Classical complexity

A brief review:

- We usually aim for **polynomial-time** algorithms: the worst-case running time is $O(n^c)$, where n is the input size and c is a constant.
- Classical polynomial-time algorithms: shortest path, perfect matching, minimum spanning tree, 2SAT, convex hull, planar drawing, linear programming, etc.
- It is unlikely that polynomial-time algorithms exist for **NP-hard** problems.
- Unfortunately, many problems of interest are NP-hard: Hamiltonian Cycle, 3-Coloring, 3SAT, etc.
- We expect that these problems can be solved only in exponential time (i.e., $O(c^n)$).

Can we say anything nontrivial about NP-hard problems?

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

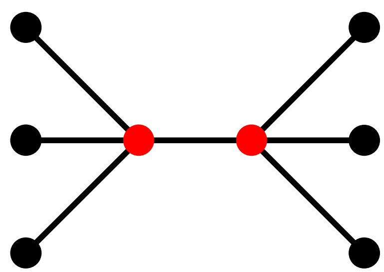
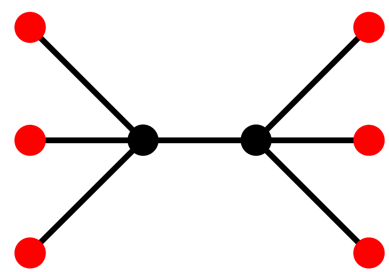
Instead of expressing the running time as a function $T(n)$ of n, we express it as a function $T(n, k)$ of the input size n and some parameter k of the input.

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?

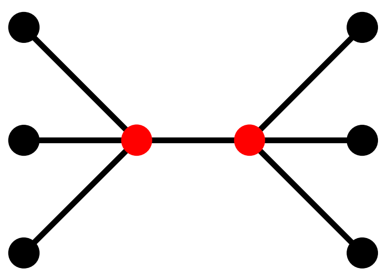
- The size k of the solution we are looking for.
- The maximum degree Δ of the input graph.
- The dimension d of the point set in the input.
- The length L of the strings in the input.
- The length ℓ of clauses in the input Boolean formula.
- ...

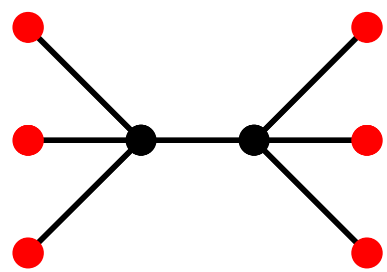
Parameterized complexity

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?

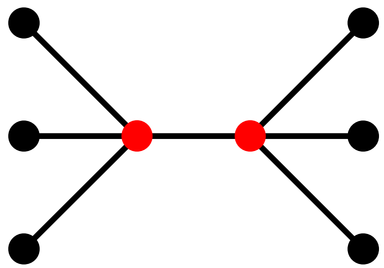


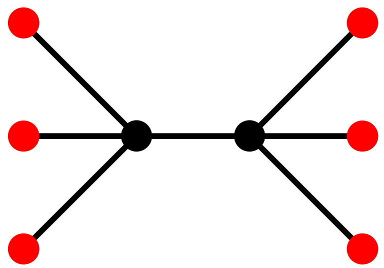


Complexity:	NP-complete	NP-complete
Brute force:	$O(n^k)$ possibilities	$O(n^k)$ possibilities

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 exists**	**No** $n^{o(k)}$ **algorithm known**

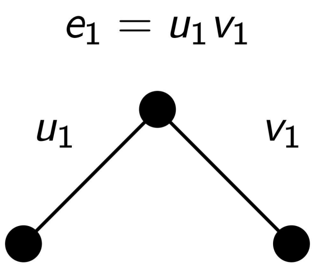
Bounded search tree method

Algorithm for VERTEX COVER:

$e_1 = u_1 v_1$

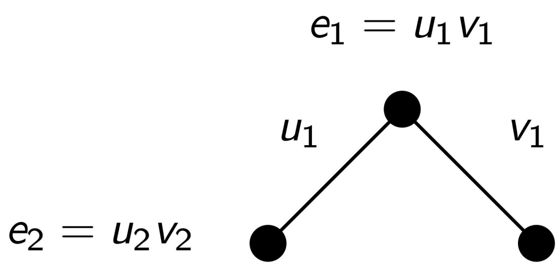
Bounded search tree method

Algorithm for Vertex Cover:



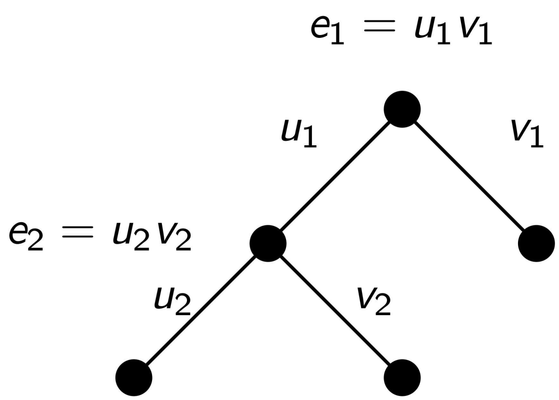
Bounded search tree method

Algorithm for VERTEX COVER:



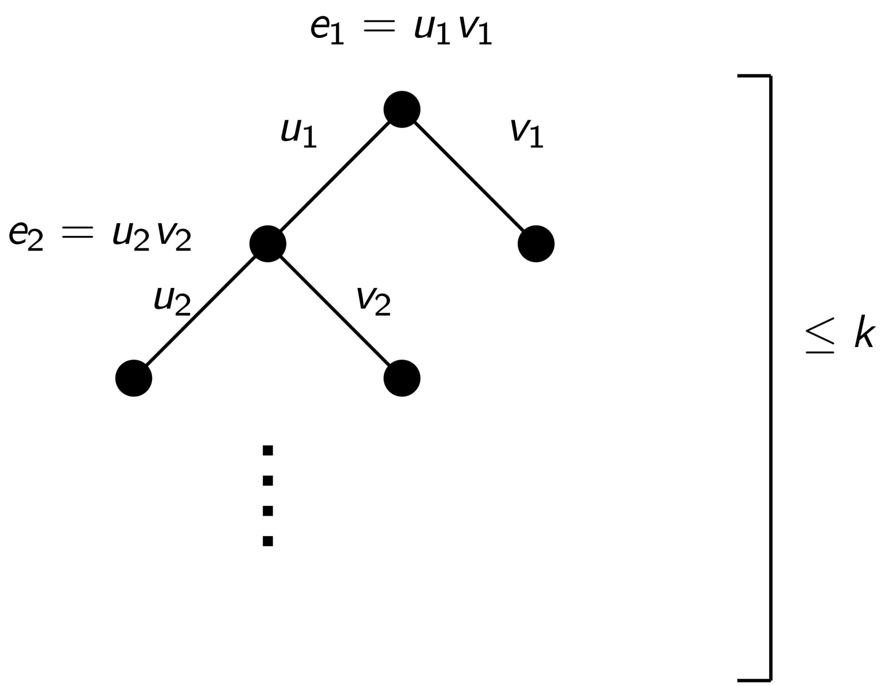
Bounded search tree method

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.
- ...

More formally

- We consider only **decision problems** here.
- Let Σ be a finite alphabet used to encode the inputs
  - ($\Sigma = \{0, 1\}$ for binary encodings)

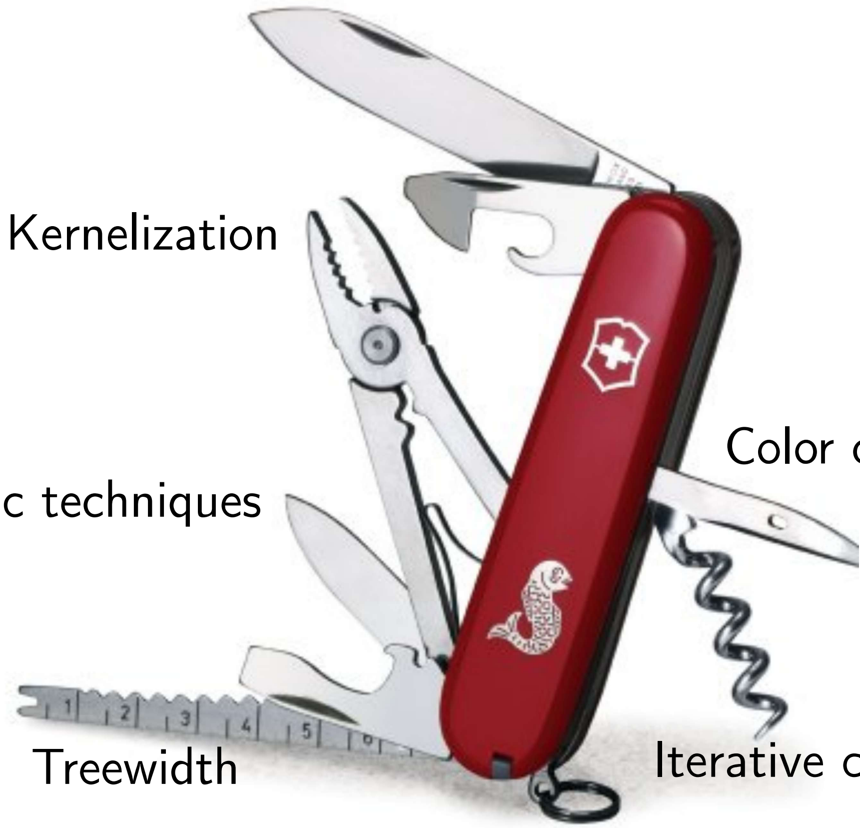
More formally

- We consider only **decision problems** here.
- Let Σ be a finite alphabet used to encode the inputs
  - ($\Sigma = \{0, 1\}$ for binary encodings)
- A **parameterized problem** is a set $P \subseteq \Sigma^* \times \mathbb{N}$
  - $P = \{(x_1, k_1), (x_2, k_2), \dots\}$
- The set P contain the tuples (x, k) where the answer to the question encoded by (x, k) is yes; k is the **parameter**

More formally

- We consider only **decision problems** here.
- Let Σ be a finite alphabet used to encode the inputs
  - ($\Sigma = \{0, 1\}$ for binary encodings)
- A **parameterized problem** is a set $P \subseteq \Sigma^* \times \mathbb{N}$
  - $P = \{(x_1, k_1), (x_2, k_2), \ldots\}$
- The set P contain the tuples (x, k) where the answer to the question encoded by (x, k) is yes; k is the **parameter**
- A parameterized problem P is **fixed-parameter tractable** if there is an algorithm that, given an input (x, k)
  - decides if (x, k) belongs to P or not, and
  - the running time is $f(k)n^c$ for some computable function f and constant c.

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.
- . . .

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].

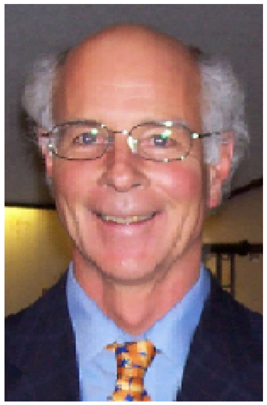
Some W[1]-hard problems:

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

General principle of hardness

With an appropriate **reduction** from k-CLIQUE to problem P, we show that if problem P is FPT, then k-CLIQUE is also FPT.

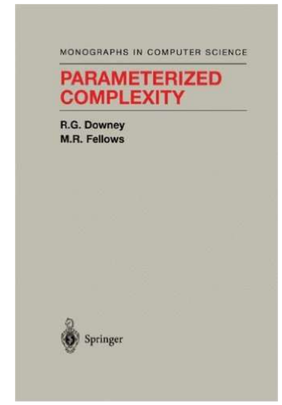
Parameterized complexity



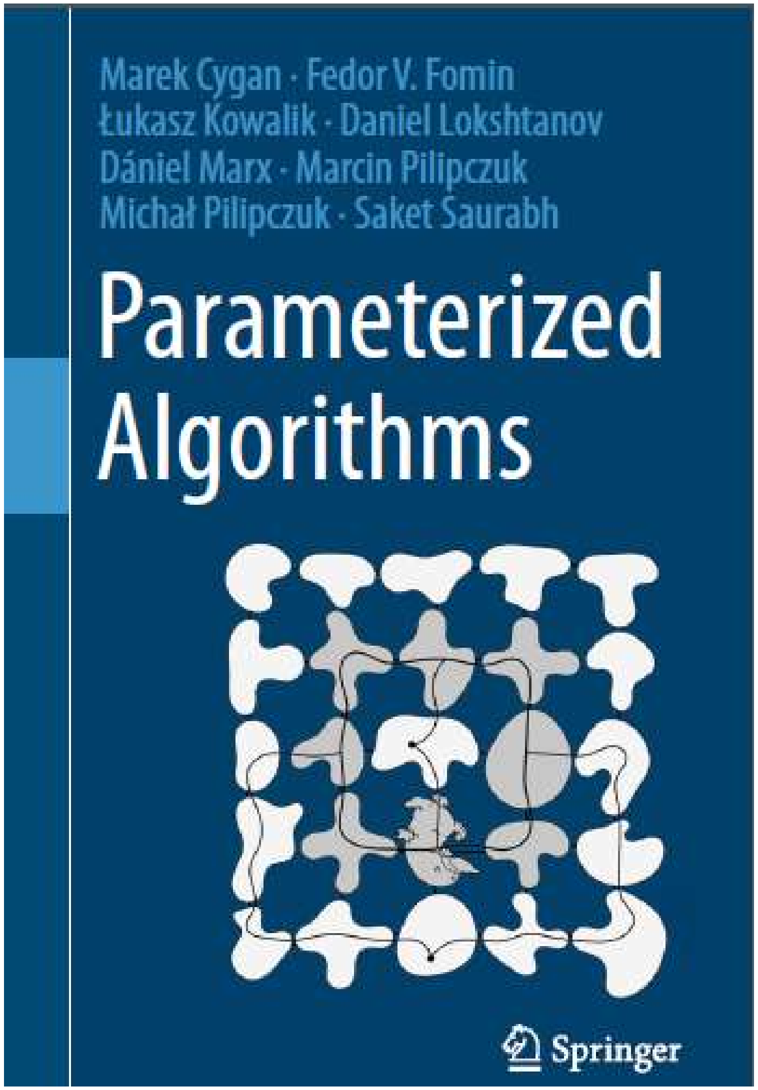
Rod G. Downey
Michael R. Fellows

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