Exact Algorithms

Sommer Term 2020

Lecture 10 Kernelization

Based on: [Parameterized Algorithms: §1, 2.1]

and [Gramm, Guo, Hüffner, Niedermeier; Theory Comput. Syst. 38:373–392, 2005. doi.org/10.1007/s00224-004-1178-y]

(slides by J. Spoerhase, Th. van Dijk, S. Chaplick, and A. Wolff)

Alexander Wolff

Lehrstuhl für Informatik I

Complexity Theory

Natural problems tend to be either:

- in **P** $\leftarrow$ Easy
- **NP-hard** $\leftarrow$ Nasty

How to solve nasty problems?

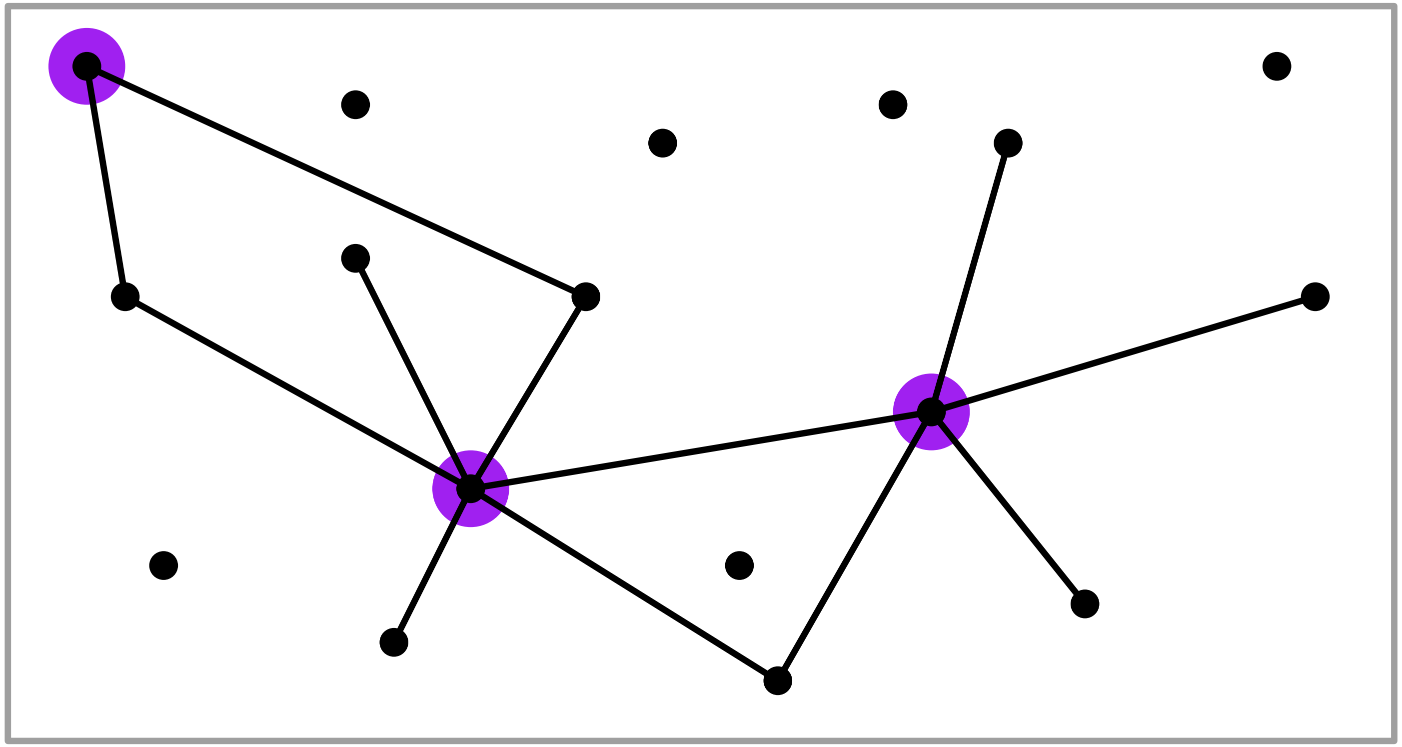
- Solve only small instances (e.g., integer programming)
- Don't solve the problem in full generality:
    - . . . approximate
    - . . . exploit properties of "reasonable" instances

Fixed-Parameter Tractability:

- In many applications, some aspect of a problem assumed small.
- Runtime of algorithm polynomial except for this small aspect.

Vertex Cover

Minimum-size vertex set such that every edge is covered.



Problem: *k-Vertex Cover (k-VC)*

Given: graph G

Parameter: number k

Question: Does G contain a size $\leq k$ vertex cover?

Last time: FPT :)

Coloring

Given: graph G, number k

Question: Does G have a coloring with $\leq k$ colors?

For any $k \geq 3$, not in polynomial time

assuming $\mathrm{P} \neq \mathrm{NP}$

k-Coloring

Given: graph G

Parameter: number k

Question: Does G have a coloring with $\leq k$ colors?

For any $k \geq 3$, not in polynomial time

assuming $\mathrm{P} \neq \mathrm{NP}$

$\Rightarrow$ Not in FPT !!!!

k-Independent Set

Given: graph G
Parameter: number k
Question: Does G have a size $\geq k$ independent set?

How to find a 2-Independent Set? 3-Independent Set?

Easily in $O(n^k)$ time, i.e., polynomial for any k

Requires $\Omega(n^{f(k)})$ time...
assuming FPT $\neq$ W[1]

Fixed-Parameter Complexity Theory

Fixed-Parameter Tractable

Solvable in $O(f(k)n^c)$ time – computable function f, const. c

Complexity Class: FPT

Slice-Wise Polynomial

Solvable in $O(f(k)n^{g(k)})$ time – computable functions f, g

Complexity Class: XP

Evidence against Fixed-Parameter Tractability

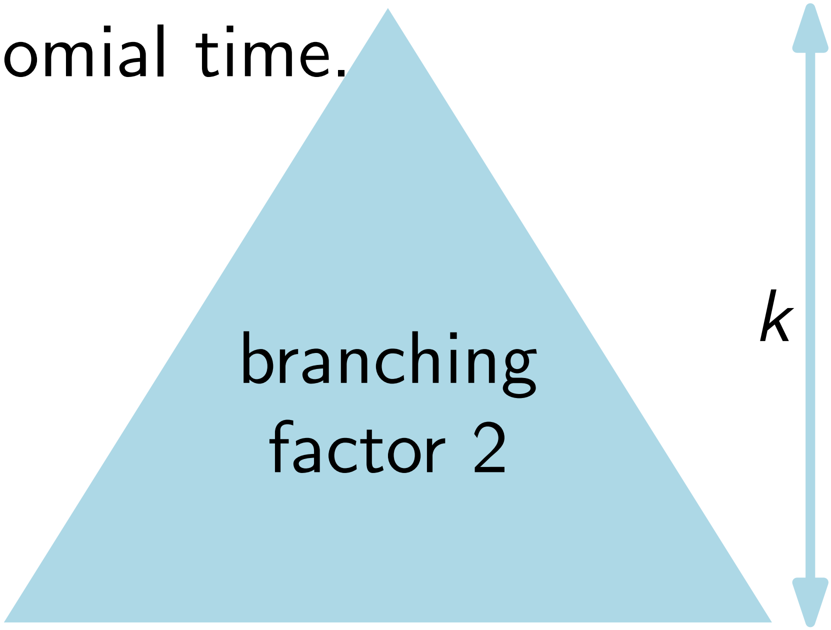
Complexity classes FPT $\subseteq$ W[1] $\subseteq$ W[2] $\ldots$

e.g. FPT $=$ W[1] $\Rightarrow$ NP $\subseteq$ DTIME($2^{o(n)}$), contradicting ETH

↳ see Parameterized Algorithms §14.4

Fixed-Parameter Problems

- k-Coloring
  NP-complete for each $k \geq 3$.
  Seems to require more than polynomial time.
- k-Independent Set
  In XP, but W[1]-complete...
  Seems to require $\Omega(n^{f(k)})$ time.
- k-Vertex Cover
  Solvable in $O(2^k(n+m))$ time.



Obs: If there are no edges, solution size 0.
Obs: For each edge uv, either u or v in the cover.

Branching Rule:
Pick an edge uv, branch on $(G-u, k-1)$ and $(G-v, k-1)$.

Kernelisation

Preprocessing with quality guarantees

Kernelisation Algorithms

Parameterized Problem: input graph G, parameter k.

Polynomial-time algorithm:
Input: Graph G, parameter k
Output: Graph G', parameter $k' \leq k$ such that

- (G', k') is YES $\Leftrightarrow$ (G, k) is YES.
- G' has $O(f(k))$ vertices.

eg, recall: Buss' algorithm for k-VC

size-$O(k^2)$ kernel

I) Reduce to the kernel *of the instance*

$C = \{v \in V \mid \deg(v) > k\}$; **if** $|C| > k$ **then return** (“NO”, $\emptyset$)

$G' = (V', E') := G[V \setminus (C \cup L)]$, $k' = k - |C|$ (L = isolated vertices)
if $|E'| > k^2$ **then return** (“NO”, $\emptyset$)

II) solve the reduced problem exactly

Kernel $\Leftrightarrow$ FPT

Kernelisation $\Rightarrow$ FPT

- run kernelisation algorithm
- solve kernel by any exact algorithm

FPT $\Rightarrow$ Kernelisation

Suppose we have an $f(k)|I|^c$-time algorithm for an instance I.

Run the algorithm for $|I|^{c+1}$ steps.

If the decision is reached, output it.

Otherwise, $f(k) \geq |I|$.

So, we have an $f(k)$-size kernel. $\square$

Typical Form of Kernelisation

Repeat some **rules**, until no **rule** is possible

- Rules can do some necessary modification and decrease k.
- Rules can remove some part of the graph.
- Rules can output YES or NO.
- Sometimes add ‘annotations’ to the graph

Cluster Editing

Given: graph $G = (V, E)$
Parameter: number k
Question: Can we make $\leq k$ **modifications** to G so that each connected component is a clique?
Accepted modifications: adding / deleting edges

Known kernelizations:

$O(k^2)$ vertices [Gramm, Guo, Hüffner, Niedermeier; TCSyst'05]

← *Let's prove this!*

$O(k)$ vertices [Fellows, Langston, Rosamund, Shaw; FCT'07]

$2k$ vertices [Chen, Meng; JCSS'12]

Trivial Rules and Plan

Rule 1: If a connected component C of G is a clique, remove this connected component:

$G' \leftarrow G - C$; $k' \leftarrow k$

Rule 2: If we have more than k connected components and Rule 1 does not apply: Answer **NO**.

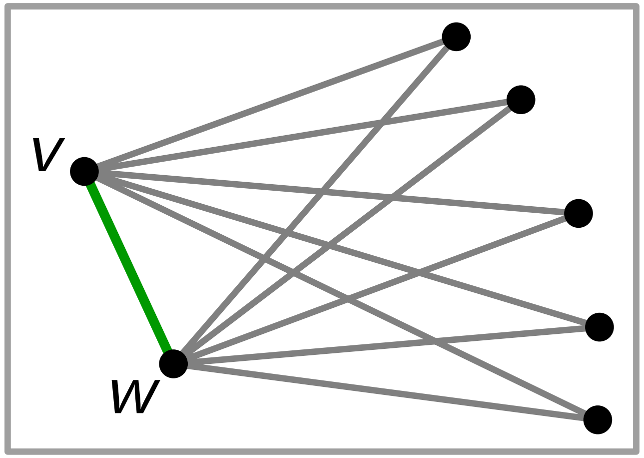
Obs. After applying Rules 1 and 2, at most k connected components remain.

Plan: Find rules that make connected component small!

Annotate the graph:
some pairs of vertices are permanent and others are forbidden.

Rule 3: Common Neighbors

Obs. If two vertices have $k+1$ neighbors in common, they must belong to the same clique in any solution



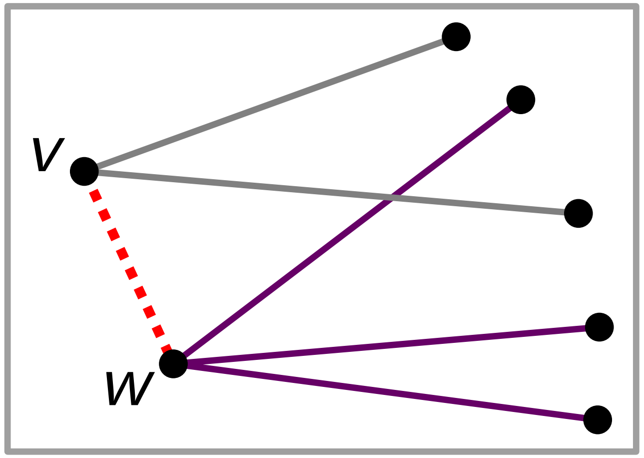
Rule 3: Let v, w be vertices with $k+1$ common neighbors. If vw is not present, add it and decrease k by 1. Set the edge vw to be **permanent**.

$$G' \leftarrow G \cup \{vw\} \qquad P \leftarrow P \cup \{vw\}$$

$$k' = \begin{cases} k & \text{if } vw \text{ present,} \\ k-1 & \text{otherwise.} \end{cases}$$

Rule 4: Private Neighbors

Obs. If vtc. v and w have $k+1$ "uncommon" neighbors, then vw cannot be an edge in the solution.



Rule 4: Let v, w be vtc. with $k+1$ uncommon neighbors. If vw is present, remove it and decrease k by 1. Set the edge vw to be **forbidden**.

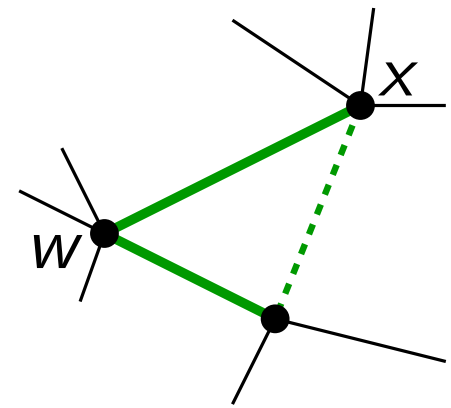
$$G' \leftarrow G - vw; \qquad F \leftarrow F \cup \{vw\}$$

$$k' \leftarrow \begin{cases} k & \text{if } vw \text{ not present,} \\ k-1 & \text{otherwise.} \end{cases}$$

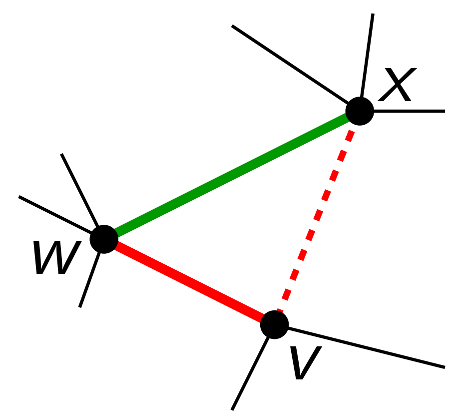
Rule 5: If some edge is both permanent and forbidden, then there is no solution. Answer **NO**.

Rules 6 and 7: Transitivity (Triangles)

Rule 6: If *vw* and *wx* are **permanent**
then set *vx* to be **permanent**.
If *vx* is not present,
then add it and decrease *k* by 1.



Rule 7: If *vw* is **permanent**
and *wx* is **forbidden**,
then set *vx* to be **forbidden**.
If *vx* is present,
then remove it and decrease *k* by 1.



Runtime

The rules can be computed in polynomial time.

With carefully chosen data structures, they can be exhaustively applied in $O(n^3)$ time.

Challenge

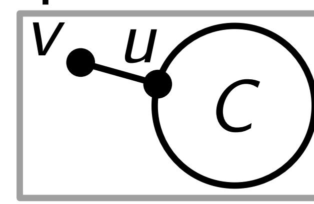
- Already know: at most k connected components
- Aim for a quadratic kernel: $O(k^2)$ vertices

So, “small” components are fine –
but what if we have a “big” component?

Analysis

Apply rules exhaustively; call resulting graph $G = (V, E)$ *reduced*.
G is connected (otherwise treat each component separately).
Rule 1 $\Rightarrow$ G is not a clique. $\xrightarrow{\text{edge modifications}}$ solution $G' = (V, E')$
Let $a = \#\text{additions}$, $d = \#\text{deletions}$ $\Rightarrow k = a + d$.
Assume $|V| > (2k+1) \cdot k$. Two cases; show contradiction.

Case 1: $a = 0 \Rightarrow k = d$. Let $C \subset V$ be the largest clique in G'.
Since $a = 0$, C is a clique in G, too.
G connected $\Rightarrow \exists uv \in E \colon u \in C, v \in V \setminus C$



Case 1A: v is not adjacent to any $u' \in C \setminus \{u\}$. Then (if $k \geq 2$)

$$|C| \geq |V|/(d+1) > \frac{(2k+1)\cdot k}{k+1} = \frac{(k+1)\cdot k}{k+1} + \frac{k^2}{k+1} \geq k+1$$

$\Rightarrow |C| \geq k+2$

$\Rightarrow u$ and v have $k+1$ uncommon neighbors ↯ Rule 4

Exercise: Find a contradiction for the case $k = 1$!

Analysis

Apply rules exhaustively; call resulting graph $G = (V,E)$ *reduced*.

G is connected (otherwise treat each component separately).

Rule 1 $\Rightarrow$ G is not a clique. $\xrightarrow[\text{edge modifications}]{}$ solution $G' = (V, E')$

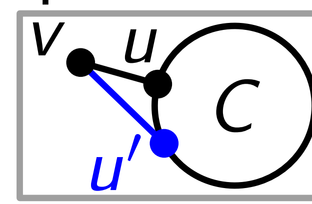
Let $a = \#\text{additions}$, $d = \#\text{deletions}$ $\Rightarrow k = a + d$.

Assume $|V| > (2k+1) \cdot k$. Two cases; show contradiction.

Case 1: $a = 0 \Rightarrow k = d$. Let $C \subset V$ be the largest clique in G'.

Since $a = 0$, C is a clique in G, too.

G connected $\Rightarrow \exists uv \in E \colon u \in C, v \in V \setminus C$



Case 1B: v is adjacent to some $u' \in C \setminus \{u\}$. Then

$|C| \geq |V|/d > 2k + 1$

v has at most k neighbors in C (otherwise G is a NO)

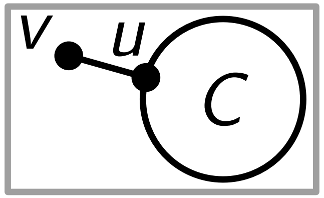
$\Rightarrow$ u and v have $k+1$ uncommon neighbors ↯ Rule 4

Analysis – Case 2

Case 2: $a > 0 \Rightarrow k > d$. Let $C \subset V$ be the largest clique in G'.
$|C| \geq |V|/(d+1) \geq |V|/k > 2k+1$

Case 2A: $a = k \Rightarrow V = C$, and G' consists of one clique.
Since $a \geq 1$, $\exists uv \notin E$. Recall that $|V| > (2k+1) \cdot k$.
$\Rightarrow u$ and v have $\geq k+1$ common neighbors. ↯ Rule 3

Case 2B: $a < k \Rightarrow \exists uv \in E \colon u \in C, v \in V \setminus C$



Recall that $|C| > 2k+1$.

Note: u has $\geq 2k+1-a \overset{??}{-} 1$ neighbors in C.
v has $\leq d-1$ neighbors in $C \setminus \{u\}$.

Let $x = \#$ uncommon neighbors of u and v.
$\Rightarrow x \geq 2k+1-a-d = k+1$ ↯ Rule 4

Exercise: Find a simple branching rule for Cluster Editing!