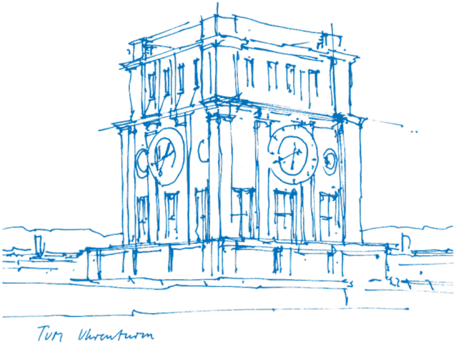
Parametrized Complexity

Seminar: Advanced Algorithms

Lukas Retschmeier

Informatik 7 - Theoretical Foundations of Artificial Intelligence
Faculty of Informatics
Technical University of Munich

February 3rd, 2022



Classical Algorithm Design

- Usually aims for a ("good") polynomial-time (PTIME) algorithm
- For NP-c problems, we do not expect a PTIME algorithms
- **For Example:** `3SAT`, `3COLORING`, `CLIQUE`, `VERTEX COVER`, ...

But ... Can we say anything more about those problems?

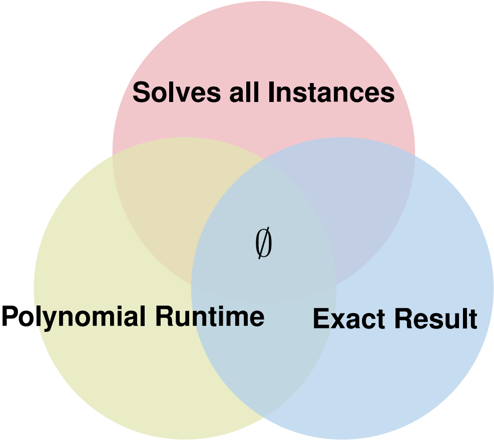
Agenda: Our Plan for Today

1. **Introduction and Definitions**
2. **Fixed Parameter (In)Tractibility & ω-Hardness**
3. **A Stronger Assumption: (S)ETH and Proving Lower Bounds**

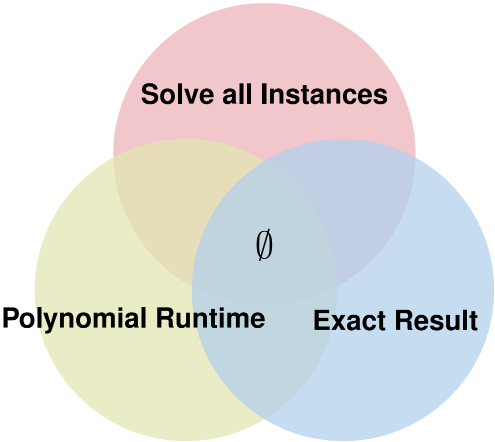
Part I

Introduction and Definitions

Ways to Cope with NP-Complete Problems



Ways to Cope with NP-Complete Problems



We must give up at least one:

- **Exactness:** Approximation Algorithms
- **Polynomial Runtime:** Exact Exponential Time Algorithms
- **Generality:** **FPT Algorithms**

Parametrized Complexity

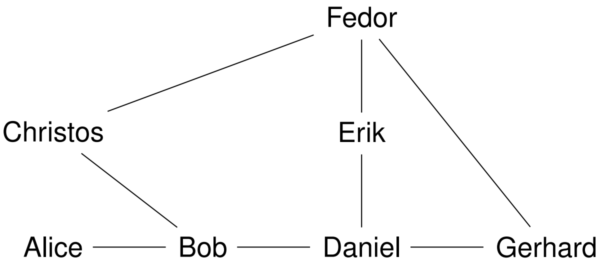
- *Parametrized Complexity* can be seen as a *2-Dimensional* complexity analysis
- Looking deep into the *nature of the problem* to find some hidden (in)-feasibility
  - Graph of small size?
  - Planar Graph? A Tree?
  - A tree "with a lot of fantasy"?
  - Forbidden Minor?
  - Regular? Degree-Bounded?
  - Bipartite? Chordal?[1]
  - ...

[1] A State-of-the-art collection: https://www.graphclasses.org/

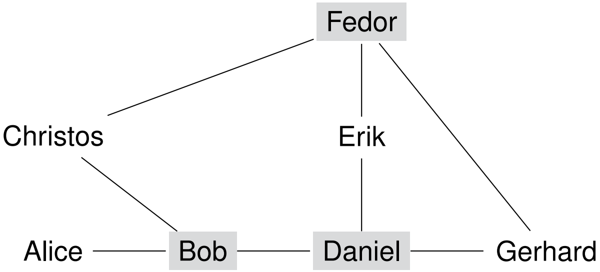
The Idea Behind: *Bar Fight Prevention*

The Problem

Owning a Bar is very difficult! You already know that some people might fight so you prevent certain trouble makers from entering. **How many do you have to block at least to resolve all conflicts?**



The Idea Behind: *Bar Fight Prevention*



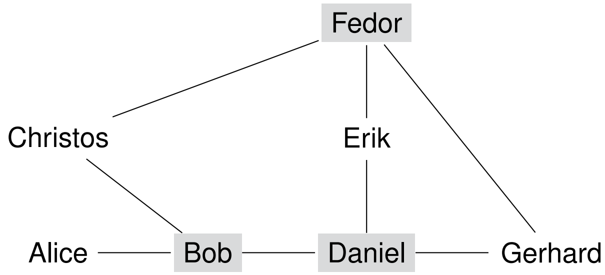
Observation

Removing *Fedor*, *Daniel* and *Bob* resolves all conflicts.

Assuming 1.000 guests: $2^{1000} \approx 1.07 \cdot 10^{301}$ **Absolutely infeasible**

Restricting the Problem

Question: What happens if you just have a budget of k-people you would like to refuse?

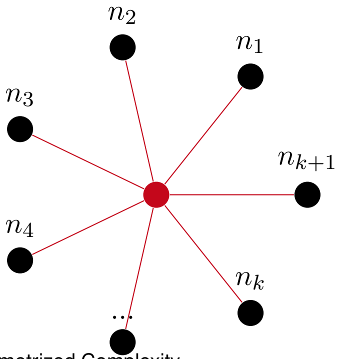


Assuming 1.000 guests and $k = 10$: $= \binom{1000}{10} \approx 2.62 \cdot 10^{23}$ **Still pretty infeasible**

Can we do better?

Observation

Someone fighting with at least $k+1$ other guests must be refused, because otherwise **all other** $k+1$ guests must be refused, thus already exceeding our budget!



Kernelization I

- $\mathtt{max}_{\mathtt{deg}} \leq k$
- Rejecting a guest will now resolve at most k conflicts
- We are allowed to remove **at most** k guests each having at most k conflicts
- If $> k^2$ conflicts remaining: No way to resolve all: **Refuse Instance**

$$\binom{2k^2}{k} \leq \binom{200}{10} \approx 2.24 \cdot 10^{16}$$

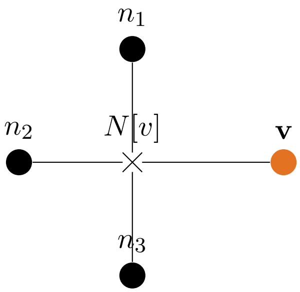
Feasible, but still ...

Note: This technique is called *Kernelization*.

Kernelization II: Simple Improvement

Observation

If $deg(v) = 1$ refuse $N[v]$ and decrease k



Analysis

- Degree now bounded by $1 < \mathtt{deg}(v) <= k$
- $\binom{k^2}{k} \leq \binom{100}{10} \approx 1.73 \cdot 10^{13}$
- **Even Better!**

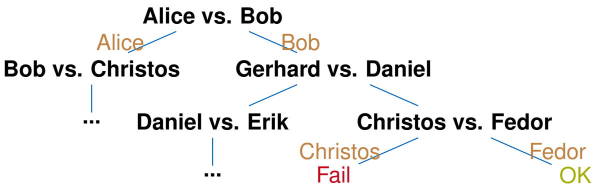
A Different Approach: Bounded Search Trees

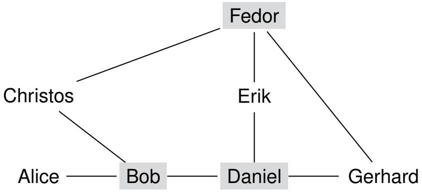
Crucial Observation

Every conflict *must* be resolved.

$\Rightarrow$ For every conflicting pair **at least one must be refused** [a]

[a] This also leads to 2-approximation algorithm! (See: Cormen et al. 2009, Ch. 35.1)





Final Runtime Using Branching

- We branch into **two sub-branches** and always **decrease k by one**.
- Traversing the graph yields $\mathcal{O}(m + n)$ where m is the number of potential conflicts.
- Recall $m \leq \frac{nk}{2}$ after our preciously discussed pre-processing procedure

So we finally get:

$$\mathcal{O}(2^k \cdot n \cdot k)$$

For $n = 1.000$ and $k = 10$: $2^{10} \cdot 1.000 \cdot 10 = 10.240.000$ 🙂

Parametrized Problem

Main Idea: Instead of expressing the running time as a function $T(n)$ of n...
...we express it as a function $T(n, k)$ of the input size n and *some* parameter k of the input.

Definition 1: Parametrized Problem

A parametrized problem is a $L \subseteq \Sigma^* \times \mathbb{N}$ (Σ finite fixed alphabet) for an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, where k is called the parameter.

Examples for a parameter k:

- size k of a `VERTEX COVER`
- size k of a `INDEPENDENT SET`
- Treewidth k of a given graph

The Class FPT

Definition 2: Fixed-Parameter Tractable

A parametrized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called *fixed-parameter tractable (FPT)* if there exists an algorithm A (called a *fixed-parameter algorithm*), a computable function $f : \mathbb{N} \to \mathbb{N}$ and a constant c such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, the algorithm A correctly decides whether $(x, k) \in L$ in time bounded by $f(k) \cdot |(x, k)|^c$. The complexity class containing all fixed-parameter tractable problems is called FPT.

Note: We often omit the polynomial-factor and rewrite the running time simply as $\mathcal{O}^*(f(k))$

The Class XP

Definition 3: Slice-Wise Polynomial

A parametrized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called *slice-wise polynomial* (XP) if there exists an algorithm A and two computable functions $f, g : \mathbb{N} \to \mathbb{N}$ such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, A correctly decides whether $(x, k) \in L$ in time bounded by $f(k) \cdot |(x, k)|^{g(k)}$. The complexity class containing all slice-wise polynomial problems is called XP.

XP vs FPT

The class XP allows algorithms of the form $f(k) \cdot n^{g(k)}$ in contrast to FPT which tries to fix a polynomial constant c: $f(k) \cdot |(x, k)|^c$.
It can be shown: $FPT \subset XP$ by *Time Hierarchy Theorem*.

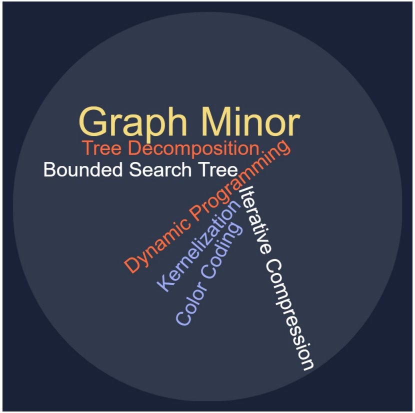
Vertex Cover

- The attentive listener might already have noticed that the introductory problem presented equals the NP-Complete `VERTEX COVER` problem!

`MIN VERTEX COVER` (Cygan et al. 2015)

Input:	Graph G and an Integer k
Question:	Does there exist a set S of vertices of size at most k s.t. $G - S$ is edgeless?
	In other words: Is it possible to cover all edges of G with at most k vertices?

Outlook: Advanced Algorithmic Techniques



- There exists many techniques to deduce fast FPT algorithms.
- *PACE* challenges competitors to solve as many very hard instances as possible: `https://pacechallenge.org/`

Key Takeaways I

If there would be just three things, you should take away...

- Problems that are only exponential in a fixed parameter k while polynomial to the input size are called **Fixed-Parameter Tractable**
- Uses additional information or properties about a specific instance of a problem.
- There exists many different algorithmic techniques to obtain different FPT algorithms.

Part II: Fixed Parameter (In)Tractability & w-Hardness

Stepping Towards Lower-Bounds for FPT

Parametrized Hardness

By Now: Denote $\omega[1]$ as problems that might not expose a FPT algorithm.
Goal: A theory of Intractability for Parametrized Problems

	NP-Hardness	**W[1]-Hardness**
Objects of Study	"Classical" $L \subseteq \{0,1\}^*$	"Parametrized" $L \subseteq \{0,1\}^* \times \mathbb{N}$
Tractability	PTIME	FPT
Hardness Assumption	$\mathtt{SAT} \notin \mathtt{PTIME}$	$\mathtt{CLIQUE}_k \notin \mathtt{FPT}$
Reductions	Poly-Time Karb Reductions	FPT Reductions

Parametrized Reductions

Definition 4: Parametrized Reduction (Cygan et al. 2015, Def 13.1)

Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ two parametrized problems. A *Parametrized Reduction* from A to B is an algorithm that, given an instance (x, k) of A, outputs an instance (x', k') of B such that

- (x, k) is a yes instance of A **iff** (x', k') is a yes instance of B
- $k' \leq g(k)$ for some computable function g
- the running time is $f(k) \cdot |x|^{\mathcal{O}(1)}$ (FPT!)

Parametrized Reductions

Theorem 1: Central Property of Parametrized Reductions (Cygan et al. 2015, Th. 13.2)

If there is a Parametrized Reduction from L to Q and Q is FPT, then L is FPT as well.

Proof: Follows from Definition

- Suppose Q can be solved in FPTTIME $f(l) \cdot |y|^c$ and the reduction $L \leq_{\texttt{FPT}} Q$ takes time $g(k) \cdot |x|^d$
- Then L solves in $f(h(k)) \cdot |g(k) \cdot |x|^d|^c$ as l bounded by $l \leq h(k)$ (Property II) and the instance can not be larger then the duration of the reduction
- The final runtime only exponential in k and polynomial in |x| and hence FPT ■

Parametrized Reductions

Theorem 2: Transitivity (See Cygan et al. 2015, Th. 13.3)

If there are Parametrized Reductions from L to Q and from Q to T, then there is a Parametrized Reduction from L to T.

Proof: Omitted. Again directly from the Definition

Towards a New Hierarchy

(Rather Informal) Definition: The class $\omega[1]$

$\omega[1] := [\texttt{CLIQUE}_k]^{\texttt{FPT}}$ (All problems FPT-reducible to $\texttt{CLIQUE}_k$)

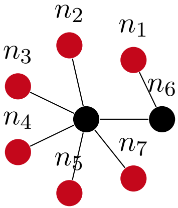
- Problem Q is $\omega[1]$-hard iff $\texttt{CLIQUE}_k$ reduces to it.
- $\texttt{FPT} \subseteq \omega[1]$
- $P = NP \Rightarrow \texttt{FPT} = \omega[1]$
- If $\omega[1]$-hard problem is FPT then also collapse $\texttt{FPT} = \omega[1]$

Recap: Independent Set

INDEPENDENT SET (See Cygan et al. 2015)

Input: Graph G and integer k
Question: Does G has an independent set of size k?

In other words: Is there a vertex-set S of size k that are non-adjacent?



A First Trivial Reduction: $\texttt{CLIQUE}_k \leq_{\texttt{FPT}} \texttt{IS}_k$

- It is known: Graph G has `IS` of size k if and only if G^{-1} has a `CLIQUE` of size k.
- Therefore: $(G, k) \longrightarrow (G^{-1}, k)$ directly gives the desired reduction (Even in PTIME!).

Why Standard Reductions Not Always Work: $\mathtt{VC}_k \leq_{\mathtt{FPT??}} \mathtt{IS}_k$

- Again, it is known: X is a $\mathtt{VC}$ if and only if $G - X$ is an $\mathtt{IS}$.
- Therefore: $(G, k) \longrightarrow (G, n - k)$ gives indeed a reduction, **but**
- The parameter k is not bounded any more just by k!

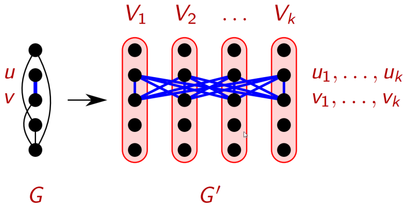
Multicolored Clique

MULTICOLORED CLIQUE (PARTITIONED CLIQUE) Cygan et al. 2015, p. 428

Input:	Graph G, integer k, partition $(V_1, ...V_k)$
Question:	Does G has a k-Clique containing **exactly** one vertex from each set V_i?

$\texttt{CLIQUE}_k \leq_{\text{FPT}} \texttt{MULTICOLORED CLIQUE}_k$

$$\forall i \neq j : u_i v_j \in E(G') \Leftrightarrow uv \in E(G)$$



Claim: G' has MULTICOLORED CLIQUE **iff** G has CLIQUE of size k

Proof

- $\texttt{CLIQUE}_k \Rightarrow \texttt{MCLIQUE}_k$: Distribute original clique to partitions
- $\texttt{MCLIQUE} \Rightarrow \texttt{CLIQUE}_k$: Project them back to a set of vertices of G

Therefore: $(G, k) \longrightarrow (G', k')$ is a FPT reduction.

Notes

- For proofing $\omega[1]$ hardness, it is often useful to start with a $\texttt{MULTICOLORED CLIQUE}$
- Similar, there is a FPT reduction from $\texttt{INDEPENDENT SET} \leq_{\text{FPT}} \texttt{MULTICOLORED INDEPENDENT SET}$

Dominating Set

DOMINATING SET (Cygan et al. 2015)

Input: Graph G and Integer k
Question: Is there X of size k s.t. $N[X] = V(G)$?

Where we denote $N[X]$ as the **close neighborhood** of X

Corollary: DOMINATING SET is $\omega[1]$-hard and $\omega[2]$-complete

Intuition About Differences of $\omega[1]$ and $\omega[2]$

Formulating CLIQUE and DOMINATING SET in logical terms [2]

- X is a CLIQUE iff

$$\forall_{\mathrm{(u,v)\notin G}} : \neg(u \in X \wedge v \in X)$$

- X is a DOMINATING SET iff

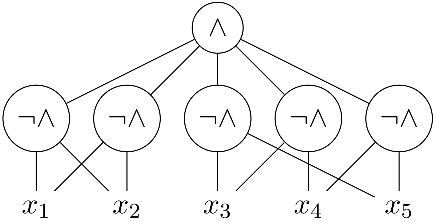
$$\forall u \exists v : (v \in X \wedge (u, v) \in E(G)) \vee u = v$$

Recall: The Polynomial Hierarchy also defined via quantifiers.
Maybe something similiar applies also in the FPT case?

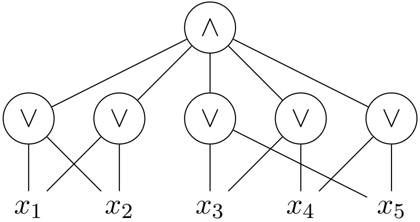
[2] More precisely: Monadic Second-Order-Logic of Graphs

Intuition About Differences of $\omega[1]$ and $\omega[2]$

CLIQUE



DOMINATING SET



Quick Outlook: Weighted Circuit Satisfiability

`WEIGHTED CIRCUIT SATISFIABILITY(WCS)` (Cygan et al. 2015)

Input:	Circuit C and Integer k
Question:	Is there a satisfying valuation of the input **with exactly k ones**?

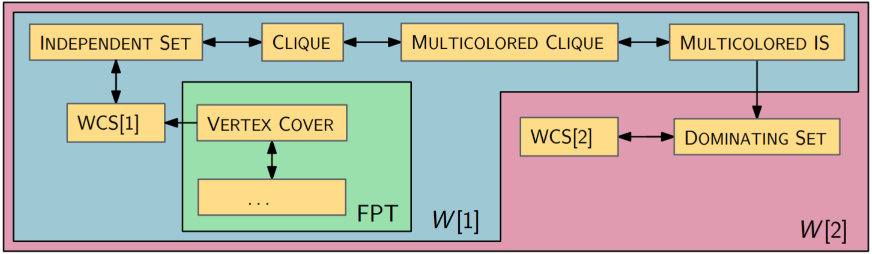
Definition: $\omega[t] :=$ Problems FPT-Reducible to `WCS` for circuits of

- Constant Depth and
- Weft at most t.

The *Weft* of a Circuit is the number of nodes with a $\texttt{fanin} > 2$

It turns out that $FPT \subseteq \omega[1] \subseteq \omega[2] \subseteq ... \subseteq XP$ while all these inclusions are strict, if ETH (Next Chapter!) is true.

Summary FPT Reductions[3]



3 (Comp. Würzburg 2019)

Key Takeaways II

If there would be just three things, you should take away...

- $\texttt{VertexCover}_k \in \texttt{FPT}$, $\texttt{Clique}_k \in \omega[1]$ and $\texttt{DominatingSet}_k \in \omega[2]$
- The class NP splits up into a whole hierarchy of more $[i]$ classes
- If you get an unknown problem, chances are high to be successful with a $\texttt{MULTICOLORED CLIQUE}$ reduction.

Part III: A Stronger Assumption: The (Strong) Exponential Time Hypothesis

Proving Lower Bounds for Subexponential Time

Why We Need a New a Assumption

We already know for the **Vertex Cover** problem:

1. $2^n \cdot \mathrm{poly}(n)$ by brute-forcing all sets
2. $2^k \cdot \mathrm{poly}(n)$ by *branching* (Introduction)

But for example: **Planar Vertex Cover** can be solved in

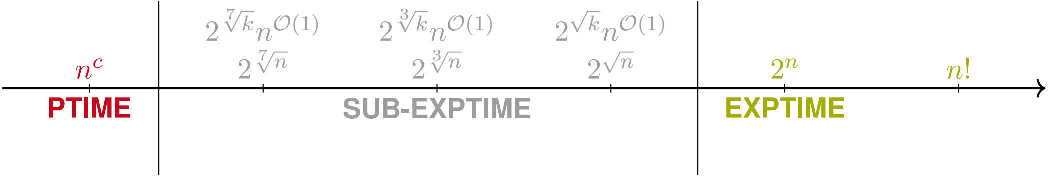
1. $2^{\mathcal{O}(\sqrt{n})}$ by a given Tree-Decomposition + Dynamic Programming [4]

[4] Because $tw(G) \leq \sqrt{n}$ for planar G. Recent 2-Approximation Algorithm see (Belbasi and Martin 2021)

Fine-Grained Complexity

Sad News: The fundamental assumption $\mathbf{P} \neq \mathbf{NP}$ rules out all attempts for finding a PTIME algorithm for NP-c problems.

Question: Can we at least **hope** for a *Sub-Exponential* Time Algorithm for NP-Complete problems?



Solution: A new assumption based again on the *Hardness of SAT*

ETH and SETH

Exponential Time Hypothesis

There is $\delta > 0$ s.t. $\texttt{3SAT}$ can not be solved in time $\mathcal{O}(2^{\delta n}) = \mathcal{O}((2^{\delta})^{n})$

Strong Exponential Time Hypothesis

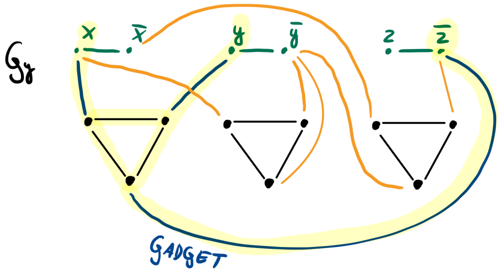
For every $\delta < 1$ there is q s.t. $\texttt{qSAT}$ cannot be solved in time $\mathcal{O}(2^{\delta n})$

1. $\texttt{ETH} \Rightarrow \texttt{3SAT}$ can not beat $2^{\mathcal{O}(n)}$
2. $\texttt{SETH} \Rightarrow \texttt{ETH}$ (Proof: Cygan et al. 2015, Theorem 14.5)
3. $\texttt{ETH}$ is commonly believed, $\texttt{SETH}$ still discussed.

Transfering Lower Bounds by Reductions

Observe the **Textbook** reduction (e.g. Schardl 2009) from $\texttt{3SAT} \leq \texttt{VERTEX COVER}$:

For example given: $\phi = (x \vee y \vee \overline{z}) \wedge (x \vee \overline{y}) \wedge (\overline{x} \vee \overline{zy})$



$\texttt{3SAT} \leq \texttt{VERTEX COVER}$: Analysis

$$\begin{bmatrix} \text{Formula } \phi \\ n \ \texttt{Variables} \\ m \ \texttt{Clauses} \end{bmatrix} \rightsquigarrow \begin{bmatrix} (G,k) \\ 2n+3m \ \texttt{Vertices} \\ n+6m \ \texttt{Edges} \\ k = 2n+m \end{bmatrix}$$

- $\#\texttt{clauses}_{\texttt{3SAT}} = m = \binom{2N}{3} \in \mathcal{O}(n^3)$
- $\Rightarrow$ size of the whole instance $N, M \in \mathcal{O}(n^3)$

Corollary

Assume $\texttt{VERTEX COVER}$ can be solved in time $2^{o(\sqrt[3]{N+M})}$
$\Rightarrow$ $\texttt{3SAT}$ could be solved in $2^{o(n)}$ by *pipelining* the reduction. ↯ Contradicting ETH

Nice. But can we do better?

Towards a Tight Bound: Sparsification Lemma

Idea: If we tighten $m = \mathcal{O}(n)$ in the input instance, then we would get $2^{o(N+M)}$ by the same (linear) reduction

Outlook: Sparsification Lemma (Impagliazzo 1999)

For all $\epsilon > 0$, there is a constant K s.t. we can compute for every formula ϕ in $\texttt{3CNF}$ with n clauses over k variables an equivalent formula $\bigvee_{t=1}^{t} \psi_i$ where each ψ_i is in $\texttt{3CNF}$ and over the same k variables and has $\leq K \cdot k$ clauses. Moreover, $t \leq 2^{\epsilon k}$ and the computation takes $\mathcal{O}(2^{\epsilon k} n^c)$ time

Proof: Omitted, but idea: Branching over a set of variables.

Sketch: Turning Back to Our Reduction

1. Using **Sparsification Lemma** we can *sparsify* our kCNF-formula to **linear size** in clauses with an appropriate ϵ.
2. We apply our reduction from $\texttt{3SAT} \leq \texttt{VERTEX COVER}$
3. Total Runtime now lowerly bounded by $2^{o(N+M)}$

More Tight Bounds For Classical Problems

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

- `INDEPENDENT SET`
- `CLIQUE`
- `DOMINATING SET`
- `VERTEX COVER`
- `HAMILTONIAN PATH`
- `FEEDBACK VERTEX SET`

Crucial Consequence for FPT Algorithms

Observation: No $2^{o(n)}$-Time Algorithm $\Rightarrow$ also no $2^{o(k)} \cdot n^{\mathcal{O}(1)}$-Time Algorithm

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

- k-INDEPENDENT SET
- k-CLIQUE
- k-DOMINATING SET
- k-VERTEX COVER
- k-HAMILTONIAN PATH
- k-FEEDBACK VERTEX SET

Last Slide: Lower Bounds for $\omega[1]$ hard problems

We can even go further:

Theorem Chen, Eickmeyer, and Flum 2004

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

Implying that we can not have **any FPT algorithms** for

- SET COVER, HITTING SET, CONNECTED DOMINATING SET, PARTIAL VERTEX COVER, ...

unless ETH fails.
This closes the cycle back to our $\omega[i]$-hierarchy.

Key Takeaways III

If there would be just three things, you should take away...

- ETH: $\exists \delta > 0$ s.t. $\texttt{3SAT}$ can not be solved in time $\mathcal{O}(2^{\delta n})$
- Lower Bounds (under ETH) can be transferred using reductions
- That I thank all of you very much for the attention!

References I

Belbasi, Mahdi and Martin Fürer Martin (2021). *Finding All Leftmost Separators of Size $\leq k$*. eprint: 2111.02614. URL: https://arxiv.org/pdf/2111.02614.pdf.

Chen, Yijia, Kord Eickmeyer, and Jörg Flum (2004). *The exponential time hypothesis and the parameterized clique problem*. URL: http://basics.sjtu.edu.cn/~chen/papers/eth.pdf.

Cormen, Thomas H. et al. (2009). *Introduction to Algorithms, 3rd Edition*. MIT Press. ISBN: 978-0-262-03384-8. URL: http://mitpress.mit.edu/books/introduction-algorithms.

Cygan, Marek et al. (2015). *Parameterized Algorithms*. Berlin, Heidelberg: Springer. ISBN: 978-3-319-21275-3.

Impagliazzo Russel;Paturi, Ramamohan (1999). *Which Problems Have Strongly Exponential Complexity*. URL: https://www.sciencedirect.com/science/article/pii/S002200000191774X?via%5C%3Dihub.

References II

Schardl, TB (2009). *Reductions*. URL: http://web.mit.edu/~neboat/www/6.046-fa09/rec8.pdf.

Würzburg, Universität (2019). *Exakte Algorithmen, Vorlesung 9: Lower Bounds*. URL: https://wuecampus2.uni-wuerzburg.de/moodle/pluginfile.php/1564087/mod_resource/content/3/vl09-part2-printerversion.pdf.