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"

November 26, 2013
1 Introduction
   - Intractability
   - Thermodynamics and intractability
   - The problem is intractable! Now what?
   - Circumventing intractability
   - The gist of Parameterized Complexity

2 The Vertex Cover problem
   - How Vertex Cover arises in Practice
   - Formal definition and complexity

3 The theory of Robertson of Seymour
   - Graph Minors
   - WQO: Well Quasi Ordering
   - Planarity in the light of the Theory of R. & S.
   - Membership in $\mathcal{P}$ via the Theory of R. & S.
   - Nonconstructivity of the Theory of R. & S.
   - Limitations of the Theory of R. & S.
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 $O(n + 2^{2k^2})$

The Theory of Parameterized Complexity

- Foundations of Parameterized Complexity
- **Vertex Cover** as a parameterized problem

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
Definition of $\mathcal{NP}$-completeness

Let $\Pi$ be a decision problem.
Let \( \Pi \) be a decision problem.

- \( \Pi \) is in the complexity class \( \mathcal{NP} \) if there is a fast algorithm verifying for every \( \text{YES} \)-instance \( X \) of \( \Pi \) that \( X \) is indeed an \( \text{YES} \)-instance.
Let $\Pi$ be a decision problem.

- $\Pi$ is in the complexity class $\mathcal{NP}$ if there is a fast algorithm verifying for every $\textsc{Yes}$-instance $X$ of $\Pi$ that $X$ is indeed an $\textsc{Yes}$-instance.
- $\Pi$ is in the complexity class $\mathcal{NP}$-complete, if $\Pi \in \mathcal{NP}$ and every problem in $\mathcal{NP}$ reduces to $\Pi$ fast.
Definition of $\mathcal{NP}$-completeness

Let $\Pi$ be a decision problem.

- $\Pi$ is in the complexity class $\mathcal{NP}$ if there is a fast algorithm verifying for every $\text{YES}$-instance $X$ of $\Pi$ that $X$ is indeed an $\text{YES}$-instance.

- $\Pi$ is in the complexity class $\mathcal{NP}$-complete, if $\Pi \in \mathcal{NP}$ and every problem in $\mathcal{NP}$ reduces to $\Pi$ fast.

- “fast” means “(in) polynomial time”.

Minko Markov  minkom@fmi.uni-sofia.bg  Parameterized Complexity
Let $\Pi$ be a decision problem.

- $\Pi$ is in the complexity class $\mathcal{NP}$ if there is a fast algorithm verifying for every $\text{YES}$-instance $X$ of $\Pi$ that $X$ is indeed an $\text{YES}$-instance.

- $\Pi$ is in the complexity class $\mathcal{NP}$-complete, if $\Pi \in \mathcal{NP}$ and every problem in $\mathcal{NP}$ reduces to $\Pi$ fast.

- “fast” means “(in) polynomial time”.

- “to verify” means “to verify based on some information that is somehow supplied”. This information is *the certificate*. 
Definition of $\mathcal{NP}$-completeness

Belonging versus reducibility

- A proof of belonging to $\mathcal{NP}$ without reducibility is not a big deal (there are exceptions). All classical easy problems such as Shortest Path are trivially in $\mathcal{NP}$.
Definition of \( \mathcal{NP} \)-completeness

Belonging versus reducibility

- A proof of belonging to \( \mathcal{NP} \) without reducibility is not a big deal (there are exceptions). All classical easy problems such as \textsc{Shortest Path} are trivially in \( \mathcal{NP} \).

- A proof of reducibility without belonging to \( \mathcal{NP} \) implies \( \mathcal{NP} \)-hardness. In this case the problem is \textit{at least} as hard as any problem in \( \mathcal{NP} \) and maybe worse. For instance, \textsc{The Halting Problem} is \( \mathcal{NP} \)-hard.
A proof of belonging to \( \mathcal{NP} \) without reducibility is not a big deal (there are exceptions). All classical easy problems such as Shortest Path are trivially in \( \mathcal{NP} \).

A proof of reducibility without belonging to \( \mathcal{NP} \) implies \( \mathcal{NP} \)-hardness. In this case the problem is at least as hard as any problem in \( \mathcal{NP} \) and maybe worse. For instance, The Halting Problem is \( \mathcal{NP} \)-hard. How?
Definition of $NP$-completeness

Belonging versus reducibility

- A proof of belonging to $NP$ without reducibility is not a big deal (there are exceptions). All classical easy problems such as Shortest Path are trivially in $NP$.

- A proof of reducibility without belonging to $NP$ implies $NP$-hardness. In this case the problem is at least as hard as any problem in $NP$ and maybe worse. For instance, The Halting Problem is $NP$-hard. How?

  Given any instance $X$ of any problem $\Pi \in NP$, poly-time reduce it to some $Z \in SAT$. Let $T$ be the encoding of a Turing machine that takes as input any instance $A$ of SAT and halts iff $A \in YES_{SAT}$. The ordered pair $\langle T, Z \rangle$ is an instance of The Halting Problem. Furthermore, $\langle T, Z \rangle$ is an YES-instance iff $X \in YES_{\Pi}$. Note that $T$ has constant size.
Thousands of problems arising in practice turn out to be intractable. That is considered *A Bad Thing*.
Intractability as a curse

- Thousands of problems arising in practice turn out to be intractable. That is considered a Bad Thing.
- As a rule, they are $\mathcal{NP}$-hard, with rare exceptions such as Graph Isomorphism or Integer Factorization.
Thousands of problems arising in practice turn out to be intractable. That is considered **A Bad Thing**.

As a rule, they are $\mathcal{NP}$-hard, with rare exceptions such as **Graph Isomorphism** or **Integer Factorization**.

A considerable part of them are the $\mathcal{NP}$-complete ones. In spite of a great effort, it is only conjectured they are intractable. A solid proof would be nice...
Intractability as a curse

- Thousands of problems arising in practice turn out to be intractable. That is considered a **Bad Thing**.
- As a rule, they are $\mathcal{NP}$-hard, with rare exceptions such as **Graph Isomorphism** or **Integer Factorization**.
- A considerable part of them are the $\mathcal{NP}$-complete ones. In spite of a great effort, it is only conjectured they are intractable. A solid proof would be nice...
- The famous question $\mathcal{P} \overset{?}{=} \mathcal{NP}$ is in fact: are there really any intractable problems in $\mathcal{NP}$?
Intractability as a blessing

- If $\mathcal{P} = \mathcal{NP}$ then public-key cryptography is *impossible*.
Intractability as a blessing

- If $\mathcal{P} = \mathcal{NP}$ then public-key cryptography is impossible.
- A side note: the inverse is not necessarily true. $\mathcal{P} \neq \mathcal{NP}$ implies public-key cryptography is possible only under additional assumptions as pointed out by Russell Impagliazzo. (a video lecture of R. Impagliazzo on the same topic).
Intractability as a blessing

- If $\mathcal{P}=\mathcal{NP}$ then public-key cryptography is \textit{impossible}. 
- A side note: the inverse is not necessarily true. $\mathcal{P} \neq \mathcal{NP}$ implies public-key cryptography is possible only under additional assumptions \textit{as pointed out by Russell Impagliazzo}. \textit{(a video lecture of R. Impagliazzo on the same topic)}. 
- If $\mathcal{P}=\mathcal{NP}$ then “true” creativity and intuition do not exist. Everything that we call creative turns out to be achievable through purely mechanical means, \textit{i.e.} an algorithm.
Intractability as a blessing

- If $\mathcal{P} = \mathcal{NP}$ then public-key cryptography is impossible.
- A side note: the inverse is not necessarily true. $\mathcal{P} \neq \mathcal{NP}$ implies public-key cryptography is possible only only under additional assumptions as pointed out by Russell Impagliazzo. (a video lecture of R. Impagliazzo on the same topic).
- If $\mathcal{P} = \mathcal{NP}$ then “true ”creativity and intuition do not exist. Everything that we call creative turns out to be achievable through purely mechanical means, i.e. an algorithm.
- Consequently, if—and this is a big IF—$\mathcal{P}$ equals $\mathcal{NP}$, the world we live in is, in a profound sense, simple and uninteresting. Mathematical proofs and software constructions are easily obtainable via a general procedure. Artificial Intelligence is possible.
Intractability as a physical phenomenon

Most arguments for the intractability implied by $2^n$ go along the line ”we do not have that much time”. There is another argument: we do not have that much energy.

While algorithmic undecidability is a phenomenon from mathematics and logic, intractability as in $NP$-complete problems is rooted in physics. To establish The Halting Problem is not solvable by an algorithm takes (theoretically...) only paper and pencil. No knowledge of the real world is required to get the result.

In contrast, assuming $NP$-complete problems require exponential time at worst, the reason $NP$-complete problems are intractable cannot be grasped without knowledge of the real world. $2^n$ implies intractability because no real computer can perform, say, $2^{200}$ elementary operations, and 200 is a reasonable value of $n$.

The following argument is suggested by Bruce Schneier’s Applied Cryptography (Schneier, pp. 157–158).
In 1961 Rolf Landauer shows that $kT \ln 2$ is an absolute lower bound for the energy necessary to flip a bit during an irreversible computation. That is independent of the specifics of the computational device.

$k = 1.3806488 \times 10^{-23} \text{ J K}^{-1}$ is Boltzmann’s constant and $T$ is the temperature in kelvins.
In 1961 Rolf Landauer shows that $kT \ln 2$ is an absolute lower bound for the energy necessary to flip a bit during an irreversible computation. That is independent of the specifics of the computational device.

$k = 1.3806488 \times 10^{-23} \text{JK}^{-1}$ is Boltzmann’s constant and $T$ is the temperature in kelvins.

Of course, if $T = 0$ then the bound is 0 but $T = 0$ is unrealistic. The Cosmic Microwave Background Radiation is about 3 kelvins.
Suppose an $n$-bit register goes through all possible states from 00...0 to 11...1 by adding one. The number of the bit flips in that process is $T(n) = 2T(n-1) + n$, $T(1) = 1$. The solution to that recurrence is $T(n) = 2 \times 2^n - n - 2 \approx 2^{n+1}$.
Suppose an \( n \)-bit register goes through all possible states from 00...0 to 11...1 by adding one. The number of the bit flips in that process is \( T(n) = 2T(n-1) + n, \) \( T(1) = 1. \) The solution to that recurrence is \( T(n) = 2 \times 2^n - n - 2 \approx 2^{n+1}. \)

Note that \( T(192) \approx 10^{59}. \) Assuming 1K background temperature, Landauer’s principle imposes a lower bound of \( \approx 10^{36} J \) to the energy required to perform all bit flips. That is roughly the energy the Sun emits for a hundred years. Not the solar energy the Earth gets but the total energy released by Sun.
Suppose an $n$-bit register goes through all possible states from 00...0 to 11...1 by adding one. The number of the bit flips in that process is $T(n) = 2T(n-1) + n$, $T(1) = 1$. The solution to that recurrence is $T(n) = 2 \times 2^n - n - 2 \approx 2^{n+1}$.

Note that $T(192) \approx 10^{59}$. Assuming 1K background temperature, Landauer’s principle imposes a lower bound of $\approx 10^{36} \text{J}$ to the energy required to perform all bit flips. That is roughly the energy the Sun emits for a hundred years. Not the solar energy the Earth gets but the total energy released by Sun.

For a 235-bit register that lower bound is $\approx 10^{47} \text{J}$ and that is about the energy released by the most powerful cosmic eruption recorded (GRB).
Landauer’s principle implies the intractability of the exponential time complexity

Clearly, a device that can count up to $2^{235}$ or $2^{192}$ is physically impossible.
Landauer’s principle implies the intractability of the exponential time complexity

Clearly, a device that can count up to $2^{235}$ or $2^{192}$ is physically impossible.

Within the current framework, of course. Bruce Schneier says:

*The very notion of predicting computing power 10 years in the future, let alone 50 years is absolutely ridiculous.*
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!*
Heuristics: good ideas with no or insufficient analysis of their goodness.
Approaches against intractability

1. **Heuristics**: good ideas with no or insufficient analysis of their goodness.

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.
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.
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.

Randomized algorithms: using randomness as a resource. It is not known whether that can turn intractable problems into tractable ones, in the probabilistic sense.
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.

Randomized algorithms: using randomness as a resource. It is not known whether that can turn intractable problems into tractable ones, in the probabilistic sense.

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.
Consider 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.
Consider 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.
Consider 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 group of professor X. has made 200 experiments. However, the results of some pairs of experiments are in conflict. Has the group achieved a scientific breakthrough, or some experiments were shoddy? First of all, professor X. would like to know which experiments cause the conflicts (assuming they are a definite minority). In other words, the removal of which experiments leaves no conflicts.
The group of professor X. has made 200 experiments. However, the results of some pairs of experiments are in conflict. Has the group achieved a scientific breakthrough, or some experiments were shoddy? First of all, professor X. would like to know which experiments cause the conflicts (assuming they are a definite minority). In other words, the removal of which experiments leaves no conflicts.

Of course, if all experiments are removed from consideration there will be no conflicts left...
The group of professor X. has made 200 experiments. However, the results of some pairs of experiments are in conflict. Has the group achieved a scientific breakthrough, or some experiments were shoddy? First of all, professor X. would like to know which experiments cause the conflicts (assuming they are a definite minority). In other words, the removal of which experiments leaves no conflicts.

Of course, if all experiments are removed from consideration there will be no conflicts left...

What is desired is a \textit{minimum} subset of troublesome experiments.
The problem of the conflicting experiments
Modeling by an undirected graph

If the graph of the conflicts is a star graph, the answer is obvious.
The problem of the conflicting experiments
Modeling by an undirected graph

If the graph of the conflicts is arbitrary the answer can be extremely *unobvious*.
The problem of the conflicting experiments
Modeling by an undirected graph

If the graph of the conflicts is arbitrary the answer can be extremely *unobvious*.
The problem of the conflicting experiments
Modeling by an undirected graph

If the graph of the conflicts is arbitrary the answer can be extremely *unobvious*.
The problem of the conflicting experiments
Modeling by an undirected graph

A side note: I am not sure 8 is the optimum but the fact the graph has 13 vertices and is Hamiltonian imposes a lower bound $7 = \lceil \frac{13}{2} \rceil$ on the number of conflict-causing experiments.
In the classical complexity theory, every decision problem is defined by two components: a generic instance and a \textsc{Yes}/\textsc{No} question.
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**:
In the classical complexity theory, every decision problem is defined by two components: a generic instance and a \textsc{Yes}/\textsc{No} question.

**Definition of Vertex Cover:**

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

Minko Markov  
minkom@fmi.uni-sofia.bg
In the classical complexity theory, every decision problem is defined by two components: a generic instance and a \textit{Yes}/\textit{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 $\leq k$?
In the classical complexity theory, every decision problem is defined by two components: a generic instance and a \textsc{Yes/No} question.

Definition of \textsc{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

$$\forall (x, y) \in E : x \in U \lor y \in U$$
Vertex Cover is one of the “original” 21 $\mathcal{NP}$-complete problems of Karp \textit{from 1972}. Under the assumption that $\mathcal{P} \neq \mathcal{NP}$, the problem is intractable.
**Vertex Cover** is one of the “original” 21 $NP$-complete problems of Karp *from 1972*. Under the assumption that $P \neq NP$, 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.
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
$\binom{n}{\lfloor n/2 \rfloor} = \Theta(2^n / \sqrt{n})$. 

Minko Markov minkom@fmi.uni-sofia.bg

Parameterized Complexity
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.
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*

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

where \( \alpha \) 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**).
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.

Ostensibly, $n^3$ 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”.
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 $2^{2^2 \cdots^2}$ 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.
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

\[ 2^{2^{\ldots^{2}}} \]

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

$$2^{2^2 \cdots 2}$$

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 algorithms, 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

$$2^{2^2^{\ldots^2}}$$

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 algorithms, since for every $k$ there is a different algorithm.

- The existence proof follows from the theory of Robertson and Seymour.
The theory of Robertson of Seymour

Neil Robertson

Paul Seymour

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 $\mathcal{F}$.
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: $W_4 \preceq Q_3$
An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$
An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$
An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$
An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$
An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$
An example of edge removals, edge contractions, and minor: $W_4 \preceq Q_3$
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 simple 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 simple 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

$K_{3,3} \not\leq Q_3$ because $Q_3$ is planar and $K_{3,3}$ is not
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 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 \( \leq \) relation is preorder (quasi-order)

- The \( \leq \) 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 \( A \):

\[
\forall x \forall y : xRy \iff |x| \leq |y|
\]
Definition

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

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

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

\[
F(X) = \{ z \in S | \exists a \in X (a \leq z) \}
\]

- The ideal generated by \( X \) is

\[
I(X) = \{ z \in S | \exists a \in X (z \leq a) \}
\]

- 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)

\[
\langle S, \leq \rangle \text{ is a WQO iff for every subset } X \subseteq S, F(X) \text{ is finitely generated.}
\]
Definition

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 $\langle S, \leq \rangle$ we can compute in polynomial time whether any $z \in S$ is 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”

**Definition**

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 $\langle S, \leq \rangle$ we can compute in polynomial time whether any $z \in S$ is 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.
The theory of Robertson of Seymour

WQO

WQO principle: statement using “filters”

Definition

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 $\langle S, \leq \rangle$ we can compute in polynomial time whether any $z \in S$ is 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.
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.
Wagner’s conjecture is true: the set of all finite graphs is well quasi ordered by the minor order $\preceq$.

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

Minko Markov  minkom@fmi.uni-sofia.bg

Parameterized Complexity
The theory of Robertson of Seymour

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 $K_{3,3}$.
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 $K_{3,3}$. 
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 $K_{3,3}$. 
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 $K_{3,3}$. 
The theory of Robertson of Seymour

Planarity

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 $K_{3,3}$.
The theory of Robertson of Seymour

Planarity

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 $K_{3,3}$.
The theory of Robertson of Seymour

Planarity

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.
The theory of Robertson of Seymour

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.
The theory of Robertson of Seymour

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.
Consider the following facts:

- Any graph is either planar or non-planar (obvious).
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.).
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.
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.
There exists a finite set of obstructions to planarity (follows trivially from the equivalent definition of WQO).
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).
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: \( \{K_5, K_{3,3}\} \).
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).
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 \textit{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.
Every compact closed orientable surface is topologically equivalent to a sphere with added $\geq 0$ “handles”. The number of the “handles” is its \textit{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.
**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.
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 $\text{NP}$-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.
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).
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*. 
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**
If $G$ is covered by $\leq k$ vertices and $H \leq G$ then $H$ is covered by $\leq k$ vertices, too.
If $G$ is covered by $\leq k$ vertices and $H \subseteq G$ then $H$ is covered by $\leq k$ vertices, too.

Therefore VERTEX COVER is solvable in $O(n^3)$ for all fixed $k$. 
If $G$ is covered by $\leq k$ vertices and $H \leq G$ then $H$ is covered by $\leq k$ vertices, too.

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

The obstruction set for every $k$ value is \textbf{different}. There are 188 connected obstructions for $k = 6$. 
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 $n^k$. 
The theory does not give a clue how to compute the obstructions. It merely 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 merely 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 $F$. 

Minko Markov  minkom@fmi.uni-sofia.bg  Parameterized Complexity
Embeddability in surfaces of genus $k$, linkless 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).
Embeddability in surfaces of genus $k$, linkless 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).
Embeddability in surfaces of genus $k$, linkless 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*:
Given a graph $G$ and number $k$ construct a binary tree of height $\leq k$ \textit{in the following way}:

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


Minko Markov  

minkom@fmi.uni-sofia.bg
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.
Given a graph $G$ and number $k$ construct a binary tree of height $\leq k \text{ in the following way:}$

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.
correctness In every tree vertex at height $\ell$ the label contains a set of graph vertices of size $\ell$; 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 $\ell$ 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.
correctness In every tree vertex at height $\ell$ the label contains a set of graph vertices of size $\ell$; 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 $\ell$ 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.
Fast algorithms for Vertex Cover for small $k$  

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

![Graph Image]

Complexity $O(2^k n)$
Fast algorithms for Vertex Cover for small $k$

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

Construct the root of the tree

$\langle \emptyset, G \rangle$
The algorithm of Downey and Fellows at work, $k = 2$
Choose arbitrarily $(u, b)$. Any vertex cover of $G$ must cover $(u, b)$.
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

Complexity $O(2^k n)$
Fast algorithms for Vertex Cover for small $k$

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

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

There is no vertex cover of size 2 that contains $u$
The algorithm of Downey and Fellows at work, $k = 2$

However, there is vertex cover of size 2 that contains $b$
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.
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 $k = 20$ and $n = 100,000$. A simple calculation

$$100,000^{20} = 10^{25}$$

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

shows the difference is 14 decimal orders of magnitude...
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 \textsc{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.
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.
A trivial observation: if the graph has vertex $u$ of degree $> k$ then $u$ is in every vertex cover of size $\leq k$. 
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 $\text{No}$. 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 \texttt{No}. Else, let $\ell = k - p$.
3. $H \leftarrow G - W$. 

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 \( \text{No} \). Else, let \( \ell = k - p \).

3. \( H \leftarrow G - W \).

4. If \( |E(H)| > k\ell \) then return \( \text{No} \).
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 $\ell$-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 $\ell$-vertex cover then return No.
6. Else, the union of any $\ell$-vertex cover of $H$ and $W$ is a $k$-vertex cover for $G$. 
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 $H$. $\ell$ 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 $n$. 
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 $O(\sqrt{5}^k n)$ 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 $O(1.2738^k n^{O(1)})$. 
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.
A parameterized decision problem is $L \subseteq \Sigma^* \times \Sigma^*$ where $\Sigma$ is the input alphabet. If $(\sigma, k) \in L$, the parameter is $k$.

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

The creators of the theory

Michael Ralph Fellows and Rodney Graham Downey

Mike Fellows

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

Definition of $p$-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 $k$. 
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 \( k \)
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 $\leq k$?
Every parameterized decision problem is defined by *three* components: generic instance, parameter, and question with *Yes/No* answer.

**Definition of \texttt{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 \texttt{Hamiltonian Cycle}—while every classical decision problem can be parameterized. For instance, \texttt{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 $x$. 

As we saw, $p$-Vertex Cover $\in \mathcal{FPT}$. The complexity expression can as well be $O(g(y) + n^{O(1)})$: both formulations are equivalent.
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, $p$-**Vertex Cover** $\in \mathcal{FPT}$. 
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, $p$-**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 \( \text{FPT} \).

It has been proved that \( \text{p-Dominating Set} \not\in \text{FPT} \), under the assumption that every \( \text{NP} \)-complete problem necessitates deterministic exponential time.
Not all parameterized problems are in $FPT$

It has been proved that \textit{p-Dominating Set} $\not\in FPT$, under the assumption that every $\mathcal{NP}$-complete problem necessitates deterministic exponential time.

Definition of \textit{p-Dominating Set}:
Not all parameterized problems are in \( \text{FPT} \)

It has been proved that \( p\)-\textsc{Dominating Set} \( \not\in \text{FPT} \), under the assumption that every \( \text{NP}\)-complete problem necessitates deterministic exponential time.

**Definition of \( p\)-\textsc{Dominating Set}**:  

- An ordered pair of undirected graph \( G = (V, E) \) and a natural number \( k \).
Not all parameterized problems are in \( \mathcal{FPT} \)

It has been proved that \( p\text{-Dominating Set} \not\in \mathcal{FPT} \), under the assumption that every \( \mathcal{NP} \)-complete problem necessitates deterministic exponential time.

**Definition of \( p\text{-Dominating Set} \):**

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

Minko Markov  
minkom@fmi.uni-sofia.bg  
Parameterized Complexity
Not all parameterized problems are in $\mathcal{FPT}$

It has been proved that $p$-DOMINATING SET $\not\in \mathcal{FPT}$, under the assumption that every $\mathcal{NP}$-complete problem necessitates deterministic exponential time.

**Definition of $p$-DOMINATING SET:**

- An ordered pair of undirected graph $G = (V, E)$ and a natural number $k$.
- Parameter $k$.
- Does $G$ dominating set of size $\leq k$?
It has been proved that $p$-\textsc{Dominating Set} $\not\in \mathcal{FPT}$, under the assumption that every $\mathcal{NP}$-complete problem necessitates deterministic exponential time.

Definition of $p$-\textsc{Dominating Set}:

- An ordered pair of undirected 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

$$\forall x \in V : x \in U \lor \exists y \in U((x, y) \in E)$$
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 $10^{20}$. 
<table>
<thead>
<tr>
<th>FPT</th>
<th>Klam value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>klam values for (p-V_\text{e}\text{r}\text{t}\text{e}\xspace x\text{C}\text{o}\text{v}\text{e}\text{r})</strong></td>
<td></td>
</tr>
</tbody>
</table>

The complexities of the current fastest algorithms for a number of FPT problems can be seen at [fpt.wikidot.com/fpt-races](http://fpt.wikidot.com/fpt-races). The \(f(k)\) function of \(p-V_\text{e}\text{r}\text{t}\text{e}\xspace x\text{C}\text{o}\text{v}\text{e}\text{r}\) is \(1.2738^k\). The solution to the equation \(1.2738^k = 10^{20}\) is approximately \(k = 190\) and that is the current klam value for that problem.
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 equation $1.2738^k = 10^{20}$ 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 is $1.1616^k$ and the klam value is 307.
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 equation $1.2738^k = 10^{20}$ 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 is $1.1616^k$ and the klam value is 307.

For $2^k$ (the algorithm for Vertex Cover of Downey and Fellows) the klam value is 66.
\textbf{p-Vertex Cover} is the champion of efficient parameterization

Clearly, \textbf{p-Vertex Cover} stands out among all \textit{parameterized problems}. For some reason, \textbf{Vertex Cover} is much more amenable to efficient parameterization than the remaining FPT problems.

Compare the klam value of \textbf{Vertex Cover} with the one of \textbf{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

\textit{p-Vertex Cover} seems to be the most natural way to parameterize \textit{Vertex Cover}. However, the latter can be parameterized in a number of ways. The results are \textit{different} parameterized problems that have different properties.
Suppose we are given a graph with $n$ vertices and a number $k$. 

As parameterized problems, they are different problems.
Suppose we 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).
Suppose we 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.
We can parameterize by the treewidth of the graph. It is \textit{well-known} that if the treewidth is $\leq t$ \textsc{Vertex Cover} can be solved in time $O(2^t n)$ regardless of the vertex cover of 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 \textsc{Vertex Cover}.
- Reducing the problem to a kernel, *e.g.* the algorithm of Buss for \textsc{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.
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**

<table>
<thead>
<tr>
<th><strong>Definition (Definition 2.1, Flum, Grohe 2006)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Let $\Pi = \langle Q, \kappa \rangle$ and $\Pi' = \langle Q', \kappa' \rangle$ be parameterized problems over the alphabet $\Sigma$. <em><em>An FPT reduction from $\Pi$ to $\Pi'$ is a function $\psi : \Sigma^</em> \rightarrow \Sigma^</em>$ such that:**</td>
</tr>
<tr>
<td>- $\forall x \in \Sigma^* (x \in Q \leftrightarrow \psi(x) \in Q')$.</td>
</tr>
<tr>
<td>- $\psi$ is computable by an FRP algorithm with respect to $\kappa$. That means, $\psi(x)$ is computable in time $f(\kappa(x)) \cdot</td>
</tr>
<tr>
<td>- There exists a computable function $g : \mathbb{N} \rightarrow \mathbb{N}$, such that $\forall x \in \Sigma^* : \kappa'(\psi(x)) \leq g(\kappa(x))$.</td>
</tr>
</tbody>
</table>
\( \mathcal{FPT} \) is the easy parameterized class. 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.

**Lemma (Lemma 2.2, Flum, Grohe 2006)**

*The complexity class \( \mathcal{FPT} \) is closed with respect to FPT reductions. That is, if \( \Pi \) is reduced to \( \Pi' \) via FPT reduction and \( \Pi' \in \mathcal{FPT} \), then \( \Pi \in \mathcal{FPT} \).*
Parameterized Complexity Theory provides powerful tools for dealing with intractability, in case the instances have small parameter.

Often initial nonconstructive 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 unknown. 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


[Cou90] Bruno Courcelle.
The monadic second-order logic of graphs. I. Recognizable sets of finite graphs.

Fixed-parameter tractability and completeness.

*Parameterized Complexity.*
Springer-Verlag, 1999.
530 pp.
[Die12] Reinhard Diestel.

*Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*.

[DX02] Michael J. Dinneen and Liu Xiong.

Minor-order obstructions for the graphs of vertex cover six.

[FG06] Jörg Flum and Martin Grohe.

*Parameterized Complexity Theory*.
Nonconstructive advances in polynomial-time complexity. 

Nonconstructive tools for proving polynomial-time decidability. 

On search decision and the efficiency of polynomial-time algorithms. 


Efficient planarity testing.

[Int95] Russell Impagliazzo.
A personal view of average-case complexity.

Sur le Probleme des Courbes Gauches en Topologie.

[Lan61] Rolf Landauer.
Irreversibility and heat generation in the computing process.

[MR95] Rajeev Motwani and Prabhakar Raghavan.
*Randomized algorithms*.

*Quantum Computation and Quantum Information*.
[Nie06] Rolf Niedermeier.  
*Invitation to Fixed-Parameter Algorithms.*  

The physics of forgetting: Landauers erasure principle and information theory.  
*Contemporary Physics, 42:25–60, 2001.*

Graph minors. I. Excluding a forest.  
Graph minors. III. Planar tree-width.  

Graph minors. II. Algorithmic aspects of tree-width.  

Graph minors. V. Excluding a planar graph.  

Graph minors. VI. Disjoint paths across a disc.  


Graph minors. X. Obstructions to tree-decomposition. 

Graph minors. XI. Circuits on a surface. 

[RS95a] Neil Robertson and Paul Seymour. 
Graph minors XII. Distance on a surface. 

Graph minors. XIII. The disjoint paths problem. 
Graph minors. XIV. Extending an embedding.  

Graph minors. XV. Giant steps.  

Graph minors. XVII. Taming a vortex.  

[RS03a] Neil Robertson and Paul Seymour.  
Graph minors. XVI. Excluding a non-planar graph.  
[RS03b] Neil Robertson and Paul Seymour.
Graph minors. XVIII. Tree-decompositions and well-quasiordering.

Graph minors. XIX. Well-quasi-ordering on a surface.

Graph minors. XX. Wagners conjecture.

Graph minors. XXI. Graphs with unique linkages.


ber eine eigenschaft der ebenen komplexe.

[Wor94] Thomas Worsch.