Fine-Grained Complexity and Algorithm Design Boot Camp

Lower Bounds Based on ETH

Dániel Marx

Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary

Simons Institute, Berkeley, CA
September 3, 2015
Exponential Time Hypothesis (ETH)

Hypothesis introduced by Impagliazzo, Paturi, and Zane:

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

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

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

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

Hypothesis introduced by Impagliazzo, Paturi, and Zane:

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

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

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

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

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

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

$\Leftrightarrow$

There is a $2^{o(m)}$-time algorithm for $m$-clause 3SAT.
Lower bounds for exact and parameterized problems
Lower bounds based on ETH

Exponential Time Hypothesis (ETH)

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

The textbook reduction from 3SAT to 3-COLORING:

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

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

Corollary

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

Exponential Time Hypothesis (ETH)

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

The textbook reduction from 3SAT to 3-Coloring:

<table>
<thead>
<tr>
<th>3SAT formula $\phi$</th>
<th>Graph $G$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$ variables</td>
<td>$O(m)$ vertices</td>
</tr>
<tr>
<td>$m$ clauses</td>
<td>$O(m)$ edges</td>
</tr>
</tbody>
</table>

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, \textbf{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

- \textbf{Independent Set}
- \textbf{Clique}
- \textbf{Dominating Set}
- \textbf{Vertex Cover}
- \textbf{Hamiltonian Path}
- \textbf{Feedback Vertex Set}
- \textellipsis
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, \textbf{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

- \textbf{$k$-Independent Set}
- \textbf{$k$-Clique}
- \textbf{$k$-Dominating Set}
- \textbf{$k$-Vertex Cover}
- \textbf{$k$-Path}
- \textbf{$k$-Feedback Vertex Set}
- \ldots
Transfering bounds

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

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

- \textsc{k-Independent Set}
- \textsc{k-Clique}
- \textsc{k-Dominating Set}
- \textsc{k-Vertex Cover}
- \textsc{k-Path}
- \textsc{k-Feedback Vertex Set}
- \ldots
Lower bounds based on ETH

What about \textbf{3-Coloring} on planar graphs?

The textbook reduction from \textbf{3-Coloring} to \textbf{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 \textsc{3-Coloring} on planar graphs?

The textbook reduction from \textsc{3-Coloring} to \textsc{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 \textsc{3-Coloring} on planar graphs?

The textbook reduction from \textsc{3-Coloring} to \textsc{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 \textsc{3-Coloring} to \textsc{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.

\begin{align*}
\textbf{3SAT formula } & \phi \\
\text{ } & n \text{ variables} \\
\text{ } & m \text{ clauses} \\
\Rightarrow & \quad \text{Graph } G \\
& O(m) \text{ vertices} \\
& O(m) \text{ edges} \\
\Rightarrow & \quad \text{Planar graph } G' \\
& O(m^2) \text{ vertices} \\
& O(m^2) \text{ edges}
\end{align*}

\textbf{Corollary}

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

(Essentially observed by [Cai and Juedes 2001])
Lower bounds for planar problems

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

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

Note: Reduction to planar graphs does not work for **Clique** (why?).
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?).
Recall from Tuesday:
FPT algorithms parameterized by treewidth.
Given a tree decomposition of width $w$, FPT algorithms with running time $2^{O(w)} \cdot n^{O(1)}$ for

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

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

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

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

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

- **Vertex Coloring**
- **Cycle Packing**
- **Vertex Disjoint Paths**
Treewidth

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

- **Vertex Coloring**
- **Cycle Packing**
- **Vertex Disjoint Paths**

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

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

![Graph with 6 cliques]
**Edge Clique Cover**

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

**Equivalently:** can $G$ be represented as an intersection graph over a $k$ element universe?
**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 $|V(G)| > 2^k$, then 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 [Cygan, Pilipczuk, Pilipczuk 2013]**

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

**Proof:** Reduce an $n$-variable 3SAT instance into an instance of **Edge Clique Cover** with $k = O(\log n)$. 
Lower bounds for W[1]-hard problems
## Exponential Time Hypothesis

### Engineers’ Hypothesis

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

### Theorists’ Hypothesis

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

### Exponential Time Hypothesis (ETH)

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

What do we have to show to prove that ETH implies Engineers’ Hypothesis?
Exponential Time Hypothesis

**Engineers’ Hypothesis**

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

**Theorists’ Hypothesis**

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

**Exponential Time Hypothesis (ETH)**

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

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

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

Engineers’ Hypothesis

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

Theorists’ Hypothesis

\[ k\text{-Step Halting Problem} \] (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)

\[ n\text{-variable 3SAT} \] cannot be solved in time \( 2^{o(n)} \).

We actually show something much stronger and more interesting:

Theorem [Chen et al. 2004]

Assuming ETH, there is no \( f(k) \cdot n^{o(k)} \) algorithm for \( k\text{-Clique} \) for any computable function \( f \).
Lower bound on the exponent

**Theorem [Chen et al. 2004]**

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

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

**Theorem [Chen et al. 2004]**

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

Suppose that \( k\text{-CLIQUE} \) 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 \( 3\text{-COLORING} \) on an \( n \)-vertex graph \( G \) in time \( 2^{o(n)} \).

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

Split the vertices of \( G \) into \( k \) groups. Let us build a graph \( H \) where each vertex corresponds to a proper \( 3 \)-coloring of one of the groups. Connect two vertices if they are not conflicting.
**Lower bound on the exponent**

**Theorem [Chen et al. 2004]**

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

Suppose that \( k\text{-Clique} \) 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 \( 3\text{-Coloring} \) 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 \( G \).

\[ \Rightarrow \text{A 3-coloring of } G \text{ can be found in time} \]

\[ f(k) \cdot |V(H)|^{k/s(k)} \leq n \cdot \left(3^{n/k}ight)^{k/s(k)} = n \cdot k^{k/s(k)} \cdot 3^{n/s(k)} = 2^{o(n)}. \]
Tight bounds

Theorem [Chen et al. 2004]
Assuming ETH, there is no $f(k) \cdot n^{o(k)}$ algorithm for $k$-CLIQUE for any computable function $f$.

Transfering to other problems:

$k$-CLIQUE $(x, k)$ \quad \Rightarrow \quad \text{Problem } A (x', O(k))

\quad f(k) \cdot n^{o(k)} \quad \text{algorithm} \quad \Leftarrow \quad f(k) \cdot n^{o(k)} \quad \text{algorithm}
Tight bounds

Theorem [Chen et al. 2004]

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

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

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 [Chen et al. 2004]**

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

**Transfering to other problems:**

<table>
<thead>
<tr>
<th>$k$-CLIQUE $(x, k)$</th>
<th>$f(k) \cdot n^{o(k)}$ algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Rightarrow$</td>
<td>Problem $A$ $(x', k^2)$</td>
</tr>
<tr>
<td></td>
<td>$\Leftrightarrow$ $f(k) \cdot n^{o(\sqrt{k})}$ algorithm</td>
</tr>
</tbody>
</table>
Tight bounds

**Theorem [Chen et al. 2004]**

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

Transfering to other problems:

<table>
<thead>
<tr>
<th>$k$-CLIQUE $(x, k)$</th>
<th>Problem A $(x', k^2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(k) \cdot n^{o(k)}$ algorithm</td>
<td>$f(k) \cdot n^{o(\sqrt{k})}$ algorithm</td>
</tr>
</tbody>
</table>

**Bottom line:**

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

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

- **Set Cover**
- **Hitting Set**
- **Connected Dominating Set**
- **Independent Dominating Set**
- **Partial Vertex Cover**
- **Dominating Set** in bipartite graphs
- ...
The odd case of **Odd Set**

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

We have seen:

**Theorem**

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

New parameter: $k' := k + \binom{k}{2} = O(k^2)$. 
The odd case of **Odd Set**

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

We have seen:

**Theorem**

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

We immediately get:

**Corollary**

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

But this does not seem to be tight...

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

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

Trivial reduction from $k$-Clique:

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

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

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

Trivial reduction from $k$-Clique:

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

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

Is this tight?
**Subgraph Isomorphism**

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

Trivial reduction from $k$-Clique:

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

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

Is this tight?

An almost tight result:

**Theorem [M. 2010]**

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

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

Reduction from \( k\text{-Clique} \) to **Odd Set**:

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

Assuming ETH, there is no \( f(k) \) \( n^\Theta(k/\log k) \) time algorithm for **Odd Set**.
Odd Set

Reduction from Subgraph Isomorphism to Odd Set:

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

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

Reduction from **Subgraph Isomorphism** to **Odd Set**:

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

(Where \( k := |E(H)| \))

**Theorem**

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

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

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

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.
Tight bounds

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

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

What about planar problems?

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

**GRID TILING**

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

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

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

**Find:**

<table>
<thead>
<tr>
<th>(1,1)</th>
<th>(5,1)</th>
<th>(1,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3,1)</td>
<td>(1,4)</td>
<td>(2,4)</td>
</tr>
<tr>
<td>(2,4)</td>
<td>(5,3)</td>
<td>(3,3)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(2,2)</th>
<th>(3,1)</th>
<th>(2,2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,4)</td>
<td>(1,2)</td>
<td>(2,3)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(1,3)</th>
<th>(1,1)</th>
<th>(2,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,3)</td>
<td>(1,3)</td>
<td>(5,3)</td>
</tr>
</tbody>
</table>

$k = 3, \ D = 5$
Grid Tiling

**Grid Tiling**

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

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

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>(5,1)</td>
<td>(1,1)</td>
</tr>
<tr>
<td>(3,1)</td>
<td>(1,4)</td>
<td>(2,4)</td>
</tr>
<tr>
<td>(2,4)</td>
<td>(5,3)</td>
<td>(3,3)</td>
</tr>
<tr>
<td>(2,2)</td>
<td>(3,1)</td>
<td>(2,2)</td>
</tr>
<tr>
<td>(1,4)</td>
<td>(1,2)</td>
<td>(2,3)</td>
</tr>
<tr>
<td>(1,3)</td>
<td>(1,1)</td>
<td>(2,3)</td>
</tr>
<tr>
<td>(2,3)</td>
<td>(1,3)</td>
<td>(5,3)</td>
</tr>
</tbody>
</table>

$k = 3, D = 5$
Grid Tiling

**Grid Tiling**

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

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

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

**Simple proof:**

**Fact**

There is a parameterized reduction from $k$-Clique to $k \times k$ Grid Tiling.
Grid Tiling is W[1]-hard

Reduction from \( k \text{-CLIQUE} \)

Definition of the sets:

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

Each diagonal cell defines a value \( v_i \ldots \)
Grid Tiling is \( \text{W}[1]\)-hard

**Reduction from \( k\text{-CLIQUÉ} \)**

**Definition of the sets:**

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

\[
\begin{array}{|c|c|c|c|c|}
\hline
& (v_i,.) & & & \\
\hline
(., v_i) & (v_i, v_i) & (., v_i) & (., v_i) & (., v_i) \\
\hline
(v_i,.) & & & & \\
\hline
(v_i,.) & & & & \\
\hline
\end{array}
\]

... 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.

$v_i$ 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.

<table>
<thead>
<tr>
<th></th>
<th>$(v_i, .)$</th>
<th>$(v_j, .)$</th>
<th>$(v_i, v_i)$</th>
<th>$(v_j, v_i)$</th>
<th>$(v_i, v_i)$</th>
<th>$(v_j, v_i)$</th>
<th>$(v_j, v_j)$</th>
<th>$(v_i, v_i)$</th>
<th>$(v_j, v_i)$</th>
<th>$(v_j, v_j)$</th>
<th>$(v_i, v_i)$</th>
<th>$(v_j, v_i)$</th>
<th>$(v_j, v_j)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(., v_i)$</td>
<td>$(v_i, v_i)$</td>
<td>$(., v_i)$</td>
<td>$(v_j, v_i)$</td>
<td>$(., v_i)$</td>
<td>$(v_i, v_i)$</td>
<td>$(v_j, v_i)$</td>
<td>$(v_j, v_j)$</td>
<td>$(v_i, v_i)$</td>
<td>$(v_j, v_i)$</td>
<td>$(v_j, v_j)$</td>
<td>$(v_i, v_i)$</td>
<td>$(v_j, v_i)$</td>
<td>$(v_j, v_j)$</td>
</tr>
<tr>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
</tr>
<tr>
<td>$(., v_j)$</td>
<td>$(v_i, v_j)$</td>
<td>$(., v_j)$</td>
<td>$(v_j, v_j)$</td>
<td>$(., v_j)$</td>
<td>$(v_i, v_j)$</td>
<td>$(v_j, v_j)$</td>
<td>$(v_i, v_j)$</td>
<td>$(v_j, v_j)$</td>
<td>$(v_i, v_j)$</td>
<td>$(v_j, v_j)$</td>
<td>$(v_i, v_j)$</td>
<td>$(v_j, v_j)$</td>
<td>$(v_i, v_j)$</td>
</tr>
<tr>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
<td>$(v_i, .)$</td>
<td>$(v_j, .)$</td>
</tr>
</tbody>
</table>

$v_i$ and $v_j$ are adjacent for every $1 \leq i < j \leq k$. 
**Grid Tiling** and planar problems

**Theorem**

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

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

**Examples:**

- **Multiway Cut** on planar graphs with $k$ terminals
- **Independent Set** for unit disks
A classical problem

\[ s - t \text{ Cut} \]

Input: A graph \( G \), an integer \( p \), vertices \( s \) and \( t \)

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

Theorem [Ford and Fulkerson 1956]

A minimum \( s - t \) cut can be found in polynomial time.

What about separating more than two terminals?
More than two terminals

**k-Terminal Cut (aka Multiway Cut)**

Input: A graph $G$, an integer $p$, and a set $T$ of $k$ terminals
Output: A set $S$ of at most $p$ edges such that removing $S$ separates any two vertices of $T$

**Theorem [Dalhaus et al. 1994]**

NP-hard already for $k = 3$. 

26
More than two terminals

**k-Terminal Cut (aka Multiway Cut)**

Input: A graph $G$, an integer $p$, and a set $T$ of $k$ terminals
Output: A set $S$ of at most $p$ edges such that removing $S$ separates any two vertices of $T$

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

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

**Theorem** [Klein and M. 2012]

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

**Theorem [Klein and M. 2012]**

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

**Natural questions:**

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

**Theorem [Klein and M. 2012]**

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

Natural questions:

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

**Lower bounds:**

**Theorem [M. 2012]**

**Planar** $k$-**Terminal Cut** is **W[1]**-hard and has no $f(k) \cdot n^{o(\sqrt{k})}$ time algorithm (assuming ETH).
Reduction from $k \times k$ Grid Tiling to Planar $k^2$-Terminal Cut

For every set $S_{i,j}$, we construct a gadget with 4 terminals such that
- for every $(x, y) \in S_{i,j}$, there is a minimum multiway cut that represents $(x, y)$.
- every minimum multiway cut represents some $(x, y) \in S_{i,j}$.

Main part of the proof: constructing these gadgets.
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.

[Diagram of a grid with 4 terminals and 2 sets of 5 terminals at the top and bottom, labeled $U_L$, $U_R$, $D_L$, $D_R$, with cuts represented by lines and terminals denoted $u_1, u_2, u_3, u_4, u_5$, $r_1, r_2, r_3, r_4, r_5$, $d_1, d_2, d_3, d_4, d_5$]
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

Oops!
Putting together the gadgets
Grid Tiling with $\leq$

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

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

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

**Find:**

\[
\begin{array}{ccc}
(5,1) & (4,3) & (2,3) \\
(1,2) & (3,2) & (2,5) \\
(3,3) & & \\
(2,1) & (4,2) & (5,1) \\
(5,5) & (5,3) & (3,2) \\
(3,5) & & \\
(5,1) & (2,1) & (3,1) \\
(2,2) & (4,2) & (3,2) \\
(5,3) & (4,2) & (3,3) \\
\end{array}
\]

$k = 3$, $D = 5$
**Grid Tiling with \( \leq \)**

<table>
<thead>
<tr>
<th><strong>Input:</strong></th>
<th>A ( k \times k ) matrix and a set of pairs ( S_{i,j} \subseteq [D] \times [D] ) for each cell.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A pair ( s_{i,j} \in S_{i,j} ) for each cell such that</td>
</tr>
<tr>
<td><strong>Find:</strong></td>
<td>- 1st coordinate of ( s_{i,j} \leq 1 )st coordinate of ( s_{i+1,j} ).</td>
</tr>
<tr>
<td></td>
<td>- 2nd coordinate of ( s_{i,j} \leq 2 )nd coordinate of ( s_{i,j+1} ).</td>
</tr>
</tbody>
</table>

Variant of the previous proof:

**Theorem**

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

Very useful starting point for geometric problems!
**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:**

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

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

Reduction to unit disks

<table>
<thead>
<tr>
<th>(5,1)</th>
<th>(4,3)</th>
<th>(2,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>(3,2)</td>
<td>(2,5)</td>
</tr>
<tr>
<td>(3,3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(5,1)</th>
<th>(4,2)</th>
<th>(5,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,1)</td>
<td>(5,3)</td>
<td>(3,2)</td>
</tr>
<tr>
<td>(5,5)</td>
<td>(3,5)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(5,1)</th>
<th>(2,1)</th>
<th>(3,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,2)</td>
<td>(4,2)</td>
<td>(3,2)</td>
</tr>
<tr>
<td>(5,3)</td>
<td>(4,2)</td>
<td>(3,3)</td>
</tr>
</tbody>
</table>

Every pair is represented by a unit disk in the plane.
\( \leq \) relation between coordinates \( \iff \) disks do not intersect.
Reduction to unit disks

<table>
<thead>
<tr>
<th>(5,1)</th>
<th>(4,3)</th>
<th>(2,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>(3,2)</td>
<td>(2,5)</td>
</tr>
<tr>
<td>(3,3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(2,1)</th>
<th>(4,2)</th>
<th>(5,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5,5)</td>
<td>(5,3)</td>
<td>(3,2)</td>
</tr>
<tr>
<td>(3,5)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(5,1)</th>
<th>(2,1)</th>
<th>(3,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,2)</td>
<td>(4,2)</td>
<td>(3,2)</td>
</tr>
<tr>
<td>(5,3)</td>
<td>(3,2)</td>
<td>(3,3)</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>(5,1)</th>
<th>(4,3)</th>
<th>(2,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>(3,2)</td>
<td>(2,5)</td>
</tr>
<tr>
<td>(3,3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(2,1)</th>
<th>(4,2)</th>
<th>(5,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5,5)</td>
<td>(5,3)</td>
<td>(3,2)</td>
</tr>
<tr>
<td>(3,5)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(5,1)</th>
<th>(2,1)</th>
<th>(3,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,2)</td>
<td>(4,2)</td>
<td>(3,2)</td>
</tr>
<tr>
<td>(5,3)</td>
<td>(3,3)</td>
<td></td>
</tr>
</tbody>
</table>

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

$\leq$ relation between coordinates $\iff$ disks do not intersect.
Center-pivot irrigation
Higher dimensions

Bidimensionalty for planar graphs:

- $2^{O(\sqrt{n})}, \ 2^{O(\sqrt{k})} \cdot n^{O(1)}, \ n^{O(\sqrt{k})}$ time algorithms.
- There is no tridimensionalty!
Higher dimensions

Bidimensionality for 2-dimensional geometric problems:

- $2^{O(\sqrt{n})}$, $2^{O(\sqrt{k})} \cdot n^{O(1)}$, $n^{O(\sqrt{k})}$ time algorithms.
- What about higher dimensions?
Higher dimensions

Bidimensionality for 2-dimensional geometric problems:

- \(2^O(\sqrt{n})\), \(2^O(\sqrt{k}) \cdot n^O(1)\), \(n^O(\sqrt{k})\) time algorithms.
- What about higher dimensions?

“Limited blessing of low dimensionality:”

**Theorem**

**Independent Set** for unit spheres in \(d\) dimensions can be solved in time \(n^O(k^{1-1/d})\).

Matching lower bound:

**Theorem [M. and Sidiropoulos 2014]**

Assuming ETH, **Independent Set** for unit spheres in \(d\) dimensions cannot be solved in time \(n^{o(k^{1-1/d})}\).
Higher dimensions

Bidimensionality for 2-dimensional geometric problems:

- $2^O(\sqrt{n})$, $2^O(\sqrt{k}) \cdot n^{O(1)}$, $n^{O(\sqrt{k})}$ time algorithms.
- What about higher dimensions?

“Limited blessing of low dimensionality:”

**Theorem [Smith and Wormald 1998]**

**Euclidean TSP** in $d$ dimensions can be solved in time $2^O(n^{1-1/d+\epsilon})$.

Matching lower bound:

**Theorem [M. and Sidiropoulos 2014]**

Assuming ETH, **Euclidean TSP** in $d$ dimension cannot be solved in time $2^O(n^{1-1/d-\epsilon})$ for any $\epsilon > 0$. 
Summary

We used ETH to rule out

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

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