Deterministic parameterized connected vertex cover

Marek Cygan

IDSIA, University of Lugano, Switzerland

July 4, Helsinki, SWAT 2012

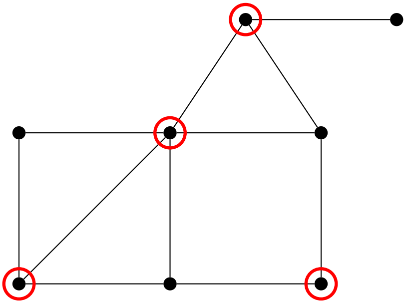
Outline

1. Introduction.
2. Our algorithm.
3. Time complexity analysis.
4. Conclusions.

Introduction

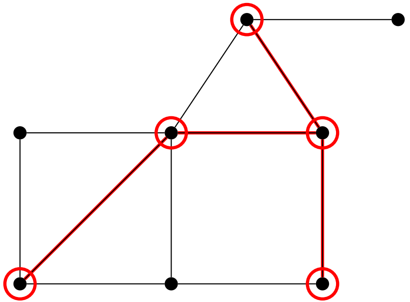
Introduction - definitions

Deterministic parameterized connected vertex cover.



Introduction - definitions

Deterministic parameterized connected vertex cover.



Introduction - definitions

Deterministic parameterized connected vertex cover.

- A parameterized problem instance comes with an additional integer (G, k).
- A problem is FPT if it admits an algorithm with $f(k)\mathrm{poly}(n)$ running time.
- Goal: for problems known to be FPT design the fastest algorithm possible.
- We are interested in the best possible function f and as $O^*(f(k))$ denote $O(f(k)\mathrm{poly}(n))$.

Introduction - history

CVC problem def.

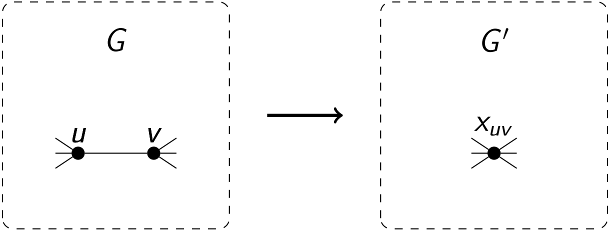
Given an undirected graph $G = (V, E)$ and an integer k, decide whether there exists a connected vertex cover of G of cardinality at most k?

$O^*(6^k)$	GNW'05
$O^*(3.2361^k)$	MRR'06
$O^*(2.9316^k)$	FM'06
$O^*(2.7606^k)$	MRR'08
$O^*(2.4882^k)$	B'10
$O^*(2^k)$(randomized)	CNPPRW'11
$O^*(2^k)$	this paper

Algorithm

Algorithm

CVC is contraction closed, i.e., if (G, k) is a YES-instance than (G', k) is a YES-instance.



Algorithm

We use the *iterative compression* technique.

- Consider any edge uv of G.

Algorithm

We use the *iterative compression* technique.

- Consider any edge uv of G.
- Solve the problem for G' with u and v identified into x.

Algorithm

We use the *iterative compression* technique.

- Consider any edge uv of G.
- Solve the problem for G' with u and v identified into x.
- If (G', k) is NO-instance, return NO.

Algorithm

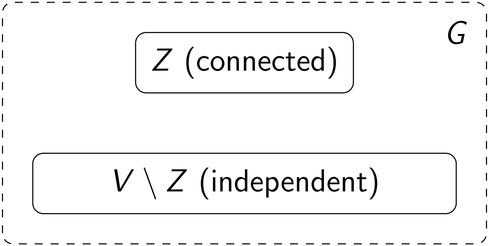
We use the *iterative compression* technique.

- Consider any edge uv of G.
- Solve the problem for G' with u and v identified into x.
- If (G', k) is NO-instance, return NO.
- If X' is cvc of G', then $Z := (X' \setminus \{x\}) \cup \{u, v\}$ is cvc of G.

Algorithm

We use the *iterative compression* technique.

- Consider any edge uv of G.
- Solve the problem for G' with u and v identified into x.
- If (G', k) is NO-instance, return NO.
- If X' is cvc of G', then $Z := (X' \setminus \{x\}) \cup \{u, v\}$ is cvc of G.
- Use Z to exploit the structure of G.

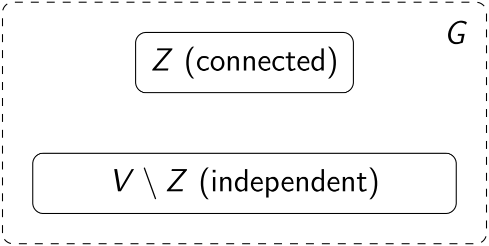


Algorithm

By a factor n (can be reduced to $2k$), it is enough to solve:

Compression CVC

Given $G = (V, E)$, k and a cvc $Z \subseteq V$ of size at most $k + 2$ find cvc of G of size at most k.



Algorithm

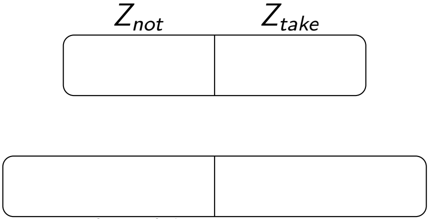
- Guess (by trying $2^{|Z|}$ possibilities) subset of Z used by solution.

Algorithm

- Guess (by trying $2^{|Z|}$ possibilities) subset of Z used by solution.
- $Z = Z_{not} \cup Z_{take}$, where Z_{not} is independent.

Algorithm

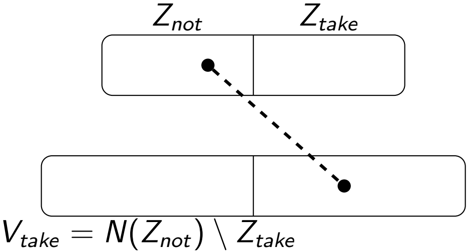
- Guess (by trying $2^{|Z|}$ possibilities) subset of Z used by solution.
- $Z = Z_{not} \cup Z_{take}$, where Z_{not} is independent.
- Define V_{take} as vertices of $V \setminus Z$ with $\geqslant 1$ neighbour in Z_{not}.



$$V_{take} = N(Z_{not}) \setminus Z_{take}$$

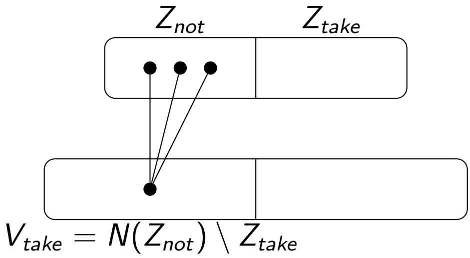
Algorithm

- Guess (by trying $2^{|Z|}$ possibilities) subset of Z used by solution.
- $Z = Z_{not} \cup Z_{take}$, where Z_{not} is independent.
- Define V_{take} as vertices of $V \setminus Z$ with $\geqslant 1$ neighbour in Z_{not}.
- $V_{take} \cup Z_{take}$ form a vc of G.



Algorithm

- Guess (by trying $2^{|Z|}$ possibilities) subset of Z used by solution.
- $Z = Z_{not} \cup Z_{take}$, where Z_{not} is independent.
- Define V_{take} as vertices of $V \setminus Z$ with $\geqslant 1$ neighbour in Z_{not}.
- $V_{take} \cup Z_{take}$ form a vc of G.
- If a vertex of V_{take} has no neighbor in Z_{take}, then terminate the branch.



Algorithm

- Since $X_0 := V_{take} \cup Z_{take}$ is already a vc of G it remains to find the smallest cardinality set $X_1 \subseteq V_{maybe} := V \setminus (Z \cup V_{take})$, such that $G[X_0 \cup X_1]$ is connected.

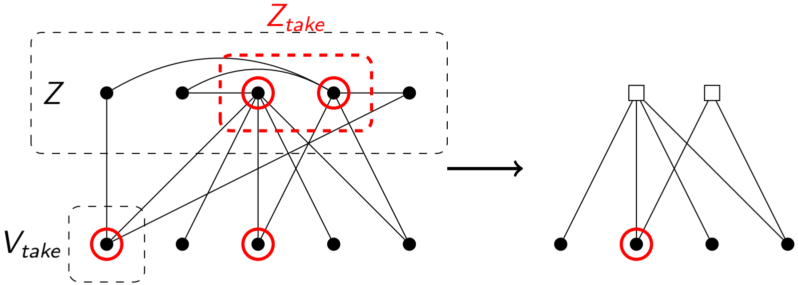
Algorithm

- Since $X_0 := V_{take} \cup Z_{take}$ is already a vc of G it remains to find the smallest cardinality set $X_1 \subseteq V_{maybe} := V \setminus (Z \cup V_{take})$, such that $G[X_0 \cup X_1]$ is connected.
- This is a Steiner tree problem, where as terminals we take contracted connected components of $G[X_0]$.

Algorithm

- Since $X_0 := V_{take} \cup Z_{take}$ is already a vc of G it remains to find the smallest cardinality set $X_1 \subseteq V_{maybe} := V \setminus (Z \cup V_{take})$, such that $G[X_0 \cup X_1]$ is connected.
- This is a Steiner tree problem, where as terminals we take contracted connected components of $G[X_0]$.
- Therefore we can find X_1 in $O^*(2^{\mathrm{cc}(G[X_0])})$ time by using algorithm of Nederlof for Steiner tree (or dynamic programming over subsets).

Algorithm - example



Complexity analysis

Complexity analysis

- For each subset $Z_{take} \subseteq Z$ such that $Z \setminus Z_{take}$ is independent we have $O^*(2^z)$ running time, where $z = \mathrm{cc}(G[Z_{take}])$.

Complexity analysis

- For each subset $Z_{take} \subseteq Z$ such that $Z \setminus Z_{take}$ is independent we have $O^*(2^z)$ running time, where $z = \mathrm{cc}(G[Z_{take}])$.
- The running time can be upper bounded by the cardinality of

$$P := \{(Z_{take}, \mathcal{C}) : Z_{take} \text{ is vc of G[Z]}, \mathcal{C} \subseteq \mathrm{CC}(G[Z_{take}])\}$$

Complexity analysis

- For each subset $Z_{take} \subseteq Z$ such that $Z \setminus Z_{take}$ is independent we have $O^*(2^z)$ running time, where $z = \mathrm{cc}(G[Z_{take}])$.
- The running time can be upper bounded by the cardinality of

$$P := \{(Z_{take}, \mathcal{C}) : Z_{take} \text{ is vc of G[Z]}, \mathcal{C} \subseteq \mathrm{CC}(G[Z_{take}])\}$$

- It is easy to show $3^{|Z|}$ upper bound, since each vertex of Z can be (i) not taken to Z_{take}, (ii) taken and its cc belongs to $\mathcal{C}$, (iii) taken and its cc does not belong to $\mathcal{C}$.

Complexity analysis

- For each subset $Z_{take} \subseteq Z$ such that $Z \setminus Z_{take}$ is independent we have $O^*(2^z)$ running time, where $z = \mathrm{cc}(G[Z_{take}])$.
- The running time can be upper bounded by the cardinality of

$$P := \{(Z_{take}, \mathcal{C}) : Z_{take} \text{ is vc of G[Z]}, \mathcal{C} \subseteq \mathrm{CC}(G[Z_{take}])\}$$

- It is easy to show $3^{|Z|}$ upper bound, since each vertex of Z can be (i) not taken to Z_{take}, (ii) taken and its cc belongs to $\mathcal{C}$, (iii) taken and its cc does not belong to $\mathcal{C}$.
- Observe that knowing the type of each vertex of Z gives us at most one corresponding pair of P.

Complexity analysis

- We want to show $3 \cdot 2^{|Z|-1}$ upper bound on $|P|$.

Complexity analysis

- We want to show $3 \cdot 2^{|Z|-1}$ upper bound on $|P|$.
- Consider any spanning tree T of $G[Z]$ and root it in an arbitrary vertex.

Complexity analysis

- For the root we have three choices, as previously: (i) not taken to Z_{take}, (ii) taken and its cc belongs to $\mathcal{C}$, (iii) taken and its cc does not belong to $\mathcal{C}$.

Complexity analysis

- For the root we have three choices, as previously: (i) not taken to Z_{take}, (ii) taken and its cc belongs to $\mathcal{C}$, (iii) taken and its cc does not belong to $\mathcal{C}$.
- Consider any non-root node v of T and let p be its parent.

Complexity analysis

- For the root we have three choices, as previously: (i) not taken to Z_{take}, (ii) taken and its cc belongs to $\mathcal{C}$, (iii) taken and its cc does not belong to $\mathcal{C}$.
- Consider any non-root node v of T and let p be its parent.
- If p is (i), then v cannot be (i), because Z_{take} is vc in $G[Z]$.

Complexity analysis

- For the root we have three choices, as previously: (i) not taken to Z_{take}, (ii) taken and its cc belongs to $\mathcal{C}$, (iii) taken and its cc does not belong to $\mathcal{C}$.
- Consider any non-root node v of T and let p be its parent.
- If p is (i), then v cannot be (i), because Z_{take} is vc in $G[Z]$.
- If p is (ii), then v cannot be (iii), as they cannot be in two different components of $\mathcal{C}$.

Complexity analysis

- For the root we have three choices, as previously: (i) not taken to Z_{take}, (ii) taken and its cc belongs to $\mathcal{C}$, (iii) taken and its cc does not belong to $\mathcal{C}$.
- Consider any non-root node v of T and let p be its parent.
- If p is (i), then v cannot be (i), because Z_{take} is vc in $G[Z]$.
- If p is (ii), then v cannot be (iii), as they cannot be in two different components of $\mathcal{C}$.
- Similarly if p is (iii), then v cannot be (ii).

Complexity analysis

- For the root we have three choices, as previously: (i) not taken to Z_{take}, (ii) taken and its cc belongs to $\mathcal{C}$, (iii) taken and its cc does not belong to $\mathcal{C}$.
- Consider any non-root node v of T and let p be its parent.
- If p is (i), then v cannot be (i), because Z_{take} is vc in $G[Z]$.
- If p is (ii), then v cannot be (iii), as they cannot be in two different components of $\mathcal{C}$.
- Similarly if p is (iii), then v cannot be (ii).
- This gives $3 \cdot 2^{|Z|-1}$ upper bound on $|P|$ and since $|Z| \leqslant k+2$ we have $O^*(2^k)$ algorithm for CVC.

Conclusions

Conclusions and open problems

1. We have shown how to solve CVC deterministically in $O^*(2^k)$ time.
2. Our algorithm can be extended to weighted and counting variants.
3. By recent work [CDLMNOPSW'12], one can not solve the counting variant in $0^*((2-\varepsilon)^k)$ unless SETH fails.
4. Open problem: is it possible to show that there is no $0^*((2-\varepsilon)^k)$ algorithm for the decision version unless SETH fails?

Questions?

Thank you!