

# Lower bounds for parameterized problems

Dániel Marx<sup>1</sup>

<sup>1</sup>Institute for Computer Science and Control,  
Hungarian Academy of Sciences (MTA SZTAKI)  
Budapest, Hungary

ICERM  
Providence, RI  
April 26, 2014

## Lower bounds

So far we have seen positive results: algorithms.

What kind of negative results we have?

- Can we show that a problem (e.g., **CLIQUE**) is **not** FPT?
- Can we show that a problem (e.g., **VERTEX COVER**) has **no** algorithm with running time, say,  $2^{o(k)} \cdot n^{O(1)}$ ?

## Lower bounds

So far we have seen positive results: algorithms.

What kind of negative results we have?

- Can we show that a problem (e.g., **CLIQUE**) is **not** FPT?
- Can we show that a problem (e.g., **VERTEX COVER**) has **no** algorithm with running time, say,  $2^{o(k)} \cdot n^{O(1)}$ ?

This would require showing that  $P \neq NP$ : if  $P = NP$ , then, e.g.,  $k$ -**CLIQUE** is polynomial-time solvable, hence FPT.

Can we give some evidence for negative results?

## Classical complexity

**Nondeterministic Turing Machine (NTM):** single tape, finite alphabet, finite state, head can move left/right only one cell. In each step, the machine can branch into an arbitrary number of directions. Run is successful if at least one branch is successful.

**NP:** The class of all languages that can be recognized by a polynomial-time NTM.

**Polynomial-time reduction** from problem  $P$  to problem  $Q$ : a function  $\phi$  with the following properties:

- $\phi(x)$  can be computed in time  $|x|^{O(1)}$ ,
- $\phi(x)$  is a yes-instance of  $Q$  if and only if  $x$  is a yes-instance of  $P$ .

**Definition:** Problem  $Q$  is **NP-hard** if any problem in **NP** can be reduced to  $Q$ .

If an **NP-hard** problem can be solved in polynomial time, then every problem in **NP** can be solved in polynomial time (i.e.,  $P = NP$ ).

Part I:  
Reductions and  
the W-hierarchy

## Parameterized complexity

To build a complexity theory for parameterized problems, we need two things:

- An appropriate notion of reduction.
- An appropriate hypothesis.

Polynomial-time reductions are not good for our purposes.

## Parameterized complexity

To build a complexity theory for parameterized problems, we need two things:

- An appropriate notion of reduction.
- An appropriate hypothesis.

Polynomial-time reductions are not good for our purposes.

**Example:** Graph  $G$  has an independent set  $k$  if and only if it has a vertex cover of size  $n - k$ .

⇒ Transforming an **INDEPENDENT SET** instance  $(G, k)$  into a **VERTEX COVER** instance  $(G, n - k)$  is a correct polynomial-time reduction.

However, **VERTEX COVER** is FPT, but **INDEPENDENT SET** is not known to be FPT.

# Parameterized reduction

## Definition

**Parameterized reduction** from problem  $P$  to problem  $Q$ : a function  $\phi$  with the following properties:

- $\phi(x)$  can be computed in time  $f(k) \cdot |x|^{O(1)}$ , where  $k$  is the parameter of  $x$ ,
- $\phi(x)$  is a yes-instance of  $Q \iff x$  is a yes-instance of  $P$ .
- If  $k$  is the parameter of  $x$  and  $k'$  is the parameter of  $\phi(x)$ , then  $k' \leq g(k)$  for some function  $g$ .

**Fact:** If there is a parameterized reduction from problem  $P$  to problem  $Q$  and  $Q$  is FPT, then  $P$  is also FPT.



# Parameterized reduction

## Definition

**Parameterized reduction** from problem  $P$  to problem  $Q$ : a function  $\phi$  with the following properties:

- $\phi(x)$  can be computed in time  $f(k) \cdot |x|^{O(1)}$ , where  $k$  is the parameter of  $x$ ,
- $\phi(x)$  is a yes-instance of  $Q \iff x$  is a yes-instance of  $P$ .
- If  $k$  is the parameter of  $x$  and  $k'$  is the parameter of  $\phi(x)$ , then  $k' \leq g(k)$  for some function  $g$ .

**Fact:** If there is a parameterized reduction from problem  $P$  to problem  $Q$  and  $Q$  is FPT, then  $P$  is also FPT.

**Non-example:** Transforming an **INDEPENDENT SET** instance  $(G, k)$  into a **VERTEX COVER** instance  $(G, n - k)$  is not a parameterized reduction.

**Example:** Transforming an **INDEPENDENT SET** instance  $(G, k)$  into a **CLIQUE** instance  $(\overline{G}, k)$  is a parameterized reduction.

# MULTICOLORED CLIQUE

A useful variant of **CLIQUE**:

**MULTICOLORED CLIQUE**: The vertices of the input graph  $G$  are colored with  $k$  colors and we have to find a clique containing one vertex from each color.

## Theorem

There is a parameterized reduction from **CLIQUE** to **MULTICOLORED CLIQUE**.

Proof by reduction from **CLIQUE** (see board).

Also: reduction to **MULTICOLORED INDEPENDENT SET**.

## A reduction

### Theorem

There is a parameterized reduction from **MULTICOLORED INDEPENDENT SET** to **DOMINATING SET**.

**Proof:** Let  $G$  be a graph with  $n$  vertices,  $m$  edges, and let  $k$  be an integer. We construct a graph  $H$  such that  $G$  has an independent set of size  $k$  if and only if  $H$  has a dominating set of size  $k$ .

The dominating set has to contain one vertex from each of the  $k$  cliques. Additional vertices ensure that these selections describe an independent set.

(See the blackboard.)

# Hard problems

Hundreds of parameterized problems are known to be at least as hard as **CLIQUE**:

- SET COVER
- HITTING SET
- CONNECTED DOMINATING SET
- INDEPENDENT DOMINATING SET
- PARTIAL VERTEX COVER
- DOMINATING SET in bipartite graphs
- ...

## Basic hypotheses

Parameterized complexity theory cannot be built on assuming  $P \neq NP$  – we have to assume something stronger.

Let us choose a basic hypothesis:

### Engineers' Hypothesis

$k$ -CLIQUE cannot be solved in time  $f(k) \cdot n^{O(1)}$ .

## Basic hypotheses

Parameterized complexity theory cannot be built on assuming  $P \neq NP$  – we have to assume something stronger.

Let us choose a basic hypothesis:

### Engineers' Hypothesis

$k$ -CLIQUE cannot be solved in time  $f(k) \cdot n^{O(1)}$ .

### Theorists' Hypothesis

$k$ -STEP HALTING PROBLEM (is there a path of the given NTM that stops in  $k$  steps?) cannot be solved in time  $f(k) \cdot n^{O(1)}$ .

## Basic hypotheses

Parameterized complexity theory cannot be built on assuming  $P \neq NP$  – we have to assume something stronger.

Let us choose a basic hypothesis:

### Engineers' Hypothesis

$k$ -CLIQUE cannot be solved in time  $f(k) \cdot n^{O(1)}$ .

### Theorists' Hypothesis

$k$ -STEP HALTING PROBLEM (is there a path of the given NTM that stops in  $k$  steps?) cannot be solved in time  $f(k) \cdot n^{O(1)}$ .

### Exponential Time Hypothesis (ETH)

$n$ -variable 3SAT cannot be solved in time  $2^{o(n)}$ .

Which hypothesis is the most plausible?

## Basic hypotheses

Parameterized complexity theory cannot be built on assuming  $P \neq NP$  – we have to assume something stronger.

Let us choose a basic hypothesis:

### Engineers' Hypothesis

$k$ -CLIQUE cannot be solved in time  $f(k) \cdot n^{O(1)}$ .



### Theorists' Hypothesis

$k$ -STEP HALTING PROBLEM (is there a path of the given NTM that stops in  $k$  steps?) cannot be solved in time  $f(k) \cdot n^{O(1)}$ .



### Exponential Time Hypothesis (ETH)

$n$ -variable 3SAT cannot be solved in time  $2^{o(n)}$ .

Which hypothesis is the most plausible?



# INDEPENDENT SET and Turing machines

## Theorem

There is a parameterized reduction from INDEPENDENT SET to the  $k$ -STEP HALTING PROBLEM.

**Proof:** Given a graph  $G$  and an integer  $k$ , we construct a Turing machine  $M$  and an integer  $k' = O(k^2)$  such that  $M$  halts in  $k'$  steps if and only if  $G$  has an independent set of size  $k$ .

# INDEPENDENT SET and Turing machines

## Theorem

There is a parameterized reduction from INDEPENDENT SET to the  $k$ -STEP HALTING PROBLEM.

**Proof:** Given a graph  $G$  and an integer  $k$ , we construct a Turing machine  $M$  and an integer  $k' = O(k^2)$  such that  $M$  halts in  $k'$  steps if and only if  $G$  has an independent set of size  $k$ .

The alphabet of  $M$  is the vertices of  $G$ .

- In the first  $k$  steps,  $M$  nondeterministically writes  $k$  vertices to the first  $k$  cells.
- For every  $1 \leq i \leq k$ ,  $M$  moves to the  $i$ -th cell, stores the vertex in the internal state, and goes through the tape to check that every other vertex is nonadjacent with the  $i$ -th vertex (otherwise  $M$  loops).
- $M$  does  $k$  checks and each check can be done in  $2k$  steps  $\Rightarrow k' = O(k^2)$ .

# INDEPENDENT SET and Turing machines

## Theorem

There is a parameterized reduction from the  $k$ -STEP HALTING PROBLEM to INDEPENDENT SET.

**Proof:** Given a Turing machine  $M$  and an integer  $k$ , we construct a graph  $G$  that has an independent set of size  $k' := k^2$  if and only if  $M$  halts in  $k$  steps.

# INDEPENDENT SET and Turing machines

## Theorem

There is a parameterized reduction from the  $k$ -STEP HALTING PROBLEM to INDEPENDENT SET.

**Proof:** Given a Turing machine  $M$  and an integer  $k$ , we construct a graph  $G$  that has an independent set of size  $k' := k^2$  if and only if  $M$  halts in  $k$  steps.

- $G$  consists of  $k^2$  cliques, thus a  $k'$ -independent set has to contain one vertex from each.
- The selected vertex from clique  $K_{i,j}$  describes what happens in Step  $i$  at cell  $j$ : what is written there, is the head there, and if so, what the state is, and what the next transition is.
- We add edges between the cliques to rule out inconsistencies: head is at more than one location at the same time, wrong character is written, head moves in the wrong direction etc.

## Summary

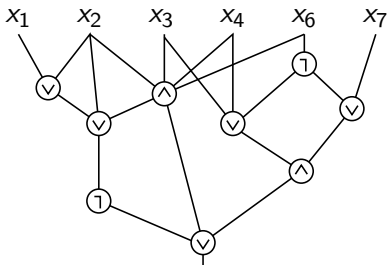
- INDEPENDENT SET and  $k$ -STEP HALTING PROBLEM can be reduced to each other  $\Rightarrow$  Engineers' Hypothesis and Theorists' Hypothesis are equivalent!
- INDEPENDENT SET and  $k$ -STEP HALTING PROBLEM can be reduced to DOMINATING SET.

# Summary

- INDEPENDENT SET and  $k$ -STEP HALTING PROBLEM can be reduced to each other  $\Rightarrow$  Engineers' Hypothesis and Theorists' Hypothesis are equivalent!
- INDEPENDENT SET and  $k$ -STEP HALTING PROBLEM can be reduced to DOMINATING SET.
- Is there a parameterized reduction from DOMINATING SET to INDEPENDENT SET?
- Probably not. Unlike in NP-completeness, where most problems are equivalent, here we have a hierarchy of hard problems.
  - INDEPENDENT SET is  $W[1]$ -complete.
  - DOMINATING SET is  $W[2]$ -complete.
- Does not matter if we only care about whether a problem is FPT or not!

## Boolean circuit

A **Boolean circuit** consists of input gates, negation gates, AND gates, OR gates, and a single output gate.



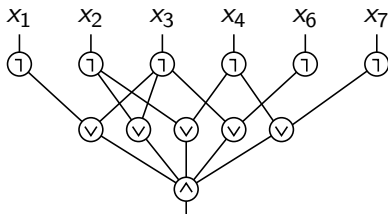
**CIRCUIT SATISFIABILITY:** Given a Boolean circuit  $C$ , decide if there is an assignment on the inputs of  $C$  such that the output is true.

**WEIGHTED CIRCUIT SATISFIABILITY:** Given a Boolean circuit  $C$  and an integer  $k$ , decide if there is an assignment of weight  $k$  such that the output is true.

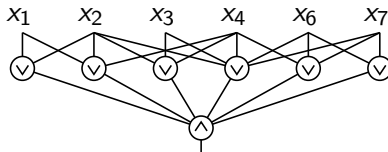
Weight of an assignment: number of true values.

# WEIGHTED CIRCUIT SATISFIABILITY

INDEPENDENT SET can be reduced to WEIGHTED CIRCUIT SATISFIABILITY:



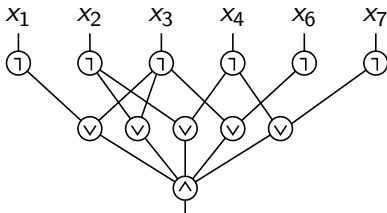
DOMINATING SET can be reduced to WEIGHTED CIRCUIT SATISFIABILITY:



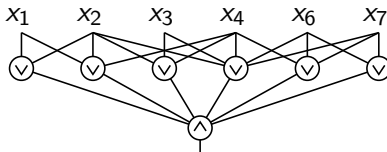


# WEIGHTED CIRCUIT SATISFIABILITY

INDEPENDENT SET can be reduced to WEIGHTED CIRCUIT SATISFIABILITY:



DOMINATING SET can be reduced to WEIGHTED CIRCUIT SATISFIABILITY:



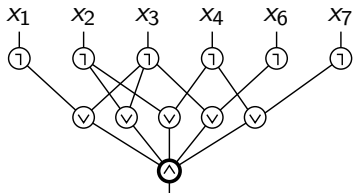
To express DOMINATING SET, we need more complicated circuits.

## Depth and weft

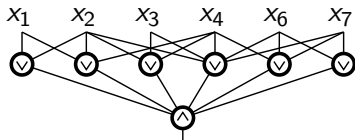
The **depth** of a circuit is the maximum length of a path from an input to the output.

A gate is **large** if it has more than 2 inputs. The **weft** of a circuit is the maximum number of large gates on a path from an input to the output.

**INDEPENDENT SET:** weft 1, depth 3



**DOMINATING SET:** weft 2, depth 2



## The W-hierarchy

Let  $C[t, d]$  be the set of all circuits having weft at most  $t$  and depth at most  $d$ .

### Definition

A problem  $P$  is in the class  $W[t]$  if there is a constant  $d$  and a parameterized reduction from  $P$  to **WEIGHTED CIRCUIT SATISFIABILITY** of  $C[t, d]$ .

We have seen that **INDEPENDENT SET** is in  $W[1]$  and **DOMINATING SET** is in  $W[2]$ .

**Fact:** **INDEPENDENT SET** is  $W[1]$ -complete.

**Fact:** **DOMINATING SET** is  $W[2]$ -complete.

## The W-hierarchy

Let  $C[t, d]$  be the set of all circuits having weft at most  $t$  and depth at most  $d$ .

### Definition

A problem  $P$  is in the class  $W[t]$  if there is a constant  $d$  and a parameterized reduction from  $P$  to **WEIGHTED CIRCUIT SATISFIABILITY** of  $C[t, d]$ .

We have seen that **INDEPENDENT SET** is in  $W[1]$  and **DOMINATING SET** is in  $W[2]$ .

**Fact:** **INDEPENDENT SET** is  $W[1]$ -complete.

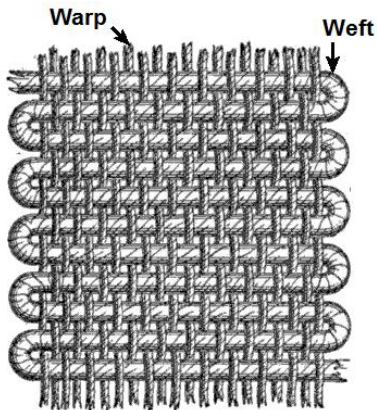
**Fact:** **DOMINATING SET** is  $W[2]$ -complete.

If any  $W[1]$ -complete problem is FPT, then  $FPT = W[1]$  and every problem in  $W[1]$  is FPT.

If any  $W[2]$ -complete problem is in  $W[1]$ , then  $W[1] = W[2]$ .

$\Rightarrow$  If there is a parameterized reduction from **DOMINATING SET** to **INDEPENDENT SET**, then  $W[1] = W[2]$ .

# Weft



**Weft** is a term related to weaving cloth: it is the thread that runs from side to side in the fabric.

# LIST COLORING

LIST COLORING is a generalization of ordinary vertex coloring:  
given a

- graph  $G$ ,
- a set of colors  $C$ , and
- a list  $L(v) \subseteq C$  for each vertex  $v$ ,

the task is to find a coloring  $c$  where  $c(v) \in L(v)$  for every  $v$ .

## Theorem

VERTEX COLORING is FPT parameterized by treewidth.

However, list coloring is more difficult:

## Theorem

LIST COLORING is  $W[1]$ -hard parameterized by treewidth.

# LIST COLORING

## Theorem

LIST COLORING is  $W[1]$ -hard parameterized by treewidth.

**Proof:** By reduction from MULTICOLORED INDEPENDENT SET.

- Let  $G$  be a graph with color classes  $V_1, \dots, V_k$ .
- In the LIST COLORING instance, the set  $C$  of colors is the set of vertices of  $G$ .
- The colors appearing on vertices  $u_1, \dots, u_k$  correspond to the  $k$  vertices of the clique, hence we set  $L(u_i) = V_i$ .

# LIST COLORING

## Theorem

LIST COLORING is  $W[1]$ -hard parameterized by treewidth.

**Proof:** By reduction from MULTICOLORED INDEPENDENT SET.

- Let  $G$  be a graph with color classes  $V_1, \dots, V_k$ .
- In the LIST COLORING instance, the set  $C$  of colors is the set of vertices of  $G$ .
- The colors appearing on vertices  $u_1, \dots, u_k$  correspond to the  $k$  vertices of the clique, hence we set  $L(u_i) = V_i$ .
- If  $x \in V_i$  and  $y \in V_j$  are adjacent in  $G$ , then we need to ensure that  $c(u_i) = x$  and  $c(u_j) = y$  are not true at the same time  $\Rightarrow$  we add a vertex adjacent to  $u_i$  and  $u_j$  whose list is  $\{x, y\}$ .



## Summary

- By parameterized reductions, we can show that lots of problems are at least as hard as **CLIQUE**, hence unlikely to be fixed-parameter tractable.
- Connection with Turing machine gives some supporting evidence for hardness (only of theoretical interest).
- The **W**-hierarchy classifies the problems according to hardness (only of theoretical interest).

Part II:  
Exponential Time  
Hypothesis

# Exponential Time Hypothesis (ETH)

Hypothesis introduced by Impagliazzo, Paturi, and Zane:

## Exponential Time Hypothesis (ETH)

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

**Note:** current best algorithm is  $1.30704^n$  [Hertli 2011].

**Note:** an  $n$ -variable 3SAT formula can have  $\Omega(n^3)$  clauses.

# Exponential Time Hypothesis (ETH)

Hypothesis introduced by Impagliazzo, Paturi, and Zane:

## Exponential Time Hypothesis (ETH)

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

**Note:** current best algorithm is  $1.30704^n$  [Hertli 2011].

**Note:** an  $n$ -variable 3SAT formula can have  $\Omega(n^3)$  clauses.

## Sparsification Lemma [Impagliazzo, Paturi, Zane 2001]

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



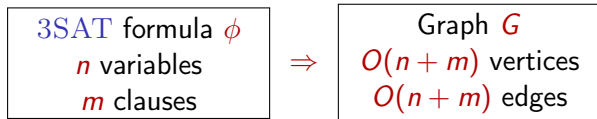
There is a  $2^{o(m)}$ -time algorithm for  $m$ -clause 3SAT.

## 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

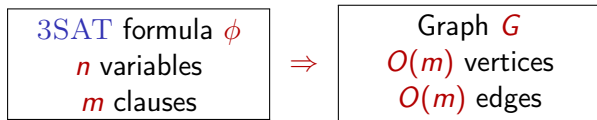
Assuming ETH, there is no  $2^{o(n)}$  algorithm for 3-COLORING on an  $n$ -vertex graph  $G$ .

## 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  $G$ .

## Transferring 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
- ...

## Transferring 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
- ...



## Transferring 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

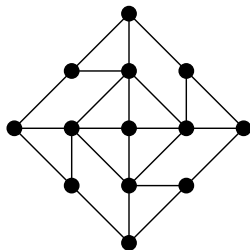
- ~~$k$ -INDEPENDENT SET~~
- ~~$k$ -CLIQUE~~
- ~~$k$ -DOMINATING SET~~
- $k$ -VERTEX COVER
- $k$ -PATH
- $k$ -FEEDBACK VERTEX SET
- ...

## 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:



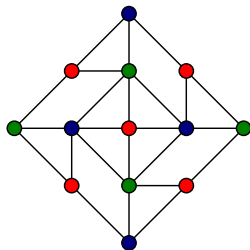
- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to the whole gadget.
- If two edges cross, replace them with a crossover gadget.

## 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:



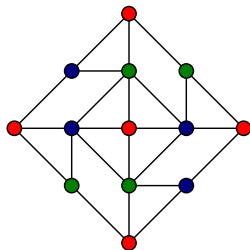
- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to the whole gadget.
- If two edges cross, replace them with a crossover gadget.

## Lower bounds based on ETH

What about 3-COLORING on planar graphs?

The textbook reduction from 3-COLORING to PLANAR

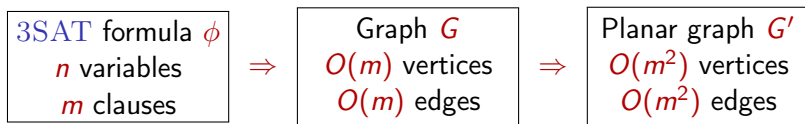
3-COLORING uses a “crossover gadget” with 4 external connectors:



- 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.



### 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?).



# Exponential Time Hypothesis

## Engineers' Hypothesis

$k$ -CLIQUE cannot be solved in time  $f(k) \cdot n^{O(1)}$ .



## Theorists' Hypothesis

$k$ -STEP HALTING PROBLEM (is there a path of the given NTM that stops in  $k$  steps?) cannot be solved in time  $f(k) \cdot n^{O(1)}$ .



## Exponential Time Hypothesis (ETH)

$n$ -variable 3SAT cannot be solved in time  $2^{o(n)}$ .

What do we have to show to prove that ETH implies Engineers' Hypothesis?

# Exponential Time Hypothesis

## Engineers' Hypothesis

$k$ -CLIQUE cannot be solved in time  $f(k) \cdot n^{O(1)}$ .



## Theorists' Hypothesis

$k$ -STEP HALTING PROBLEM (is there a path of the given NTM that stops in  $k$  steps?) cannot be solved in time  $f(k) \cdot n^{O(1)}$ .



## Exponential Time Hypothesis (ETH)

$n$ -variable 3SAT cannot be solved in time  $2^{o(n)}$ .

What do we have to show to prove that ETH implies Engineers' Hypothesis?

We have to show that an  $f(k) \cdot n^{O(1)}$  algorithm implies that there is a  $2^{o(n)}$  time algorithm for  $n$ -variable 3SAT.

# Exponential Time Hypothesis

## Engineers' Hypothesis

$k$ -CLIQUE cannot be solved in time  $f(k) \cdot n^{O(1)}$ .



## Theorists' Hypothesis

$k$ -STEP HALTING PROBLEM (is there a path of the given NTM that stops in  $k$  steps?) cannot be solved in time  $f(k) \cdot n^{O(1)}$ .



## Exponential Time Hypothesis (ETH)

$n$ -variable 3SAT cannot be solved in time  $2^{o(n)}$ .

We actually show something much stronger and more interesting:

## Theorem [Chen et al. 2004]

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

## Lower bound on the exponent

Theorem [Chen et al. 2004]

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

Suppose that  $k$ -CLIQUE can be solved in time  $f(k) \cdot n^{k/s(k)}$ , where  $s(k)$  is a monotone increasing unbounded function. We use this algorithm to solve 3-COLORING on an  $n$ -vertex graph  $G$  in time  $2^{o(n)}$ .

## Lower bound on the exponent

### Theorem [Chen et al. 2004]

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

Suppose that  $k$ -CLIQUE can be solved in time  $f(k) \cdot n^{k/s(k)}$ , where  $s(k)$  is a monotone increasing unbounded function. We use this algorithm to solve 3-COLORING on an  $n$ -vertex graph  $G$  in time  $2^{o(n)}$ .

Let  $k$  be the largest integer such that  $f(k) \leq n$  and  $k^{k/s(k)} \leq n$ . Function  $k := k(n)$  is monotone increasing and unbounded.

Split the vertices of  $G$  into  $k$  groups. Let us build a graph  $H$  where each vertex corresponds to a proper 3-coloring of one of the groups. Connect two vertices if they are not conflicting.

## Lower bound on the exponent

### Theorem [Chen et al. 2004]

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

Suppose that  $k$ -CLIQUE can be solved in time  $f(k) \cdot n^{k/s(k)}$ , where  $s(k)$  is a monotone increasing unbounded function. We use this algorithm to solve 3-COLORING on an  $n$ -vertex graph  $G$  in time  $2^{o(n)}$ .

Let  $k$  be the largest integer such that  $f(k) \leq n$  and  $k^{k/s(k)} \leq n$ . Function  $k := k(n)$  is monotone increasing and unbounded.

Split the vertices of  $G$  into  $k$  groups. Let us build a graph  $H$  where each vertex corresponds to a proper 3-coloring of one of the groups. Connect two vertices if they are not conflicting.

Every  $k$ -clique of  $H$  corresponds to a proper 3-coloring of  $G$ .

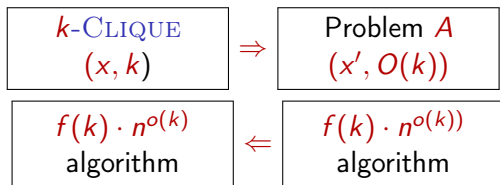
$\Rightarrow$  A 3-coloring of  $G$  can be found in time  $f(k) \cdot |V(H)|^{k/s(k)} \leq n \cdot (k^{3n/k})^{k/s(k)} = n \cdot k^{k/s(k)} \cdot 3^{n/s(k)} = 2^{o(n)}$ .

## Tight bounds

Theorem [Chen et al. 2004]

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

Transferring to other problems:

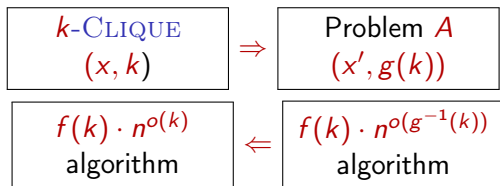


## Tight bounds

Theorem [Chen et al. 2004]

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

Transferring to other problems:



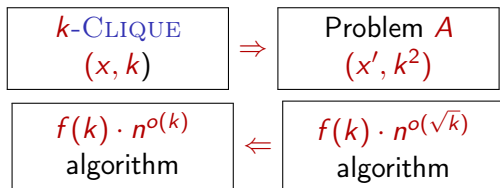


## Tight bounds

Theorem [Chen et al. 2004]

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

Transferring to other problems:

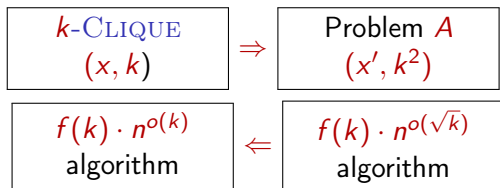


# Tight bounds

## Theorem [Chen et al. 2004]

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

Transferring 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
- ...

# 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.

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

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

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

$$k = 3, D = 5$$

# 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.

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

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

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

$$k = 3, D = 5$$

# 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.

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

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

## 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.

	$(v_i, v_i)$			

Each diagonal cell defines a value  $v_i \dots$



# 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.

	$(v_i, \cdot)$			
$(\cdot, v_i)$	$(v_i, v_i)$	$(\cdot, v_i)$	$(\cdot, v_i)$	$(\cdot, v_i)$
	$(v_i, \cdot)$			
	$(v_i, \cdot)$			
	$(v_i, \cdot)$			

... 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.

	$(v_i, \cdot)$			
$(\cdot, v_i)$	$(v_i, v_i)$	$(\cdot, v_i)$	$(\cdot, v_i)$	$(\cdot, v_i)$
	$(v_i, \cdot)$			
	$(v_i, \cdot)$		$(v_j, v_j)$	
	$(v_i, \cdot)$			

$v_i$  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.

	$(v_i, \cdot)$		$(v_j, \cdot)$	
$(\cdot, v_i)$	$(v_i, v_i)$	$(\cdot, v_i)$	$(v_i, v_j)$	$(\cdot, v_i)$
	$(v_i, \cdot)$		$(v_j, \cdot)$	
$(\cdot, v_j)$	$(v_j, v_i)$	$(\cdot, v_j)$	$(v_j, v_j)$	$(\cdot, v_j)$
	$(v_i, \cdot)$		$(v_j, \cdot)$	

$v_i$  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:

- LIST COLORING on planar graphs
- MULTIWAY CUT on planar graphs with  $k$  terminals
- INDEPENDENT SET for unit disks

# LIST COLORING for planar graphs

## Theorem

LIST COLORING for planar graphs is  $W[1]$ -hard parameterized by treewidth.

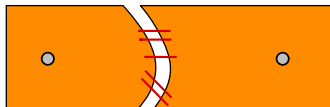
Proof is similar to the reduction from MULTICOLORED CLIQUE to LIST COLORING, but now the resulting graph is planar.

## A classical problem

### $s - t$ CUT

Input: A graph  $G$ , an integer  $p$ , vertices  $s$  and  $t$

Output: A set  $S$  of at most  $p$  edges such that removing  $S$  separates  $s$  and  $t$ .



### Theorem [Ford and Fulkerson 1956]

A minimum  $s - t$  cut can be found in polynomial time.

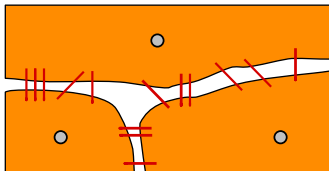
What about separating more than two terminals?

## More than two terminals

### $k$ -TERMINAL CUT (aka MULTIWAY CUT)

Input: A graph  $G$ , an integer  $p$ , and a set  $T$  of  $k$  terminals

Output: A set  $S$  of at most  $p$  edges such that removing  $S$  separates any two vertices of  $T$



Theorem [Dalhaus et al. 1994]

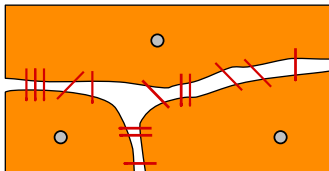
NP-hard already for  $k = 3$ .

## More than two terminals

### $k$ -TERMINAL CUT (aka MULTIWAY CUT)

Input: A graph  $G$ , an integer  $p$ , and a set  $T$  of  $k$  terminals

Output: A set  $S$  of at most  $p$  edges such that removing  $S$  separates any two vertices of  $T$



Theorem [Dalhaus et al. 1994] [Hartvigsen 1998] [Bentz 2012]

PLANAR  $k$ -TERMINAL CUT can be solved in time  $n^{O(k)}$ .

Theorem [Klein and M. 2012]

PLANAR  $k$ -TERMINAL CUT can be solved in time  $2^{O(k)} \cdot n^{O(\sqrt{k})}$ .



## Lower bounds

Theorem [Klein and M. 2012]

PLANAR  $k$ -TERMINAL CUT can be solved in time  $2^{O(k)} \cdot n^{O(\sqrt{k})}$ .

Natural questions:

- Is there an  $f(k) \cdot n^{o(\sqrt{k})}$  time algorithm?
- Is there an  $f(k) \cdot n^{O(1)}$  time algorithm (i.e., is it fixed-parameter tractable)?

## Lower bounds

Theorem [Klein and M. 2012]

PLANAR  $k$ -TERMINAL CUT can be solved in time  $2^{O(k)} \cdot n^{O(\sqrt{k})}$ .

Natural questions:

- Is there an  $f(k) \cdot n^{o(\sqrt{k})}$  time algorithm?
- Is there an  $f(k) \cdot n^{O(1)}$  time algorithm (i.e., is it fixed-parameter tractable)?

**Lower bounds:**

Theorem [M. 2012]

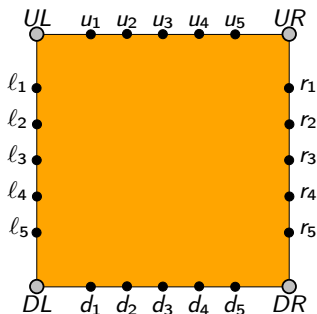
PLANAR  $k$ -TERMINAL CUT is  $W[1]$ -hard and has no  $f(k) \cdot n^{o(\sqrt{k})}$  time algorithm (assuming ETH).

## Reduction from $k \times k$ GRID TILING to PLANAR $k^2$ -TERMINAL CUT

For every set  $S_{i,j}$ , we construct a gadget with 4 terminals such that

- for every  $(x, y) \in S_{i,j}$ , there is a minimum multiway cut that represents  $(x, y)$ .
- every minimum multiway cut represents some  $(x, y) \in S_{i,j}$ .

Main part of the proof: constructing these gadgets.



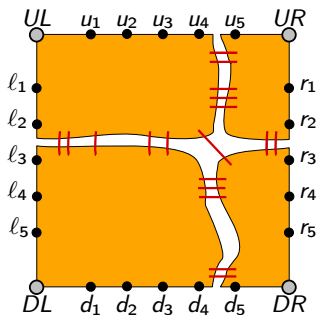
The gadget.

## Reduction from $k \times k$ GRID TILING to PLANAR $k^2$ -TERMINAL CUT

For every set  $S_{i,j}$ , we construct a gadget with 4 terminals such that

- for every  $(x, y) \in S_{i,j}$ , there is a minimum multiway cut that represents  $(x, y)$ .
- every minimum multiway cut represents some  $(x, y) \in S_{i,j}$ .

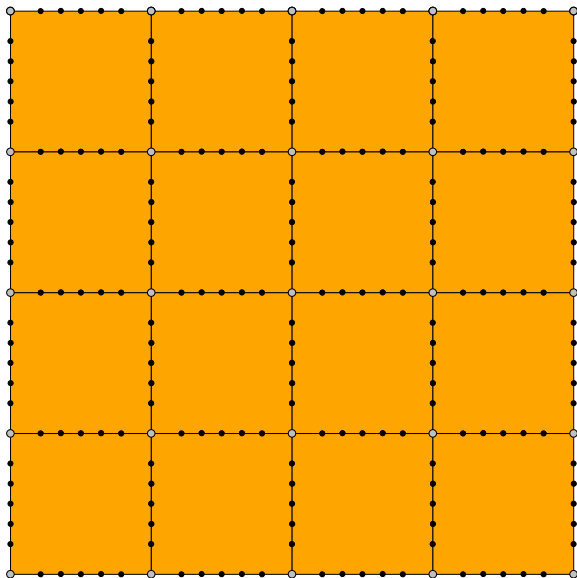
Main part of the proof: constructing these gadgets.



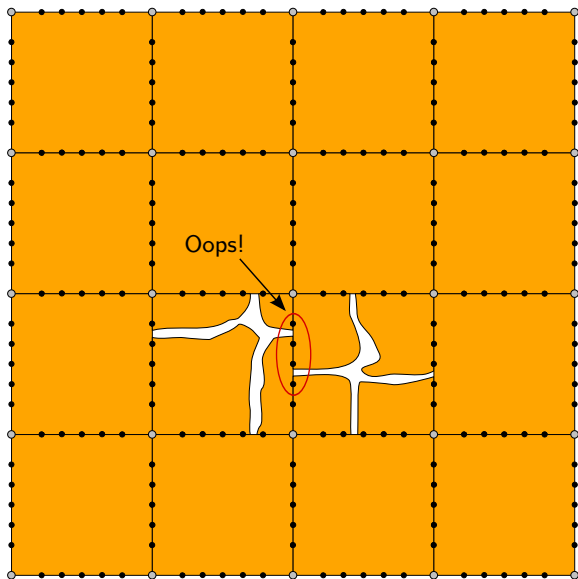
A cut representing  $(4, 2)$ .



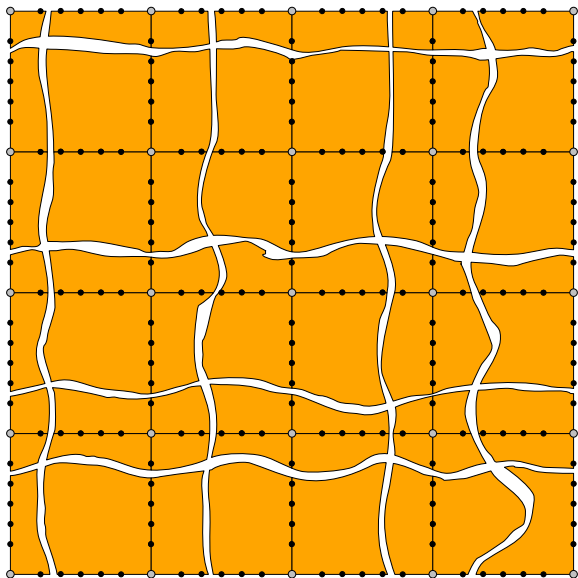
## Putting together the gadgets



## Putting together the gadgets



## Putting together the gadgets





# 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.

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

- Find:*
- 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.

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

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

### Theorem

There is a parameterized reduction from  $k \times k$ -GRID TILING to  $O(k) \times O(k)$  GRID TILING WITH  $\leq$ .

## $k$ -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

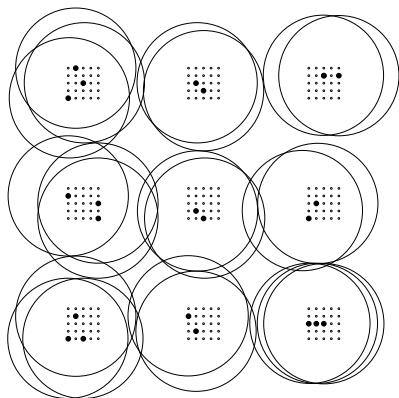
(5,1) (1,2) (3,3)	(4,3) (3,2)	(2,3) (2,5)			
(2,1) (5,5) (3,5)	(4,2) (5,3)	(5,1) (3,2)			
(5,1) (2,2) (5,3)	(2,1) (4,2)	(3,1) (3,2) (3,3)			

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)

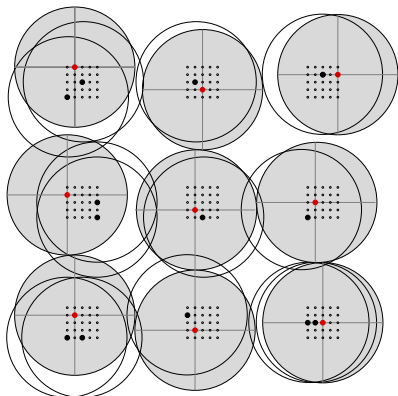


Every pair is represented by a unit disk in the plane.

$\leq$  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**.

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.



Part III:  
Approximation schemes

# Approximation schemes

## Polynomial-time approximation scheme (PTAS)

Input: Instance  $x$ ,  $\epsilon > 0$   
Output:  $(1 + \epsilon)$ -approximate solution  
Running time: polynomial in  $|x|$  for every fixed  $\epsilon$

- **PTAS:** running time is  $|x|^{f(1/\epsilon)}$
- **EPTAS:** (Efficient PTAS) running time is  $f(1/\epsilon) \cdot |x|^{O(1)}$
- **FPTAS:** (Fully polynomial approximation scheme) running time is  $(1/\epsilon)^{O(1)} \cdot |x|^{O(1)}$

For some problems, there is a PTAS, but no EPTAS is known. Can we show that no EPTAS is possible?

## Standard parameterization

Given an **optimization** problem we can turn it into a **decision** problem: the input is a pair  $(x, k)$  and we have to decide if there is a solution for  $x$  with cost at least/at most  $k$ .

The **standard parameterization** of an optimization problem is the associated decision problem, with the value  $k$  appearing in the input being the parameter.

**Example:**

### VERTEX COVER

Input:  $(G, k)$

Parameter:  $k$

Question: Is there a vertex cover of size at most  $k$ ?

If the standard parameterization of an optimization problem is FPT, then (intuitively) it means that we can solve it efficiently if the optimum is small.

# No EPTAS

## Theorem

If the standard parameterization of an optimization problem is  $W[1]$ -hard, then there is no EPTAS for the optimization problem, unless  $FPT = W[1]$ .

**Proof:** Suppose an  $f(1/\epsilon) \cdot n^{O(1)}$  time EPTAS exists. Running this EPTAS with  $\epsilon := 1/(k+1)$  decides if the optimum is at most/at least  $k$ .

# No EPTAS

## Theorem

If the standard parameterization of an optimization problem is  $W[1]$ -hard, then there is no EPTAS for the optimization problem, unless  $FPT = W[1]$ .

**Proof:** Suppose an  $f(1/\epsilon) \cdot n^{O(1)}$  time EPTAS exists. Running this EPTAS with  $\epsilon := 1/(k+1)$  decides if the optimum is at most/at least  $k$ .

**Example:** The  $W[1]$ -hardness results immediately shows that (assuming  $W[1] \neq FPT$ ), there is no EPTAS for **INDEPENDENT SET** for unit disks.

# Summary

- Parameterized reductions show that many problems are at least as hard as **CLIQUE**, hence unlikely to be FPT.
- ETH gives tighter lower bounds, e.g., no  $2^{o(k)}n^{O(1)}$  for FPT problems and no  $f(k)n^{o(k)}$  algorithms for **W[1]**-hard problems.
- Connection to approximation: ruling out EPTAS via **W[1]**-hardness.