Complexity of parameterized problems

Dániel Marx

Lecture #3
May 22, 2020
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., \textsc{Clique}) is not FPT?
- Can we show that a problem (e.g., \textsc{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?
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., \textsc{Clique}) is \textbf{not} FPT?
- Can we show that a problem (e.g., \textsc{Vertex Cover}) has \textbf{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., $k$-\textsc{Clique} is polynomial-time solvable, hence FPT.

Can we give some evidence for negative results?
Classical complexity — reminder

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

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.

Polynomial-time reduction from problem $P$ to problem $Q$: a function $\phi$ with the following properties:
- $\phi(x)$ is a yes-instance of $Q \iff x$ is a yes-instance of $P$,
- $\phi(x)$ can be computed in time $|x| \leq O(1)$.

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., $P = NP$).
Classical complexity — reminder

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

**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.

**Polynomial-time reduction** from problem $P$ to problem $Q$: a function $\phi$ with the following properties:
- $\phi(x)$ is a yes-instance of $Q \iff x$ is a yes-instance of $P$,
- $\phi(x)$ can be computed in time $|x|^{O(1)}$.

**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., $P = NP$).
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.

**Fact:** Graph $G$ has an independent set $k \iff G$ has a vertex cover of size $n - k$.

This is a correct polynomial-time reduction.

However, $\text{Vertex Cover}$ is FPT, but $\text{Independent Set}$ is not known to be FPT.
Parameterized reductions

Definition

Parameterized reduction from problem $A$ to problem $B$: a function $\phi$ with the following properties:

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

Definition

Parameterized reduction from problem $A$ to problem $B$: a function $\phi$ with the following properties:

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

Theorem

If there is a parameterized reduction from problem $A$ to problem $B$ and $B$ is FPT, then $A$ is also FPT.

Intuitively: Reduction $A \rightarrow B +$ algorithm for $B$ gives and algorithm for $A$. 
Parameterized reductions

Definition

Parameterized reduction from problem $A$ to problem $B$: a function $\phi$ with the following properties:

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

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.
Parameterized reductions

**Theorem**

If there is a parameterized reduction from problem $A$ to problem $B$ and $B$ is FPT, then $A$ is also FPT.

**Proof:** Suppose that

- the reduction has running time $f(k)n^{c_1}$,
- the reduction creates an instance with parameter at most $g(k)$, and
- $B$ can be solved in time $h(k)n^{c_2}$.

Then running the reduction an solving the created instance of $B$ gives an algorithm for $A$ with running time

$$f(k)n^{c_1} + h(g(k)) \cdot (f(k)n^{c_1})^{c_2} \leq f'(k)n^{c_1c_2}$$

for some function $f'$. 
**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**.
**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 $G'$. 

\[ G \quad \iff \quad G' \]
Theorem

There is a parameterized reduction from \textit{Clique} to \textit{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 \textit{Multicolored Independent Set}. 

\textbf{Multicolored Clique}
**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 $y_i$. 
**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**.
Basic hypotheses

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

**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)**

A $n$-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 $P \neq NP$ – we have to assume something stronger.

**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 $2^{o(n)}$.
Basic hypotheses

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

**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 $P \neq NP$ – we have to assume something stronger.

<table>
<thead>
<tr>
<th>Engineers’ Hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$-CLIQUE cannot be solved in time $f(k) \cdot n^{O(1)}$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Theorists’ Hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$-STEP HALTING PROBLEM (is there a path of the given NTM that stops in $k$ steps?) cannot be solved in time $f(k) \cdot n^{O(1)}$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exponential Time Hypothesis (ETH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$-variable 3SAT cannot be solved in time $2^{o(n)}$.</td>
</tr>
</tbody>
</table>

Which hypothesis is the most plausible?
Summary

- **Independent Set** and **k-Step Halting Problem** can be reduced to each other ⇒ Engineers’ Hypothesis and Theorists’ Hypothesis are equivalent!

- **Independent Set** and **k-Step Halting Problem** can be reduced to **Dominating 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!
Summary

- **Independent Set** and **k-Step Halting Problem** can be reduced to each other ⇒ 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!
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:

\[
x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_6 \quad x_7
\]

Dominating Set can be reduced to Weighted Circuit Satisfiability:

\[
x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_6 \quad x_7
\]
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 $\text{FPT} = W[1]$ and every problem in $W[1]$ is FPT.


$\Rightarrow$ If there is a parameterized reduction from Dominating Set to Independent Set, then $W[1] = W[2]$. 
Weft is a term related to weaving cloth: it is the thread that runs from side to side in the fabric.
Parameterized reductions

Typical \textbf{NP}-hardness proofs: reduction from e.g., \textsc{Clique} or \textsc{3SAT}, representing each vertex/edge/variable/clause with a gadget.

\begin{center}
\begin{tikzpicture}
\node[draw=yellow,fill=yellow,shape=circle,minimum size=1cm] (v1) at (0,0) {$v_1$};
\node[draw=yellow,fill=yellow,shape=circle,minimum size=1cm] (v2) at (1,0) {$v_2$};
\node[draw=yellow,fill=yellow,shape=circle,minimum size=1cm] (v3) at (2,0) {$v_3$};
\node[draw=yellow,fill=yellow,shape=circle,minimum size=1cm] (v4) at (3,0) {$v_4$};
\node[draw=yellow,fill=yellow,shape=circle,minimum size=1cm] (v5) at (4,0) {$v_5$};
\node[draw=yellow,fill=yellow,shape=circle,minimum size=1cm] (v6) at (5,0) {$v_6$};
\node[draw=pink,fill=pink,shape=ellipse,minimum size=1cm] (c1) at (1,-1) {$C_1$};
\node[draw=pink,fill=pink,shape=ellipse,minimum size=1cm] (c2) at (2,-1) {$C_2$};
\node[draw=pink,fill=pink,shape=ellipse,minimum size=1cm] (c3) at (3,-1) {$C_3$};
\node[draw=pink,fill=pink,shape=ellipse,minimum size=1cm] (c4) at (4,-1) {$C_4$};
\foreach \i in {1,...,6}
\foreach \j in {1,...,4}
\draw[gray] (\i) -- (\j);
\end{tikzpicture}
\end{center}

Usually doesn’t work for parameterized reduction: 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 doesn’t work for parameterized reduction: cannot afford the parameter increase.

Types of parameterized reductions:
- Reductions keeping the structure of the graph.
  - Clique $\Rightarrow$ Independent Set
- Reductions with vertex representations.
  - Multicolored Independent Set $\Rightarrow$ Dominating Set
- Reductions with vertex and edge representations.
**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 \( k \).
Theorem

**Odd Set** is \(W[1]\)-hard parameterized by \(k\).

**First try:** Reduction from **Multicolored Independent Set**. Let \(U = V_1 \cup \ldots \cup V_k\) and introduce each set \(V_i\) into \(F\).

⇒ 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?
**Theorem**

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

**First try:** Reduction from **Multicolored Independent Set**. Let $U = V_1 \cup \ldots \cup V_k$ and introduce each set $V_i$ into $F$.

⇒ 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 $F$ 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 \( U = V_1 \cup \ldots \cup V_k \) 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 \( F \).

⇒ 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 \( F \) forces that exactly one of \( x \) and \( y \) appears in the solution,
- introducing \( \{x\} \cup (V_j \setminus \{y\}) \) into \( F \) 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 $U = V_1 \cup \ldots \cup V_k$ 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.

1. \( U := \bigcup_{i=1}^{k} V_i \cup \bigcup_{1 \leq i < j \leq k} E_{i,j} \).
2. \( k' := k + \binom{k}{2} \).
3. Let \( F \) contain \( V_i \ (1 \leq i \leq k) \) and \( E_{i,j} \ (1 \leq i < j \leq k) \).
Odd Set

Reduction from Multicolored Clique.

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

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

- \( U := \bigcup_{i=1}^{k} V_i \cup \bigcup_{1 \leq i < j \leq k} E_{i,j} \).
- \( k' := k + \binom{k}{2} \).
- Let \( \mathcal{F} \) contain \( V_i \) \( (1 \leq i \leq k) \) and \( E_{i,j} \) \( (1 \leq i < j \leq k) \).
- 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 $\iff$ edge $v_i v_j$ is selected in $E_{i,x}$
- $v_j \in V_j$ selected

![Diagram showing the reduction process]
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 Hitting Set

The following problems are $W[1]$-hard, with very similar proofs:

- **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 **Hitting Set**

The following problems are $\text{W}[1]$-hard, with very similar proofs:

- **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)

A problem that is also $\text{W}[1]$-hard, but requires very different techniques:

- **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.
Shift of focus

qualitative question

FPT or W[1]-hard?
Shift of focus

FPT or W[1]-hard?

qualitative question

What is the best possible multiplier \( f(k) \) in the running time \( f(k) \cdot n^{O(1)} \)?

quantitative question

FPT

What is the best possible exponent \( g(k) \) in the running time \( f(k) \cdot n^{g(k)} \)?

W[1]-hard

\( 2^k \), \( 1.0001^k \), \( 2^{\sqrt{k}} \)

\( n^{O(k)} \), \( n^{\log k} \), \( n^{\log \log k} \)
Better algorithms for \textsc{Vertex Cover}

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

⇒ We can hope only for conditional lower bounds.
We have seen a $2^k \cdot n^{O(1)}$ time algorithm.

Easy to improve to, e.g., $1.618^k \cdot n^{O(1)}$.

Current best $f(k): 1.2738^k \cdot n^{O(1)}$.

Lower bounds?

- Is, say, $1.001^k \cdot n^{O(1)}$ time possible?
- Is $2^k/\log k \cdot n^{O(1)}$ time possible?

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

$\Rightarrow$ We can hope only for conditional lower bounds.
Exponential Time Hypothesis (ETH)

3CNF: $\phi$ is a conjunction of clauses, where each clause is a disjunction of at most 3 literals (= a variable or its negation), e.g., $(x_1 \lor x_3 \lor \overline{x}_4) \land (\overline{x}_2 \lor x_3) \lor (x_1 \lor x_2 \lor x_4)$.

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

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

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

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

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

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

**Exponential Time Hypothesis (ETH) [consequence of]**

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

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

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

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

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

Exponential Time Hypothesis (ETH) [real statement]

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

**Exponential Time Hypothesis (ETH) [consequence of]**

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

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

Are there algorithms that are subexponential in the size $n + m$ of the 3SAT formula?
Sparsification

**Exponential Time Hypothesis (ETH) [consequence of]**

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

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

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

**Sparsification Lemma**

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

\[\Leftrightarrow\]

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

**Intuitively:** When considering a hard 3SAT instance, we can assume that it has $m = O(n)$ clauses.
Lower bounds based on ETH

**Exponential Time Hypothesis (ETH) + Sparsification Lemma**

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

The textbook reduction from 3SAT to Vertex Cover:

\[ x_1 \overline{x_1} x_2 \overline{x_2} x_3 \overline{x_3} x_4 \overline{x_4} \]

\[ \begin{align*}
&\quad \triangle \\
\quad \triangle & \quad \triangle & \quad \triangle & \quad \triangle & \quad \triangle & \quad \triangle \\
\end{align*} \]
Lower bounds based on ETH

**Exponential Time Hypothesis (ETH) + Sparsification Lemma**

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

The textbook reduction from 3SAT to Vertex Cover:

formula is satisfiable $\Leftrightarrow$ there is a vertex cover of size $n + 2m$
Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma
There is no \(2^{o(n+m)}\)-time algorithm for \(n\)-variable \(m\)-clause 3SAT.

The textbook reduction from 3SAT to Vertex Cover:

3SAT formula \(\phi\)
- \(n\) variables
- \(m\) clauses

\[\Rightarrow\]

Graph \(G\)
- \(O(n + m)\) vertices
- \(O(n + m)\) edges
Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma

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

The textbook reduction from 3SAT to Vertex Cover:

3SAT formula $\phi$
- $n$ variables
- $m$ clauses

⇒
Graph $G$
- $O(n + m)$ vertices
- $O(n + m)$ edges

Corollary

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

Exponential Time Hypothesis (ETH) + Sparsification Lemma

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

The textbook reduction from $3$SAT to Vertex Cover:

- **3SAT formula $\phi$**
  - $n$ variables
  - $m$ clauses

- **Graph $G$**
  - $O(n + m)$ vertices
  - $O(n + m)$ edges

Corollary

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

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

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

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

Seems to be the natural behavior of FPT problems?
The race for better FPT algorithms

Double exponential

Tower of exponentials

"Slightly super-exponential"

Single exponential

Subexponential
**Edge Clique Cover**

**Edge Clique Cover**: Given a graph $G$ and an integer $k$, cover the edges of $G$ with at most $k$ cliques. (the cliques need not be edge disjoint)

Equivalently: can $G$ be represented as an intersection graph over a $k$ element universe?
**Edge Clique Cover**

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

(the cliques need not be edge disjoint)

**Equivalently:** can $G$ be represented as an intersection graph over a $k$ element universe?
**Edge Clique Cover**

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

(the cliques need not be edge disjoint)

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

5 cliques
Edge Clique Cover

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

(the cliques need not be edge disjoint)

**Simple algorithm (sketch)**

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

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

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

(the cliques need not be edge disjoint)

Double-exponential dependence on $k$ cannot be avoided!

**Theorem**

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

**Proof:**

3SAT  
$n$ variables  \[\rightarrow\]  
**Edge Clique Cover**  
$k = O(\log n)$
The race for better FPT algorithms

Double exponential

"Slightly super-exponential"

Single exponential

Subexponential

Tower of exponentials
Slightly superexponential algorithms

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

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

   **Example:** Feedback Vertex Set in the first lecture.

2. Trying $k! = 2^{O(k \log k)}$ permutations of $k$ elements (or partitions, matchings, . . .)

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

Given strings $s_1, \ldots, s_k$ of length $L$ over alphabet $\Sigma$, and an integer $d$, find a string $s$ (of length $L$) such that Hamming distance $d(s, s_i) \leq d$ for every $1 \leq i \leq k$.

(Hamming distance: number of differing positions)

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>C</th>
<th>B</th>
<th>D</th>
<th>C</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>B</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_2$</td>
<td>A</td>
<td>B</td>
<td>D</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>D</td>
<td>B</td>
</tr>
<tr>
<td>$s_3$</td>
<td>C</td>
<td>D</td>
<td>D</td>
<td>B</td>
<td>A</td>
<td>C</td>
<td>C</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>$s_4$</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>C</td>
<td>C</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>$s_5$</td>
<td>A</td>
<td>C</td>
<td>D</td>
<td>B</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>
**Closest String**

Given strings $s_1, \ldots, s_k$ of length $L$ over alphabet $\Sigma$, and an integer $d$, find a string $s$ (of length $L$) such that Hamming distance $d(s, s_i) \leq d$ for every $1 \leq i \leq k$.

(Hamming distance: number of differing positions)

<table>
<thead>
<tr>
<th></th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$s_4$</th>
<th>$s_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>A</td>
<td>C</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>B</td>
<td>D</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>C</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>A</td>
<td>C</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>D</td>
<td>B</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>D</td>
<td>B</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>B</td>
<td>D</td>
<td>C</td>
<td>B</td>
</tr>
</tbody>
</table>

We can ask for running time for example $f(d) \in O(1)$: FPT parameterized by $d$ $f(k, |\Sigma|) \in O(1)$: FPT with combined parameters $k$ and $|\Sigma|$.
Closest String

Given strings $s_1, \ldots, s_k$ of length $L$ over alphabet $\Sigma$, and an integer $d$, find a string $s$ (of length $L$) such that Hamming distance $d(s, s_i) \leq d$ for every $1 \leq i \leq k$.

(Hamming distance: number of differing positions)

Different parameters:
- Number $k$ of strings.
- Length $L$ of strings
- Maximum distance $d$.
- Alphabet size $|\Sigma|$.
Given strings $s_1, \ldots, s_k$ of length $L$ over alphabet $\Sigma$, and an integer $d$, find a string $s$ (of length $L$) such that Hamming distance $d(s, s_i) \leq d$ for every $1 \leq i \leq k$.

(Hamming distance: number of differing positions)

Different parameters:
- Number $k$ of strings.
- Length $L$ of strings
- Maximum distance $d$.
- Alphabet size $|\Sigma|$.

We can ask for running time for example
- $f(d)n^{O(1)}$: FPT parameterized by $d$
- $f(k, |\Sigma|)n^{O(1)}$: FPT with combined parameters $k$ and $|\Sigma|$
Closest String

Theorem

**Closest String** can be solved in time $2^{O(d \log d)} n^{O(1)}$.

- **Main idea:** Given a string $y$ at Hamming distance $\ell$ from some solution, we use branching to find a string at distance at most $\ell - 1$ from some solution.

- Initially, $y = x_1$ is at distance at most $d$ from some solution.
**Closest String**

**Theorem**

*Closest String* can be solved in time $2^{O(d \log d)} n^{O(1)}$.

- **Main idea:** Given a string $y$ at Hamming distance $\ell$ from some solution, we use branching to find a string at distance at most $\ell - 1$ from some solution.
- Initially, $y = x_1$ is at distance at most $d$ from some solution.
- If $y$ is not a solution, then there is an $x_i$ with $d(y, x_i) \geq d + 1$.
  - Look at the first $d + 1$ positions $p$ where $x_i[p] \neq y[p]$. For every solution $z$, it is true for one such $p$ that $x_i[p] = z[p]$.
  - Branch on choosing one of these $d + 1$ positions and replace $y[p]$ with $x_i[p]$: distance of $y$ from solution $z$ decreases to $\ell - 1$.
- Running time $(d + 1)^d \cdot n^{O(1)} = 2^{O(d \log d)} n^{O(1)}$. 
Theorem
Assuming ETH, \texttt{Closest String} has no $2^{o(d \log d)} n^{O(1)}$ algorithm.

Proof:

\begin{align*}
\text{3SAT} \\
O(d \log d) \text{ variables} & \quad \Rightarrow \quad \text{Closest String} \\
& \quad \text{distance } d
\end{align*}
Shift of focus

FPT or W[1]-hard?

What is the best possible multiplier $f(k)$ in the running time $f(k) \cdot n^{O(1)}$?

What is the best possible exponent $g(k)$ in the running time $f(k) \cdot n^{g(k)}$?

- $2^k$?
- $1.0001^k$?
- $2^{\sqrt{k}}$?
- $n^{O(k)}$?
- $n^{\log k}$?
- $n^{\log \log k}$?
Better algorithms for W[1]-hard problems

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

\[ n^{\sqrt{k}} \]
\[ n^{k/\log \log k} \]
\[ 2^k \cdot n^{\log \log \log k} \]
Better algorithms for W[1]-hard problems

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

**Theorem**

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

In particular, ETH implies that \(k\text{-Clique}\) is not FPT.
### Basic hypotheses

#### 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 $2^{o(n)}$. 
Lower bound for \( k\)-CLIQUE

**Theorem**

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

**Proof:**

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

\[
\begin{align*}
\text{3-COLORING} & \quad \quad \quad \quad \text{CLIQUE} \\
\text{n vertices} & \quad \quad \quad \quad (G, k) \text{ with} \\
& \quad \quad \quad \quad N \approx \frac{3^n}{k} \text{ vertices} \\
\end{align*}
\]

\( N^{o(k)} \) algorithm for CLIQUE \( \Rightarrow (3^{n/k})^{o(k)} = 3^{o(n)} \) algorithm for 3-COLORING
Lower bound for $k$-CLIQUE

Theorem

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

Proof:

Create a vertex per each consistent coloring of each group.
Theorem

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

Proof:

Create a vertex per each consistent coloring of each group.
Lower bound for \(k\)-CLIQUE

**Theorem**

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

**Proof:**

Create a vertex per each consistent coloring of each group.
Lower bound for $k$-CLIQUE

**Theorem**

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

**Proof:**

Connect two vertices if they represent colorings that are consistent together.
Lower bound for $k$-CLIQUE

**Theorem**

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

**Proof:**

Connect two vertices if they represent colorings that are consistent together.
Lower bound for $k$-CLIQUE

Theorem

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

Proof:

Left graph has a 3-coloring $\iff$ Right graph contains a $k$-clique
Lower bound for $k$-CLIQUE

**Theorem**

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

**Proof:**

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

$$2^k \cdot N^{o(k)} = n \cdot (\log n)^{o(\log n)} \cdot 3^{n \cdot o(\log n) / \log n} = 2^{o(n)}.$$
Lower bound for $k$-CLIQUE

**Theorem**

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

**Proof:**

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

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

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

**Theorem**

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

Transfering to other problems:

- $k$-CLIQUE $(x, k)$
- $f(k) \cdot n^{o(k)}$ algorithm

$\Rightarrow$

- Problem $A (x', O(k))$
- $f(k) \cdot n^{o(k)}$ 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

**Theorem**

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

Transfering to other problems:

\[
\begin{align*}
\text{\( k\text{-CLIQUE} \)} & \quad \Rightarrow \quad \text{Problem } A \\
(x, k) & \quad \Rightarrow \quad (x', k^2) \\
&& \quad \Leftrightarrow \quad f(k) \cdot n^{o(k)} \text{ algorithm} \\
&& \quad \Leftrightarrow \quad f(k) \cdot n^{o(\sqrt{k})} \text{ algorithm}
\end{align*}
\]
Tight bounds

**Theorem**

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

Transfering to other problems:

\[
\begin{align*}
k\text{-CLIQUE} & \quad \Rightarrow \\
(x, k) & \quad \text{Problem } A \\
f(k) \cdot n^{o(k)} & \quad (x', g(k)) \\
\text{algorithm} & \quad f(k) \cdot n^{o(g^{-1}(k))} \\
& \quad \text{algorithm}
\end{align*}
\]
Tight bounds

**Theorem**

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

Transfering to other problems:

\[
\begin{align*}
\text{\textsc{k-Clique}} (x,k) & \quad \Rightarrow \quad \text{Problem $A$} (x', g(k)) \\
 f(k) \cdot n^{o(k)} \quad \text{algorithm} & \quad \Leftrightarrow \quad f(k) \cdot n^{o(g^{-1}(k))} \quad \text{algorithm}
\end{align*}
\]

**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
- ...
Parameterized reductions from \textit{Clique} or \textit{Independent Set} can give evidence that a problem is not FPT.

ETH can give tight bounds on the $f(k)$ for FPT problems.

ETH can give tight bounds on the exponent of $n$ for W[1]-hard problems.