Fixed Parameter Algorithms

Dániel Marx

Tel Aviv University, Israel

Open lectures for PhD students in computer science

January 9, 2010, Warsaw, Poland

Parameterized complexity

- **Parameterized problem:** a parameter k is associated with each input instance.
- A parameterized problem is **fixed-parameter tractable (FPT)** if it can be solved in time $f(k) \cdot n^c$ for some function f depending only on k and constant c **not** depending on k.
- For some important parameterized problems, for example k-CLIQUE and k-INDEPENDENT SET, no FPT algorithm is known.
- Can we show that these problems are **not** FPT?
- This would require to show that P $\neq$ NP: if P $=$ NP, then k-CLIQUE is polynomial-time solvable, hence FPT.
- Can we give some evidence that certain problems are not FPT?

Classical complexity

Nondeterministic Turing Machine (NTM): single tape, finite alphabet, finite state, head can move left/right only one cell. In each step, the machine can branch into an arbitrary number of directions. Run is successful if at least one branch is successful.

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

Polynomial-time reduction from problem P to problem Q: a function ϕ with the following properties:

- $\phi(x)$ can be computed in time $|x|^{O(1)}$,
- $\phi(x)$ is a yes-instance of Q if and only if x is a yes-instance of P.

Definition: Problem Q is NP-hard if any problem in NP can be reduced to Q.

If an NP-hard problem can be solved in polynomial time, then every problem in NP can be solved in polynomial time (i.e., P $=$ NP).

Part I:

Reductions and the W-hierarchy

Parameterized complexity

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

- An appropriate notion of reduction.
- An appropriate hypothesis.

Polynomial-time reduction is not good for our purposes.

Example: Graph G has an independent set k if and only if it has a vertex cover of size $n-k$.

$\Rightarrow$ Transforming an INDEPENDENT SET instance (G,k) into a VERTEX COVER instance $(G,n-k)$ is a correct polynomial-time reduction.

However, VERTEX COVER is FPT, but INDEPENDENT SET is not known to be FPT.

Parameterized reduction

Parameterized reduction from problem P to problem Q: a function ϕ with the following properties:

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

Fact: If there is a parameterized reduction from problem P to problem Q and Q is FPT, then P is also FPT.

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.

A reduction

Fact: There is a parameterized reduction from INDEPENDENT SET to DOMINATING SET.

Proof: Let G be a graph with n vertices, m edges, and let k be an integer. We construct a graph H such that G has an independent set of size k if and only if H has a dominating set of size k.

The dominating set has to contain one vertex from each of the k cliques. Additional vertices ensure that these selections describe an independent set.

(See the blackboard.)

Basic hypotheses

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

Let us choose a basic hypothesis:

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

- **Theorists' Hypothesis**: k-STEP HALTING PROBLEM (is there a branch 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

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

Let us choose a basic hypothesis:

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

$$\Updownarrow$$

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

$$\Uparrow$$

- **Exponential Time Hypothesis (ETH)**: n-variable 3SAT cannot be solved in time $2^{o(n)}$.

Which hypothesis is the most plausible?

INDEPENDENT SET *and* k-STEP HALTING PROBLEM

Fact: There is a parameterized reduction from INDEPENDENT SET to the k-STEP HALTING PROBLEM.

Proof: Given a graph G and an integer k, we construct a Turing machine M and an integer $k' = O(k^2)$ such that M halts in k' steps if and only if G has an independent set of size k.

The alphabet of M is the vertices of G.

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

(See the blackboard.)

INDEPENDENT SET *and* k-STEP HALTING PROBLEM

Fact: There is a parameterized reduction from the k-STEP HALTING PROBLEM to INDEPENDENT SET.

Proof: Given a Turing machine M and an integer k, we construct a graph G that has an independent set of size $k' := k^2$ if and only if M halts in k steps.

- G consists of k^2 cliques, thus a k'-independent set has to contain one vertex from each.
- The selected vertex from clique $K_{i,j}$ describes what happens in Step i at cell j: what is written there, is the head there, and if so, what is the state.
- We add edges between the cliques to rule out inconsistencies: head is at more than one location at the same time, wrong character is written, head moves in the wrong direction etc.

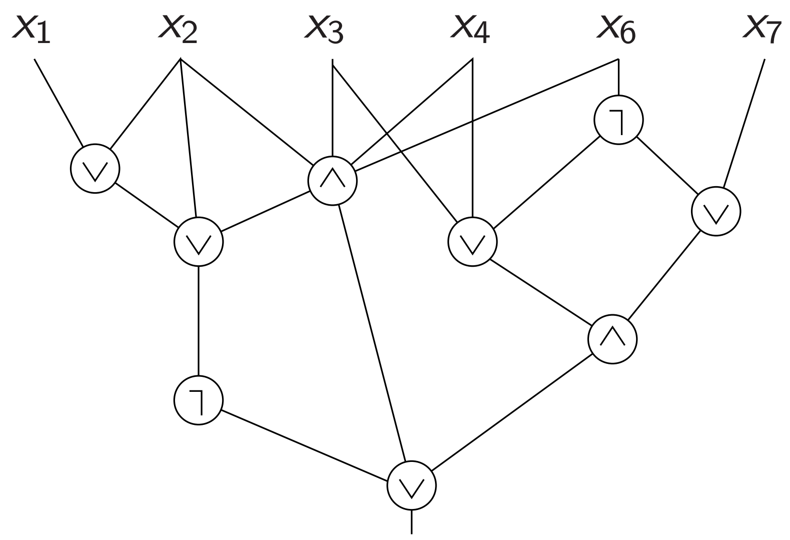
(See the blackboard.)

Summary

- INDEPENDENT SET and k-STEP HALTING PROBLEM can be reduced to each other $\Rightarrow$ Engineers' Hypothesis and Theorists' Hypothesis are equivalent!
- INDEPENDENT SET and k-STEP HALTING PROBLEM can be reduced to DOMINATING SET.
- Is there a parameterized reduction from DOMINATING SET to INDEPENDENT SET?
- Probably not. Unlike in NP-completeness, where most problems are equivalent, here we have a hierarchy of hard problems.
- 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.



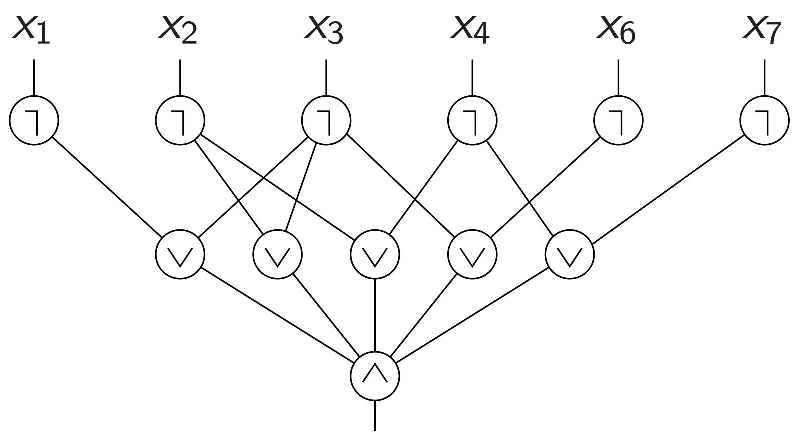
CIRCUIT SATISFIABILITY: Given a Boolean circuit C, decide if there is an assignment on the inputs of C such that the output is true.

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

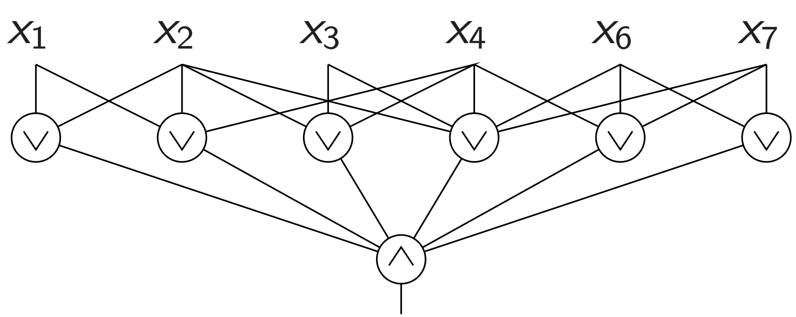
Weight of an assignment: number of true values.

WEIGHTED CIRCUIT SATISFIABILITY

INDEPENDENT SET can be reduced to WEIGHTED CIRCUIT SATISFIABILITY:

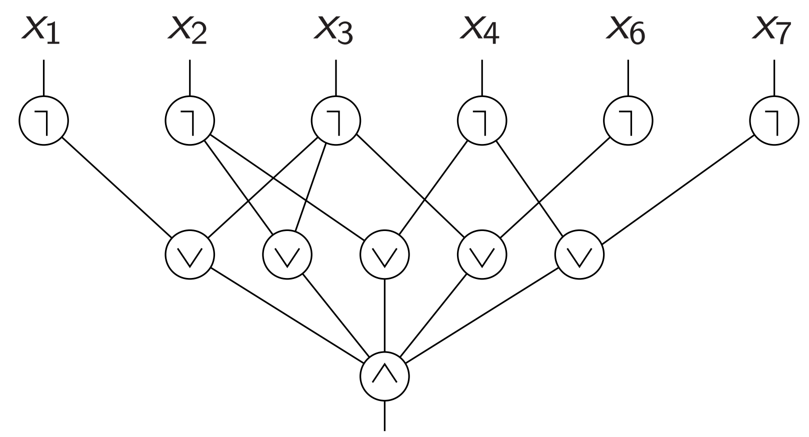


DOMINATING SET can be reduced to WEIGHTED CIRCUIT SATISFIABILITY:

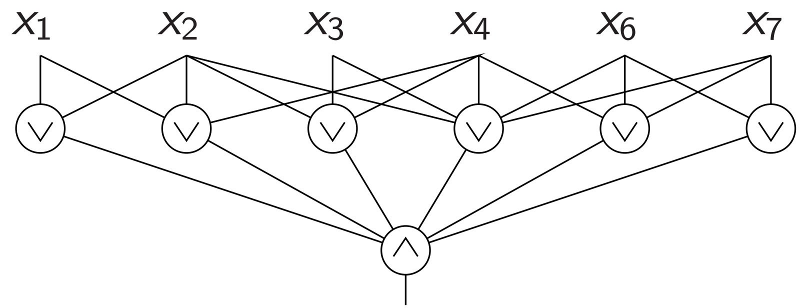


WEIGHTED CIRCUIT SATISFIABILITY

INDEPENDENT SET can be reduced to WEIGHTED CIRCUIT SATISFIABILITY:



DOMINATING SET can be reduced to WEIGHTED CIRCUIT SATISFIABILITY:



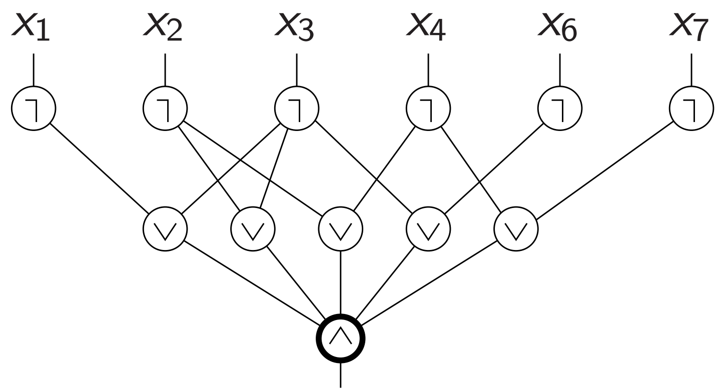
To express DOMINATING SET, we need more complicated circuits.

Depth and weft

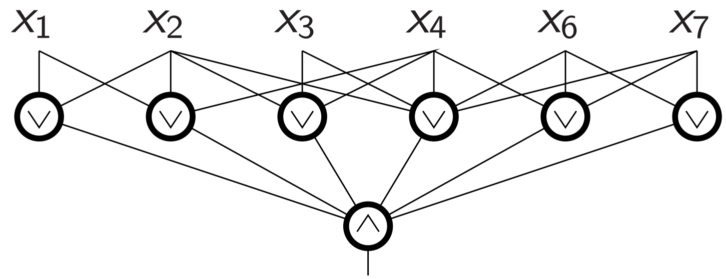
The **depth** of a circuit is the maximum length of a path from an input to the output.

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 in W[1]-complete.
Fact: DOMINATING SET is in 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].

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.

Fact: MULTICOLORED CLIQUE is W[1]-hard.

Proof by reduction from CLIQUE (see blackboard).

List Coloring

List Coloring is a generalization of ordinary vertex coloring: given a graph G, a set of colors C, and a list $L(v) \subseteq C$ for each vertex v, the task is to find a coloring c where $c(v) \in L(v)$ for every v.

Fact: Vertex Coloring is FPT parameterized by treewidth.

However, list coloring is more difficult:

Fact: List Coloring is W[1]-hard parameterized by treewidth.

LIST COLORING

Fact: LIST COLORING is W[1]-hard parameterized by treewidth.

Proof: By reduction from MULTICOLORED CLIQUE.

- Let G be a graph with color classes $V_1, \ldots, V_k$.
- In the LIST COLORING instance, the set C of colors is the set of vertices of G.
- The colors of vertices $u_1, \ldots, u_k$ select the k vertices of the clique, hence we set $L(u_i) = V_i$.
- If $x \in V_i$ and $y \in V_j$ are not adjacent in G, then we need to ensure that $c(u_i) = x$ and $c(u_j) = y$ are not true at the same time $\Rightarrow$ we add a vertex adjacent to u_i and u_j whose list is $\{x, y\}$.

List Coloring

Fact: List Coloring is W[1]-hard parameterized by treewidth.

Proof: By reduction from Multicolored Clique.

- Let G be a graph with color classes $V_1, \dots, V_k$.
- In the List Coloring instance, the set C of colors is the set of vertices of G.
- The colors of vertices $u_1, \dots, u_k$ select the k vertices of the clique, hence we set $L(u_i) = V_i$.
- If $x \in V_i$ and $y \in V_j$ are not adjacent in G, then we need to ensure that $c(u_i) = x$ and $c(u_j) = y$ are not true at the same time $\Rightarrow$ we add a vertex adjacent to u_i and u_j whose list is $\{x, y\}$.

What about planar graphs?

MULTICOLORED GRID

MULTICOLORED GRID: Given a graph G partitioned into k^2 color classes $V_{i,j}$ ($1 \le i, j \le k$), find a $k \times k$ grid subgraph such that vertex $v_{i,j}$ appears in $V_{i,j}$.

Fact: MULTICOLORED GRID is W[1]-hard.

Proof: By reduction from MULTICOLORED CLIQUE.

- Let G be a graph with color classes $V_1, \ldots, V_k$.
- Each vertex of the constructed graph H is labeled by a **pair** of vertices of G.
- The color class $V_{i,j}$ contains vertex (x, y), $x \in V_i$, $y \in V_j$ if
  - $i = j$ and $x = y$,
  - $i \neq j$ and x, y are adjacent.
- Edges:
  - $(x, y) \in V_{i,j}$ and $(x', y') \in V_{i+1,j}$ are adjacent if $x = x'$.
  - $(x, y) \in V_{i,j}$ and $(x', y') \in V_{i,j+1}$ are adjacent if $y = y'$.

LIST COLORING *for planar graphs*

Fact: LIST COLORING for planar graphs is W[1]-hard parameterized by treewidth.

Proof is the same as the reduction from MULTICOLORED CLIQUE to LIST COLORING, but now the resulting graph is planar.

Part II:

Exponential Time Hypothesis

Exponential Time Hypothesis

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

$$\Updownarrow$$

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

$$\Uparrow$$

- **Exponential Time Hypothesis (ETH)**: n-variable 3SAT cannot be solved in time $2^{o(n)}$.

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

Exponential Time Hypothesis

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

$$\Updownarrow$$

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

$$\Uparrow$$

- **Exponential Time Hypothesis (ETH)**: n-variable 3SAT cannot be solved in time $2^{o(n)}$.

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

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

Exponential Time Hypothesis

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

$$\Updownarrow$$

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

$$\Uparrow$$

- **Exponential Time Hypothesis (ETH)**: n-variable 3SAT cannot be solved in time $2^{o(n)}$.

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

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

We show something much stronger:

Fact: If there is an $f(k) \cdot n^{o(k)}$ time algorithm for k-CLIQUE, then ETH fails.

Lower bound on the exponent

Fact: If there is an $f(k) \cdot n^{o(k)}$ time algorithm for k-CLIQUE, then ETH fails.

We use the following result:

Fact: [Sparsification Lemma]

$$n\text{-variable 3SAT can be solved in time } 2^{o(n)}$$
$$\Updownarrow$$
$$m\text{-clause 3SAT can be solved in time } 2^{o(m)}$$

3-COLORING is NP-complete and there is a polynomial-time reduction from m-clause 3SAT to $O(m)$-vertex 3-COLORING, thus:

Fact: If n-vertex 3-COLORING can be solved in time $2^{o(n)}$, then m-clause 3SAT can be solved in time $2^{o(m)}$ and ETH fails.

Lower bound on the exponent

Fact: If there is an $f(k) \cdot n^{o(k)}$ time algorithm for k-CLIQUE, then ETH fails.

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

Let $f^{-1}(n)$ be the largest integer i such that $f(i) \leq n$.
Function $f^{-1}(n)$ is monotone increasing and unbounded.

Let $k := f^{-1}(n)$. 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.

A k-clique of H corresponds to a proper 3-coloring of G.

$\Rightarrow$ A 3-coloring of G can be found in time
$f(k) \cdot n^{k/s(k)} \leq n \cdot (3^{n/k})^{k/s(k)} = n \cdot 3^{n/s(f^{-1}(n))} = 2^{o(n)}$.

Transferring lower bounds

If we have a lower bound for problem P and there is a parameterized reduction from P to Q, then we get a lower bound for Q as well.

Parameterized reduction from problem P to problem Q: a function ϕ with the following properties:

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

Suppose there is no $f(k) \cdot n^{o(k)}$ algorithm for P.

- If $g(k) = O(k)$, then we know that there is no $f(k) \cdot n^{o(k)}$ time algorithm for Q.
- If $g(k) = O(k^2)$, then we know that there is no ???? algorithm for Q.

Transferring lower bounds

If we have a lower bound for problem P and there is a parameterized reduction from P to Q, then we get a lower bound for Q as well.

Parameterized reduction from problem P to problem Q: a function ϕ with the following properties:

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

Suppose there is no $f(k) \cdot n^{o(k)}$ algorithm for P.

- If $g(k) = O(k)$, then we know that there is no $f(k) \cdot n^{o(k)}$ time algorithm for Q.
- If $g(k) = O(k^2)$, then we know that there is no $f(k) \cdot n^{o(\sqrt{k})}$ algorithm for Q.

Lower bounds for FPT algorithms

We know that VERTEX COVER can be solved in time $O^*(c^k)$.

Can we do it much faster, for example in time $O^*(c^{\sqrt{k}})$ or $O^*(c^{k/\log k})$?

Fact: If VERTEX COVER can be solved in time $2^{o(k)} \cdot n^{O(1)}$, then ETH fails.

Proof: There is a polynomial-time reduction from m-clause 3SAT to $O(m)$-vertex VERTEX COVER. The assumed algorithm would solve the latter problem in time $2^{o(m)} \cdot n^{O(1)}$, violating ETH.

Lower bounds for planar FPT algorithms

Yesterday we have seen that VERTEX COVER, INDEPENDENT SET, DOMINATING SET can be solved in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$ on planar graphs.

Can we get much better dependence on k?

Fact: if VERTEX COVER, INDEPENDENT SET, or DOMINATING SET can be solved in time $2^{o(\sqrt{k})} \cdot n^{O(1)}$ for planar graphs, then ETH fails.

Proof: There are polynomial-time reductions from m-clause 3SAT to $O(m^2)$-vertex instances of these problems. Thus a $2^{o(\sqrt{k})} \cdot n^{O(1)}$ time algorithm would solve m-clause 3SAT in time $2^{o(\sqrt{m^2})} \cdot n^{O(1)} = 2^{o(m)}$, violating ETH.

Part III:

Approximation schemes

Approximation schemes

Polynomial-time approximation scheme (PTAS):

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

- **PTAS:** running time is $|x|^{f(1/\epsilon)}$
- **EPTAS:** (Efficient PTAS) running time is $f(1/\epsilon) \cdot |x|^{O(1)}$
- **FPTAS:** (Fully polynomial approximation scheme) running time is $(1/\epsilon)^{O(1)} \cdot |x|^{O(1)}$

Yesterday, we have seen an EPTAS for Independent Set on planar graphs.

For some problems, there is a PTAS, but no EPTAS is known. Can we show that no EPTAS is possible?

Standard parameterization

Given an **optimization** problem we can turn it into a **decision** problem: the input is a pair (x, k) and we have to decide if there is a solution for x with cost at least/at most k.

The **standard parameterization** of an optimization problem is the associated decision problem, with the value k appearing in the input being the parameter.

Example:

VERTEX COVER

Input:	(G, k)
Parameter:	k
Question:	Is there a vertex cover of size at most k?

If the standard parameterization of an optimization problem is FPT, then (intuitively) it means that we can solve it efficiently if the optimum is small.

No EPTAS

Fact: If the standard parameterization of an optimization problem is W[1]-hard, then there is no EPTAS for the optimization problem, unless FPT $=$ W[1].

Proof: Suppose an $f(1/\epsilon) \cdot n^{O(1)}$ time EPTAS exists. Running this EPTAS with $\epsilon := 1/(k+1)$ decides if the optimum is at most/at least k.

No EPTAS

Fact: If the standard parameterization of an optimization problem is W[1]-hard, then there is no EPTAS for the optimization problem, unless FPT $=$ W[1].

Proof: Suppose an $f(1/\epsilon) \cdot n^{O(1)}$ time EPTAS exists. Running this EPTAS with $\epsilon := 1/(k+1)$ decides if the optimum is at most/at least k.

Thus W[1]-hardness results immediately show that (assuming W[1] $\neq$ FPT)

- No EPTAS for INDEPENDENT SET for unit disks/squares.
- No EPTAS for DOMINATING SET for unit disks/squares.
- No EPTAS for planar TMIN, TMAX, MPSAT.
- No EPTAS for CLOSEST STRING.

Part IV:

Lower bounds for kernels

Kernelization

Definition: **Kernelization** is a polynomial-time transformation that maps an instance (I, k) to an instance (I', k') such that

- (I, k) is a yes-instance if and only if (I', k') is a yes-instance,
- $k' \leq k$, and
- $|I'| \leq f(k)$ for some function $f(k)$.

Kernelization

Definition: **Kernelization** is a polynomial-time transformation that maps an instance (I, k) to an instance (I', k') such that

- (I, k) is a yes-instance if and only if (I', k') is a yes-instance,
- $k' \leq k$, and
- $|I'| \leq f(k)$ for some function $f(k)$.

Simple fact: If a problem has a kernelization algorithm, then it is FPT.

Proof: Solve the instance (I', k') by brute force.

Kernelization

Definition: **Kernelization** is a polynomial-time transformation that maps an instance (I, k) to an instance (I', k') such that

- (I, k) is a yes-instance if and only if (I', k') is a yes-instance,
- $k' \leq k$, and
- $|I'| \leq f(k)$ for some function $f(k)$.

Simple fact: If a problem has a kernelization algorithm, then it is FPT.
Proof: Solve the instance (I', k') by brute force.

Converse: Every FPT problem has a kernelization algorithm.
Proof: Suppose there is an $f(k)n^c$ algorithm for the problem.

- If $f(k) \leq n$, then solve the instance in time $f(k)n^c \leq n^{c+1}$, and output a trivial yes- or no-instance.
- If $n < f(k)$, then we are done: a kernel of size $f(k)$ is obtained.

Polynomial kernels

Asking which problems have kernels is not interesting: it is the same as asking which problems are FPT.

A more relevant question: which problems have **polynomial kernels** (i.e., the size of instance I' is $O(k^c)$ for some constant c)?

We have seen some polynomial kernels:

- $3k$-vertex kernel for VERTEX COVER
- k^2 kernel for COVERING POINTS WITH LINES
- k^d kernel for d-HITTING SET

But if the problem is FPT by some other technique (color coding, iterative compression, etc.), then it is not clear whether it has a polynomial kernel.

Kernel lower bounds

Recall: k-PATH can be solved in (randomized) time $O^*((2e)^k)$ by color coding.

Very recent result (2008): k-PATH has no poly kernel, unless coNP $\subseteq$ NP/poly and the polynomial hierarchy collapses.

Similar results for other problems: under the same assumption, no polynomial kernel for

- k-CYCLE
- STEINER TREE
- CONNECTED VERTEX COVER
- VERTEX DISJOINT PATHS
- ...

Very-very recent result: VERTEX COVER has no $O(k^{2-\epsilon})$ kernel, unless ...

Note: The $3k$-vertex kernel has size $O(k^2)$.

k-PATH

Intuition why k-PATH has no polynomial kernel (not a proof!).

Suppose that k-PATH has a kernel of size k^c.

Set $t = k^c + 1$ and consider t instances (G_1, k), ... , (G_t, k) with the same parameter k.

The instance $(G_1 \cup \ldots G_t, k)$ is a yes-instance if and only if **at least** one (G_i, k) is a yes-instance.

Kernelization gives an instance of $k^c < t$ bits. Less than one bit per original instance. Intuitively, we managed to solve at least one instance.

OR-distillation algorithms

An **OR-distillation algorithm** for a problem P is an algorithm with the following properties:

- The input is a sequence $I_1, \ldots, I_t$ of instances of P.
- The running time is polynomial.
- The output is an instance O of P with
  - $|O| \leq \max_{i=1}^{t} \text{poly}(|I_1|)$
  - O is a yes-instance $\Leftrightarrow$ at least one I_i is a yes instance.

We are able to compress arbitrarily many instances into a single instance. Should not be possible for NP-hard problems.

Fact: If an NP-hard problem has an OR-distillation algorithm, then coNP $\subseteq$ NP/poly and the polynomial hierarchy collapses.

Proof for k-PATH

Fact: k-PATH has no poly kernel, unless coNP $\subseteq$ NP/poly and the polynomial hierarchy collapses.

We show that if k-PATH has a polynomial kernel, then it has an OR-distillation.

- Suppose we have t instances, each of size n.
- Group them by the parameter.
- Make each group a single graph (with many components) $\Rightarrow$ n instances.
- Kernelize each group $\Rightarrow$ n instance, each of size poly(n).
- Asking if at least one instance is YES is a problem in NP $\Rightarrow$ As k-PATH is NP-complete, we can construct a k-PATH instance of size poly(n) answering this question $\Rightarrow$ OR-distillation.

OR-composition

What properties of k-PATH were used in the proof?

- It is NP-hard.
- By taking the union, we can join instances with the same parameter into a single instance.

OR-composition

What properties of k-PATH were used in the proof?

- It is NP-hard.
- By taking the union, we can join instances with the same parameter into a single instance.

An **OR-composition algorithm** formalizes the second property:

- The input is a sequence $I_1, \ldots, I_t$ of instances with the same parameter k.
- The running time is polynomial.
- The output is an instance O with parameter k' and
  - $k' \leq \text{poly}(k)$
  - O is a yes-instance $\Leftrightarrow$ at least one I_i is a yes instance.

Fact: NP-hard + OR-composition $\Rightarrow$ OR-distillation $\Rightarrow$ No poly kernel, unless …

AND-composition

We can define AND-composition and AND-distillation in a similar way:
they create one instance that is a yes-instance if and only if **every** input instance is a yes-instance.

Example: TREEWIDTH (given a graph G and an integer k, is the treewidth of G at most k?) has an AND-composition: The union of graphs $G_1, \ldots, G_t$ has treewidth at most k if and only if every G_i has treewidth at most k.

- It is conjectured that NP-complete problems have no AND-distillation, but currently no result similar to OR-distillation is known.
- Such a result could be used to show that TREEWIDTH has no polynomial kernel.

Parameterized complexity

- Possibility to give evidence that certain problems are not FPT.
- Parameterized reduction.
- The W-hierarchy.
- ETH gives much stronger and tighter lower bounds.
- PTAS vs. EPTAS
- Kernel lower bounds.