

The Optimality Program in Parameterized Algorithms

Dániel Marx

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

Highlights of Algorithms
Paris, France
June 6, 2016

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.
- ...

Parameterized complexity

Problem:

VERTEX COVER

INDEPENDENT SET

Input:

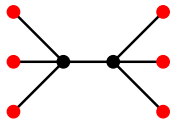
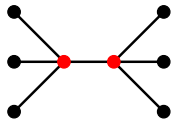
Graph G , integer k

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:

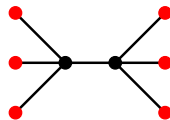
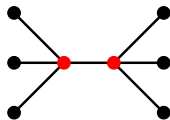
Graph G , integer k

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:

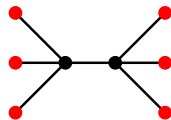
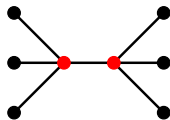
Graph G , integer k

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

No $n^{o(k)}$ algorithm

exists 😊

known 😞

Bounded search tree method

Algorithm for VERTEX COVER:

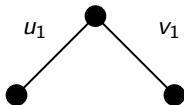
$$e_1 = u_1 v_1$$



Bounded search tree method

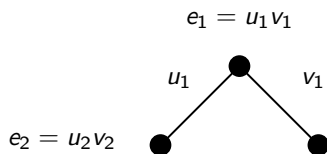
Algorithm for VERTEX COVER:

$$e_1 = u_1 v_1$$



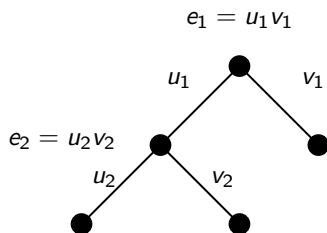
Bounded search tree method

Algorithm for VERTEX COVER:



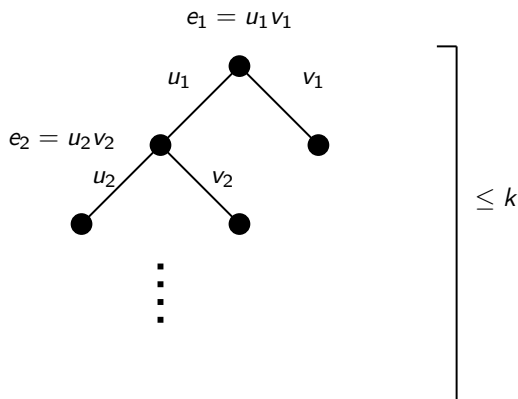
Bounded search tree method

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

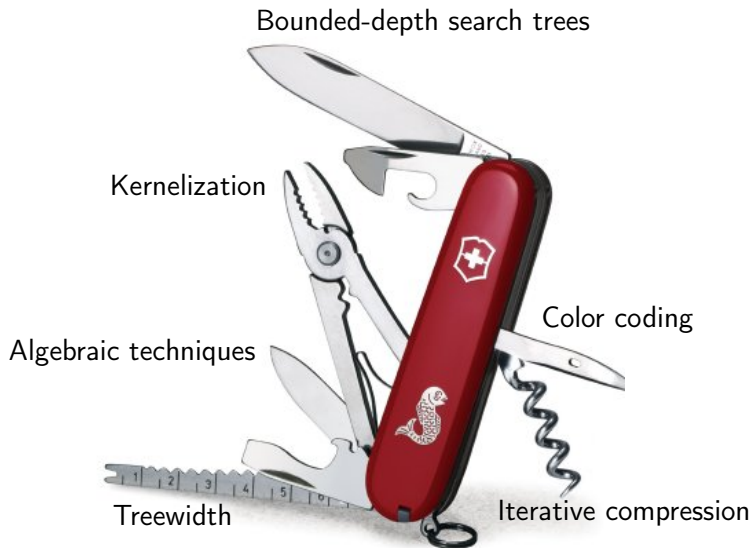
Main definition

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.
- ...

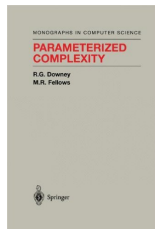
Parameterized complexity



Rod G. Downey
Michael R. Fellows

Parameterized Complexity

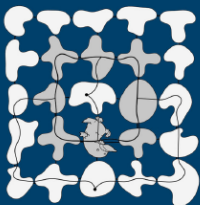
Springer 1999



- 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.

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

Parameterized Algorithms



 Springer

Parameterized Algorithms

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

Springer 2015

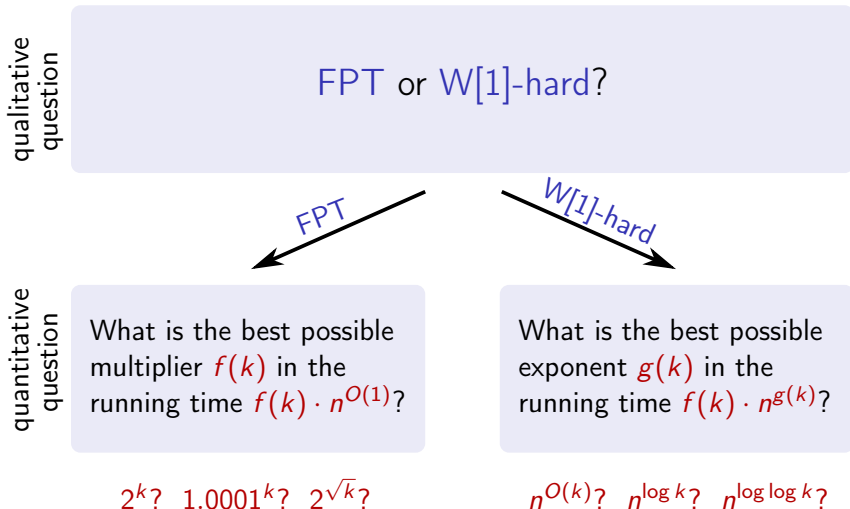


Shift of focus

qualitative
question

FPT or $W[1]$ -hard?

Shift of focus



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 3SAT.

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

Note: an n -variable 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 3SAT.

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

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

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

Sparsification Lemma [Impagliazzo, Paturi, Zane 2001]

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



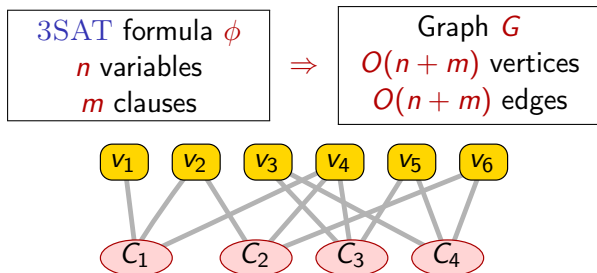
There is a $2^{o(n+m)}$ -time algorithm for n -variable m -clause 3SAT.

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 3-COLORING:



Corollary

Assuming ETH, there is no $2^{o(n)}$ algorithm for 3-COLORING on an n -vertex graph G .

Other problems

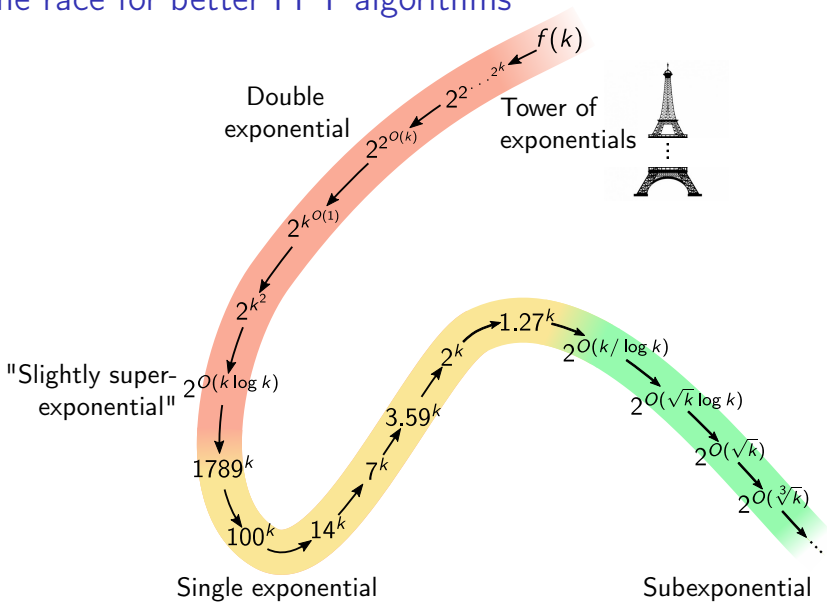
There are polytime reductions from 3SAT to many problems such that the reduction creates a graph with $O(n + m)$ vertices/edges.

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

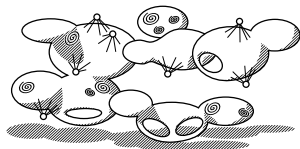


Paul Seymour

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.

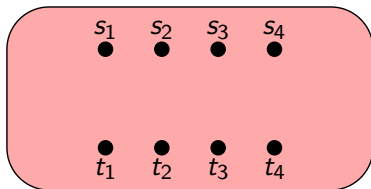


[figure by Felix Reid]

Disjoint paths

k -DISJOINT PATHS

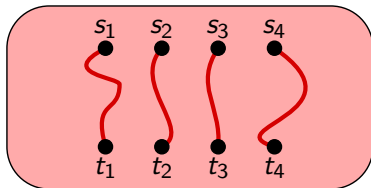
Given a graph G and pairs of vertices $(s_1, t_1), \dots, (s_k, t_k)$, find pairwise vertex-disjoint paths P_1, \dots, P_k such that P_i connects s_i and t_i .



Disjoint paths

k -DISJOINT PATHS

Given a graph G and pairs of vertices $(s_1, t_1), \dots, (s_k, t_k)$, find pairwise vertex-disjoint paths P_1, \dots, 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 $(s_1, t_1), \dots, (s_k, t_k)$, find pairwise vertex-disjoint paths P_1, \dots, 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].

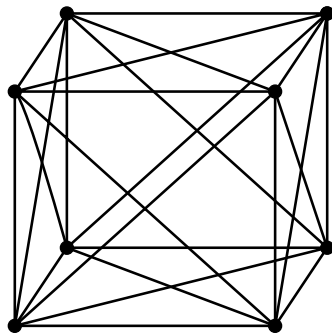
Open: can we have a $2^{k^{O(1)}} \cdot n^{O(1)}$ time algorithm?

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?

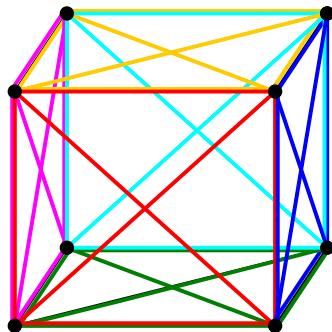


EDGE CLIQUE COVER

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?



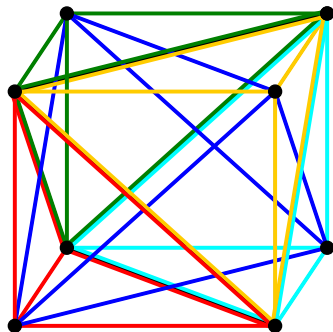
6 cliques

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]



$2^{O(k)} \cdot n^{O(1)}$ 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]



$2^{O(k)} \cdot n^{O(1)}$ 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]



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

CLOSEST STRING

CLOSEST STRING

Given strings s_1, \dots, 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, \dots, 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, \dots, 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, \dots, 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

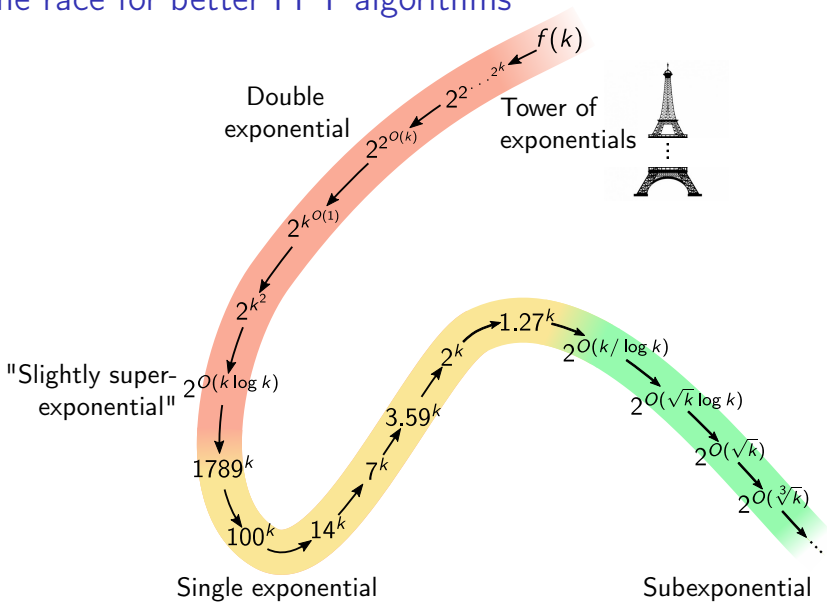
Theorem [Gramm, Niedermeier, Rossmanith 2003]

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.

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.



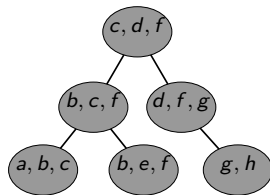
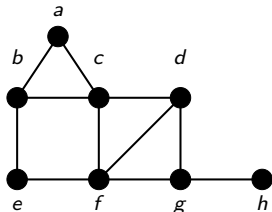
Treewidth

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.



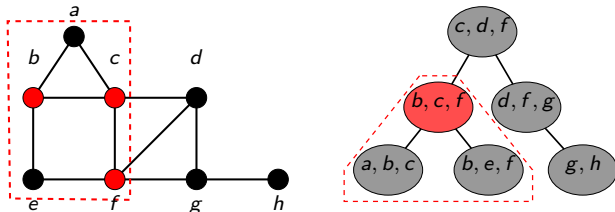
Treewidth

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 .

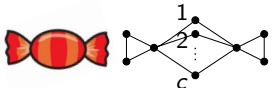
treewidth: width of the best decomposition.



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	$2^{O(w \log w)}$
Cut & Count [Cygan et al. 2011]	$2^{O(w)}$
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]	$2^{2^{2^{O(w)}}}$

(see Valia Mitsou's talk Tue 14:10B)

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...

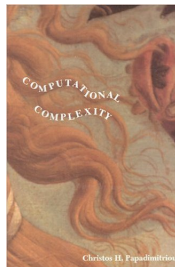
INDEPENDENT SET	no $(2 - \epsilon)^w$
DOMINATING SET	no $(3 - \epsilon)^w$
c -COLORING	no $(c - \epsilon)^w$
ODD CYCLE TRANSVERSAL	no $(3 - \epsilon)^w$
PARTITION INTO TRIANGLES	no $(2 - \epsilon)^w$
MAX CUT	no $(2 - \epsilon)^w$
#PERFECT MATCHING	no $(2 - \epsilon)^w$

Strength of the evidence?



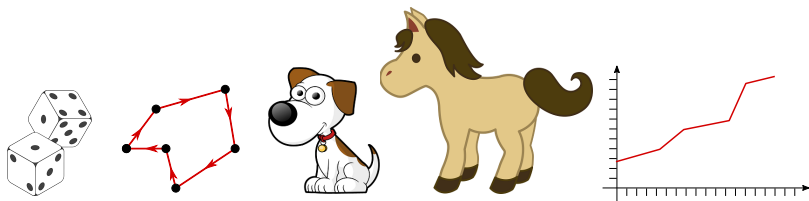
Christos H. Papadimitriou Computational Complexity

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**.

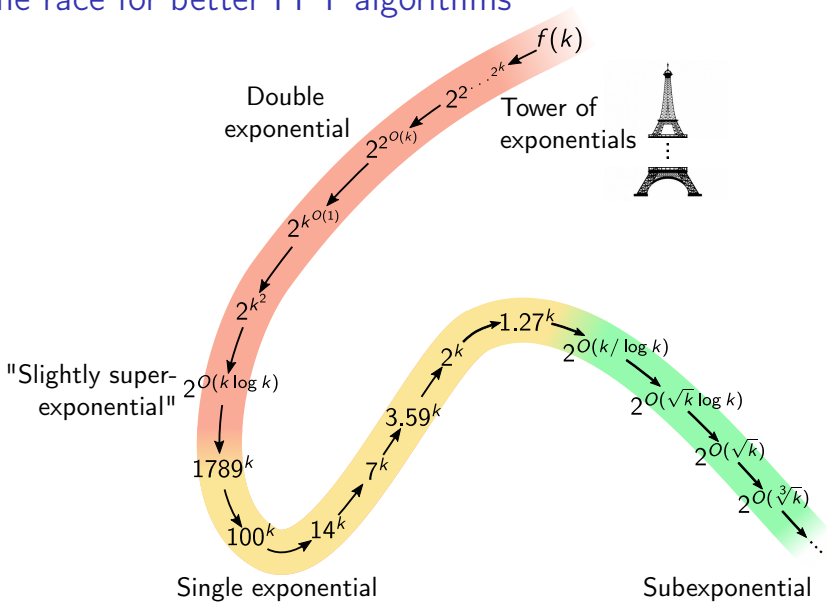
Strength of the evidence?

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

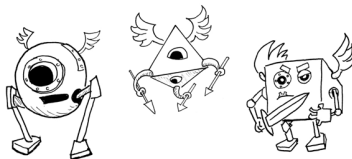
There are two main domains where subexponential parameterized algorithms appear:

- 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



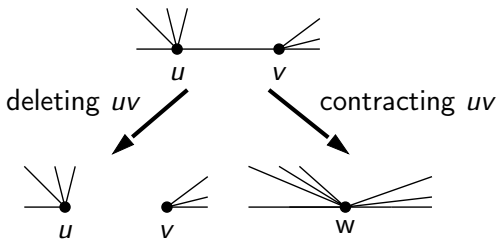
Geometric objects



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 .

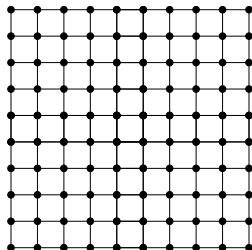
Minors

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.

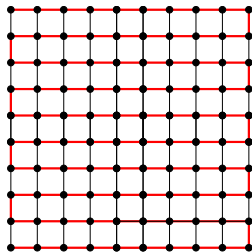
Theorem [Robertson, Seymour, Thomas 1994]

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 .

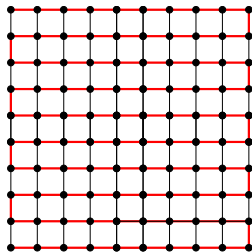


Bidimensionality for k -PATH

- 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 $f(k) \cdot n^{O(1)}$?

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

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 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 $O(k)$?

$$\begin{array}{l} n^{\sqrt{k}} \\ n^{\log k} \quad n^{k/\log \log k} \\ 2^{2^k} \cdot n^{\log \log \log k} \quad n^{\sqrt{k}} \end{array}$$

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 $O(k)$?

$$\begin{array}{l} n^{\sqrt{k}} \\ n^{0.01k} \end{array} \quad \begin{array}{l} n^{k/\log \log k} \\ 2^{2^k} \cdot n^{0.99k} \\ n^{\log \log \log k} \end{array}$$

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 $O(k)$?



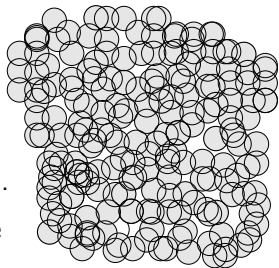
Theorem [Pătraşcu and Williams 2010]

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

Better algorithms for $W[1]$ -hard problems

Square root phenomenon:

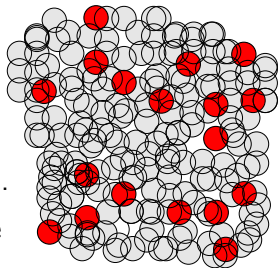
- **INDEPENDENT SET** for unit disks:
select k pairwise nonintersecting disks.
- $n^{O(k)}$ algorithm by brute force.
- $n^{O(\sqrt{k})}$ time algorithms exist
[Alber and Fiala 2004], [M. and Pilipczuk 2015].
- Assuming ETH, there is no $f(k)n^{o(\sqrt{k})}$ time
algorithm for any function f .



Better algorithms for $W[1]$ -hard problems

Square root phenomenon:

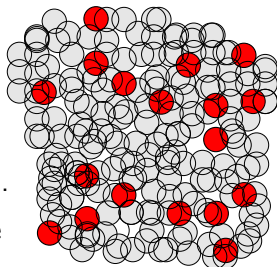
- **INDEPENDENT SET** for unit disks:
select k pairwise nonintersecting disks.
- $n^{O(k)}$ algorithm by brute force.
- $n^{O(\sqrt{k})}$ time algorithms exist
[Alber and Fiala 2004], [M. and Pilipczuk 2015].
- Assuming ETH, there is no $f(k)n^{o(\sqrt{k})}$ time
algorithm for any function f .



Better algorithms for $W[1]$ -hard problems

Square root phenomenon:

- **INDEPENDENT SET** for unit disks:
select k pairwise nonintersecting disks.
- $n^{O(k)}$ algorithm by brute force.
- $n^{O(\sqrt{k})}$ time algorithms exist
[Alber and Fiala 2004], [M. and Pilipczuk 2015].
- Assuming ETH, there is no $f(k)n^{o(\sqrt{k})}$ time
algorithm for any function f .



Limited blessing of low dimensionality:

Generalization to d -dimensional unit balls:

- $n^{O(k^{1-1/d})}$ time algorithm.
- Assuming ETH, there is no $f(k)n^{O(k^{1-1/d-\epsilon})}$ time algorithm
for any $\epsilon > 0$ and function f . [Sidiropoulos and M. 2014]

CLOSEST SUBSTRING

CLOSEST SUBSTRING

Given strings s_1, \dots, s_k over alphabet Σ , and integers d and L , find a string s of length L such that each s_i has a consecutive substring of length L at Hamming distance at most d from s .

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

CLOSEST SUBSTRING

CLOSEST SUBSTRING

Given strings s_1, \dots, s_k over alphabet Σ , and integers d and L , find a string s of length L such that each s_i has a consecutive substring of length L at Hamming distance at most d from s .

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

CLOSEST SUBSTRING

CLOSEST SUBSTRING

Given strings s_1, \dots, s_k over alphabet Σ , and integers d and L , find a string s of length L such that each s_i has a consecutive substring of length L at Hamming distance at most d from s .

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 SUBSTRING

CLOSEST SUBSTRING

Given strings s_1, \dots, s_k over alphabet Σ , and integers d and L , find a string s of length L such that each s_i has a consecutive substring of length L at Hamming distance at most d from s .

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	...
s		A	D	D	B	C	A	C	B	D	

CLOSEST SUBSTRING

CLOSEST SUBSTRING

Given strings s_1, \dots, s_k over alphabet Σ , and integers d and L , find a string s of length L such that each s_i has a consecutive substring of length L at Hamming distance at most d from s .

Parameterization by d :

Theorem [M. 2008]

CLOSEST SUBSTRING over an alphabet of constant size

- can be solved in time $f(d) \cdot n^{O(\log d)}$,
- assuming ETH, cannot be solved in time $f(d) \cdot n^{o(\log d)}$ for any function f .

CLOSEST SUBSTRING

CLOSEST SUBSTRING

Given strings s_1, \dots, s_k over alphabet Σ , and integers d and L , find a string s of length L such that each s_i has a consecutive substring of length L at Hamming distance at most d from s .

Parameterization by d and k :

Theorem [M. 2008]

CLOSEST SUBSTRING over an alphabet of constant size

- can be solved in time $f(d, k) \cdot n^{O(\log \log k)}$,
- assuming ETH, cannot be solved in time $f(d, k) \cdot n^{o(\log \log k)}$ for any 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.

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.

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.
- Reason for strange running times: the size of the search space is restricted by some combinatorial property.

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.
- Reason for strange running times: the size of the search space is restricted by some combinatorial property.
- Algorithm design and computational complexity have healthy influence on each other: optimality program needs both.

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.
- Reason for strange running times: the size of the search space is restricted by some combinatorial property.
- Algorithm design and computational complexity have healthy influence on each other: optimality program needs both.



Think of lower bounds
when designing algorithms

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.
- Reason for strange running times: the size of the search space is restricted by some combinatorial property.
- Algorithm design and computational complexity have healthy influence on each other: optimality program needs both.



Think of lower bounds
when designing algorithms



Think of algorithms
when doing lower bounds