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., 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?

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.

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 ϕ 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 $k \Leftrightarrow G$ has a vertex cover of size $n - k$.

Independent Set
(G, k)

$\Rightarrow$

Vertex Cover
$(G, n - k)$

- This is a correct polynomial-time reduction.
- However, Vertex Cover is FPT, but Independent Set is not known to be FPT.

Parameterized reductions

Definition

Parameterized reduction from problem A to problem B: a function ϕ 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 ϕ 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 $A \to B$ + algorithm for B gives and algorithm for A.

Parameterized reductions

Definition

Parameterized reduction from problem A to problem B: a function ϕ 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

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

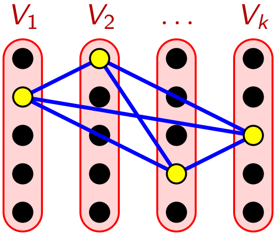
for some function f'.

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

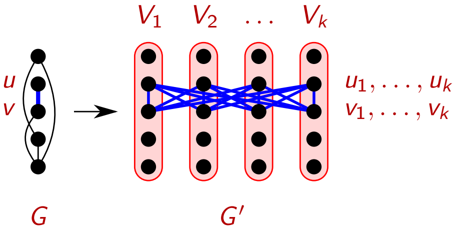
There is a parameterized reduction from CLIQUE to MULTICOLORED CLIQUE.

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.



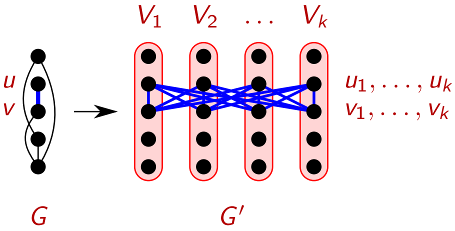
k-clique in G $\iff$ multicolored k-clique in G'.

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'.

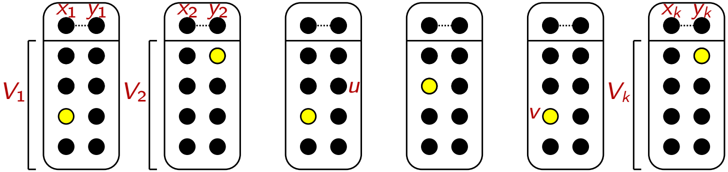
Similarly: reduction to Multicolored Independent Set.

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.



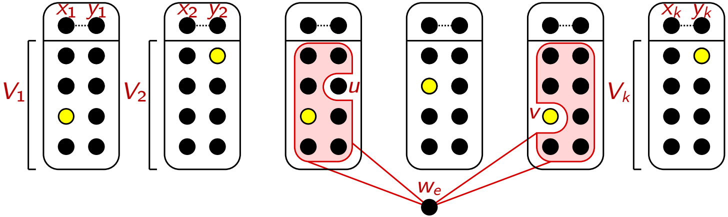
- The dominating set has to contain one vertex from each of the k cliques $V_1, \ldots, V_k$ to dominate every x_i and y_i.

DOMINATING SET

Theorem

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

Proof: Let G be a graph with color classes $V_1, \dots, V_k$. 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 $V_1, \dots, V_k$ 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 $\mathsf{P} \neq \mathsf{NP}$ – we have to assume something stronger.

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.

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.

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 $\mathrm{P} \neq \mathrm{NP}$ – we have to assume something stronger.

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?

Summary

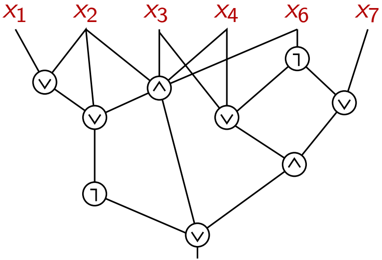
- INDEPENDENT SET and k-STEP HALTING PROBLEM can be reduced to each other ⇒ Engineers' Hypothesis and Theorists' Hypothesis are equivalent!
- INDEPENDENT SET and k-STEP HALTING PROBLEM can be reduced to DOMINATING SET.

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!

Boolean circuit

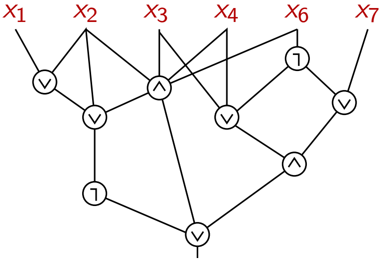
A **Boolean circuit** consists of input gates, negation gates, AND gates, OR gates, and a single output gate.



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.



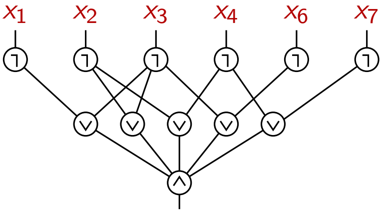
CIRCUIT SATISFIABILITY: Given a Boolean circuit C, decide if there is an assignment on the inputs of C making the output true.

Weight of an assignment: number of true values.

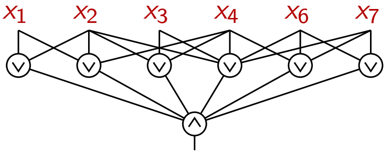
WEIGHTED CIRCUIT SATISFIABILITY: Given a Boolean circuit C and an integer k, decide if there is an assignment of weight k making the output true.

WEIGHTED CIRCUIT SATISFIABILITY

INDEPENDENT SET can be reduced to WEIGHTED CIRCUIT SATISFIABILITY:

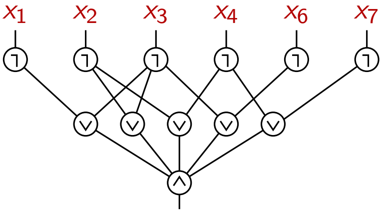


DOMINATING SET can be reduced to WEIGHTED CIRCUIT SATISFIABILITY:

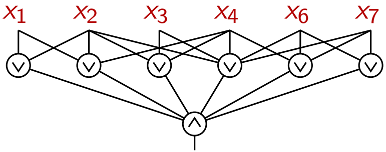


WEIGHTED CIRCUIT SATISFIABILITY

INDEPENDENT SET can be reduced to WEIGHTED CIRCUIT SATISFIABILITY:



DOMINATING SET can be reduced to WEIGHTED CIRCUIT SATISFIABILITY:



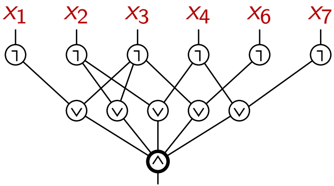
To express DOMINATING SET, we need more complicated circuits.

Depth and weft

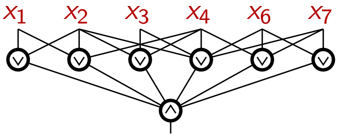
The **depth** of a circuit is the maximum length of a path from an input to the output.

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].

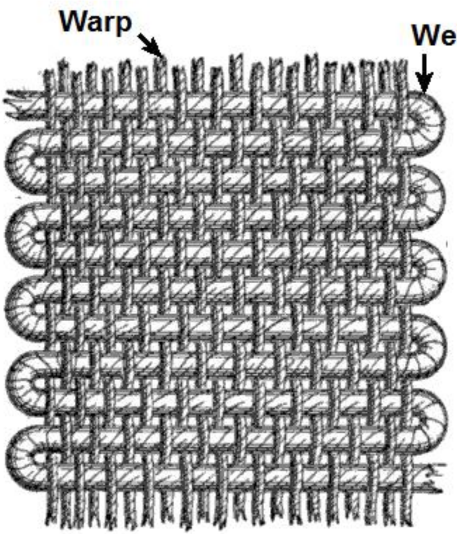
Fact: Independent Set is W[1]-complete.
Fact: Dominating Set is W[2]-complete.

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].

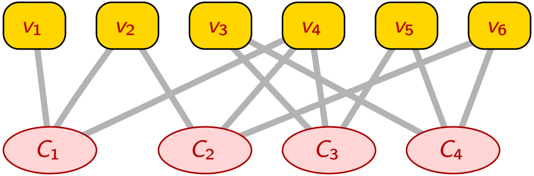
$\Rightarrow$ If there is a parameterized reduction from Dominating Set to Independent Set, then W[1] $=$ W[2].

Weft



Parameterized reductions

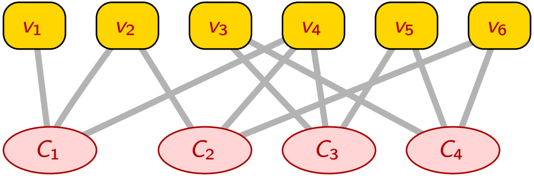
Typical NP-hardness proofs: reduction from e.g., CLIQUE or 3SAT, representing each vertex/edge/variable/clause with a gadget.



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

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 ⇒ INDEPENDENT SET
- Reductions with vertex representations.
  - MULTICOLORED INDEPENDENT SET ⇒ 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.

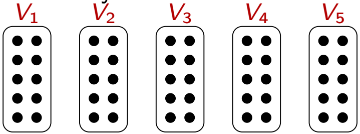
ODD SET

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?

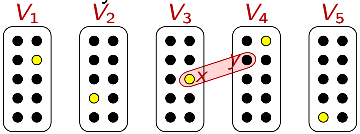
Odd Set

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,

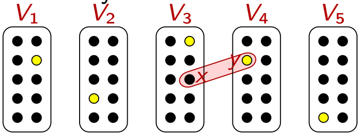
ODD SET

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,

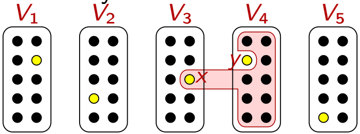
ODD SET

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.

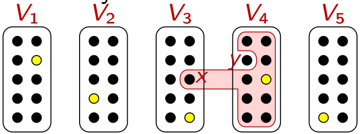
ODD SET

Theorem

ODD SET is W[1]-hard parameterized by k.

First try: Reduction from MULTICOLORED INDEPENDENT SET. Let $U = V_1 \cup \dots V_k$ and introduce each set V_i into $\mathcal{F}$.

$\Rightarrow$ The solution has to contain exactly one element from each V_i.



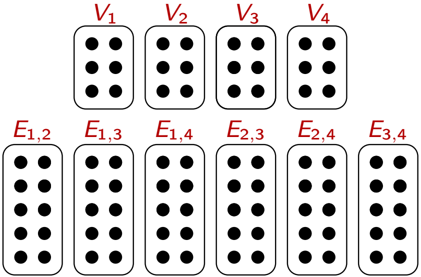
If $xy \in E(G)$, how can we express that $x \in V_i$ and $y \in V_j$ cannot be selected simultaneously? Seems difficult:

- 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.

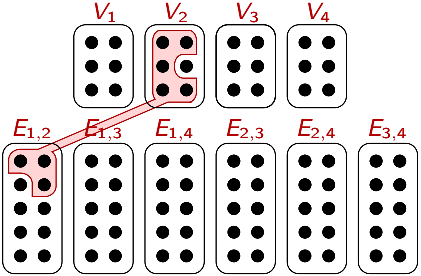
- $U := \bigcup_{i=1}^{k} V_i \cup \bigcup_{1 \leq i < j \leq k} E_{i,j}$.
- $k' := k + \binom{k}{2}$.
- Let $\mathcal{F}$ contain V_i $(1 \leq i \leq k)$ and $E_{i,j}$ $(1 \leq i < j \leq k)$.



ODD SET

Reduction from MULTICOLORED CLIQUE.

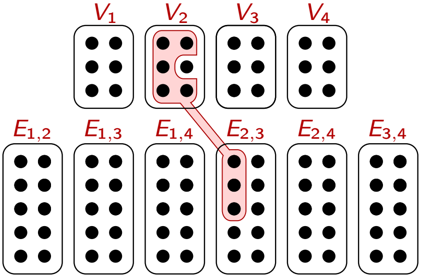
- $U := \bigcup_{i=1}^{k} V_i \cup \bigcup_{1 \le i < j \le k} E_{i,j}$.
- $k' := k + \binom{k}{2}$.
- Let $\mathcal{F}$ contain V_i ($1 \le i \le k$) and $E_{i,j}$ ($1 \le i < j \le k$).
- For every $v \in V_i$ and $x \ne i$, we introduce the sets:
  $(V_i \setminus \{v\}) \cup \{$every edge from $E_{i,x}$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



ODD SET

Reduction from MULTICOLORED CLIQUE.

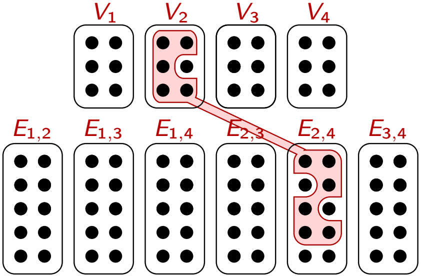
- $U := \bigcup_{i=1}^{k} V_i \cup \bigcup_{1 \le i < j \le k} E_{i,j}$.
- $k' := k + \binom{k}{2}$.
- Let $\mathcal{F}$ contain V_i $(1 \le i \le k)$ and $E_{i,j}$ $(1 \le i < j \le k)$.
- For every $v \in V_i$ and $x \neq i$, we introduce the sets:
  $(V_i \setminus \{v\}) \cup \{$every edge from $E_{i,x}$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



ODD SET

Reduction from MULTICOLORED CLIQUE.

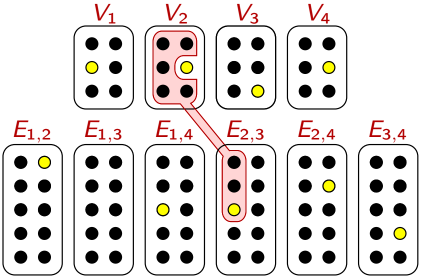
- $U := \bigcup_{i=1}^{k} V_i \cup \bigcup_{1 \le i < j \le k} E_{i,j}$.
- $k' := k + \binom{k}{2}$.
- Let $\mathcal{F}$ contain V_i $(1 \le i \le k)$ and $E_{i,j}$ $(1 \le i < j \le k)$.
- For every $v \in V_i$ and $x \neq i$, we introduce the sets:
  $(V_i \setminus \{v\}) \cup \{\text{every edge from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



ODD SET

Reduction from MULTICOLORED CLIQUE.

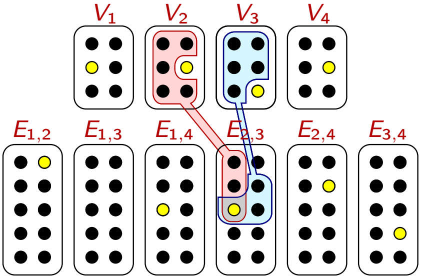
- For every $v \in V_i$ and $x \neq i$, we introduce the sets:
  $(V_i \setminus \{v\}) \cup \{\text{every edge from } E_{i,x} \text{ with endpoint } v\}$
  $(V_i \setminus \{v\}) \cup \{\text{every edge from } E_{x,i} \text{ with endpoint } v\}$
- $v \in V_i$ selected $\iff$ edges with endpoint v are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

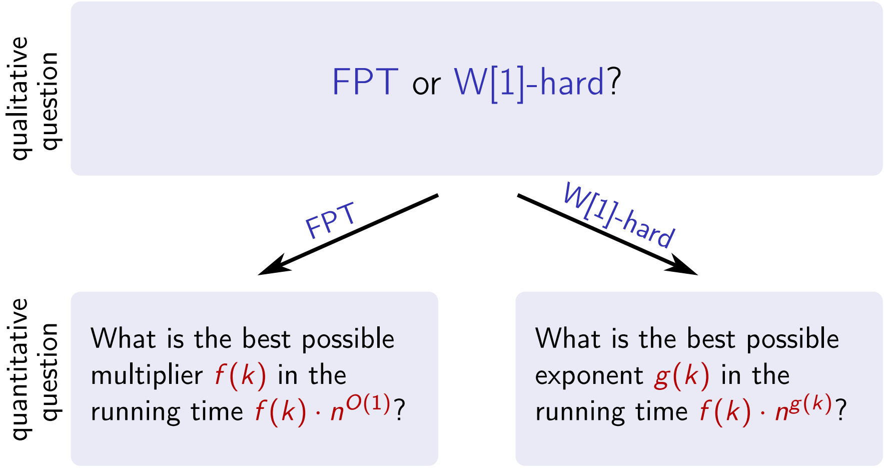
- By parameterized reductions, we can show that lots of parameterized problems are at least as hard as CLIQUE, hence unlikely to be fixed-parameter tractable.
- Connection with Turing machines gives some supporting evidence for hardness (only of theoretical interest).
- The W-hierarchy classifies the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



qualitative question

FPT or W[1]-hard?

FPT

W[1]-hard

quantitative question

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 VERTEX COVER

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

Better algorithms for Vertex Cover

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

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

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

Exponential Time Hypothesis (ETH)

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

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

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

Exponential Time Hypothesis (ETH)

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

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

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

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 $\mathrm{3SAT}$.

Exponential Time Hypothesis (ETH)

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

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

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

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 $\mathrm{3SAT}$.

Sparsification

Exponential Time Hypothesis (ETH) [consequence of]

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

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

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

Sparsification

Exponential Time Hypothesis (ETH) [consequence of]

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

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

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

Sparsification Lemma

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

$$\Updownarrow$$

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

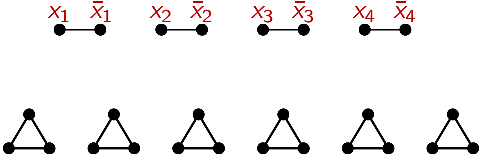
Intuitively: When considering a hard $\mathrm{3SAT}$ instance, we can assume that it has $m = O(n)$ clauses.

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:



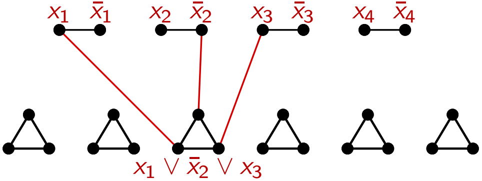
Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma

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

The textbook reduction from 3SAT to VERTEX COVER:

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

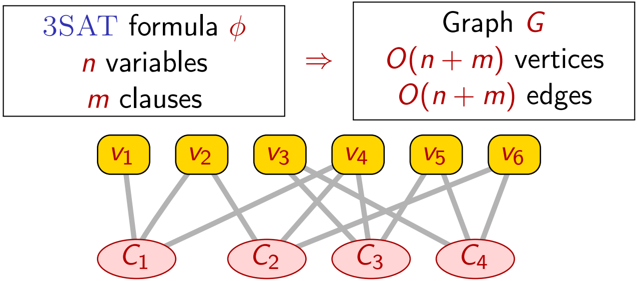


Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma

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

The textbook reduction from 3SAT to Vertex Cover:



3SAT formula ϕ
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

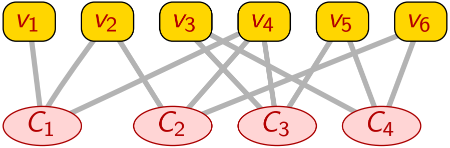
There is no $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

The textbook reduction from 3SAT to VERTEX COVER:

3SAT formula ϕ
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

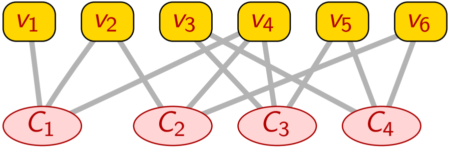
There is no $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

The textbook reduction from 3SAT to VERTEX COVER:

3SAT formula ϕ
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
- ...

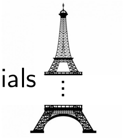
Seems to be the natural behavior of FPT problems?

The race for better FPT algorithms

$f(k)$

$2^{2^{\cdot^{\cdot^{\cdot^{2^k}}}}}$

Tower of exponentials



Double exponential

$2^{2^{O(k)}}$

$2^{k^{O(1)}}$

2^{k^2}

"Slightly super-exponential"

$2^{O(k \log k)}$

1789^k

100^k

14^k

7^k

3.59^k

2^k

1.27^k

Single exponential

$2^{O(k/\log k)}$

$2^{O(\sqrt{k} \log k)}$

$2^{O(\sqrt{k})}$

$2^{O(\sqrt[3]{k})}$

$\ldots$

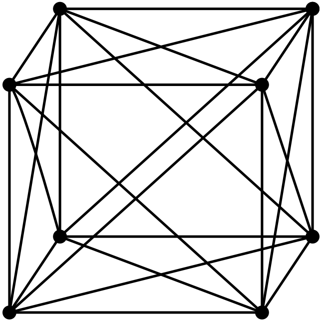
Subexponential

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?

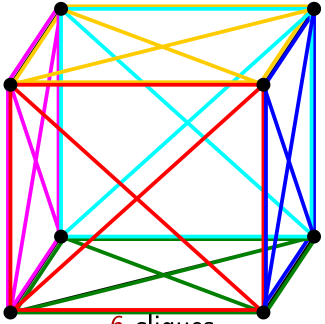


EDGE CLIQUE COVER

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?



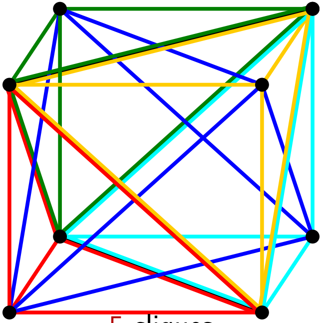
6 cliques

EDGE CLIQUE COVER

EDGE CLIQUE COVER: Given a graph G and an integer k, cover the edges of G with at most k cliques.

(the cliques need not be edge disjoint)

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


5 cliques

EDGE CLIQUE COVER

EDGE CLIQUE COVER: Given a graph G and an integer k, cover the edges of G with at most k cliques.

(the cliques need not be edge disjoint)

Simple algorithm (sketch)

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

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

EDGE CLIQUE COVER

EDGE CLIQUE COVER: Given a graph G and an integer k, cover the edges of G with at most k cliques.

(the cliques need not be edge disjoint)

Double-exponential dependence on k cannot be avoided!

Theorem

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

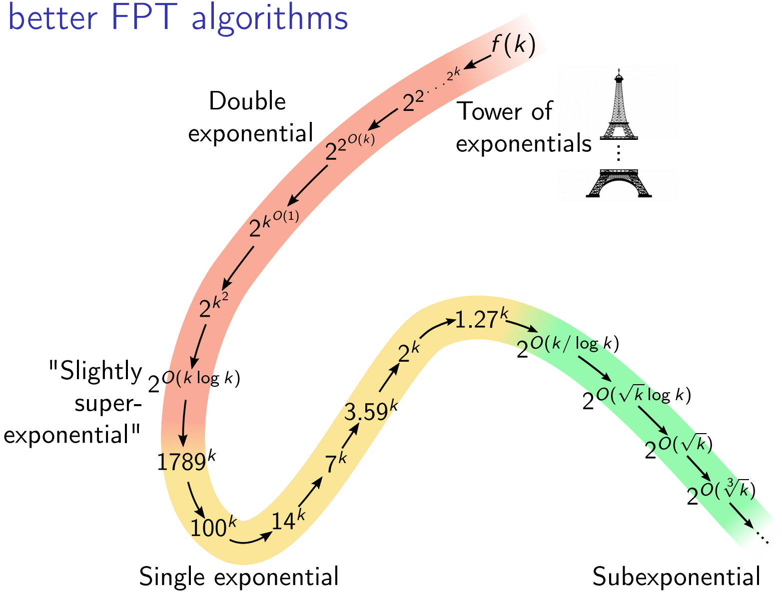
Proof:

3SAT
n variables

$\Rightarrow$

EDGE CLIQUE COVER
$k = O(\log n)$

The race for better FPT algorithms



$f(k)$

$2^{2^{\cdot^{\cdot^{\cdot^{2^k}}}}}$

Tower of exponentials

$2^{2^{O(k)}}$

Double exponential

$2^{k^{O(1)}}$

2^{k^2}

$2^{O(k \log k)}$

"Slightly super-exponential"

1789^k

100^k

14^k

7^k

3.59^k

2^k

1.27^k

Single exponential

$2^{O(k/\log k)}$

$2^{O(\sqrt{k} \log k)}$

$2^{O(\sqrt{k})}$

$2^{O(\sqrt[3]{k})}$

$\dots$

Subexponential

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

Closest String

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

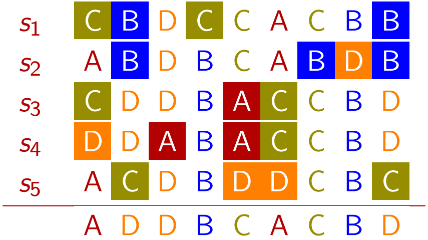
(Hamming distance: number of differing positions)

s_1	C	B	D	C	C	A	C	B	B
s_2	A	B	D	B	C	A	B	D	B
s_3	C	D	D	B	A	C	C	B	D
s_4	D	D	A	B	A	C	C	B	D
s_5	A	C	D	B	D	D	C	B	C

CLOSEST STRING

CLOSEST STRING
Given strings $s_1, \dots, s_k$ of length L over alphabet Σ, 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)



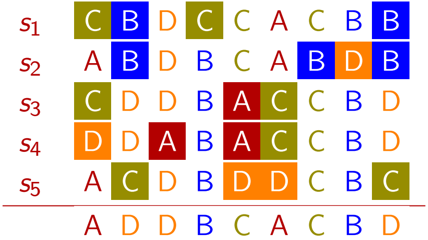
s_1	C	B	D	C	C	A	C	B	B
s_2	A	B	D	B	C	A	B	D	B
s_3	C	D	D	B	A	C	C	B	D
s_4	D	D	A	B	A	C	C	B	D
s_5	A	C	D	B	D	D	C	B	C
	A	D	D	B	C	A	C	B	D

CLOSEST STRING

CLOSEST STRING

Given strings $s_1, \ldots, s_k$ of length L over alphabet Σ, 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)



s_1	C	B	D	C	C	A	C	B	B
s_2	A	B	D	B	C	A	B	D	B
s_3	C	D	D	B	A	C	C	B	D
s_4	D	D	A	B	A	C	C	B	D
s_5	A	C	D	B	D	D	C	B	C
	A	D	D	B	C	A	C	B	D

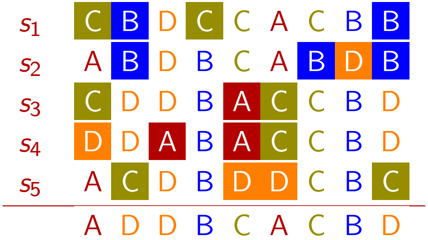
Different parameters:

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

CLOSEST STRING

CLOSEST STRING
Given strings $s_1, \ldots, s_k$ of length L over alphabet Σ, 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)



s_1	C	B	D	C	C	A	C	B	B
s_2	A	B	D	B	C	A	B	D	B
s_3	C	D	D	B	A	C	C	B	D
s_4	D	D	A	B	A	C	C	B	D
s_5	A	C	D	B	D	D	C	B	C
	A	D	D	B	C	A	C	B	D

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 ℓ 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 ℓ 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 $(d + 1)^d \cdot n^{O(1)} = 2^{O(d \log d)} n^{O(1)}$.

Closest String

Theorem

Assuming ETH, Closest String has no $2^{o(d \log d)} n^{O(1)}$ algorithm.

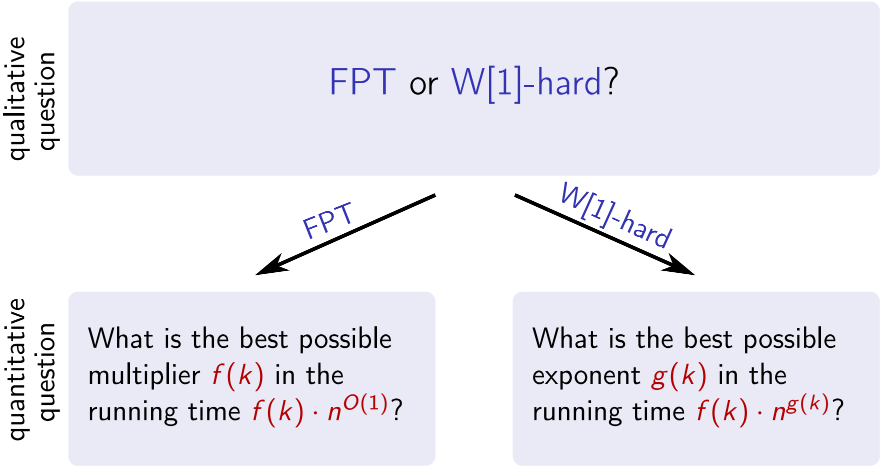
Proof:

3SAT
$O(d \log d)$ variables

$\Longrightarrow$

Closest String
distance d

Shift of focus



qualitative question

FPT or W[1]-hard?

FPT

W[1]-hard

quantitative question

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 $O(k)$?

$n^{\sqrt{k}}$

$n^{k/\log\log k}$

$n^{\log k}$

$n^{\sqrt{k}}$

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

Better algorithms for W[1]-hard problems

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

Theorem

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

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

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

3-coloring
n vertices

$\Rightarrow$

Clique
(G, k) with
$N \approx 3^{n/k}$ vertices

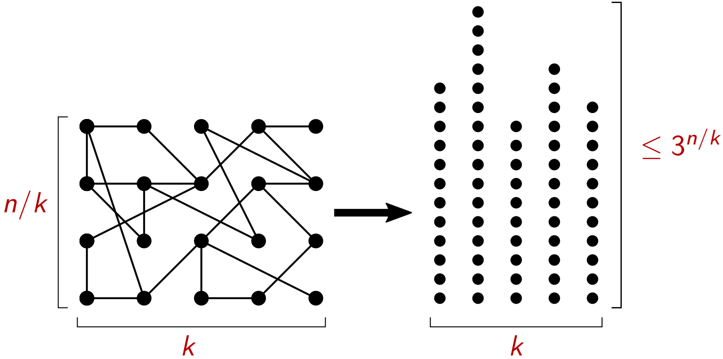
$N^{o(k)}$ algorithm for Clique $\Rightarrow (3^{n/k})^{o(k)} = 3^{o(n)}$ algorithm for 3-coloring

Lower bound for k-Clique

Theorem

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

Proof:



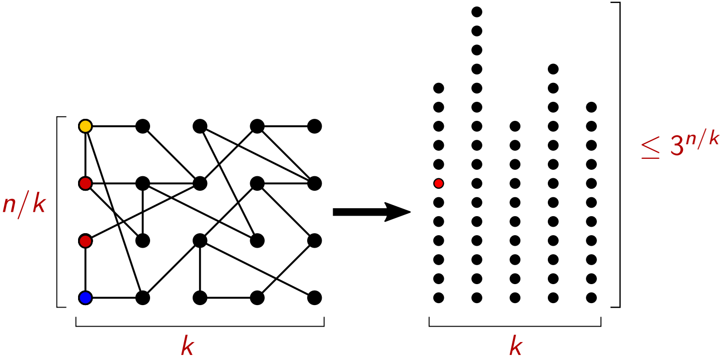
Create a vertex per each consistent coloring of each group.

Lower bound for k-Clique

Theorem

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

Proof:



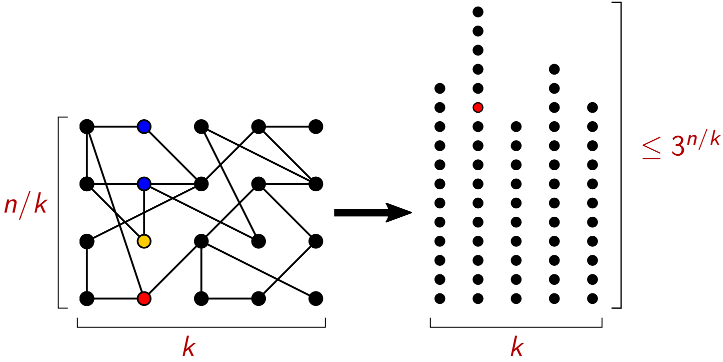
Create a vertex per each consistent coloring of each group.

Lower bound for k-CLIQUE

Theorem

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

Proof:



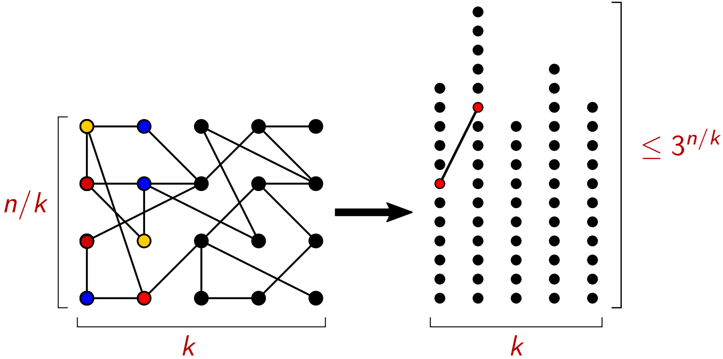
Create a vertex per each consistent coloring of each group.

Lower bound for k-CLIQUE

Theorem

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

Proof:



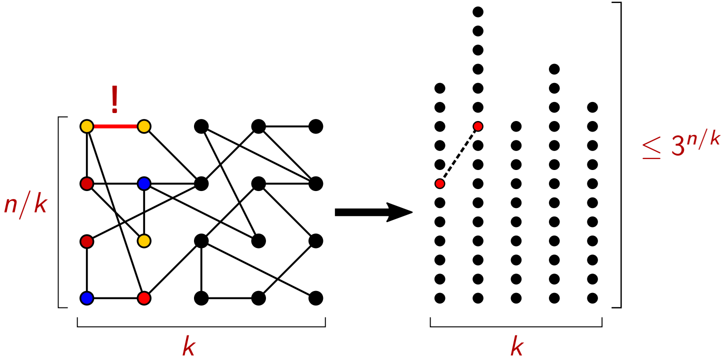
Connect two vertices if they represent colorings that are consistent together.

Lower bound for k-CLIQUE

Theorem

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

Proof:



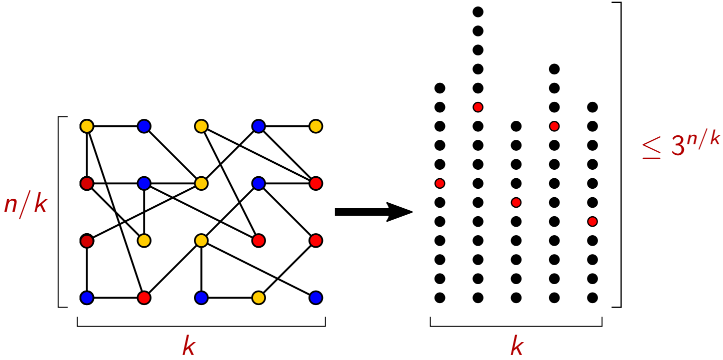
Connect two vertices if they represent colorings that are consistent together.

Lower bound for k-CLIQUE

Theorem

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

Proof:



Left graph has a 3-coloring ⇔ 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

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

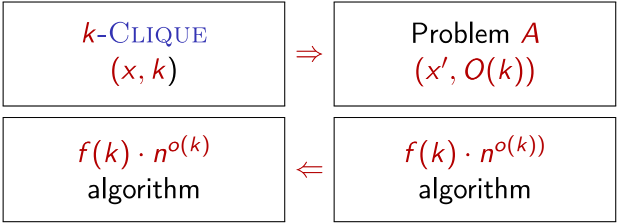
- Choosing $k := \log\log n$ would rule out a $2^{2^k} \cdot N^{o(k)}$ algorithm etc.
- In general, we need to choose roughly $k := f^{-1}(n)$ groups (technicalities omitted).

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)

$\Rightarrow$

Problem A
$(x', O(k))$

$f(k) \cdot n^{o(k)}$
algorithm

$\Leftarrow$

$f(k) \cdot n^{o(k))}$
algorithm

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:

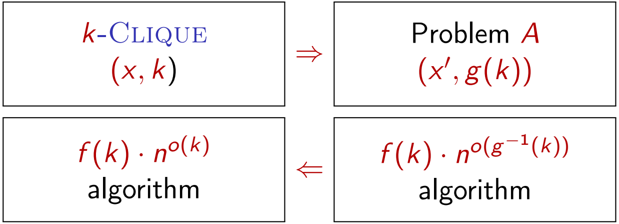
k-Clique (x, k)	$\Rightarrow$	Problem A (x', k^2)
$f(k) \cdot n^{o(k)}$ algorithm	$\Leftarrow$	$f(k) \cdot n^{o(\sqrt{k})}$ algorithm

Tight bounds

Theorem

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

$\Leftarrow$

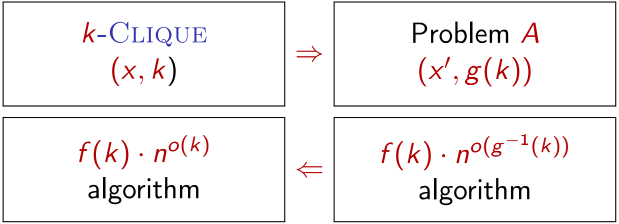
$f(k) \cdot n^{o(g^{-1}(k))}$
algorithm

Tight bounds

Theorem

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

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
- ...

Summary

- Parameterized reductions from Clique or 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.