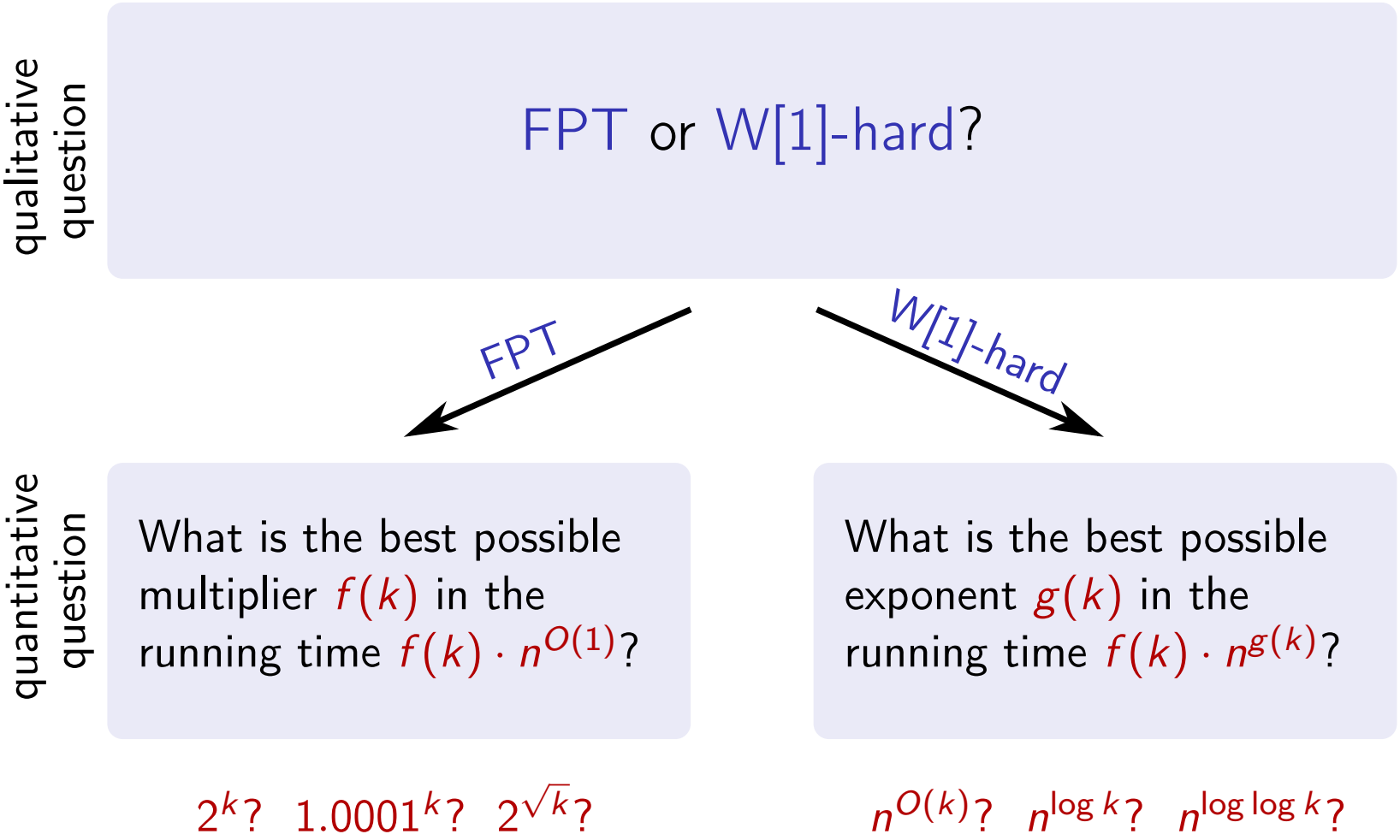
Shift of focus

qualitative question

FPT or W[1]-hard?

Shift of focus



Better algorithms for Vertex Cover

- We have seen a $2^k \cdot n^{O(1)}$ time algorithm.
- Easy to improve to, e.g., $1.618^k \cdot n^{O(1)}$.
- Current best $f(k)$: $1.2738^k \cdot n^{O(1)}$.
- Lower bounds?
  - Is, say, $1.001^k \cdot n^{O(1)}$ time possible?
  - Is $2^{k/\log k} \cdot n^{O(1)}$ time possible?

Better algorithms for Vertex Cover

- We have seen a $2^k \cdot n^{O(1)}$ time algorithm.
- Easy to improve to, e.g., $1.618^k \cdot n^{O(1)}$.
- Current best $f(k)$: $1.2738^k \cdot n^{O(1)}$.
- Lower bounds?
  - Is, say, $1.001^k \cdot n^{O(1)}$ time possible?
  - Is $2^{k/\log k} \cdot n^{O(1)}$ time possible?

Of course, for all we know, it is possible that P = NP and Vertex Cover is polynomial-time solvable.

$\Rightarrow$ We can hope only for conditional lower bounds.

Exponential Time Hypothesis (ETH)

3CNF: ϕ is a conjuction of clauses, where each clause is a disjunction of at most 3 literals (= a variable or its negation), e.g., $(x_1 \vee x_3 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3) \vee (x_1 \vee x_2 \vee x_4)$.

3SAT: given a 3CNF formula ϕ with n variables and m clauses, decide whether ϕ is satisfiable.

- Current best algorithm is 1.30704^n [Hertli 2011].
- Can we do **significantly** better, e.g, $2^{O(n/\log n)}$?

Exponential Time Hypothesis (ETH)

3CNF: ϕ is a conjuction of clauses, where each clause is a disjunction of at most 3 literals (= a variable or its negation), e.g., $(x_1 \vee x_3 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3) \vee (x_1 \vee x_2 \vee x_4)$.

$3\mathrm{SAT}$: given a 3CNF formula ϕ with n variables and m clauses, decide whether ϕ is satisfiable.

- Current best algorithm is 1.30704^n [Hertli 2011].
- Can we do **significantly** better, e.g, $2^{O(n/\log n)}$?

Hypothesis introduced by Impagliazzo, Paturi, and Zane in 2001:

Exponential Time Hypothesis (ETH) [consequence of]

There is no $2^{o(n)}$-time algorithm for n-variable $3\mathrm{SAT}$.

Exponential Time Hypothesis (ETH)

3CNF: ϕ is a conjuction of clauses, where each clause is a disjunction of at most 3 literals (= a variable or its negation), e.g., $(x_1 \vee x_3 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3) \vee (x_1 \vee x_2 \vee x_4)$.

$3\mathrm{SAT}$: given a 3CNF formula ϕ with n variables and m clauses, decide whether ϕ is satisfiable.

- Current best algorithm is 1.30704^n [Hertli 2011].
- Can we do **significantly** better, e.g, $2^{O(n/\log n)}$?

Hypothesis introduced by Impagliazzo, Paturi, and Zane in 2001:

Exponential Time Hypothesis (ETH) [real statement]

There is a constant $\delta > 0$ such that there is no $O(2^{\delta n})$ time algorithm for $3\mathrm{SAT}$.

Sparsification

Exponential Time Hypothesis (ETH) [consequence of]

There is no $2^{o(n)}$-time algorithm for n-variable $\mathrm{3SAT}$.

Observe: an n-variable $\mathrm{3SAT}$ formula can have $m = \Omega(n^3)$ clauses.

Are there algorithms that are subexponential in the size $n + m$ of the $\mathrm{3SAT}$ formula?

Sparsification

Exponential Time Hypothesis (ETH) [consequence of]

There is no $2^{o(n)}$-time algorithm for n-variable $3\mathrm{SAT}$.

Observe: an n-variable $3\mathrm{SAT}$ formula can have $m = \Omega(n^3)$ clauses.

Are there algorithms that are subexponential in the size $n + m$ of the $3\mathrm{SAT}$ formula?

Sparsification Lemma

There is a $2^{o(n)}$-time algorithm for n-variable $3\mathrm{SAT}$.

$\Updownarrow$

There is a $2^{o(n+m)}$-time algorithm for n-variable m-clause $3\mathrm{SAT}$.

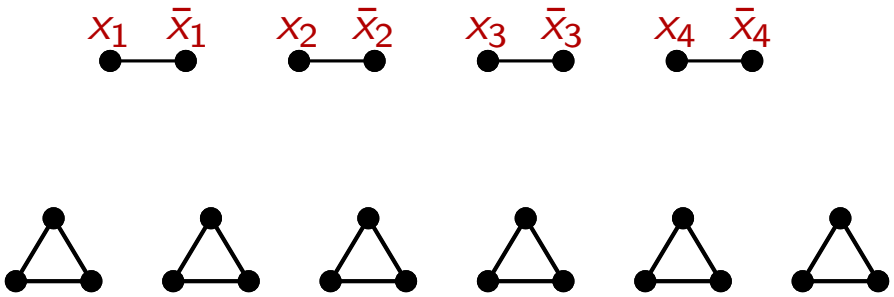
Intuitively: When considering a hard $3\mathrm{SAT}$ instance, we can assume that it has $m = O(n)$ clauses.

Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma

There is no $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

The textbook reduction from 3SAT to Vertex Cover:



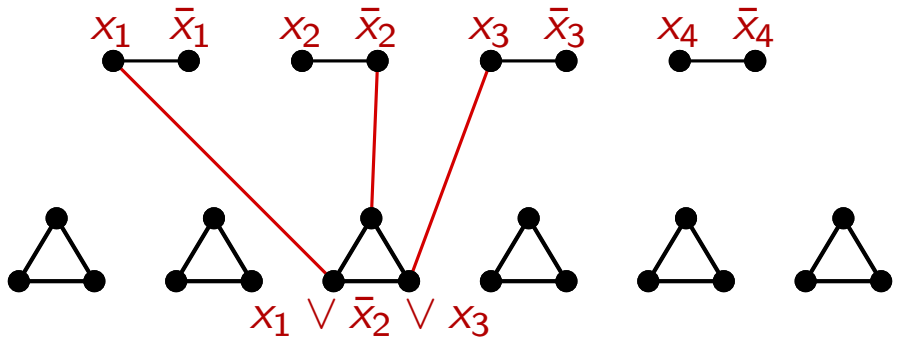
Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma

There is no $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

The textbook reduction from 3SAT to Vertex Cover:

formula is satisfiable $\Leftrightarrow$ there is a vertex cover of size $n + 2m$



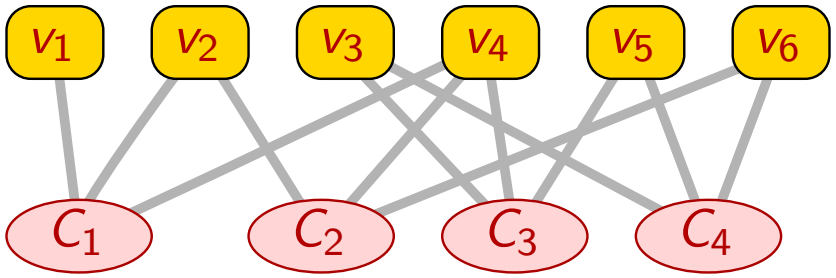
Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma

There is no $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

The textbook reduction from 3SAT to Vertex Cover:

3SAT formula ϕ	$\Rightarrow$	Graph G
n variables		$O(n+m)$ vertices
m clauses		$O(n+m)$ edges

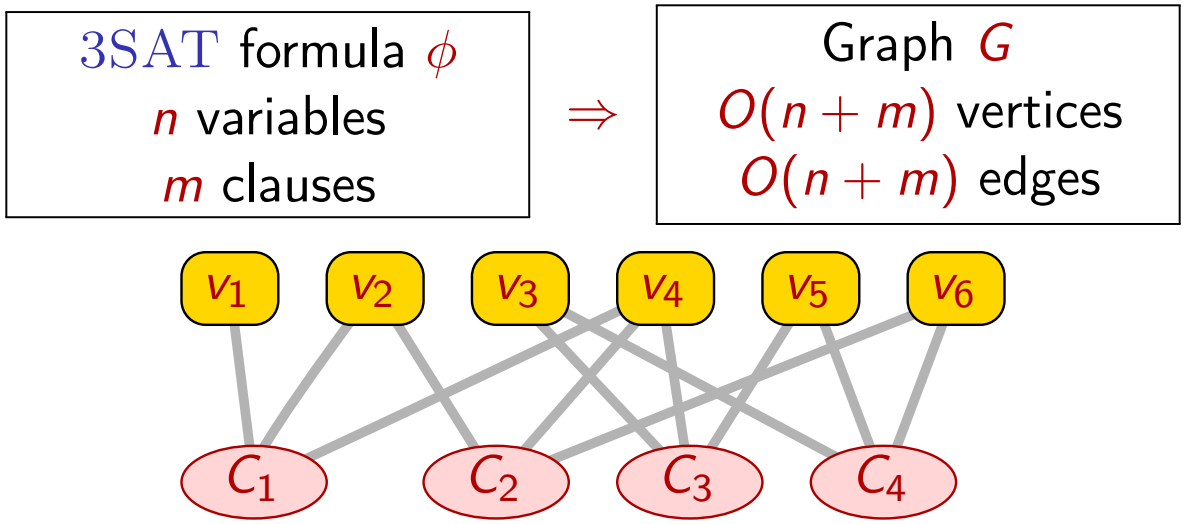


Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma

There is no $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

The textbook reduction from 3SAT to Vertex Cover:



Corollary

Assuming ETH, there is no $2^{o(n)}$ algorithm for Vertex Cover on an n-vertex graph.

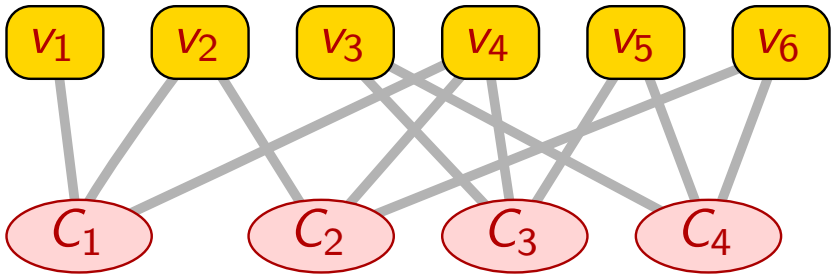
Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma

There is no $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

The textbook reduction from 3SAT to Vertex Cover:

3SAT formula ϕ	$\Rightarrow$	Graph G
n variables		$O(n+m)$ vertices
m clauses		$O(n+m)$ edges



Corollary

Assuming ETH, there is no $2^{o(k)} \cdot n^{O(1)}$ algorithm for Vertex Cover.

Other problems

There are polytime reductions from 3SAT to many problems such that the reduction creates a graph with $O(n+m)$ vertices/edges.

Consequence: Assuming ETH, the following problems cannot be solved in time $2^{o(n)}$ and hence in time $2^{o(k)} \cdot n^{O(1)}$ (but $2^{O(k)} \cdot n^{O(1)}$ time algorithms are known):

- Vertex Cover
- Longest Cycle
- Feedback Vertex Set
- Multiway Cut
- Odd Cycle Transversal
- Steiner Tree
- ...

Seems to be the natural behavior of FPT problems?

Lower bounds based on ETH

Exponential Time Hypothesis (ETH)

There is no $2^{o(m)}$-time algorithm for m-clause 3SAT.

The textbook reduction from 3SAT to 3-Coloring:

3SAT formula ϕ	$\Rightarrow$	Graph G
n variables		$O(n+m)$ vertices
m clauses		$O(n+m)$ edges

Corollary

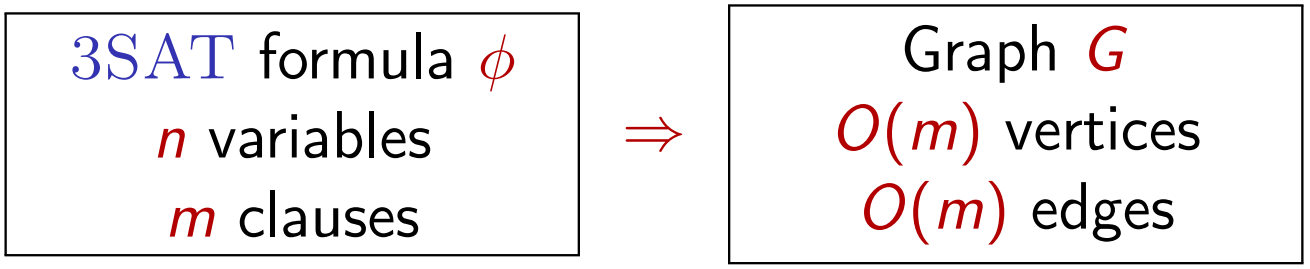
Assuming ETH, there is no $2^{o(n)}$ algorithm for 3-Coloring on an n-vertex graph G.

Lower bounds based on ETH

Exponential Time Hypothesis (ETH)

There is no $2^{o(m)}$-time algorithm for m-clause 3SAT.

The textbook reduction from 3SAT to 3-Coloring:



Corollary

Assuming ETH, there is no $2^{o(n)}$ algorithm for 3-Coloring on an n-vertex graph G.

Transfering bounds

There are polynomial-time reductions from, say, 3-Coloring to many other problems such that the reduction increases the number of vertices by at most a constant factor.

Consequence: Assuming ETH, there is no $2^{o(n)}$ time algorithm on n-vertex graphs for

- Independent Set
- Clique
- Dominating Set
- Vertex Cover
- Hamiltonian Path
- Feedback Vertex Set
- ...

Transfering bounds

There are polynomial-time reductions from, say, 3-Coloring to many other problems such that the reduction increases the number of vertices by at most a constant factor.

Consequence: Assuming ETH, there is no $2^{o(k)} \cdot n^{O(1)}$ time algorithm for

- k-Independent Set
- k-Clique
- k-Dominating Set
- k-Vertex Cover
- k-Path
- k-Feedback Vertex Set
- ...

Transfering bounds

There are polynomial-time reductions from, say, 3-Coloring to many other problems such that the reduction increases the number of vertices by at most a constant factor.

Consequence: Assuming ETH, there is no $2^{o(k)} \cdot n^{O(1)}$ time algorithm for

- ~~k-Independent Set~~
- ~~k-Clique~~
- ~~k-Dominating Set~~
- k-Vertex Cover
- k-Path
- k-Feedback Vertex Set
- ...

Transfering bounds

There are polynomial-time reductions from, say, 3-Coloring to many other problems such that the reduction increases the number of vertices by at most a constant factor.

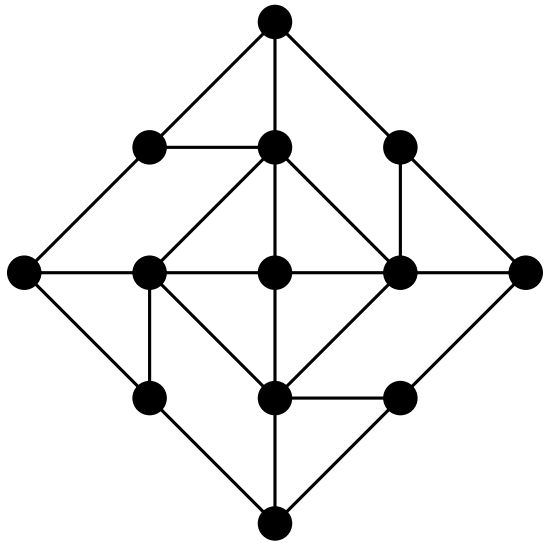
Consequence: Assuming ETH, there is no $2^{o(k)} \cdot n^{O(1)}$ time algorithm for

- ~~k-Independent Set~~
- ~~k-Clique~~
- ~~k-Dominating Set~~
- k-Vertex Cover
- k-Path
- k-Feedback Vertex Set
- ...

Lower bounds based on ETH

What about 3-Coloring on planar graphs?

The textbook reduction from 3-Coloring to Planar 3-Coloring uses a "crossover gadget" with 4 external connectors:

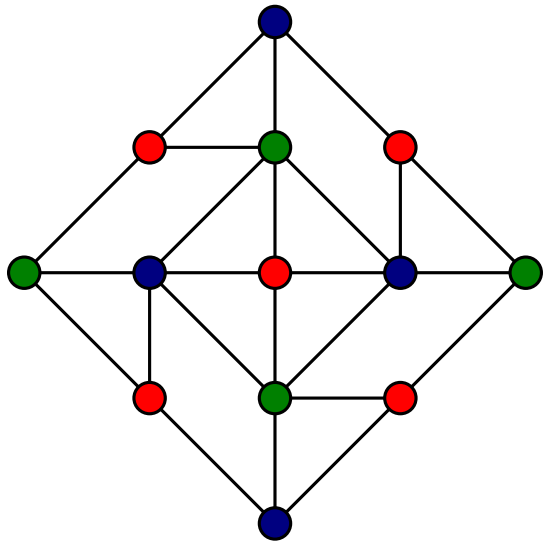


- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to the whole gadget.
- If two edges cross, replace them with a crossover gadget.

Lower bounds based on ETH

What about 3-Coloring on planar graphs?

The textbook reduction from 3-Coloring to Planar 3-Coloring uses a "crossover gadget" with 4 external connectors:

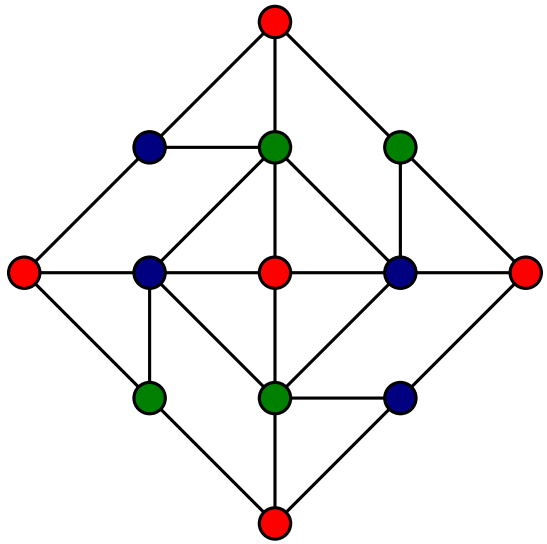


- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to the whole gadget.
- If two edges cross, replace them with a crossover gadget.

Lower bounds based on ETH

What about 3-Coloring on planar graphs?

The textbook reduction from 3-Coloring to Planar 3-Coloring uses a "crossover gadget" with 4 external connectors:



- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to the whole gadget.
- If two edges cross, replace them with a crossover gadget.

Lower bounds based on ETH

- The reduction from 3-Coloring to Planar 3-Coloring introduces $O(1)$ new edges/vertices for each crossing.
- A graph with m edges can be drawn with $O(m^2)$ crossings.

3SAT formula ϕ	$\Rightarrow$	Graph G	$\Rightarrow$	Planar graph G'
n variables		$O(m)$ vertices		$O(m^2)$ vertices
m clauses		$O(m)$ edges		$O(m^2)$ edges

Corollary

Assuming ETH, there is no $2^{o(\sqrt{n})}$ algorithm for 3-Coloring on an n-vertex planar graph G.

(Essentially observed by [Cai and Juedes 2001])

Lower bounds for planar problems

Consequence: Assuming ETH, there is no $2^{o(\sqrt{n})}$ time algorithm on n-vertex **planar graphs** for

- Independent Set
- Dominating Set
- Vertex Cover
- Hamiltonian Path
- Feedback Vertex Set
- ...

Lower bounds for planar problems

Consequence: Assuming ETH, there is no $2^{o(\sqrt{k})} \cdot n^{O(1)}$ time algorithm on **planar graphs** for

- k-Independent Set
- k-Dominating Set
- k-Vertex Cover
- k-Path
- k-Feedback Vertex Set
- ...

Lower bounds for planar problems

Consequence: Assuming ETH, there is no $2^{o(\sqrt{k})} \cdot n^{O(1)}$ time algorithm on **planar graphs** for

- k-Independent Set
- k-Dominating Set
- k-Vertex Cover
- k-Path
- k-Feedback Vertex Set
- ...

Note: Reduction to planar graphs does not work for Clique (why?).

Treewidth

Given a tree decomposition of width w, FPT algorithms with running time $2^{O(w)} \cdot n^{O(1)}$ for

- Independent Set
- Dominating Set
- 3-Coloring
- Hamiltonian Cycle
- ...

Treewidth

Given a tree decomposition of width w, FPT algorithms with running time $2^{O(w)} \cdot n^{O(1)}$ for

- Independent Set
- Dominating Set
- 3-Coloring
- Hamiltonian Cycle
- ...

Observation: A $2^{o(w)} \cdot n^{O(1)}$ algorithm implies a $2^{o(n)} \cdot n^{O(1)}$ algorithm.

$\Rightarrow$ Assuming ETH, no $2^{o(w)} \cdot n^{O(1)}$ algorithms for these problems!

Treewidth

The following problems have $w^{O(w)} \cdot n^{O(1)} = 2^{O(w \log w)} \cdot n^{O(1)}$ algorithms:

- Vertex Coloring
- Cycle Packing
- Vertex Disjoint Paths

Treewidth

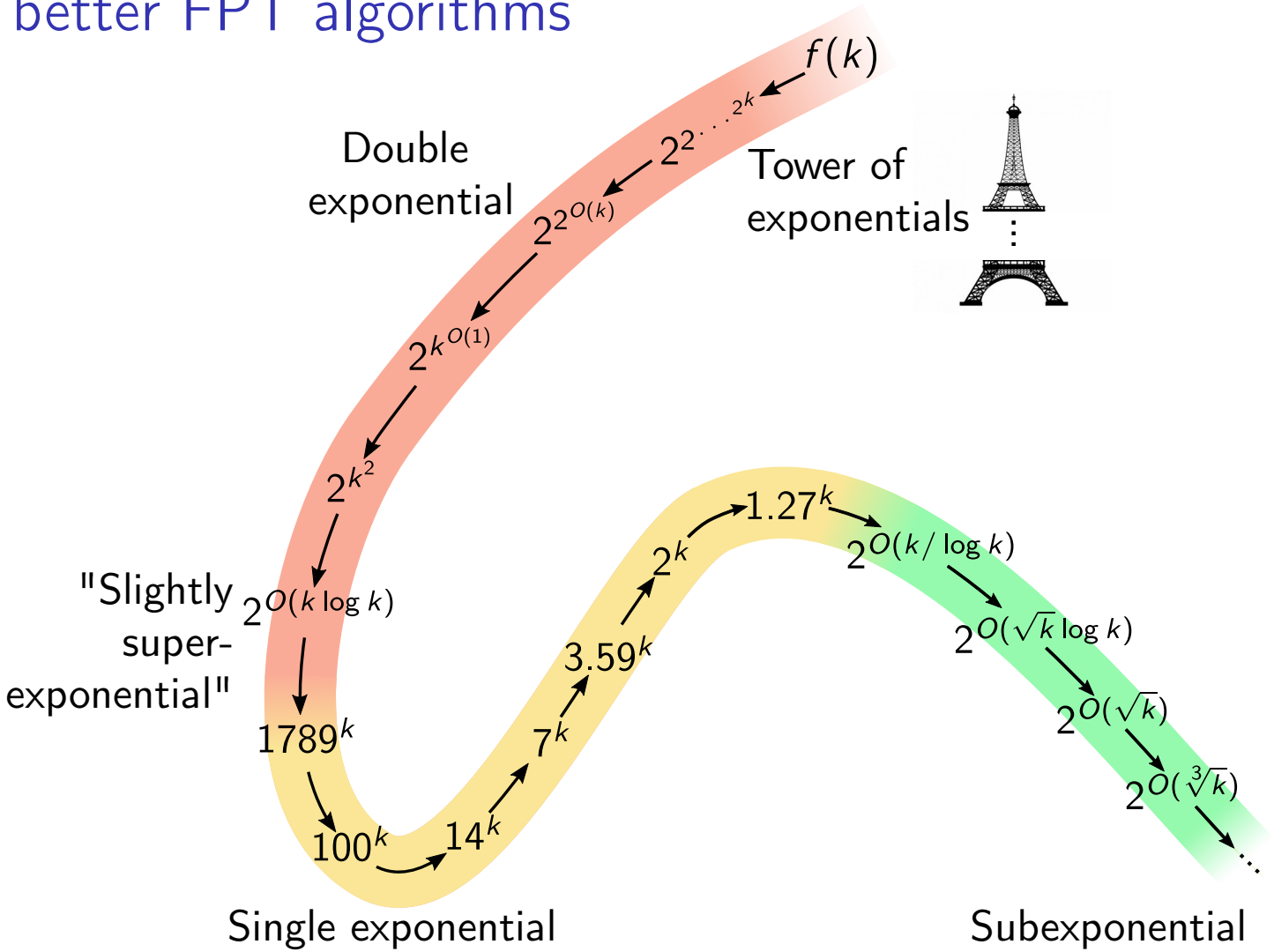
The following problems have $w^{O(w)} \cdot n^{O(1)} = 2^{O(w \log w)} \cdot n^{O(1)}$ algorithms:

- Vertex Coloring
- Cycle Packing
- Vertex Disjoint Paths

. . . and assuming ETH, they do not have $2^{o(w \log w)} \cdot n^{O(1)}$ algorithms.

Proof: Reduce an instance of a graph problem on N vertices to an instance with treewidth $O(N/\log N)$.

The race for better FPT algorithms

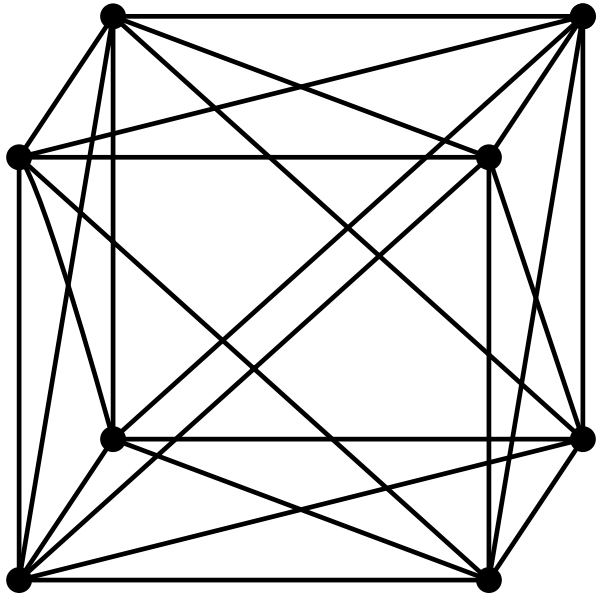


Edge Clique Cover

Edge Clique Cover: Given a graph G and an integer k, cover the edges of G with at most k cliques.

(the cliques need not be edge disjoint)

Equivalently: can G be represented as an intersection graph over a k element universe?

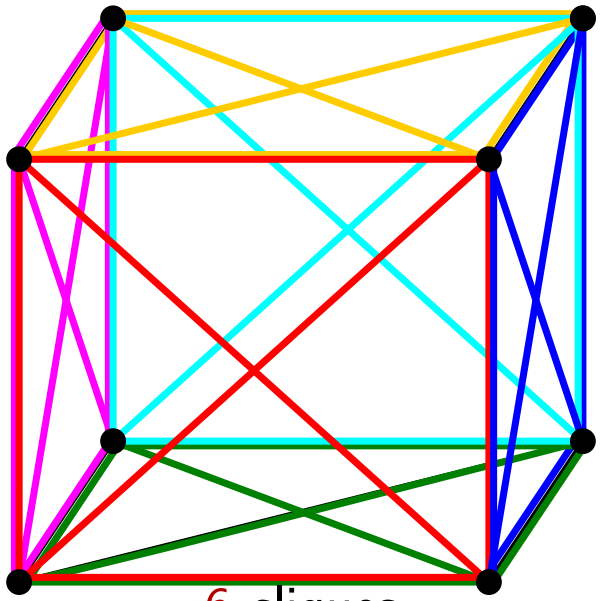


Edge Clique Cover

Edge Clique Cover: Given a graph G and an integer k, cover the edges of G with at most k cliques.

(the cliques need not be edge disjoint)

Equivalently: can G be represented as an intersection graph over a k element universe?



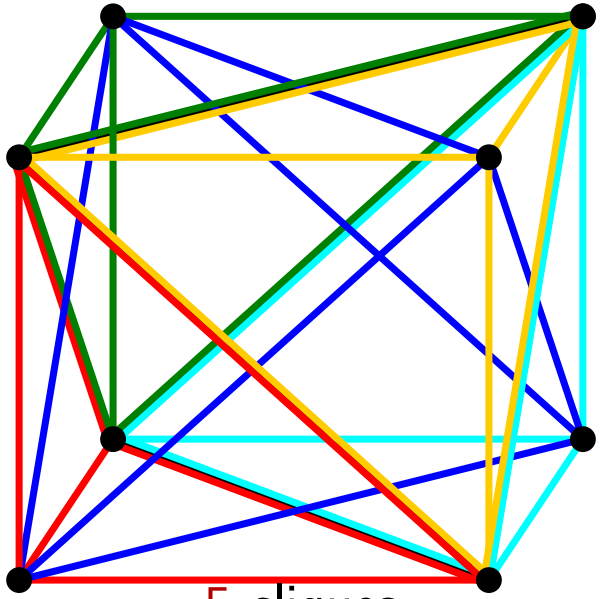
6 cliques

Edge Clique Cover

Edge Clique Cover: Given a graph G and an integer k, cover the edges of G with at most k cliques.

(the cliques need not be edge disjoint)

Equivalently: can G be represented as an intersection graph over a k element universe?



5 cliques

Edge Clique Cover

Edge Clique Cover: Given a graph G and an integer k, cover the edges of G with at most k cliques.

(the cliques need not be edge disjoint)

Simple algorithm (sketch)

- If two adjacent vertices have the same neighborhood ("twins"), then remove one of them.
- If there are no twins and isolated vertices, then $|V(G)| > 2^k$ implies that there is no solution.
- Use brute force.

Running time: $2^{2^{O(k)}} \cdot n^{O(1)}$ — double exponential dependence on k!

Edge Clique Cover

Edge Clique Cover: Given a graph G and an integer k, cover the edges of G with at most k cliques.

(the cliques need not be edge disjoint)

Double-exponential dependence on k cannot be avoided!

Theorem

Assuming ETH, there is no $2^{2^{o(k)}} \cdot n^{O(1)}$ time algorithm for Edge Clique Cover.

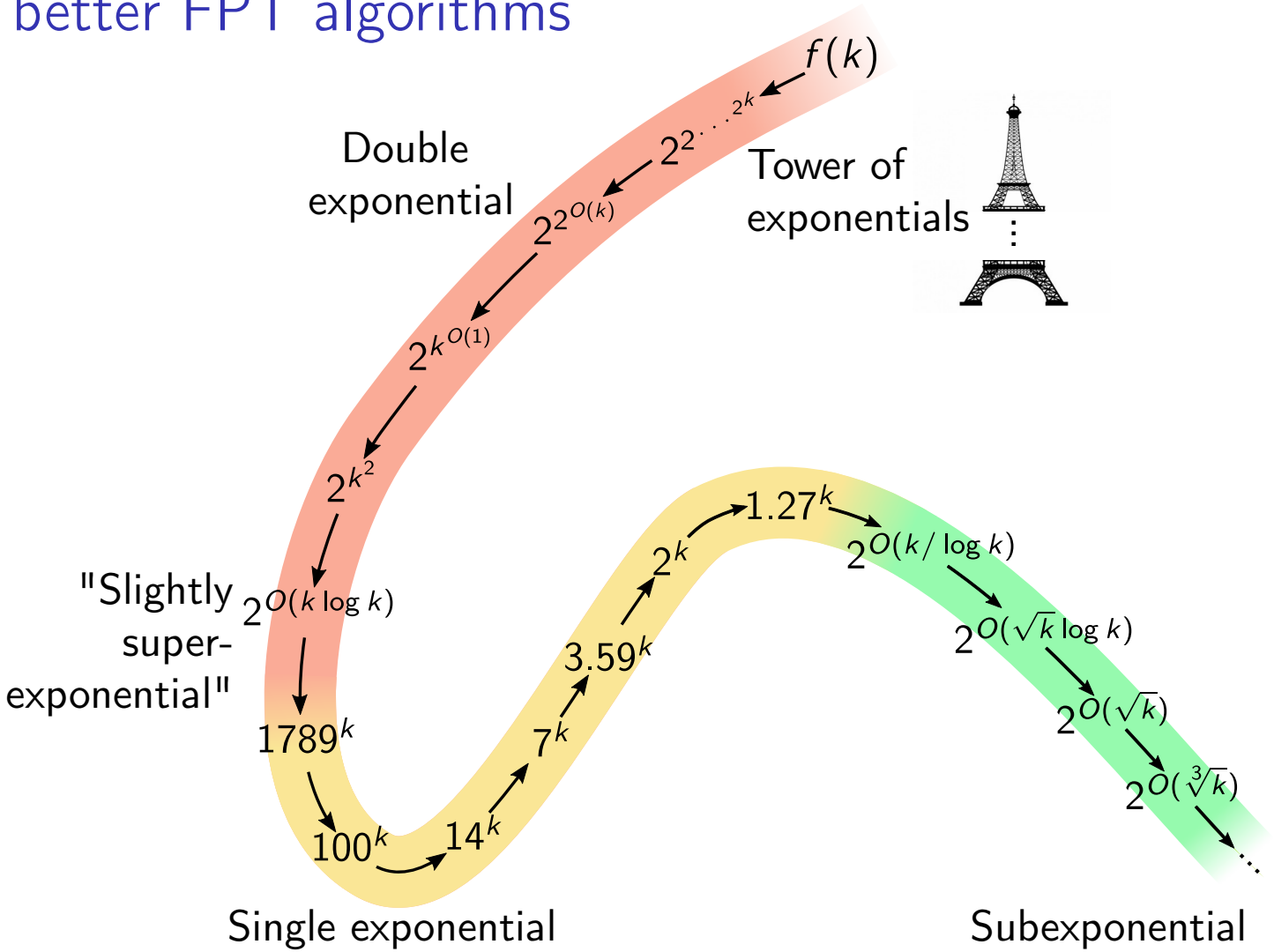
Proof:

3SAT
n variables

$\Rightarrow$

Edge Clique Cover
$k = O(\log n)$

The race for better FPT algorithms



Slightly superexponential algorithms

Running time of the form $2^{O(k \log k)} \cdot n^{O(1)}$ appear naturally in parameterized algorithms usually because of one of two reasons:

1. Branching into k directions at most k times explores a search tree of size $k^k = 2^{O(k \log k)}$.

   Example: Feedback Vertex Set in the first lecture.
2. Trying $k! = 2^{O(k \log k)}$ permutations of k elements (or partitions, matchings, ...)

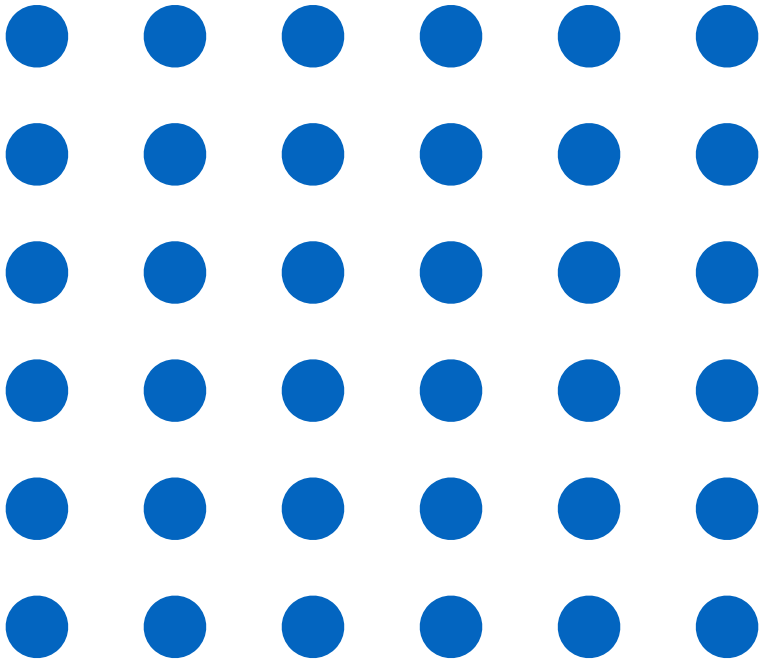
Can we avoid these steps and obtain $2^{O(k)} \cdot n^{O(1)}$ time algorithms?

Input

A graph over the vertex set [k] x [k].

Is there a clique that picks one vertex from each row?

Question



[k]x[k] Clique

Unless ETH fails,
there is no algorithm that solves 3-Colorability in $2^{o(n)}$ time.

Unless ETH fails,
there is no algorithm that solves 3-Colorability in $2^{o(n)}$ time.

Unless ETH fails,
there is no algorithm that solves [k]x[k] Clique in $2^{o(k \log k)}$ time.

No $2^{o(k)}$ algorithm.

A $2^{(k \log k)}$ algorithm.

A $2^{(k \log k)}$ algorithm.

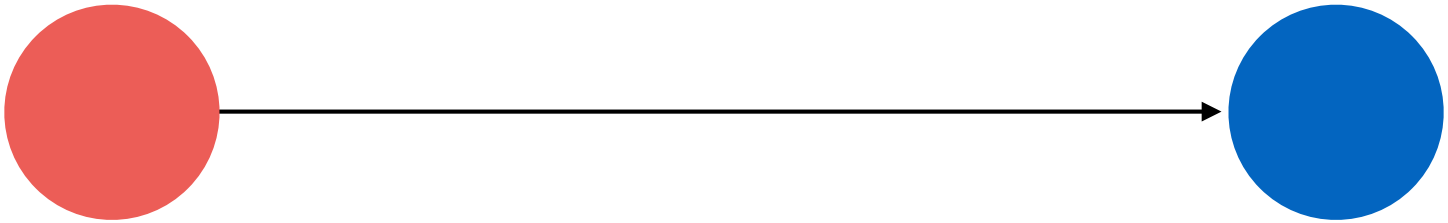
A $2^{(k \log k)}$ algorithm.

Unless ETH fails,
there is no algorithm that solves 3-Colorability in $2^{o(n)}$ time.

Unless ETH fails,
there is no algorithm that solves [k]x[k] Clique in $2^{o(k \log k)}$ time.

3-Colorability [N]

[k]x[k] Clique



Reduce 3-COL to [k]x[k] Clique, and suppose n —> k*

Run a $2^{o(k^* \log k^*)}$ algorithm.

This should be a $2^{o(n)}$ algorithm.

Reduce 3-SAT to Edge Clique Cover, and suppose n —> k*

Run a $2^{o(2^{k^*})}$ algorithm.

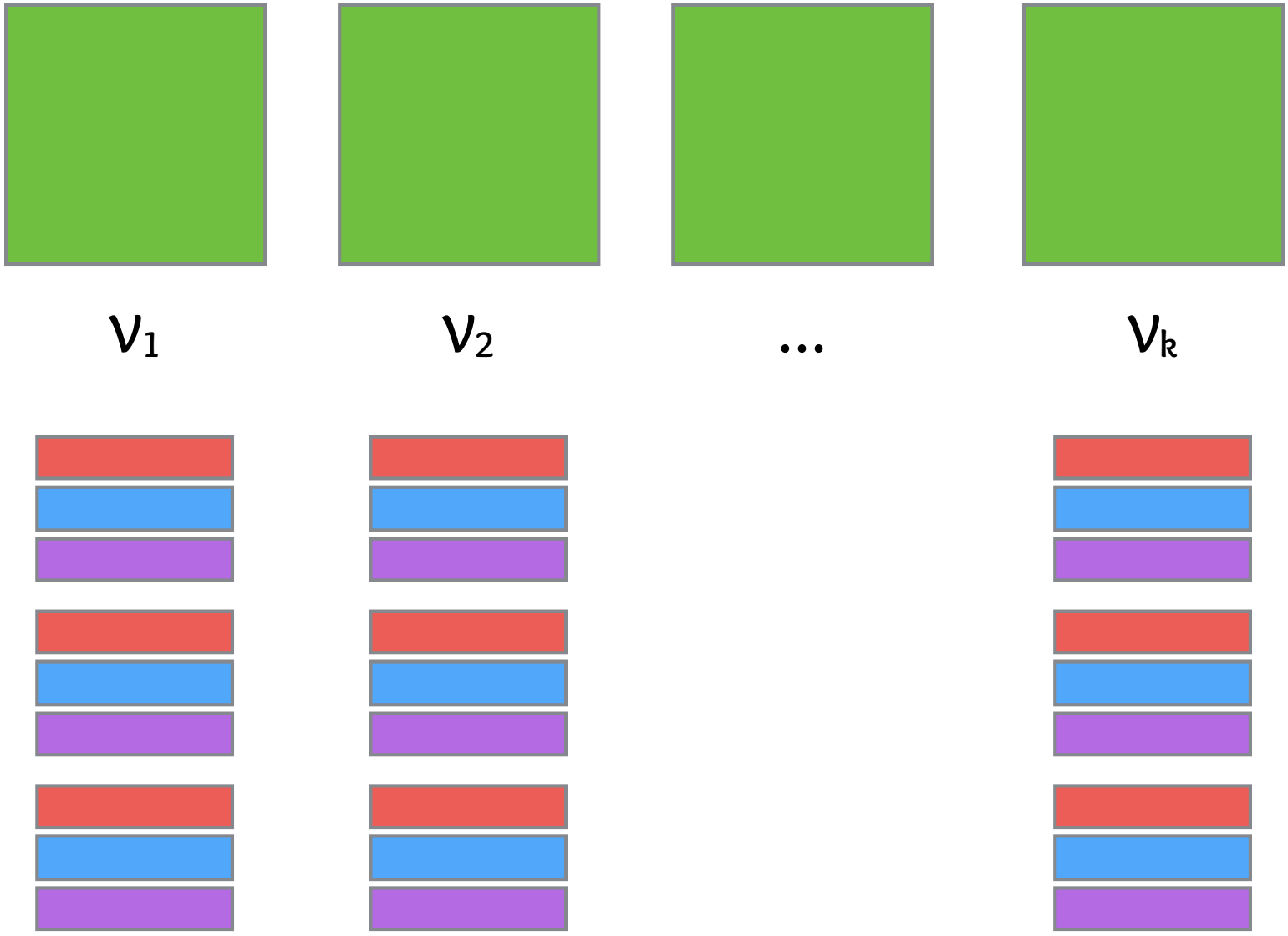
This should be a $2^{o(n)}$ algorithm.

3-Colorability for a graph with N vertices
reduces to [k]x[k] Clique with k = O(N/log N).

$$k = \left\lceil \frac{2N}{\log_3 N} \right\rceil$$

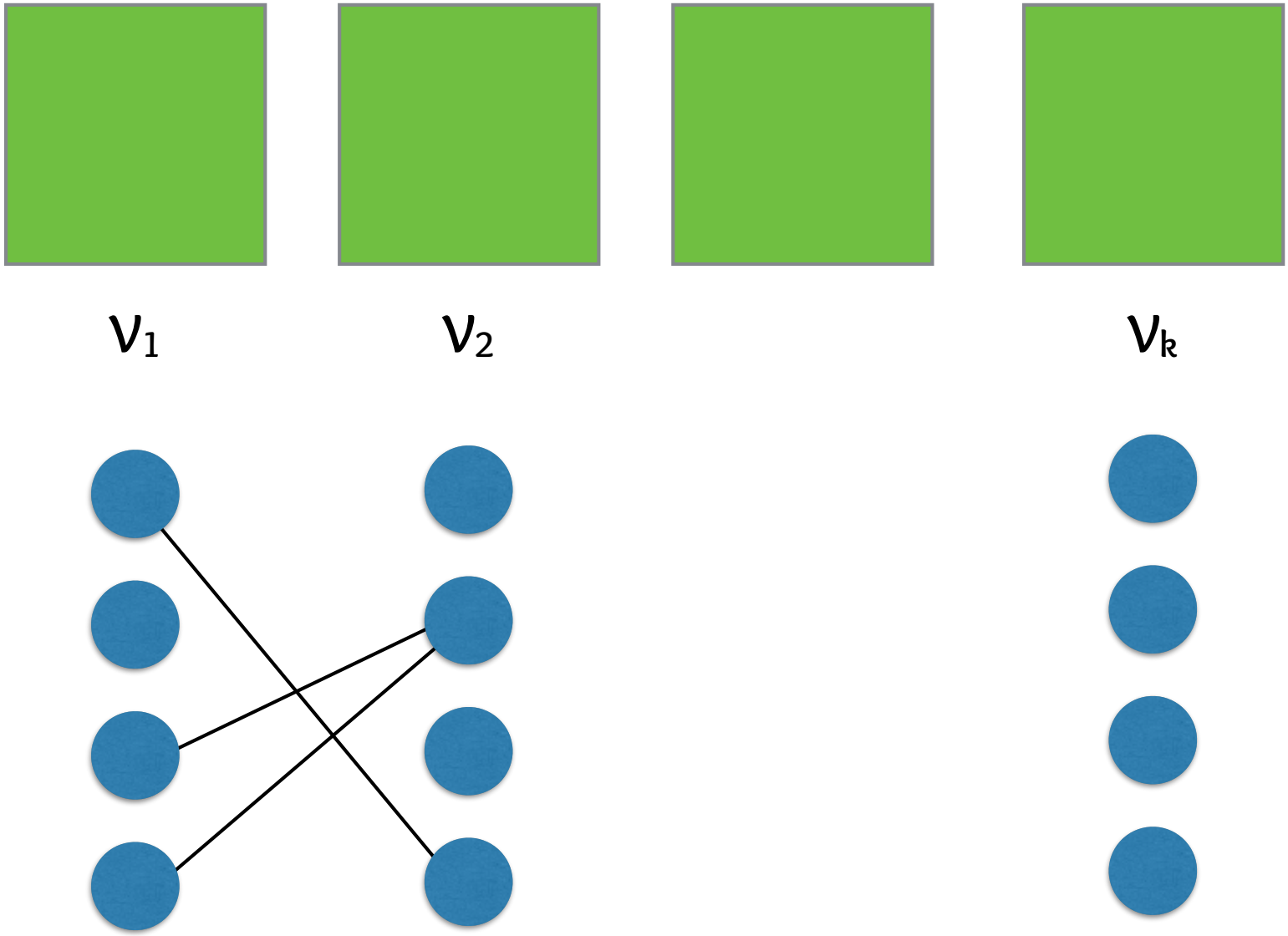
v_1 v_2 ... v_k

$$k = \left\lceil \frac{2N}{\log_3 N} \right\rceil$$



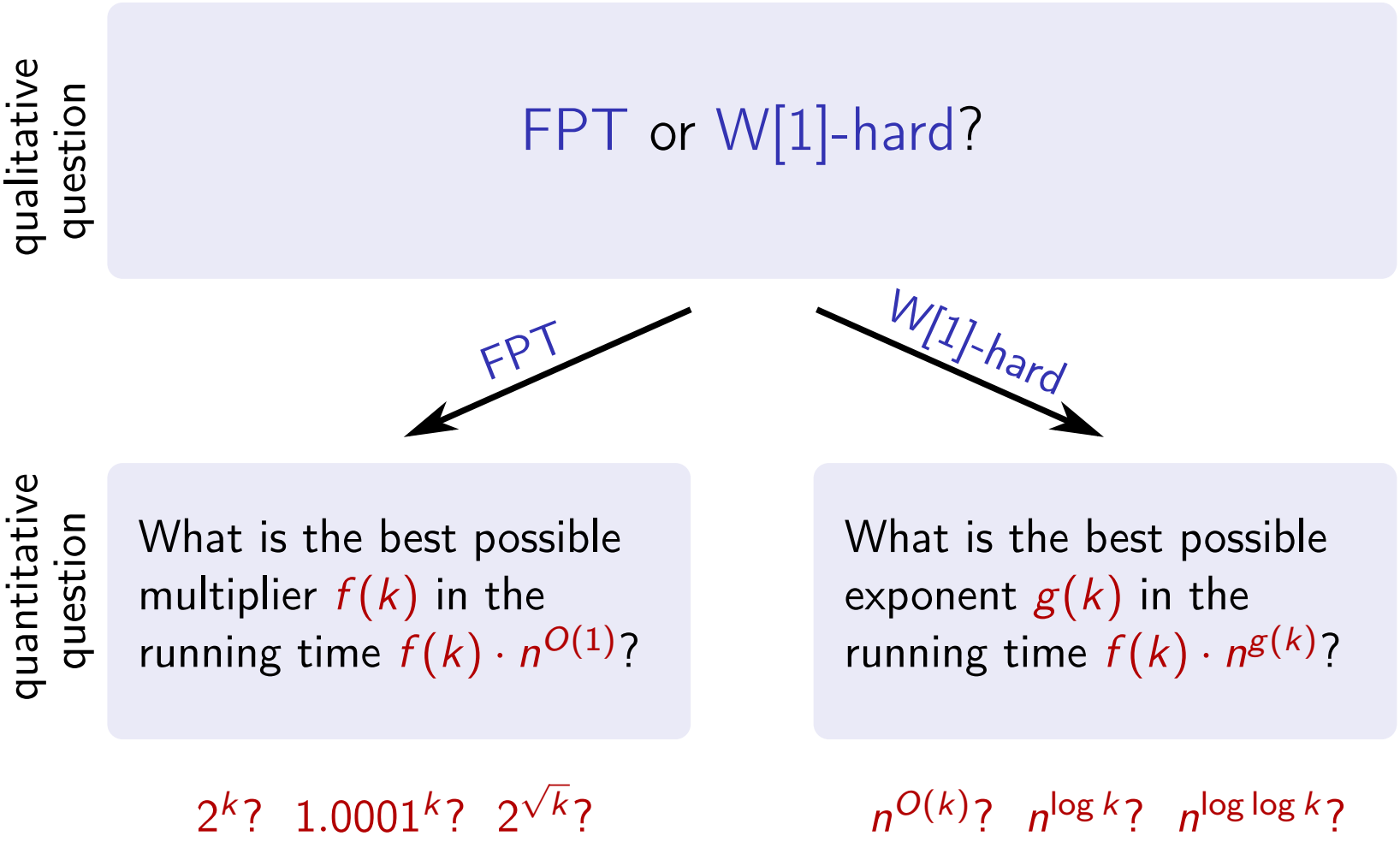
All possible 3-colorings of the V_i's.

$$k = \left\lceil \frac{2N}{\log_3 N} \right\rceil$$



Add edges between compatible colorings…

Shift of focus



Better algorithms for W[1]-hard problems

- $O(n^k)$ algorithm for k-Clique by brute force.
- $O(n^{0.79k})$ algorithms using fast matrix multiplication.
- W[1]-hardness of k-Clique gives evidence that there is no $f(k) \cdot n^{O(1)}$ time algorithm.
- But what about improvements of the exponent $O(k)$?

$n^{\sqrt{k}}$ $n^{k/\log\log k}$ $n^{\log k}$ $n^{\sqrt{k}}$ $2^{2^k} \cdot n^{\log\log\log k}$

Better algorithms for W[1]-hard problems

- $O(n^k)$ algorithm for k-Clique by brute force.
- $O(n^{0.79k})$ algorithms using fast matrix multiplication.
- W[1]-hardness of k-Clique gives evidence that there is no $f(k) \cdot n^{O(1)}$ time algorithm.
- But what about improvements of the exponent $O(k)$?

Theorem

Assuming ETH, k-Clique has no $f(k) \cdot n^{o(k)}$ algorithm for any computable function f.

In particular, ETH implies that k-Clique is not FPT.

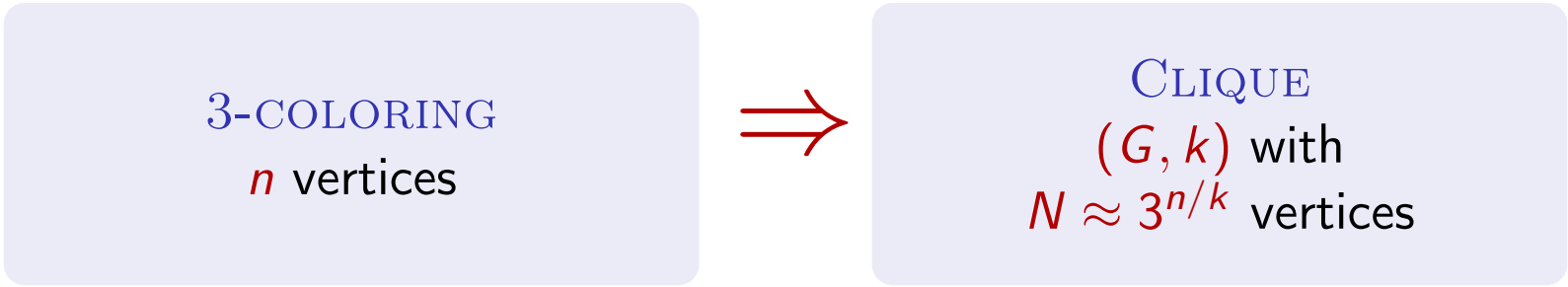
Lower bound for k-Clique

Theorem

Assuming ETH, k-Clique has no $f(k) \cdot N^{o(k)}$ algorithm for any computable function f.

Proof:

Textbook reduction from 3SAT to 3-Coloring shows that, assuming ETH, there is no $2^{o(n)}$ time algorithm for 3-Coloring on an n-vertex graph. Then



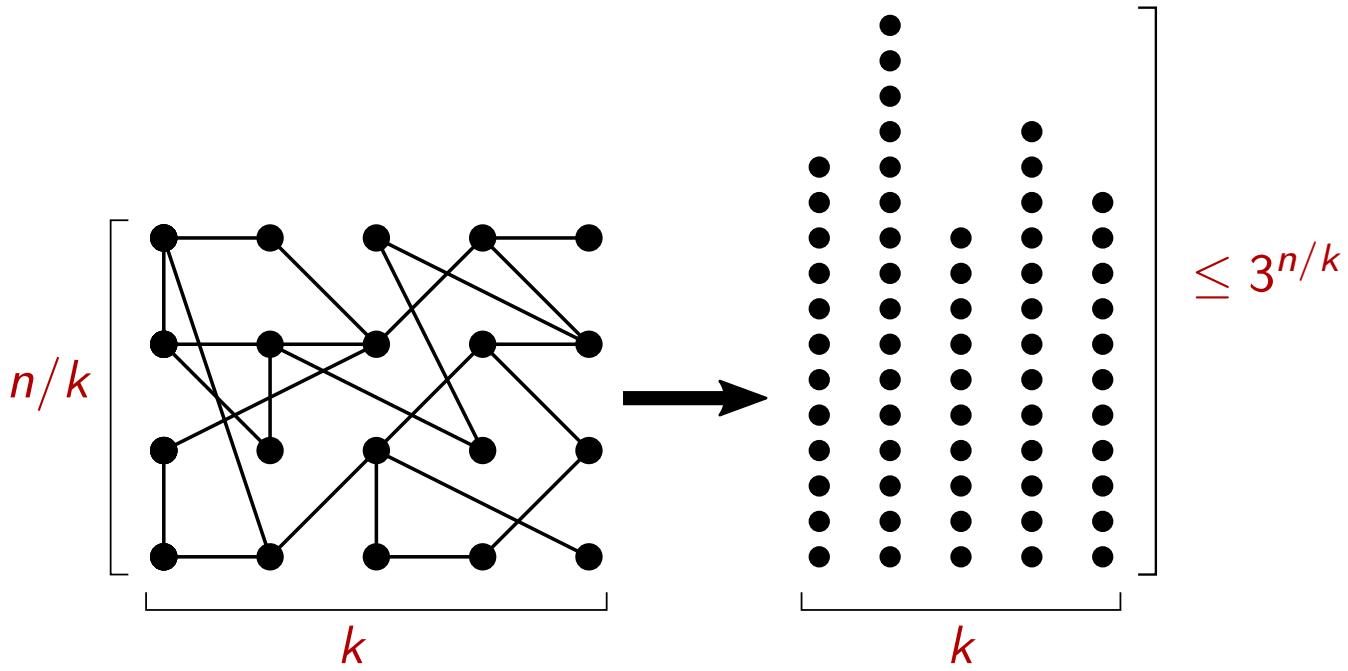
$N^{o(k)}$ algorithm for Clique $\Rightarrow (3^{n/k})^{o(k)} = 3^{o(n)}$ algorithm for 3-coloring

Lower bound for k-Clique

Theorem

Assuming ETH, k-Clique has no $f(k) \cdot N^{o(k)}$ algorithm for any computable function f.

Proof:



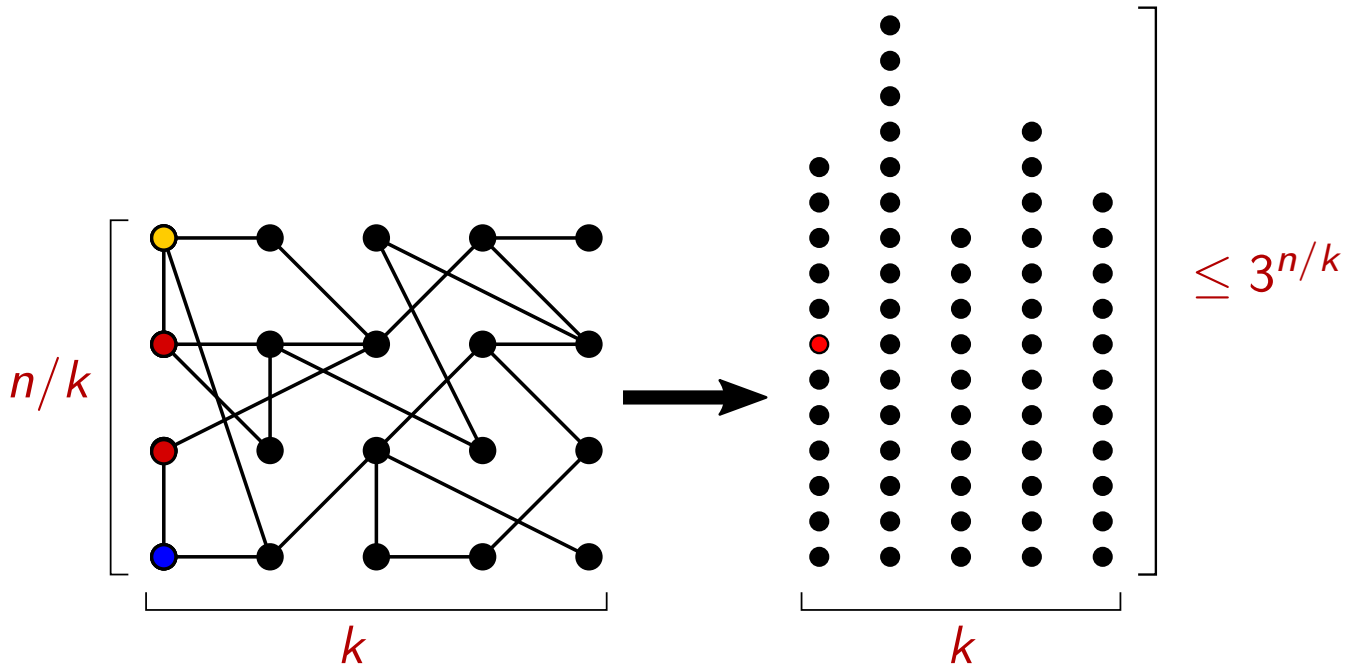
Create a vertex per each consistent coloring of each group.

Lower bound for k-Clique

Theorem

Assuming ETH, k-Clique has no $f(k) \cdot N^{o(k)}$ algorithm for any computable function f.

Proof:



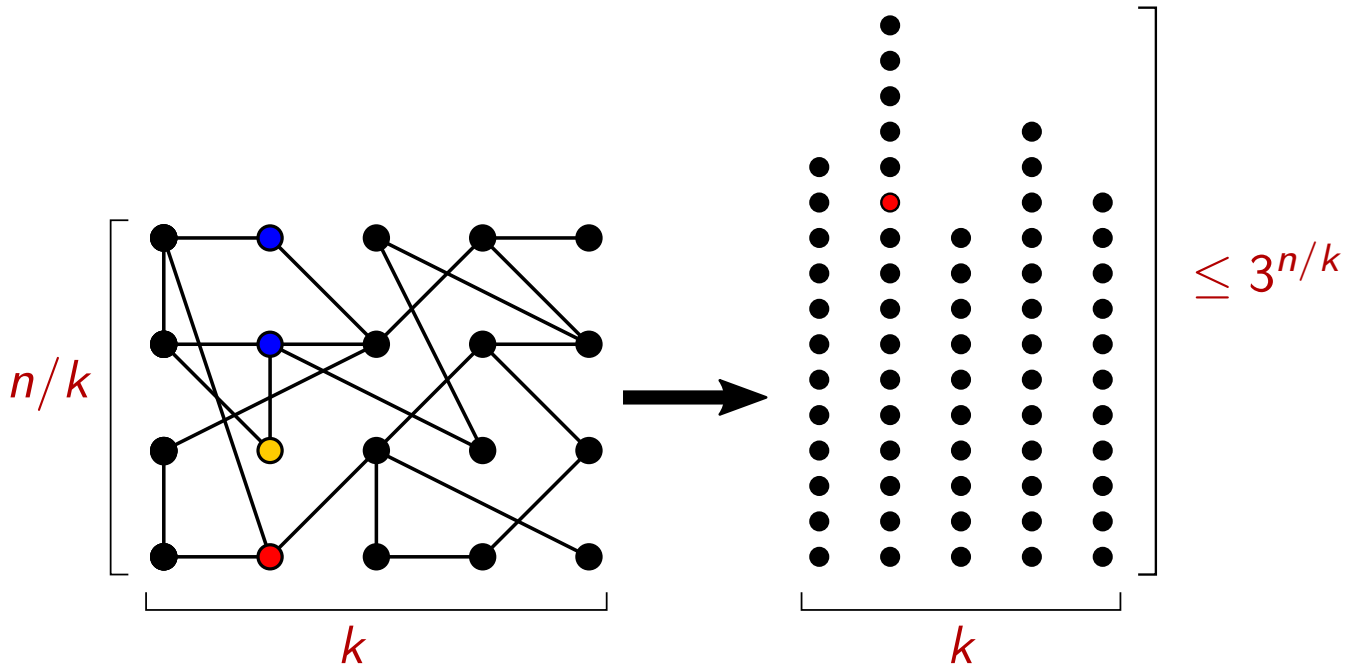
Create a vertex per each consistent coloring of each group.

Lower bound for k-Clique

Theorem

Assuming ETH, k-Clique has no $f(k) \cdot N^{o(k)}$ algorithm for any computable function f.

Proof:



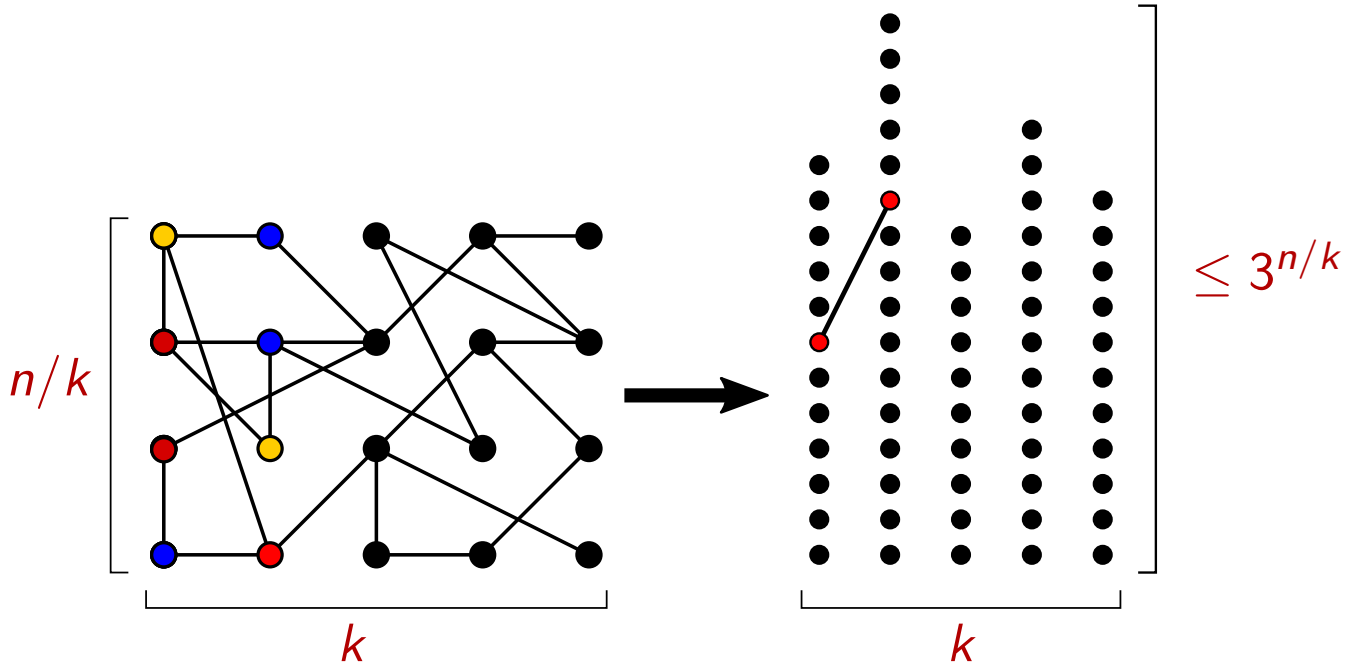
Create a vertex per each consistent coloring of each group.

Lower bound for k-CLIQUE

Theorem

Assuming ETH, k-CLIQUE has no $f(k) \cdot N^{o(k)}$ algorithm for any computable function f.

Proof:



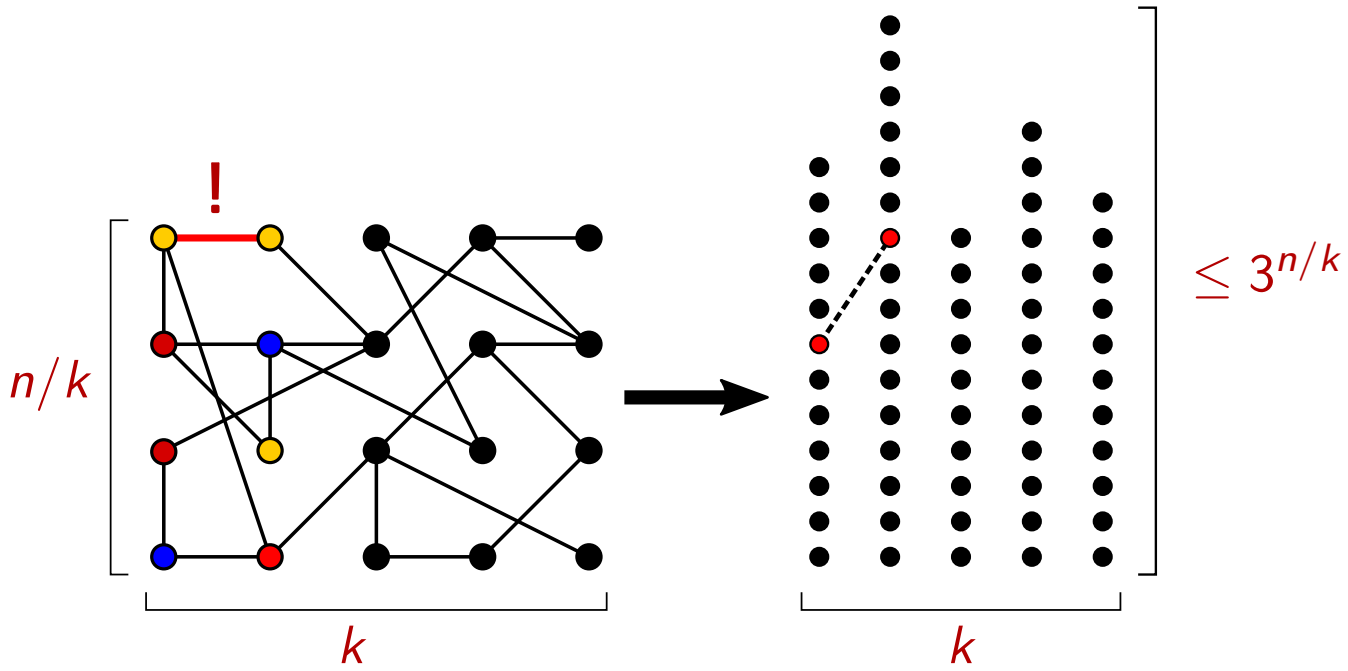
Connect two vertices if they represent colorings that are consistent together.

Lower bound for k-Clique

Theorem

Assuming ETH, k-Clique has no $f(k) \cdot N^{o(k)}$ algorithm for any computable function f.

Proof:



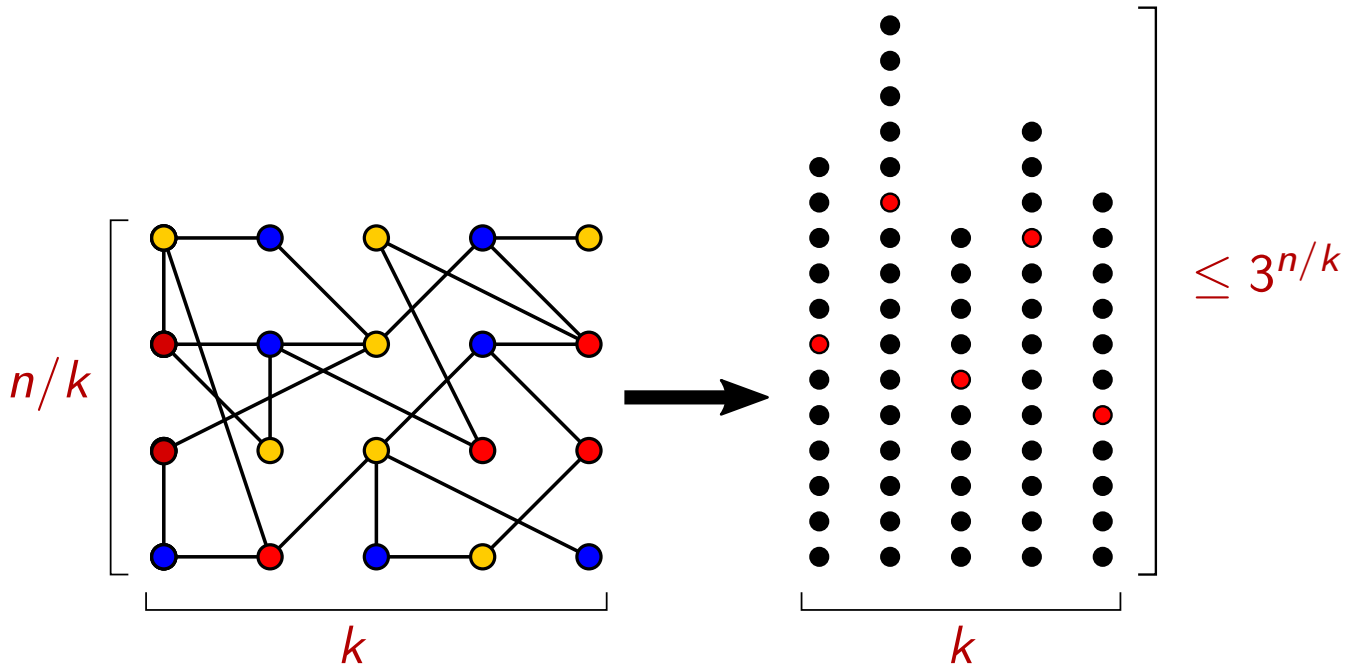
Connect two vertices if they represent colorings that are consistent together.

Lower bound for k-Clique

Theorem

Assuming ETH, k-Clique has no $f(k) \cdot N^{o(k)}$ algorithm for any computable function f.

Proof:



Left graph has a 3-coloring $\Leftrightarrow$ Right graph contains a k-clique

Lower bound for k-Clique

Theorem

Assuming ETH, k-Clique has no $f(k) \cdot N^{o(k)}$ algorithm for any computable function f.

Proof:

- We have constructed a new graph with $N = k \cdot 3^{n/k}$ vertices that has a k-clique if and only if the original graph is 3-colorable.
- Suppose that k-Clique has a $2^k \cdot N^{o(k)}$ time algorithm.
- Doing the reduction with $k := \log n$ gives us an algorithm for 3-Coloring with running time

$$2^k \cdot N^{o(k)} = n \cdot (\log n)^{o(\log n)} \cdot 3^{n \cdot o(\log n)/\log n} = 2^{o(n)}.$$

Lower bound for k-Clique

Theorem

Assuming ETH, k-Clique has no $f(k) \cdot N^{o(k)}$ algorithm for any computable function f.

Proof:

- We have constructed a new graph with $N = k \cdot 3^{n/k}$ vertices that has a k-clique if and only if the original graph is 3-colorable.
- Suppose that k-Clique has a $2^k \cdot N^{o(k)}$ time algorithm.
- Doing the reduction with $k := \log n$ gives us an algorithm for 3-Coloring with running time

$$2^k \cdot N^{o(k)} = n \cdot (\log n)^{o(\log n)} \cdot 3^{n \cdot o(\log n)/\log n} = 2^{o(n)}.$$

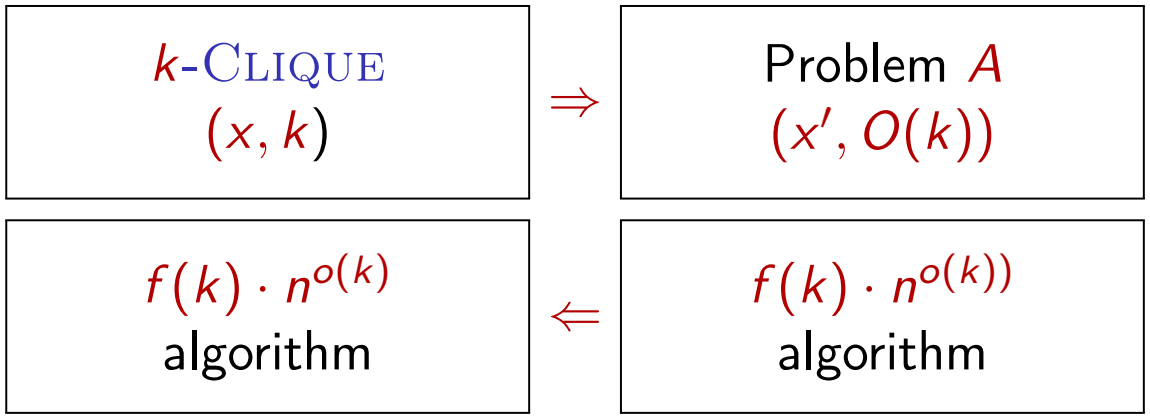
- Choosing $k := \log \log n$ would rule out a $2^{2^k} \cdot N^{o(k)}$ algorithm etc.
- In general, we need to choose roughly $k := f^{-1}(n)$ groups (technicalities omitted).

Tight bounds

Theorem

Assuming ETH, k-Clique has no $f(k) \cdot n^{o(k)}$ algorithm for any computable function f.

Transfering to other problems:

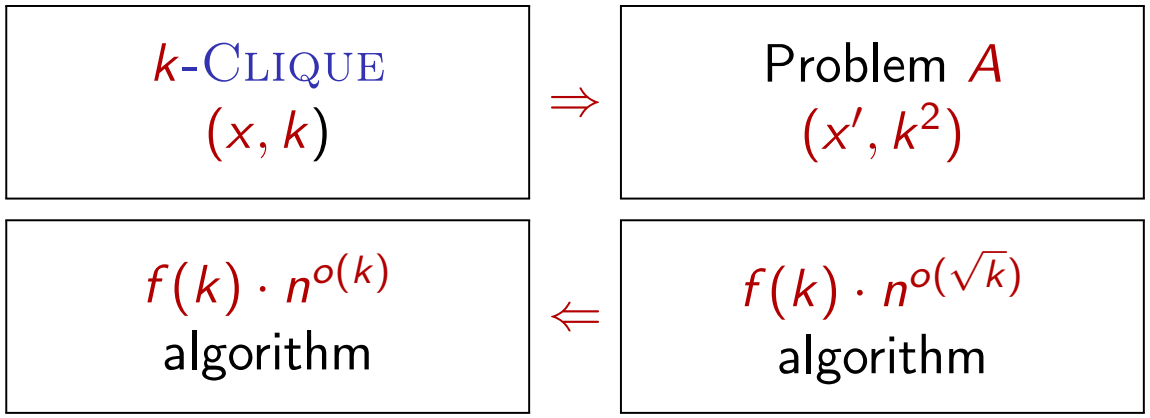


Tight bounds

Theorem

Assuming ETH, k-Clique has no $f(k) \cdot n^{o(k)}$ algorithm for any computable function f.

Transfering to other problems:

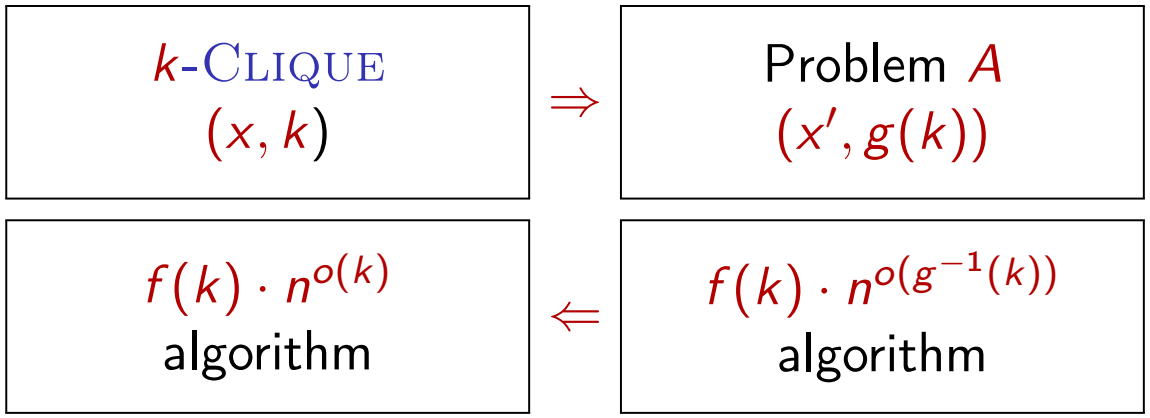


Tight bounds

Theorem

Assuming ETH, k-Clique has no $f(k) \cdot n^{o(k)}$ algorithm for any computable function f.

Transfering to other problems:

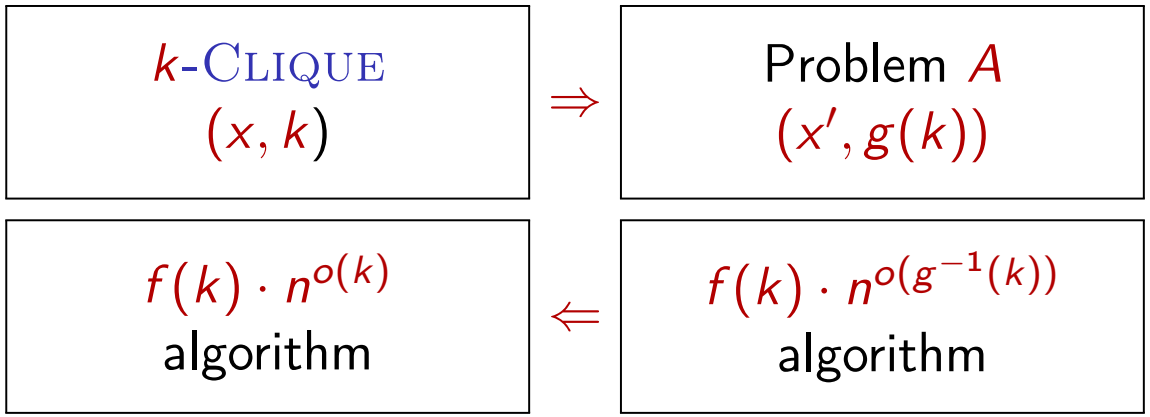


Tight bounds

Theorem

Assuming ETH, k-Clique has no $f(k) \cdot n^{o(k)}$ algorithm for any computable function f.

Transfering to other problems:



Bottom line:

- To rule out $f(k) \cdot n^{o(k)}$ algorithms, we need a parameterized reduction that blows up the parameter at most *linearly.*
- To rule out $f(k) \cdot n^{o(\sqrt{k})}$ algorithms, we need a parameterized reduction that blows up the parameter at most *quadratically.*

Tight bounds

Assuming ETH, there is no $f(k)n^{o(k)}$ time algorithms for

- Set Cover
- Hitting Set
- Connected Dominating Set
- Independent Dominating Set
- Partial Vertex Cover
- Dominating Set in bipartite graphs
- ...

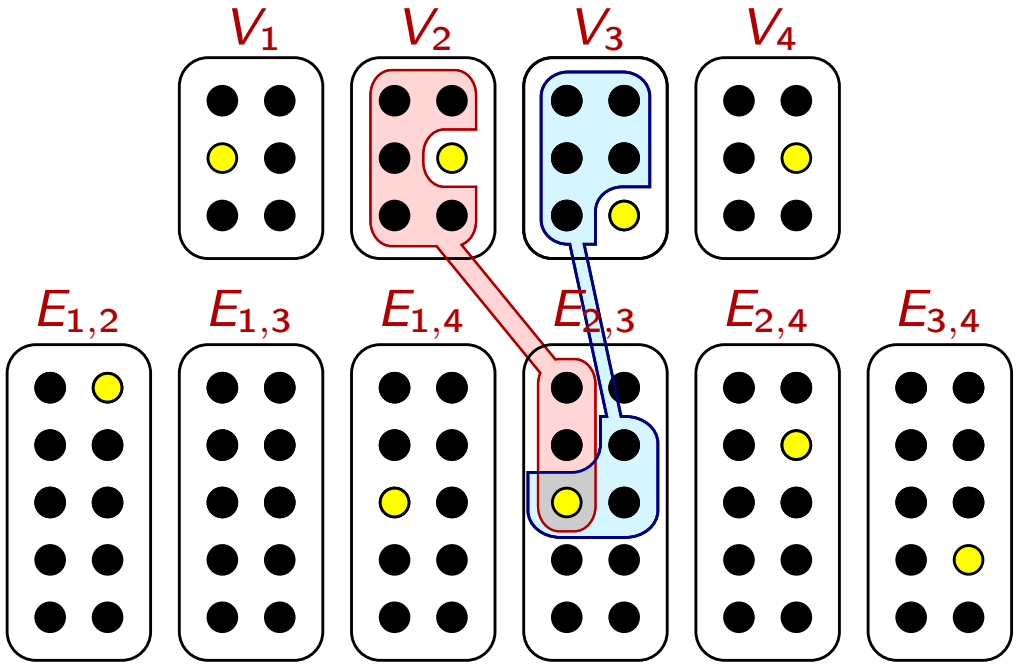
The odd case of Odd Set

Odd Set: Given a set system $\mathcal{F}$ over a universe U and an integer k, find a set S of at most k elements such that $|S \cap F|$ is odd for every $F \in \mathcal{F}$.

We have seen:

Theorem

Odd Set is W[1]-hard parameterized by k.



New parameter: $k' := k + \binom{k}{2} = O(k^2)$.

The odd case of Odd Set

Odd Set: Given a set system $\mathcal{F}$ over a universe U and an integer k, find a set S of at most k elements such that $|S \cap F|$ is odd for every $F \in \mathcal{F}$.

We have seen:

Theorem

Odd Set is W[1]-hard parameterized by k.

We immediately get:

Corollary

Assuming ETH, there is no $f(k)n^{o(\sqrt{k})}$ time algorithm for Odd Set.

But this does not seem to be tight...

Problem: k-Clique is a very densely constrained problem, which makes the reduction very expensive.

Subgraph Isomorphism

Subgraph Isomorphism: Given two graphs H and G, decide if H is isomorphic to a subgraph of G.

Trivial reduction from k-Clique:

Corollary (parameterized by no. of **vertices** of H)

Assuming ETH, Subgraph Isomorphism parameterized by $k := |V(H)|$ has no $f(k)n^{o(k)}$ time algorithm.

Subgraph Isomorphism

Subgraph Isomorphism: Given two graphs H and G, decide if H is isomorphic to a subgraph of G.

Trivial reduction from k-Clique:

Corollary (parameterized by no. of **edges** of H)

Assuming ETH, Subgraph Isomorphism parameterized by $k := |E(H)|$ has no $f(k)n^{o(\sqrt{k})}$ time algorithm.

Is this tight?

Subgraph Isomorphism

Subgraph Isomorphism: Given two graphs H and G, decide if H is isomorphic to a subgraph of G.

Trivial reduction from k-Clique:

Corollary (parameterized by no. of **edges** of H)

Assuming ETH, Subgraph Isomorphism parameterized by $k := |E(H)|$ has no $f(k)n^{o(\sqrt{k})}$ time algorithm.

Is this tight?

An almost tight result:

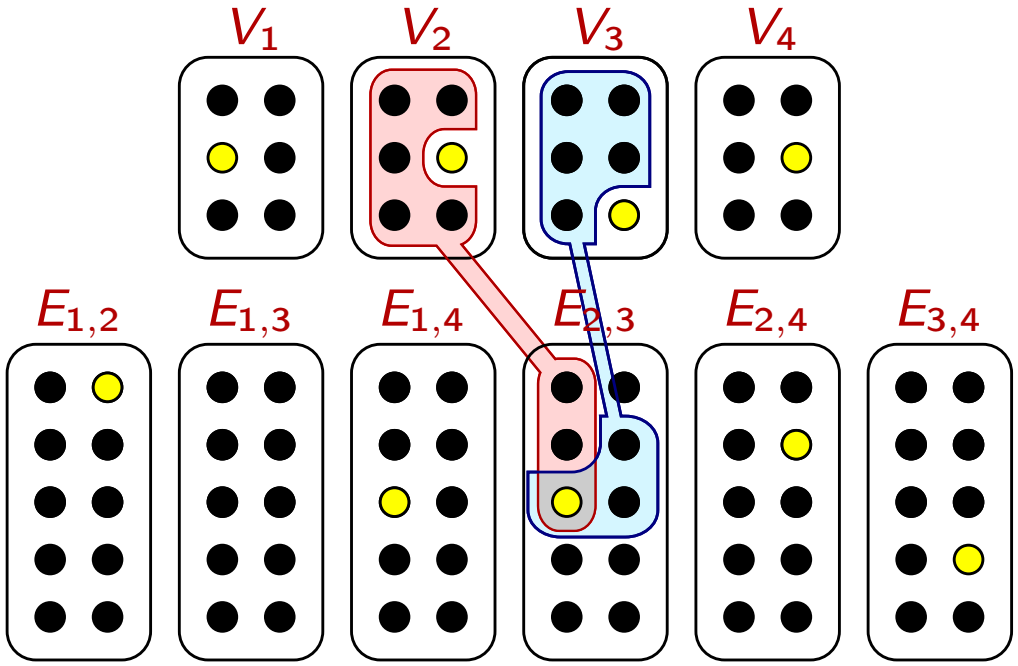
Theorem [M. 2010]

Assuming ETH, Subgraph Isomorphism parameterized by $k := |E(H)|$ has no $f(k)n^{o(k/\log k)}$ time algorithm.

Open question: can we remove the $\log k$ from this lower bound?

Odd Set

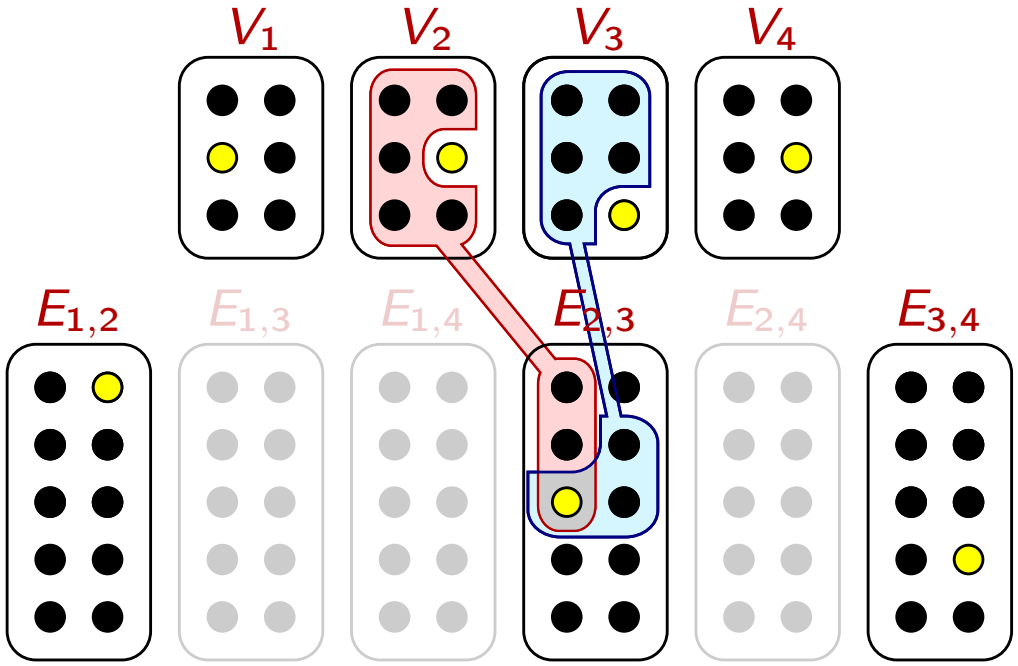
Reduction from k-Clique to Odd Set:



New parameter: $k' := k + \binom{k}{2} = O(k^2)$.

Odd Set

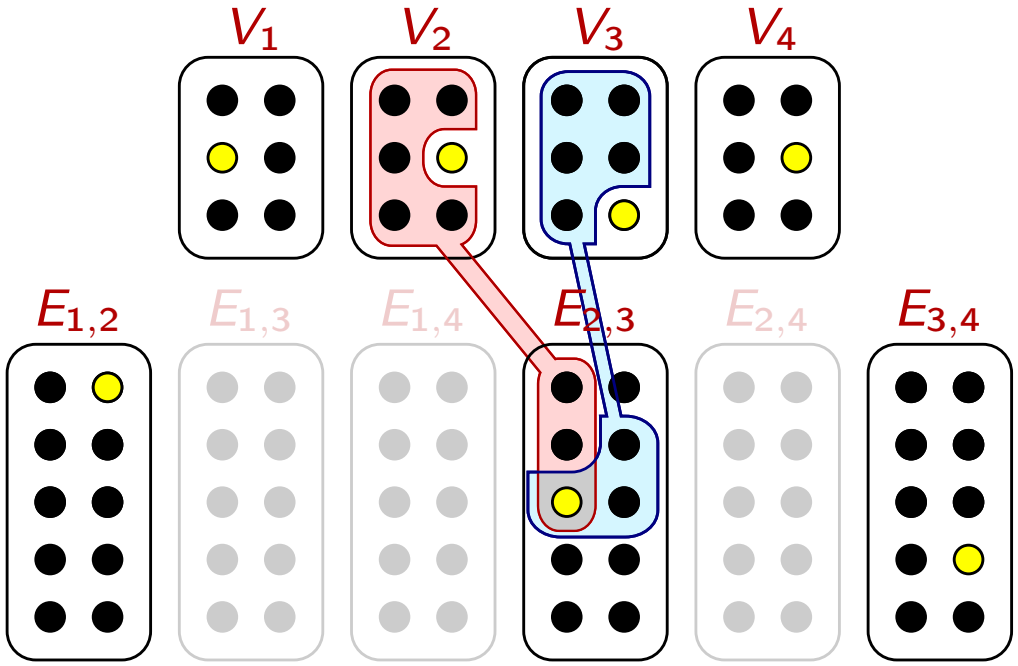
Reduction from Subgraph Isomorphism to Odd Set:



New parameter: $k' := |V(H)| + |E(H)| = O(k)$.
(where $k := |E(H)|$)

Odd Set

Reduction from Subgraph Isomorphism to Odd Set:



New parameter: $k' := |V(H)| + |E(H)| = O(k)$.
(where $k := |E(H)|$)

Theorem

Assuming ETH, there is no $f(k)n^{o(k/\log k)}$ time algorithm for Odd Set.

Tight bounds

Assuming ETH, there is no $f(k)n^{o(k)}$ time algorithms for

- Set Cover
- Hitting Set
- Connected Dominating Set
- Independent Dominating Set
- Partial Vertex Cover
- Dominating Set in bipartite graphs
- ...

Tight bounds

Assuming ETH, there is no $f(k)n^{o(k)}$ time algorithms for

- Set Cover
- Hitting Set
- Connected Dominating Set
- Independent Dominating Set
- Partial Vertex Cover
- Dominating Set in bipartite graphs
- ...

What about planar problems?

- More problems are FPT, more difficult to prove W[1]-hardness.
- The problem Grid Tiling is the key to many of these results.

Grid Tiling

Grid Tiling

Input: A $k \times k$ matrix and a set of pairs $S_{i,j} \subseteq [D] \times [D]$ for each cell.

Find: A pair $s_{i,j} \in S_{i,j}$ for each cell such that

- Vertical neighbors agree in the 1st coordinate.
- Horizontal neighbors agree in the 2nd coordinate.

(1,1) (3,1) (2,4)	(5,1) (1,4) (5,3)	(1,1) (2,4) (3,3)
(2,2) (1,4)	(3,1) (1,2)	(2,2) (2,3)
(1,3) (2,3) (3,3)	(1,1) (1,3)	(2,3) (5,3)

$k = 3$, $D = 5$

Grid Tiling

Grid Tiling

Input: A $k \times k$ matrix and a set of pairs $S_{i,j} \subseteq [D] \times [D]$ for each cell.

Find: A pair $s_{i,j} \in S_{i,j}$ for each cell such that

- Vertical neighbors agree in the 1st coordinate.
- Horizontal neighbors agree in the 2nd coordinate.

(1,1) (3,1) (2,4)	(5,1) (1,4) (5,3)	(1,1) (2,4) (3,3)
(2,2) (1,4)	(3,1) (1,2)	(2,2) (2,3)
(1,3) (2,3) (3,3)	(1,1) (1,3)	(2,3) (5,3)

$k = 3$, $D = 5$

Grid Tiling

Grid Tiling

Input: A $k \times k$ matrix and a set of pairs $S_{i,j} \subseteq [D] \times [D]$ for each cell.

Find: A pair $s_{i,j} \in S_{i,j}$ for each cell such that

- Vertical neighbors agree in the 1st coordinate.
- Horizontal neighbors agree in the 2nd coordinate.

Simple proof:

Fact

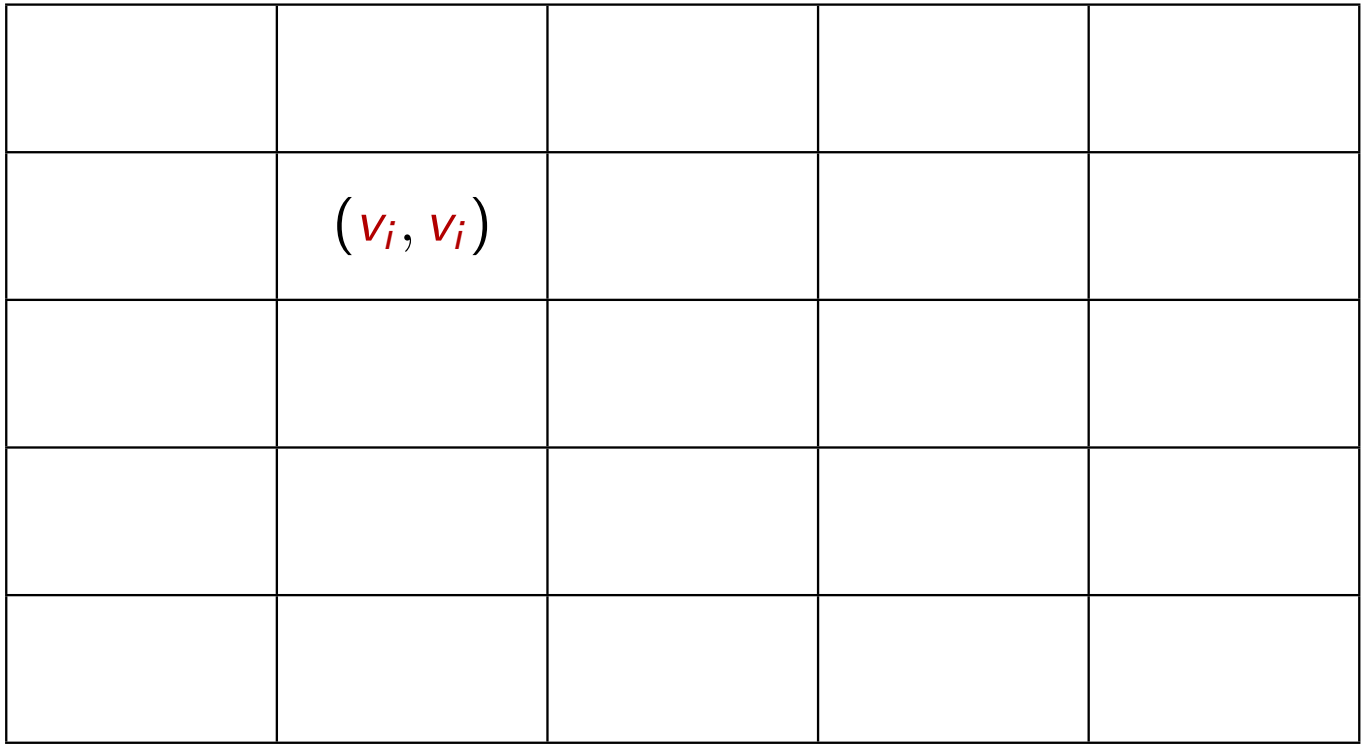
There is a parameterized reduction from k-Clique to $k \times k$ Grid Tiling.

Grid Tiling is W[1]-hard

Reduction from k-Clique

Definition of the sets:

- For $i = j$: $(x, y) \in S_{i,j} \iff x = y$
- For $i \neq j$: $(x, y) \in S_{i,j} \iff x$ and y are adjacent.



Each diagonal cell defines a value $v_i \ldots$

Grid Tiling is W[1]-hard

Reduction from k-Clique

Definition of the sets:

- For $i = j$: $(x, y) \in S_{i,j} \iff x = y$
- For $i \neq j$: $(x, y) \in S_{i,j} \iff x$ and y are adjacent.

	$(v_i, .)$			
$(., v_i)$	(v_i, v_i)	$(., v_i)$	$(., v_i)$	$(., v_i)$
	$(v_i, .)$			
	$(v_i .,)$			
	$(v_i, .)$			

. . . which appears on a "cross"

Grid Tiling is W[1]-hard

Reduction from k-Clique

Definition of the sets:

- For $i = j$: $(x, y) \in S_{i,j} \iff x = y$
- For $i \neq j$: $(x, y) \in S_{i,j} \iff x$ and y are adjacent.

	$(v_i, .)$			
$(., v_i)$	(v_i, v_i)	$(., v_i)$	$(., v_i)$	$(., v_i)$
	$(v_i, .)$			
	$(v_i, .)$		(v_j, v_j)	
	$(v_i, .)$			

v_i and v_j are adjacent for every $1 \leq i < j \leq k$.

Grid Tiling is W[1]-hard

Reduction from k-Clique

Definition of the sets:

- For $i = j$: $(x, y) \in S_{i,j} \iff x = y$
- For $i \neq j$: $(x, y) \in S_{i,j} \iff x$ and y are adjacent.

	$(v_i, .)$		$(v_j, .)$	
$(., v_i)$	(v_i, v_i)	$(., v_i)$	(v_j, v_i)	$(., v_i)$
	$(v_i, .)$		$(v_j, .)$	
$(., v_j)$	(v_i, v_j)	$(., v_j)$	(v_j, v_j)	$(., v_j)$
	$(v_i, .)$		$(v_j, .)$	

v_i and v_j are adjacent for every $1 \leq i < j \leq k$.

Grid Tiling and planar problems

Theorem

$k \times k$ Grid Tiling is W[1]-hard and, assuming ETH, cannot be solved in time $f(k)n^{o(k)}$ for any function f.

This lower bound is the key for proving hardness results for planar graphs.

Examples:

- Multiway Cut on planar graphs with k terminals
- Independent Set for unit disks

Grid Tiling with $\leq$

Grid Tiling with $\leq$

Input: A $k \times k$ matrix and a set of pairs $S_{i,j} \subseteq [D] \times [D]$ for each cell.

Find: A pair $s_{i,j} \in S_{i,j}$ for each cell such that

- 1st coordinate of $s_{i,j} \leq$ 1st coordinate of $s_{i+1,j}$.
- 2nd coordinate of $s_{i,j} \leq$ 2nd coordinate of $s_{i,j+1}$.

(5,1) (1,2) (3,3)	(4,3) (3,2)	(2,3) (2,5)
(2,1) (5,5) (3,5)	(4,2) (5,3)	(5,1) (3,2)
(5,1) (2,2) (5,3)	(2,1) (4,2)	(3,1) (3,2) (3,3)

$k = 3$, $D = 5$

Grid Tiling with $\leq$

Grid Tiling with $\leq$

Input: A $k \times k$ matrix and a set of pairs $S_{i,j} \subseteq [D] \times [D]$ for each cell.

Find: A pair $s_{i,j} \in S_{i,j}$ for each cell such that

- 1st coordinate of $s_{i,j} \leq$ 1st coordinate of $s_{i+1,j}$.
- 2nd coordinate of $s_{i,j} \leq$ 2nd coordinate of $s_{i,j+1}$.

Variant of the previous proof:

Theorem

There is a parameterized reduction from $k \times k$-Grid Tiling to $O(k) \times O(k)$ Grid Tiling with $\leq$.

Very useful starting point for geometric problems!

k-Independent Set for unit disks

Theorem

Given a set of n unit disks in the plane, we can find k independent disks in time $n^{O(\sqrt{k})}$.

k-Independent Set for unit disks

Theorem

Given a set of n unit disks in the plane, we can find k independent disks in time $n^{O(\sqrt{k})}$.

Matching lower bound:

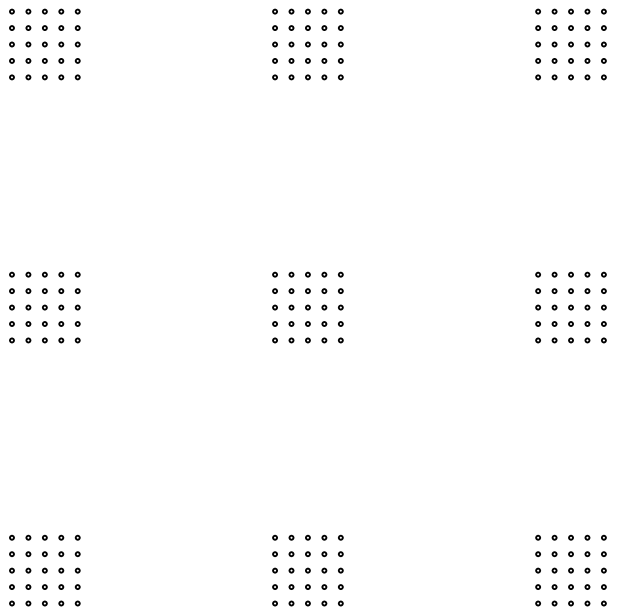
Theorem

There is a reduction from $k \times k$ Grid Tiling with $\leq$ to k^2-Independent Set for unit disks. Consequently, Independent Set for unit disks is

- is W[1]-hard, and
- cannot be solved in time $f(k)n^{o(\sqrt{k})}$ for any function f.

Reduction to unit disks

(5,1) (1,2) (3,3)	(4,3) (3,2)	(2,3) (2,5)
(2,1) (5,5) (3,5)	(4,2) (5,3)	(5,1) (3,2)
(5,1) (2,2) (5,3)	(2,1) (4,2)	(3,1) (3,2) (3,3)

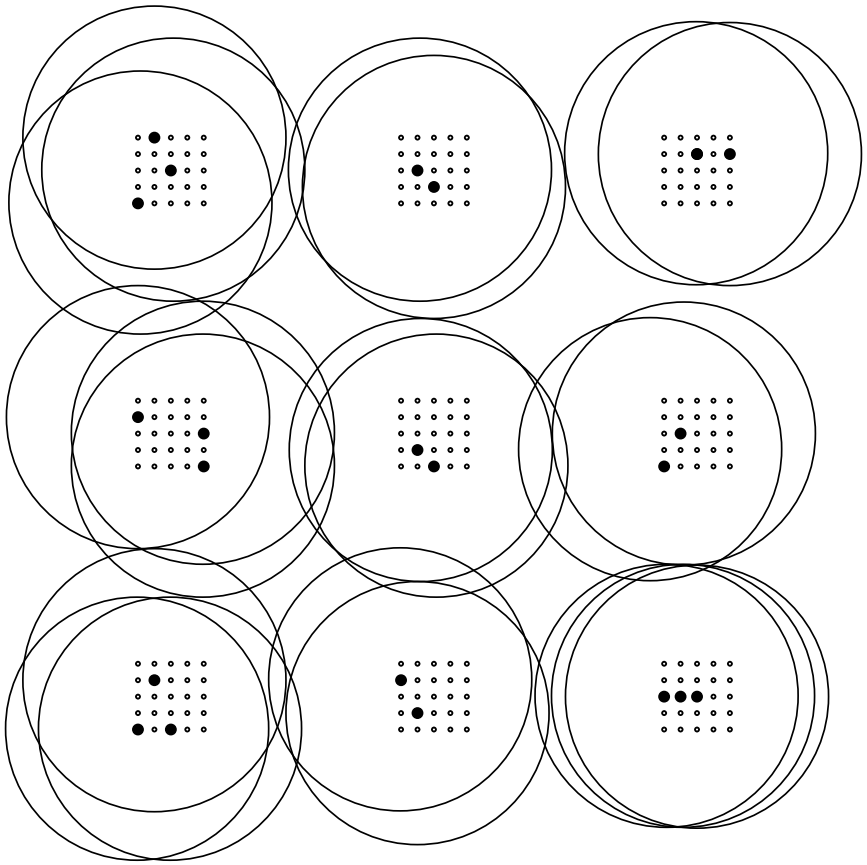


Every pair is represented by a unit disk in the plane.

$\leq$ relation between coordinates $\iff$ disks do not intersect.

Reduction to unit disks

(5,1) (1,2) (3,3)	(4,3) (3,2)	(2,3) (2,5)
(2,1) (5,5) (3,5)	(4,2) (5,3)	(5,1) (3,2)
(5,1) (2,2) (5,3)	(2,1) (4,2)	(3,1) (3,2) (3,3)

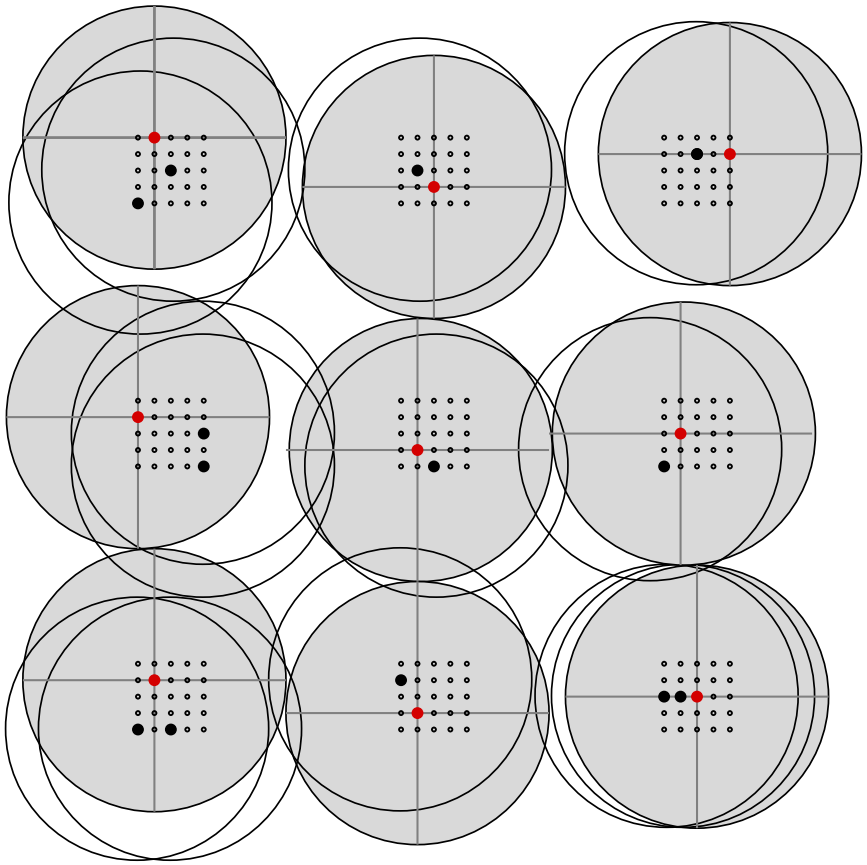


Every pair is represented by a unit disk in the plane.

$\leq$ relation between coordinates $\iff$ disks do not intersect.

Reduction to unit disks

(5,1) (1,2) (3,3)	(4,3) (3,2)	(2,3) (2,5)
(2,1) (5,5) (3,5)	(4,2) (5,3)	(5,1) (3,2)
(5,1) (2,2) (5,3)	(2,1) (4,2)	(3,1) (3,2) (3,3)



Every pair is represented by a unit disk in the plane.

$\leq$ relation between coordinates $\iff$ disks do not intersect.

Summary

We used ETH to rule out

1. $2^{o(n)}$ time algorithms for, say, Independent Set.
2. $2^{o(\sqrt{n})}$ time algorithms for, say, Independent Set on planar graphs.
3. $2^{o(k)} \cdot n^{O(1)}$ time algorithms for, say, Vertex Cover.
4. $2^{o(\sqrt{k})} \cdot n^{O(1)}$ time algorithms for, say, Vertex Cover on planar graphs.
5. $f(k)n^{o(k)}$ time algorithms for Clique.
6. $f(k)n^{o(\sqrt{k})}$ time algorithms for planar problems such as k-Terminal Cut and Independent Set for unit disks.

Other tight lower bounds on $f(k)$ having the form $2^{o(k \log k)}$, $2^{o(k^2)}$, or $2^{2^{o(k)}}$ exist.