The Optimality Program in Parameterized Algorithms

Dániel Marx

Institute for Computer Science and Control,
Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary

COCOON 2017
Hong Kong
August 3, 2017

Parameterized problems

Main idea

Instead of expressing the running time as a function $T(n)$ of n, we express it as a function $T(n, k)$ of the input size n and some parameter k of the input.

In other words: we do not want to be efficient on all inputs of size n, only for those where k is small.

Parameterized problems

Main idea

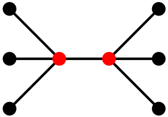
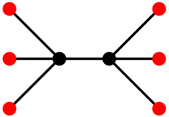
Instead of expressing the running time as a function $T(n)$ of n, we express it as a function $T(n, k)$ of the input size n and some parameter k of the input.

In other words: we do not want to be efficient on all inputs of size n, only for those where k is small.

What can be the parameter k?

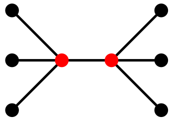
- The size k of the solution we are looking for.
- The maximum degree of the input graph.
- The dimension of the point set in the input.
- The length of the strings in the input.
- The length of clauses in the input Boolean formula.
- $\dots$

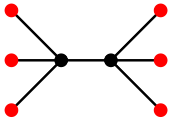
Parameterized complexity

Problem:	VERTEX COVER	INDEPENDENT SET
Input:	Graph G, integer k	Graph G, integer k
Question:	Is it possible to cover the edges with k vertices?	Is it possible to find k independent vertices?
		
Complexity:	NP-complete	NP-complete

Parameterized complexity

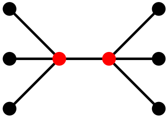
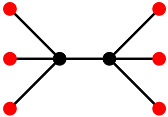
Problem:	VERTEX COVER	INDEPENDENT SET
Input:	Graph G, integer k	Graph G, integer k
Question:	Is it possible to cover the edges with k vertices?	Is it possible to find k independent vertices?





Complexity:	NP-complete	NP-complete
Brute force:	$O(n^k)$ possibilities	$O(n^k)$ possibilities

Parameterized complexity

Problem:	VERTEX COVER	INDEPENDENT SET
Input:	Graph G, integer k	Graph G, integer k
Question:	Is it possible to cover the edges with k vertices?	Is it possible to find k independent vertices?
		
Complexity:	NP-complete	NP-complete
Brute force:	$O(n^k)$ possibilities	$O(n^k)$ possibilities
	$O(2^k n^2)$ **algorithm exists**	**No** $n^{o(k)}$ **algorithm known**

Bounded search tree method

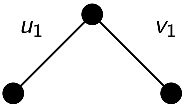
Algorithm for Vertex Cover:

$e_1 = u_1 v_1$

Bounded search tree method

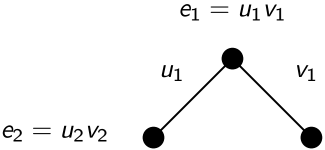
Algorithm for VERTEX COVER:

$e_1 = u_1 v_1$



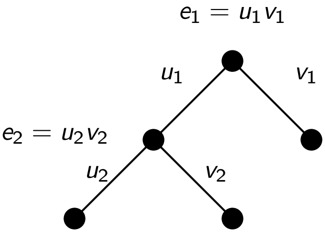
Bounded search tree method

Algorithm for Vertex Cover:



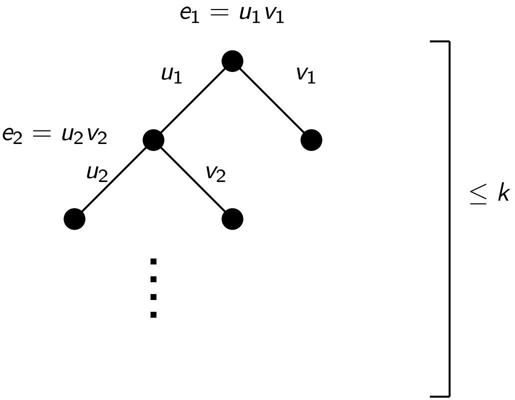
Bounded search tree method

Algorithm for Vertex Cover:



Bounded search tree method

Algorithm for VERTEX COVER:



Height of the search tree $\leq k$ $\Rightarrow$ at most 2^k leaves $\Rightarrow$ $2^k \cdot n^{O(1)}$ time algorithm.

Fixed-parameter tractability

Main definition

A parameterized problem is **fixed-parameter tractable (FPT)** if there is an $f(k)n^c$ time algorithm for some constant c.

Fixed-parameter tractability

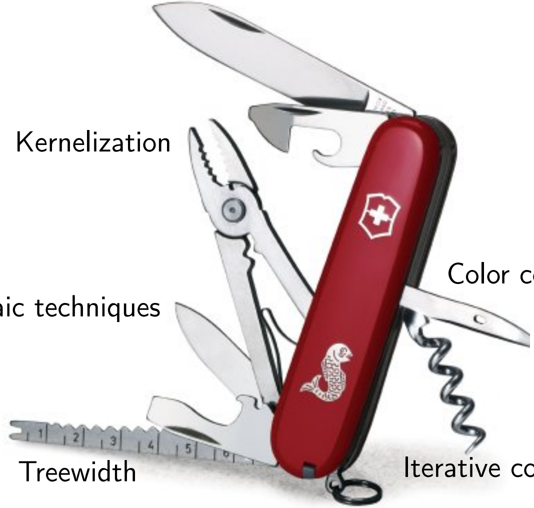
Main definition

A parameterized problem is **fixed-parameter tractable (FPT)** if there is an $f(k)n^c$ time algorithm for some constant c.

Examples of NP-hard problems that are FPT:

- Finding a vertex cover of size k.
- Finding a path of length k.
- Finding k disjoint triangles.
- Drawing the graph in the plane with k edge crossings.
- Finding disjoint paths that connect k pairs of points.
- ...

FPT techniques



W[1]-hardness

Negative evidence similar to NP-completeness. If a problem is **W[1]-hard,** then the problem is not FPT unless FPT=W[1].

Some W[1]-hard problems:

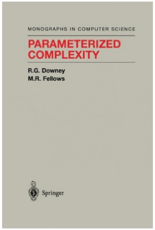
- Finding a clique/independent set of size k.
- Finding a dominating set of size k.
- Finding k pairwise disjoint sets.
- ...

Parameterized complexity

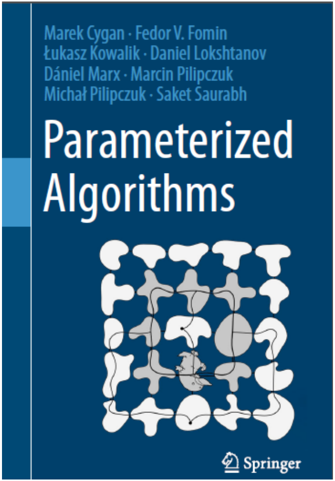
Rod G. Downey
Michael R. Fellows

Parameterized Complexity

Springer 1999



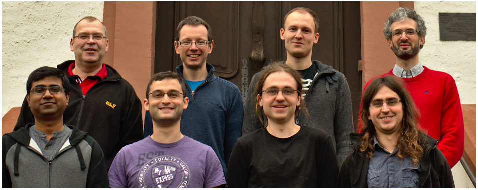
- The study of parameterized complexity was initiated by Downey and Fellows in the early 90s.
- First monograph in 1999.
- By now, strong presence in most algorithmic conferences.



Parameterized Algorithms

Marek Cygan, Fedor V. Fomin,
Lukasz Kowalik, Daniel Lokshtanov,
Dániel Marx, Marcin Pilipczuk,
Michał Pilipczuk, Saket Saurabh

Springer 2015



Shift of focus

qualitative question

FPT or W[1]-hard?

Shift of focus

qualitative question

FPT or W[1]-hard?

FPT

W[1]-hard

quantitative question

What is the best possible multiplier $f(k)$ in the running time $f(k) \cdot n^{O(1)}$?

2^k? 1.0001^k? $2^{\sqrt{k}}$?

What is the best possible exponent $g(k)$ in the running time $f(k) \cdot n^{g(k)}$?

$n^{O(k)}$? $n^{\log k}$? $n^{\log \log k}$?

Better algorithms for Vertex Cover

- We have seen a $2^k \cdot n^{O(1)}$ time algorithm.
- Easy to improve to, e.g., $1.618^k \cdot n^{O(1)}$.
- Current best $f(k)$: $1.2738^k \cdot n^{O(1)}$ [Chen, Kanj, Xia 2010].
- Lower bounds?
  - Is, say, $1.001^k \cdot n^{O(1)}$ time possible?
  - Is $2^{k/\log k} \cdot n^{O(1)}$ time possible?

Better algorithms for Vertex Cover

- We have seen a $2^k \cdot n^{O(1)}$ time algorithm.
- Easy to improve to, e.g., $1.618^k \cdot n^{O(1)}$.
- Current best $f(k)$: $1.2738^k \cdot n^{O(1)}$ [Chen, Kanj, Xia 2010].
- Lower bounds?
  - Is, say, $1.001^k \cdot n^{O(1)}$ time possible?
  - Is $2^{k/\log k} \cdot n^{O(1)}$ time possible?

Of course, for all we know, it is possible that $P = NP$ and Vertex Cover is polynomial-time solvable.

$\Rightarrow$ We can hope only for conditional lower bounds.

Exponential Time Hypothesis (ETH)

Hypothesis introduced by Impagliazzo, Paturi, and Zane:

Exponential Time Hypothesis (ETH) [consequence of]

There is no $2^{o(n)}$-time algorithm for n-variable $\mathrm{3SAT}$.

Note: current best algorithm is 1.30704^n [Hertli 2011].

Note: an n-variable $\mathrm{3SAT}$ formula can have $m = \Omega(n^3)$ clauses.

Exponential Time Hypothesis (ETH)

Hypothesis introduced by Impagliazzo, Paturi, and Zane:

Exponential Time Hypothesis (ETH) [consequence of]

There is no $2^{o(n)}$-time algorithm for n-variable $\mathrm{3SAT}$.

Note: current best algorithm is 1.30704^n [Hertli 2011].

Note: an n-variable $\mathrm{3SAT}$ formula can have $m = \Omega(n^3)$ clauses.

Are there algorithms that are subexponential in the size $n + m$ of the $\mathrm{3SAT}$ formula?

Sparsification Lemma [Impagliazzo, Paturi, Zane 2001]

There is a $2^{o(n)}$-time algorithm for n-variable $\mathrm{3SAT}$.

$$\Updownarrow$$

There is a $2^{o(n+m)}$-time algorithm for n-variable m-clause $\mathrm{3SAT}$.

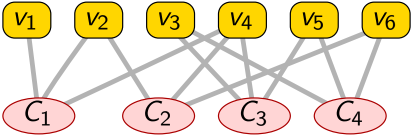
Lower bounds based on ETH

Exponential Time Hypothesis (ETH)

There is no $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

The textbook reduction from 3SAT to VERTEX COVER:

3SAT formula ϕ		Graph G
n variables	$\Rightarrow$	$O(n+m)$ vertices
m clauses		$O(n+m)$ edges



Corollary

Assuming ETH, there is no $2^{o(n)}$ algorithm for VERTEX COVER on an n-vertex graph G.

Other problems

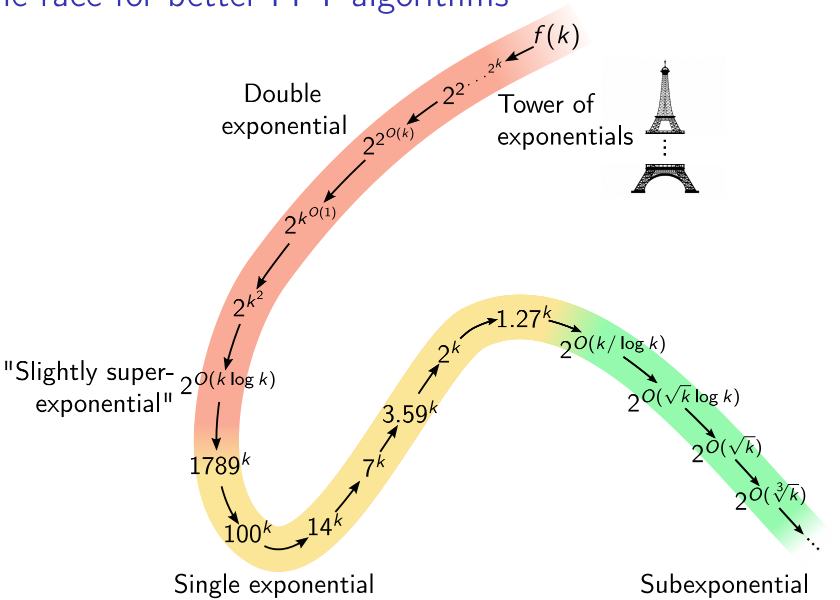
There are polytime reductions from 3SAT to many problems such that the reduction creates a graph with $O(n+m)$ vertices/edges.

Consequence: Assuming ETH, the following problems cannot be solved in time $2^{o(n)}$ and hence in time $2^{o(k)} \cdot n^{O(1)}$ (but $2^{O(k)} \cdot n^{O(1)}$ time algorithms are known):

- Vertex Cover
- Longest Cycle
- Feedback Vertex Set
- Multiway Cut
- Odd Cycle Transversal
- Steiner Tree
- ...

Seems to be the natural behavior of FPT problems?

The race for better FPT algorithms



Graph Minors Theory

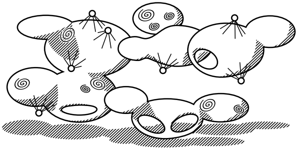
Neil Robertson

Paul Seymour

Theory of graph minors developed in the monumental series

Graph Minors I–XXIII.
J. Combin. Theory, Ser. B
1983–2012

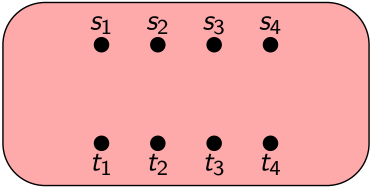
- Structure theory of graphs excluding minors (and much more).
- Galactic combinatorial bounds and running times.
- Important early influence for parameterized algorithms.



[figure by Felix Reidl]

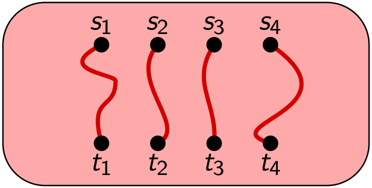
Disjoint paths

k-DISJOINT PATHS
Given a graph G and pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$, find pairwise vertex-disjoint paths $P_1, \ldots, P_k$ such that P_i connects s_i and t_i.



Disjoint paths

k-DISJOINT PATHS
Given a graph G and pairs of vertices (s_1, t_1), $\ldots$, (s_k, t_k), find pairwise vertex-disjoint paths P_1, $\ldots$, P_k such that P_i connects s_i and t_i.



Disjoint paths

k-Disjoint Paths
Given a graph G and pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$, find pairwise vertex-disjoint paths $P_1, \ldots, P_k$ such that P_i connects s_i and t_i.

- NP-hard, but FPT parameterized by k: can be solved in time $f(k)n^3$ for some horrible function $f(k)$ [Robertson and Seymour].
- More "efficient" algorithm where $f(k)$ is only quadruple exponential [Kawarabayashi and Wollan 2010].
- The Polynomial Excluded Grid Theorem improves this to triple exponential [Chekuri and Chuzhoy 2014].
- Double-exponential is possible on planar graphs [Adler et al. 2011].

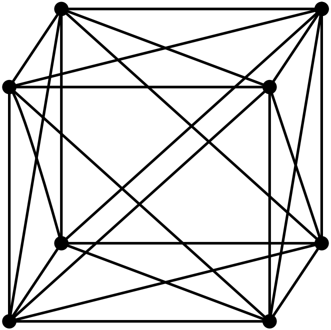
Open: can we have a $2^{k^{O(1)}} \cdot n^{O(1)}$ time algorithm?

Edge Clique Cover

Edge Clique Cover: Given a graph G and an integer k, cover the edges of G with at most k cliques.

(the cliques need not be edge disjoint)

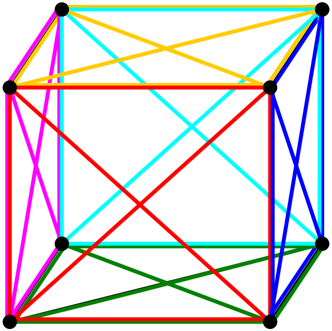
Equivalently: can G be represented as an intersection graph over a k element universe?



Edge Clique Cover

> **Edge Clique Cover**: Given a graph G and an integer k, cover the edges of G with at most k cliques.
>
> (the cliques need not be edge disjoint)

Equivalently: can G be represented as an intersection graph over a k element universe?



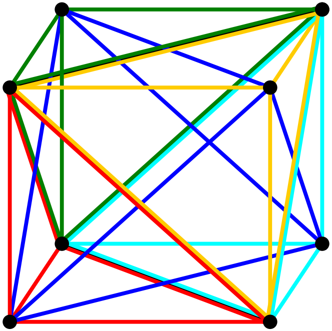
6 cliques

Edge Clique Cover

Edge Clique Cover: Given a graph G and an integer k, cover the edges of G with at most k cliques.

(the cliques need not be edge disjoint)

Equivalently: can G be represented as an intersection graph over a k element universe?



5 cliques

Edge Clique Cover

Edge Clique Cover: Given a graph G and an integer k, cover the edges of G with at most k cliques.

(the cliques need not be edge disjoint)

Simple algorithm (sketch)

- If two adjacent vertices have the same neighborhood ("twins"), then remove one of them.
- If there are no twins and isolated vertices, then $|V(G)| > 2^k$ implies that there is no solution.
- Use brute force.

Running time: $2^{2^{O(k)}} \cdot n^{O(1)}$ — double exponential dependence on k!

Edge Clique Cover

Edge Clique Cover: Given a graph G and an integer k, cover the edges of G with at most k cliques.

(the cliques need not be edge disjoint)

Double-exponential dependence on k cannot be avoided!

Theorem [Cygan, Pilipczuk, Pilipczuk 2013]

Assuming ETH, there is no $2^{2^{o(k)}} \cdot n^{O(1)}$ time algorithm for Edge Clique Cover.

Proof: Reduce an n-variable 3SAT instance into an instance of Edge Clique Cover with $k = O(\log n)$.

Slightly superexponential algorithms

Running time of the form $2^{O(k \log k)} \cdot n^{O(1)}$ appear naturally in parameterized algorithms usually because of one of two reasons:

1. Branching into k directions at most k times explores a search tree of size $k^k = 2^{O(k \log k)}$.
2. Trying $k! = 2^{O(k \log k)}$ permutations of k elements (or partitions, matchings, ...)

Can we avoid these steps and obtain $2^{O(k)} \cdot n^{O(1)}$ time algorithms?

Slightly superexponential algorithms

The improvement to $2^{O(k)}$ often required significant new ideas:

k-Path:

$2^{O(k \log k)} \cdot n^{O(1)}$ using **representative sets** [Monien 1985]

$\Downarrow$

$2^{O(k)} \cdot n^{O(1)}$ using **color coding** [Alon, Yuster, Zwick 1995]

Feedback Vertex Set:

$2^{O(k \log k)} \cdot n^{O(1)}$ using k-**way branching** [Downey and Fellows 1995]

$\Downarrow$

$2^{O(k)} \cdot n^{O(1)}$ using **iterative compression** [Guo et al. 2005]

Planar Subgraph Isomorphism:

$2^{O(k \log k)} \cdot n^{O(1)}$ using **tree decompositions** [Eppstein et al. 1995]

$\Downarrow$

$2^{O(k)} \cdot n^{O(1)}$ using **sphere cut decompositions** [Dorn 2010]

Closest String

> **Closest String**
> Given strings $s_1, \ldots, s_k$ of length L over alphabet Σ, and an integer d, find a string s (of length L) such that Hamming distance $d(s, s_i) \leq d$ for every $1 \leq i \leq k$.

s_1	C	B	D	C	C	A	C	B	B
s_2	A	B	D	B	C	A	B	D	B
s_3	C	D	D	B	A	C	C	B	D
s_4	D	D	A	B	A	C	C	B	D
s_5	A	C	D	B	D	D	C	B	C

Closest String

Closest String

Given strings $s_1, \ldots, s_k$ of length L over alphabet Σ, and an integer d, find a string s (of length L) such that Hamming distance $d(s, s_i) \leq d$ for every $1 \leq i \leq k$.

s_1	C	B	D	C	C	A	C	B	B
s_2	A	B	D	B	C	A	B	D	B
s_3	C	D	D	B	A	C	C	B	D
s_4	D	D	A	B	A	C	C	B	D
s_5	A	C	D	B	D	D	C	B	C
	A	D	D	B	C	A	C	B	D

Closest String

Closest String

Given strings $s_1, \ldots, s_k$ of length L over alphabet Σ, and an integer d, find a string s (of length L) such that Hamming distance $d(s, s_i) \leq d$ for every $1 \leq i \leq k$.

s_1	C	B	D	C	C	A	C	B	B
s_2	A	B	D	B	C	A	B	D	B
s_3	C	D	D	B	A	C	C	B	D
s_4	D	D	A	B	A	C	C	B	D
s_5	A	C	D	B	D	D	C	B	C
	A	D	D	B	C	A	C	B	D

Theorem [Gramm, Niedermeier, Rossmanith 2003]

Closest String can be solved in time $2^{O(d \log d)} \cdot n^{O(1)}$.

Closest String

Closest String

Given strings $s_1, \ldots, s_k$ of length L over alphabet Σ, and an integer d, find a string s (of length L) such that Hamming distance $d(s, s_i) \leq d$ for every $1 \leq i \leq k$.

s_1	C	B	D	C	C	A	C	B	B
s_2	A	B	D	B	C	A	B	D	B
s_3	C	D	D	B	A	C	C	B	D
s_4	D	D	A	B	A	C	C	B	D
s_5	A	C	D	B	D	D	C	B	C
	A	D	D	B	C	A	C	B	D

Theorem [Gramm, Niedermeier, Rossmanith 2003]

Closest String can be solved in time $2^{O(d \log d)} \cdot n^{O(1)}$.

Theorem [Lokshtanov, M., Saurabh 2011]

Assuming ETH, Closest String has no $2^{o(d \log d)} n^{O(1)}$ algorithm.

Slightly superexponential problems

DISTORTION Given a graph G and an integer d, find an embedding $g : V(G) \to \mathbb{Z}$ such that $\mathrm{dist}_G(u,v) \le \lvert g(u) - g(v) \rvert \le d \cdot \mathrm{dist}_G(u,v)$.

- DISTORTION can be solved in time $2^{O(d \log d)} \cdot n^{O(1)}$ [Fellows et al. 2013] ...
- ... but, assuming ETH, cannot be solved in time $2^{o(d \log d)} \cdot n^{O(1)}$ [Lokshtanov, M., Saurabh 2011].

Slightly superexponential problems

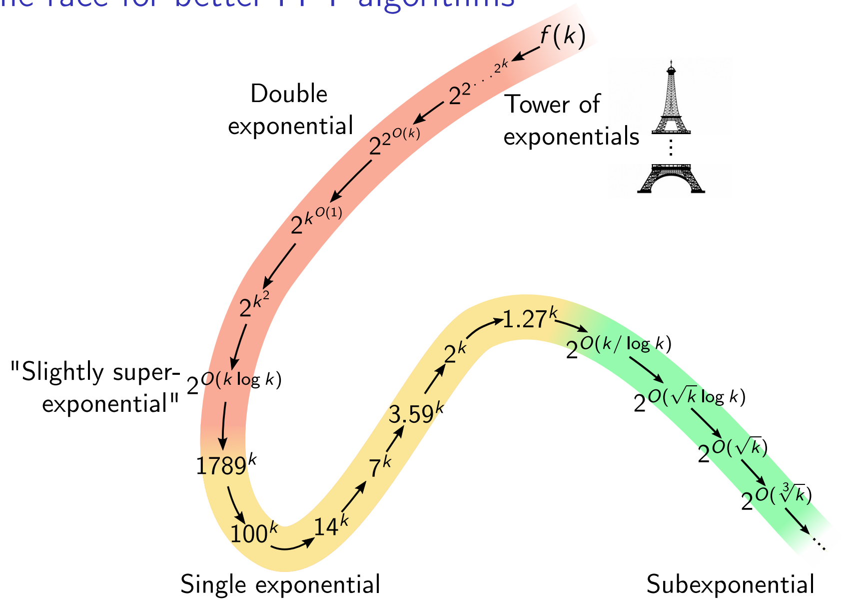
Distortion
Given a graph G and an integer d, find an embedding $g : V(G) \to \mathbb{Z}$ such that $\text{dist}_G(u,v) \le |g(u) - g(v)| \le d \cdot \text{dist}_G(u,v)$.

- Distortion can be solved in time $2^{O(d \log d)} \cdot n^{O(1)}$ [Fellows et al. 2013] ...
- ... but, assuming ETH, cannot be solved in time $2^{o(d \log d)} \cdot n^{O(1)}$ [Lokshtanov, M., Saurabh 2011].

Directed Feedback Vertex Set
Given a graph G and an integer k, find a set S of k vertices such that $G - S$ has no directed cycle.

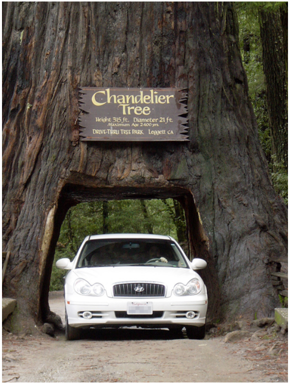
- Directed Feedback Vertex Set can be solved in time $2^{O(k \log k)} \cdot n^{O(1)}$ [Chen et al. 2008].
- **Open question:** Is there a $2^{o(k \log k)} \cdot n^{O(1)}$ time algorithm?

The race for better FPT algorithms



Treewidth

- Treewidth is a measure of "tree-likeness."
- Dynamic programming algorithms for trees can be often generalized to bounded-treewidth graphs.
- These algorithms formalize the concept of "solving the problem recursively on small separators."
- Treewidth pops up in unexpected places, e.g., in algorithms for planar graphs.



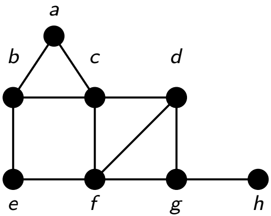
Treewidth

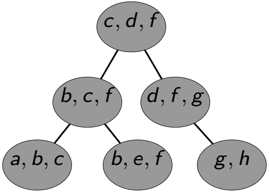
Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.





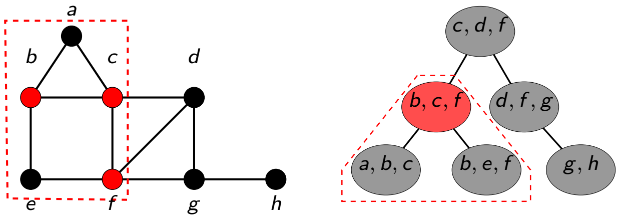
Treewidth

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

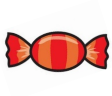
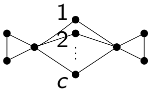
treewidth: width of the best decomposition.



A subtree communicates with the outside world only via the root of the subtree.

Optimal algorithms for tree decompositions

Assuming ETH, these running times are best possible:

Maximum Independent Set	$2^{O(w)}$
Hamiltonian Cycle Cut & Count [Cygan et al. 2011]	$\cancel{2^{O(w \log w)}}$ $2^{O(w)}$
Chromatic Number [Lokshtanov et al. 2011]	$2^{O(w \log w)}$
Hitting Candy Graphs H_c :   [Cygan et al. 2014]	$2^{O(w^c)}$
3-Choosability [M. and Mitsou 2016]	$2^{2^{O(w)}}$
3-Choosability Deletion [M. and Mitsou 2016]	$2^{2^{2^{O(w)}}}$

Best possible bases

Algorithms given a tree decomposition of width w:

Independent Set	2^w
Dominating Set	3^w
c-Coloring	c^w
Odd Cycle Transversal	3^w
Partition into Triangles	2^w
Max Cut	2^w
#Perfect Matching	2^w

Are these constants best possible?

Can we improve 2 to 1.99?

Best possible bases

We need a new complexity assumption:

Strong Exponential-Time Hypothesis (SETH) [consequence of]

There is no $(2-\epsilon)^n$ time algorithm for n-variable CNF-SAT for any $\epsilon > 0$.

Best possible bases

We need a new complexity assumption:

Strong Exponential-Time Hypothesis (SETH) [consequence of]

There is no $(2-\epsilon)^n$ time algorithm for n-variable CNF-SAT for any $\epsilon > 0$.

Assuming SETH...

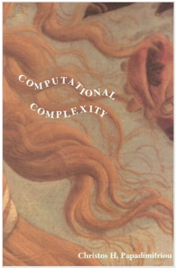
Independent Set	no $(2-\epsilon)^w$
Dominating Set	no $(3-\epsilon)^w$
c-Coloring	no $(c-\epsilon)^w$
Odd Cycle Transversal	no $(3-\epsilon)^w$
Partition into Triangles	no $(2-\epsilon)^w$
Max Cut	no $(2-\epsilon)^w$
#Perfect Matching	no $(2-\epsilon)^w$

Strength of the evidence?

Christos H. Papadimitriou

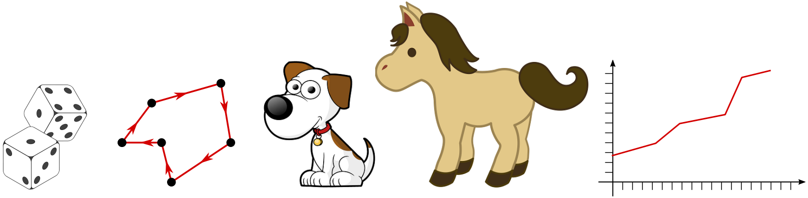
Computational Complexity

Addison-Wesley 1994



“ There is nothing wrong with trying to prove that **P** = **NP** by developing a polynomial-time algorithm for an **NP**-complete problem. The point is that without an **NP**-completeness proof we would be trying the same thing without knowing it! ”

Strength of the evidence?



- Suppose that Stochastic Traveling Dog and Pony Problem with Piecewise Linear Costs is NP-hard.
- There is nothing wrong with trying to prove P = NP by trying to give a polynomial-time algorithm for this problem.
- But at least you should be aware that this is what you are trying to do…
- …and then ask if this is really the most promising approach for proving P = NP.

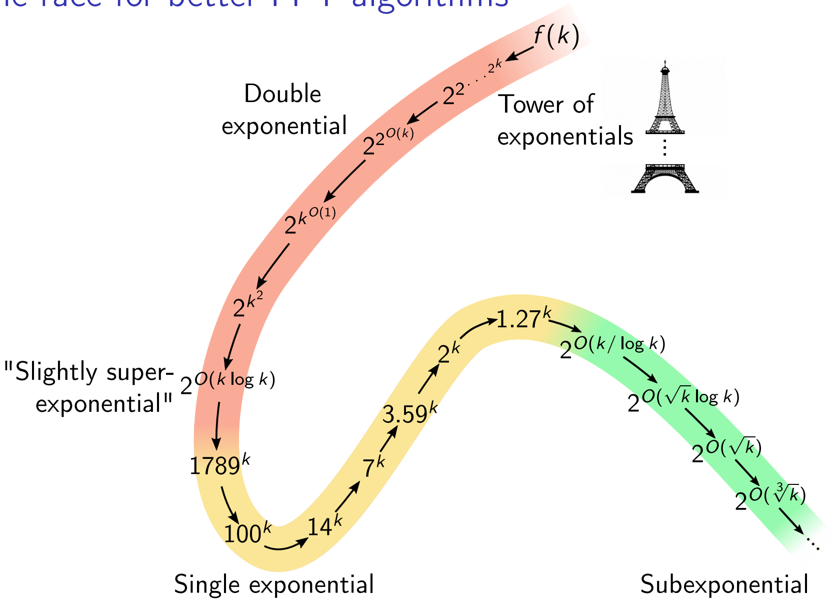
Strength of the evidence?

Theorem

Assuming SETH, there is no $(3-\epsilon)^w \cdot n^{O(1)}$ algorithm for Dominating Set on a tree decomposition of width w.

- There is nothing wrong with trying to refute SETH by trying to give a $2.99^w \cdot n^{O(1)}$ time algorithm Dominating Set.
- But at least you should be aware that this is what you are trying to do...
- ...and then ask if this is really the most promising approach for refuting SETH.

The race for better FPT algorithms



Subexponential parameterized algorithms

There are two main domains where subexponential parameterized algorithms appear:

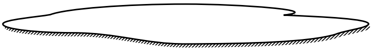
1. Some graph modification problems:
   - Chordal Completion [Fomin and Villanger 2013]
   - Interval Completion [Bliznets et al. 2016]
   - Unit Interval Completion [Bliznets et al. 2015]
   - Feedback Arc Set in Tournaments [Alon et al. 2009]

Subexponential parameterized algorithms

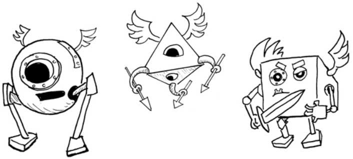
There are two main domains where subexponential parameterized algorithms appear:

1. Some graph modification problems:
   - Chordal Completion [Fomin and Villanger 2013]
   - Interval Completion [Bliznets et al. 2016]
   - Unit Interval Completion [Bliznets et al. 2015]
   - Feedback Arc Set in Tournaments [Alon et al. 2009]
2. "Square root phenomenon" for planar graphs and geometric objects: most NP-hard problems are easier and usually exactly by a square root factor.

Planar graphs



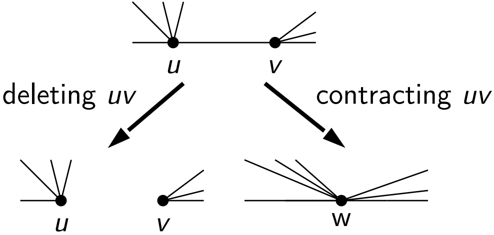
Geometric objects



Minors

Definition

Graph H is a **minor** of G ($H \leq G$) if H can be obtained from G by deleting edges, deleting vertices, and contracting edges.



Note: length of the longest path in H is at most the length of the longest path in G.

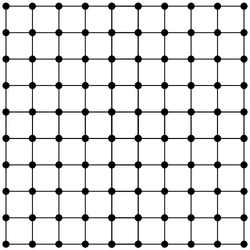
Minors

Definition

Graph H is a **minor** of G ($H \leq G$) if H can be obtained from G by deleting edges, deleting vertices, and contracting edges.

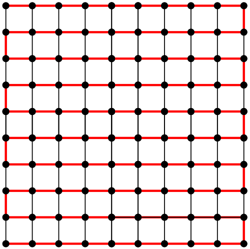
Theorem [Robertson, Seymour, Thomas 1994]

Every planar graph with treewidth at least $5k$ has a $k \times k$ grid minor.



Bidimensionality for k-Path

Observation: If the treewidth of a planar graph G is at least $5\sqrt{k}$
$\Rightarrow$ It has a $\sqrt{k} \times \sqrt{k}$ grid minor (Planar Excluded Grid Theorem)
$\Rightarrow$ The grid has a path of length at least k.
$\Rightarrow$ G has a path of length at least k.

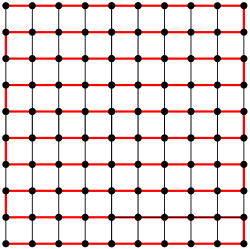


Bidimensionality for k-Path

Observation: If the treewidth of a planar graph G is at least $5\sqrt{k}$
$\Rightarrow$ It has a $\sqrt{k} \times \sqrt{k}$ grid minor (Planar Excluded Grid Theorem)
$\Rightarrow$ The grid has a path of length at least k.
$\Rightarrow$ G has a path of length at least k.

Win/Win approach for finding a path of length k in planar graphs:

- If treewidth w of G is at least $5\sqrt{k}$: we answer "there is a path of length at least k."
- If treewidth w of G is less than $5\sqrt{k}$, then we can solve the problem in time $2^{O(w)} \cdot n^{O(1)} = 2^{O(\sqrt{k})} \cdot n^{O(1)}$.



Shift of focus

qualitative question

FPT or W[1]-hard?

FPT

W[1]-hard

quantitative question

What is the best possible multiplier $f(k)$ in the running time $f(k) \cdot n^{O(1)}$?

What is the best possible exponent $g(k)$ in the running time $f(k) \cdot n^{g(k)}$?

2^k? 1.0001^k? $2^{\sqrt{k}}$?

$n^{O(k)}$? $n^{\log k}$? $n^{\log \log k}$?

Better algorithms for W[1]-hard problems

- $O(n^k)$ algorithm for k-Clique by brute force.
- $O(n^{0.79k})$ algorithms using fast matrix multiplication.
- W[1]-hardness of k-Clique gives evidence that there is no $f(k) \cdot n^{O(1)}$ time algorithm.
- But what about improvements of the exponent $O(k)$?

$n^{\sqrt{k}}$

$n^{k/\log\log k}$

$n^{\log k}$

$n^{\sqrt{k}}$

$2^{2^k} \cdot n^{\log\log\log k}$

Better algorithms for W[1]-hard problems

- $O(n^k)$ algorithm for k-CLIQUE by brute force.
- $O(n^{0.79k})$ algorithms using fast matrix multiplication.
- W[1]-hardness of k-CLIQUE gives evidence that there is no $f(k) \cdot n^{O(1)}$ time algorithm.
- But what about improvements of the exponent $O(k)$?

Theorem [Chen et al. 2004]

Assuming ETH, k-CLIQUE has no $f(k) \cdot n^{o(k)}$ time algorithm for any computable function f.

Better algorithms for W[1]-hard problems

- $O(n^k)$ algorithm for Dominating Set by brute force.
- W[1]-hardness of Dominating Set gives evidence that there is no $f(k) \cdot n^{O(1)}$ time algorithm.
- But what about improvements of the exponent $O(k)$?

$n^{\sqrt{k}}$

$n^{k/\log\log k}$

$n^{0.01k}$

$2^{2^k} \cdot n^{0.99k}$

$n^{\log\log\log k}$

Better algorithms for W[1]-hard problems

- $O(n^k)$ algorithm for Dominating Set by brute force.
- W[1]-hardness of Dominating Set gives evidence that there is no $f(k) \cdot n^{O(1)}$ time algorithm.
- But what about improvements of the exponent $O(k)$?

Theorem [Pătrașcu and Williams 2010]

Assuming SETH, Dominating Set has no $f(k) \cdot n^{k-\epsilon}$ time algorithm for any $\epsilon > 0$ and computable function f.

What did we learn, Palmer?

- Asking quantitative questions instead of FPT vs. W[1]-hard reveals a rich complexity landscape of parameterized problems.
- Conditional hardness results based on ETH and SETH.
- Algorithm design and computational complexity have healthy influence on each other: optimality program needs both.