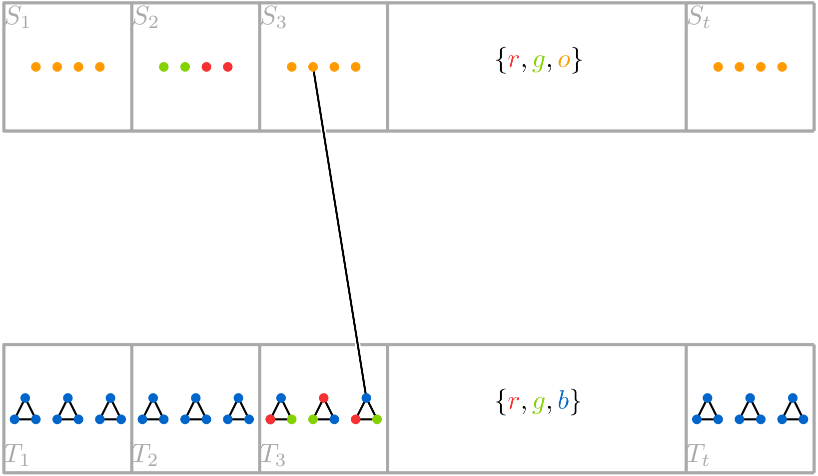
4-Coloring: First idea



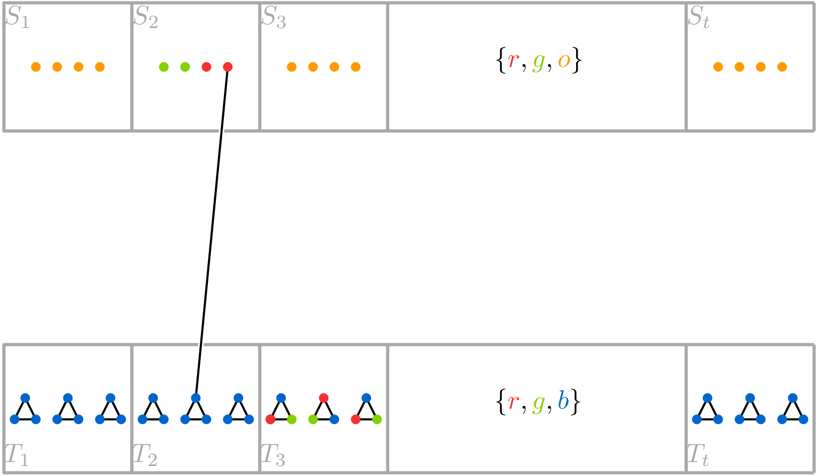
4-Coloring: First idea



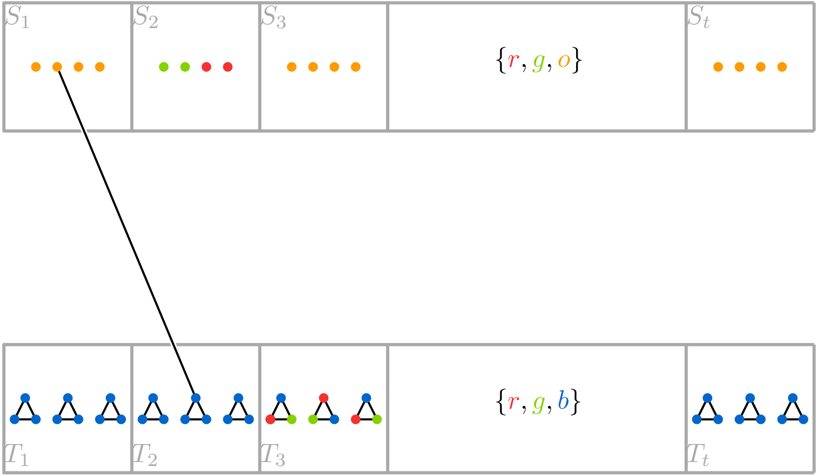
4-Coloring: First idea



4-Coloring: First idea

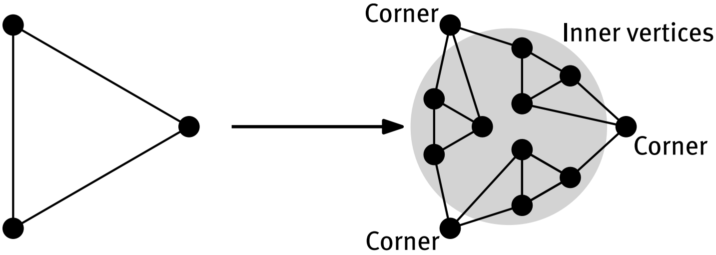


4-Coloring: First idea



4-Coloring: Triangular gadget

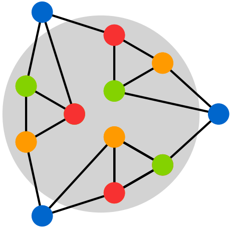
This is not a valid coloring of the triangles. Replace them:



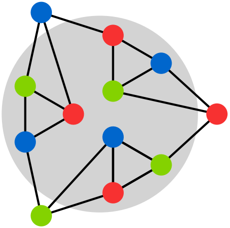
Allow the new vertices to be red, green, orange, or blue.

Useful properties

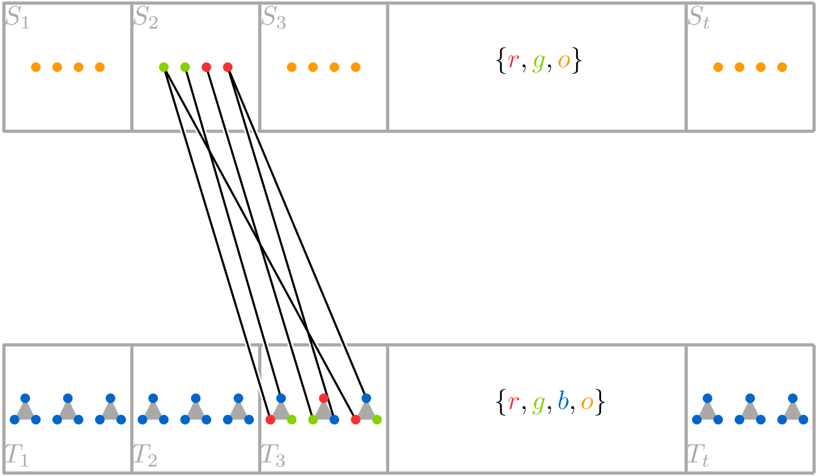
If orange is allowed in the inner vertices, we can color all corners with blue.



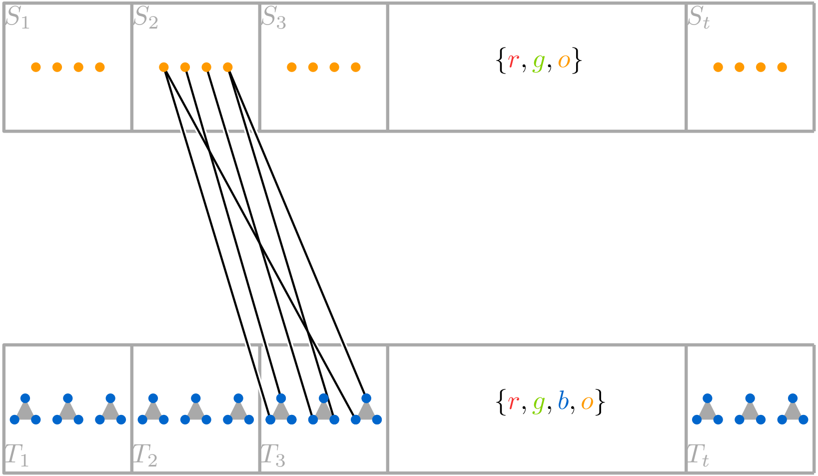
If 3-colored: ordinary triangle. All corners get distinct color.



4-Coloring: Improved



4-Coloring: Improved

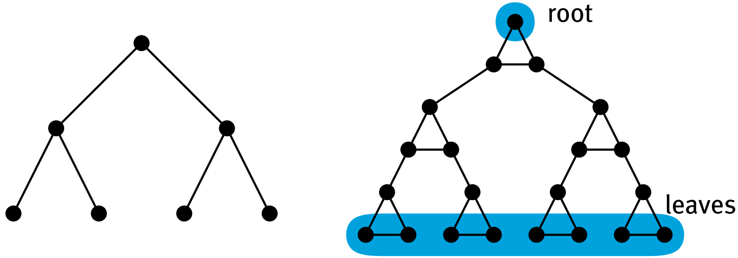


4-Coloring: Treegadgets

Make sure that at least one S_i and one T_j are colored using red, green, and blue.

Treegadget

A *treegadget* is a balanced binary tree where every vertex is replaced by a triangle:

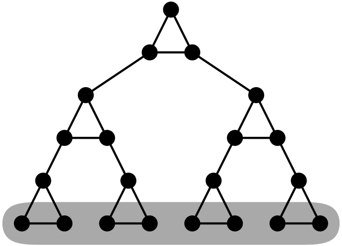


4-Coloring: Treegadgets

Suppose we want to 3-color a gadget, using red, green and orange.

Property 1

If none of the leaves is colored orange, the root must be orange.

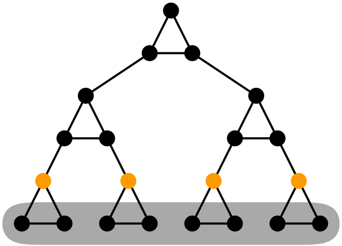


4-Coloring: Treegadgets

Suppose we want to 3-color a gadget, using red, green and orange.

Property 1

If none of the leaves is colored orange, the root must be orange.

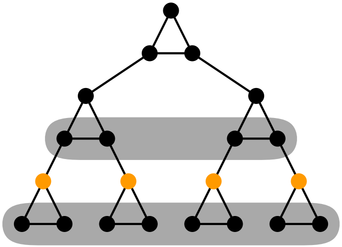


4-Coloring: Treegadgets

Suppose we want to 3-color a gadget, using red, green and orange.

Property 1

If none of the leaves is colored orange, the root must be orange.

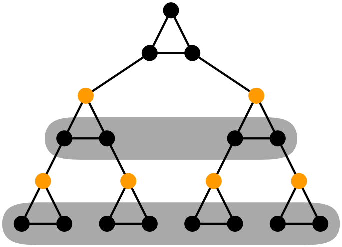


4-Coloring: Treegadgets

Suppose we want to 3-color a gadget, using red, green and orange.

Property 1

If none of the leaves is colored orange, the root must be orange.

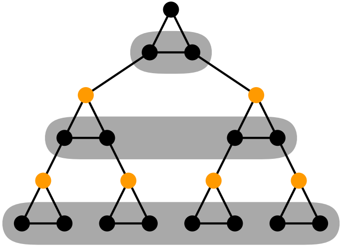


4-Coloring: Treegadgets

Suppose we want to 3-color a gadget, using red, green and orange.

Property 1

If none of the leaves is colored orange, the root must be orange.

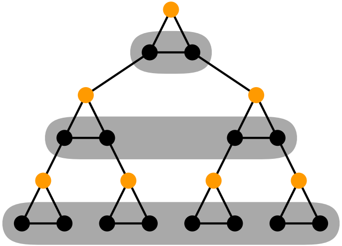


4-Coloring: Treegadgets

Suppose we want to 3-color a gadget, using red, green and orange.

Property 1

If none of the leaves is colored orange, the root must be orange.

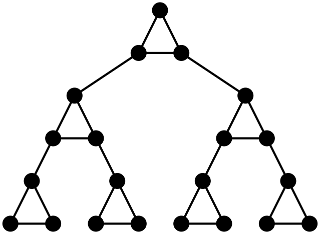


4-Coloring: Treegadgets

Suppose we want to 3-color a gadget, using red, green and orange.

Property 2

We can extend a coloring of the leaves, to color the entire tree. If at least one of the leaves is colored orange, the root can be red or green.

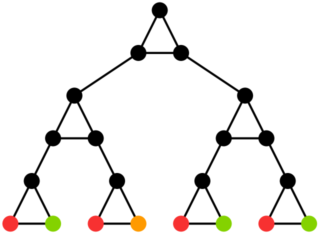


4-Coloring: Treegadgets

Suppose we want to 3-color a gadget, using red, green and orange.

Property 2

We can extend a coloring of the leaves, to color the entire tree. If at least one of the leaves is colored orange, the root can be red or green.

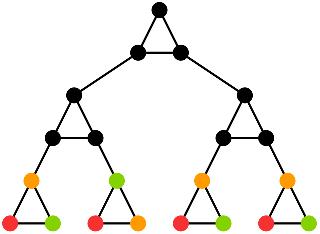


4-Coloring: Treegadgets

Suppose we want to 3-color a gadget, using red, green and orange.

Property 2

We can extend a coloring of the leaves, to color the entire tree. If at least one of the leaves is colored orange, the root can be red or green.

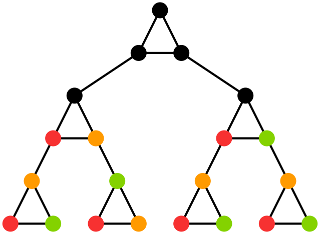


4-Coloring: Treegadgets

Suppose we want to 3-color a gadget, using red, green and orange.

Property 2

We can extend a coloring of the leaves, to color the entire tree. If at least one of the leaves is colored orange, the root can be red or green.

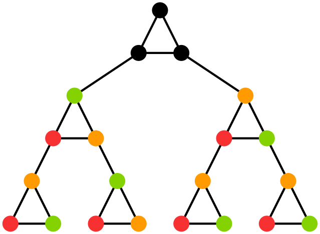


4-Coloring: Treegadgets

Suppose we want to 3-color a gadget, using red, green and orange.

Property 2

We can extend a coloring of the leaves, to color the entire tree. If at least one of the leaves is colored orange, the root can be red or green.

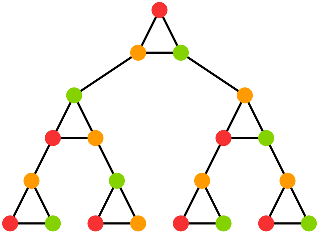


4-Coloring: Treegadgets

Suppose we want to 3-color a gadget, using red, green and orange.

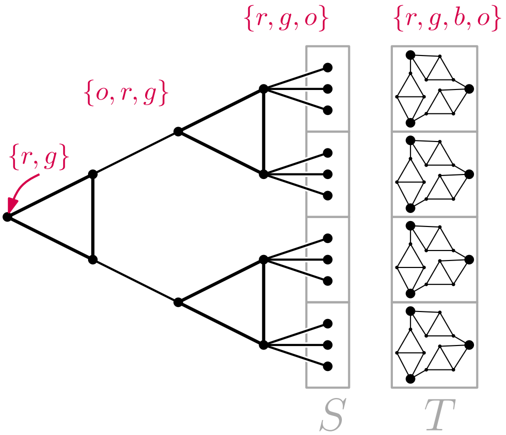
Property 2

We can extend a coloring of the leaves, to color the entire tree. If at least one of the leaves is colored orange, the root can be red or green.



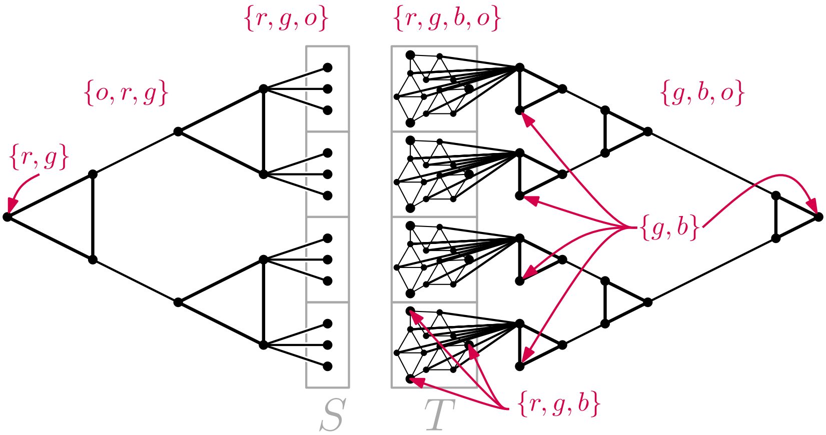
4-Coloring: Cross-composition

Ensure one group in S is colored with red and green.



4-Coloring: Cross-composition

Ensure one group in T is colored with red, green and orange.



4-Coloring: Cross-composition

To prove

- The number of vertices of G is allowed: $O(t \cdot \max |X_{ij}|)$.
- Can be done in polynomial time.
- ✓ If some X_{ij} is 2-3-colorable, G is 4-colorable.
- ✓ If G is 4-colorable, there exists an X_{ij} that is 2-3-colorable.

4-Coloring: Cross-composition

Proof: $|V|$

Count the number of vertices

- S: $t \cdot n$ vertices
- T: $t \cdot 12m$ vertices
- Gadgets: $O(t)$ vertices

Total: $O(t \cdot (n + m))$ vertices as required.

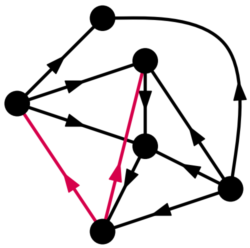
Proof: Polynomial time

The graph has polynomial size and is straightforward to construct.

Results

Problems without a generalized kernel of size $O(|V|^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathrm{NP} \subseteq \mathrm{coNP/poly}$.

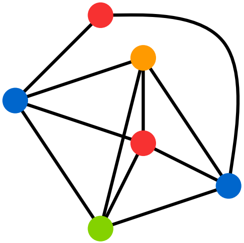
- Feedback Arc Set
- 4-Coloring
  - And thereby k-coloring for $k \geqslant 4$
- Hamiltonian Cycle
- Dominating Set
  - Non Blocker
- Connected Dominating Set
  - Maximum Leaf Spanning Tree



Results

Problems without a generalized kernel of size $O(|V|^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $NP \subseteq coNP/poly$.

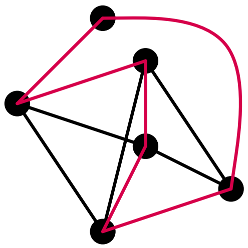
- Feedback Arc Set
- 4-Coloring
  - And thereby k-coloring for $k \geqslant 4$
- Hamiltonian Cycle
- Dominating Set
  - Non Blocker
- Connected Dominating Set
  - Maximum Leaf Spanning Tree



Results

Problems without a generalized kernel of size $O(|V|^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $NP \subseteq coNP/poly$.

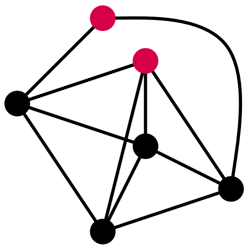
- Feedback Arc Set
- 4-Coloring
  - And thereby k-coloring for $k \geqslant 4$
- Hamiltonian Cycle
- Dominating Set
  - Non Blocker
- Connected Dominating Set
  - Maximum Leaf Spanning Tree



Results

Problems without a generalized kernel of size $O(|V|^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathrm{NP} \subseteq \mathrm{coNP/poly}$.

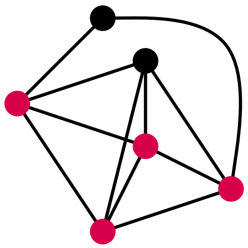
- Feedback Arc Set
- 4-Coloring
  - And thereby k-coloring for $k \geqslant 4$
- Hamiltonian Cycle
- Dominating Set
  - Non Blocker
- Connected Dominating Set
  - Maximum Leaf Spanning Tree



Results

Problems without a generalized kernel of size $O(|V|^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $NP \subseteq coNP/poly$.

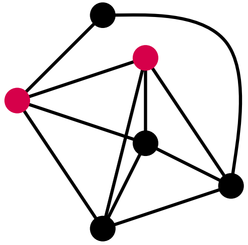
- Feedback Arc Set
- 4-Coloring
  - And thereby k-coloring for $k \geqslant 4$
- Hamiltonian Cycle
- Dominating Set
  - Non Blocker
- Connected Dominating Set
  - Maximum Leaf Spanning Tree



Results

Problems without a generalized kernel of size $O(|V|^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $NP \subseteq coNP/poly$.

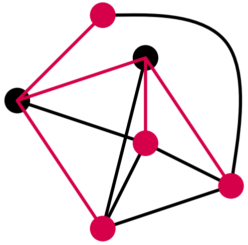
- Feedback Arc Set
- 4-Coloring
  - And thereby k-coloring for $k \geqslant 4$
- Hamiltonian Cycle
- Dominating Set
  - Non Blocker
- Connected Dominating Set
  - Maximum Leaf Spanning Tree



Results

Problems without a generalized kernel of size $O(|V|^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $NP \subseteq coNP/poly$.

- Feedback Arc Set
- 4-Coloring
  - And thereby k-coloring for $k \geqslant 4$
- Hamiltonian Cycle
- Dominating Set
  - Non Blocker
- Connected Dominating Set
  - Maximum Leaf Spanning Tree



Results

Problems that do not have a generalized kernel of size $O(n^{d-1-\varepsilon})$, unless $NP \subseteq coNP/poly$.

- d-Hypergraph 2-Colorability
- d-NAE-Sat

And these bounds are tight.

Open problems

- 3-Coloring: Kernel of size $O(|V|^{2-\varepsilon})$ – or not??
- Are there graph problems with kernels with $O(|V|^{2-\varepsilon})$ edges?

- Questions or remarks?

Open problems

- 3-Coloring: Kernel of size $O(|V|^{2-\varepsilon})$ – or not??
- Are there graph problems with kernels with $O(|V|^{2-\varepsilon})$ edges?

- Questions or remarks?