3a. Basics of Parameterized Complexity
COMP6741: Parameterized and Exact Computation

Serge Gaspers[1,2]

[1]School of Computer Science and Engineering, UNSW Sydney, Asutralia
[2]Decision Sciences, Data61, CSIRO, Australia

Semester 2, 2018

Outline

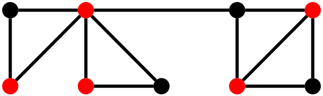
Outline

Outline

Vertex Cover

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

VERTEX COVER	
Input:	A graph $G = (V, E)$ and an integer k
Parameter:	k
Question:	Does G have a vertex cover of size k?



Algorithms for Vertex Cover

- brute-force: $O^*(2^n)$
- brute-force: $O^*(n^k)$
- vc1: $O^*(2^k)$ (cf. Lecture 1)
- vc2: $O^*(1.4656^k)$ (cf. Lecture 1)
- fastest known: $O(1.2738^k + k \cdot n)$ [Chen, Kanj, Xia, 2010]

Running times in practice

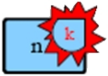
$n = 1000$ vertices,
$k = 20$ parameter

Theoretical	Running Time Nb 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
$2^k \cdot n$	$1.05 \cdot 10^9$	15.26 milliseconds
$1.4656^k \cdot n$	$2.10 \cdot 10^6$	0.31 milliseconds
$1.2738^k + k \cdot n$	$2.02 \cdot 10^4$	0.0003 milliseconds

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.



(1) Which problem–parameter combinations are fixed-parameter tractable (FPT)? In other words, for which problem–parameter combinations are there 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?

(2) How small can we make the $f(k)$?

Examples of Parameters

A Parameterized Problem

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

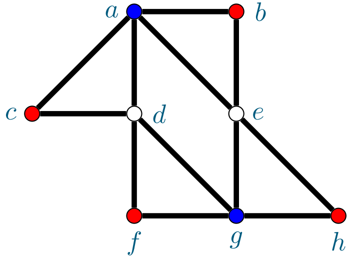
- A parameter can be
  - solution size
  - input size (trivial parameterization)
  - related to the structure of the input (maximum degree, treewidth, branchwidth, genus, ...)
  - combinations of parameters
  - etc.

Outline

Coloring

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

Coloring
Input: Graph G, integer k
Parameter: k
Question: Does G have a k-coloring?



Brute-force: $O^*(k^n)$, where $n = |V(G)|$.
Inclusion-Exclusion: $O^*(2^n)$.
FPT?

Coloring is probably not FPT

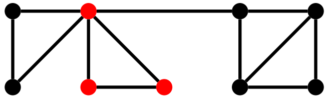
- Known: COLORING is NP-complete when $k = 3$
- Suppose there was a $O^*(f(k))$-time algorithm for COLORING
  - Then, 3-COLORING can be solved in $O^*(f(3)) \subseteq O^*(1)$ time
  - Therefore, P = NP
- Therefore, COLORING is not FPT unless P = NP

Outline

Clique

A clique in a graph $G = (V, E)$ is a subset of its vertices $S \subseteq V$ such that every two vertices from S are adjacent in G.

CLIQUE	
Input:	Graph $G = (V, E)$, integer k
Parameter:	k
Question:	Does G have a clique of size k?



Is CLIQUE NP-complete when k is a fixed constant? Is it FPT?

Algorithm for Clique

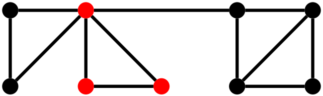
- For each subset $S \subseteq V$ of size k, check whether all vertices of S are adjacent
- Running time: $O^*\left(\binom{n}{k}\right) \subseteq O^*(n^k)$
- When $k \in O(1)$, this is polynomial
- But: we do not currently know an FPT algorithm for Clique
- Since Clique is W[1]-hard, we believe it is not FPT. (See lecture on W-hardness.)

Outline

A different parameter for Clique

Δ-CLIQUE

Input:	Graph $G = (V, E)$, integer k
Parameter:	$\Delta(G)$, i.e., the maximum degree of G
Question:	Does G have a clique of size k?



Is Δ-CLIQUE FPT?

Algorithm for Δ-Clique

Input: A graph G and an integer k.
Output: YES if G has a clique of size k, and NO otherwise.

if $k = 0$ **then**
  return YES
else if $k > \Delta(G) + 1$ **then**
  return NO
else
  /* A clique of size k contains at least one vertex v.

Algorithm for Δ-Clique

Input: A graph G and an integer k.
Output: YES if G has a clique of size k, and NO otherwise.

if $k = 0$ **then**
  return YES
else if $k > \Delta(G) + 1$ **then**
  return NO
else
  /* A clique of size k contains at least one vertex v. For each $v \in V$, we check whether G has a k-clique S containing v (note that $S \subseteq N_G[v]$ in this case). */

Algorithm for Δ-Clique

Input: A graph G and an integer k.
Output: YES if G has a clique of size k, and NO otherwise.

```
if k = 0 then
  return YES
else if k > Δ(G) + 1 then
  return NO
else
  /* A clique of size k contains at least one vertex v.
     For each v ∈ V, we check whether G has a k-clique S
     containing v (note that S ⊆ N_G[v] in this case).          */
  foreach v ∈ V do
    foreach S ⊆ N_G[v] with |S| = k do
      if S is a clique in G then
        return YES
  return NO
```

Algorithm for Δ-Clique

Input: A graph G and an integer k.
Output: YES if G has a clique of size k, and NO otherwise.

```
if k = 0 then
  return YES
else if k > Δ(G) + 1 then
  return NO
else
  /* A clique of size k contains at least one vertex v.
     For each v ∈ V, we check whether G has a k-clique S
     containing v (note that S ⊆ N_G[v] in this case).        */
  foreach v ∈ V do
    foreach S ⊆ N_G[v] with |S| = k do
      if S is a clique in G then
        return YES
  return NO
```

Running time: $O^*((\Delta+1)^k) \subseteq O^*((\Delta+1)^\Delta)$. (FPT for parameter Δ)

Outline

Main Parameterized Complexity Classes

n: instance size
k: parameter

P: class of problems that can be solved in $n^{O(1)}$ time
FPT: class of parameterized problems that can be solved in $f(k) \cdot n^{O(1)}$ time
XP: class of parameterized problems that can be solved in $f(k) \cdot n^{g(k)}$ time
("polynomial when k is a constant")

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

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

Note: We assume that f is computable and non-decreasing.

Outline

Further Reading

- Chapter 1, *Introduction* in
  Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, MichałPilipczuk, and Saket Saurabh. Parameterized Algorithms. Springer, 2015.
- Chapter 2, *The Basic Definitions* in
  Rodney G. Downey and Michael R. Fellows. Fundamentals of Parameterized Complexity. Springer, 2013.
- Chapter I, *Foundations* in
  Rolf Niedermeier. Invitation to Fixed Parameter Algorithms. Oxford University Press, 2006.
- *Preface* in
  Jörg Flum and Martin Grohe. Parameterized Complexity Theory. Springer, 2006.