Algorithmic Graph Theory

Sommer Term 2015

Lecture #11

Fixed Parameter Tractability

Prof. R. Inkulu

IIT Guwahati
Approaches to NP-Hard Problems

- Exponential-time algorithms, for example: back-tracking
- Approximation algorithms: Trade quality against runtime
- Heuristics: empirical investigations on benchmark sets
- Randomization: Searching for a needle in a stack of hay
- Design of parametrized algorithms
Today’s Example: Vertex Cover

Def. (Let’s recall...) Let \( G = (V, E) \) be an undirected graph. We call \( C \subseteq V \) a vertex cover of \( G \) if \( \{u, v\} \cap C \neq \emptyset \) for all \( uv \in E \).

Prob. Minimum Vertex Cover – Optimization problem
Given: Graph \( G \)
Find: a minimum-cardinality vertex cover of \( G \)

Prob. \( k \)-vertex cover (\( k \)-VC) – Decision problem
Given: Graph \( G \), positive integer \( k \)
Find: Vertex cover of size \( \leq k \) of \( G \) – if such a cover exists (otherwise return „no“)
Previous Work

- one of the first problems that was shown NP-hard
  \((\text{SAT} \preceq_p \text{CLIQUE} \preceq_p \text{VC} \preceq_p \ldots)\) \[\text{[Karp, 1972]}\]

- one of the six fundamental NP-hard problems in the classic book:
  \[\text{[Garey & Johnson, 1979]}\]

- can be approximated:
  Maximal matching “yields” factor-2 approximation.

- \ldots, but not arbitrarily well:
  There is no faktor-1.361 approximation algorithm for VC if \(\mathcal{P} \neq \mathcal{NP}\).
  \[\text{[Dinur & Safra, 2004]}\]
An Exact Algorithm for $k$-VC

BruteForceVC(Graph $G$, Integer $k$)

foreach $C \in \binom{V}{k}$ do

  // test whether $C$ is a VC

  $vc = true$

  foreach $uv \in E$ do

    if $\{u, v\} \cap C = \emptyset$ then

      $vc = false$

    if $vc$ then

      return ("yes", $C$

  return ("no", $\emptyset$

Runtime. $O(n^k m)$ – This is not polynomial in the size of the input ($= n + m$) since $k$ is not a constant, but part of the input.
A New Aim

Find an algorithm for $k$-VC with running time

$$O(f(k) + |I|^c),$$

where $f : \mathbb{N} \rightarrow \mathbb{N}$ is a computable function (independent of $I$), $I$ is the given instance, $c$ a constant (independent of $I$).

That is, the runtime must depend
– arbitrarily on the parameter $k$, hardness of the problem
– polynomially on the size $|I|$ of the instance $I$.

A problem that can be solved within this time bound is called fixed-parameter tractable with respect to $k$.

$\mathcal{FPT} =$ class of fixed-parameter tractable problems.

Comment. The class $\mathcal{FPT}$ does not change if we replace $+$ by $\cdot$. 
A Few Simple Observations . . .

Let \( G \) be a graph, \( C \) VC of \( G \), \( v \) vertex not in \( C \). Then which vertices are certainly contained in \( C \)?

**Obs. 1.** Let \( G \) be a graph, \( C \) a VC of \( G \), \( v \) a vertex. Then \( v \in C \) or \( N(v) \subseteq C \).

Consider decision problem \( k\)-VC.
What holds for vertices of degree \( > k \)?

**Obs. 2.** Every vertex of degree \( > k \) is contained in every \( k\)-VC.

What happens if \( |E| > k^2 \) and all vertices are of degree \( \leq k \)?

**Obs. 3.** If \( |E| > k^2 \) and \( \Delta(G) := \max_{v \in V} \deg v \leq k \), then \( G \) has no \( k\)-VC.
Algorithm of Buss

BussVC(Graph $G$, Integer $k$)

I) Reduce to the kernel of the problem

$C = \{v \in V | \deg v > k\}$
if $|C| > k$ then return “no”

$G' = (V', E') := G[V \setminus C]$ (without isolated vertices)
$k' = k - |C|$
if $|E'| > k'^2$ then return “no”

II) Solve remaining problem by brute force

$(vc, C') = \text{BruteForceVC}(G', k')$
return $(vc, C \cup C')$

Runtime. $O(n + m) \text{ time}$

Hence: $k$-VC $\in \mathcal{FPT}$!
Search Tree Algorithm

**Idea.** Improve Phase II by building a search tree.

If there is a leaf $\ell$ with $E_\ell = \emptyset$, then $C_\ell$ is a $k$-VC of $G$. If such a leaf doesn’t exist, then $G$ doesn’t have any $k$-VC.

#nodes: $S(k) \leq 2S(k-1) + 1$, $S(0) = 1 \implies S(k) \leq 2^{k+1} - 1 \in O(2^k) \implies \text{Runtime: } O^*(2^k)$

If there is a leaf $\ell$ with $E_\ell = \emptyset$, then $C_\ell$ is a $k$-VC of $G$. If such a leaf doesn’t exist, then $G$ doesn’t have any $k$-VC.
**The Degree-3 Algorithm**

**Idea.** Improve bound for $|N(v)|$.

⇒ tree size $S(k) = S(k - 3) + S(k - 1) + 1$, $S(\leq 3) = \text{const.}$

branching vector $(3, 1)$

Test $S(k) = z^k - 1 \Rightarrow z^k = z^{k-3} + z^{k-1} \cdot \frac{1}{z^{k-3}}$

⇒ Characteristic polynomial: $z^3 = 1 + z^2$

⇒ Largest real solution: $z \approx 1.466$ (branching number)

⇒ $S(k) \in O(1.466^k)$ – but how do we guarantee $\deg v \geq 3$?
Kernelization II

Previous Kernelization:

rule k: Put vertices of deg. $\geq k$ into cover $C$.
rule 0: Eliminate vertices of degree 0.

Improved Kernelization:

rule 1: Eliminate vertices of degree 1:

- set $G' = G - \{v, w\}$
- set $k' = k - 1$
- solve $(G', k', C')$
- set $C = C' \cup \{w\}$

Claim. $C$ is a $k$-VC of $G$ $\iff$ $C'$ is a $k'$-VC of $G'$.

$\Rightarrow$ Suppose $\exists$ edge $e$ in $G'$ not covered by $C'$. But then $e$ is not covered by $C$ either.

$\Leftarrow$ $w$ covers all edges not contained in $G'$. So if $C'$ covers $G'$, then $C' \cup \{w\}$ covers $G$. 
Improved Kernelization:

rule 2.1: Eliminate deg-2 vertices whose neighbors aren’t adjacent.

- set $k' = k - 1$
- solve $(G', k', C')$
- If vertex $uw$ is in $C'$, put $u$ and $w$ in $C$, otherwise put $v$ in $C$.

rule 2.2: Eliminate deg-2 vertices whose neighbors are adjacent.

- set $k' = k - 2$
- solve $(G', k', C')$
- set $C = C' \cup \{u, w\}$
The Degree-3 Algorithm

**Idea:** Apply the improved kernelization *in each node* of the search tree *exhaustively*!

\[ \Rightarrow \textbf{Runtime: } O(nk + k^2 \cdot 1,466^k) \subseteq O^*(1.466^k) \]

The Degree-4 Algorithm

**rule 3.1:** Eliminate deg-3 vertices whose neighbors aren’t adjacent.

\[ \Rightarrow S(k) \leq S(k - 4) + S(k - 1) + 1, \quad S(\leq 4) = \text{const.} \]

**rule 3.2:** Eliminate deg-3 vertices whose neighbors are adjacent.

\[ \Rightarrow \text{Charakteristic polynomial: } z^4 = 1 + z^3 \]

\[ \Rightarrow \text{Largest real solution: } z \approx 1.38 \]

\[ \Rightarrow \text{Search tree has size } S(k) \in O(1.38^k) \]
Conclusions

- $k$-VC can be solved exactly in $O(nk + 1.38^k k^2)$ time by the degree-4 algorithm.

Currently fastest algorithm for $k$-VC:
$O(nk + 1.274^k)$ time \[\text{[Chen, Kanj, Xia, MFCS’06]}\]

- Parametrized complexity =
  new toolbox for NP-hard problems:
  kernelization, DP tables, search trees, . . .

- It always makes sense to identify bounded parameters –
  FPT exploits them!

- hope:
  “natural” problem $P \in \mathcal{FPT} \Rightarrow f(k)$ bearable.
Books about **FPT**

- **Parameterized Complexity**
  - R.G. Downey
  - M.R. Fellows
  - 1999

- **Parameterized Complexity Theory**
  - Jörg Flum
  - Martin Grohe
  - 2006

- **Invitation to Fixed-Parameter Algorithms**
  - Rolf Niedermeier
  - 2006