Minicourse on parameterized algorithms and complexity

Part 7: Parameterized complexity

Dániel Marx

Jagiellonian University in Kraków
April 21-23, 2015

Lower bounds

So far we have seen positive results: basic algorithmic techniques for fixed-parameter tractability.

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: basic algorithmic techniques for fixed-parameter tractability.

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 $\mathrm{P} \neq \mathrm{NP}$: if $\mathrm{P} = \mathrm{NP}$, then, e.g., k-Clique is polynomial-time solvable, hence FPT.

Can we give some evidence for negative results?

Goals of this talk

Two goals:

1. Explain the theory behind parameterized intractability.
2. Show examples of parameterized reductions.

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 ϕ 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., $\mathrm{P} = \mathrm{NP}$).

Parameterized complexity

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

- 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 concepts:

- 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 $n - k$.

$\Rightarrow$ 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 ϕ 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 ϕ 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.

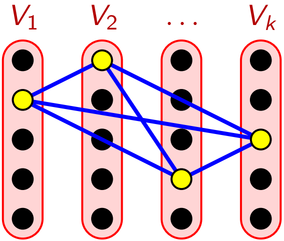
Example: Transforming an Independent Set instance (G, k) into a Clique instance $(\overline{G}, k)$ **is** a parameterized reduction.

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.

(or Partitioned Clique)



Theorem

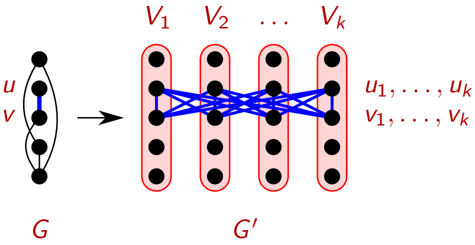
There is a parameterized reduction from Clique to Multicolored Clique.

Multicolored Clique

Theorem

There is a parameterized reduction from Clique to Multicolored Clique.

Create G' by replacing each vertex v with k vertices, one in each color class. If u and v are adjacent in the original graph, connect all copies of u with all copies of v.



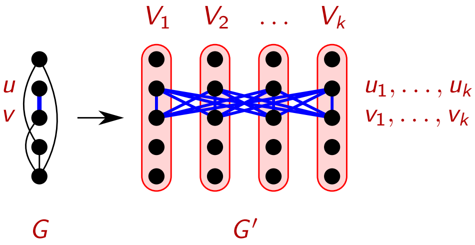
k-clique in G $\iff$ multicolored k-clique in G'.

Multicolored Clique

Theorem

There is a parameterized reduction from Clique to Multicolored Clique.

Create G' by replacing each vertex v with k vertices, one in each color class. If u and v are adjacent in the original graph, connect all copies of u with all copies of v.



k-clique in G $\iff$ multicolored k-clique in G'.

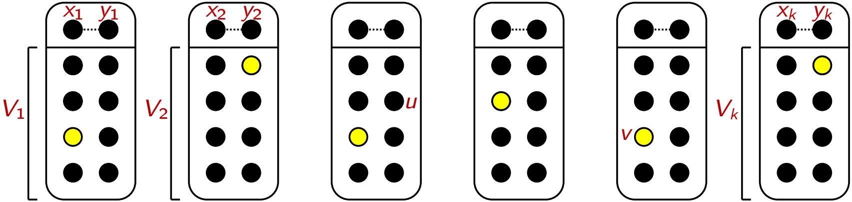
Similarly: reduction to Multicolored Independent Set.

Dominating Set

Theorem

There is a parameterized reduction from Multicolored Independent Set to Dominating Set.

Proof: Let G be a graph with color classes $V_1, \ldots, V_k$. We construct a graph H such that G has a multicolored k-clique iff H has a dominating set of size k.



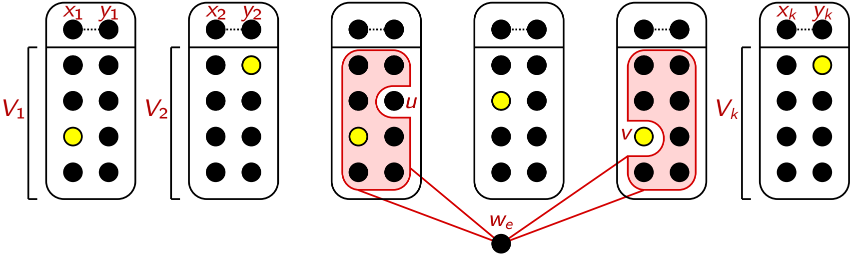
- The dominating set has to contain one vertex from each of the k cliques $V_1, \ldots, V_k$ to dominate every x_i and y_i.

Dominating Set

Theorem

There is a parameterized reduction from Multicolored Independent Set to Dominating Set.

Proof: Let G be a graph with color classes $V_1, \ldots, V_k$. We construct a graph H such that G has a multicolored k-clique iff H has a dominating set of size k.



- The dominating set has to contain one vertex from each of the k cliques $V_1, \ldots, V_k$ to dominate every x_i and y_i.
- For every edge $e = uv$, an additional vertex w_e ensures that these selections describe an independent set.

Variants of Dominating Set

- Dominating Set: Given a graph, find k vertices that dominate every vertex.
- Red-Blue Dominating Set: Given a bipartite graph, find k vertices on the red side that dominate the blue side.
- Set Cover: Given a set system, find k sets whose union covers the universe.
- Hitting Set: Given a set system, find k elements that intersect every set in the system.

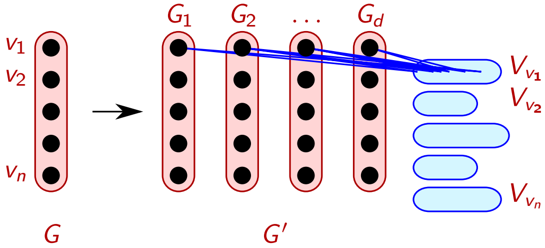
All of these problems are equivalent under parameterized reductions, hence at least as hard as Clique.

Regular graphs

Theorem

There is a parameterized reduction from CLIQUE to CLIQUE on regular graphs.

Proof: Given a graph G and an integer k, let d be the maximum degree of G. Take d copies of G and for every $v \in V(G)$, fully connect every copy of v with a set V_v of $d - d(v)$ vertices.



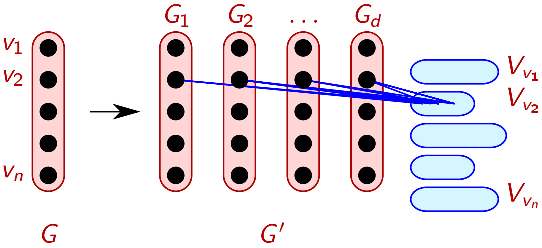
Observe the edges incident to V_v do not appear in any triangle, hence every k-clique of G' is a k-clique of G (assuming $k \geq 3$).

Regular graphs

Theorem

There is a parameterized reduction from Clique to Clique on regular graphs.

Proof: Given a graph G and an integer k, let d be the maximum degree of G. Take d copies of G and for every $v \in V(G)$, fully connect every copy of v with a set V_v of $d - d(v)$ vertices.



Observe the edges incident to V_v do not appear in any triangle, hence every k-clique of G' is a k-clique of G (assuming $k \geq 3$).

Partial Vertex Cover

Partial Vertex Cover: Given a graph G, integers k and s, find k vertices that cover at least s edges.

Theorem

There is a parameterized reduction from Independent Set on regular graphs parameterized by k to Partial Vertex Cover parameterized by k.

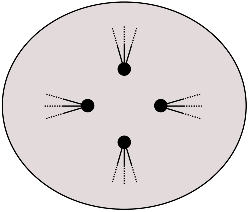
Partial Vertex Cover

Partial Vertex Cover: Given a graph G, integers k and s, find k vertices that cover at least s edges.

Theorem

There is a parameterized reduction from Independent Set on regular graphs parameterized by k to Partial Vertex Cover parameterized by k.

Proof: If G is d-regular, then k vertices can cover $s := kd$ edges if and only if there is a independent set of size k.



$d = 3$, $k = 4$, $s = 12$

Hard problems

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

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

We believe that none of these problems are FPT.

Basic hypotheses

It seems that parameterized complexity theory cannot be built on assuming $\mathrm{P} \neq \mathrm{NP}$ – 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

It seems that parameterized complexity theory cannot be built on assuming $\mathrm{P} \neq \mathrm{NP}$ – 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

It seems that parameterized complexity theory cannot be built on assuming $\mathrm{P} \neq \mathrm{NP}$ – 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

It seems that parameterized complexity theory cannot be built on assuming $\mathsf{P} \neq \mathsf{NP}$ – we have to assume something stronger.

Let us choose a basic hypothesis:

Engineers' Hypothesis

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

$\Updownarrow$

Theorists' Hypothesis

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

$\Uparrow$

Exponential Time Hypothesis (ETH)

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

Which hypothesis is the most plausible?

Independent Set $\Rightarrow$ 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 $\Rightarrow$ 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 Σ of M is the set of vertices of G.

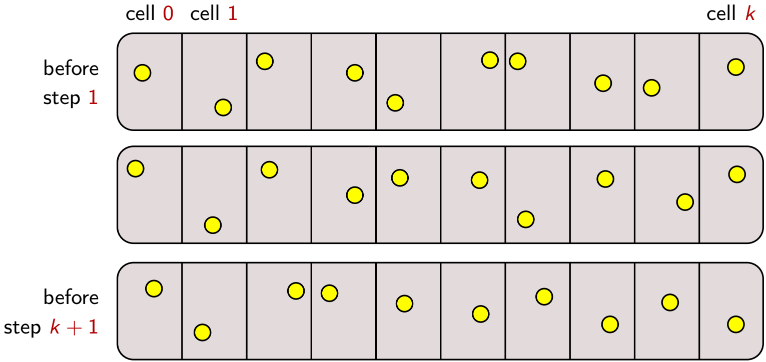
- In the first k steps, M nondeterministically writes k vertices to the first k cells.
- For every $1 \leq i \leq k$, M moves to the i-th cell, stores the vertex in the internal state, and goes through the tape to check that every other vertex is nonadjacent with the i-th vertex (otherwise M loops).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

Turing machines ⇒ Independent Set

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 $k' := (k+1)^2$ if and only if M halts in k steps.



Turing machines ⇒ INDEPENDENT SET

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 $k' := (k+1)^2$ if and only if M halts in k steps.

- G consists of $(k+1)^2$ cliques, thus a k'-independent set has to contain one vertex from each.
- The selected vertex from clique $K_{i,j}$ describes the situation before 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

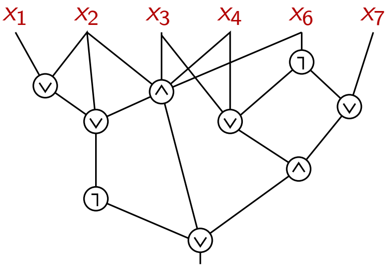
- INDEPENDENT SET and k-STEP HALTING PROBLEM can be reduced to each other $\Rightarrow$ Engineers' Hypothesis and Theorists' Hypothesis are equivalent!
- INDEPENDENT SET and k-STEP HALTING PROBLEM can be reduced to DOMINATING SET.

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

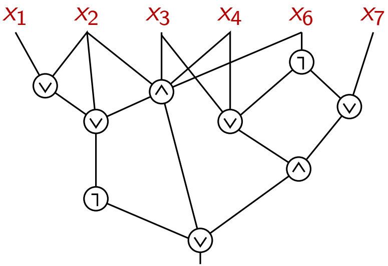
A **Boolean circuit** consists of input gates, negation gates, AND gates, OR gates, and a single output gate.



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

Boolean circuit

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



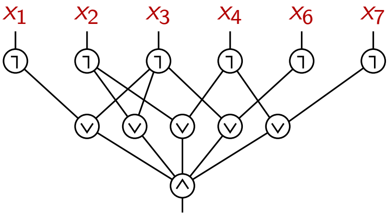
CIRCUIT SATISFIABILITY: Given a Boolean circuit C, decide if there is an assignment on the inputs of C making the output true.

Weight of an assignment: number of true values.

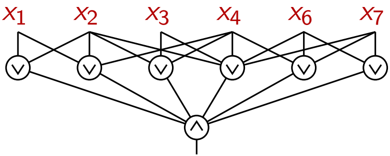
WEIGHTED CIRCUIT SATISFIABILITY: Given a Boolean circuit C and an integer k, decide if there is an assignment of weight k making the output true.

Weighted Circuit Satisfiability

Independent Set can be reduced to Weighted Circuit Satisfiability:

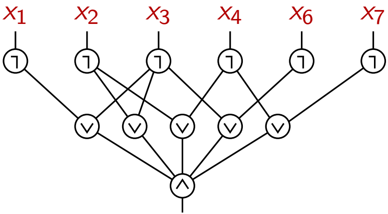


Dominating Set can be reduced to Weighted Circuit Satisfiability:

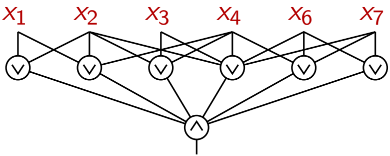


Weighted Circuit Satisfiability

Independent Set can be reduced to Weighted Circuit Satisfiability:



Dominating Set can be reduced to Weighted Circuit Satisfiability:



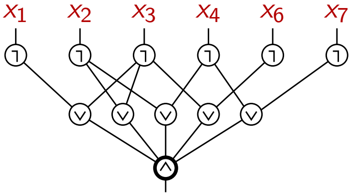
To express Dominating Set, we need more complicated circuits.

Depth and weft

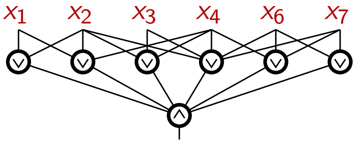
The **depth** of a circuit is the maximum length of a path from an input to the output.

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].

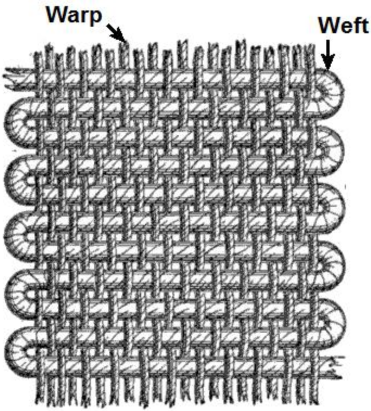
Fact: Independent Set is W[1]-complete.
Fact: Dominating Set is W[2]-complete.

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].

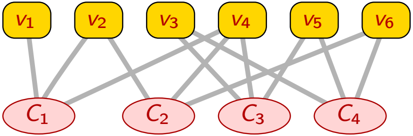
Weft



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

Parameterized reductions

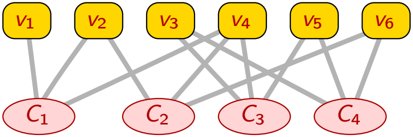
Typical NP-hardness proofs: reduction from e.g., CLIQUE or 3SAT, representing each vertex/edge/variable/clause with a gadget.



Usually does not work for parameterized reductions: cannot afford the parameter increase.

Parameterized reductions

Typical NP-hardness proofs: reduction from e.g., CLIQUE or 3SAT, representing each vertex/edge/variable/clause with a gadget.



Usually does not work for parameterized reductions: cannot afford the parameter increase.

Types of parameterized reductions:

- Reductions keeping the structure of the graph.
  - CLIQUE ⇒ INDEPENDENT SET
  - INDEPENDENT SET on regular graphs ⇒ PARTIAL VERTEX COVER
- Reductions with vertex representations.
  - MULTICOLORED INDEPENDENT SET ⇒ DOMINATING SET
- Reductions with vertex and edge representations.

Balanced Vertex Separator

Balanced Vertex Separator: Given a graph G and an integer k, find a set S of at most k vertices such that every component of $G - S$ has at most $|V(G)|/2$ vertices.

Theorem

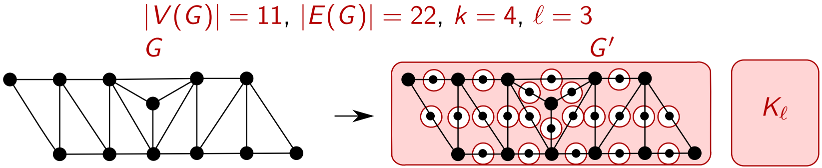
Balanced Vertex Separator parameterized by k is W[1]-hard.

Balanced Vertex Separator

Theorem

Balanced Vertex Separator parameterized by k is W[1]-hard.

Proof: By reduction from Clique.



We form G' by

- Subdividing every edge of G.
- Making the original vertices of G a clique.
- Adding an ℓ-clique for $\ell = |V(G)| + |E(G)| - 2(k + \binom{k}{2})$ (assuming the graph is sufficiently large, we have $\ell \geq 1$).

We have $|V(G')| = 2|V(G)| + 2|E(G)| - 2(k + \binom{k}{2})$ and the "big component" of G' has size $|V(G)| + |E(G)|$.

Balanced Vertex Separator

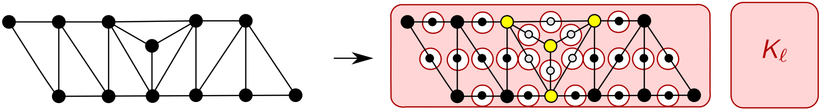
Theorem

Balanced Vertex Separator parameterized by k is W[1]-hard.

Proof: By reduction from Clique.

$|V(G)| = 11$, $|E(G)| = 22$, $k = 4$, $\ell = 3$

G $\quad\quad$ G'



We have $|V(G')| = 2|V(G)| + 2|E(G)| - 2(k + \binom{k}{2})$ and the "big component" of G' has size $|V(G)| + |E(G)|$.

$\Rightarrow$: A k-clique in G cuts away $\binom{k}{2}$ vertices, reducing the size of the big component to $|V(G)| + |E(G)| - (k + \binom{k}{2}) = |V(G')|/2$.

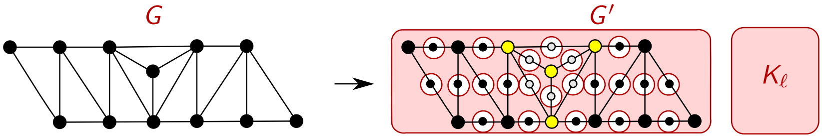
Balanced Vertex Separator

Theorem

Balanced Vertex Separator parameterized by k is W[1]-hard.

Proof: By reduction from Clique.

$|V(G)| = 11$, $|E(G)| = 22$, $k = 4$, $\ell = 3$



We have $|V(G')| = 2|V(G)| + 2|E(G)| - 2(k + \binom{k}{2})$ and the "big component" of G' has size $|V(G)| + |E(G)|$.

$\Leftarrow$: We need to reduce the size of the large component of G' by $k + \binom{k}{2}$ by removing k vertices. This is only possible if the k vertices cut away $\binom{k}{2}$ isolated vertices, i.e., the k-vertices form a k-clique in G.

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.

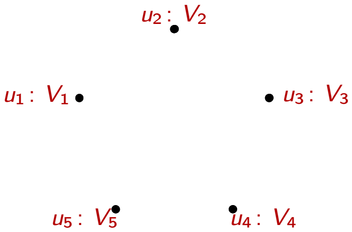
List Coloring

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 $V_1, \ldots, V_k$.
- Set C of colors: the set of vertices of G.
- The colors appearing on vertices $u_1, \ldots, u_k$ correspond to the k vertices of the clique, hence we set $L(u_i) = V_i$.



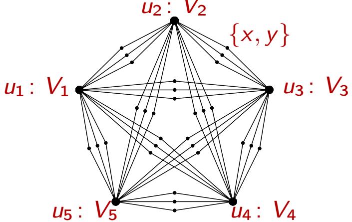
List Coloring

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 $V_1, \ldots, V_k$.
- Set C of colors: the set of vertices of G.
- The colors appearing on vertices $u_1, \ldots, u_k$ 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 $\{x, y\}$.



Vertex representation

Key idea

- Represent the k vertices of the solution with k gadgets.
- Connect the gadgets in a way that ensures that the represented values are **compatible.**

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 $F \in \mathcal{F}$.

Theorem

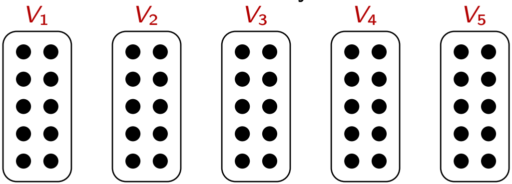
Odd Set is W[1]-hard parameterized by k.

Odd Set

Theorem

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

First try: Reduction from Multicolored Independent Set.
Let $U = V_1 \cup \ldots V_k$ and introduce each set V_i into $\mathcal{F}$.
$\Rightarrow$ The solution has to contain exactly one element from each V_i.



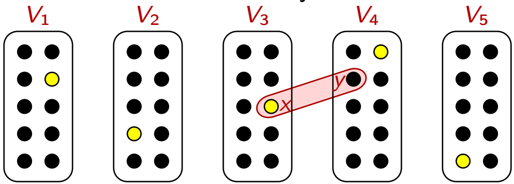
If $xy \in E(G)$, how can we express that $x \in V_i$ and $y \in V_j$ cannot be selected simultaneously?

Odd Set

Theorem

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

First try: Reduction from Multicolored Independent Set.
Let $U = V_1 \cup \ldots V_k$ and introduce each set V_i into $\mathcal{F}$.
$\Rightarrow$ The solution has to contain exactly one element from each V_i.



If $xy \in E(G)$, how can we express that $x \in V_i$ and $y \in V_j$ cannot be selected simultaneously? Seems difficult:

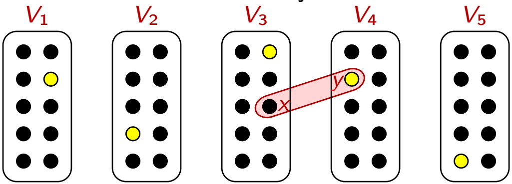
- introducing $\{x, y\}$ into $\mathcal{F}$ forces that **exactly one** of x and y appears in the solution,

Odd Set

Theorem

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

First try: Reduction from Multicolored Independent Set.
Let $U = V_1 \cup \ldots V_k$ and introduce each set V_i into $\mathcal{F}$.
$\Rightarrow$ The solution has to contain exactly one element from each V_i.



If $xy \in E(G)$, how can we express that $x \in V_i$ and $y \in V_j$ cannot be selected simultaneously? Seems difficult:

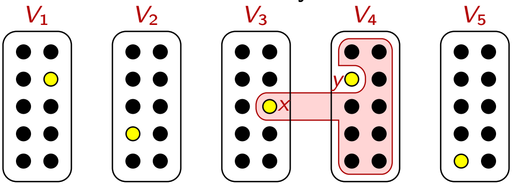
- introducing $\{x, y\}$ into $\mathcal{F}$ forces that **exactly one** of x and y appears in the solution,

Odd Set

Theorem

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

First try: Reduction from Multicolored Independent Set.
Let $U = V_1 \cup \ldots V_k$ and introduce each set V_i into $\mathcal{F}$.
$\Rightarrow$ The solution has to contain exactly one element from each V_i.



If $xy \in E(G)$, how can we express that $x \in V_i$ and $y \in V_j$ cannot be selected simultaneously? Seems difficult:

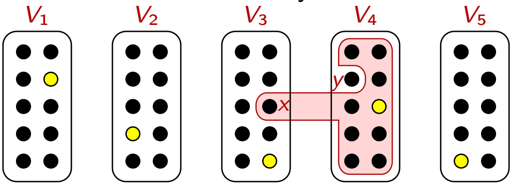
- introducing $\{x, y\}$ into $\mathcal{F}$ forces that **exactly one** of x and y appears in the solution,
- introducing $\{x\} \cup (V_j \setminus \{y\})$ into $\mathcal{F}$ forces that either **both** x and y or **none** of x and y appear in the solution.

Odd Set

Theorem

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

First try: Reduction from Multicolored Independent Set.
Let $U = V_1 \cup \ldots V_k$ and introduce each set V_i into $\mathcal{F}$.
$\Rightarrow$ The solution has to contain exactly one element from each V_i.



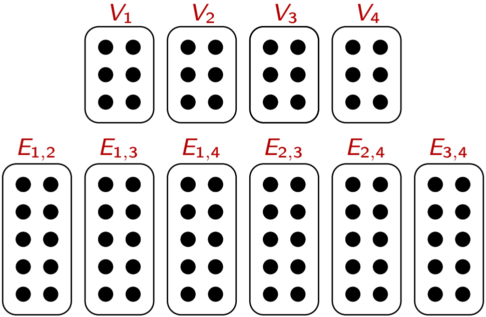
If $xy \in E(G)$, how can we express that $x \in V_i$ and $y \in V_j$ cannot be selected simultaneously? Seems difficult:

- introducing $\{x, y\}$ into $\mathcal{F}$ forces that **exactly one** of x and y appears in the solution,
- introducing $\{x\} \cup (V_j \setminus \{y\})$ into $\mathcal{F}$ forces that either **both** x and y or **none** of x and y appear in the solution.

Odd Set

Reduction from Multicolored Clique.

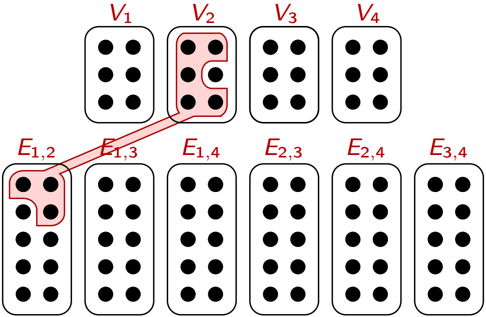
- $U := \bigcup_{i=1}^{k} V_i \cup \bigcup_{1 \le i < j \le k} E_{i,j}$.
- $k' := k + \binom{k}{2}$.
- Let $\mathcal{F}$ contain V_i $(1 \le i \le k)$ and $E_{i,j}$ $(1 \le i < j \le k)$.



Odd Set

Reduction from Multicolored Clique.

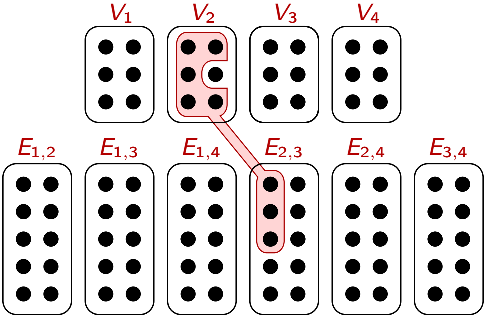
- $U := \bigcup_{i=1}^{k} V_i \cup \bigcup_{1 \le i < j \le k} E_{i,j}$.
- $k' := k + \binom{k}{2}$.
- Let $\mathcal{F}$ contain V_i $(1 \le i \le k)$ and $E_{i,j}$ $(1 \le i < j \le k)$.
- For every $v \in V_i$ and $x \neq i$, we introduce the sets:
  $(V_i \setminus \{v\}) \cup \{\text{every edge from } E_{i,x} \text{ with endpoint } v\}$
  $(V_i \setminus \{v\}) \cup \{\text{every edge from } E_{x,i} \text{ with endpoint } v\}$



Odd Set

Reduction from Multicolored Clique.

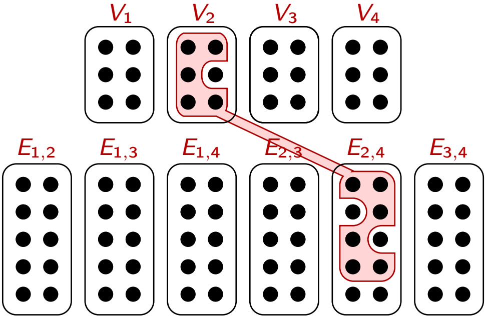
- $U := \bigcup_{i=1}^{k} V_i \cup \bigcup_{1 \le i < j \le k} E_{i,j}$.
- $k' := k + \binom{k}{2}$.
- Let $\mathcal{F}$ contain V_i ($1 \le i \le k$) and $E_{i,j}$ ($1 \le i < j \le k$).
- For every $v \in V_i$ and $x \neq i$, we introduce the sets:
  $(V_i \setminus \{v\}) \cup \{\text{every edge from } E_{i,x} \text{ with endpoint } v\}$
  $(V_i \setminus \{v\}) \cup \{\text{every edge from } E_{x,i} \text{ with endpoint } v\}$



Odd Set

Reduction from Multicolored Clique.

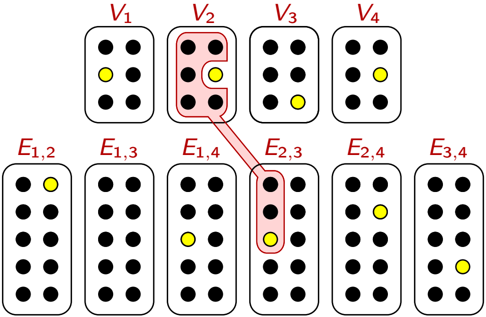
- $U := \bigcup_{i=1}^{k} V_i \cup \bigcup_{1 \le i < j \le k} E_{i,j}$.
- $k' := k + \binom{k}{2}$.
- Let $\mathcal{F}$ contain V_i $(1 \le i \le k)$ and $E_{i,j}$ $(1 \le i < j \le k)$.
- For every $v \in V_i$ and $x \neq i$, we introduce the sets:
  $(V_i \setminus \{v\}) \cup \{\text{every edge from } E_{i,x} \text{ with endpoint } v\}$
  $(V_i \setminus \{v\}) \cup \{\text{every edge from } E_{x,i} \text{ with endpoint } v\}$



Odd Set

Reduction from Multicolored Clique.

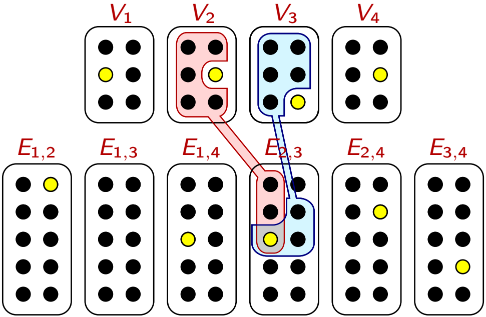
- For every $v \in V_i$ and $x \neq i$, we introduce the sets:
  $(V_i \setminus \{v\}) \cup \{\text{every edge from } E_{i,x} \text{ with endpoint } v\}$
  $(V_i \setminus \{v\}) \cup \{\text{every edge from } E_{x,i} \text{ with endpoint } v\}$
- $v \in V_i$ selected $\iff$ edges with endpoint v are selected from $E_{i,x}$ and $E_{x,i}$



Odd Set

Reduction from Multicolored Clique.

- For every $v \in V_i$ and $x \neq i$, we introduce the sets:
  $(V_i \setminus \{v\}) \cup \{\text{every edge from } E_{i,x} \text{ with endpoint } v\}$
  $(V_i \setminus \{v\}) \cup \{\text{every edge from } E_{x,i} \text{ with endpoint } v\}$
- $v \in V_i$ selected $\iff$ edges with endpoint v are selected from $E_{i,x}$ and $E_{x,i}$
- $v_i \in V_i$ selected, $v_j \in V_j$ selected $\iff$ edge $v_i v_j$ is selected in $E_{i,x}$



Vertex and edge representation

Key idea

- Represent the vertices of the clique by k gadgets.
- Represent the edges of the clique by $\binom{k}{2}$ gadgets.
- Connect edge gadget $E_{i,j}$ to vertex gadgets V_i and V_j such that if $E_{i,j}$ represents the edge between $x \in V_i$ and $y \in V_j$, then it **forces** V_i to x and V_j to y.

Variants of ODD SET

The following problems are W[1]-hard:

- ODD SET
- EXACT ODD SET (find a set of size exactly k ...)
- EXACT EVEN SET
- UNIQUE HITTING SET
  (at most k elements that hit each set exactly once)
- EXACT UNIQUE HITTING SET
  (exactly k elements that hit each set exactly once)

Variants of Odd Set

The following problems are W[1]-hard:

- Odd Set
- Exact Odd Set (find a set of size exactly k ...)
- Exact Even Set
- Unique Hitting Set
  (at most k elements that hit each set exactly once)
- Exact Unique Hitting Set
  (exactly k elements that hit each set exactly once)

Open question:

? Even Set: Given a set system $\mathcal{F}$ and an integer k, find a **nonempty** set S of at most k elements such $|F \cap S|$ is even for every $F \in \mathcal{F}$.

Summary

- By parameterized reductions, we can show that lots of parameterized problems are at least as hard as CLIQUE, hence unlikely to be fixed-parameter tractable.
- Connection with Turing machines gives some supporting evidence for hardness (only of theoretical interest).
- The W-hierarchy classifies the problems according to hardness (only of theoretical interest).
- Important trick in W[1]-hardness proofs: vertex and edge representations.

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 $\mathrm{3SAT}$.

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

Note: an n-variable $\mathrm{3SAT}$ 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 $\mathrm{3SAT}$.

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

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

Sparsification Lemma [Impagliazzo, Paturi, Zane 2001]

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

$$\Updownarrow$$

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

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:

$$\boxed{\begin{array}{c}\text{3SAT formula } \phi \\ n \text{ variables} \\ m \text{ clauses}\end{array}} \Rightarrow \boxed{\begin{array}{c}\text{Graph } G \\ O(n+m) \text{ vertices} \\ O(n+m) \text{ edges}\end{array}}$$

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:

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

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.

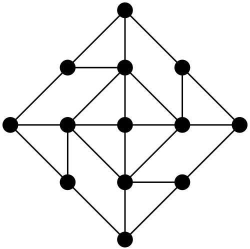
Consequence: Assuming ETH, there is no $2^{o(k)} \cdot n^{O(1)}$ time algorithm for

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

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:

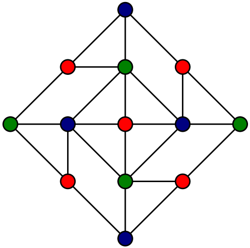


- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to the whole gadget.
- If two edges cross, replace them with a crossover gadget.

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:

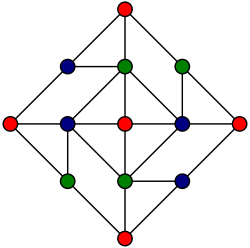


- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to the whole gadget.
- If two edges cross, replace them with a crossover gadget.

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		Graph G		Planar graph G'
n variables	$\Rightarrow$	$O(m)$ vertices	$\Rightarrow$	$O(m^2)$ 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.

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 $f(k) \cdot n^{O(1)}$.

$$\Updownarrow$$

Theorists' Hypothesis

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

$$\Uparrow$$

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 $f(k) \cdot n^{O(1)}$.

$$\Updownarrow$$

Theorists' Hypothesis

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

$$\Uparrow$$

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 $f(k) \cdot n^{O(1)}$.

$$\Updownarrow$$

Theorists' Hypothesis

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

$$\Uparrow$$

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

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

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

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

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 $f(k) \cdot |V(H)|^{k/s(k)} \leq n \cdot (k3^{n/k})^{k/s(k)} = n \cdot k^{k/s(k)} \cdot 3^{n/s(k)} = 2^{o(n)}$.

Tight bounds

Theorem

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

Transfering to other problems:

$$\begin{array}{ccc}
\boxed{\begin{array}{c} k\text{-Clique} \\ (x, k) \end{array}} & \Rightarrow & \boxed{\begin{array}{c} \text{Problem } A \\ (x', O(k)) \end{array}} \\
\boxed{\begin{array}{c} f(k) \cdot n^{o(k)} \\ \text{algorithm} \end{array}} & \Leftarrow & \boxed{\begin{array}{c} f(k) \cdot n^{o(k))} \\ \text{algorithm} \end{array}}
\end{array}$$

Tight bounds

Theorem

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

Transfering to other problems:

k-Clique (x, k)	$\Rightarrow$	Problem A $(x', g(k))$
$f(k) \cdot n^{o(k)}$ algorithm	$\Leftarrow$	$f(k) \cdot n^{o(g^{-1}(k))}$ algorithm

Tight bounds

Theorem

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

Transfering to other problems:

k-Clique (x, k)	$\Rightarrow$	Problem A (x', k^2)
$f(k) \cdot n^{o(k)}$ algorithm	$\Leftarrow$	$f(k) \cdot n^{o(\sqrt{k})}$ algorithm

Tight bounds

Theorem

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

Transfering to other problems:

k-Clique (x, k)	$\Rightarrow$	Problem A (x', k^2)
$f(k) \cdot n^{o(k)}$ algorithm	$\Leftarrow$	$f(k) \cdot n^{o(\sqrt{k})}$ algorithm

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.

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.

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 $v_i \ldots$

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_i, v_j)	$(., v_i)$
	$(v_i, .)$		$(v_j, .)$	
$(., v_j)$	(v_j, v_i)	$(., v_j)$	(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:

- 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.

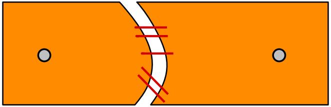
Proof is similar to the reduction from Multicolored Clique to List Coloring, but now the resulting graph is planar.

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]

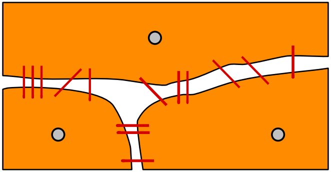
A minimum $s - t$ cut can be found in polynomial time.

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



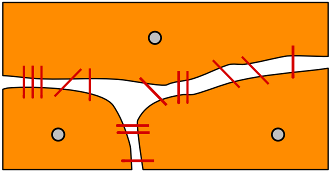
Theorem

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

Planar k-Terminal Cut can be solved in time $2^{O(k)} \cdot n^{O(\sqrt{k})}$.

Lower bounds

Theorem

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

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

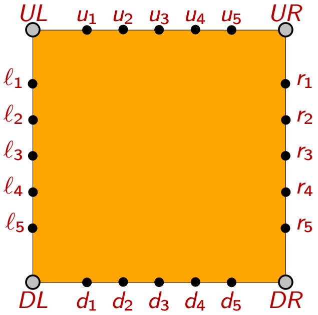
PLANAR k-TERMINAL CUT is W[1]-hard and has no $f(k) \cdot n^{o(\sqrt{k})}$ time algorithm (assuming ETH).

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.



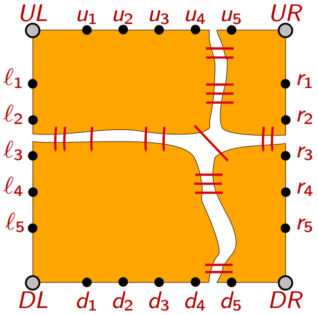
The gadget.

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.



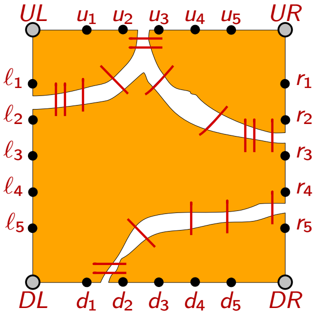
A cut representing $(4, 2)$.

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

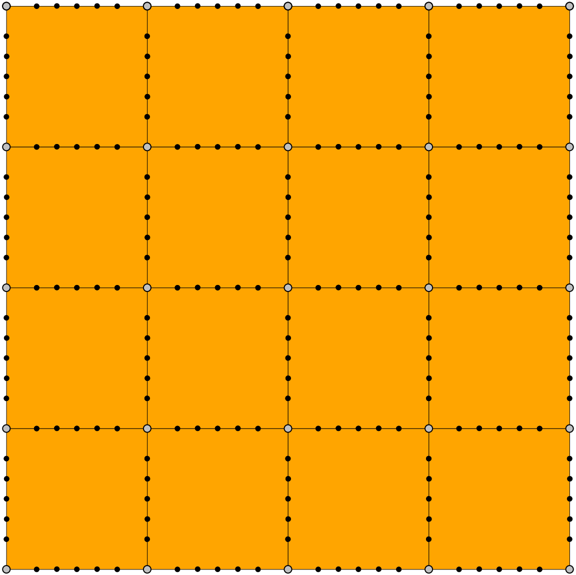
- for every $(x, y) \in S_{i,j}$, there is a minimum multiway cut that represents (x, y).
- every minimum multiway cut represents some $(x, y) \in S_{i,j}$.

Main part of the proof: constructing these gadgets.

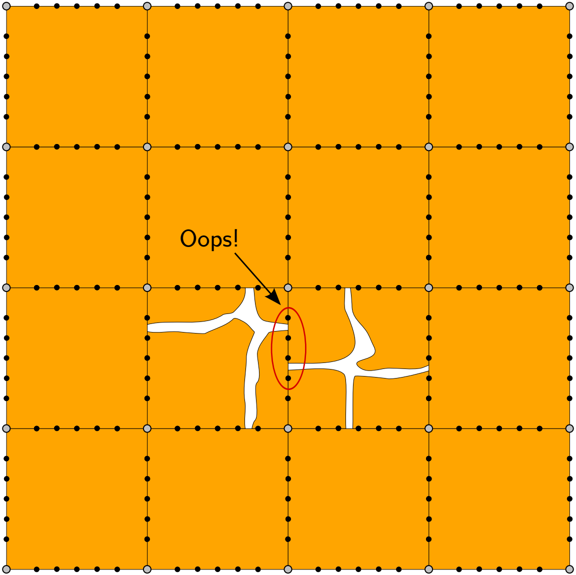


A cut not representing any pair.

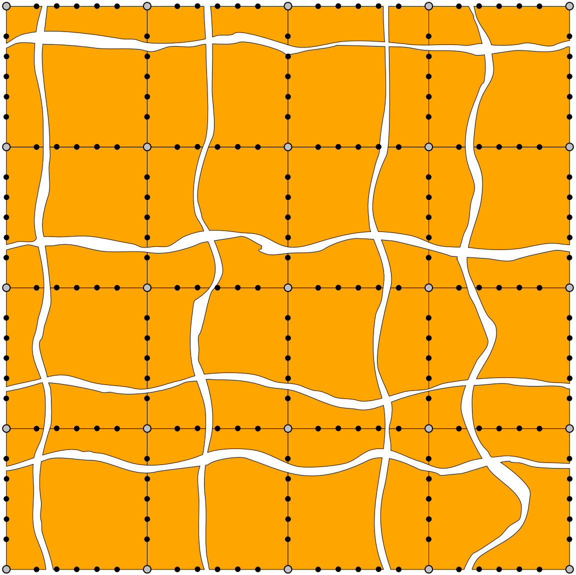
Putting together the gadgets



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.

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 $s_{i,j+1}$.

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:

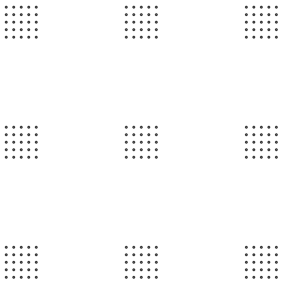
Theorem

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)

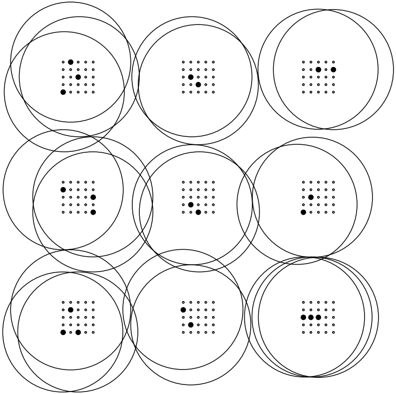


Every pair is represented by a unit disk in the plane.

$\leq$ 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)

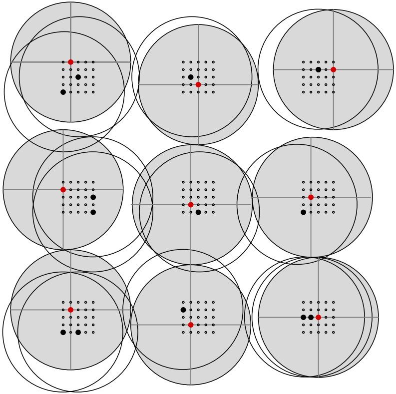


Every pair is represented by a unit disk in the plane.

$\leq$ 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.

Approximation schemes

Polynomial-time approximation scheme (PTAS)

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

- **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.