Fixed-parameter Tractable (FPT) problems

Brute-force Vertex Cover:

recall VC: Instance (G, k)

Q: $\exists\ C \subseteq V(G)$ of size $|C| \leq k$ s.t. each edge is "covered" by (at least) one vertex in C

[Formal: $\forall e \in E(G)\ \exists\ v \in C : v \in e$].

natural parameter is here $k_I = k$

- try all $\binom{V}{k} + \underbrace{\binom{V}{k-1} + \ldots + \binom{V}{0}}_{\text{can be skipped}}$ subsets of V of size $\leq k$

  (if $\exists$ VC of size ℓ $\Rightarrow \exists$ VC of size $k \geq \ell$)

- test if each edge is "covered" in $O(E)$ time

$\Rightarrow$

$O(V^k \cdot E)$ time (polynomial for fixed k)

but even for small k ($k=2$) can become impractable on large graphs)

exponent depends on k = BAD.

AIM: Alg where $|I|^{(..)}$ exponent does not depend on k!

= Small change with big effect.

Vertex Cover can be solved much faster

("Bounded search tree" algorithm) $\subseteq$ VC

Instance (G, k)

- consider any edge $e = \{u, v\}$.

$\Rightarrow$ u in VC or $v \in V$ or both in VC

(in 2-approx we put both in, but we want to have exact solution & we don't know if u or v or both!)

- either $u \in C$ or $v \in C$ (or both)

- GUESS ① $u \in C$:

$\Rightarrow$ 

then remove u & all incident edges

$\Rightarrow$ get new graph but since we added u to C decrement k.

$\Rightarrow$ new instance $(G - u, k-1)$

now recurse on this new instance.

② $v \in S$ (analogously)

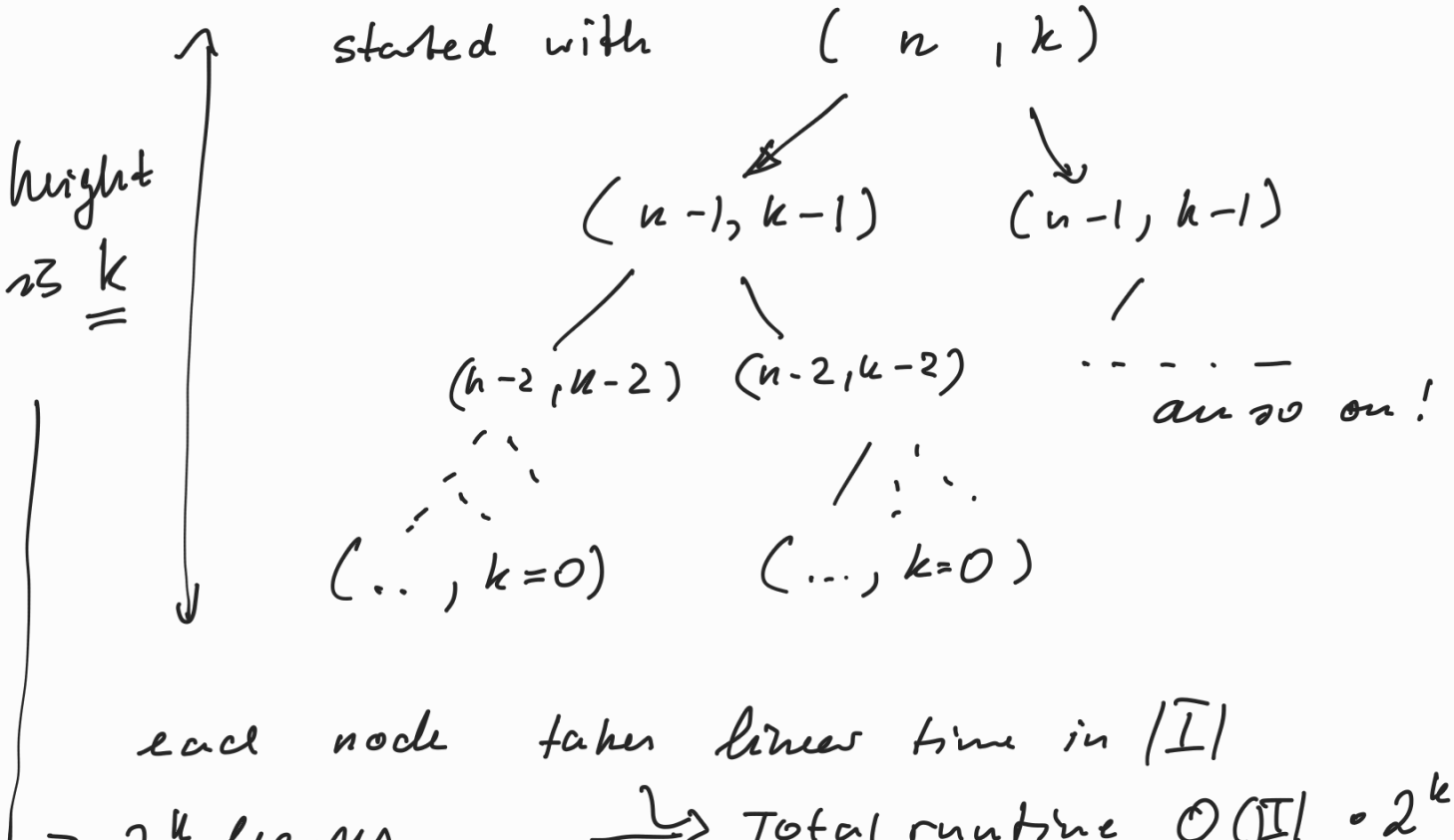
"Base Case" where Alg stops:

$k = 0$ → $|E| = \emptyset$ we have VC of size k

→ $|E| \neq \emptyset$ no such VC exists.

RUNTIME ? can think of this as Dynamic Programming (without memorization)

Look at recursion tree: let n = nr of vertices in G



each node takes linear time in $|I|$

↳ 2^k leaves ⟶ Total runtime $O(|I| \cdot 2^k)$.

$\Rightarrow$ for fixed k (constant we have linear time Alg!).

Important we used parameter k to bound size of search tree!

More to Bounding search Trees:

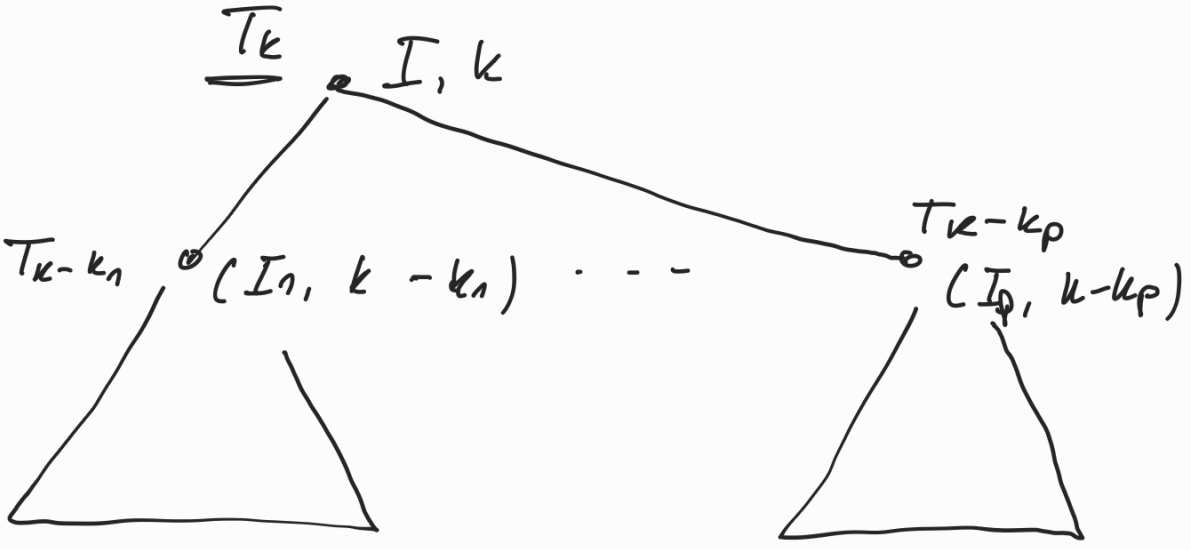
IF we have "Branching algorithm" that for (I, k_I) produces [simply: $k := k_I$]

p subproblem $(I_1, k - k_1), \ldots (I_p, k - k_p)$

(in last exmp $k_1 = \ldots = k_p = 1$ & $p = 2$)

↳ $(k_1, \ldots, k_p)$ is called Branching Vector

then we get search tree: T_k



Ann. $L(T_k) = \alpha^k \quad \forall\, k$

$$\Rightarrow \quad L(T_k) = \sum_{i=1}^{p} L(T_{k-k_i})$$

$$\alpha^k \qquad\qquad \sum_{i=1}^{p} \alpha^{(k-k_i)}$$

Branching vector in last exmpl (VC): $(1,1)$

$$\Rightarrow \quad \alpha^k = \alpha^{k-1} + \alpha^{k-1} \qquad (\alpha > 0)$$

$$\Leftrightarrow \alpha = 1 + 1 = 2$$

$$\Leftrightarrow L(T_k) = \alpha^k = 2^k \quad \checkmark$$

$$\Leftrightarrow \quad \alpha^k - \sum_{i=1}^{p} \alpha^{(k-k_i)} = 0 \quad \text{getting the roots!}$$

FOR branching vector $(k_1 .. k_p)$

the <u>characteristic polynom</u> is

$$z^k = \sum_{i=1}^{p} z^{k-k_i} \quad \text{with} \quad k = \max(k_1 .. k_p)$$

root of this eqn. $(z_k - \Sigma(\,) = 0)$

is <u>the branching-Factor</u>

in VC exmpl: $k = \max(1,1) \Rightarrow$ charact. poly:

$z^1 = z^0 + z^0 = 2$ ✓

For more sophisticated branching alg,
we may have branching vectors (1,3)
or (1, 4, 8 ..)

e.g For (1,3) $\Rightarrow z^3 = z^2 + 1 \Rightarrow$ root
for (I, k) $z = 1,47$

$\Rightarrow$ # leaves in T_k :

$$L(T_k) \in O(1,47^k)$$

$\Rightarrow$ checking these may then
yield $O(|I| \cdot 1,47^k)$ alg ..

[when you see alg like this
then ofte the reason are
these branching vectors].

Fastest VC-Alg:

2010: $O(|I|^c + 1,2783^k)$ Chen, Kanj & Xia

improved 2024: $O(|I|^c + 1,25284^k)$ Harris & Narayanaswamy.

Thm: $\exists\ f(k_I)\cdot |I|^c$ alg $\iff$ $\exists\ \tilde{f}(k_I)+|I|^c$ alg.

proof „$\Leftarrow$" clear assuming $f(k_I), |I|^c \geq 1$

then

$$f(k_I)+|I|^c \in O(f(k_I)\cdot |I|^c)$$

„$\Rightarrow$" 2 cases: $|I| \leq f(k_I)$ as $f(k_I) \leq |I|$.

$$|I| \leq f(k_I) \Rightarrow f(k_I)\cdot |I|^c \leq f(k_I)\cdot f(k_I)^c = f(k_I)^{c+1}$$

$$f(k_I) \leq |I| \Rightarrow f(k_I)\cdot |I|^c \leq |I|\,|I|^c = |I|^{c+1}$$

$\tilde{c} = c+1$

$$\Rightarrow f(k_I)\cdot |I|^c \leq \max\{f(k_I)^{\tilde{c}}, |I|^{\tilde{c}}\} \leq \underbrace{f(k_I)^{\tilde{c}} + |I|^{\tilde{c}}}_{=\tilde{f}(k)} \quad \square$$

[alternative proof: If always holds $x, y \geq 1$

$$0 \leq (x-y)^2 = x^2 - 2xy + y^2 \iff xy \leq 2xy \leq x^2+y^2$$

so $\hat{f}(k) = f(k)^2$, $\tilde{c} = 2c$ does the job

eg: $O(2^k n) \leq O(4^k + n^2)$]

Theorem: (Π, k) is FPT $\Leftrightarrow$ $\exists$ kernelization of (Π, k).

Proof: „$\Leftarrow$"

kernelize (Π, k)

$\Rightarrow$ $\forall$ (I, k_I) we obtain in polytime $\mathcal{O}(|I|^c)$ an instance $(I', k_{I'})$

of size $|I'| \leq \tilde{f}(k_I)$

now can use any finite algorithm (e.g. Brute force) will runtime $g(|I'|)$ (could even be exponential)

Total runtime to solve I: $g(I')$ + Kernelization

$$= \mathcal{O}(|I|^c) + \underbrace{g(\tilde{f}(k_I))}_{= f(k_I)}$$

$\Rightarrow$ FPT

„$\Rightarrow$" let (I, k_I) be solvable in $|I|^c \cdot f(k_I)$ time

- if $|I| \leq f(k_I)$ we are done

Since here I already „kernalized" (satisfies all points of Def)

- $f(k_I) \leq |I|$

Since (Π, k) FPT, use FPT-Alg to solve (I, k_I)

in $f(k_I)\,|I|^c \leq |I|^{c+1}$ time.

i.e. in polynomial time!

$\Rightarrow$ the FPT Alg is in fact a kernelization algorithm.
(it even solves the problem)
just adjust to have as output (I, k_I) again + answer yes/no

(or canonical
$(\circ\!-\!\circ, 1)$ Yes
$(\circ\;\;\circ, 0)$ No instance)

Kernelization for Vertex Cover

given instance $(G, k_G = k)$

where Q: $\exists$ VC C of G s.t. $|C| \leq k$?

kernelization as follows:

Rule 0: if G has vertex v with $\deg(v) = 0$ can remove v from G. (clear)

lemma 0: (G,k) has yes answer $\Leftrightarrow$ $(G-v, k)$ has yes answer $\forall v \in G : \deg(v) = 0$

Rule 1: if $\deg(v) > k$ then v must be in any VC C of G (else its neighbors must in $C \Rightarrow |C| > k$!)

lemma 1 (G,k) has yes answer $\Leftrightarrow$ $(G-v, k-1)$ has yes answer $\forall v \in G : \deg(v) > k$

Algorithm:

Reduce(G,k) wrt VC

In: G, k

WHILE (G contains vertex v with $\deg(v) > k$ or $\deg(v) = 0$) DO

  let v in G be with $\deg(v) = 0$ or $\deg(v) > k$.

  IF ($\deg(v) = 0$)

    $G \leftarrow G - v$

  ELSE // $\deg(v) > k$

    $G \leftarrow G - v$

    $k \leftarrow k - 1$

IF $|V| > 2 \cdot k^2$ RETURN (⟶○, 0) (return (instance with "No" & size $\leq k$))

ELSE RETURN (G,k)

// (G',k') below.

$(|I| = |V| \cdot |E|)$

- Runs in polytime $O(|I|^c)$ size of I (Exc).
- $k' \leq k$

remains to show that $|I'| \leq \tilde{f}(k)$!

+ transform instance (G,k) to instance (G',k') s.t. (G,k) yes $\overset{L0+L1}{\Longleftrightarrow}$ (G',k') YES

Lemma: „Reduce(G,k) wrt VC" with input (G,k) returns a instance (G',k')

s.t. (G,k) yes $\Leftrightarrow$ (G',k') yes.

& such that size of G' is bounded by k^2.

proof: IF „(G',k')" returned in „ELSE"

then we know already

by LO+L1 that

(G,k) yes $\Leftrightarrow$ (G',k') yes.

must ensure that $(\multimap, \phi)$ is correct and that size of G' is bounded by k^2.

Let $G' = (V', E')$

By construction, $\forall\, v \in V' : 1 \leq \deg(v) \leq k'$

$\Rightarrow$ each vertex covers $\leq k'$ edges

$\Rightarrow$ IF G' has VC C of size $\leq k'$

THEN $|E'| \leq (k')^2$ ($k' \cdot k'$)

$\Rightarrow$ (each $e \in E'$ contributes 2 vertices of V)

so $|V'| = \left|\bigcup_{e \in E'} e\right| \leq 2 \cdot (k')^2$ ($\lightning$ $|V'| = 2k'^2$)

D.h. if G' has VC C with $|C| \leq k'$
then $|V'| \leq 2 \cdot (k')^2$ must be satisfied

$\Rightarrow$ IF $|V'| > 2(k')^2$ we definitely have a NO-instance of (G', k')
& thus of (G, k)

$\Rightarrow$ now NO-instance $(G', k') = (\circ\!\!-\!\!\circ, 0)$
is returned (IF)
& $G' \leq k' \leq k$

IF $|V'| \leq (2k')^2$
then (G', k') may or may not have VC of size $\leq k'$

$(\Leftrightarrow (G, k)$ — " — $\leq k)$

Returned (G', k'): $|E'| \leq (k')^2 \leq k^2$
$|V'| \leq 2(k')^2 \leq 2k^2$

$\Rightarrow$ size (G', k') is in all cases bounded $|I'| \leq k(I')$

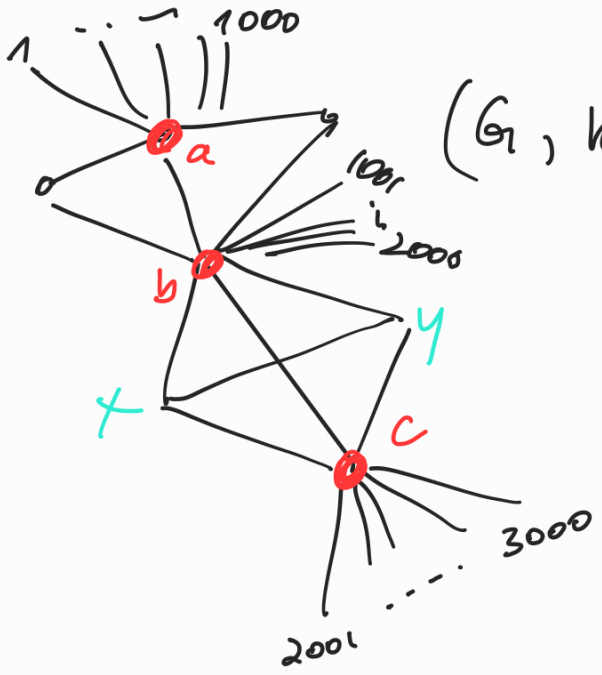
□

In summary: „ Kernelize(G,k) wrt VC "

is a correct kernelization.

Given the $O(k^2)$ kernel we can now apply any brute-force alg. to solve the problem.

Example

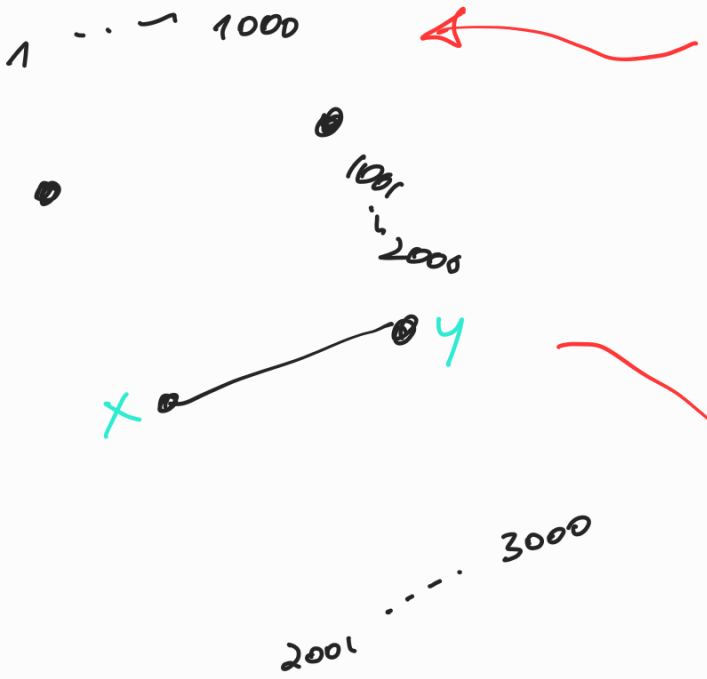


$(G, k=4)$

$\Rightarrow$ 3007 vertices test of all 4 elem. subsets $\sim 3{,}34$ Mrd subsets (BAD).

adapt alg:

Rule 1 {
- add a to C, $k \leftarrow 4-1=3$
- add b to C, $k \leftarrow 3-1=2$
- add c to C, $k \leftarrow 2-1=1$



get $((b-a)-b)-c$

Remove all deg 0 vertices.

$(b', k'=1)$

Do is there VC of size 1 in b' [Brute Force].

YES.