Fixed parameter tractability

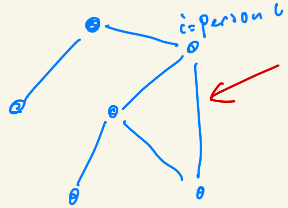
Based on parts from M. Cygan et al "Parametrized Algorithms"

Running example: vertex-cover (= bar fight prevention)

- Suppose n people want to enter a bar
- We know which pairs do not like each other and hence will start a fight if they are let in together
- Question: For a given $k<n$ is it possible to exclude at most k persons so that no fight between remaining people?

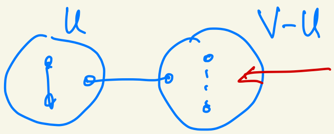
Model as a vertex cover problem:

$G=(V,E)$



will fight if both are allowed in

Answer is 'yes' if we can partition V into $U, V-U$ s.t.



no edges (independent)

and $|U| \leq k$

We know: Vertex cover is NP-complete

Given $\langle G,k \rangle$ is there a VC of size $\leq k$?

We must solve the bar fight prevention problem so what do we do?

Suppose there are $n=1000$ people wanting to go to the bar (Let assume, like in DK recently, the all have to book access)

suppose we will allow at most $k=10$ persons excluded

Naive method: try all 2^n subsets and check if at least one is a VC of size $\le k$. Bad: $2^{1000} \approx 1.07 \cdot 10^{301}$ subsets to t

Better method: try all $\binom{n}{k}$ subsets of size k

still not good as $\binom{1000}{10} \sim 2.63 \cdot 10^{23}$ cases to check

We need a better approach!!

Solution: Problem reduction / kernelization

Consider conflict graph $G=(V,E)$ and k

Rule 1: If $d(v)=0$: $\langle G,k\rangle \to \langle G-v,k\rangle$ (delete v)

v is never in minimum VC

Rule 2: If $d(v)\ge k+1$: $\langle G,k\rangle \to \langle G-v,k-1\rangle$ we must include v in any VC of size $\le k$

Rule 3: If $d(v)=1$ and w is only neighbour of v: $\langle G,k\rangle \to \langle G-\{v,w\},k-1\rangle$ adding w to C is at least as good as adding v to

Rule 1 : If $d(v)=0$: $\langle G,k\rangle \to \langle G-v,k\rangle$

Rule 2 : If $d(v)\geq k+1$: $\langle G,k\rangle \to \langle G-v,k-1\rangle$

Rule 3 : If $d(v)=1$ and w is only neighbour of v : $\langle G,k\rangle \to \langle G-\{v,w\},k-1\rangle$

Apply Rule 1-3 until none applies and let $G'=(V',E')$ be resulting graph and k' the resulting parameter

Notes

1) if current parameter k' is $=0$ and still edges in actual graph then we can reject original input $\langle G,k\rangle$. We only decreased parameter when we found a vertex that should be in VC in Rule 2 or had to add v or w to VC in Rule 3

2) $d_{G'}(v)\leq k' \;\forall v\in V'$ as Rule 2 cannot be applied

$\Rightarrow |E'|\leq k'^2$ each vertex can cover at most k' edges

3) $|V(G')|\leq k^2$:

$$|V'| = \sum_{v\in V'} 1 = \frac{1}{2}\sum_{v\in V'} 2 \leq \frac{1}{2}\sum_{v\in V'} d(v) = |E'| \leq k'^2 \leq k^2$$

as Rule 3 does not apply

Conclusion (For general n, k)

After applying Rules 1-3 until none apply to current instance $\langle G' = (V', E'), k' \rangle$ we know

1. $\langle G', k' \rangle$ is a yes instance (G' has a VC of size $\leq k'$)

$\Updownarrow$

$\langle G, k \rangle$ is a yes instance

2. $|V'| \leq k'^2 \leq k^2$ and $|E'| \leq k'^2 \leq k^2$

Now we can solve $\langle G', k' \rangle$ brute force:

try all k' subsets of V'

at most $\binom{k'^2}{k'} \leq \binom{k^2}{k}$ of these

when $k = 10$ we have to try at most $\binom{100}{10} \approx 1.73 \cdot 10^{13}$ subsets

Thus in polynomial time we have reduced the original instance to one where the problem is <u>much</u> faster to solve

Let C_1 be vertices decided to add to the VC when applying rules and let C' be VC found when showing that $\langle G', k' \rangle$ is a yes-instance

Then $C_1 \cup C'$ is a VC of G of size $\leq k$

- Applying each of the rules 1-3 can be done in linear time in size of the current instance

- Hence after doing $O((n+m)k)$ work we have either decided that $\langle G, k \rangle$ is a no instance or we have derived an equivalent instance $\langle G', k' \rangle$ s.t $\langle G, k \rangle$ is a yes-instance

$\Updownarrow$ $\langle G', k' \rangle$ is a yes-instance

- We can decide $\langle G', k' \rangle$ using at most $\binom{k^2}{k}$ checks of subsets of size k

Each of them take time $O(|V'| + |E'|) = O(k^2)$

- Putting it together we solve $\langle G, k \rangle$ in time

$$O((n+m)k) + O\left(\binom{k^2}{k} k^2\right) = O(g(k)(n+m))$$
$$= O(g(k) \cdot n^2)$$

We have shown that an instance $\langle G, k \rangle$, where G has n vertices, can be decided by an algorithm running in time $O(g(k) n^2)$ for some function g depending only on k.

We parameterized the Vertex-cover problem using the natural parameter k = bound on size of VC

<u>Definition</u> a parameterized problem Q is Fixed Parameter Tractable (FPT) if there exists an algorithm A_Q solving Q in time $O(f(k) n^c)$ for some computable function f and constant $c \in \mathbb{R}_+$

We showed above that Vertex-cover is FPT as the algorithm we described is an FPT algorithm.

The way we solved Vertex-cover was by problem reduction.

What we obtained after exhaustively applying rules 1-3 is called a problem kernel

Definition (Kernelisation, kernel)

A kernalization algorithm (a kernel) for a parameterized problem Q is an algorithm $\mathcal{A}_Q$ that given an instance $\langle I, k \rangle$ works in polynomial time in $|\langle I, k \rangle|$ and outputs an equivalent instance $\langle I', k' \rangle$, where $|I'| + k' \leq g(k)$ for every instance $\langle I, k \rangle$ of Q and g is a fixed computable function.

For vertex cover the algorithm we described took input $\langle G, k \rangle$ and produced a kernel $\langle G', k' \rangle$ satisfying that $|G'| + k' \leq 2k^2 + k$

Note that if a parametrized problem Q with parameter k has a kernel of size $O(g(k))$ for some g then we can solve Q by first finding a kernel and then checking all possible solutions for the kernel
(brute force approach)

Natural questions

1. Can we find a better kernel? (smaller size)
2. Solve kernel faster than by brute force.

A kernel of size at most $2k$ for vertex-cover

Recall the LP-based approximation algorithm for VC:

1. Solve $z^*_{LP} = \min \sum_{v \in V} x(v)$, s.t $x(u)+x(v) \geq 1 \ \forall uv \in E$ $\quad 0 \leq x(v) \leq 1$

2. Let $\hat{x}$ be an optimal LP-solution

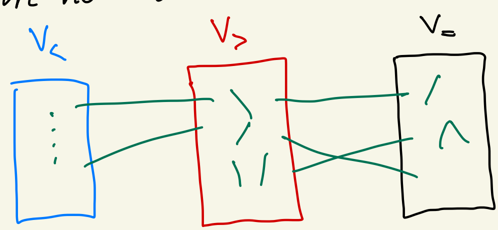
3. Take $U = \{v \mid \hat{x}(v) \geq \frac{1}{2}\}$

We saw that $\frac{|U|}{|U_{opt}|} \leq 2$

Split $V = V(G)$ into 3 sets $V_<, V_=, V_>$ when

$V_< = \{v \in V \mid \hat{x}(v) < \frac{1}{2}\}$, $V_= = \{v \in V \mid \hat{x}(v) = \frac{1}{2}\}$ and $V_> = \{v \in V \mid \hat{x}(v) > \frac{1}{2}\}$

Note that $V_<$ is independent (no edges inside)
and there are no edges between $V_<$ and $V_=$ (as $x(u)+x(v) < 1$ when $u \in V_<, v \in V_=$)



Theorem (Nemhauser and Trotter)
There exist an optimal VC U^* s.t. $V_> \subseteq U^* \subseteq V_= \cup V_>$

Theorem (Nemhauser and Trotter)

Then exist an optimal VC U^* s.t. $V_> \subseteq U^* \subseteq V_= \cup V_>$

Proof suppose X is an optimal VC of G and that $X \cap V_< \neq \emptyset$

- let $X' = (X \setminus V_<) \cup V_>$. Then X' is a VC as $V_<$ is independent and no edge uv with $u \in V_<, v \in V_=$
- If $|X'| \leq |X|$ we are done so assume $|X'| > |X|$

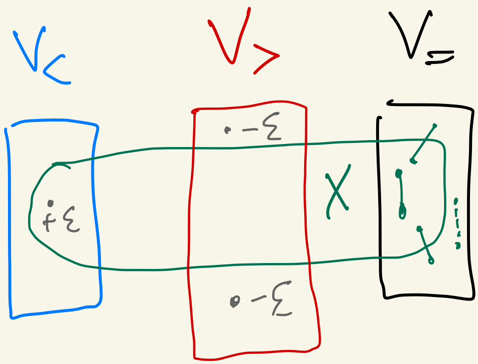
then $|V_> \setminus X| > |X \cap V_<|$. (*)

- let $\varepsilon = \min_{v \in V_< \cup V_>} |\frac{1}{2} - \hat{x}(v)|$

- For $v \in V_> \setminus X$ let $x(v) = \hat{x}(v) - \varepsilon$

For $v \in V_< \cap X$ let $x(v) = \hat{x}(v) + \varepsilon$

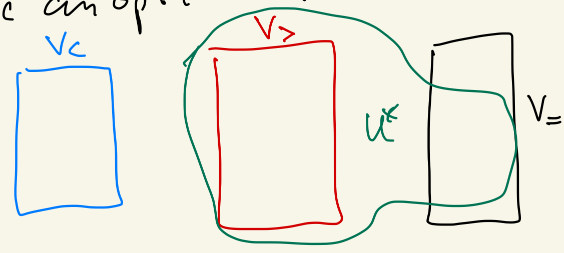
For all other v let $x(v) = \hat{x}(v)$



Now $\sum_{v \in V} x(v) < \sum_{v \in V} \hat{x}(v)$ by (*),

contradicting that $\hat{x}$ is an optimal LP-sol □.

So we have an optimal VC U^* with



Let $k^* = |U^*|$ and recall that we may assume $V_> \subseteq U^*$

Reduction rule (after solving LP and getting opt sol $\hat{x}$)

- let $G' = G[V_=]$ subgraph induced by $V_=$
- and let $k' = k - |V_>|$

Note: If $|V_>| > k$, then $k^* > k$ as the theorem guarantees that $V_>$ is contained in some optimal solution.
Hence we can answer 'no' for $\langle G, k \rangle$ output $\langle G' = K_{k'+2}, k' \rangle$

If $|V_>| = k$, then check if $V_=$ is independent.
If 'yes' $|V_>|$ is a VC of size k for G.
Else answer no for $\langle G, k \rangle$ output $\langle G' = K_{k'+2}, k' \rangle$

Now we can assume $|V_>| < k$ and then $\langle G', k' \rangle$ is a kernel and $|V(G')| = |V_=| \le 2k$

a) $k \ge \cancel{k^*} \ge \sum_{v \in V_=} \hat{x}(v) = \frac{1}{2}|X_=|$

$\boxed{\text{Conclusion: We have found a kernel of size at most } 2k \text{ for VC with parameter } k}$

Note after lecture: We may assume that $\sum_{v \in V} \hat{x}(v) \le k$ since otherwise there is no VC of size k.
Now $k \ge \sum_{v \in X_=} \hat{x}(v) = \frac{1}{2}|X_=|$ so $|X_=| \le 2k$

NB!

Back to our bar fight problem with $n = 1000$ $k = 10$

- First solve associated LP (in polynomial time)

- Find $V_<$, $V_=$ and $V_>$

- If $|V_>| \geq k$ reject (answer no) unless $|V_>| = k$ and $|V_>|$ is a VC

- Otherwise solve the bar fight problem for $G[V_=]$ with parameter $k' \leq k$

  This can be done i time $O\left(\binom{2k}{k} \cdot k^2\right)$

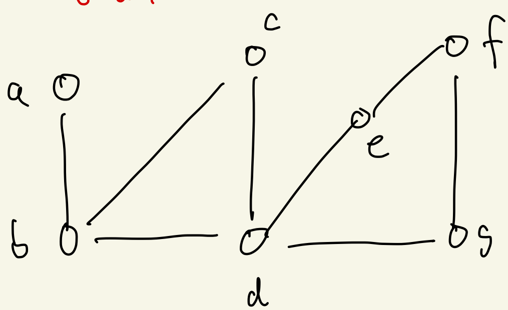
  For $k = 10$ $\binom{20}{10} = 184756$ so we easy solve problem in a few seconds, even using brute force

$\binom{100}{50} \sim 1{,}01 \cdot 10^{29}$ so we cannot hope to use brute force except when k is small even on a small kernel!

$\binom{2k}{k} < 2^{2k} = 4^k$ so brute force alg is $O(4^k \cdot k^2)$

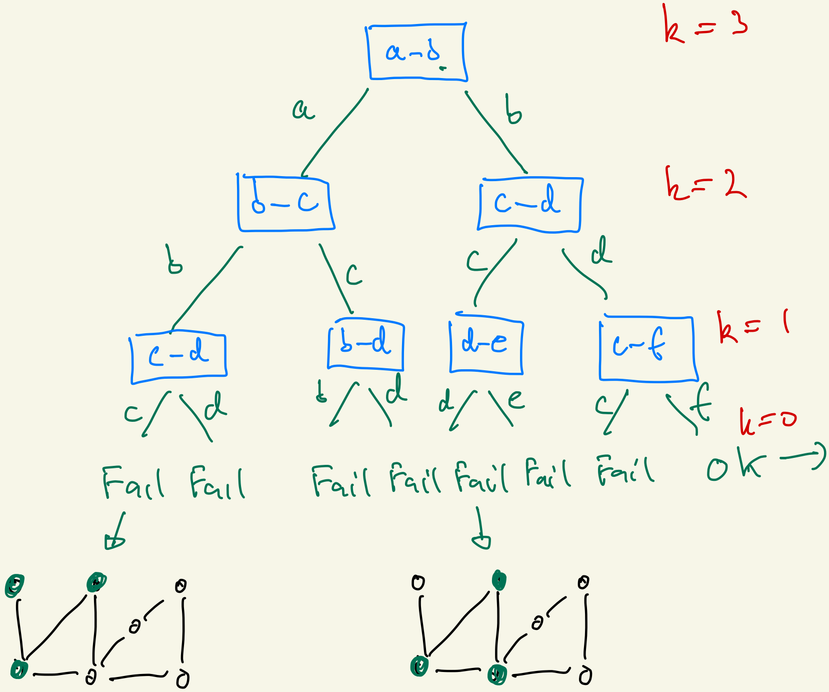
Tree search

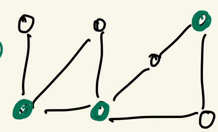
Idea: Given instance ⟨G, k⟩, try the edges in some order, using that if uv is an edge, then at least one of u, v must be in the cover.

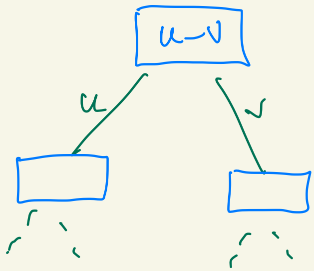


assume k = 3

k decreases by 1 for each level







2 recursive calls, each with parameter decreased by 1

We look for VC of size $\leq k$ so height of tree is at most $k-1$ (depth is at most k)

$\Downarrow$ at most 2^k (-1) subproblems

By preprocessing via Rule 2 ($d(v) \geq k+1$?) we can assume that $d(v) \leq k$ for all vertices so $E = \frac{1}{2} \sum_{v \in V} d(v) \leq \frac{1}{2} nk$

Thus each of the 2^k subproblems can be solved in time $O(nk)$

So total running time is $O(m + nk \cdot 2^k)$

from checking degrees of vertices in G.