An Introduction to Parameterized Complexity

Dimitris Chatzidimitriou

Advanced Topics on Algorithms and Complexity

Corelab

June 30, 2014

Part I

Introduction

FPT

para-NP

XP

Parameterized Complexity... a new notion of feasibility?

Let's revisit some classic NP-complete problems.

GRAPHKA

Why parameterized complexity?

Parameterized Complexity... a new notion of feasibility?

Let's revisit some classic NP-complete problems.

Three NP-complete problems

Vertex Coloring
Instance: A graph G and an integer $k \geq 0$.
Question: $\exists\sigma : V(G) \to \{1, \ldots, k\} : \forall\{v, u\} \in E(G)\ \sigma(v) \neq \sigma(u)$?

It can be solved in $O(n^2 \cdot k^n)$ steps.

GRAPHKA

Vertex Coloring

Instance: A graph G and an integer $k \geq 0$.

Question: $\exists \sigma : V(G) \rightarrow \{1, \ldots, k\} : \forall \{v, u\} \in E(G)\ \sigma(v) \neq \sigma(u)$?

It can be solved in $O(n^2 \cdot k^n)$ steps.

Three NP-complete problems

Independent Set

Instance: A graph G and an integer $k \geq 0$.

Question: $\exists S \in V(G)^k : \forall e \in E(G)\ |e \cap S| \leq 1$?

It can be solved in $O(n^{k+1})$ steps.

Three NP-complete problems

Independent Set
Instance: A graph G and an integer $k \geq 0$.
Question: $\exists S \in V(G)^k : \forall e \in E(G)\ |e \cap S| \leq 1$?

It can be solved in $O(n^{k+1})$ steps.

GRAPHKA

Three NP-complete problems

Vertex Cover
Instance: A graph G and an integer $k \geq 0$.
Question: $\exists S \in V(G)^k : \forall e \in E(G) \; |e \cap S| \geq 1$?

It can be solved in $(1.2738)^k + O(n)$ steps. *(Chen, Kanj, Xia. 2010)*

GRAPHKA

Three NP-complete problems

Vertex Cover
Instance: A graph G and an integer $k \geq 0$.
Question: $\exists S \in V(G)^k : \forall e \in E(G)\ |e \cap S| \geq 1$?

It can be solved in $(1.2738)^k + O(n)$ steps. *(Chen, Kanj, Xia. 2010)*

GRAPHKA

Parameterized problems

Notice that, for fixed values of k, Vertex Cover can be solved in linear time, Independent Set in polynomial, while Vertex Coloring still needs exponential time.

Given an alphabet Σ, a *parameterization* of Σ^* is a recursive mapping $\kappa : \Sigma^* \to \mathbb{N}$.

A *parameterized problem* (with respect to Σ) is a pair (L, κ) where $L \subseteq \Sigma^*$ and κ is a parameterization of Σ^*.

GRAPHKA

Notice that, for fixed values of k, VERTEX COVER can be solved in linear time, INDEPENDENT SET in polynomial, while VERTEX COLORING still needs exponential time.

Given an alphabet Σ, a *parameterization* of Σ^* is a recursive mapping $\kappa : \Sigma^* \to \mathbb{N}$.

A *parameterized problem* (with respect to Σ) is a pair (L, κ) where $L \subseteq \Sigma^*$ and κ is a parameterization of Σ^*.

GRAPHKA

A parameterization κ of SAT:

$$\kappa(x) = \begin{cases} \text{number of variables in } x, & \text{if } x \text{ is a valid encoding} \\ 1, & \text{otherwise} \end{cases}$$

κ defines the following parameterized problem:

p-SAT
Instance: A propositional formula ϕ.
Parameter: The number of variables in ϕ.
Question: Is ϕ satisfiable?

Parameterized Satisfiability

A parameterization κ of SAT:

$$\kappa(x) = \begin{cases} \text{number of variables in } x, & \text{if } x \text{ is a valid encoding} \\ 1, & \text{otherwise} \end{cases}$$

κ defines the following parameterized problem:

p-SAT
Instance: A propositional formula ϕ.
Parameter: The number of variables in ϕ.
Question: Is ϕ satisfiable?

GRAPHKA

Other examples

A parameterization of Independent Set can be defined as $\kappa(G, k) = k$.

We can do the same with all the problems that have some integer in their instances, such as Vertex Coloring or Vertex Cover.

That way, we define the parameterized problems p-Vertex Coloring and p-Vertex Cover.

Other examples

A parameterization of Independent Set can be defined as $\kappa(G, k) = k$.

We can do the same with all the problems that have some integer in their instances, such as Vertex Coloring or Vertex Cover.

That way, we define the parameterized problems p-Vertex Coloring and p-Vertex Cover.

Other examples

A parameterization of Independent Set can be defined as $\kappa(G, k) = k$.

We can do the same with all the problems that have some integer in their instances, such as Vertex Coloring or Vertex Cover.

That way, we define the parameterized problems p-Vertex Coloring and p-Vertex Cover.

New & exciting complexity classes!

- The classes FPT, para-NP, XP
- The classes W[P] and W[SAT]
- The classes W[1], W[2],...
- The classes A[P] and A[SAT]
- The classes A[1], A[2],...

Given an alphabet Σ and a parameterization $\kappa : \Sigma^* \to \mathbb{N}$,

(A) An algorithm $\mathcal{A}$ is a FPT-*algorithm with respect to* κ if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ and a polynomial function $p : \mathbb{N} \to \mathbb{N}$ such that for every $x \in \Sigma^*$, the algorithm $\mathcal{A}$ requires

$$\leq f(\kappa(x)) \cdot p(|x|) \text{ steps}$$

(B) A parameterized problem (L, κ) is *fixed parameter tractable* if there exists an FPT-algorithm with respect to κ that decides L. We will then say that $(L, \kappa) \in$ FPT.

p-SAT is in FPT while *p*-Vertex Coloring is not!

GRAPHKA

THE CLASS FPT

Given an alphabet Σ and a parameterization $\kappa : \Sigma^* \to \mathbb{N}$,

(A) An algorithm $\mathcal{A}$ is a FPT-*algorithm with respect to* κ if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ and a polynomial function $p : \mathbb{N} \to \mathbb{N}$ such that for every $x \in \Sigma^*$, the algorithm $\mathcal{A}$ requires

$$\leq f(\kappa(x)) \cdot p(|x|) \text{ steps}$$

(B) A parameterized problem (L, κ) is *fixed parameter tractable* if there exists an FPT-algorithm with respect to κ that decides L. We will then say that $(L, \kappa) \in \text{FPT}$.

p-SAT is in FPT while p-VERTEX COLORING is not!

GRAPHKA

THE CLASS FPT

Given an alphabet Σ and a parameterization $\kappa : \Sigma^* \to \mathbb{N}$,

(A) An algorithm $\mathcal{A}$ is a FPT-*algorithm with respect to* κ if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ and a polynomial function $p : \mathbb{N} \to \mathbb{N}$ such that for every $x \in \Sigma^*$, the algorithm $\mathcal{A}$ requires

$$\leq f(\kappa(x)) \cdot p(|x|) \text{ steps}$$

(B) A parameterized problem (L, κ) is *fixed parameter tractable* if there exists an FPT-algorithm with respect to κ that decides L. We will then say that $(L, \kappa) \in$ FPT.

p-SAT is in FPT while p-VERTEX COLORING is not!

GRAPHKA

The class FPT

Given an alphabet Σ and a parameterization $\kappa : \Sigma^* \to \mathbb{N}$,

(A) An algorithm $\mathcal{A}$ is a *FPT-algorithm with respect to κ* if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ and a polynomial function $p : \mathbb{N} \to \mathbb{N}$ such that for every $x \in \Sigma^*$, the algorithm $\mathcal{A}$ requires

$$\leq f(\kappa(x)) \cdot p(|x|) \text{ steps}$$

(B) A parameterized problem (L, κ) is *fixed parameter tractable* if there exists an FPT-algorithm with respect to κ that decides L. We will then say that $(L, \kappa) \in \text{FPT}$.

p-SAT is in FPT while p-VERTEX COLORING is not!

GRAPHKA

FPT-reductions

Let (L, κ) and (L', κ') be parameterized problems
(with respect to the alphabets Σ and Σ').

A FPT-reduction from (L, κ) to (L', κ'), is a mapping $R : \Sigma^* \to (\Sigma')^*$ where

1. $\forall x \in \Sigma^* \; : \; x \in L \Leftrightarrow R(x) \in L'$
2. R is computable by an FPT-algorithm (with respect to κ)
   [i.e. R is computable in $f(\kappa(x)) \cdot p(|x|)$ steps]
3. there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that
   $\forall x \in \Sigma^* : \kappa'(R(x)) \leq g(\kappa(x))$

Observation: The class FPT is closed under FPT-reductions.

GRAPHKA

FPT-reductions

Let (L, κ) and (L', κ') be parameterized problems (with respect to the alphabets Σ and Σ').

A FPT-reduction from (L, κ) to (L', κ'), is a mapping $R : \Sigma^* \to (\Sigma')^*$ where

1. $\forall x \in \Sigma^* \;:\; x \in L \Leftrightarrow R(x) \in L'$
2. R is computable by an FPT-algorithm (with respect to κ)
   [i.e. R is computable in $f(\kappa(x)) \cdot p(|x|)$ steps]
3. there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that
   $\forall x \in \Sigma^* : \kappa'(R(x)) \leq g(\kappa(x))$

Observation: The class FPT is closed under FPT-reductions.

GRAPHKA

FPT-reductions

Let (L, κ) and (L', κ') be parameterized problems
(with respect to the alphabets Σ and Σ').

A FPT-reduction from (L, κ) to (L', κ'), is a mapping $R : \Sigma^* \to (\Sigma')^*$ where

1. $\forall x \in \Sigma^* \;:\; x \in L \Leftrightarrow R(x) \in L'$
2. R is computable by an FPT-algorithm (with respect to κ)
   [i.e. R is computable in $f(\kappa(x)) \cdot p(|x|)$ steps]
3. there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that
   $\forall x \in \Sigma^* : \kappa'(R(x)) \leq g(\kappa(x))$

Observation: The class FPT is closed under FPT-reductions.

GRAPHKA

Two examples

From the definitions of the problems Independent Set and Clique we get a straightforward FPT-reduction (since a graph has an independent set of size k iff its complement contains a clique of size k), hence

$$k\text{-Independent Set} \equiv^{fpt} k\text{-Clique}$$

On the other hand, the classic reduction of Independent Set to Vertex Cover (where a graph has an independent set of size k iff it has a vertex cover of size $V(G) - k$) is **not** a FPT-reduction, since the size of the parameter is not *fixed*.

GRAPHKA

Two examples

From the definitions of the problems Independent Set and Clique we get a straightforward FPT-reduction (since a graph has an independent set of size k iff its complement contains a clique of size k), hence

$$k\text{-Independent Set} \equiv^{fpt} k\text{-Clique}$$

On the other hand, the classic reduction of Independent Set to Vertex Cover (where a graph has an independent set of size k iff it has a vertex cover of size $V(G) - k$) is **not** a FPT-reduction, since the size of the parameter is not *fixed*.

Hardness - Completeness

If C is a class of parameterized problems,

- (L, κ) is *C-hard under* FPT*-reductions* if all the parameterized problems in C are FPT-reducible to (L, κ).
- (L, κ) is *C-complete under* FPT*-reductions* if $(L, \kappa) \in C$ and is C-hard under FPT-reductions.

THE CLASS para-NP

(L, κ): A parameterized problem with alphabet Σ.

$(L, \kappa) \in$ para-NP if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$, a polynomial function $p : \mathbb{N} \to \mathbb{N}$, and a non-deterministic algorithm that, given a $x \in \Sigma^*$, decides if $x \in L$ in $O(f(\kappa(x)) \cdot p(|x|))$ steps.

Observation: If $L \in$ NP, then every parameterization of L is in para-NP.

Observation: p-VERTEX COLORING $\in$ para-NP.
(And is in fact para-NP-complete.)

The class para-NP

(L, κ): A parameterized problem with alphabet Σ.

$(L, \kappa) \in$ para-NP if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$, a polynomial function $p : \mathbb{N} \to \mathbb{N}$, and a non-deterministic algorithm that, given a $x \in \Sigma^*$, decides if $x \in L$ in $O(f(\kappa(x)) \cdot p(|x|))$ steps.

Observation: If $L \in$ NP, then every parameterization of L is in para-NP.

Observation: p-Vertex Coloring $\in$ para-NP.
(And is in fact para-NP-complete.)

The class para-NP

(L, κ): A parameterized problem with alphabet Σ.

$(L, \kappa) \in$ para-NP if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$, a polynomial function $p : \mathbb{N} \to \mathbb{N}$, and a non-deterministic algorithm that, given a $x \in \Sigma^*$, decides if $x \in L$ in $O(f(\kappa(x)) \cdot p(|x|))$ steps.

Observation: If $L \in$ NP, then every parameterization of L is in para-NP.

Observation: p-Vertex Coloring $\in$ para-NP.
(And is in fact para-NP-complete.)

GRAPHKA

Relation of NP, para-NP

(L, κ): A parameterized problem with alphabet Σ

(L, κ) is *trivial* if $L = \emptyset$ or $L = \Sigma^*$

We define the *i-th slice of* (L, κ) as the problem:
$(L, \kappa)_i = \{x \in L \mid \kappa(x) = i\}$

Theorem: Let $(L, \kappa) \in$ para-NP, be a non-trivial parameterized problem. Then the union of finitely many slices of (L, κ) is NP-complete iff (L, κ) is para-NP-complete (under FPT-reductions).

Corollary: A nontrivial parameterized problem in para-NP with **at least one** NP-complete slice is para-NP-complete.

GRAPHKA

Relation of NP, para-NP

(L, κ): A parameterized problem with alphabet Σ

(L, κ) is *trivial* if $L = \emptyset$ or $L = \Sigma^*$

We define the *i-th slice of* (L, κ) as the problem:
$(L, \kappa)_i = \{x \in L \mid \kappa(x) = i\}$

Theorem: Let $(L, \kappa) \in$ para-NP, be a non-trivial parameterized problem. Then the union of finitely many slices of (L, κ) is NP-complete iff (L, κ) is para-NP-complete (under FPT-reductions).

Corollary: A nontrivial parameterized problem in para-NP with **at least one** NP-complete slice is para-NP-complete.

GRAPHKA

Relation of NP, para-NP

(L, κ): A parameterized problem with alphabet Σ

(L, κ) is *trivial* if $L = \emptyset$ or $L = \Sigma^*$

We define the *i-th slice of* (L, κ) as the problem:
$(L, \kappa)_i = \{x \in L \mid \kappa(x) = i\}$

Theorem: Let $(L, \kappa) \in$ para-NP, be a non-trivial parameterized problem. Then the union of finitely many slices of (L, κ) is NP-complete iff (L, κ) is para-NP-complete (under FPT-reductions).

Corollary: A nontrivial parameterized problem in para-NP with **at least one** NP-complete slice is para-NP-complete.

p-Vertex Coloring is para-NP-complete since, for any fixed $k \geq 3$, *k*-Vertex Coloring is NP-complete.

The following parameterized problem is para-NP-complete:

> *p*-lit-SAT
> *Instance:* A propositional formula ϕ
> *Parameter:* Maximum number of literals in the clauses of ϕ
> *Question:* is ϕ satisfiable?

p-Vertex Coloring is para-NP-complete since, for any fixed $k \geq 3$, *k*-Vertex Coloring is NP-complete.

The following parameterized problem is para-NP-complete:

p-lit-SAT
Instance: A propositional formula ϕ
Parameter: Maximum number of literals in the clauses of ϕ
Question: is ϕ satisfiable?

Conclusions about the class para-NP:

A: If P$\neq$NP, then the problems *p*-Independent Set, *p*-Clique, *p*-Vertex Cover, and other similar such as *p*-Dominating Set and *p*-Hitting Set are **not** para-NP-complete with respect to FPT-reductions.

B: Problems such as *p*-Vertex Coloring and *p*-lit-SAT are not interesting from the parameterized complexity point of view.

C: The class para-NP is for the parameterized complexity the equivalent of NP for classic complexity.
(And in fact FPT=para-NP, iff P=NP)

GRAPHKA

CONCLUSIONS ABOUT THE CLASS para-NP:

A: If P$\neq$NP, then the problems p-INDEPENDENT SET, p-CLIQUE, p-VERTEX COVER, and other similar such as p-DOMINATING SET and p-HITTING SET are **not** para-NP-complete with respect to FPT-reductions.

B: Problems such as p-VERTEX COLORING and p-LIT-SAT are not interesting from the parameterized complexity point of view.

C: The class para-NP is for the parameterized complexity the equivalent of NP for classic complexity.
(And in fact FPT=para-NP, iff P=NP)

Conclusions about the class para-NP:

A: If P$\neq$NP, then the problems *p*-Independent Set, *p*-Clique, *p*-Vertex Cover, and other similar such as *p*-Dominating Set and *p*-Hitting Set are **not** para-NP-complete with respect to FPT-reductions.

B: Problems such as *p*-Vertex Coloring and *p*-lit-SAT are not interesting from the parameterized complexity point of view.

C: The class para-NP is for the parameterized complexity the equivalent of NP for classic complexity.
(And in fact FPT=para-NP, iff P=NP)

(L, κ): A parameterized problem with alphabet Σ.

$(L, \kappa) \in$ XP if there exists a computable function f and an algorithm that, given $x \in \Sigma^*$, decides if $x \in L$ in $O(|x|^{f(\kappa(x))})$ steps.

Observation: The problems p-Independent Set, p-Clique, p-Vertex Cover, and p-Dominating Set all belong in XP.

Observation: The class XP is for the parameterized complexity the equivalent of EXP for the classic complexity.

(L, κ): A parameterized problem with alphabet Σ.

$(L, \kappa) \in$ XP if there exists a computable function f and an algorithm that, given $x \in \Sigma^*$, decides if $x \in L$ in $O(|x|^{f(\kappa(x))})$ steps.

Observation: The problems p-INDEPENDENT SET, p-CLIQUE, p-VERTEX COVER, and p-DOMINATING SET all belong in XP.

Observation: The class XP is for the parameterized complexity the equivalent of EXP for the classic complexity.

GRAPHKA

The class XP

(L, κ): A parameterized problem with alphabet Σ.

$(L, \kappa) \in$ XP if there exists a computable function f and an algorithm that, given $x \in \Sigma^*$, decides if $x \in L$ in $O(|x|^{f(\kappa(x))})$ steps.

Observation: The problems p-Independent Set, p-Clique, p-Vertex Cover, and p-Dominating Set all belong in XP.

Observation: The class XP is for the parameterized complexity the equivalent of EXP for the classic complexity.

GRAPHKA

An XP-complete problem!

p-EXP-DTM-Halt
Instance: A deterministic Turing Machine $\mathbb{M}$,
a $x \in \Sigma$, and a $k \in \mathbb{N}$.
Parameter: k
Question: Does $\mathbb{M}$ with input the string x accept in
at most $|x|^k$ steps?

GRAPHKA

Corollary: FPT $\subset$ XP

Proof: If p-EXP-DTM-HALT$\in$ FPT, then there exists a $c \in \mathbb{N}$ such that every slice of p-EXP-DTM-HALT belongs in DTIME(n^c).

Then the $(c+1)$-th slice of p-EXP-DTM-HALT can be resolved in DTIME(n^c).

This means that DTIME(n^{c+1}) $\subseteq$ DTIME(n^c) and this contradicts the Polynomial Hierarchy Theorem.

GRAPHKA

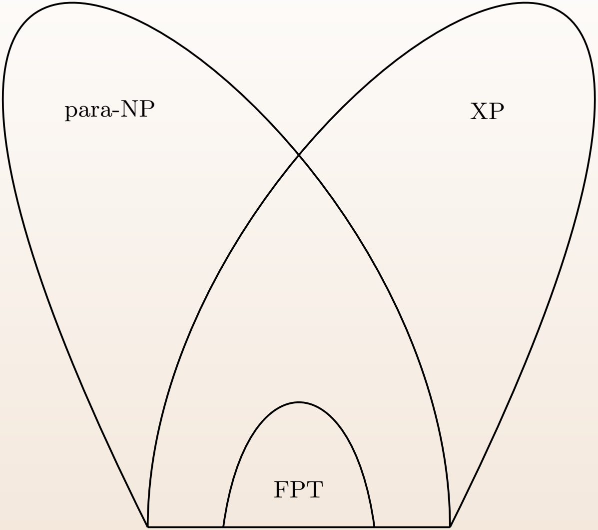
Corollary: FPT $\subset$ XP

Proof: If p-EXP-DTM-HALT$\in$ FPT, then there exists a $c \in \mathbb{N}$ such that every slice of p-EXP-DTM-HALT belongs in DTIME(n^c).

Then the $(c+1)$-th slice of p-EXP-DTM-HALT can be resolved in DTIME(n^c).

This means that DTIME$(n^{c+1}) \subseteq$ DTIME(n^c) and this contradicts the Polynomial Hierarchy Theorem.

Summary: FPT, para-NP and XP



Part II

W[P]

W[SAT]

The W-Hierarchy: W[1],W[2],...

The A-Hierarchy: A[1],A[2],...

Σ is an alphabet and $\kappa : \Sigma^* \to \mathbb{N}$ is a parameterization.

A non-deterministic Turing Machine $\mathbb{M}$ with alphabet Σ, is called *κ-restricted* if there are computable functions $f, h : \mathbb{N} \to \mathbb{N}$ and a polynomial function $p : \mathbb{N} \to \mathbb{N}$, such that the machine $\mathbb{M}$ requires $f(\kappa(x)) \cdot p(|x|)$ steps, **but** at most $h(\kappa(x)) \cdot \log |x|$ of them are non-determininstic.

The class W[P]

W[P] is the class of all parameterized problems (L, κ) that can be decided by a κ-restricted non-deterministic Turing Machine.

Proposition: The class W[P] is closed under FPT-reductions.

Observation: The problems p-Independent Set, p-Clique, p-Vertex Cover, and p-Dominating Set all belong in W[P].

GRAPHKA

The class W[P]

W[P] is the class of all parameterized problems (L, κ) that can be decided by a κ-restricted non-deterministic Turing Machine.

Proposition: The class W[P] is closed under FPT-reductions.

Observation: The problems p-Independent Set, p-Clique, p-Vertex Cover, and p-Dominating Set all belong in W[P].

The class W[P]

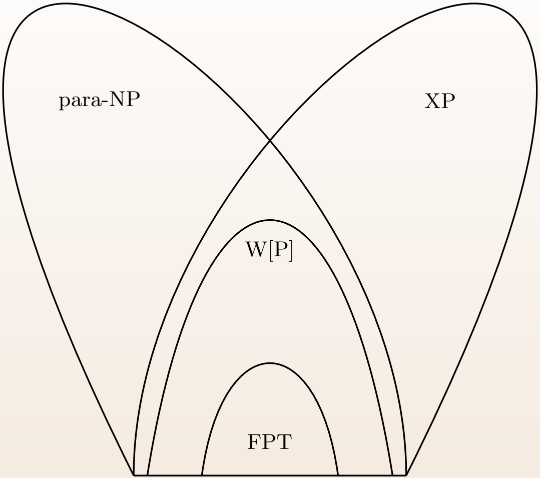
W[P] is the class of all parameterized problems (L, κ) that can be decided by a κ-restricted non-deterministic Turing Machine.

Proposition: The class W[P] is closed under FPT-reductions.

Observation: The problems p-Independent Set, p-Clique, p-Vertex Cover, and p-Dominating Set all belong in W[P].

GRAPHKA

Summary: FPT, para-NP, XP, and W[P]



$\mathsf{FPT} \subseteq \mathsf{W[P]} \subseteq \mathsf{XP} \cap \text{para-NP}$

A W[P]-complete problem

The ***weight*** of a string $x = x_1 x_2 \cdots x_{n-1} x_n \in \{0,1\}^n$ is defined as $\Sigma_{i=1,\dots,n} x_i$ (i.e. the number of ones of the string).

A circuit $\mathcal{C}$ is ***k-satisfiable*** if there exists an input $x \in \{0,1\}^n$ such that $\mathcal{C}(x) = 1$ and the weight of x is k.

p-WSAT(CIRC)
Instance: A circuit $\mathcal{C}$ and an integer $k \geq 0$.
Parameter: k
Question: is $\mathcal{C}$ k-satisfiable?

GRAPHKA

The ***weight*** of a string $x = x_1 x_2 \cdots x_{n-1} x_n \in \{0,1\}^n$ is defined as $\Sigma_{i=1,\dots,n} x_i$ (i.e. the number of ones of the string).

A circuit $\mathcal{C}$ is ***k-satisfiable*** if there exists an input $x \in \{0,1\}^n$ such that $\mathcal{C}(x) = 1$ and the weight of x is k.

> p-WSAT(CIRC)
> *Instance:* A circuit $\mathcal{C}$ and an integer $k \geq 0$.
> *Parameter:* k
> *Question:* is $\mathcal{C}$ k-satisfiable?

GRAPHKA

Another W[P]-complete problem

p-Bounded-NTM-Halt

Instance: A non-deterministic Turing Machine $\mathbb{M}$, a $x \in \Sigma$, and a $k \in \mathbb{N}$.

Parameter: k

Question: Does $\mathbb{M}$ with input the string x accept in at most $|x|$ steps using at most k non-deterministic steps?

GRAPHKA

Alternative definition of W[P]:

Fact 1: $[(L,\kappa)]^{fpt} = \{(L',\kappa') \mid (L',\kappa') \leq^{fpt} (L,\kappa)\}$

Fact 2: p-WSAT(CIRC) is W[P]-complete.

Therefore W[P] can be defined as follows:

$$\text{W[P]} = [p\text{-WSAT(CIRC)}]^{fpt}$$

...where CIRC is the class of all circuits.

Alternative definition of W[P]:

Fact 1: $[(L,\kappa)]^{fpt} = \{(L',\kappa') \mid (L',\kappa') \leq^{fpt} (L,\kappa)\}$

Fact 2: p-WSAT(CIRC) is W[P]-complete.

Therefore W[P] can be defined as follows:

$$\text{W[P]} = [p\text{-WSAT(CIRC)}]^{fpt}$$

...where CIRC is the class of all circuits.

GRAPHKA

THE CLASS W[SAT]

PROP is the class of all propositional formulas.

> *p*-WSAT(PROP)
> *Instance:* A propositional formula ϕ and a $k \in \mathbb{N}$.
> *Parameter:* k
> *Question:* Is ϕ k-satisfiable?

W[SAT]:= $[p\text{-WSAT(PROP)}]^{fpt}$

Observation: W[SAT]⊆W[P]

GRAPHKA

THE CLASS W[SAT]

PROP is the class of all propositional formulas.

p-WSAT(PROP)
Instance: A propositional formula ϕ and a $k \in \mathbb{N}$.
Parameter: k
Question: Is ϕ k-satisfiable?

W[SAT]:= $[p\text{-WSAT(PROP)}]^{fpt}$

Observation: W[SAT]$\subseteq$W[P]

The class W[SAT]

PROP is the class of all propositional formulas.

p-Wsat(PROP)
Instance: A propositional formula ϕ and a $k \in \mathbb{N}$.
Parameter: k
Question: Is ϕ k-satisfiable?

W[SAT]:= $[p\text{-}\mathrm{Wsat}(\mathrm{PROP})]^{fpt}$

Observation: W[SAT]$\subseteq$W[P]

GRAPHKA

The class W[SAT]

PROP is the class of all propositional formulas.

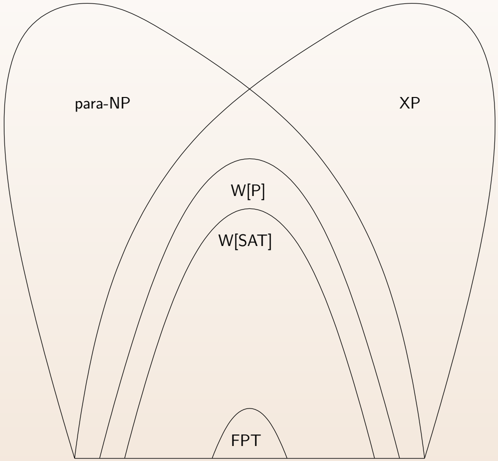
> p-WSAT(PROP)
> *Instance:* A propositional formula ϕ and a $k \in \mathbb{N}$.
> *Parameter:* k
> *Question:* Is ϕ k-satisfiable?

W[SAT]:= $[p\text{-WSAT(PROP)}]^{fpt}$

Observation: W[SAT]$\subseteq$W[P]

GRAPHKA

Summary: FPT, para-NP, XP, W[P], and W[SAT]



GRAPHKA

Subclasses of PROP

$$\Gamma_{0,d} = \{\lambda_1 \wedge \ldots \wedge \lambda_c \mid 1 \leq c \leq d, \text{ and } \lambda_1, \ldots, \lambda_c \text{ are literals}\}$$

$$\Delta_{0,d} = \{\lambda_1 \vee \ldots \vee \lambda_c \mid 1 \leq c \leq d, \text{ and } \lambda_1, \ldots, \lambda_c \text{ are literals}\}$$

$$\Gamma_{t+1,d} = \{\bigwedge_{i \in I} \delta_i \mid I \text{ is a set of indices and } \forall_{i \in I}\, \delta_i \in \Delta_{t,d}\}$$

$$\Delta_{t+1,d} = \{\bigvee_{i \in I} \gamma_i \mid I \text{ is a set of indices and } \forall_{i \in I}\, \gamma_i \in \Gamma_{t,d}\}$$

Observation: $\Gamma_{2,1}$ is the class of formulas in normal conjunctive form (CNF-formulas).

Observation: $\Gamma_{1,3}$ is the class of the 3-CNF-formulas.

GRAPHKA

$$\Gamma_{0,d} = \{\lambda_1 \wedge \ldots \wedge \lambda_c \mid 1 \leq c \leq d, \text{ and } \lambda_1, \ldots, \lambda_c \text{ are literals}\}$$

$$\Delta_{0,d} = \{\lambda_1 \vee \ldots \vee \lambda_c \mid 1 \leq c \leq d, \text{ and } \lambda_1, \ldots, \lambda_c \text{ are literals}\}$$

$$\Gamma_{t+1,d} = \{\bigwedge_{i \in I} \delta_i \mid I \text{ is a set of indices and } \forall_{i \in I}\ \delta_i \in \Delta_{t,d}\}$$

$$\Delta_{t+1,d} = \{\bigvee_{i \in I} \gamma_i \mid I \text{ is a set of indices and } \forall_{i \in I}\ \gamma_i \in \Gamma_{t,d}\}$$

Observation: $\Gamma_{2,1}$ is the class of formulas in normal conjunctive form (CNF-formulas).

Observation: $\Gamma_{1,3}$ is the class of the 3-CNF-formulas.

Subclasses of PROP

$$\Gamma_{0,d} = \{\lambda_1 \wedge \ldots \wedge \lambda_c \mid 1 \leq c \leq d, \text{ and } \lambda_1, \ldots, \lambda_c \text{ are literals}\}$$

$$\Delta_{0,d} = \{\lambda_1 \vee \ldots \vee \lambda_c \mid 1 \leq c \leq d, \text{ and } \lambda_1, \ldots, \lambda_c \text{ are literals}\}$$

$$\Gamma_{t+1,d} = \{\bigwedge_{i \in I} \delta_i \mid I \text{ is a set of indices and } \forall_{i \in I}\ \delta_i \in \Delta_{t,d}\}$$

$$\Delta_{t+1,d} = \{\bigvee_{i \in I} \gamma_i \mid I \text{ is a set of indices and } \forall_{i \in I}\ \gamma_i \in \Gamma_{t,d}\}$$

Observation: $\Gamma_{2,1}$ is the class of formulas in normal conjunctive form (CNF-formulas).

Observation: $\Gamma_{1,3}$ is the class of the 3-CNF-formulas.

The classes W[t], for $t \geq 1$

For every $t \geq 1$, we define: W[t]:= $[\{\, p\text{-WSAT}(\,\Gamma_{t,d}) \mid d \geq 1\}]^{fpt}$

Observation: FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \cdots \subseteq$ W[SAT] $\subseteq$ W[P]

GRAPHKA

THE CLASSES W[t], FOR $t \geq 1$

For every $t \geq 1$, we define: W[t]:= $[\{\, p\text{-WSAT}(\,\Gamma_{t,d}) \mid d \geq 1\}]^{fpt}$

Observation: FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \cdots \subseteq$ W[SAT] $\subseteq$ W[P]

GRAPHKA

The classes W[t], for $t \geq 1$

For every $t \geq 1$, we define: W[t]:= $[\{\, p\text{-WSAT}(\,\Gamma_{t,d}) \mid d \geq 1\}]^{fpt}$

Observation: FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \cdots \subseteq$ W[SAT] $\subseteq$ W[P]

GRAPHKA

Depth and Weft of circuits

In a circuit $\mathcal{C}$ we call ***small*** gates those that have at most 2 inputs and ***large*** gates those that have more than 2 inputs.

The *depth* of $\mathcal{C}$ is the maximum number of gates between an input and the output of $\mathcal{C}$.

The *weft* of $\mathcal{C}$ is the maximum number of **large** gates between an input and the output of $\mathcal{C}$.

Observation: $\text{depth}(\mathcal{C}) \geq \text{weft}(\mathcal{C})$

GRAPHKA

Depth and Weft of circuits

In a circuit $\mathcal{C}$ we call ***small*** gates those that have at most 2 inputs and ***large*** gates those that have more than 2 inputs.

The *depth* of $\mathcal{C}$ is the maximum number of gates between an input and the output of $\mathcal{C}$.

The *weft* of $\mathcal{C}$ is the maximum number of **large** gates between an input and the output of $\mathcal{C}$.

Observation: $\text{depth}(\mathcal{C}) \geq \text{weft}(\mathcal{C})$

If $d \geq t \geq 0$, we define:

$$C_{t,d} = \{\, \mathcal{C} \mid \mathcal{C} \text{ is a circuit such that: weft}(\mathcal{C}) \leq t \text{ \& depth}(\mathcal{C}) \leq d \,\}$$

Example: 3CNF-SAT$\in C_{1,2}$

Alternative Definition: *(Downey and Fellows, 1991)*

For every $t \geq 1$ we define W[t]$:= [\{p\text{-WSAT}(C_{t,d}) \mid d \geq 1\}]^{fpt}$

GRAPHKA

ALTERNATIVE DEFINITION OF THE CLASSES W[t]

If $d \geq t \geq 0$, we define:

$$C_{t,d} = \{ \mathcal{C} \mid \mathcal{C} \text{ is a circuit such that: weft}(\mathcal{C}) \leq t \text{ \& depth}(\mathcal{C}) \leq d \}$$

Example: 3CNF-SAT$\in C_{1,2}$

Alternative Definition: *(Downey and Fellows, 1991)*

For every $t \geq 1$ we define W[t]:$= [\{p\text{-}\mathrm{WSAT}(C_{t,d}) \mid d \geq 1\}]^{fpt}$

GRAPHKA

p-WSAT$(\Gamma^+_{t,1})$: we restrict the instances of p-WSAT$(\Gamma_{t,1})$ to those that have only positive literals.

p-WSAT$(\Gamma^-_{t,1})$: we restrict the instances of p-WSAT$(\Gamma_{t,1})$ to those that have only negative literals.

Theorem: If t is even, then p-WSAT$(\Gamma^+_{t,1})$ is W[t]-complete.

Theorem: If t is odd, then p-WSAT$(\Gamma^-_{t,1})$ is W[t]-complete.

GRAPHKA

A general structural theorem

p-WSAT$(\Gamma^{+}_{t,1})$: we restrict the instances of p-WSAT$(\Gamma_{t,1})$ to those that have only positive literals.

p-WSAT$(\Gamma^{-}_{t,1})$: we restrict the instances of p-WSAT$(\Gamma_{t,1})$ to those that have only negative literals.

Theorem: If t is even, then p-WSAT$(\Gamma^{+}_{t,1})$ is W[t]-complete.

Theorem: If t is odd, then p-WSAT$(\Gamma^{-}_{t,1})$ is W[t]-complete.

GRAPHKA

A general structural theorem

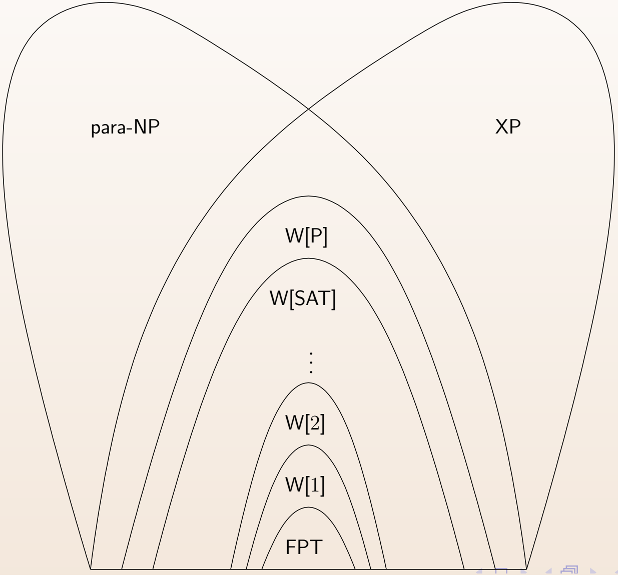
p-WSAT($\Gamma^+_{t,1}$): we restrict the instances of p-WSAT($\Gamma_{t,1}$) to those that have only positive literals.

p-WSAT($\Gamma^-_{t,1}$): we restrict the instances of p-WSAT($\Gamma_{t,1}$) to those that have only negative literals.

Theorem: If t is even, then p-WSAT($\Gamma^+_{t,1}$) is W[t]-complete.

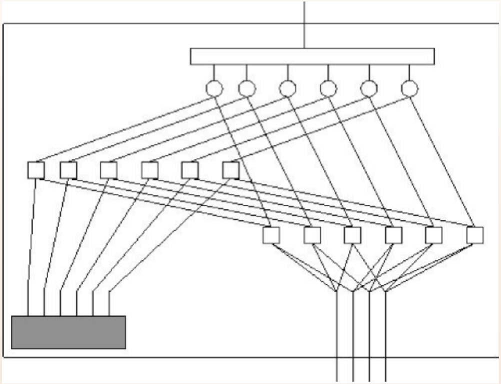
Theorem: If t is odd, then p-WSAT($\Gamma^-_{t,1}$) is W[t]-complete.

A panorama of the classes so far



p-CLIQUE $\in$ W[1]

Proof. The following circuit proves that p-CLIQUE$\leq^{fpt}p$-WSAT$(C_{1,t})$.
(the weft of the circuit is 1)



□: gate AND, ○: gate EQUIV

GRAPHKA

W[1] is arguably the most important class of intractable parameterized problems, because the parameterized version of a great number of prominent NP-complete problems is naturally complete for this class.

Theorem: *p*-Clique is W[1]-complete.

Corollary: *p*-Independent Set is W[1]-complete.

Corollary: *p*-Vertex Cover is W[1]-complete.

GRAPHKA

W[1]-complete problems

W[1] is arguably the most important class of intractable parameterized problems, because the parameterized version of a great number of prominent NP-complete problems is naturally complete for this class.

Theorem: *p*-Clique is W[1]-complete.

Corollary: *p*-Independent Set is W[1]-complete.

Corollary: *p*-Vertex Cover is W[1]-complete.

W[1]-COMPLETE PROBLEMS

W[1] is arguably the most important class of intractable parameterized problems, because the parameterized version of a great number of prominent NP-complete problems is naturally complete for this class.

Theorem: p-CLIQUE is W[1]-complete.

Corollary: p-INDEPENDENT SET is W[1]-complete.

Corollary: p-VERTEX COVER is W[1]-complete.

GRAPHKA

W[1]-COMPLETE PROBLEMS

W[1] is arguably the most important class of intractable parameterized problems, because the parameterized version of a great number of prominent NP-complete problems is naturally complete for this class.

Theorem: p-CLIQUE is W[1]-complete.

Corollary: p-INDEPENDENT SET is W[1]-complete.

Corollary: p-VERTEX COVER is W[1]-complete.

Theorem: The following problems are W[1]-complete.

p-SET PACKING
Instance: A finite family of finite sets $S_1, \dots, S_r$ and $k \in \mathbb{N}$.
Parameter: k
Question: $\exists I \in \{1, \dots, r\}^{[=k]} : \forall_{i \neq j,\, i,j \in I} S_i \cap S_j = \emptyset$?

p-SHORT-NSTM-HALT
Instance: A non-deterministic Turing Machine $\mathbb{M}$, with **exactly one** tape and a $k \in \mathbb{N}$.
Parameter: k
Question: Does $\mathbb{M}$ halt with input the empty string in **at most** k steps?

GRAPHKA

More W[1]-complete problems

Theorem: The following problems are W[1]-complete.

p-Set Packing
Instance: A finite family of finite sets $S_1, \ldots, S_r$ and $k \in \mathbb{N}$.
Parameter: k
Question: $\exists I \in \{1, \ldots, r\}^{[=k]} : \forall_{i \neq j,\, i,j \in I} S_i \cap S_j = \emptyset$?

p-short-NSTM-Halt
Instance: A non-deterministic Turing Machine $\mathbb{M}$, with **exactly one** tape and a $k \in \mathbb{N}$.
Parameter: k
Question: Does $\mathbb{M}$ halt with input the empty string in **at most** k steps?

Theorem: The following problem is W[2]-complete.

p-short-NMTM-Halt

Instance: A non-deterministic Turing Machine $\mathbb{M}$,
with **one or more** tapes and a $k \in \mathbb{N}$.

Parameter: k

Question: Does $\mathbb{M}$ halt with input the empty string in **at most** k steps?

More W[2]-complete problems

Theorem: *p*-Dominating Set is W[2]-complete.

Corollary: *p*-Hitting Set is W[2]-complete.

Theorem: The following problem is W[2]-complete.

> *p*-Steiner Tree
> *Instance:* A graph G, $S \subseteq V(G)^{[\leq k]}$, $k \in \mathbb{N}$.
> *Parameter:* m
> *Question:* $\exists R \in (V(G) \setminus S)^{[\leq m]} : G[S \cup R]$ is connected?

Note: The same problem, parameterized by k instead of m, is in FPT.

GRAPHKA

More W[2]-complete problems

Theorem: p-Dominating Set is W[2]-complete.

Corollary: p-Hitting Set is W[2]-complete.

Theorem: The following problem is W[2]-complete.

> p-Steiner Tree
> *Instance:* A graph G, $S \subseteq V(G)^{[\leq k]}$, $k \in \mathbb{N}$.
> *Parameter:* m
> *Question:* $\exists R \in (V(G) \setminus S)^{[\leq m]} : G[S \cup R]$ is connected?

Note: The same problem, parameterized by k instead of m, is in FPT.

Theorem: p-Dominating Set is W[2]-complete.

Corollary: p-Hitting Set is W[2]-complete.

Theorem: The following problem is W[2]-complete.

p-Steiner Tree *Instance:* A graph G, $S \subseteq V(G)^{[\leq k]}$, $k \in \mathbb{N}$. *Parameter:* m *Question:* $\exists R \in (V(G) \setminus S)^{[\leq m]} : G[S \cup R]$ is connected?

Note: The same problem, parameterized by k instead of m, is in FPT.

More W[2]-complete problems

Theorem: p-Dominating Set is W[2]-complete.

Corollary: p-Hitting Set is W[2]-complete.

Theorem: The following problem is W[2]-complete.

> p-Steiner Tree
> *Instance:* A graph G, $S \subseteq V(G)^{[\leq k]}$, $k \in \mathbb{N}$.
> *Parameter:* m
> *Question:* $\exists R \in (V(G) \setminus S)^{[\leq m]} : G[S \cup R]$ is connected?

Note: The same problem, parameterized by k instead of m, is in FPT.

We define the following natural model-checking problem for a class Φ of formulas.

p-$\mathrm{MC}(\Phi)$
Instance: A structure A and a formula $\phi \in \Phi$.
Parameter: $\lvert\phi\rvert$
Question: Is $\phi(\mathcal{A}) \neq \emptyset$?

For every $t \geq 1$, we define: A[t]:= $[p\text{-}\mathrm{MC}(\Sigma_t)]^{fpt}$.

Observation: FPT $\subseteq$ A[1] $\subseteq$ A[2] $\subseteq \cdots \subseteq$ A[P] $\subseteq$ XP

We define the following natural model-checking problem for a class Φ of formulas.

> *p*-$\mathrm{MC}(\Phi)$
> *Instance:* A structure A and a formula $\phi \in \Phi$.
> *Parameter:* $|\phi|$
> *Question:* Is $\phi(\mathcal{A}) \neq \emptyset$?

For every $t \geq 1$, we define: A[t]:$= [\, p\text{-}\mathrm{MC}(\,\Sigma_t)]^{fpt}$.

Observation: FPT $\subseteq$ A[1] $\subseteq$ A[2] $\subseteq \cdots \subseteq$ A[P] $\subseteq$ XP

We define the following natural model-checking problem for a class Φ of formulas.

p-$\mathrm{MC}(\Phi)$
Instance: A structure A and a formula $\phi \in \Phi$.
Parameter: $|\phi|$
Question: Is $\phi(\mathcal{A}) \neq \emptyset$?

For every $t \geq 1$, we define: A[t]:= $[\, p\text{-}\mathrm{MC}(\, \Sigma_t)]^{fpt}$.

Observation: FPT $\subseteq$ A[1] $\subseteq$ A[2] $\subseteq \cdots \subseteq$ A[P] $\subseteq$ XP

p-Clique, *p*-Independent Set, *p*-Vertex Cover, *p*-Set Packing, and *p*-Short-NSTM-Halt are all A[1]-complete.
In fact...

Theorem: W[1] = A[1]

Also, *p*-Dominating Set and *p*-Hitting Set are in A[2].
In fact...

Theorem: $\forall t \geq 1$, W[t] $\subseteq$ A[t]

GRAPHKA

p-Clique, *p*-Independent Set, *p*-Vertex Cover, *p*-Set Packing, and *p*-Short-NSTM-Halt are all A[1]-complete.
In fact...

Theorem: W[1] = A[1]

Also, *p*-Dominating Set and *p*-Hitting Set are in A[2].
In fact...

Theorem: $\forall t \geq 1$, W[t] $\subseteq$ A[t]

GRAPHKA

p-Clique, *p*-Independent Set, *p*-Vertex Cover, *p*-Set Packing, and *p*-Short-NSTM-Halt are all A[1]-complete.
In fact...

Theorem: W[1] = A[1]

Also, *p*-Dominating Set and *p*-Hitting Set are in A[2].
In fact...

Theorem: $\forall t \geq 1$, W[t] $\subseteq$ A[t]

GRAPHKA

Relation of the two hierarchies

p-Clique, *p*-Independent Set, *p*-Vertex Cover, *p*-Set Packing, and *p*-Short-NSTM-Halt are all A[1]-complete.
In fact...

Theorem: W[1] = A[1]

Also, *p*-Dominating Set and *p*-Hitting Set are in A[2].
In fact...

Theorem: $\forall t \geq 1,$ W[t] $\subseteq$ A[t]

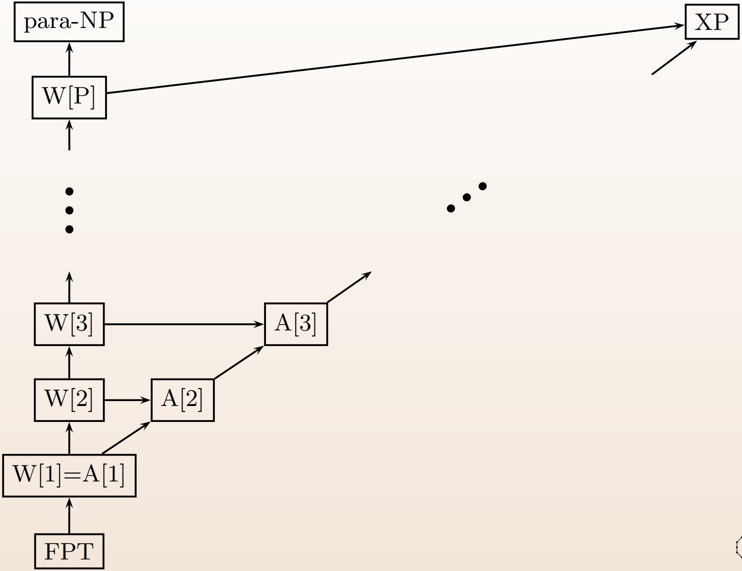
GRAPHKA

An A[2]-complete problem

Theorem: The following problem is A[2]-complete.

p-Clique-Dominating-Set
Instance: A graph G and $k, m \in \mathbb{N}$.
Parameter: $k + m$
Question: Does G contain a set of k vertices that dominates every clique of size m?

The two hierarchies



A last remark

- The W-Hierarchy was defined as a "refinement" of NP for parameterized problems.

- The A-Hierarchy was defined as the analog of the Polynomial Hierarchy.

For this and other reasons it is unlikely that the two hierarchies coincide.

GRAPHKA

A last remark

- The W-Hierarchy was defined as a "refinement" of NP for parameterized problems.
- The A-Hierarchy was defined as the analog of the Polynomial Hierarchy.

For this and other reasons it is unlikely that the two hierarchies coincide.

A last remark

- The W-Hierarchy was defined as a "refinement" of NP for parameterized problems.
- The A-Hierarchy was defined as the analog of the Polynomial Hierarchy.

For this and other reasons it is unlikely that the two hierarchies coincide.

Bibliography & Suggestions for further study

1. *Confronting Intractability via Parameters* - R. Downey, D. Thilikos
2. *Parameterized Complexity Theory* - J. Flum, M. Grohe
3. *Fundamentals of Parameterized Complexity* - R. Downey, M. Fellows
4. An Introduction to Parameterized Algorithms and Complexity (Course) - D. Thilikos [MPLA]

Thank you!

¡Muchas gracias!

GRAPHKA