Monday, September 24, 2018 12:15 AM

FPT algos

Want to solve NP-hard problems faster

E.g. vertex cover: delete min # vtcs s.t. no edges remaining

Idea 1: Let's approximate soln
- 2-apx soln in polytime
- 2 is tight assuming UGC

Idea 2: Let's restrict the input somehow.
- Disallow "lower bound instances"

This lecture: restrict input by parameterization
- Restrict to instances where $OPT \le k$

Vertex cover parameterized by OPT

Input: (unweighted) graph G

Output: If $OPT \le k$, output OPT

Else, output $\perp$

Def: Parameterized problem

Input: (I, k) where I is an instance and $k \in \mathbb{N}$ is a parameter

Output: (specified by problem)

Runtime: $f(k)n^c$ for absolute fn $f: \mathbb{N} \to \mathbb{N}$ const $c \in \mathbb{N}$

"exponential in k, poly in n"

"Capture the hardness of the problem into a param k"

① Parameterized vertex cover:

We'll see: ① $2^k \text{poly}(n)$

② $\text{poly}(n) + 2^k \text{poly}(k)$

② k-path

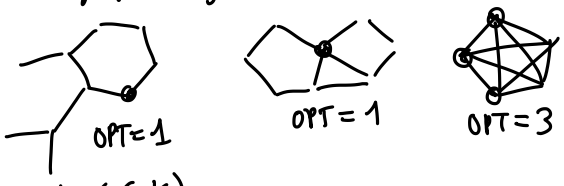
Input: (G, k)

Output: a simple path of length k in G (or $\perp$ if none exist)

Runtime: ① ~~$k!\,\text{poly}(n)$~~ $k^k \text{poly}(n)$

② $c^k \text{poly}(n)$

③ Feedback vertex set: delete min # vertices to make graph acyclic



Input: (G, k)

Output: if $OPT \le k$, then output OPT

else, output $\perp$

Runtime: $4^k \, \text{poly}(n)$

---

② k-path

Brute-force: n^k (not FPT!)

Idea: <u>randomization!</u>

Algo:
- Every vtx gets random label in $[k]$
- We want:

  $1 - 2 - 3 - \cdots - k$

  - Prob that fixed k-path looks like this is $\cancel{\frac{1}{k!}}$ $\frac{1}{k^k}$
- Given labeling, find longest path w/ increasing labels
- Can be done using DP
- One easy way to look at it:
  - Direct the edges and look for

    $1 \to 2 \to 3 \to \cdots \to k$
  - For each edge in G, direct it to higher labeled vertex:

    $l_u \to l_v \iff l_u < l_v$

    If $l_u = l_v$, then remove edge.
  - Graph is acyclic (very important!)
  - Find longest path in DAG [DP]

Runtime: poly(n)

Success prob: $\frac{1}{k!}$

Repeat $\Theta(k! \log n)$ times

replace $k!$ w/ k^k

$\Rightarrow$ fail prob $\leq \left(1 - \frac{1}{k!}\right)^{10 k! \log n} \leq e^{-\frac{1}{k!} \cdot 10 k! \log n} \leq \frac{1}{n^{10}}$.

Main Insight: by using randomization, we were able to ensure that EVERY size-k subset of vtcs is "rainbow-colored" with prob $\frac{1}{k!}$.

Hard to ensure deterministically: there are n^k such subsets!

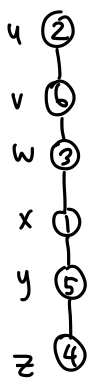
Speed up? Key idea: Requiring <u>sorted order</u> on k-path was too strict. Let's relax to:

- We want:

  $2 - 6 - 3 - 1 - 5 - 4$ $k=6$
  - All vertices on fixed k-path have unique labels
  - Prob is $\frac{k!}{k^k} \sim \frac{\sqrt{2\pi k}\left(\frac{k}{e}\right)^k}{k^k} = \Omega\left(\frac{1}{e^k}\right)$.
- Task: given labeled graph, find a k-path w/ all vtcs different labels
- Higher success prob, but harder

algorithmic task

- We'll show $2^k \text{poly}(n)$ time
  $\Rightarrow$ repeat $\Theta(e^k \log n)$ times
  $\Rightarrow (2e)^k \text{poly}(n)$ time.
- "Subset DP"
  $DP(v, I)$: $v \in V$, $I \subseteq [k]$
  "a path ending at v whose vertices have labels in I, one of each (in particular, path has length $|I|$), or $\perp$ if none exist"
  Final answer: find a v s.t. $DP(v, [k]) \neq \perp$
  else, output $\perp$



$DP(u, \{2\}) = (u)$
$DP(v, \{2,6\}) = (u, v)$
$DP(v, \{2,3,6\}) = (u,v,w)$
$\vdots$
$DP(v, [6]) = (u,v,w,x,y,z)$

Initialize: $DP(v, \{l_v\}) = (v)$
Recursion $DP(v,S)$, $l_v \in S$:
If $\exists$ neighbor u s.t. $DP(u, S \setminus l_v) \neq \perp$,
then set $DP(v,S) = (DP(u, S \setminus l_v), v)$
(append v to end of path)

Else, $= \perp$

- Technique called "color coding"
  [Alon, Yuster, Zwick '95]
- Further improvements:
  $2^k \text{poly}(n)$ [Williams '09]
  $2^{3k/4} \text{poly}(n)$ undirected [Bjorklund '10]
  - also implies $2^{3n/4}$ algo for Hamilton path (first $(2-\varepsilon)^n$ algo)

---

① Vertex Cover

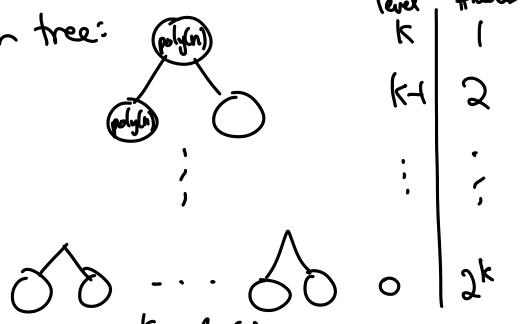
Let's solve <u>decision</u> version for simplicity:
Input: (G, k)
Output: Does G have a VC of size k?

Simple Randomized algo:
- For k iters: pick an arbitrary edge, choose random vertex

- Prob of success: $\frac{1}{2^k}$
- Repeat $\Theta(2^k \log n)$ times $\Rightarrow$ runtime $2^k \text{poly}(n)$.

"Derandomize"? <u>Branching</u> "brute force to bounded depth"

- Pick an arbitrary edge uv, <u>branch</u> on whether to choose u or v, recurse on each
- Algo $VC(G, k)$
  - If $k=0$:
    - If $G=\emptyset$: return YES
    - Else return NO
  - $uv \leftarrow$ arbitrary edge in G
  - return $VC(G-u, k-1)$ OR $VC(G-v, k-1)$
- Recursion tree:



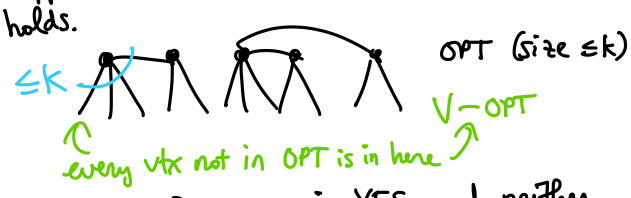
level	#nodes
k	1
$k-1$	2
$\vdots$	$\vdots$
0	2^k

$\Rightarrow$ runtime $2^k \text{poly}(n)$.

Kernelization

- Obs: If $\exists v: \deg(v) \geq k+1$, then greedily choose v.

  "Reduction 1" (G,k) is YES $\Leftrightarrow$ $(G-v, k-1)$ is YES
- Reduction 2: if $\exists$ isolated vtx v, remove v.
- Suppose answer is YES, but neither reduction holds.

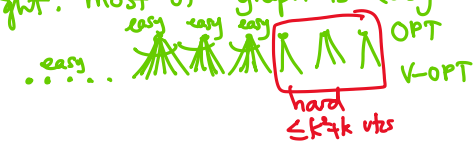


- LEMMA: If answer is YES and neither reduction holds, then $n \leq k^2+k$.
- Algo VC_kernel(G,k)
  - (Red. 1) If $\exists v: \deg(v) \geq k+1$, return VCker$(G-v, k-1)$
  - (Red. 2) If $\exists v$ isolated, return VCker$(G-v, k)$
  - at this point, neither red holds — If $n > k^2+k$, return NO
  - $O(2^k k^2)$ time — Return $VC(G,k)$

Total: $O(n^2 + 2^k \text{poly}(k))$

- Total time: poly(n) + 2^k poly(n)

Main Insight: most of graph is "easy"



- DEF (Kernelization)

  A kernelization algo inputs (G,k), runs in poly time, and outputs (G',k') s.t.
  - (G,k) is YES $\Leftrightarrow (G',k')$ is YES
  - $|G| \le f(k)$
  - $k' \le k$
- polynomial time kernelization algo if $f(k) = \text{poly}(k)$
- VC has a polynomial kernel.

---

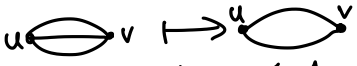
③ Feedback Vertex set

Decision version:

Input: (G,k)

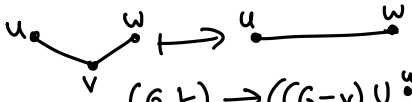
Output: Can we delete k vtcs in G to make it acyclic?

- Randomized branching with reductions
- Setting: multigraphs (loops & multi-edges)
- Reduction 1: If v has a loop, then greedily choose v. $(G,k) \to (G-v, k-1)$
- Reduction 2: If uv has edge multiplicity ≥ 3, then reduce it to 2.



- Reduction 3: If v has degree ≤ 1, delete v

  $(G,k) \to (G-v, k)$
- Reduction 4: If v has degree $= 2$, delete v and connect its neighbors by an edge.



$(G,k) \to ((G-v) \cup \overset{u}{\bullet}\!-\!\overset{w}{\bullet}, k)$

- Obs: if no more reductions, G has min deg ≥ 3.

  $\Rightarrow G$ has $\ge \frac{3}{2}n$ edges

V-OPT

acyclic $\Rightarrow$ #edges $\le |V\text{-OPT}| - 1$

$\le n - k$

OPT

$\Rightarrow$ at least $\left(\frac{3}{2}n - (n-k)\right)$ edges not inside $G[V\text{-OPT}]$

$\ge \frac{1}{2}n \ge \frac{1}{3}|E|$.

- Pick random edge. With prob $\ge \frac{1}{3}$, pick one adjacent to OPT.
- Pick random endpoint. With prob $\ge \frac{1}{2}$, pick correctly.

- Success prob $\geq \frac{1}{6}$, decrease k by 1

  $\Rightarrow$ overall success prob $\geq \frac{1}{6^k}$

  repeat $\Theta(6^k \log n)$ times.