The Theory of Parameterized Complexity

A Branch of Algorithmic Complexity Theory

Minko Markov

minkom@fmi.uni-sofia.bg

Faculty of Mathematics and Informatics
Sofia University "St. Climent Ohridski"

January 16, 2015

Content I

Content II

4. Fast algorithms for VERTEX COVER when k is small
   - Algorithm of Fellows and Downey with complexity $O(2^k n)$
   - Algorithm of Buss with complexity $O(n + 2^{2k^2})$
5. The Theory of Parameterized Complexity
   - Foundations of Parameterized Complexity
   - VERTEX COVER as a parameterized problem
6. Fixed Parameter Tractable (FPT) Problems
   - $\mathcal{FPT}$: the parameterized analogue of $\mathcal{P}$
   - Klam value
   - The same classical problem can be turned into a parameterized problem in different ways
   - Constructing FPT algorithms
   - FPT reductions

Intractability proofs as ends in themselves

When we want to get something done, knowing it contains a provably intractable part is little consolation

Joseph Felsenstein, *a molecular biologist*, *said* in 1997:

> *About ten years ago some computer scientists came by and said they heard we have some really cool problems. They showed that the problems are $\mathcal{NP}$-complete and went away!*

Approaches against intractability

1. Heuristics: good ideas with no or insufficient analysis of their goodness. Genetic algorithms, for instance.

Approaches against intractability

1. Heuristics: good ideas with no or insufficient analysis of their goodness. Genetic algorithms, for instance.
2. *Average case complexity*. The classical intractability results are based on considering worst cases only. If those worst cases occur seldom enough there can exist algorithms that are fast on most inputs and thus the intractable problem is, in a very practical sense, tractable. Despite the progress in that research, no definite breakthrough has come from it.

Approaches against intractability

3. Approximation algorithms: an approximate answer with guaranteed deviation from the optimum one is better than no answer at all. However, a lot of problems turn out to be unapproximable.

Approaches against intractability

3. Approximation algorithms: an approximate answer with guaranteed deviation from the optimum one is better than no answer at all. However, a lot of problems turn out to be unapproximable.
4. Randomized algorithms: using randomness as a resource. It is not known whether that can turn intractable problems into tractable ones, in the probabilstic sense.

Approaches against intractability

3. Approximation algorithms: an approximate answer with guaranteed deviation from the optimum one is better than no answer at all. However, a lot of problems turn out to be unapproximable.
4. Randomized algorithms: using randomness as a resource. It is not known whether that can turn intractable problems into tractable ones, in the probabilstic sense.
5. Quantum computers: utilise Quantum Mechanics to bypass the limitations of classical computers. So far they are a mere possibility. Furthermore, it is unclear if they can solve $\mathcal{NP}$-complete problems efficiently.

Approaches against intractability
Restricted easy versions

6. Conisder a *restricted* easy version of the problem. Every hard problem has easy restricted versions (just as any easy problem can be generalized to a hard one). The restricted version should better be useful, of course.

Approaches against intractability
Restricted easy versions

6. Conisder a *restricted* easy version of the problem. Every hard problem has easy restricted versions (just as any easy problem can be generalized to a hard one). The restricted version should better be useful, of course.

   That is a very old idea. For example:
   - The $O(nB)$ algorithm for PARTITION where B is the sum of the values. The restriction is on the values.
   - The $O(n)$ algorithms for almost all $\mathcal{NP}$-complete graph problems on trees or series-parallel graphs. The restriction is on the graphs.

Approaches against intractability
Restricted easy versions

6. Conisder a *restricted* easy version of the problem. Every hard problem has easy restricted versions (just as any easy problem can be generalized to a hard one). The restricted version should better be useful, of course.

   That is a very old idea. For example:

   - The $O(nB)$ algorithm for PARTITION where B is the sum of the values. The restriction is on the values.
   - The $O(n)$ algorithms for almost all $\mathcal{NP}$-complete graph problems on trees or series-parallel graphs. The restriction is on the graphs.

   In *some* sense, Parameterized Complexity belongs to this approach.

Parameterized Complexity Theory

- It deals with intractable problems and takes for granted intractability exists.

Parameterized Complexity Theory

- It deals with intractable problems and takes for granted intractability exists.
- Parameterized Complexity considers the structure of intractable problems in much more detail than classical Complexity Theory.

Parameterized Complexity Theory

- It deals with intractable problems and takes for granted intractability exists.
- Parameterized Complexity considers the structure of intractable problems in much more detail than classical Complexity Theory.
- According to it, complexity is a function of two variables:
  - the size of the input
  - something called *parameter*; roughly speaking, that is what makes the problem intractable.

The Vertex Cover problem

In the classical complexity theory, every decision problem is defined by two components: a generic instance and a Yes/No question.

The Vertex Cover problem

In the classical complexity theory, every decision problem is defined by two components: a generic instance and a Yes/No question.

Definition of Vertex Cover:

The Vertex Cover problem

In the classical complexity theory, every decision problem is defined by two components: a generic instance and a Yes/No question.

Definition of Vertex Cover:

- An ordered pair of an undirected graph $G = (V, E)$ and a natural number k.

The Vertex Cover problem

In the classical complexity theory, every decision problem is defined by two components: a generic instance and a Yes/No question.

Definition of Vertex Cover:

- An ordered pair of an undirected graph $G = (V, E)$ and a natural number k.
- Does G have vertex cover of size $\leq k$?

The VERTEX COVER problem

In the classical complexity theory, every decision problem is defined by two components: a generic instance and a YES/NO question.

Definition of VERTEX COVER:

- An ordered pair of an undirected graph $G = (V, E)$ and a natural number k.
- Does G have vertex cover of size $\leq k$?

Vertex cover of $G = (V, E)$ is every subset $U \subseteq V$ such that

$$\forall (x, y) \in E : x \in U \vee y \in U$$

Complexity of VERTEX COVER

VERTEX COVER is one of the "original" 21 $\mathcal{NP}$-complete problems of Karp *from 1972*. Under the assumption that $\mathcal{P} \neq \mathcal{NP}$, the problem is intractable.

Complexity of Vertex Cover

Vertex Cover is one of the "original" 21 $\mathcal{NP}$-complete problems of Karp *from 1972*. Under the assumption that $\mathcal{P} \neq \mathcal{NP}$, the problem is intractable.

In the worst case its solution reduces to testing all subsets of sizes $2, 3, \ldots, k$. Their number imposes a lower bound $\sum_{j=2}^{k} \binom{n}{j}$ on the complexity of the brutal force approach.

Complexity of Vertex Cover

Vertex Cover is one of the "original" 21 $\mathcal{NP}$-complete problems of Karp *from 1972*. Under the assumption that $\mathcal{P} \neq \mathcal{NP}$, the problem is intractable.

In the worst case its solution reduces to testing all subsets of sizes $2, 3, \ldots, k$. Their number imposes a lower bound $\sum_{j=2}^{k} \binom{n}{j}$ on the complexity of the brutal force approach.

$\sum_{j=2}^{k} \binom{n}{j}$ is a very fast growing function. If $k \approx n$ then $\sum_{j=2}^{k} \binom{n}{j} \approx 2^n$. The middle binomial coefficient alone is such that $\binom{n}{\lfloor n/2 \rfloor} = \Theta(2^n/\sqrt{n})$.

Complexity of VERTEX COVER
On the complexity of the brutal force approach

To claim that $\sum_{j=2}^{k} \binom{n}{j} = \Theta(n^k)$ is **wrong**, unless k is a constant. If that were true then it would be the case that $\sum_{j=2}^{n} \binom{n}{j} = \Theta(n^n)$, which is patently **not true**. The claim $\sum_{j=2}^{k} \binom{n}{j} = O(n^k)$ is true but unconvincing because we discuss lower bounds. Finding a good asymptotic estimation for $\sum_{j=2}^{k} \binom{n}{j}$ is *hard*.

Complexity of Vertex Cover
On the complexity of the brutal force approach

To claim that $\sum_{j=2}^{k} \binom{n}{j} = \Theta(n^k)$ is **wrong**, unless k is a constant. If that were true then it would be the case that $\sum_{j=2}^{n} \binom{n}{j} = \Theta(n^n)$, which is patently **not true**. The claim $\sum_{j=2}^{k} \binom{n}{j} = O(n^k)$ is true but unconvincing because we discuss lower bounds. Finding a good asymptotic estimation for $\sum_{j=2}^{k} \binom{n}{j}$ is *hard*.

It is known that

$$\sum_{j=0}^{\alpha n} \binom{n}{j} = 2^{nH(\alpha) - \frac{1}{2}\lg n + O(1)}$$

where α is a constant such that $0 < \alpha < \frac{1}{2}$ and $H(\alpha) = \alpha \lg \frac{1}{\alpha} + (1-\alpha)\lg\left(\frac{1}{1-\alpha}\right)$ is (*binary entropy*).

An improvement over n^k

The n^k function describes adequately the complexity of the brute force when k is small and n is very big. In numerous appearances of VERTEX COVER in practice, n is certainly huge, for example in *Computational Biology*.

An improvement over n^k

The n^k function describes adequately the complexity of the brute force when k is small and n is very big. In numerous appearances of VERTEX COVER in practice, n is certainly huge, for example in *Computational Biology*.

In 1987 *Mike Fellows* and *Michael Langston* in their *Nonconstructive advances in polynomial-time complexity* (*Nonconstructive Tools for Proving Polynomial-Time Decidability* is free for download) prove the existence of an $O(n^3)$ algorithm for VERTEX COVER, in case **k is fixed**.

An improvement over n^k

The n^k function describes adequately the complexity of the brute force when k is small and n is very big. In numerous appearances of Vertex Cover in practice, n is certainly huge, for example in *Computational Biology*.

In 1987 *Mike Fellows* and *Michael Langston* in their *Nonconstructive advances in polynomial-time complexity* (*Nonconstructive Tools for Proving Polynomial-Time Decidability* is free for download) prove the existence of an $O(n^3)$ algorithm for Vertex Cover, in case **k is fixed**.

Ostensilbly, n^3 is a tremendous progress in comparison with n^k.

An improvement over n^k

The n^k function describes adequately the complexity of the brute force when k is small and n is very big. In numerous appearances of Vertex Cover in practice, n is certainly huge, for example in *Computational Biology*.

In 1987 *Mike Fellows* and *Michael Langston* in their *Nonconstructive advances in polynomial-time complexity* (*Nonconstructive Tools for Proving Polynomial-Time Decidability* is free for download) prove the existence of an $O(n^3)$ algorithm for Vertex Cover, in case **k is fixed**.

Ostensilbly, n^3 is a tremendous progress in comparison with n^k.

But note the term "nonconstructive".

On the n^3 algorithm of Fellows and Langston

- In fact, the expression of the complexity is $O(f(k)n^3)$. The function $f(k)$ grows extremely fast. It is "tower" of exponents

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

On the n^3 algorithm of Fellows and Langston

- In fact, the expression of the complexity is $O(f(k)n^3)$. The function $f(k)$ grows extremely fast. It is "tower" of exponents

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

whose height is described by a tower of exponent whose height is described by a tower of exponents and so on, a fixed number of times, whose height is a function of k. That is completely impractical even for $k = 1$.

On the n^3 algorithm of Fellows and Langston

- In fact, the expression of the complexity is $O(f(k)n^3)$. The function $f(k)$ grows extremely fast. It is "tower" of exponents

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

  whose height is described by a tower of exponent whose height is described by a tower of exponents and so on, a fixed number of times, whose height is a function of k. That is completely impractical even for $k = 1$.
- The existence proof is *nonconstructive*. No one knows what the algorithm is. In fact, the algorithm**s**, since for every k there is a different algorithm.

On the n^3 algorithm of Fellows and Langston

- In fact, the expression of the complexity is $O(f(k)n^3)$. The function $f(k)$ grows extremely fast. It is "tower" of exponents

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

  whose height is described by a tower of exponent whose height is described by a tower of exponents and so on, a fixed number of times, whose height is a function of k. That is completely impractical even for $k = 1$.
- The existence proof is *nonconstructive*. No one knows what the algorithm is. In fact, the algorithm**s**, since for every k there is a different algorithm.
- The existence proof follows from the theory of Robertson and Seymour.

Neil Robertson

Paul Seymour

The pivotal theory of Robertson and Seymour

Altogether 23 articles named “Graph minors ...” published between 1983 and 2012 in *Journal of Combinatorial Theory, Series B* and *Journal of Algorithms* on 765 pages totally.

The pivotal theory of Robertson and Seymour

Altogether 23 articles named "Graph minors ..." published between 1983 and 2012 in *Journal of Combinatorial Theory, Series B* and *Journal of Algorithms* on 765 pages totally.

The first 20 papers are basically the proof of what was previously known as *Wagner's conjecture*, now *Theorem of Robertson-Seymour*.

The main result of Robertson and Seymour

Theorem (former Wagner's conjecture)

In every infinite set $\mathcal{F}$ of graphs there is a graph that is isomorphic to a minor of another graph from $\mathcal{F}$.

The main result of Robertson and Seymour

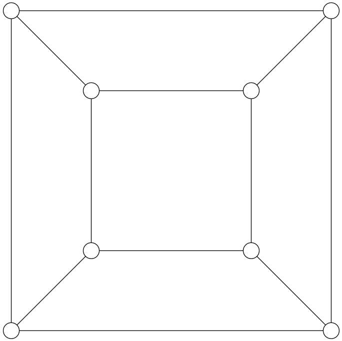
Theorem (former Wagner's conjecture)

In every infinite set $\mathcal{F}$ of graphs there is a graph that is isomorphic to a minor of another graph from $\mathcal{F}$.

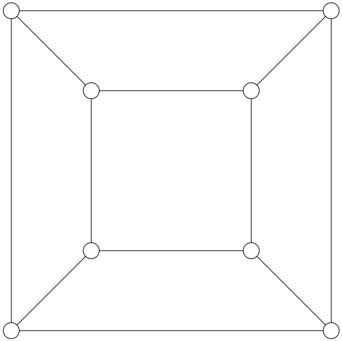
Definition (graph minor)

Let G and H are undirected (finite) graphs. H is a *minor* of G iff H can be obtained from G as the result of a sequence of edge removals (the endpoints stay), vertex deletions (the incident edges are removed) and **edge contractions**. We write $H \preceq G$.

An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$

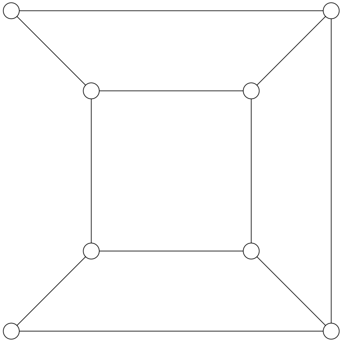


An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$

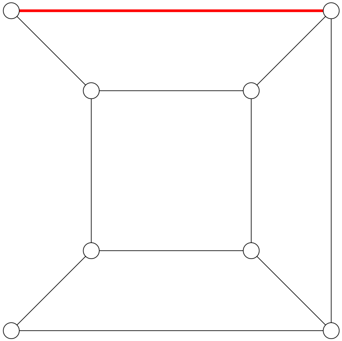


edge removal

An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$

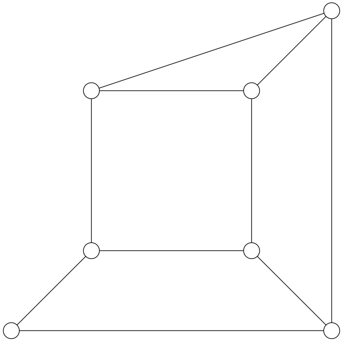


An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$

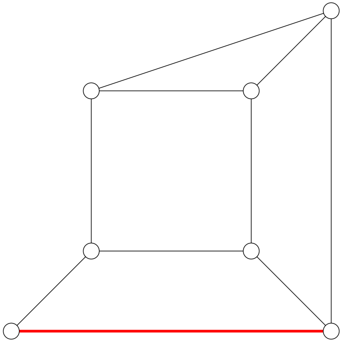


edge contraction

An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$

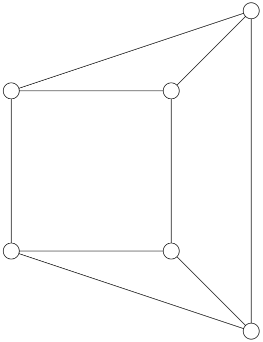


An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$

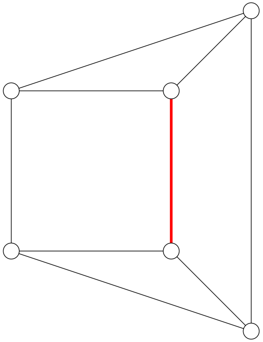


edge contraction

An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$

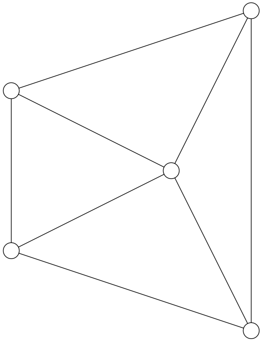


An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$

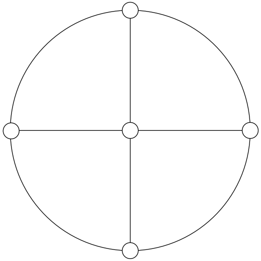


edge contraction

An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$



An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$



Properties of minors

- Edge contraction may lead to multiple edges or loops. We simply ignore them.

Properties of minors

- Edge contraction may lead to multiple edges or loops. We simply ignore them.
- “Minor” is a generalisation of “subgraph” and “subgraph homeomorphic to”. For example, Q_3 has *no* subgraph homeomorphic to W_4 since Q_3 has no degree 4 vertex.

Properties of minors

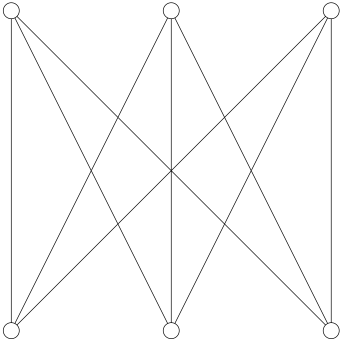
- Edge contraction may lead to multiple edges or loops. We simply ignore them.
- "Minor" is a generalisation of "subgraph" and "subgraph homeomorphic to". For example, Q_3 has *no* subgraph homeomorphic to W_4 since Q_3 has no degree 4 vertex.
- Even if H has less vertices and less edges than G and the degree sequence of H is "comparable" to that of G, it can be the case that $H \not\preceq G$. For example, $K_{3,3} \not\preceq Q_3$ ($K_{3,3}$ is not planar while Q_3 is planar).

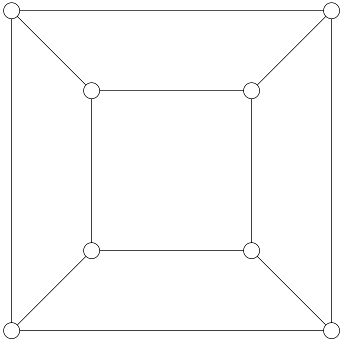
Properties of minors

- Edge contraction may lead to multiple edges or loops. We simply ignore them.
- "Minor" is a generalisation of "subgraph" and "subgraph homeomorphic to". For example, Q_3 has *no* subgraph homeomorphic to W_4 since Q_3 has no degree 4 vertex.
- Even if H has less vertices and less edges than G and the degree sequence of H is "comparable" to that of G, it can be the case that $H \not\preceq G$. For example, $K_{3,3} \not\preceq Q_3$ ($K_{3,3}$ is not planar while Q_3 is planar).
- Roughly speaking, $H \preceq G$ means G contains an H-like substructure, though in a weaker sense than subgraph.

Properties of minors

$K_{3,3} \not\preceq Q_3$ because Q_3 is planar and $K_{3,3}$ is not





The $\preceq$ relation is preorder (quasi-order)

- The $\preceq$ relation on the set of all finite non-isomorphic graphs is a partial order: it is obviously reflexive and transitive, and any two graphs are minors of each other whenever they are isomorphic, therefore it is antisymmetric as well.

The $\preceq$ relation is preorder (quasi-order)

- The $\preceq$ relation on the set of all finite non-isomorphic graphs is a partial order: it is obviously reflexive and transitive, and any two graphs are minors of each other whenever they are isomorphic, therefore it is antisymmetric as well.
- The relation $\preceq$ on the set of all finite graphs with labeled vertices is not a partial order. Such a relation—reflexive and transitive but not necessarily antisymmetric—is *preorder* (alternatively, *quasi-order*).

The $\preceq$ relation is preorder (quasi-order)

- The $\preceq$ relation on the set of all finite non-isomorphic graphs is a partial order: it is obviously reflexive and transitive, and any two graphs are minors of each other whenever they are isomorphic, therefore it is antisymmetric as well.
- The relation $\preceq$ on the set of all finite graphs with labeled vertices is not a partial order. Such a relation—reflexive and transitive but not necessarily antisymmetric—is *preorder* (alternatively, *quasi-order*).
- An example of a preorder, not on graphs though, is the relation R on the finite subsets of an enumerable set A:

$$\forall x\, \forall y : xRy \leftrightarrow |x| \leq |y|$$

WQO: Well Quasi Ordering

Definition

A quasi-order that has no infinite (strictly) decreasing chains and has no infinite antichains is called Well Quasi Ordering (WQO).

Filter and Ideal

Let $\langle S, \leq \rangle$ be a quasi-order and $X \subseteq S$.

- X is *a filter* iff it is upward closed with respect to $\leq$.

Filter and Ideal

Let $\langle S, \leq \rangle$ be a quasi-order and $X \subseteq S$.

- X is *a filter* iff it is upward closed with respect to $\leq$.
- X is *an ideal* iff it is downward closed with respect to $\leq$.

Filter and Ideal

Let $\langle S, \leq \rangle$ be a quasi-order and $X \subseteq S$.

- X is *a filter* iff it is upward closed with respect to $\leq$.
- X is *an ideal* iff it is downward closed with respect to $\leq$.
- *The filter generated by* X is

$$F(X) = \{z \in S \mid \exists a \in X (a \leq z)\}$$

Filter and Ideal

Let $\langle S, \leq \rangle$ be a quasi-order and $X \subseteq S$.

- X is *a filter* iff it is upward closed with respect to $\leq$.
- X is *an ideal* iff it is downward closed with respect to $\leq$.
- *The filter generated by X* is

$$F(X) = \{z \in S \mid \exists a \in X(a \leq z)\}$$

- *The ideal generated by X* is

$$I(X) = \{z \in S \mid \exists a \in X(z \leq a)\}$$

Filter and Ideal

Let $\langle S, \leq \rangle$ be a quasi-order and $X \subseteq S$.

- X is *a filter* iff it is upward closed with respect to $\leq$.
- X is *an ideal* iff it is downward closed with respect to $\leq$.
- *The filter generated by* X is

$$F(X) = \{z \in S \mid \exists a \in X(a \leq z)\}$$

- *The ideal generated by* X is

$$I(X) = \{z \in S \mid \exists a \in X(z \leq a)\}$$

- The filter (ideal) generated by a finite subset of its called *basis*, is a *finitely generated filter (ideal)*.

Alternative characterisation of WQO: Theorem 7.3 from Parameterized Complexity of Fellows and Downey

Theorem (folklore; quoted from Downey and Fellows, 1999)

$\langle S, \leq \rangle$ is a WQO iff for every subset $X \subseteq S$, $F(X)$ is finitely generated.

WQO principle: statement using “filters”

Proposition

Let $\langle S, \leq \rangle$ be a WQO. Assume we can compute in polynomial time whether $x \leq y$ for any $x, y \in S$. Then for every filter F of $\langle S, \leq \rangle$ and any $z \in S$ there is a polynomial time algorithm computing whether $z \in F$.

WQO principle: statement using "filters"

Proposition

Let $\langle S, \leq \rangle$ be a WQO. Assume we can compute in polynomial time whether $x \leq y$ for any $x, y \in S$. Then for every filter F of $\langle S, \leq \rangle$ and any $z \in S$ there is a polynomial time algorithm computing whether $z \in F$.

To make sure z is in the filter it suffices to check that $x \leq z$ for all x of some finite basis.

WQO principle: statement using “filters”

Proposition

Let $\langle S, \leq \rangle$ be a WQO. Assume we can compute in polynomial time whether $x \leq y$ for any $x, y \in S$. Then for every filter F of $\langle S, \leq \rangle$ and any $z \in S$ there is a polynomial time algorithm computing whether $z \in F$.

To make sure z is in the filter it suffices to check that $x \leq z$ for all x of some finite basis.

That principle is better known under the dual formulation using “ideals”. Note that $\mathcal{I}$ is an ideal iff $S \setminus \mathcal{I}$ is a filter.

WQO principle: statement using “ideals”

Definition

Let $\langle S, \leq \rangle$ be a WQO and $\mathcal{I}$ is an ideal of it. Any subset $\mathcal{O} \subseteq S$ is called *an obstruction set for* $\mathcal{I}$ if

$$\forall x : x \in \mathcal{I} \leftrightarrow \forall y_{\, y \in \mathcal{O}} \, (y \not\leq x)$$

WQO principle: statement using "ideals"

Definition

Let $\langle S, \leq \rangle$ be a WQO and $\mathcal{I}$ is an ideal of it. Any subset $\mathcal{O} \subseteq S$ is called *an obstruction set for* $\mathcal{I}$ if

$$\forall x : x \in \mathcal{I} \leftrightarrow \forall y_{\, y \in \mathcal{O}} \, (y \not\leq x)$$

WQO principle (statement using ideals)

Let $\langle S, \leq \rangle$ be a WQO. For every ideal $\mathcal{I}$ in $\langle S, \leq \rangle$ there is a finite obstruction set. Furthermore, if we can compute in polynomial time whether $x \leq y$ for any $x, y \in S$ then we can test membership in the ideal in polynomial time.

Pivotal results of Robertson and Seymour

Theorem (Robertson and Seymour, Graph Minors. XX.)

Wagner's conjecture is true: the set of all finite graphs is well quasi ordered by the minor order $\preceq$.

Theorem (Robertson and Seymour, Graph Minors. XIII.)

The test whether $H \preceq G$ for a **fixed** *graph H can be accomplished in time $O(|V(G)|^3)$.*

Classical results on planarity

Theorem (Kuratowski, 1930)

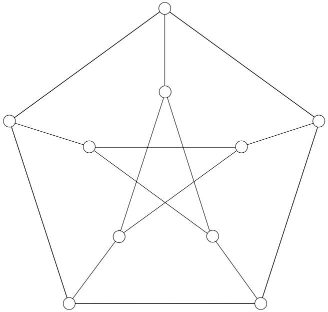
A graph is planar iff it has no subgraph homeomorphic to K_5 or $K_{3,3}$.

Classical results on planarity

Theorem (Kuratowski, 1930)

A graph is planar iff it has no subgraph homeomorphic to K_5 or $K_{3,3}$.

Petersen's graph is not planar because it has a subgraph homeomorphic to $K_{3,3}$.

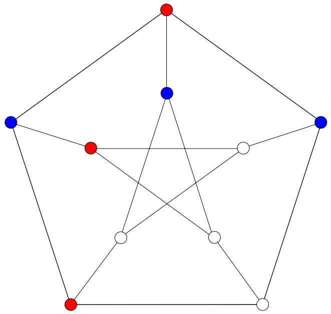


Classical results on planarity

Theorem (Kuratowski, 1930)

A graph is planar iff it has no subgraph homeomorphic to K_5 or $K_{3,3}$.

Petersen's graph is not planar because it has a subgraph homeomorphic to $K_{3,3}$.

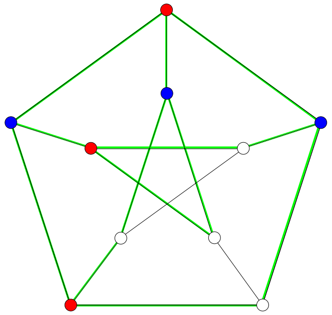


Classical results on planarity

Theorem (Kuratowski, 1930)

A graph is planar iff it has no subgraph homeomorphic to K_5 or $K_{3,3}$.

Petersen's graph is not planar because it has a subgraph homeomorphic to $K_{3,3}$.

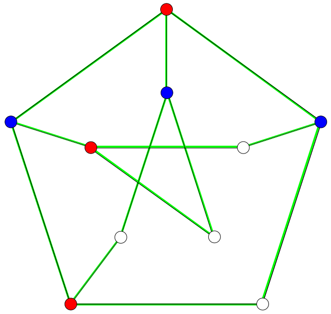


Classical results on planarity

Theorem (Kuratowski, 1930)

A graph is planar iff it has no subgraph homeomorphic to K_5 or $K_{3,3}$.

Petersen's graph is not planar because it has a subgraph homeomorphic to $K_{3,3}$.

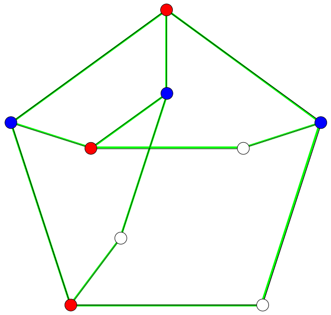


Classical results on planarity

Theorem (Kuratowski, 1930)

A graph is planar iff it has no subgraph homeomorphic to K_5 or $K_{3,3}$.

Petersen's graph is not planar because it has a subgraph homeomorphic to $K_{3,3}$.

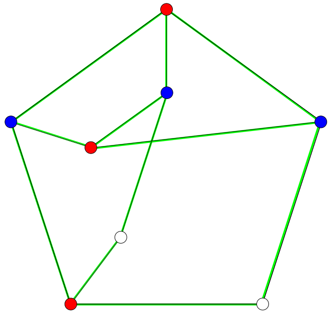


Classical results on planarity

Theorem (Kuratowski, 1930)

A graph is planar iff it has no subgraph homeomorphic to K_5 or $K_{3,3}$.

Petersen's graph is not planar because it has a subgraph homeomorphic to $K_{3,3}$.

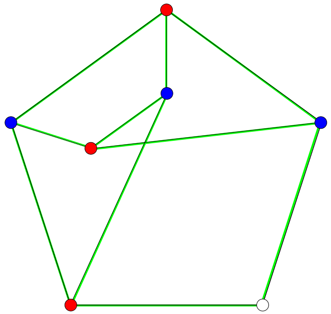


Classical results on planarity

Theorem (Kuratowski, 1930)

A graph is planar iff it has no subgraph homeomorphic to K_5 or $K_{3,3}$.

Petersen's graph is not planar because it has a subgraph homeomorphic to $K_{3,3}$.

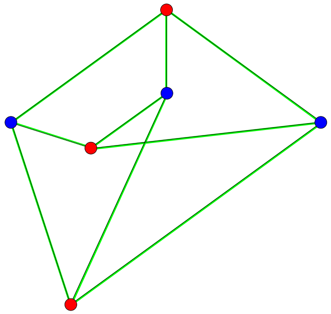


Classical results on planarity

Theorem (Kuratowski, 1930)

A graph is planar iff it has no subgraph homeomorphic to K_5 or $K_{3,3}$.

Petersen's graph is not planar because it has a subgraph homeomorphic to $K_{3,3}$.

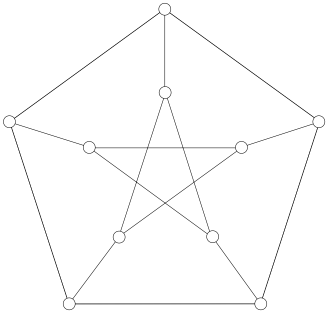


Classical results on planarity

Theorem (Wagner, 1937)

A graph is planar iff it contains neither K_5 nor $K_{3,3}$ as a minor.

Petersen's graph is not planar because it contains K_5 as a minor.

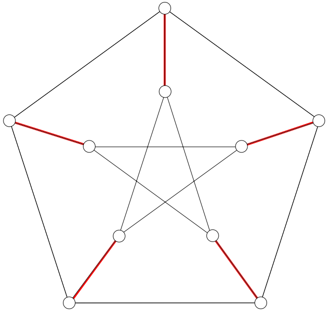


Classical results on planarity

Theorem (Wagner, 1937)

A graph is planar iff it contains neither K_5 nor $K_{3,3}$ as a minor.

Petersen's graph is not planar because it contains K_5 as a minor.

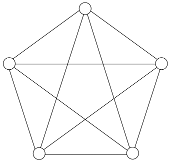


Classical results on planarity

Theorem (Wagner, 1937)

A graph is planar iff it contains neither K_5 nor $K_{3,3}$ as a minor.

Petersen's graph is not planar because it contains K_5 as a minor.



Planarity in the light of the Theory of Robertson and Seymour

Consider the following facts:

- Any graph is either planar or non-planar (obvious).

Planarity in the light of the Theory of Robertson and Seymour

Consider the following facts:

- Any graph is either planar or non-planar (obvious).
- All graphs are well quasi ordered by $\preceq$ (extremely hard to prove part of the Theory of R. & S.).

Planarity in the light of the Theory of Robertson and Seymour

Consider the following facts:

- Any graph is either planar or non-planar (obvious).
- All graphs are well quasi ordered by $\preceq$ (extremely hard to prove part of the Theory of R. & S.).
- The planar graphs are an ideal with respect to $\preceq$: if a graph is planar then every graph that is its minor is planar, too (obvious).

Planarity in the light of the Theory of Robertson and Seymour

Consider the following facts:

- Any graph is either planar or non-planar (obvious).
- All graphs are well quasi ordered by $\preceq$ (extremely hard to prove part of the Theory of R. & S.).
- The planar graphs are an ideal with respect to $\preceq$: if a graph is planar then every graph that is its minor is planar, too (obvious).
- Dually, the non-planar graphs are a filter with respect to $\preceq$: if a graph is not planar then every graph to which it is minor, is not planar either.

Planarity in the light of the Theory of Robertson and Seymour

Consider the following facts:

- Any graph is either planar or non-planar (obvious).
- All graphs are well quasi ordered by $\preceq$ (extremely hard to prove part of the Theory of R. & S.).
- The planar graphs are an ideal with respect to $\preceq$: if a graph is planar then every graph that is its minor is planar, too (obvious).
- Dually, the non-planar graphs are a filter with respect to $\preceq$: if a graph is not planar then every graph to which it is minor, is not planar either.
- The question whether a graph is planar is equivalent to the question whether it is an element of the ideal of planar graphs.

Planarity in the light of the Theory of R. & S.

- There exists a finite set of obstructions to planarity (follows trivially from the equivalent definition of WQO).

Planarity in the light of the Theory of R. & S.

- There exists a finite set of obstructions to planarity (follows trivially from the equivalent definition of WQO).
    - The dual statement is: the filter of all non-planar graphs is finitely generated (follows trivially from the definitions).

Planarity in the light of the Theory of R. & S.

- There exists a finite set of obstructions to planarity (follows trivially from the equivalent definition of WQO).
  - The dual statement is: the filter of all non-planar graphs is finitely generated (follows trivially from the definitions).
  - From the theorems of Kuratowski and Wagner we even know the obstruction set: $\{K_5, K_{3,3}\}$.

Planarity in the light of the Theory of R. & S.

- There exists a finite set of obstructions to planarity (follows trivially from the equivalent definition of WQO).
  - The dual statement is: the filter of all non-planar graphs is finitely generated (follows trivially from the definitions).
  - From the theorems of Kuratowski and Wagner we even know the obstruction set: $\{K_5, K_{3,3}\}$.
- In $O(n^3)$ we can answer whether a graph G is planar or not by checking whether $K_5 \preceq G$ or $K_{3,3} \preceq G$ (hard to prove part of the Theory of R. & S.; non-constructive result).

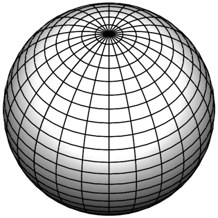
Planarity in the light of the Theory of R. & S.

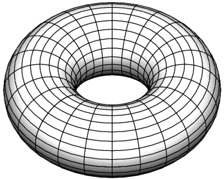
And so the Theory of Robertson and Seymour proves that $\textsc{Planarity} \in \mathcal{P}$.

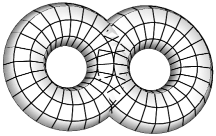
Indeed, a practical linear time algorithm for planarity testing has been known *since 1974* but the Theory of Robertson and Seymour is a tool that can be used to prove the membership in $\mathcal{P}$ of a huge number of problems.

Membership in $\mathcal{P}$ via the Theory of R. & S.
Definition of genus of a surface

Every compact closed orientable surface is topologically equivalent to a sphere with added ≥ 0 "handles". The number of the "handles" is its *genus*.


genus 0


genus 1


genus 2

Membership in $\mathcal{P}$ via the Theory of R. & S.
Graph Genus

- *The genus of graph G*, shortly gen(G), is the smallest genus of a surfaces that G can be embedded into. The planar graphs are precisely the graphs with genus 0.

Membership in $\mathcal{P}$ via the Theory of R. & S.

Graph Genus

- *The genus of graph G*, shortly $\text{gen}(G)$, is the smallest genus of a surfaces that G can be embedded into. The planar graphs are precisely the graphs with genus 0.
- To compute $\text{gen}(G)$ is an $\mathcal{NP}$-complete problem.

Membership in $\mathcal{P}$ via the Theory of R. & S.

Graph Genus

- *The genus of graph G*, shortly $\text{gen}(G)$, is the smallest genus of a surfaces that G can be embedded into. The planar graphs are precisely the graphs with genus 0.
- To compute $\text{gen}(G)$ is an $\mathcal{NP}$-complete problem.
- For every **fixed genus** there is a polynomial-time algorithm. That is an immediate consequence of the Theory of Robertson and Seymour and the fact that $G_1 \preceq G_2$ implies $\text{gen}(G_1) \leq \text{gen}(G_2)$.

Membership in $\mathcal{P}$ via the Theory of R. & S.

Graph Genus

- *The genus of graph G*, shortly $\text{gen}(G)$, is the smallest genus of a surfaces that G can be embedded into. The planar graphs are precisely the graphs with genus 0.
- To compute $\text{gen}(G)$ is an $\mathcal{NP}$-complete problem.
- For every **fixed genus** there is a polynomial-time algorithm. That is an immediate consequence of the Theory of Robertson and Seymour and the fact that $G_1 \preceq G_2$ implies $\text{gen}(G_1) \leq \text{gen}(G_2)$.
- The Theory of Robertson and Seymour says that for every genus there is a finite set of obstructions. For the torus (genus 1) they are at least *tens of thousands*.

Membership in $\mathcal{P}$ via the Theory of R. & S.
Linkless Embedding

The Linkless Embedding problem is, can we embed a given graph G in three-dimensional Euclidean space so that no two vertex-disjoint cycles are *topologically linked*. Informally, topologically linked closed curves are ones that are linked like chain links. It is known that K_6 has no linkless embedding (Conway, Gordon 1983).

Membership in $\mathcal{P}$ via the Theory of R. & S.
Linkless Embedding

The Linkless Embedding problem is, can we embed a given graph G in three-dimensional Euclidean space so that no two vertex-disjoint cycles are *topologically linked*. Informally, topologically linked closed curves are ones that are linked like chain links. It is known that K_6 has no linkless embedding (Conway, Gordon 1983).

From basic considerations it is not obvious the problem is algorithmically solvable. However, the Theory of Robertson and Seymour implies the problem is in fact solvable in polynomial time. It has been proved by Robertson and Seymour that the obstruction set is *the Petersen family*.

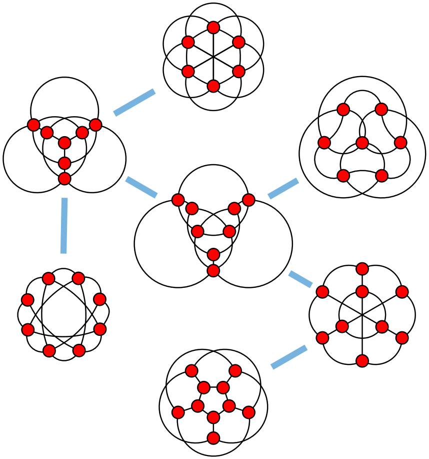
Membership in $\mathcal{P}$ via the Theory of R. & S.
Linkless Embedding

The Linkless Embedding problem is, can we embed a given graph G in three-dimensional Euclidean space so that no two vertex-disjoint cycles are *topologically linked*. Informally, topologically linked closed curves are ones that are linked like chain links. It is known that K_6 has no linkless embedding (Conway, Gordon 1983).

From basic considerations it is not obvious the problem is algorithmically solvable. However, the Theory of Robertson and Seymour implies the problem is in fact solvable in polynomial time. It has been proved by Robertson and Seymour that the obstruction set is *the Petersen family*.

Finding an efficient algorithm is an open problem.

The Petersen family is the obstruction set of LINKLESS EMBEDDING



Membership in $\mathcal{P}$ via the Theory of R. & S.

VERTEX COVER for fixed k

If G is covered by $\leq k$ vertices and $H \preceq G$ then H is covered by $\leq k$ vertices, too.

Membership in $\mathcal{P}$ via the Theory of R. & S.
Vertex Cover for fixed k

If G is covered by $\leq k$ vertices and $H \preceq G$ then H is covered by $\leq k$ vertices, too.

Therefore Vertex Cover is solvable in $O(n^3)$ for all fixed k.

Membership in $\mathcal{P}$ via the Theory of R. & S.
Vertex Cover for fixed k

If G is covered by $\leq k$ vertices and $H \preceq G$ then H is covered by $\leq k$ vertices, too.

Therefore Vertex Cover is solvable in $O(n^3)$ for all fixed k.

The obstruction set for every k value is **different**. There are 188 *connected obstructions* for $k = 6$.

Membership in $\mathcal{P}$ via the Theory of R. & S.
VERTEX COVER for fixed k

If G is covered by $\leq k$ vertices and $H \preceq G$ then H is covered by $\leq k$ vertices, too.

Therefore VERTEX COVER is solvable in $O(n^3)$ for all fixed k.

The obstruction set for every k value is **different**. There are 188 *connected obstructions* for $k = 6$.

There is a huge difference between the order of growth n^3 (the algorithm of Fellows and Langston) and the order of growth n^k of the brute force approach. Nonconstructivity aside, the complexity function $f(k)n^3$ is qualitatively better than n^k.

Nonconstructivity of the results of R. & S.

- The theory **does not** give a clue how to compute the obstructions. It mereley proves they exist. The said toroidal embedding obstructions were computed by a specially designed for that purpose algorithm.

Nonconstructivity of the results of R. & S.

- The theory **does not** give a clue how to compute the obstructions. It mereley proves they exist. The said toroidal embedding obstructions were computed by a specially designed for that purpose algorithm.
- Moreover, the problem of computing the obstruction set in general is *algorithmically unsolvable*.

Theorem (Fellows, Langston 1989)

There is no algorithm to compute, from a finite description of a minor-closed family F of graphs as represented by a Turing machine that accepts precisely the graphs in F, the set of obstructions for F.

Limitations of the Theory of R. & S.

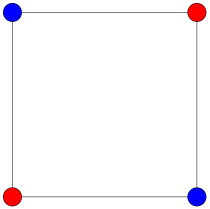
Embeddability in surfaces of genus k, linkess embeddings in 3D, and having k-vertex cover are all properties that are preserved in minors. Not all graph properties are preserved in that way, though:

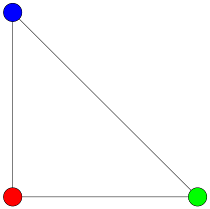
- k vertex colourability *is not preserved* on minors (though it is preserved on subgraphs).

Limitations of the Theory of R. & S.

Embeddability in surfaces of genus k, linkess embeddings in 3D, and having k-vertex cover are all properties that are preserved in minors. Not all graph properties are preserved in that way, though:

- k vertex colourability *is not preserved* on minors (though it is preserved on subgraphs).

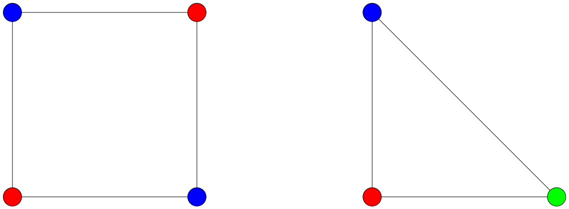




Limitations of the Theory of R. & S.

Embeddability in surfaces of genus k, linkess embeddings in 3D, and having k-vertex cover are all properties that are preserved in minors. Not all graph properties are preserved in that way, though:

- k vertex colourability *is not preserved* on minors (though it is preserved on subgraphs).



- Other properties that are not preserved are Hamiltonicity, Eulericity, domination by at most k vertices.

A $O(2^k n)$ algorithm for VERTEX COVER
The authors are Downey and Fellows

Given a graph G and number k construct a binary tree of height $\leq k$ *in the following way*:

A $O(2^k n)$ algorithm for Vertex Cover
The authors are Downey and Fellows

Given a graph G and number k construct a binary tree of height $\leq k$ *in the following way*:

1. Construct the root with label $\langle \emptyset, G \rangle$.

A $O(2^k n)$ algorithm for VERTEX COVER

The authors are Downey and Fellows

Given a graph G and number k construct a binary tree of height $\leq k$ *in the following way*:

1. Construct the root with label $\langle \emptyset, G \rangle$.
2. Choose an arbitrary $(u, v) \in E(G)$.

A $O(2^k n)$ algorithm for VERTEX COVER

The authors are Downey and Fellows

Given a graph G and number k construct a binary tree of height $\leq k$ *in the following way*:

1. Construct the root with label $\langle \emptyset, G \rangle$.
2. Choose an arbitrary $(u, v) \in E(G)$.
3. Construct two children of the root with labels $\langle \{u\}, G - u \rangle$ and $\langle \{v\}, G - v \rangle$. They are associated with $G - u$ and $G - v$, respectively.

A $O(2^k n)$ algorithm for VERTEX COVER

The authors are Downey and Fellows

Given a graph G and number k construct a binary tree of height $\leq k$ *in the following way*:

1. Construct the root with label $\langle \emptyset, G \rangle$.
2. Choose an arbitrary $(u, v) \in E(G)$.
3. Construct two children of the root with labels $\langle \{u\}, G - u \rangle$ and $\langle \{v\}, G - v \rangle$. They are associated with $G - u$ and $G - v$, respectively.
4. In each of $G - u$ and $G - v$ choose an arbitrary edge and construct analogously four tree vertices of height 2.

A $O(2^k n)$ algorithm for VERTEX COVER

The authors are Downey and Fellows

Given a graph G and number k construct a binary tree of height $\leq k$ *in the following way*:

1. Construct the root with label $\langle \emptyset, G \rangle$.
2. Choose an arbitrary $(u, v) \in E(G)$.
3. Construct two children of the root with labels $\langle \{u\}, G - u \rangle$ and $\langle \{v\}, G - v \rangle$. They are associated with $G - u$ and $G - v$, respectively.
4. In each of $G - u$ and $G - v$ choose an arbitrary edge and construct analogously four tree vertices of height 2.
5. ...

A $O(2^k n)$ algorithm for VERTEX COVER

The authors are Downey and Fellows

Given a graph G and number k construct a binary tree of height $\leq k$ *in the following way*:

1. Construct the root with label $\langle \emptyset, G \rangle$.
2. Choose an arbitrary $(u, v) \in E(G)$.
3. Construct two children of the root with labels $\langle \{u\}, G - u \rangle$ and $\langle \{v\}, G - v \rangle$. They are associated with $G - u$ and $G - v$, respectively.
4. In each of $G - u$ and $G - v$ choose an arbitrary edge and construct analogously four tree vertices of height 2.
5. ...
6. If at least one tree vertex of height $\leq k$ is such that the graph associated with it has no edges, return YES.

A $O(2^k n)$ algorithm for VERTEX COVER

The authors are Downey and Fellows

Given a graph G and number k construct a binary tree of height $\leq k$ *in the following way*:

1. Construct the root with label $\langle \emptyset, G \rangle$.
2. Choose an arbitrary $(u, v) \in E(G)$.
3. Construct two children of the root with labels $\langle \{u\}, G - u \rangle$ and $\langle \{v\}, G - v \rangle$. They are associated with $G - u$ and $G - v$, respectively.
4. In each of $G - u$ and $G - v$ choose an arbitrary edge and construct analogously four tree vertices of height 2.
5. ...
6. If at least one tree vertex of height $\leq k$ is such that the graph associated with it has no edges, return YES.
7. Otherwise, return NO.

Analysis of the algorithm of Downey and Fellows

correctness In every tree vertex at height ℓ the label contains a set of graph vertices of size ℓ; *viz.* the vertices that have been deleted. Note that the original G has vertex cover of size t iff at least one of the graphs from the tree labels at height ℓ has vertex cover of size $t - \ell$. The vertex cover of the original graph is the union of the vertex cover of the reduced graph (the graph of the label) and set of the vertices in that label.

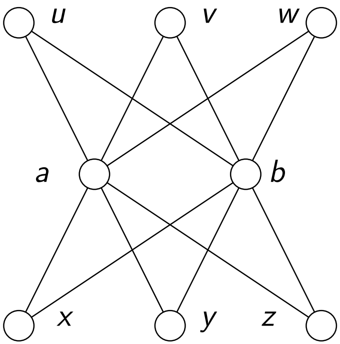
Analysis of the algorithm of Downey and Fellows

correctness In every tree vertex at height ℓ the label contains a set of graph vertices of size ℓ; *viz.* the vertices that have been deleted. Note that the original G has vertex cover of size t iff at least one of the graphs from the tree labels at height ℓ has vertex cover of size $t - \ell$. The vertex cover of the original graph is the union of the vertex cover of the reduced graph (the graph of the label) and set of the vertices in that label.

time complexity (roughly) $\Theta(2^k n)$ in the worst case.

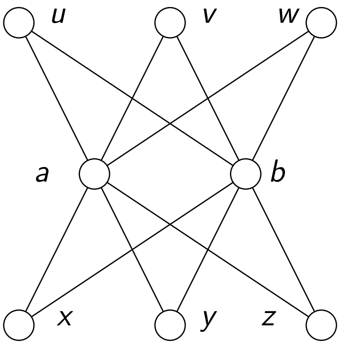
The algorithm of Downey and Fellows at work, $k = 2$

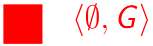
The initial graph



The algorithm of Downey and Fellows at work, $k = 2$

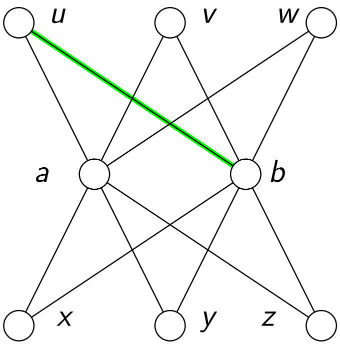
Construct the root of the tree





The algorithm of Downey and Fellows at work, $k = 2$

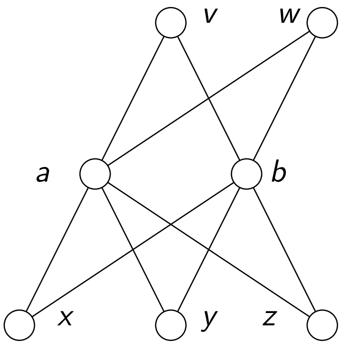
Choose arbitrarily (u, b). Any vertex cover of G must cover (u, b).

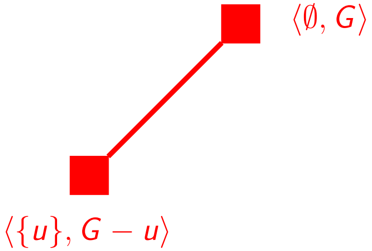


🟥 $\langle \emptyset, G \rangle$

The algorithm of Downey and Fellows at work, $k = 2$

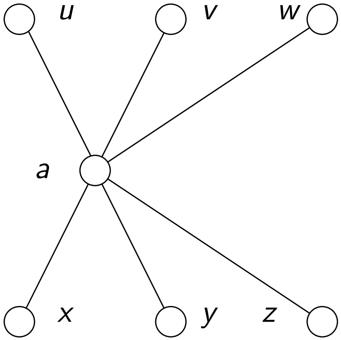
If u is in the cover, we have to cover $G - u$ with $k - 1 = 1$ vertex

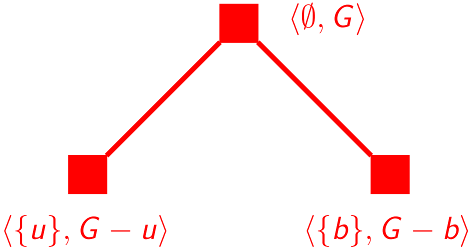




The algorithm of Downey and Fellows at work, $k = 2$

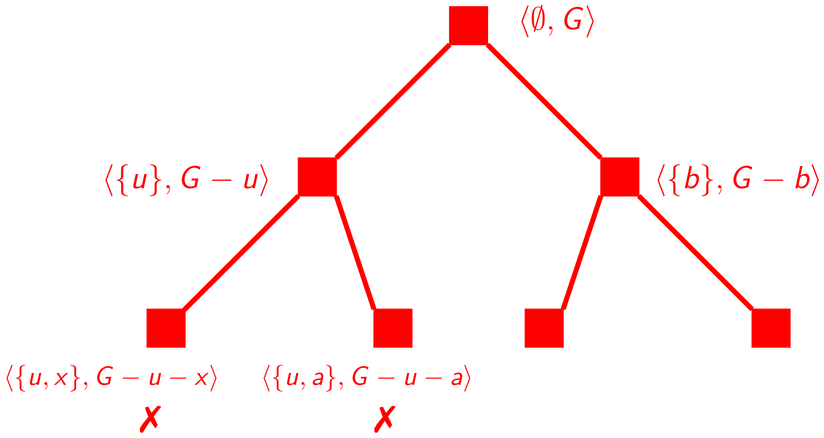
If b is in the cover, we have to cover $G - b$ with $k - 1 = 1$ vertex




$\langle \emptyset, G \rangle$
$\langle \{u\}, G - u \rangle$
$\langle \{b\}, G - b \rangle$

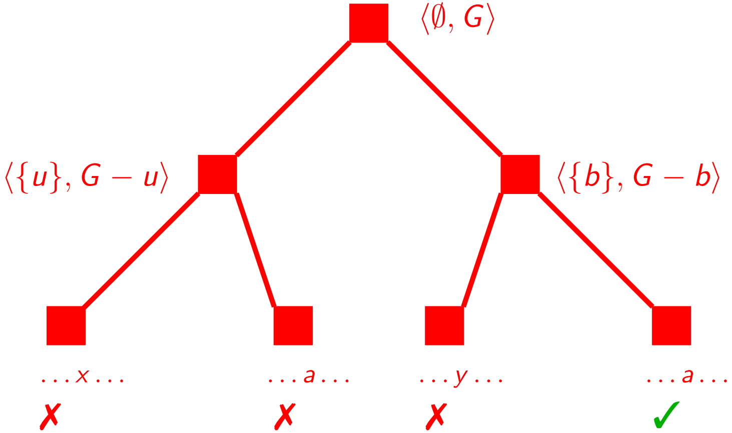
The algorithm of Downey and Fellows at work, $k = 2$

There is no vertex cover of size 2 that contains u


$\langle \emptyset, G \rangle$
$\langle \{u\}, G - u \rangle$
$\langle \{b\}, G - b \rangle$
$\langle \{u, x\}, G - u - x \rangle$
$\langle \{u, a\}, G - u - a \rangle$
✗
✗

The algorithm of Downey and Fellows at work, $k = 2$

However, there is vertex cover of size 2 that contains b


$\langle \emptyset, G \rangle$
$\langle \{u\}, G - u \rangle$
$\langle \{b\}, G - b \rangle$
$\dots x \dots$
$\dots a \dots$
$\dots y \dots$
$\dots a \dots$
✗
✗
✗
✓

Further analysis of the algorithm of Downey and Fellows

Because of the 2^k factor in the function of the order of growth the algorithm is at worst exponential. However, for small k it is *much better* than the brute force approach.

Further analysis of the algorithm of Downey and Fellows

Because of the 2^k factor in the function of the order of growth the algorithm is at worst exponential. However, for small k it is *much better* than the brute force approach.

Which order of growth is better: n^k (the brute force) or $2^k n$ (Downey and Fellows)? Suppose $k = 20$ amd $n = 100\,000$. A simple calculation

$$100\,000^{20} = 10^{25} \qquad\qquad 2^{20} \times 100\,000 \approx 10^{11}$$

shows the difference is 14 decimal orders of magnitude...

Further analysis of the algorithm of Downey and Fellows

The algorithm has to be exponential at worst. A subexponential at worst algorithm for VERTEX COVER would be something the newspapers would write about.

Further analysis of the algorithm of Downey and Fellows

The algorithm has to be exponential at worst. A subexponential at worst algorithm for Vertex Cover would be something the newspapers would write about.

From practical point of view, for small k the algorithm works reasonably well even for large inputs.

Further analysis of the algorithm of Downey and Fellows

The algorithm has to be exponential at worst. A subexponential at worst algorithm for Vertex Cover would be something the newspapers would write about.

From practical point of view, for small k the algorithm works reasonably well even for large inputs.

From theoretical point of view, for fixed k the algorithm is polynomial of degree that is *independent of* k, and the problem is tractable. Of course, n^k is polynomial function, too, when k is fixed but $2^k n$ is considerably better.

Algorithm of Sam Buss, 1989

A trivial observation: if the graph has vertex u of degree $> k$ then u is in *every* vertex cover of size $\leq k$.

Algorithm of Sam Buss, 1989

A trivial observation: if the graph has vertex u of degree $> k$ then u is in *every* vertex cover of size $\leq k$.

1. We are given a graph G and number k. Suppose $W \subseteq V(G)$ are the vertices of degree $> k$.

Algorithm of Sam Buss, 1989

A trivial observation: if the graph has vertex u of degree $> k$ then u is in *every* vertex cover of size $\leq k$.

1. We are given a graph G and number k. Suppose $W \subseteq V(G)$ are the vertices of degree $> k$.
2. Let $p = |W|$. If $p > k$ then return NO. Else, let $\ell = k - p$.

Algorithm of Sam Buss, 1989

A trivial observation: if the graph has vertex u of degree $> k$ then u is in *every* vertex cover of size $\leq k$.

1. We are given a graph G and number k. Suppose $W \subseteq V(G)$ are the vertices of degree $> k$.
2. Let $p = |W|$. If $p > k$ then return NO. Else, let $\ell = k - p$.
3. $H \leftarrow G - W$.

Algorithm of Sam Buss, 1989

A trivial observation: if the graph has vertex u of degree $> k$ then u is in *every* vertex cover of size $\leq k$.

1. We are given a graph G and number k. Suppose $W \subseteq V(G)$ are the vertices of degree $> k$.
2. Let $p = |W|$. If $p > k$ then return NO. Else, let $\ell = k - p$.
3. $H \leftarrow G - W$.
4. If $|E(H)| > k\ell$ then return NO.

Algorithm of Sam Buss, 1989

A trivial observation: if the graph has vertex u of degree $> k$ then u is in *every* vertex cover of size $\leq k$.

1. We are given a graph G and number k. Suppose $W \subseteq V(G)$ are the vertices of degree $> k$.
2. Let $p = |W|$. If $p > k$ then return NO. Else, let $\ell = k - p$.
3. $H \leftarrow G - W$.
4. If $|E(H)| > k\ell$ then return NO.
5. If H has no ℓ-vertex cover then return NO.

Algorithm of Sam Buss, 1989

A trivial observation: if the graph has vertex u of degree $> k$ then u is in *every* vertex cover of size $\leq k$.

1. We are given a graph G and number k. Suppose $W \subseteq V(G)$ are the vertices of degree $> k$.
2. Let $p = |W|$. If $p > k$ then return NO. Else, let $\ell = k - p$.
3. $H \leftarrow G - W$.
4. If $|E(H)| > k\ell$ then return NO.
5. If H has no ℓ-vertex cover then return NO.
6. Else, the union of any ℓ-vertex cover of H and W is a k-vertex cover for G.

Analysis of the algorithm of Buss

The correctness is obvious. Time complexity analysis:

- Steps 1, . . . , 4 are linear time.
- Step 5 is the search of minimum vertex cover of the reduced graph H. ℓ is at most k therefore the graph H at step 5 H has at most k^2 edges and $2k^2$ vertices. A minimum vertex cover for a graph with $2k^2$ vertices can be computed in $O\left(2^{2k^2}\right)$ using brute force.
- Altogether, the algorithm works in $O\left(n + m + 2^{2k^2}\right)$.

The crucial advantage of the algorithm of Buss is that it runs the superpolynomial procedure on the reduced graph H whose size is a function of k and **not of** n.

Improvements of the algorithm of Buss

A minimum vertex cover of the reduced graph can be computed by more sophisticated ways, *e.g.* using the algorithm of Downey and Fellows with the bounded search tree. As pointed out in (Downey and Fellows, 1999, pp. 35–36), a simple improvement of idea of the bounded search tree yeilds a $O(\sqrt[4]{5}^k n)$ algorithm.

By combining the algorithm of Buss with the said improvement, Downey and Fellows propose an algorithm for VERTEX COVER that runs in $O(n + k^2 2^k)$.

At the moment, the fastest algorithm for VERTEX COVER when k is small is the algorithm of *Chen, Kanj, and Xia*, running in $O(1.2738^k n^{O(1)})$.

Complexity as a function of two variables

A two variable complexity function is significantly more informative than the classical complexity function of a single variable – the input size.
In Parameterized Complexity one variable is the input size and the other one is called *the parameter*. The parameter is aimed at capturing the aspect of the problem that makes it intractable, to the extent that the input size participates in the overall complexity expression in a “bening” way, say as a linear or quadratic factor.

Parameterized Problems: The Formal Definition

Definition

A parameterized decision problem is $L \subseteq \Sigma^* \times \Sigma^*$ where Σ is the input alphabet. If $(\sigma, k) \in L$, the parameter is k.

Typically the parameter is a natural number, therefore we can say $L \subseteq \Sigma^* \times \mathbb{N}$.

The Theory of Parameterized Complexity

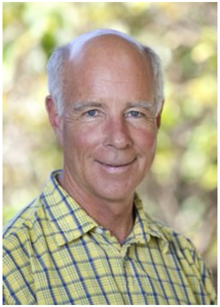
A branch of Computational Complexity that focuses on investigating the inherent difficulty of intractable problems with respect to the input size and a parameter, the complexity being a function of both.

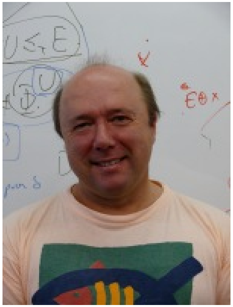
The Theory of Parameterized Complexity

A branch of Computational Complexity that focuses on investigating the inherent difficulty of intractable problems with respect to the input size and a parameter, the complexity being a function of both.

It was created in the first half of the 90's by *Mike Fellows* and *Rod Downey*.

Michael Ralph Fellows and Rodney Graham Downey


Mike Fellows


Rod Downey

Vertex Cover revisited
This time as a parameterized *p*-Vertex-Cover

Every parameterized decision problem is defined by *three* components: generic instance, parameter, and question with Yes/No answer.

VERTEX COVER revisited

This time as a parameterized *p*-VERTEX-COVER

Every parameterized decision problem is defined by *three* components: generic instance, parameter, and question with YES/NO answer.
Definition of *p*-VERTEX COVER:

VERTEX COVER revisited

This time as a parameterized *p*-VERTEX-COVER

Every parameterized decision problem is defined by *three* components: generic instance, parameter, and question with YES/NO answer.

Definition of *p*-VERTEX COVER:

- An ordered pair of an undirected graph $G = (V, E)$ and natural number k.

VERTEX COVER revisited

This time as a parameterized *p*-VERTEX-COVER

Every parameterized decision problem is defined by *three* components: generic instance, parameter, and question with YES/NO answer.

Definition of *p*-VERTEX COVER:

- An ordered pair of an undirected graph $G = (V, E)$ and natural number k.
- Parameter k

VERTEX COVER revisited

This time as a parameterized *p*-VERTEX-COVER

Every parameterized decision problem is defined by *three* components: generic instance, parameter, and question with YES/NO answer.

Definition of *p*-VERTEX COVER:

- An ordered pair of an undirected graph $G = (V, E)$ and natural number k.
- Parameter k
- Does G have a vertex cover of size $\leq k$?

VERTEX COVER revisited

This time as a parameterized p-VERTEX-COVER

Every parameterized decision problem is defined by *three* components: generic instance, parameter, and question with YES/NO answer.
Definition of p-VERTEX COVER:

- An ordered pair of an undirected graph $G = (V, E)$ and natural number k.
- Parameter k
- Does G have a vertex cover of size $\leq k$?

The parameter is *not necessarily* the number from the generic instance. Such a number may not exist—consider HAMILTONIAN CYCLE—while every classical decision problem can be parameterized. For instance, HAMILTONIAN CYCLE can be parameterized by the diameter of the graph (just an idea).

Complexity class $\mathcal{FPT}$

Definition

A parameterized problem L is **fixed parameter tractable** (FPT) if there is an algorithm that computes whether $\langle x, y \rangle \in L$ in time $O(f(y) \times n^{O(1)})$ where $f(y)$ is a computable function that *does not depend* on x.

Complexity class $\mathcal{FPT}$

Definition

A parameterized problem L is **fixed parameter tractable** (FPT) if there is an algorithm that computes whether $\langle x, y \rangle \in L$ in time $O(f(y) \times n^{O(1)})$ where $f(y)$ is a computable function that *does not depend* on x.

As we saw, p-$\textsc{Vertex Cover} \in \mathcal{FPT}$.

Complexity class $\mathcal{FPT}$

Definition

A parameterized problem L is **fixed parameter tractable** (FPT) if there is an algorithm that computes whether $\langle x, y \rangle \in L$ in time $O(f(y) \times n^{O(1)})$ where $f(y)$ is a computable function that *does not depend* on x.

As we saw, p-$\textsc{Vertex Cover} \in \mathcal{FPT}$.

The complexity expression can as well be $O(g(y) + n^{O(1)})$: both formulations are equivalent.

Not all parameterized problems are in $\mathcal{FPT}$

It has been proved that p-DOMINATING SET $\notin \mathcal{FPT}$, under the assumption that every $\mathcal{NP}$-complete problem necessitates deterministic exponential time.

Not all parameterized problems are in $\mathcal{FPT}$

It has been proved that *p*-Dominating Set $\notin \mathcal{FPT}$, under the assumption that every $\mathcal{NP}$-complete problem necessitates deterministic exponential time.

Definition of *p*-Dominating Set:

Not all parameterized problems are in $\mathcal{FPT}$

It has been proved that p-Dominating Set $\notin \mathcal{FPT}$, under the assumption that every $\mathcal{NP}$-complete problem necessitates deterministic exponential time.

Definition of p-Dominating Set:

- An ordered par of unndirected graph $G = (V, E)$ and a natural number k.

Not all parameterized problems are in $\mathcal{FPT}$

It has been proved that p-Dominating Set $\notin \mathcal{FPT}$, under the assumption that every $\mathcal{NP}$-complete problem necessitates deterministic exponential time.

Definition of p-Dominating Set:

- An ordered par of unndirected graph $G = (V, E)$ and a natural number k.
- Parameter k.

Not all parameterized problems are in $\mathcal{FPT}$

It has been proved that p-DOMINATING SET $\notin \mathcal{FPT}$, under the assumption that every $\mathcal{NP}$-complete problem necessitates deterministic exponential time.

Definition of p-DOMINATING SET:

- An ordered par of unndirected graph $G = (V, E)$ and a natural number k.
- Parameter k.
- Does G dominating set of size $\leq k$?

Not all parameterized problems are in $\mathcal{FPT}$

It has been proved that p-Dominating Set $\notin \mathcal{FPT}$, under the assumption that every $\mathcal{NP}$-complete problem necessitates deterministic exponential time.

Definition of p-Dominating Set:

- An ordered par of unndirected graph $G = (V, E)$ and a natural number k.
- Parameter k.
- Does G dominating set of size $\leq k$?

Dominating set of G is every $U \subseteq V$ such that

$$\forall x \in V : x \in U \vee \exists y \in U((x, y) \in E)$$

Klam: the maximum value of the parameter for which the problem is tractable

For an FPT problem, “the klam value” is the maximum value of the parameter for which it is realistic to solve the problem, for input sizes that arise in practice (whatever that means...).

Klam: the maximum value of the parameter for which the problem is tractable

For an FPT problem, “the klam value” is the maximum value of the parameter for which it is realistic to solve the problem, for input sizes that arise in practice (whatever that means...).

Downey and Fellows propose the klam value to be computed as follows: the maximum k for which $f(k)$ does not exceed some so called universal constant. The proposed universal constant is 10^{20}.

klam values for p-Vertex Cover

The complexities of the current fastest algorithms for a number of FPT problems can be seen at *fpt.wikidot.com/fpt-races*. The $f(k)$ function of p-Vertex Cover is 1.2738^k. The solution to the eqation $1.2738^k = 10^{20}$ is approximately $k = 190$ and that is the current klam value for that problem.

klam values for p-Vertex Cover

The complexities of the current fastest algorithms for a number of FPT problems can be seen at *fpt.wikidot.com/fpt-races*. The $f(k)$ function of p-Vertex Cover is 1.2738^k. The solution to the eqation $1.2738^k = 10^{20}$ is approximately $k = 190$ and that is the current klam value for that problem.
For the same problem with the restriction of maximum degree 3 in the graph, the function si 1.1616^k and the klam value is 307.

klam values for *p*-Vertex Cover

The complexities of the current fastest algorithms for a number of FPT problems can be seen at *fpt.wikidot.com/fpt-races*. The $f(k)$ function of *p*-Vertex Cover is 1.2738^k. The solution to the eqation $1.2738^k = 10^{20}$ is approximately $k = 190$ and that is the current klam value for that problem.
For the same problem with the restriction of maximum degree 3 in the graph, the function si 1.1616^k and the klam value is 307.
For 2^k (the algorithm for Vertex Cover of Downey and Fellows) the klam value is 66.

p-Vertex Cover is the champion of efficient parameterization

Clearly, *p*-Vertex Cover stands out among all *parameterized problems*. For some reason, Vertex Cover is much more ameanable to efficient parameterization than the remaining FPT problems.

Compare the klam value of Vertex Cover with the one of Clique Cover. Since the $f(k)$ function of the latter is 2^{2^k}, its klam value is about 6.

The choice of the parameter is not unique

p-VERTEX COVER seems to be the most natural way to parameterize VERTEX COVER. However, the latter can be parameterized in a number of ways. The results are *different* parameterized problems that have different properties.

Different ways to parameterize VERTEX COVER

As parameterized problems, they are **different problems**

Suppose we a are given a graph with n vertices and a number k.

Different ways to parameterize VERTEX COVER

As parameterized problems, they are **different problems**

Suppose we a are given a graph with n vertices and a number k.

- We can choose the parameter to be $n - k$ and ask whether the graph has vertex cover of size $n - k$. That is in fact the INDEPENDENT SET problem. It is known (Downey and Fellows, 1999) that INDEPENDENT SET is not in FPT (under certain widely believed assumptions).

Different ways to parameterize Vertex Cover

As parameterized problems, they are **different problems**

Suppose we a are given a graph with n vertices and a number k.

- We can choose the parameter to be $n - k$ and ask whether the graph has vertex cover of size $n - k$. That is in fact the Independent Set problem. It is known (Downey and Fellows, 1999) that Independent Set is not in FPT (under certain widely believed assumptions).
- Consider planar graphs. Since every planar graph is 4-colourable it has vertex cover of size $\leq \left\lfloor \frac{3n}{4} \right\rfloor$: at least $\left\lceil \frac{n}{4} \right\rceil$ vertices *are not* in the cover. We can parameterize by $\left\lfloor \frac{3n}{4} \right\rfloor - k$. Unfortunately, it is not known whether this problems is in FPT.

Different ways to parameterize Vertex Cover

As parameterized problems, they are **different problems**

- We can parameterize by the treewidth of the graph. It is *well-known* that if the treewidth is $\leq t$ Vertex Cover can be solved in time $O(2^t n)$ regardless of the vertex cover size k that we are interested in. Consequently, choosing the treewidth as a parameter yield a another parameterized problem.

Techniques for constructing FPT algorithms

Elementary techniques:

- Using bounded search trees, *e.g.* the algorithm of Downey and Fellows for VERTEX COVER.
- Reducing the problem to a kernel, *e.g.* the algorithm of Buss for VERTEX COVER.

Techniques for constructing FPT algorithms

Elementary techniques:

- Using bounded search trees, *e.g.* the algorithm of Downey and Fellows for VERTEX COVER.
- Reducing the problem to a kernel, *e.g.* the algorithm of Buss for VERTEX COVER.

Sophisticated techniques:

- Using bounded treewidth. It is possible to construct efficient divide-and-conquer algorithms or dynamic programming algorithms for graphs of bounded treewidth, even for problems that are intractable on general graphs, provided the tree decomposition is known.
- In case a graph problem is expressible in Monadic Second Order Logic (MSOL), it is solvable in linear time on graphs of fixed treewidth: *Courcelle's theorem*.

FPT reductions are more elaborate than Karp reductions

Definition (Definition 2.1, Flum, Grohe 2006)

Let $\Pi = \langle Q, \kappa \rangle$ and $\Pi' = \langle Q', \kappa' \rangle$ be parameterized problems over the alphabet Σ. **An FPT reduction from Π to Π'** is a function $\psi : \Sigma^* \rightarrow \Sigma^*$ such that:

1. $\forall x \in \Sigma^* (x \in Q \leftrightarrow \psi(x) \in Q')$.
2. ψ is computable by an FPT algorithm with respect to κ. That means, $\psi(x)$ is computable in time $f(\kappa(x)) \cdot p(|x|)$ for some computable function f and some polynomial p.
3. There exists a computable function $g : \mathbb{N} \rightarrow \mathbb{N}$, such that $\forall x \in \Sigma^* : \kappa'(\psi(x)) \leq g(\kappa(x))$.

$\mathcal{FPT}$ is the easy parameterized class

$\mathcal{FPT}$ is the analogue of $\mathcal{P}$. The FPT reductions in it are analogous to the polynomial reductions in $\mathcal{P}$. Just as a polynomial reduction from a polynomial-time solvable problem yields a polynomial-time solvable problem, FPT reductions preserve membership in FPT. Condition 3 is crucial to that end.

Lemma (Lemma 2.2, Flum, Grohe 2006)

The complexity class $\mathcal{FPT}$ is closed with respect to FPT reductions. That is, if Π is reduced to Π' via FPT reduction and $\Pi' \in \mathcal{FPT}$, then $\Pi \in \mathcal{FPT}$.

Proof of the Lemma

Assume an FPT reduction $x \to x'$ from $\langle Q, \kappa \rangle$ to $\langle Q', \kappa' \rangle$ computable in time $h(k)q(|x|)$ with $k' \leq g(k)$ where $k = \kappa(x)$, $k' = \kappa(x')$, g and h are computable functions, and q is a polynomial. Then $|x'| \leq h(k)q(|x|)$.
Let A be a an algorithm deciding $\langle Q', \kappa' \rangle$ in time $f'(k')p'(|x'|)$.
WLOG, f' is not decreasing.

Then $\boxed{x \overset{?}{\in} Q}$ is decided by first computing the corresponding x' and then deciding $\boxed{x' \overset{?}{\in} Q'}$. That takes time at most

$$h(k)q(|x|) + f'(k')p'(|x'|) \leq h(k)q(|x|) + f'(g(k))p'(h(k)q(|x|))$$

But $p'(h(k)q(|x|)) \leq p'(h(k))p'(q(|x|))$. **QED**

A Karp reduction that is not an FPT reduction

Consider the standard Karp reduction from Vertex Cover to Independent Set:

$$(G, k) \to (G, n - k)$$

That is a Karp reduction but not an FPT reduction. Note that the $n - k$ is not a function of k only. So, condition 3 of the definition of FPT reduction is violated.
Indeed, it seems Independent Set is not in FPT while Vertex Cover is in FPT.

A Karp reduction that is an FPT reduction

On the other hand, the standard Karp reduction from CLIQUE to INDEPENDENT SET:

$$(G, k) \to (\overline{G}, k)$$

is an FPT reduction.

- Parameterized Complexity Theory provides powerful tools for dealing with intractability, in case the instances have small parameter.
- Often initial nonconstrucive results or hopelessly useless from practical point of view algorithms are followed by efficient in the practical sense algorithms. That confirms the algorithmic folklore: the natural problems are either completely intractable, or efficiently solvable.
- The results are orthogonal to the results of the classical Computational Complexity Theory or the Approximation Algorithms Theory.
- A multitude of problems turn out to be intractable even for a fixed parameter. A powerful toolset for proving parameterized intractability has been developed.

The Parameterized Complexity Theory is quite new and until recently, relatively unkown. The 1998 ACM classification does not mention Parameterized Complexity, while 2012 ACM classification has the following item:

> *Theory of computation $\rightarrow$ Design and analysis of algorithms $\rightarrow$ Parameterized complexity and exact algorithms*

Nowadays it is typical when discussing an $\mathcal{NP}$-hard problem to mention whether it is FPT or not along with the approximability results.

The End

Bibliography I

[BT06] Andrej Bogdanov and Luca Trevisan.
Average-case complexity.
Foundations and Trends in Theoretical Computer Science, 2(1):1–106, October 2006.

[CG83] J. H. Conway and C. McA. Gordon.
Knots and links in spatial graphs.
Journal of Graph Theory, 7:445–453, 1983.

[CKX06] Jianer Chen, Iyad A. Kanj, and Ge Xia.
Improved parameterized upper bounds for vertex cover.
In Rastislav Kralovic and Pawel Urzyczyn, editors, *MFCS*, volume 4162 of *Lecture Notes in Computer Science*, pages 238–249. Springer, 2006.

Bibliography II

[Cou90] Bruno Courcelle.
The monadic second-order logic of graphs. I. Recognizable sets of finite graphs.
Information and Computation, 85(1):12–75, 1990.

[DF92] R. G. Downey and M. R. Fellows.
Fixed-parameter tractability and completeness.
Congressus Numerantium, 87:161–187, 1992.

[DF99] Rodney G. Downey and Michael R. Fellows.
Parameterized Complexity.
Springer-Verlag, 1999.
530 pp.

Bibliography III

[Die12] Reinhard Diestel.
Graph Theory, 4th Edition, volume 173 of *Graduate texts in mathematics*.
Springer, 2012.
Freely available at `http://diestel-graph-theory.com/`.

[DX02] Michael J. Dinneen and Liu Xiong.
Minor-order obstructions for the graphs of vertex cover six.
J. Graph Theory, 41(3):163–178, November 2002.

[FG06] Jörg Flum and Martin Grohe.
Parameterized Complexity Theory.
Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

Bibliography IV

[FL87] Michael R. Fellows and Michael A. Langston.
Nonconstructive advances in polynomial-time complexity.
Information Processing Letters, 26(3):157–162, 1987.

[FL88] Michael R. Fellows and Michael A. Langston.
Nonconstructive tools for proving polynomial-time decidability.
Journal of the ACM, 35(3):727–739, 1988.

[FL89] M. R. Fellows and M. A. Langston.
On search decision and the efficiency of polynomial-time algorithms.
In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89, pages 501–512, New York, NY, USA, 1989. ACM.

Bibliography V

[FL92] Michael R. Fellows and Michael A. Langston.
Constructivity issues in graph algorithms.
In *Constructivity in Computer Science, Summer Symposium*, pages 150–158, London, UK, UK, 1992. Springer-Verlag.

[GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik.
Concrete Mathematics: A Foundation for Computer Science.
Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994.

[GMC05] Andrei Gagarin, Wendy J. Myrvold, and John Chambers.
Forbidden minors and subdivisions for toroidal graphs with no $K_{3,3}$'s.
Electronic Notes in Discrete Mathematics, 22:151–156, 2005.

Bibliography VI

[HT74] John Hopcroft and Robert Tarjan.
Efficient planarity testing.
Journal of the ACM, 21(4):549–568, October 1974.

[Imp95] Russell Impagliazzo.
A personal view of average-case complexity.
In *Structure in Complexity Theory Conference*, pages 134–147, 1995.

[Joh87] David S. Johnson.
The NP-Completeness Column: An Ongoing Guide. The Many Faces Of Polynomial Time.
Journal of Algorithms, 8:285–303, 1987.

Bibliography VII

[Kur30] Kazimierz Kuratowski.
Sur le Probleme des Courbes Gauches en Topologie.
Fundamenta Mathematicae, 15:271–283, 1930.

[Lan61] Rolf Landauer.
Irreversibility and heat generation in the computing process.
IBM Journal of Research and Development, 5:183–191, 1961.

[MR95] Rajeev Motwani and Prabhakar Raghavan.
Randomized algorithms.
Cambridge University Press, New York, NY, USA, 1995.

[NC00] Michael A. Nielsen and Isaac L. Chuang.
Quantum Computation and Quantum Information.
Cambridge Series on Information and the Natural Sciences.
Cambridge University Press, 2000.

Bibliography VIII

[Nie06] Rolf Niedermeier.
Invitation to Fixed-Parameter Algorithms.
Oxford Lecture Series in Mathematics and Its Applications. OUP Oxford, 2006.

[PV01] M. B. Plenio and V. Vitelli.
The physics of forgetting: Landauers erasure principle and information theory.
Contemporary Physics, 42:25–60, 2001.

[RS83] Neil Robertson and Paul Seymour.
Graph minors. I. Excluding a forest.
Journal of Combinatorial Theory, Series B, 35:39–61, 1983.

Bibliography IX

[RS84] Neil Robertson and Paul Seymour.
Graph minors. III. Planar tree-width.
Journal of Combinatorial Theory, Series B, 36:49–64, 1984.

[RS86a] Neil Robertson and Paul Seymour.
Graph minors. II. Algorithmic aspects of tree-width.
Journal of Algorithms, 7:309–322, 1986.

[RS86b] Neil Robertson and Paul Seymour.
Graph minors. V. Excluding a planar graph.
Journal of Combinatorial Theory, Series B, 41:92–114, 1986.

[RS86c] Neil Robertson and Paul Seymour.
Graph minors. VI. Disjoint paths across a disc.
Journal of Combinatorial Theory, Series B, 41:115–138, 1986.

Bibliography X

[RS88] Neil Robertson and Paul Seymour.
Graph minors. VII. Disjoint paths on a surface.
Journal of Combinatorial Theory, Series B, 45:212–254, 1988.

[RS90a] Neil Robertson and Paul Seymour.
Graph minors. IV. Tree-width and well-quasi-ordering.
Journal of Combinatorial Theory, Series B, 48:227–254, 1990.

[RS90b] Neil Robertson and Paul Seymour.
Graph minors. IX. Disjoint crossed paths.
Journal of Combinatorial Theory, Series B, 49:40–77, 1990.

[RS90c] Neil Robertson and Paul Seymour.
Graph minors. VIII. A Kuratowski theorem for general surfaces.
Journal of Combinatorial Theory, Series B, 48:255–288, 1990.

Bibliography XI

[RS91] Neil Robertson and Paul Seymour.
Graph minors. X. Obstructions to tree-decomposition.
Journal of Combinatorial Theory, Series B, 52:153–190, 1991.

[RS94] Neil Robertson and Paul Seymour.
Graph minors. XI. Circuits on a surface.
Journal of Combinatorial Theory, Series B, 60:72–106, 1994.

[RS95a] Neil Robertson and Paul Seymour.
Graph minors XII. Distance on a surface.
Journal of Combinatorial Theory, Series B, 64:240–272, 1995.

[RS95b] Neil Robertson and Paul Seymour.
Graph minors. XIII. The disjoint paths problem.
Journal of Combinatorial Theory, Series B, 63:65–110, 1995.

Bibliography XII

[RS95c] Neil Robertson and Paul Seymour.
Graph minors. XIV. Extending an embedding.
Journal of Combinatorial Theory, Series B, 65:23–50, 1995.

[RS96] Neil Robertson and Paul Seymour.
Graph minors. XV. Giant steps.
Journal of Combinatorial Theory, Series B, 68:112–148, 1996.

[RS99] Neil Robertson and Paul Seymour.
Graph minors. XVII. Taming a vortex.
Journal of Combinatorial Theory, Series B, 77:162–210, 1999.

[RS03a] Neil Robertson and Paul Seymour.
Graph minors. XVI. Excluding a non-planar graph.
Journal of Combinatorial Theory, Series B, 89:43–76, 2003.

Bibliography XIII

[RS03b] Neil Robertson and Paul Seymour.
Graph minors. XVIII. Tree-decompositions and well-quasiordering.
Journal of Combinatorial Theory, Series B, 89:77–108, 2003.

[RS04a] Neil Robertson and Paul Seymour.
Graph minors. XIX. Well-quasi-ordering on a surface.
Journal of Combinatorial Theory, Series B, 90:325–385, 2004.

[RS04b] Neil Robertson and Paul Seymour.
Graph minors. XX. Wagners conjecture.
Journal of Combinatorial Theory, Series B, 92:325–357, 2004.

[RS09] Neil Robertson and Paul Seymour.
Graph minors. XXI. Graphs with unique linkages.
Journal of Combinatorial Theory, Series B, 99:583–616, 2009.

Bibliography XIV

[RS10] Neil Robertson and Paul Seymour.
Graph minors. XXIII. Nash-Williams immersion conjecture.
Journal of Combinatorial Theory, Series B, 100:181–205, 2010.

[RS12] Neil Robertson and Paul Seymour.
Graph minors. XXII. Irrelevant vertices in linkage problems.
Journal of Combinatorial Theory, Series B, 102:530–563, 2012.

[Sch96] Bruce Schneier.
Applied Crytography.
John Wiley & Sons, 2nd edition, 1996.

[Vaz01] Vijay V. Vazirani.
Approximation algorithms.
Springer-Verlag New York, Inc., New York, NY, USA, 2001.

Bibliography XV

[Wag37] K. Wagner.

ber eine eigenschaft der ebenen komplexe.

Mathematische Annalen, 114(1):570–590, 1937.

[Wor94] Thomas Worsch.

Lower and upper bounds for (sums of) binomial coefficients, 1994.