

# Introduction to solving intractable problems

Serge Gaspers

## Contents

1	Algorithms for NP-hard problems	1
2	Exponential Time Algorithms	2
3	Parameterized Complexity	5
3.1	FPT Algorithm for Vertex Cover . . . . .	6
3.2	Algorithms for Vertex Cover . . . . .	6
4	Further Reading	7

## 1 Algorithms for NP-hard problems

Central question

P vs. NP

### NP-hard problems

- no known polynomial time algorithm for any NP-hard problem
- belief:  $P \neq NP$
- What to do when facing an NP-hard problem?

### Example problem

#### Monitoring a power grid

Tammy is responsible for fault detection on the power grid of an energy company. She has access to  $k$  monitoring devices. Each one can be placed on a node of the electrical grid and can monitor the power lines that are connected to this node. Tammy's objective is to place the monitoring devices in such a way that each power line is monitored by at least one monitoring device.

Let us first give an abstraction of this problem and formulate it as a decision problem for graphs.

#### Example problem: Vertex Cover

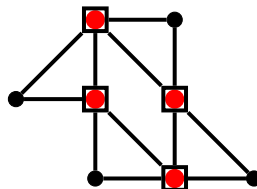
A *vertex cover* in a graph  $G = (V, E)$  is a subset of vertices  $S \subseteq V$  such that every edge of  $G$  has an endpoint in  $S$ .

VERTEX COVER

Input: Graph  $G$ , integer  $k$

Question: Does  $G$  have a vertex cover of size  $k$ ?

**Note:** VERTEX COVER is NP-complete.



## Coping with NP-hardness

- Approximation algorithms
  - There is a polynomial-time algorithm, which, given a graph  $G$ , finds a vertex cover of  $G$  of size at most  $2 \cdot \text{OPT}$ , where  $\text{OPT}$  is the size of a smallest vertex cover of  $G$ .
- Exact exponential time algorithms
  - There is an algorithm solving VERTEX COVER in time  $O(1.1970^n)$ , where  $n = |V|$  (Xiao and Nagamochi, 2017).
- Fixed parameter algorithms
  - There is an algorithm solving VERTEX COVER in time  $O(1.2738^k + kn)$  (Chen, Kanj, and Xia, 2010).
- Heuristics
  - The COVER heuristic (COVER Edges Randomly) finds a smaller vertex cover than state-of-the-art heuristics on a suite of hard benchmark instances (Richter, Helmert, and Gretton, 2007).
- Restricting the inputs
  - VERTEX COVER can be solved in polynomial time on bipartite graphs, trees, interval graphs, etc. (Golumbic, 2004).
- Quantum algorithms?
  - Not believed to solve NP-hard problems in polynomial time (Aaronson, 2005). Quadratic speedup possible in some cases.

## Aims of this course

Design and analyze algorithms for NP-hard problems.

We focus on algorithms that solve NP-hard problems *exactly* and analyze their *worst case running time*.

## 2 Exponential Time Algorithms

### Running times

Worst case running time of an algorithm.

- An algorithm is *polynomial* if  $\exists c \in \mathbb{N}$  such that the algorithm solves every instance in time  $O(n^c)$ , where  $n$  is the size of the instance. Also:  $n^{O(1)}$  or  $\text{poly}(n)$ .
- *quasi-polynomial*:  $2^{O(\log^c n)}$ ,  $c \in O(1)$
- *sub-exponential*:  $2^{o(n)}$
- *exponential*:  $2^{\text{poly}(n)}$
- *double-exponential*:  $2^{2^{\text{poly}(n)}}$

$O^*$ -notation ignores polynomial factors in the input size:

$$\begin{aligned} O^*(f(n)) &\equiv O(f(n) \cdot \text{poly}(n)) \\ O^*(f(k)) &\equiv O(f(k) \cdot \text{poly}(n)) \end{aligned}$$

### Brute-force algorithms for NP-hard problems

**Theorem 1.** *Every problem in NP can be solved in exponential time.*

For a proof, see the lecture on NP-completeness.

### Three main categories for NP-complete problems

- Subset problems
- Permutation problems
- Partition problems

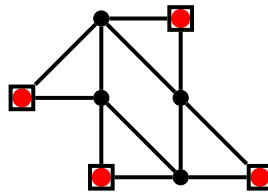
#### Subset Problem: Independent Set

An *independent set* in a graph  $G = (V, E)$  is a subset of vertices  $S \subseteq V$  such that the vertices in  $S$  are pairwise non-adjacent in  $G$ .

##### INDEPENDENT SET

Input: Graph  $G$ , integer  $k$

Question: Does  $G$  have an independent set of size  $k$ ?



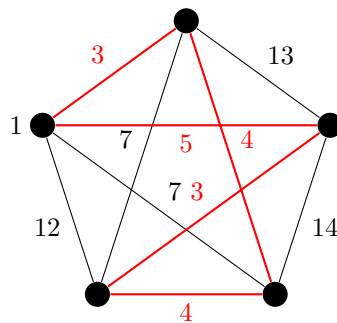
Brute-force:  $O^*(2^n)$ , where  $n = |V(G)|$

#### Permutation Problem: Traveling SalesPerson

##### TRAVELING SALESPERSON (TSP)

Input: a set of  $n$  cities, the distance  $d(i, j) \in \mathbb{N}$  between every two cities  $i$  and  $j$ , integer  $k$

Question: Is there a permutation of the cities (a *tour*) such that the total distance when traveling from city to city in the specified order, and returning back to the origin, is at most  $k$ ?



Brute-force:  $O^*(n!) \subseteq 2^{O(n \log n)}$

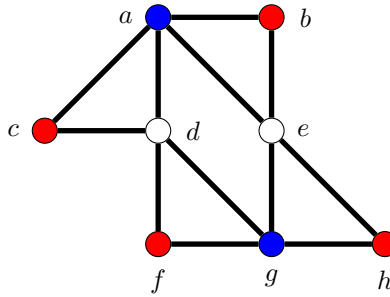
#### Partition Problem: Coloring

A  $k$ -*coloring* of a graph  $G = (V, E)$  is a function  $f : V \rightarrow \{1, 2, \dots, k\}$  assigning colors to  $V$  such that no two adjacent vertices receive the same color.

##### COLORING

Input: Graph  $G$ , integer  $k$

Question: Does  $G$  have a  $k$ -coloring?



Brute-force:  $O^*(k^n)$ , where  $n = |V(G)|$

### Exponential Time Algorithms

- natural question in Algorithms: design faster (worst-case analysis) algorithms for problems
- might lead to practical algorithms
  - for small instances
    - \* you don't want to design software where your client/boss can find with better solutions *by hand* than your software
  - subroutines for
    - \* (sub)exponential time approximation algorithms
    - \* randomized algorithms with expected polynomial run time

### Solve an NP-hard problem

- exhaustive search
  - trivial method
  - try all candidate solutions (certificates) for a ground set on  $n$  elements
  - running times for problems in NP
    - \* SUBSET PROBLEMS:  $O^*(2^n)$
    - \* PERMUTATION PROBLEMS:  $O^*(n!)$
    - \* PARTITION PROBLEMS:  $O^*(c^{n \log n})$
- faster exact algorithms
  - for some problems, it is possible to obtain provably faster algorithms
  - running times  $O(1.0836^n), O(1.4689^n), O(1.9977^n)$

### Exponential Time Algorithms in Practice

- How large are the instances one can solve in practice?

Available time nb. of operations	1 s $2^{38}$	1 min $\sim 2^{44}$	1 hour $\sim 2^{50}$	3 days $\sim 2^{56}$	6 months $\sim 2^{62}$
$n^5$	194	446	1,024	2,352	5,404
$n^{10}$	14	21	32	49	74
$1.05^n$	540	625	711	796	881
$1.1^n$	276	320	364	407	451
$1.5^n$	65	75	85	96	106
$2^n$	38	44	50	56	62
$5^n$	16	19	22	24	27
$n!$	14	16	17	19	20

**Note:** Intel Core i7-8086K executes  $\sim 2^{38}$  instructions per second at 5 GHz.

“For every polynomial-time algorithm you have, there is an exponential algorithm that I would rather run.”

– Alan Perlis (1922-1990, programming languages, 1st recipient of Turing Award)

## Hardware vs. Algorithms

- Suppose a  $2^n$  algorithm enables us to solve instances up to size  $x$
- Faster processors
  - processor speed doubles after 18–24 months (Moore’s law)
  - can solve instances up to size  $x + 1$
- Faster algorithm
  - design an  $O^*(2^{n/2}) \subseteq O(1.4143^n)$  time algorithm
  - can solve instances up to size  $2 \cdot x$

## 3 Parameterized Complexity

### A story

A computer scientist meets a biologist ... The biologist has performed  $n$  experiments. Unfortunately, the data obtained from these experiments has some conflicts. He suspects that a small number  $k$  of experiments have gone wrong, and he would like to detect whether removing  $k$  experiments can solve all the conflicts.

### Eliminating conflicts from experiments

$n = 1000$  experiments,  $k = 20$  experiments failed

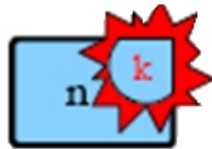
Theoretical	Running Time	
	Number of Instructions	Real
$2^n$	$1.07 \cdot 10^{301}$	$4.941 \cdot 10^{282}$ years
$n^k$	$10^{60}$	$4.611 \cdot 10^{41}$ years
$2^k \cdot n$	$1.05 \cdot 10^9$	0.01526 seconds

### Notes

- We assume that  $2^{36}$  instructions are carried out per second.
- The Big Bang happened roughly  $13.5 \cdot 10^9$  years ago.

### Goal of Parameterized Complexity

Confine the combinatorial explosion to a parameter  $k$ .



For which problem–parameter combinations can we find algorithms with running times of the form

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

where the  $f$  is a computable function independent of the input size  $n$ ?

## Examples of Parameters

### A Parameterized Problem

Input: an instance of the problem  
Parameter: a parameter  $k$   
Question: a YES/NO question about the instance and the parameter

- A parameter can be
  - input size (trivial parameterization)
  - solution size
  - related to the structure of the input (maximum degree, treewidth, branchwidth, genus, ...)
  - etc.

## Main Complexity Classes

P: class of problems that can be solved in time  $n^{O(1)}$

FPT: class of problems that can be solved in time  $f(k) \cdot n^{O(1)}$

W[·]: parameterized intractability classes

XP: class of problems that can be solved in time  $f(k) \cdot n^{g(k)}$

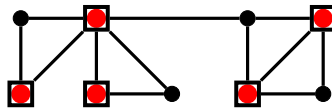
$$P \subseteq \text{FPT} \subseteq W[1] \subseteq W[2] \cdots \subseteq W[P] \subseteq \text{XP}$$

Known: If  $\text{FPT} = W[1]$ , then the Exponential Time Hypothesis fails, i.e. 3-SAT can be solved in time  $2^{o(n)}$ .

## 3.1 FPT Algorithm for Vertex Cover

### VERTEX COVER (VC)

Input: A graph  $G = (V, E)$  on  $n$  vertices, an integer  $k$   
Parameter:  $k$   
Question: Is there a set of vertices  $C \subseteq V$  of size at most  $k$  such that every edge has at least one endpoint in  $C$ ?



## 3.2 Algorithms for Vertex Cover

### Brute Force Algorithms

- $2^n \cdot n^{O(1)}$  not FPT
- $n^k \cdot n^{O(1)}$  not FPT

### An FPT Algorithm

*Algorithm*  $vc1(G, k)$ ;

```
1 if  $E = \emptyset$  then                                     // all edges are covered
2   return Yes
3 else if  $k \leq 0$  then                                    // we cannot select any vertex
4   return No
5 else
6   Select an edge  $uv \in E$ ;
7   return  $vc1(G - u, k - 1) \vee vc1(G - v, k - 1)$ 
```

## Running Time Analysis

- Let us look at an arbitrary execution of the algorithm.
- Recursive calls form a *search tree*  $T$ 
  - with depth  $\leq k$
  - where each node has  $\leq 2$  children
- $\Rightarrow T$  has  $\leq 2^k$  leaves and  $\leq 2^k - 1$  internal nodes
- at each node the algorithm spends time  $n^{O(1)}$
- The running time is  $O^*(2^k)$

## A faster FPT Algorithm

*Algorithm*  $vc2(G, k)$ ;

```
1 if  $E = \emptyset$  then                                     // all edges are covered
2   | return Yes
3 else if  $k \leq 0$  then                                   // we used too many vertices
4   | return No
5 else if  $\Delta(G) \leq 2$  then                             //  $G$  has maximum degree  $\leq 2$ 
6   | Solve the problem in polynomial time;
7 else
8   | Select a vertex  $v$  of maximum degree;
9   | return  $vc2(G - v, k - 1) \vee vc2(G - N[v], k - d(v))$ 
```

## Running time analysis of $vc2$

- Number of leaves of the search tree:

$$\begin{aligned} T(k) &\leq T(k-1) + T(k-3) \\ x^k &\leq x^{k-1} + x^{k-3} \\ x^3 - x^2 - 1 &\leq 0 \end{aligned}$$

- The equation  $x^3 - x^2 - 1 = 0$  has a unique positive real solution:  $x \approx 1.4655\dots$
- Running time:  $1.4656^k \cdot n^{O(1)}$

## 4 Further Reading

- Exponential-time algorithms
  - Chapter 1, *Introduction*, in (Fomin and Kratsch, 2010).
  - Survey on exponential-time algorithms (Woeginger, 2001).
  - Chapter 1, *Introduction*, in (Gaspers, 2010).
- Parameterized Complexity
  - Chapter 1, *Introduction*, in (Cygan et al., 2015)
  - Chapter 2, *The Basic Definitions*, in (Downey and Fellows, 2013)
  - Chapter I, *Foundations*, in (Niedermeier, 2006)
  - *Preface* in (Flum and Grohe, 2006)

## References

- Scott Aaronson (2005). “Guest Column: NP-complete problems and physical reality”. In: *SIGACT News* 36.1, pp. 30–52. DOI: 10.1145/1052796.1052804.
- Jianer Chen, Iyad A. Kanj, and Ge Xia (2010). “Improved upper bounds for vertex cover”. In: *Theoretical Computer Science* 411.40–42, pp. 3736–3756. DOI: 10.1016/j.tcs.2010.06.026.
- Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh (2015). *Parameterized Algorithms*. Springer. DOI: 10.1007/978-3-319-21275-3.
- Rodney G. Downey and Michael R. Fellows (2013). *Fundamentals of Parameterized Complexity*. Springer. DOI: 10.1007/978-1-4471-5559-1.
- Jörg Flum and Martin Grohe (2006). *Parameterized Complexity Theory*. Springer. DOI: 10.1007/3-540-29953-X.
- Fedor V. Fomin and Dieter Kratsch (2010). *Exact Exponential Algorithms*. Springer. DOI: 10.1007/978-3-642-16533-7.
- Serge Gaspers (2010). *Exponential Time Algorithms: Structures, Measures, and Bounds*. VDM Verlag Dr. Mueller.
- Martin Charles Golumbic (2004). *Algorithmic Graph Theory and Perfect Graphs*. Elsevier.
- Rolf Niedermeier (2006). *Invitation to Fixed Parameter Algorithms*. Oxford University Press. DOI: 10.1093/ACPROF:OSO/9780198566076.001.0001.
- Silvia Richter, Malte Helmert, and Charles Gretton (2007). “A Stochastic Local Search Approach to Vertex Cover”. In: *Proceedings of the 30th Annual German Conference on Artificial Intelligence (KI 2007)*. Vol. 4667. Lecture Notes in Computer Science. Springer, pp. 412–426. DOI: 10.1007/978-3-540-74565-5\_31.
- Gerhard J. Woeginger (2001). “Exact Algorithms for NP-Hard Problems: A Survey”. In: *Combinatorial Optimization*, pp. 185–208. DOI: 10.1007/3-540-36478-1\_17.
- Mingyu Xiao and Hiroshi Nagamochi (2017). “Exact algorithms for maximum independent set”. In: *Information and Computation* 255, pp. 126–146. DOI: 10.1016/j.ic.2017.06.001.