Parameterized Complexity

Stefan Szeider
Vienna University of Technology, Austria

SAT-SMT Summer School 2013
Espoo, Finland

Outline

Parameterized Problems

- A *parameterized decision problem* P is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ.
- $(x, k) \in \Sigma^* \times \mathbb{N}$ is a *problem instance*
- x the main part and k the parameter.
- P *is fixed-parameter tractable if there exist an algorithm* A*, a function* f*, and a constant* c*, such that,* A *decides whether* $(x, k) \in P$ *in time* $O(f(k)|x|^c)$.
- This is the most important definition of this tutorial.

Basic Classes of Parameterized Problems

- FPT: the class of all parameterized decision problems that are fixed-parameter tractable.
- XP: all parameterized decision problems that can be solved in polynomial time if the parameter is considered constant. That is, $(x, k) \in P$ can be decided in time $O(|x|^{f(k)})$.
- paraNP: all parameterized decision problems for $(x, k) \in P$ can be decided non-deterministically in time $O(f(k)|x|^c)$. Equivalent definition: The unparameterized problem is in NP, and for a finited number of fixed-values of the parameter, the problem is NP-complete.

Discussion

- If f is a polynomial, then A runs in polynomial time.
- Hence, if the unparameterized version of P is NP-hard, then we will assume that f is super-polynomial.
- FPT $\subseteq$ XP (can be shown to be a proper subset)
- FPT $\subseteq$ paraNP
- XP and paraNP are incomparable (subject to complexity theoretic assumptions)
- Between FPT and XP $\cap$ paraNP are the W[i]-*classes* which we will discuss separately.

Some Examples

- SAT(vars)
  - Instance: a CNF formula F.
  - Parameter: number of variables of F.
  - Question: is F satisfiable?
- SAT(clause size)
  - Instance: a CNF formula F.
  - Parameter: the size of a largest clause of F.
  - Question: is F satisfiable?
- SAT(ones) (=WeightedSAT)
  - Instance: a CNF formula F and an integer k.
  - Parameter: k.
  - Question: is there a satisfying assignment for F that sets exactly k variables to 1?

Can we locate the problems in FPT, XP, or paraNP?

- SAT(vars) is in FPT
- SAT(clause size) is paraNP-complete (unlikely in FPT)
- Bounded-SAT(ones) is in XP (unlikely in FPT)

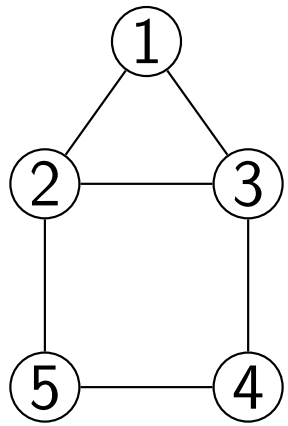
With tools from parameterized intractability one can provide strong theoretical evidence that a problem is not FPT.

Parameterized Optimization Problems

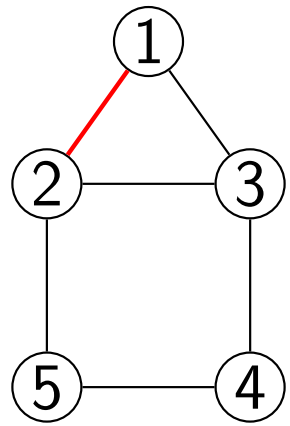
- We specify instance, parameter and question, such as:
- VERTEX COVER (VC)
  - Instance: a graph G and an integer k.
  - Parameter: k.
  - Question: does G admit a VC of size at most k?

For optimization problems, the *default parameter* or *standard parameter* is the solution size.

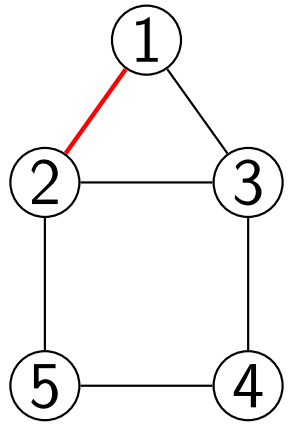
Example: searching for a VC



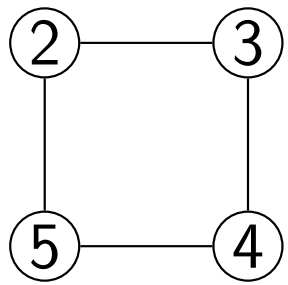
Example: searching for a VC



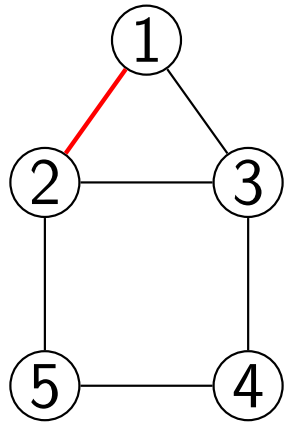
Example: searching for a VC



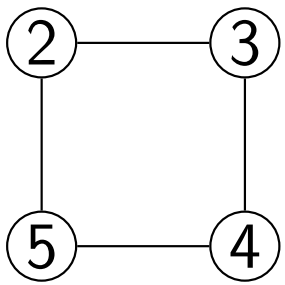
Put (1) into the VC.



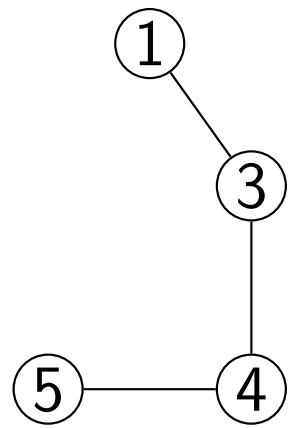
Example: searching for a VC



Put (1) into the VC.



Put (2) into the VC.



Bounded search tree approach for VC

We construct complete binary search tree T of depth k:

1. Label the root of the tree with the pair $(G, \emptyset)$.
2. Label the remaining nodes of the tree recursively as follows: Let (H, S) be the label of a node x of T whose two children are not labelled yet. Choose an edge uv from the edges of H.
   - 2.1 Label the left child of x with $(H - u, S \cup \{u\})$.
   - 2.2 Label the high child of x with $(H - v, S \cup \{v\})$.
3. If there exists a node labelled with (H, S) such that H has no edges then S is a vertex cover of G. Since the depth of T is bounded by k, the size of S is at most k.

- We conclude: *The above algorithm solves VC in time $O(2^k \cdot n^2)$ where n denotes the number of vertices of G.*
- *Key Observation:* If k is "small" in comparison to n, then the algorithm is efficient.
- If $k = O(\log n)$ then the algorithm runs even in polynomial time.

But we can use other values as parameters, e.g.,

- VC(max degree)
  - Instance: a graph G and an integer k.
  - Parameter: maximum degree of vertices in G.
  - Question: does G admit a VC of size at most k?
- VC(treewidth)
- VC(number of vertices),
- etc.
- We suffix the problem name with a description of the parameter at the end.
- VC $=$ VC(solution size).

Techniques for showing FPT

- basic techniques: *bounded search trees*, *kernelization*
- advanced techniques: *tree decompositions*, *color codings*, *iterative compression*, etc.
- We will see examples for some of these techniques later.
- Often well-known techniques from practitioners turn out to yield FPT!
  ("that's exactly what we have been doing!")

Big O-star Notation

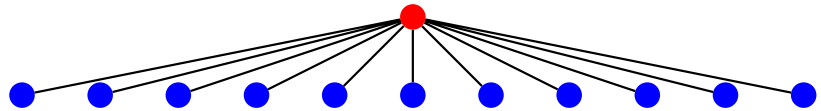
- It is convenient to denote the running time of an fpt algorithm as $O^*(f(k))$, omitting the polynomial factor $poly(n)$.

Classical Reductions

- Classical reductions for showing NP-hardness:
- Two problems $P, Q \subseteq \Sigma^*$.
- A polynomial reduction from P to Q is a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ such that for each $x \in \Sigma^*$ we have: $x \in P$ iff $f(x) \in Q$.

Example: VC to IS

- Let G be a graph with n vertices.



- G has a vertex cover of size k if and only if G has an independent set of size $k' = n - k$.
- k' is not a function of k, as it depends on n.
- Hence this does *not* give a parameterized reduction from VC to IS.
- If there were a parameterized reduction from IS to VC then it would follow that IS is fixed-parameter tractable, but we believe it is not.

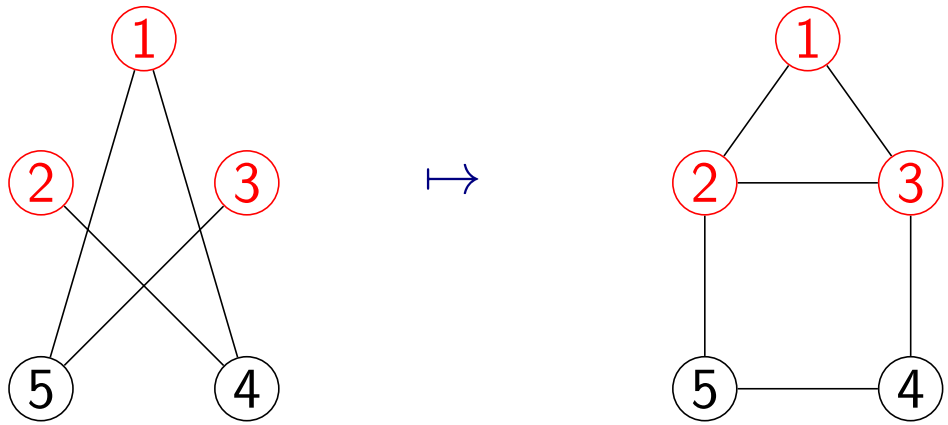
Parameterized Reductions

- Let $P, Q \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems.
- A *parameterized reduction* from P to Q is a function $f : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ such that
  1. $(x, k) \in P$ iff $f(x, k) = (x', k') \in Q$.
  2. $k' \leq h(k)$ for some function h.
  3. $f(x, k)$ can be computed in time $O(g(k) \cdot |x|^c)$ for a computable function g and a constant c.
- *Fact:* If $Q \in$ FPT and there is a parameterized reduction from P to Q, then $P \in$ FPT.

- Consider the problem CLIQUE.
  - Instance: a graph $G = (V, E)$, a nonnegative integer k.
  - Parameter: k.
  - Question: does G contain a clique on k vertices?
- A clique is a complete graph (where any two of its vertices are adjacent).

Example: from IS to CLIQUE

- Let G be a graph with n vertices. Consider its complement graph $\overline{G}$.



- G has an independent set of size k if and only if the complement graph $\bar{G}$ contains a clique on k vertices.
  ($\bar{G}$ has the same vertices as G, and two vertices in $\bar{G}$ are adjacent if and only if they are not adjacent in G).
- Hence there is a parameterized reduction from IS to CLIQUE, and a parameterized reduction from CLIQUE to IS.

- Hence either both problems IS and CLIQUE are fixed-parameter tractable or both are not.
- There are hundreds of parameterized problem known that are equivalent to IS and CLIQUE under parameterized reductions.

Intractability

- It is believed that IS or CLIQUE are not fixed-parameter tractable.
- W[1] denotes the class of all parameterized decision problems reducible to CLIQUE with parameterized reductions.
- We use similar terminology (W[1]-hard, W[1]-complete) as in classical complexity, always assuming parameterized reductions.
- Thus IS and CLIQUE are W[1]-complete.

Higher levels of param. Intractability

- Consider the parameterized problem Hitting Set (HS).
- One can show that HS is W[1]-hard, but it is not known whether HS belongs to W[1].
- In fact, one believes that there is no parameterized reduction from HS to CLIQUE.
- W[2] denotes the class where HS is a complete problem.
- There is a generic way to define the classes W[t] for $t = 1, 2, 3, \ldots$ using weighted satisfiability

Circuits and Weft

- Consider a decision circuit consisting of *large gates* with unbounded fan-in, and *small gates* with bounded fan-in (the precise value of the bound is unimportant).
- The *weft* of the circuit is the largest number of large gates on any path from the inputs to the output of the circuit.
- The *depth* of the circuit is the largest number of gates of any size on any path from the inputs to the output of the circuit.

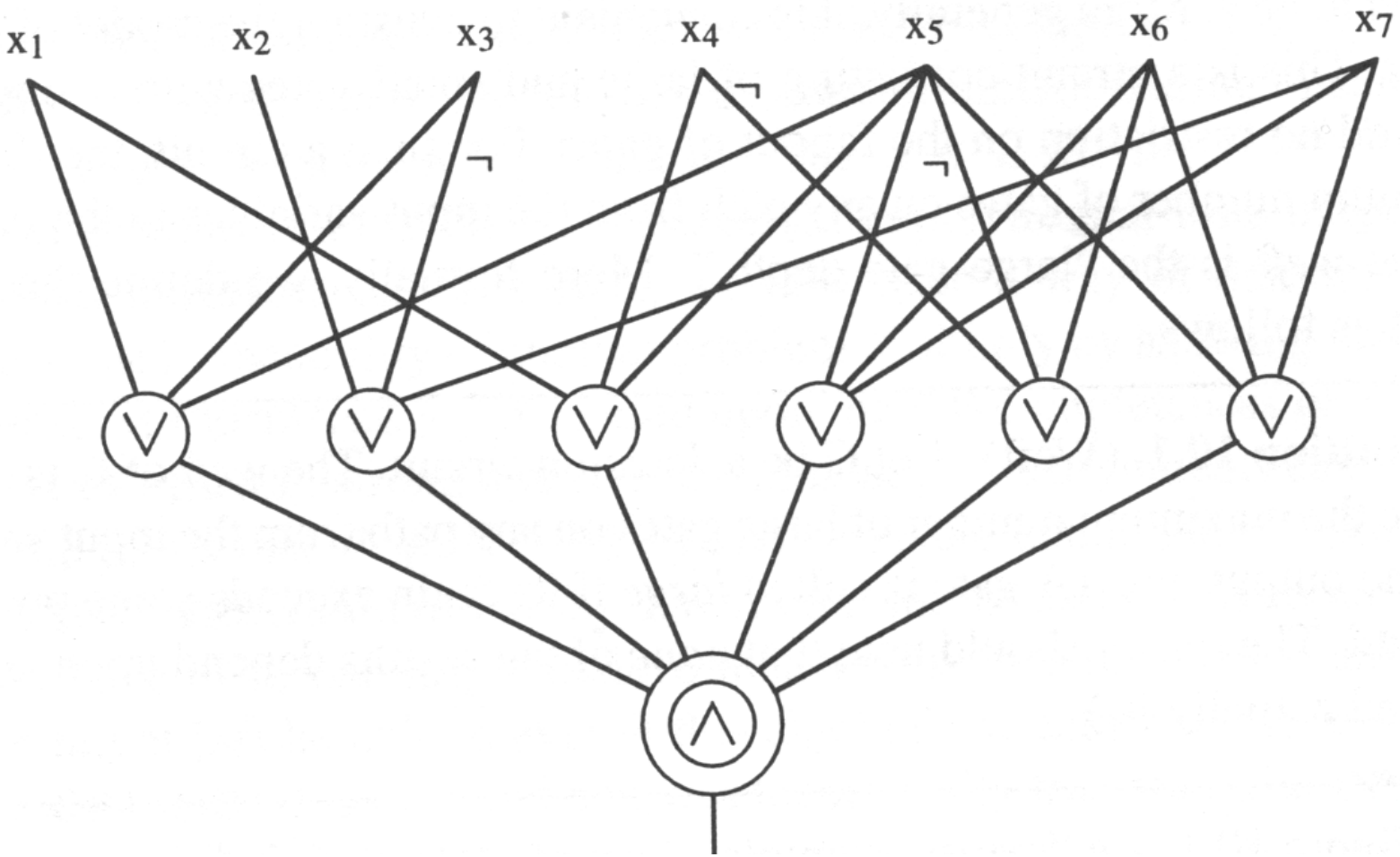
Circuits and Weft

- Weighted Weft t Depth h Circuit Satisfiability (WCS(t, h)):
  - Instance: a weft t depth h circuit C and an integer k.
  - Parameter: k.
  - Question: has C a satisfying input of weight exactly k?

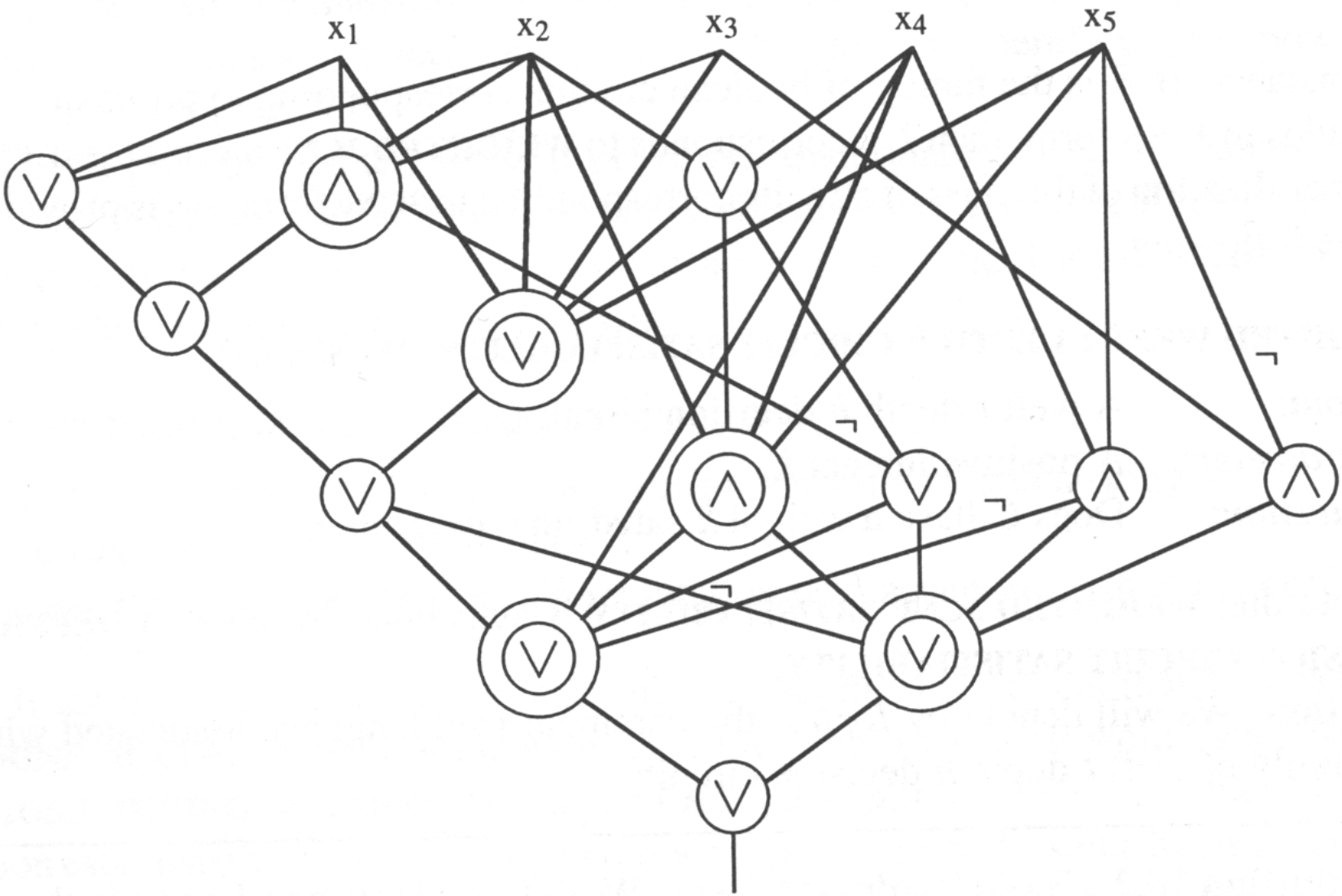
Examples

- A 3CNF formula can be considered as a weft 1 depth 2 circuit.
- A CNF formula can be considered as a weft 2 depth 2 circuit.

3CNF as a Circuit of Weft 1



Weft 2 Depth 5 Circuit



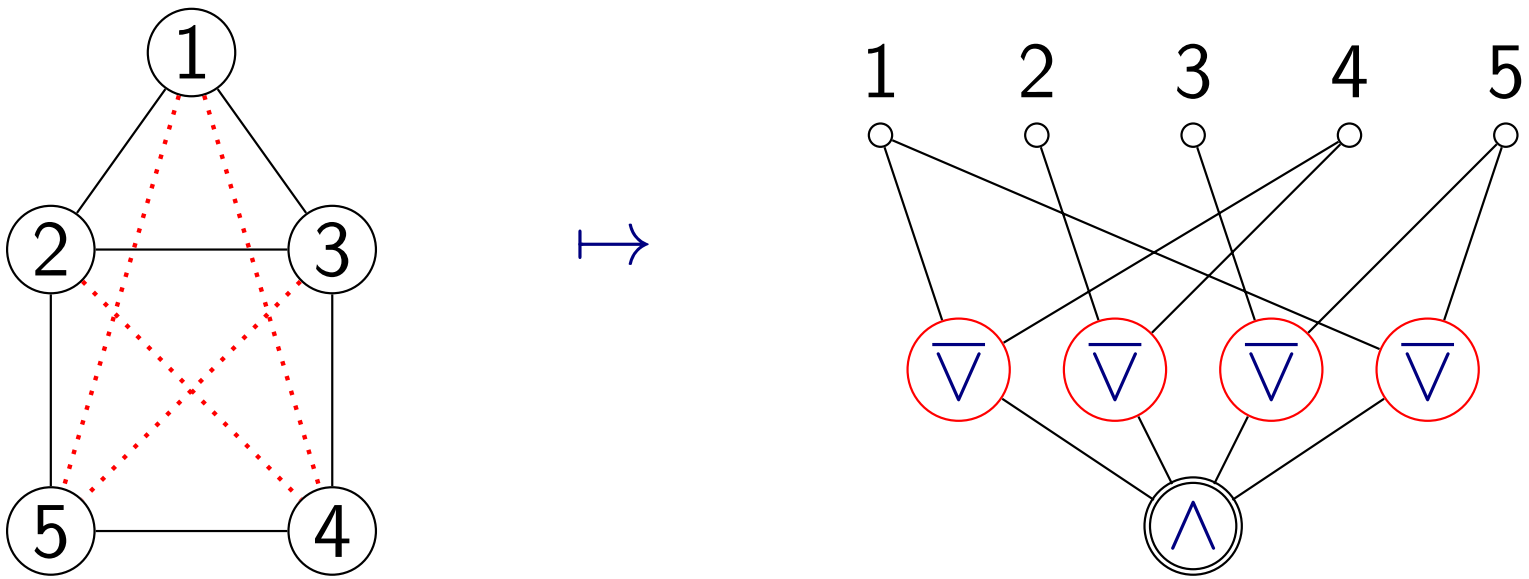
The Weft Hierarchy

- For each $t \geq 1$, the class W[t] is defined as the class of all parameterized problems that can be reduced to WCS(t, h) for some constant h.
- W[P] is the class of parameterized problems that can be reduced to WCS(∞, ∞).

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \cdots \subseteq \text{W[SAT]} \subseteq \text{W[P]}$$

All inclusions are assumed to be proper.

Example: CLIQUE $\in$ W[1]



DOMINATING SET $\in$ W[2]

- Similar: input gate for each vertex
- OR gate for each vertex, building a disjunction on its closed neighborhood. (unbounded fan-in!)
- One AND Gate at the bottom.

Logical Characterization of the W-Hierarchy

First-Order Model Checking $\mathsf{MC}(\Phi)$

- *Instance:* A structure $\mathcal{A}$ and a formula $\varphi \in \Phi$.
- *Parameter:* $|\varphi|$.
- *Question:* Decide whether $\mathcal{A} \models \varphi$.

- $\Sigma_{t,u}$ = class of all FO formulas with
  - t quantifier block alternations, starting with $\exists$,
  - all blocks after the leading one consist of at most u quantifiers.
- Example: $\exists x_1, ...x_n \forall y_1, y_2 \exists z_1, z_2, z_3, z_4 \varphi$ belongs to $\Sigma_{3,4}$.
- *Theorem* MC$(\Sigma_{t,u})$ is W[t]-complete, for every $u, t \geq 1$.

Examples

- CLIQUE:

$$\exists x_1, \ldots, x_k \bigwedge_{1 \leq i < j \leq k} E(x_i, x_j)$$

  is in $\Sigma_{1,0}$, hence CLIQUE is in W[1].

- Dominating Set:

$$\exists x_1, \ldots, x_k \forall y (\bigvee_{i=1}^{k} (y = x_i \vee E(y, x_i))$$

  is in $\Sigma_{2,1}$, hence DS is in W[2].

Further reading...

- Downey & Fellows "Parameterized Complexity" Springer 1999.
- Niedermeier "Invitation to fixed-parameter algorithms" CUP 2006.
- Flum & Grohe "Parameterized Complexity Theory" Springer 2006.
- Cesati "The Turing way to parameterized complexity" JCSS 67, 2003.
- The Computer Journal, Special Issues 51/1, 51/3, 2008.
- Downey & Thilikos "Confronting Intractability via Parameters" Computer Science Review. 5(4), 2011, pp. 279–317

Outline


disguised Horn
Tree-like
Horn
d
2CNF

Some notation

- We consider propositional formulas in Conjunctive Normal Form (CNF) as sets of clauses.
- A truth assignment is a mapping $\tau : X \to \{0, 1\}$, where X is a set of variables.
- $F[\tau]$ denotes the CNF formula obtained from F by removing all satisfied clauses and removing false literals from the remaining clauses.
- $F - X$ denotes the CNF formula obtained from F by removing all literals $x, \overline{x}$ for $x \in X$ from all clauses.
- Note that $F[\tau] \subseteq F - X$.

Motivating Example: Distance from Horn

- Consider a CNF formula $F = \big\{\{u, v, w\}, \{\overline{u}, x, \overline{y}\}, \{u, \overline{v}, \overline{x}, y\}, \{v, y, \overline{z}\}, \{u, v, \overline{w}, z\}\big\}$.
- Consider a set of variables $X = \{u, v\}$.
- We try out all possible truth assignments to X.
- $F[u = 0, v = 0] = \big\{\{w\}, \{y, \overline{z}\}, \{\overline{w}, z\}\big\} \in$ HORN.
- $F[u = 0, v = 1] \in$ HORN.
- $F[u = 1, v = 0] \in$ HORN.
- $F[u = 1, v = 1] \in$ HORN.

- For each possible truth assignment $\tau : X \to \{0, 1\}$, the formula $F[\tau]$ is Horn.
- Hence we can decide the satisfiability of F by checking $2^{|X|}$ Horn formulas, which can be checked in polynomial time.
- The time required to solve F is now $O^*(2^k)$ where k is the size of set X, hence FPT for parameter k.
- We can consider the parameter k as the *distance* from the easy class Horn.

Strictly Tractable Classes

Let $\mathcal{C}$ be a class of CNF formulas. We call $\mathcal{C}$ *strictly tractable* if it has the following properties:

- $\mathcal{C}$ can be recognized in polynomial time.
- For $F \in \mathcal{C}$ we can decide the satisfiability in polynomial time.
- $\mathcal{C}$ is closed under partial assignments (i.e., if $F \in \mathcal{C}$ then $F[\tau] \in \mathcal{C}$)
- $\mathcal{C}$ is closed under isomorphisms (if two formulas only differ in the names of their variables, either both are in $\mathcal{C}$ or none is in $\mathcal{C}$).

Some Well-Known Strictly Tractable Classes

- Horn formulas
- 2CNF formulas
- Acyclic formulas
- Matched formulas
- Renamable Horn formulas

Backdoor Sets (BDS)

Let $\mathcal{C}$ be a strictly tractable class of CNF formulas, F a CNF formula and $X \subseteq \mathsf{var}(F)$.

- X is a *strong $\mathcal{C}$-backdoor set* of F if for each $\tau \in 2^X$ we have $F[\tau] \in \mathcal{C}$.
- X is a *weak $\mathcal{C}$-backdoor set* of F if there is some $\tau \in 2^X$ with $F[\tau] \in \mathcal{C}$ such that $F[\tau]$ is satisfiable.
- X is a *deletion $\mathcal{C}$-backdoor set* of F if $F - X \in \mathcal{C}$.

Backdoor Set Evaluation and Detection

- If $\mathcal{C}$ is a strictly tractable class and we are given F and a strong $\mathcal{C}$ backdoor set of F of size k, then deciding the satisfiability of F is fixed-parameter tractable in k.
- We call this problem *backdoor set evaluation*.
- Hence the main challenging problem is to find a small backdoor set (*backdoor set detection*).

- We formulate the following parameterized problem:
  Strong $\mathcal{C}$-BDS Detection
  - Instance: a CNF formula F, an integer k.
  - Parameter: k.
  - Question: does F have a strong $\mathcal{C}$-backdoor set of size at most k?
- Weak $\mathcal{C}$-BDS Detection is defined similarly.

XP-membership

Fact: if $\mathcal{C}$ is a strictly tractable class, then

- Strong $\mathcal{C}$-BDS Detection
- Weak $\mathcal{C}$-BDS Detection

all clearly in XP.

Obstructions

- A smallest obstruction for a CNF formula F to be not in HORN are two positive literals in some clause of F.
- Hence we define the "obstruction" graph $G(F) = (\mathsf{var}(F), E)$ with $uv \in E$ iff F contains a clause C with $u, v \in C$.

Finding small BDS

- Any strong BDS must intersect with all the obstructions.
- In other words, we need to find a vertex cover of the obstruction graph
- Hence, with a bounded search tree we can find a BDS of size $\leq k$ if it exists, in time $O^*(2^k)$.
- There are much faster algorithms for VC: $O^*(1.273^k)$.
- *Strong* HORN-*BDS Detection is FPT* (standard parameterization).

Strong 2CNF-BDS Detection

- A smallest obstruction for a CNF formula F to be not in 2CNF are three literals in some clause of F.
- We define an "obstruction" hypergraph:
- $H(F) = (\mathsf{var}(F), E)$ with $uvw \in E$ if F contains a clause C with $u, v, w \in \mathsf{var}(C)$ (all three are distinct).
- We need to find a hitting set of this 3-uniform hypergraph.
- Simple bounded search tree: $O^*(3^k)$.
- Faster algorithm: $O^*(2.270^k)$.
- *Strong* 2CNF-*BDS Detection is FPT* (standard parameterization).

Weak BDS detection

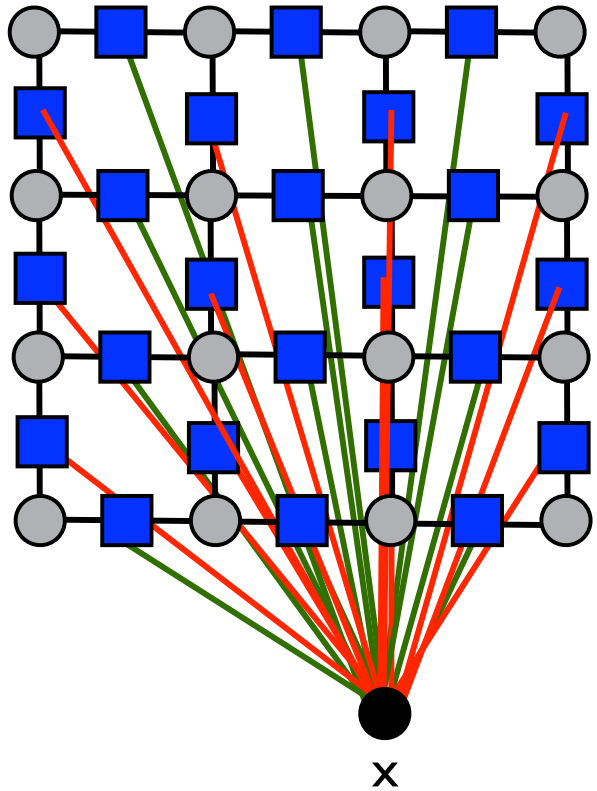
- *Theorem:* Weak HORN-BDS Detection and Weak 2CNF-BDS Detection are W[2]-complete.
- Reduce from DS.
- Let (G, k) be an instance of DS. Wlog assume the minimum degree is at least 2.
  Consider the vertices as variables.
  For each vertex v introduce a clause $N[v]$.
  Claim: for a set $X \subseteq V(G)$ the following are equivalent:
  1. X is a dominating set of G
  2. X is a weak HORN-bds of F.
  3. X is a weak 2CNF-bds of F.

- Note, however, that if the input is 3CNF, then weak BDS detection is FPT.
  (SAT 2013: Neeldhara Misra, Sebastian Ordyniak, Venkatesh Raman and Stefan Szeider: *Upper and Lower Bounds for Weak Backdoor Set Detection*.)

Model Counting

- Strong backdoors can be used for model counting if the base class admits polynomial-time model counting.

Strong vs Deletion BDS Detection



$\{x\}$ forms a strong backdoor into the class of acyclic formulas. A smallest deletion backdoor into this class needs to be large.

Strong vs Deletion BDS Detection

Class	Strong	Deletion
Horn	FPT	FPT
2CNF	FPT	FPT
RHorn	W[2]	FPT [RO'08]
Acyclic	FPT-approx [GS'12]	FPT
Bounded TW	FPT-approx [GS'13]	FPT

Further reading...

- Gottlob & Szeider "Fixed-parameter algorithms for artificial intelligence, constraint satisfaction, and database problems" The Computer Journal 51(3), 2006.
- Samer & Szeider, "Fixed-Parameter Tractability", Chapter 13 of the Handbook of Satisfiability, IOS Press, 2009.
- Gaspers & Szeider: Backdoors to Satisfaction. Survey Paper, Fellows Festschrift, Springer 2012.

Outline

The Lost Continent of Polytime Preprocessing

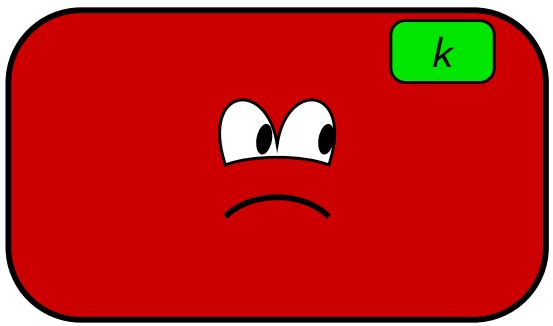
If we can reduce the size of a problem instance in polynomial time by one bit, then we can solve the problem in polynomial time.

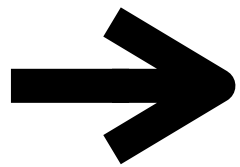
So classical (one-dimensional) Algorithmics is not well-suited for studying preprocessing.

Parameterized Complexity

- In parameterized complexity we can measure the power of preprocessing in terms of the parameter.
- For example, we can ask if we can preprocess a VC instance (G, k) such that we are left with an instance with $f(k)$ vertices only.

Parameterized Complexity





polynomial-time

size < *f*(*k*)

Preprocessing for VC

- Consider an instance (G, k) of VC.
- Consider a vertex v with more than k neighbors. If the instance has a vertex cover S of size k then $v \in S$ (because S cannot contain all neighbors of v).
- This yields the a problem kernel with a quadratic number of vertices.

Problem Kernel

Definition: Let $L \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized decision problem.

A function $R : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ is a *kernelization* of L if there exists a computable function g such that the following holds true:

- $R(x, k) = (x', k')$ implies that $k' \leq k$ and $|x'| \leq g(k)$
- $(x, k) \in L$ if and only if $R(x, k) \in L$;
- R can be computed in polynomial time (polynomial in $|x| + k$).

$R(x, k) = (x', k')$ is called a "problem kernel."

Kernelization Lemma

Kernelization Lemma: A decidable parameterized problem is fixed-parameter tractable if and only if it admits a kernelization.

Even Smaller Kernels for VC

- Using a theorem of Nemhauser-Trotter 1975 on approximation for VC on can get a kernel for VC with $\leq 2k$ vertices.
- Hence Strong Horn-BDS detection has a kernel with $2k$ variables.

No Polynomial Kernels?

- Many problems such as VC, 3HS, have kernels of polynomial-size.
- For other problems, no polynomial-size kernels are known.
- It would is highly desirable to get a theoretical justification whether a problems has no polynomial kernel.

SAT

- 3SAT(vars) has trivially a polynomial kernel.
- How about SAT(vars)?
- *Theorem: Fortnow & Santhanam STOC'08: if SAT(vars) has a polynomial kernel, then* $PH = \Sigma_p^3$ ("the Polynomial Hierarchy collapses to its third level").
- The proof is combinatorial and uses Yap's Theorem.
  (NP $\subseteq$ co-NP/poly implies PH $= \Sigma_p^3$).

Backdoor Evaluation

- Let $\mathcal{C}$ be a tractable class of CNF formulas.
- Strong $\mathcal{C}$-BDS Evaluation
  - Instance: a CNF formula F and a strong $\mathcal{C}$-backdoor of F of size k.
  - Parameter: k.
  - Question: Is F satisfiable?
- The problem is clearly fixed-parameter tractable. But does it admit a polynomial kernel?
  A polynomial kernel would be highly desirable. We will show that unfortunately a polynomial kernel is unlikely for most base classes $\mathcal{C}$.

Application to Backdoor Evaluation

- We have a direct consequence of the above:
- *Theorem:* Strong $\mathcal{C}$-BDS Evaluation has no polynomial kernel for any strictly tractable class $\mathcal{C}$ unless the PH collapses.

3CNF

- How about Strong $\mathcal{C}$-BDS Evaluation for 3CNF?
- Since 3SAT(vars) has a polynomial kernel, we cannot use such a simple approach.
- The method of composition allows to get super-polynomial kernel lower bounds also for 3CNF.

Backdoor Evaluation for 3CNF Formulas I

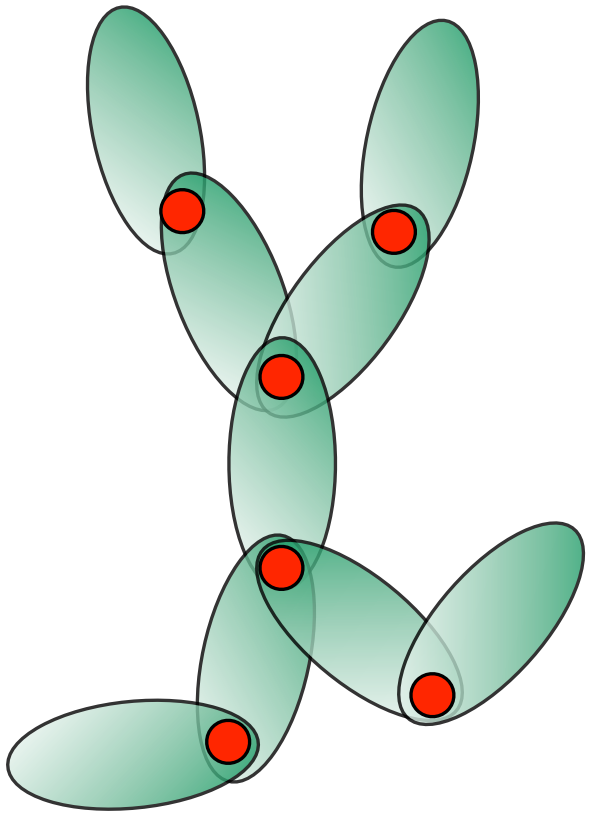
- *Theorem:* Let $\mathcal{C} \in \{\text{HORN}, \text{2CNF}\}$. Strong $\mathcal{C}$-BDS Evaluation for 3CNF formulas has no polynomial kernel unless PH collapses. [S'11]

Further reading...

- Bodlaender et al. "On problems without polynomial kernels" J. of Computer and System Sciences, 75, 423-434, 2009.
- Dom, Lokshtanov, Saurabh "Incompressibility through Colors and IDs" ICALP (1) 2009: 378-389
- Szeider "Limits of Preprocessing" AAAI 2011, 93-98.

Outline

Decmpositions: the general idea

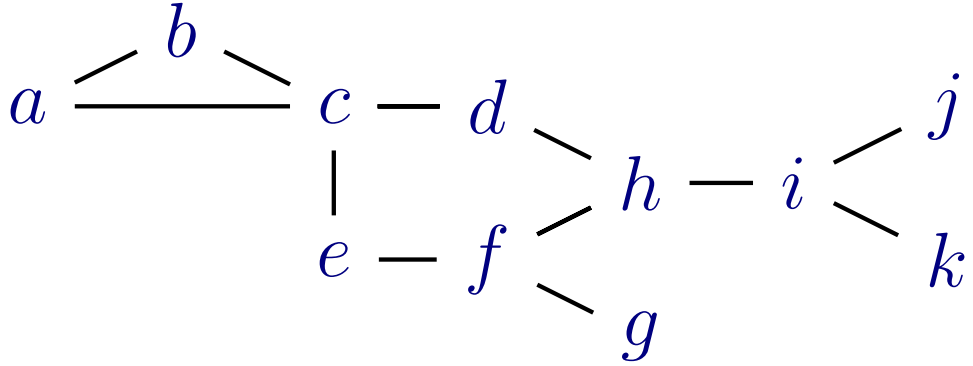


Idea: decompose the problem into subproblems and combine solutions to subproblems to a global solution.

Parameter: overlap between subproblems.

Tree decompositions (by example)

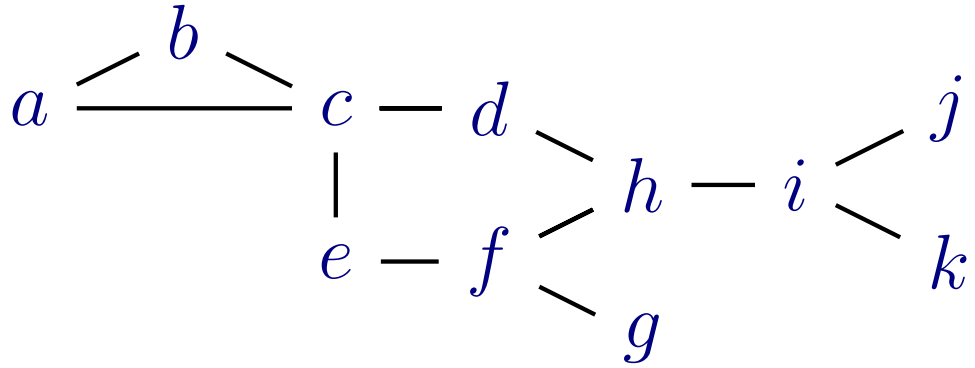
- A graph G



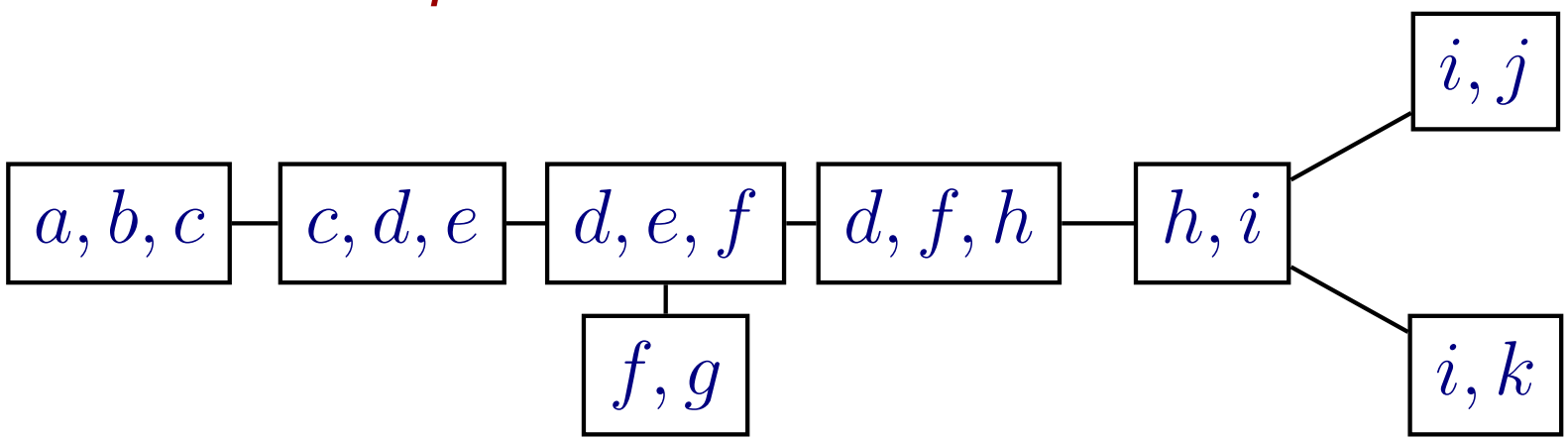
- A *tree decomposition* of G

Tree decompositions (by example)

- A graph G

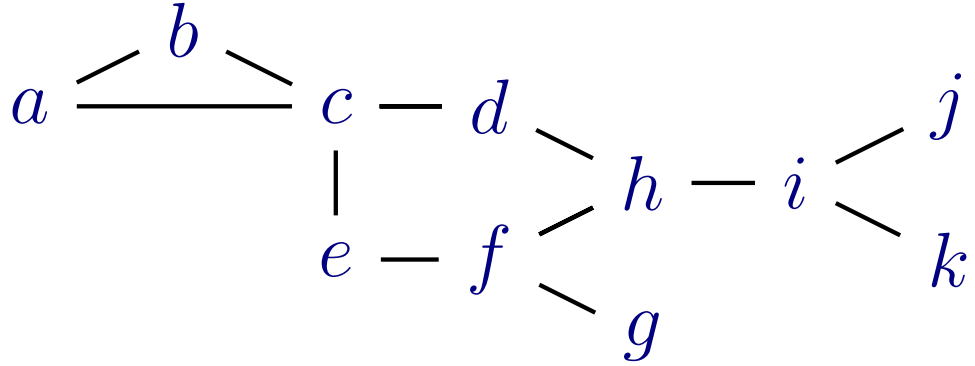


- A *tree decomposition* of G

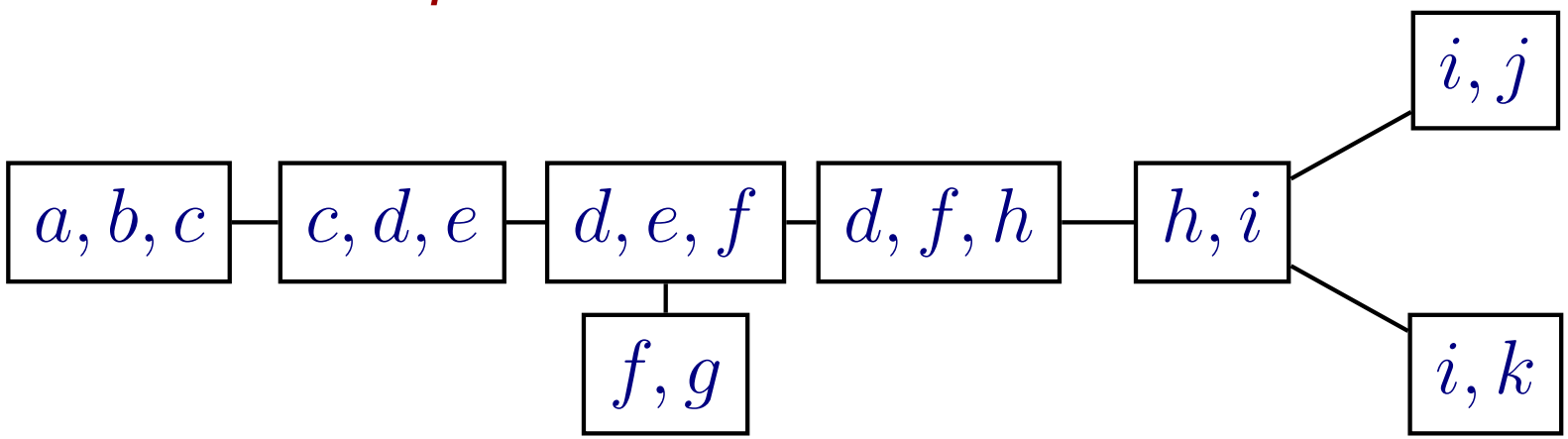


Tree decompositions (by example)

- A graph G



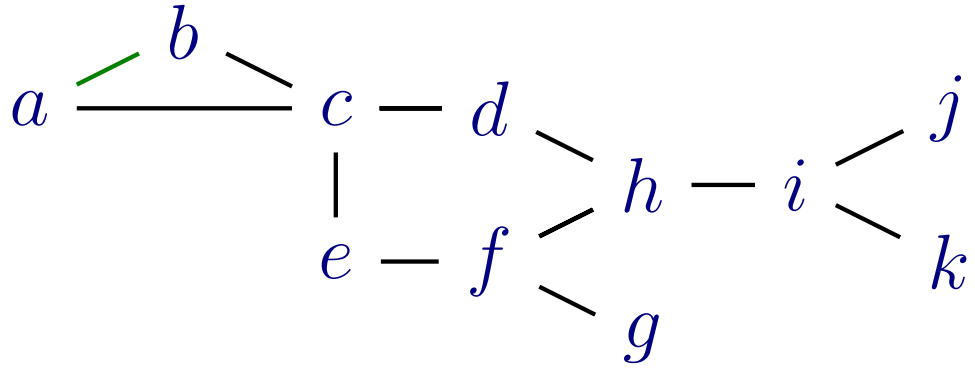
- A *tree decomposition* of G



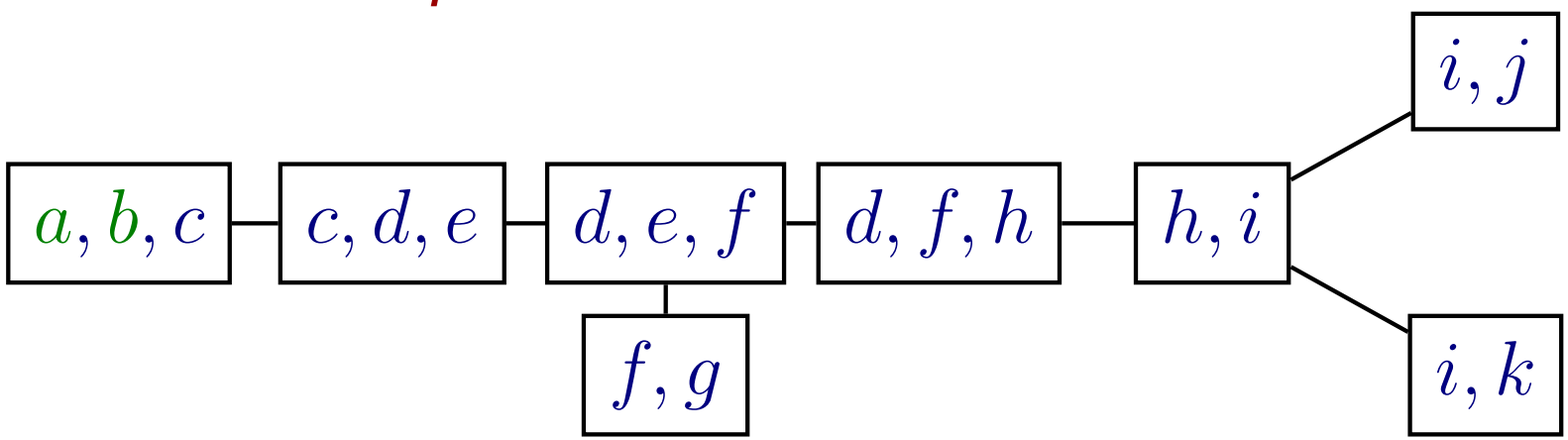
Conditions:

Tree decompositions (by example)

- A graph G



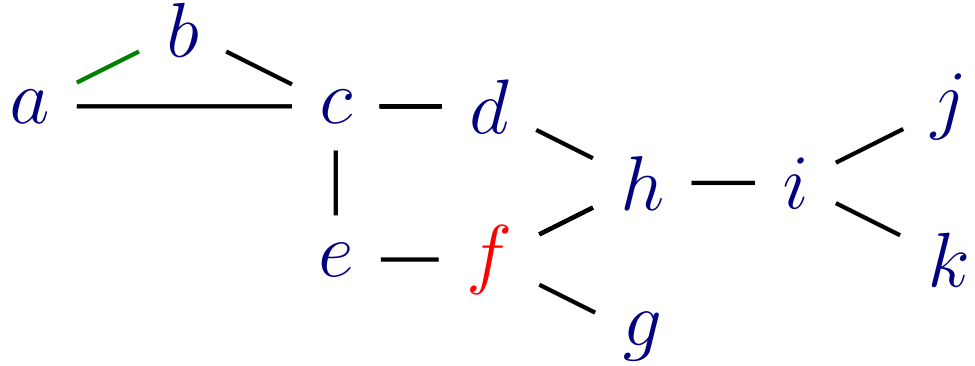
- A *tree decomposition* of G



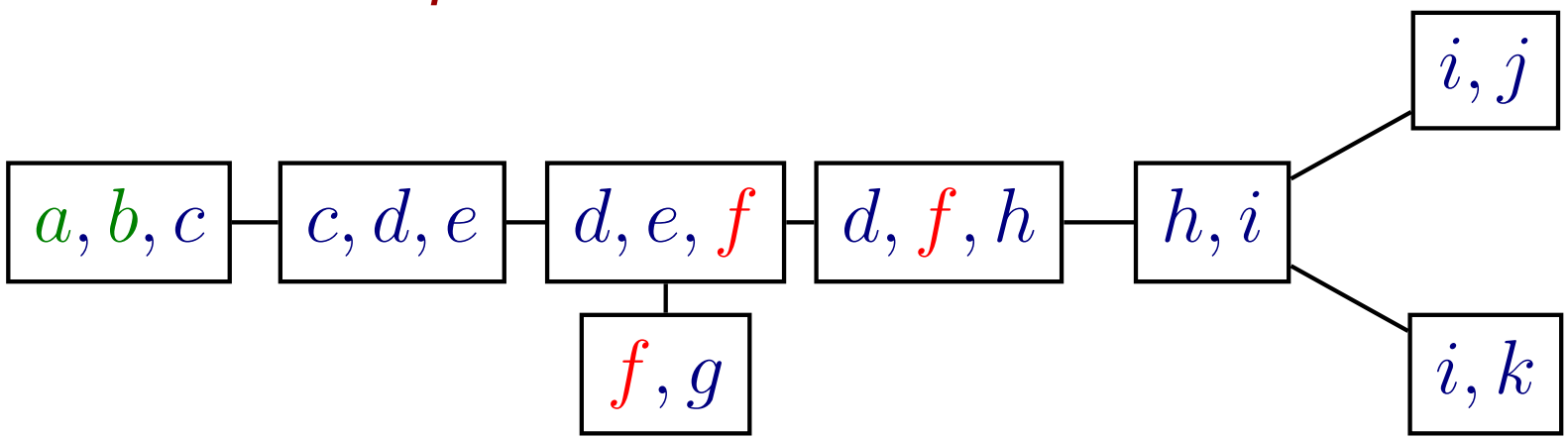
Conditions: covering

Tree decompositions (by example)

- A graph G



- A *tree decomposition* of G



Conditions: covering and connectedness.

Tree decomposition (more formally)

- Let G be a graph, T a tree, and χ a labeling of the vertices of T by sets of vertices of G.
- We refer to the vertices of T as "nodes", and we call the sets $\chi(t)$ "bags".
- The pair (T, χ) is a *tree decomposition* of G if the following three conditions hold:
  1. For every vertex v of G there exists a node t of T such that $v \in \chi(t)$.
  2. For every edge vw of G there exists a node t of T such that $v, w \in \chi(t)$ ("covering").
  3. For any three nodes t_1, t_2, t_3 of T, if t_2 lies on the unique path from t_1 to t_3, then $\chi(t_1) \cap \chi(t_3) \subseteq \chi(t_2)$ ("connectedness").

- The *width* of a tree decomposition (T, χ) is defined as the maximum $|\chi(t)| - 1$ over all nodes t of T.
- The *treewidth* $\mathsf{tw}(G)$ of a graph G is the minimum width over all its tree decompositions.

Basic Facts

- Trees have treewidth 1.
- Cycles have treewidth 2.
- The complete graph on n vertices has treewidth $n - 1$.
- If a graph G contains a clique K_r, then every tree decomposition of G contains a node t such that $K_r \subseteq \chi(t)$ (Helly property of subtrees of trees).

Complexity of Treewidth

- Determining the treewidth of a graph is NP-hard.
- For every fixed k, one can check for a graph G in linear time whether $\mathsf{tw}(G) \leq k$. (*Bodlaender's Theorem*)

Easy problems for bounded treewidth

- Many graph problems that are polynomial time solvable on trees are FPT with parameter treewdith.
- Two general methods:
  - *Dynamic programming*: compute local information in a bottom-up fashion along a tree decomposition
  - *Monadic Second Order Logic*: express graph problem in some logic formalism and use a meta-algorithm

Monadic Second Order Logic

- *Monadic Second Order* (MSO) Logic is a powerful formalism for expressing graph properties. One can quantify over vertices, edges, vertex sets, and edge sets.
- *Courcelle's theorem:* Checking whether a graph G satisfies an MSO property is FPT parameterized by the treewidth of G plus the length of the MSO expression.

MSO Logic (2)

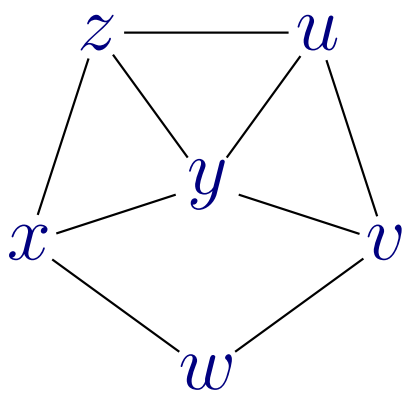
- Example: 3-Colorability,
  - "there are three sets of vertices which form a partition of V such that no edge has both ends in the same set"
  - $\exists A \subseteq V \; \exists B \subseteq V \; \exists C \subseteq V$
    $A \cup B \cup C = V \wedge A \cap B = A \cap C = B \cap C = \emptyset$
    $\wedge \forall e \in E \; \forall u \in V \; \forall v \in V$
    $\wedge inc(u, e) \wedge inc(v, e) \wedge u \neq v$
    $\rightarrow$
    $\neg(u \in A \wedge v \in A) \wedge \neg(u \in B \wedge v \in B) \wedge \neg(u \in C \wedge v \in C)$

Treewidth of a Logic Problem?

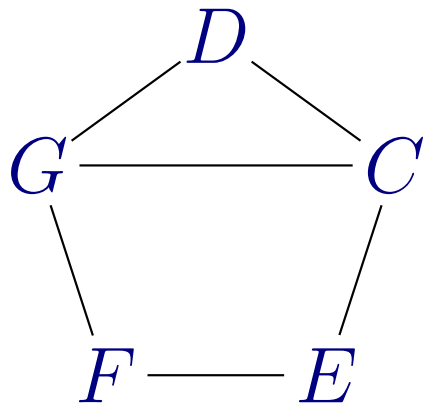
- associate a graph with the reasoning instance
- take the tree decomposition of the graph
- most widely used: primal graphs, incidence graphs, and dual graphs.

Three Treewidth Parameters

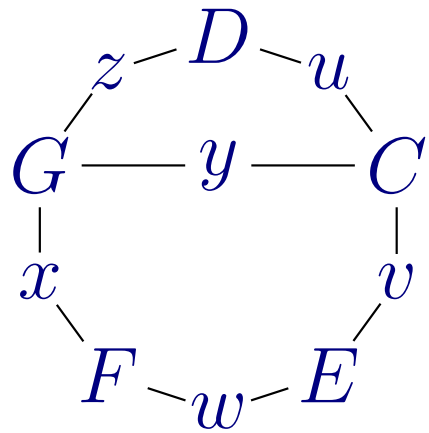
CNF Formula $F = \{C, D, E, F, G\}$ where $C = \{u, v, \overline{y}\}$, $D = \{\overline{u}, z\}$, $E = \{\overline{v}, w\}$, $F = \{\overline{w}, x\}$, $G = \{x, y, \overline{z}\}$.



primal graph



dual graph



incidence graph

Gives rise to parameters *primal treewidth, dual treewidth,* and *incidence treewidth.*

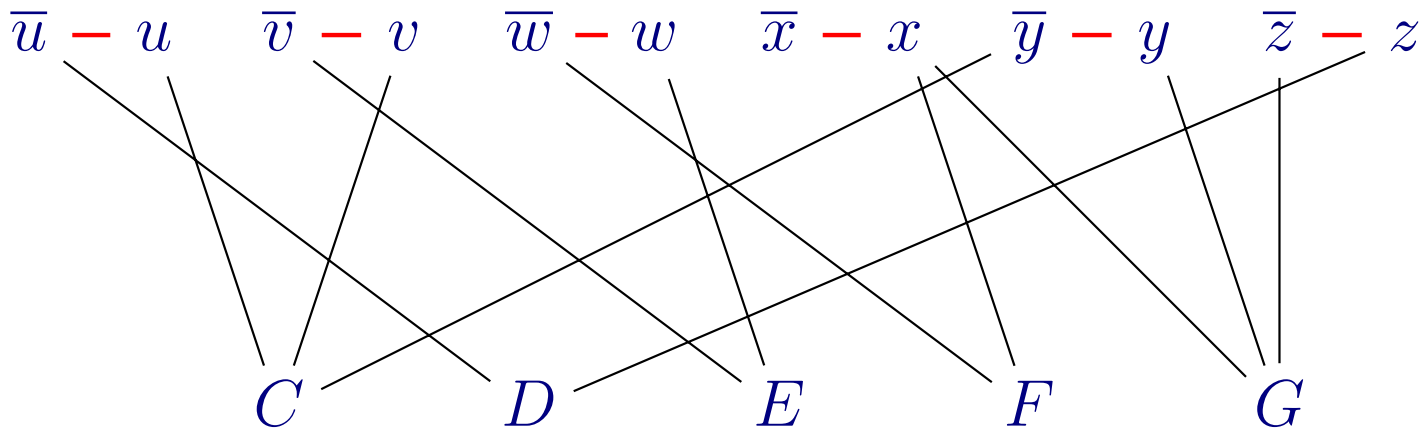
Incidence treewidth is most general

- *Incidence tw* $\leq$ *primal tw* $+1$.
  - Proof: take tree decomposition (T, χ) of primal graph.
  - For each clause C there is a node t of T with $\mathsf{var}(C) \subseteq \chi(t)$.
  - Add to t a new neighbor t' with $\chi(t') = \chi(t) \cup \{C\}$.
- *Incidence tw* $\leq$ *dual tw* $+1$. (Proof: analog)
- *Primal and dual tw are incomparable.*
  - One big clause alone gives large primal treewidth.
  - $\{\{x, y_1\}, \{x, y_2\}, \ldots, \{x, y_n\}\}$ gives large dual treewidth.

SAT is FPT for parameter incidence tw.

CNF Formula $F = \{C, D, E, F, G\}$ where $C = \{u, v, \overline{y}\}$, $D = \{\overline{u}, z\}$, $E = \{\overline{v}, w\}$, $F = \{\overline{w}, x\}$, $G = \{x, y, \overline{z}\}$.

Auxiliary graph:



- MSO Formula: *"There exists an independent set of literal vertices that dominates all the clause vertices."*
- Treewidth of auxilary graph is at most twice the treewidth of the incidence graph plus one.

FPT via MSO

Theorem: SAT(primal tw), SAT(dual tw), and SAT(incidence tw) are FPT.

Further reading...

- Kloks "Treewidth: Computations and Approximations", Springer 1994.
- Bodlaender & Koster, "Combinatorial Optimization on Graphs of Bounded Treewidth" The Computer Journal 51(3), 255-269, 2008
- Hlinený, Oum, Seese, Gottlob, "Width Parameters Beyond Tree-width and their Applications" The Computer Journal 51(3), 326-362, 2008
- Samer & Szeider, "Constraint Satisfaction with Bounded Treewidth Revisited" J. of Computer and System Sciences, 76(2), 103-114, 2010.

Outline

Local Search (LS)

- LS is one of the most fundamental algorithmic concepts
- LS has been successfully applied to a wide range of hard combinatorial optimization problems, in particular to:
  - Maximum Satisfiability (Max Sat)
    Given a CNF formula F, find an assignment that satisfies as many clauses of F as possible.
  - Traveling Salesperson Problem (TSP).
- Basic idea: move as long as possible from one candidate solution to a "better" neighboring candidate solution.

LS for Max Sat

- candidate solutions: truth assignments
- a better solution satisfies more clauses
- two candidate solutions are k-*flip neighbors* if they differ in at most k variables.

Avoiding Local Optima

- Main obstacle for LS: to get stuck at a local optimum.
- Approaches:
  - (A) Heuristic moves to non-improving solutions, random restarts, etc.
  - (B) Increase value of k (most algorithms use $k = 1$ only).

Computational Problems

- k-Flip Max Sat
  - *Instance:* A CNF formula F and a truth assignment $\tau : \mathsf{var}(F) \to \{0, 1\}$.
  - *Question:* Is there a k-flip neighbor τ' of τ that satisfies more clauses of F than τ?
- k-Flip Sat
  - *Instance:* A CNF formula F and a truth assignment $\tau : \mathsf{var}(F) \to \{0, 1\}$.
  - *Question:* Is there a k-flip neighbor τ' of τ that satisfies all clauses of F?
- k is the natural parameter.
- Both problems are trivially in XP.

Parameterized Complexity of LS

- *Question:* is k-Flip Max Sat fixed-parameter tractable for parameter k?
- *Question:* is k-Flip Sat fixed-parameter tractable for parameter k?
- If not in general, then under what reasonable side conditions?

Max Sat and variants

- Intuition suggests that k-Flip Max Sat is not FPT (is this the case indeed?)
- What if the *size of clauses is bounded* (such as in Max 3SAT or Max 2SAT?)
- What if the *number of occurrences of variables* is bounded?
- Are there cases where k-Flip Sat is of different parameterized complexity than k-Flip Max Sat?

Results

size of clauses	occurrence of variables	k-Flip Max Sat	k-Flip Sat
unbounded	unbounded	W[1]-hard	W[2]-hard
unbounded	bounded	W[1]-hard	W[1]-hard
bounded	unbounded	W[1]-hard	FPT
bounded	bounded	FPT	FPT

- Good news: There are nontrivial cases that are fixed-parameter tractable
- For k-Flip SAT, bounding the number of variable occurrences makes no difference. But it makes a difference for k-Flip Max SAT

Further Results

- Ronald de Haan, Iyad A. Kanj, and Stefan Szeider. *Local Backbones.* SAT 2013.
- Find a subset $F' \subseteq F$, $|F'| \leq k$, such that $F \models x$.
- Find a subset $F' \subseteq F$, $|F'| \leq k$, such that F is unsatisfiable.

Further reading...

- Krokhin & Marx: On the Hardness of Losing Weight. ICALP (1) 2008: 662-673.
- Fellows et al. "Local Search: Is Brute-Force Avoidable?" IJCAI 2009: 486-491.
- Szeider "The Parameterized Complexity of k-Flip Local Search for SAT and MAX SAT" Discrete Optimization 8, 139-145, 2011.

Summary

- Parameterized Complexity is a theoretical framework that supports structural/qualitative aspects for a worst-case complexity analysis.
- PC has the potential for a more realistic complexity theory.
- We still have a theory/practise gap, but PC helps to make it smaller.
- There are many interesting questions that one can ask within the PC framework, which do not make much sense in classical complexity.