Fixed Parameter Tractability

Stefan Rümmele[1,2]

[1]University of Sydney, Australia
[2]UNSW, Australia

October 23, 2017

Central question in computer science

P vs. NP

Central question in computer science

P vs. NP

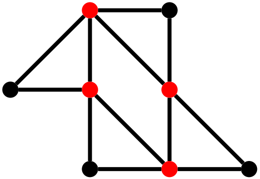
- no known polynomial time algorithm for any NP-hard problem
- belief: P $\neq$ NP
- What to do when facing an NP-hard problem?

Example problem: 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	
Instance:	Graph G, integer k.
Question:	Does G have a vertex cover of size k?

Note: VERTEX COVER is NP-complete.



Coping with NP-hardness

- Exact exponential time algorithms
  - There is an algorithm solving VERTEX COVER in time $O(1.1970^n)$, where $n = |V|$.

Coping with NP-hardness

- Exact exponential time algorithms
  - There is an algorithm solving VERTEX COVER in time $O(1.1970^n)$, where $n = |V|$.
- 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.

Coping with NP-hardness

- Exact exponential time algorithms
  - There is an algorithm solving Vertex Cover in time $O(1.1970^n)$, where $n = |V|$.
- 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.
- Approximation algorithms
  - There is an algorithm, which, given an instance (G, k) for Vertex Cover, finds a vertex cover of size at most $2k$ or correctly determines that G has no vertex cover of size k.

Coping with NP-hardness

- Exact exponential time algorithms
  - There is an algorithm solving VERTEX COVER in time $O(1.1970^n)$, where $n = |V|$.
- 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.
- Approximation algorithms
  - There is an algorithm, which, given an instance (G, k) for VERTEX COVER, finds a vertex cover of size at most $2k$ or correctly determines that G has no vertex cover of size k.
- Restricting the inputs
  - VERTEX COVER can be solved in polynomial time on bipartite graphs, trees, interval graphs, etc.

Coping with NP-hardness

- Exact exponential time algorithms
  - There is an algorithm solving VERTEX COVER in time $O(1.1970^n)$, where $n = |V|$.
- 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.
- Approximation algorithms
  - There is an algorithm, which, given an instance (G, k) for VERTEX COVER, finds a vertex cover of size at most $2k$ or correctly determines that G has no vertex cover of size k.
- Restricting the inputs
  - VERTEX COVER can be solved in polynomial time on bipartite graphs, trees, interval graphs, etc.
- Fixed parameter algorithms
  - There is an algorithm solving VERTEX COVER in time $O(1.2738^k + kn)$.

Exponential Time Algorithms in Practice

How large are the instances one can solve in practice?

Exponential Time Algorithms in Practice

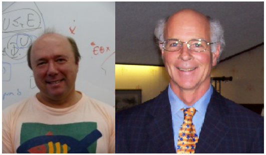
How large are the instances one can solve in practice?

Available time	1 s	1 min	1 hour	3 days	6 months
nb. of operations	2^{36}	2^{42}	2^{48}	2^{54}	2^{60}
n^5	147	337	776	1782	4096
n^{10}	12	18	27	42	64
1.05^n	511	596	681	767	852
1.1^n	261	305	349	392	436
1.5^n	61	71	82	92	102
2^n	36	42	48	54	60
5^n	15	18	20	23	25
$n!$	13	15	16	18	19

Note: Intel Core i7 920 (Quad core) executes between 2^{36} and 2^{37} instructions per second at 2.66 GHz.

Parameterized Complexity Theory

- Developed by Downey and Fellows in the early 1990s.
- Search for (hidden) parameters that make the problems hard.
- Problem instances where these parameters are small can be solved efficiently.



$\Rightarrow$ Multivariate complexity analysis.

Multivariate Complexity in Practices

Input size: $n = 1000$,
Parameter: $k = 20$

Running Time		
Theoretical	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.

Multivariate Complexity in Practices

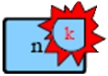
Input size: $n = 1000$,
Parameter: $k = 20$

	Running Time	
Theoretical	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 $13.8 \cdot 10^9$ years ago.

Fixed-Parameter Tractability (FPT)

Confine the combinatorial explosion to a parameter k.



Definition (FPT)

$$f(k) \cdot p(n),$$

$p(n) \ldots$ polynomial in the input size
$k \ldots$ parameter value
$f \ldots$ arbitrary computable function

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

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

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

Toolbox of Parameterized Complexity

Hardness Tools:

- W[i]-hardness
- Kernel lower bounds
- Exponential Time Hypothesis

Algorithmic Tools:

- Bounded search trees
- Iterative compression
- Logical meta-theorems
- Color coding
- Integer Linear Programming
- Kernelization

Toolbox of Parameterized Complexity

Hardness Tools:

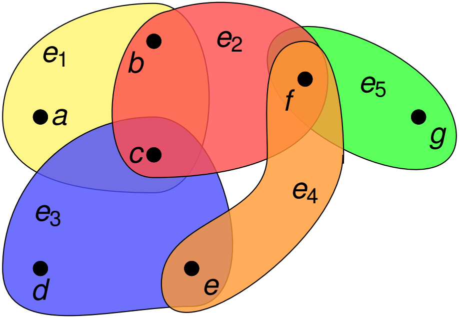
- W[i]-hardness
- Kernel lower bounds
- Exponential Time Hypothesis

Algorithmic Tools:

- **Bounded search trees**
- Iterative compression
- Logical meta-theorems
- Color coding
- Integer Linear Programming
- **Kernelization**

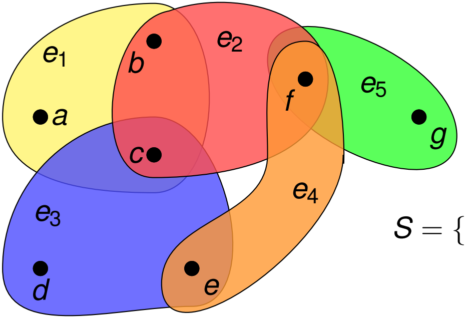
Hitting Set Problem

- A hypergraph $\mathcal{H} = (V, E)$ consists of a set of vertices V and a set of hyperedges E. A hyperedge is a subset of V.
- A hitting set of $\mathcal{H}$ is a set $S \subseteq V$ that intersects each hyperedge.
  - $S \cap e \neq \emptyset$ for all $e \in E$.



Hitting Set Problem

- A hypergraph $\mathcal{H} = (V, E)$ consists of a set of vertices V and a set of hyperedges E. A hyperedge is a subset of V.
- A hitting set of $\mathcal{H}$ is a set $S \subseteq V$ that intersects each hyperedge.
  - $S \cap e \neq \emptyset$ for all $e \in E$.



$S = \{c, f\}$

Hitting Set Problem

HITTING-SET	
Instance:	A hypergraph $\mathcal{H} = (V, E)$ and $k \in \mathbb{N}$.
Parameter:	$k + d$, where $d = \max\{\lvert e\rvert \mid e \in E\}$.
Problem:	Decide whether $\mathcal{H}$ has a hitting set of size k.

Hitting Set Problem

HITTING-SET	
Instance:	A hypergraph $\mathcal{H} = (V, E)$ and $k \in \mathbb{N}$.
Parameter:	$k + d$, where $d = \max\{|e| \mid e \in E\}$.
Problem:	Decide whether $\mathcal{H}$ has a hitting set of size k.

Observations:

- Each hyperedge $e \in E$ must be hit.
  $\Rightarrow$ Can be processed in any order.
- For every hyperedge $e \in E$ we have at most $|e| \in E \leq d$ choices.

Hitting Set Algorithm

Algorithm 1: Hitting-Set($\mathcal{H}, k$)

Input : Hypergraph $\mathcal{H} = (V, E)$, $k \geq 0$
Output : True if $\mathcal{H}$ has a hitting set of size k

if $|V| < k$ **then return** *False*
else if $E = \emptyset$ **then return** *True*
else if $k = 0$ **then return** *False*
else
    choose $e \in E$
    forall $v \in e$ **do**
        $V_v \leftarrow V \setminus \{v\}$
        $E_v \leftarrow \{e \in E \mid v \notin e\}$
        $\mathcal{H}_v \leftarrow (V_v, E_v)$
        if *Hitting-Set(*$\mathcal{H}_v, k-1$*)* **then return** *True*
    return *False*

Hitting Set Algorithm

Algorithm 2: Hitting-Set($\mathcal{H}, k$)

Input : Hypergraph $\mathcal{H} = (V, E)$, $k \geq 0$
Output : True if $\mathcal{H}$ has a hitting set of size k

if $|V| < k$ **then return** *False*
else if $E = \emptyset$ **then return** *True*
else if $k = 0$ **then return** *False*
else
    choose $e \in E$
    forall $v \in e$ **do**   branching factor at most d
        $V_v \leftarrow V \setminus \{v\}$
        $E_v \leftarrow \{e \in E \mid v \notin e\}$
        $\mathcal{H}_v \leftarrow (V_v, E_v)$
        if *Hitting-Set*($\mathcal{H}_v, k-1$) **then return** *True*   descending $\leq k$ times
    return *False*

Bounded Search Tree for Hitting Set

Theorem

HITTING-SET *is fixed-parameter tractable when parameterized by solution size k and maximum edge cardinality d. There is an algorithm solving* HITTING-SET *in time $\mathcal{O}(d^k \cdot \|\mathcal{H}\|)$.*

Bounded Search Tree for Hitting Set

Theorem

HITTING-SET *is fixed-parameter tractable when parameterized by solution size k and maximum edge cardinality d. There is an algorithm solving* HITTING-SET *in time $\mathcal{O}(d^k \cdot \|\mathcal{H}\|)$.*

- The size of the search tree is $\mathcal{O}(d^k)$.
- The computation at each search tree node is polynomial (linear) in $\|\mathcal{H}\|$.
- The size of the search tree does not depend on n.

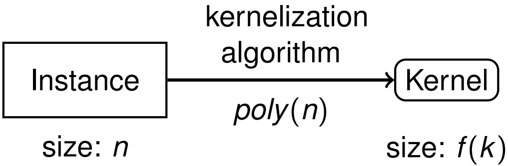
Kernelization – Formalization of preprocessing

Formalization of preprocessing in the classical setting is problematic.

Kernelization – Formalization of preprocessing

Formalization of preprocessing in the classical setting is problematic.

Idea. Use the parameter to capture how much the size of an instance is reduced



Kernelization

Kernelization is a polynomial-time transformation that maps an instance (I, k) to an instance (I', k') such that

- (I, k) is a yes-instance if and only if (I', k') is a yes-instance,
- $k' \leq k$, and
- $|I'| \leq f(k)$ for some function $f(k)$.

A Kernel for Vertex Cover

Observation: High degree vertices (degree $> k$) need to be selected.

A Kernel for Vertex Cover

Observation: High degree vertices (degree $> k$) need to be selected.

Rule 1: Delete every vertex of degree $> k$ and decrease k accordingly.

A Kernel for Vertex Cover

Observation: High degree vertices (degree $> k$) need to be selected.

Rule 1: Delete every vertex of degree $> k$ and decrease k accordingly.

Theorem

Rule 1 leads to a $\mathcal{O}(k^2)$ kernelization for VERTEX COVER.

- After applying Rule 1, the remaining graph has maximum degree k.
- Each vertex can cover at most k edges.
- The graph can contain at most k^2 edges and at most $2k^2$ vertices.

A Kernel for Vertex Cover

Observation: High degree vertices (degree $> k$) need to be selected.

Rule 1: Delete every vertex of degree $> k$ and decrease k accordingly.

Theorem

Rule 1 leads to a $\mathcal{O}(k^2)$ kernelization for VERTEX COVER.

- After applying Rule 1, the remaining graph has maximum degree k.
- Each vertex can cover at most k edges.
- The graph can contain at most k^2 edges and at most $2k^2$ vertices.

Current smallest known kernel for VERTEX COVER has $2k$ vertices and $\mathcal{O}(k^2)$ edges.

Further Reading

- Rodney G. Downey and Michael R. Fellows. Fundamentals of Parameterized Complexity. Springer, 2013.
- Rolf Niedermeier. Invitation to Fixed Parameter Algorithms. Oxford University Press, 2006.
- Jörg Flum and Martin Grohe. Parameterized Complexity Theory. Springer, 2006.
- Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, MichałPilipczuk, and Saket Saurabh. Parameterized Algorithms. Springer, 2015.

Acknowledgement:
Thanks to Serge Gaspers for providing some of his slides.