Graph Theory and Optimization
Parameterized Algorithms

Nicolas Nisse

Université Côte d'Azur, Inria, CNRS, I3S, France

October 2018

What is it about?

- Goal:
  - Find "efficient" exact algorithms for difficult problems (NP-hard).
  - For some (NP-hard) problems, the difficulty is not due to the size of the input, but to...
    the structure of the input, the size of the solution...
- Introduction to Parameterized Algorithms through Vertex Cover

A very nice book:

M. Cygan, F.V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh:

Parameterized Algorithms. Springer 2015, ISBN 978-3-319-21274-6, pp. 3-555

Outline

Reminder on Minimum Vertex Cover

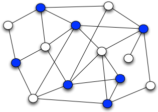
Let $G = (V, E)$ be a graph

Vertex Cover: set $K \subseteq V$ such that $\forall e \in E$, $e \cap K \neq \emptyset$

set of vertices that "touch" every edge

Finding a Vertex Cover of minimum size is "difficult"

Compute a Min. Vertex Cover is NP-complete [Garey,Johnson 1979]



Example of vertex cover of size 7 (in blue)

Reminder on Minimum Vertex Cover

Let $G = (V, E)$ be a graph

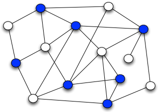
Vertex Cover: set $K \subseteq V$ such that $\forall e \in E$, $e \cap K \neq \emptyset$

set of vertices that "touch" every edge

Finding a Vertex Cover of minimum size is "difficult"

Compute a Min. Vertex Cover is NP-complete

[Garey,Johnson 1979]



Example of vertex cover of size 7 (in blue)

Exercise: Give an algorithm for computing a min. Vertex Cover in a graph

Reminder on Minimum Vertex Cover

Let $G = (V,E)$ be a graph

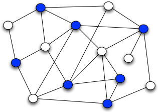
Vertex Cover: set $K \subseteq V$ such that $\forall e \in E$, $e \cap K \neq \emptyset$

set of vertices that "touch" every edge

Finding a Vertex Cover of minimum size is "difficult"

Compute a Min. Vertex Cover is NP-complete

[Garey,Johnson 1979]



Example of vertex cover of size 7 (in blue)

Exercise: Give an algorithm for computing a min. Vertex Cover in a graph

Naive Exact Algo. for Min. Vertex Cover

input: graph $G = (V,E)$

For $k = 1$ to $|V| - 1$ **do**

  For every set $S \subseteq V$ of size k **do**

    If S is a vertex cover of G

    then Return S

Reminder on Minimum Vertex Cover

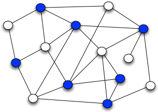
Let $G=(V,E)$ be a graph

Vertex Cover: set $K \subseteq V$ such that $\forall e \in E,\ e \cap K \neq \emptyset$

set of vertices that "touch" every edge

Finding a Vertex Cover of minimum size is "difficult"

Compute a Min. Vertex Cover is NP-complete [Garey,Johnson 1979]



Example of vertex cover of size 7 (in blue)

Exercise: Give an algorithm for computing a min. Vertex Cover in a graph

Naive Exact Algo. for Min. Vertex Cover

input: graph $G=(V,E)$

For $k=1$ to $|V|-1$ **do**
- **For** every set $S \subseteq V$ of size k **do**
  - **If** S is a vertex cover of G **then** Return S

Time-complexity: $O(2^{|V|}|E|)$

$2^{|V|}$: number of subsets of vertices

$O(|E|)$: time to check if vertex cover

$\Rightarrow$ Exponential in the size of the graph.

Toward "polynomial" algorithms

Complexity of deciding if a graph has a vertex cover of size 1? of size 2?...

Toward “polynomial" algorithms

Complexity of deciding if a graph has a vertex cover of size 1? of size 2?...

Exercise: Let $k \in \mathbb{N}$ be a fixed integer
Give an algorithm for deciding if a graph has a vertex cover of size k
What is its complexity?

Toward “polynomial" algorithms

Complexity of deciding if a graph has a vertex cover of size 1? of size 2?...

Exercise: Let $k \in \mathbb{N}$ be a fixed integer
Give an algorithm for deciding if a graph has a vertex cover of size k
What is its complexity?

Algorithm 1 for fixed k

fixed parameter: $k \in \mathbb{N}$
input: graph $G = (V,E)$
For every set $S \subseteq V$ of size k **do**

If S is a vertex cover of G

then *Return* S

Return “No vertex cover of size $\leq k$"

Toward “polynomial" algorithms

Complexity of deciding if a graph has a vertex cover of size 1? of size 2?...

Exercise: Let $k \in \mathbb{N}$ be a fixed integer
Give an algorithm for deciding if a graph has a vertex cover of size k
What is its complexity?

Algorithm 1 for fixed k

fixed parameter: $k \in \mathbb{N}$
input: graph $G = (V, E)$
For every set $S \subseteq V$ of size k **do**

  If S is a vertex cover of G

  then *Return* S

Return “No vertex cover of size $\leq k$"

Time-complexity: $O(|V|^k|E|)$

$|V|^k$: $\#$ of subsets of vertices of size k
$O(|E|)$: time to check if vertex cover

$\Rightarrow$ Polynomial in the size of the graph.

Toward “polynomial" algorithms

Complexity of deciding if a graph has a vertex cover of size 1? of size 2?...

Exercise: Let $k \in \mathbb{N}$ be a fixed integer
Give an algorithm for deciding if a graph has a vertex cover of size k
What is its complexity?

Algorithm 1 for fixed k

fixed parameter: $k \in \mathbb{N}$
input: graph $G = (V, E)$
For every set $S \subseteq V$ of size k **do**

  If S is a vertex cover of G

  then *Return* S

Return “No vertex cover of size $\leq k$"

Time-complexity: $O(|V|^k|E|)$

$|V|^k$: $\#$ of subsets of vertices of size k
$O(|E|)$: time to check if vertex cover

$\Rightarrow$ Polynomial in the size of the graph.

Remark: the algorithm is still exponential (in the size k of the solution)

Outline

Vertex Cover of size $\leq k$? Two simple Lemmas

$G = (V, E)$ be a graph $vc(G)$= min. size of a vertex cover in G

Lemma 1: $vc(G) \leq k \Rightarrow |E| \leq k(|V| - 1)$

small vertex cover $\Rightarrow$ “few" edges

Vertex Cover of size $\leq k$? Two simple Lemmas

$G = (V, E)$ be a graph $vc(G)$= min. size of a vertex cover in G

Lemma 1: $vc(G) \leq k \Rightarrow |E| \leq k(|V| - 1)$

small vertex cover $\Rightarrow$ “few" edges

proof: Let $S \subseteq V$ be any vertex cover of size at most k. Each vertex of S covers at most $|V| - 1$ edges. Each edge must be covered.

Vertex Cover of size $\leq k$? Two simple Lemmas

$G = (V, E)$ be a graph $vc(G)$= min. size of a vertex cover in G

Lemma 1: $vc(G) \leq k \Rightarrow |E| \leq k(|V| - 1)$

small vertex cover $\Rightarrow$ "few" edges

proof: Let $S \subseteq V$ be any vertex cover of size at most k. Each vertex of S covers at most $|V| - 1$ edges. Each edge must be covered.

Lemma 2: Let $\{x, y\} \in E$. $vc(G) = \min\{vc(G \setminus x), vc(G \setminus y)\} + 1$

"for any edge xy, any minimum vertex cover contains at least one of x or y..."

Vertex Cover of size $\leq k$? Two simple Lemmas

$G = (V, E)$ be a graph $vc(G)$= min. size of a vertex cover in G

Lemma 1: $vc(G) \leq k \Rightarrow |E| \leq k(|V| - 1)$

small vertex cover $\Rightarrow$ "few" edges

proof: Let $S \subseteq V$ be any vertex cover of size at most k. Each vertex of S covers at most $|V| - 1$ edges. Each edge must be covered.

Lemma 2: Let $\{x, y\} \in E$. $vc(G) = \min\{vc(G \setminus x), vc(G \setminus y)\} + 1$

"for any edge xy, any minimum vertex cover contains at least one of x or y..."

proof:

- Let $S \subseteq V$ be any vertex cover of $G \setminus x$. Then $S \cup \{x\}$ is a vertex cover of G. Hence $vc(G) \leq vc(G \setminus x) + 1$ (symmetrically for $G \setminus y$)
- Let $S \subseteq V$ be any vertex cover of G. At least one of x or y is in S.
  If $x \in S$ then $S \setminus x$ vertex cover of $G \setminus x$. Hence $vc(G \setminus x) \leq vc(G) - 1$.
  Otherwise, if $y \in S$, then $S \setminus y$ vertex cover of $G \setminus y$ and $vc(G \setminus y) \leq vc(G) - 1$.

Vertex Cover of size $\leq k$? First FPT algorithm

Lemma 2 proves the correctness of the following algorithm

Rec: Branch & Bound Algorithm for computing Minimum size Vertex Cover

input: graph $G = (V, E)$

If $|E| = 0$, *Return* 0. **Else if** $|E| = 1$, *Return* 1

Else Let $\{x, y\} \in E$, let $A = \textbf{Rec}(G \setminus x)$, $B = \textbf{Rec}(G \setminus y)$, *Return* $\min\{A, B\} + 1$

Vertex Cover of size $\leq k$? First FPT algorithm

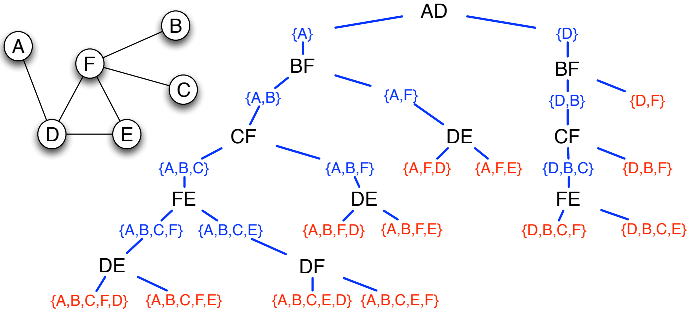
Lemma 2 proves the correctness of the following algorithm

Rec: Branch & Bound Algorithm for computing Minimum size Vertex Cover

input: graph $G = (V, E)$

If $|E| = 0$, *Return* 0. **Else if** $|E| = 1$, *Return* 1

Else Let $\{x, y\} \in E$, let $A = \textbf{Rec}(G \setminus x)$, $B = \textbf{Rec}(G \setminus y)$, *Return* $\min\{A, B\} + 1$



Binary tree of depth $O(|V|)$. Complexity: $O(2^{|V|}|E|)$.

Vertex Cover of size $\leq k$? First FPT algorithm

Lemma 2 proves the correctness of the following algorithm

Rec: Branch & Bound Algorithm for computing Minimum size Vertex Cover

input: graph $G = (V, E)$

If $|E| = 0$, *Return* 0. **Else if** $|E| = 1$, *Return* 1

Else Let $\{x, y\} \in E$, let $A = \mathbf{Rec}(G \setminus x)$, $B = \mathbf{Rec}(G \setminus y)$, *Return* $\min\{A, B\} + 1$

Question: $vc(G) \leq k$? limit recursion-depth + limit $\#$ of edges (Lemma 1)

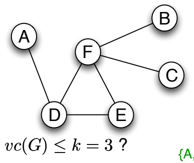
Vertex Cover of size $\leq k$? First FPT algorithm

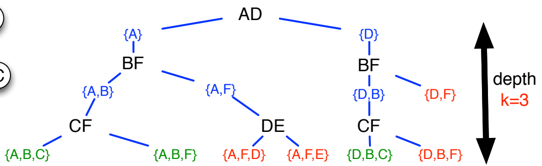
Question: $vc(G) \leq k$? limit recursion-depth + limit $\#$ of edges (Lemma 1)

Alg2: Branch & Bound Algorithm for deciding if $vc(G) \leq k$

input: graph $G = (V, E)$, integer $\ell \leq k$.

If $|E| > 0$ and $\ell = 0$, *Return* ∞. **Else if** $|E| = 0$, *Return* 0.
Else if $|E| = 1$, *Return* 1
Else Let $\{x, y\} \in E$, let $A =$ **Alg2**$(G \setminus x, \ell - 1)$, $B =$ **Alg2**$(G \setminus y, \ell - 1)$, *Return* $\min\{A, B\} + 1$





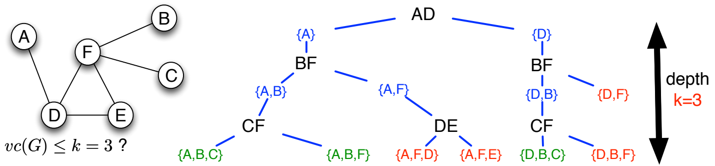
Vertex Cover of size $\leq k$? First FPT algorithm

Question: $vc(G) \leq k$? limit recursion-depth + limit # of edges (Lemma 1)

Alg2: Branch & Bound Algorithm for deciding if $vc(G) \leq k$

input: graph $G = (V, E)$, integer $\ell \leq k$.

If $|E| > 0$ and $\ell = 0$, *Return* ∞. **Else if** $|E| = 0$, *Return* 0.
Else if $|E| = 1$, *Return* 1
Else Let $\{x, y\} \in E$, let $A = \textbf{Alg2}(G \setminus x, \ell - 1)$, $B = \textbf{Alg2}(G \setminus y, \ell - 1)$, *Return* $\min\{A, B\} + 1$



Binary tree of depth $O(k)$: Complexity: $O(2^k|E|)$. By Lem. 1, $|E| = O(k|V|)$,

Alg2 decides if $vc(G) \leq k$ in time $O(2^k \cdot k|V|)$ (linear in $|G|$)
$|V|$ and k are "separated" $\Rightarrow$ Fixed Parameterized Tractable (FPT)

Outline

Parameterized Complexity in brief

Parameterized Problem

A parameterized problem is a language $L \subseteq \Sigma^* \times \Sigma^*$, where Σ is a finite alphabet. The first component corresponds to the input. The second component is called the parameter of the problem.

Class FPT

A parameterized problem is fixed-parameter tractable (FPT) if it can be determined in time $f(k) \cdot |x|^{O(1)}$ whether $(x, k) \in L$, where f is a computable function only depending on k.
The corresponding complexity class is called FPT.

In other words:

Given a (NP-hard) problem with input of size n and a parameter k, a **FPT algorithm** runs in time $f(k) \cdot n^{O(1)}$ for some computable function f.

Examples: k-Vertex Cover, k-Longest Path...

Outline

Vertex Cover of size $\leq k$? Data Reduction

A general GOOD idea: Data reduction

Find simple rules to reduce the size of the input

From input G, compute (in polynomial-time) another instance G' s.t.

$|G'| < |G|$ and a solution for G can be deduced from a solution for G'.

Hence, it is sufficient to solve the problem on the (smaller) instance G'

Back to k-Vertex Cover:

Lemma 3: Let $G = (V, E)$ and $v \in V$ with degree $> k$.
Then v belongs to any vertex cover S of size at most k

Rule: If G has a vertex v of degree $> k$, $vc(G) \leq k \Leftrightarrow vc(G \setminus v) \leq k - 1$.

Lemma 4: $G = (V, E)$. If $vc(G) \leq k$ and no vertex of degree $> k$
Then $|E| \leq k^2$

Vertex Cover of size $\leq k$? Data Reduction

A general GOOD idea: Data reduction

Find simple rules to reduce the size of the input

From input G, compute (in polynomial-time) another instance G' s.t.

$|G'| < |G|$ and a solution for G can be deduced from a solution for G'.

Hence, it is sufficient to solve the problem on the (smaller) instance G'

Back to k-Vertex Cover:

Lemma 3: Let $G = (V, E)$ and $v \in V$ with degree $> k$. Then v belongs to any vertex cover S of size at most k

Rule: If G has a vertex v of degree $> k$, $vc(G) \leq k \Leftrightarrow vc(G \setminus v) \leq k - 1$.

Lemma 4: $G = (V, E)$. If $vc(G) \leq k$ and no vertex of degree $> k$ Then $|E| \leq k^2$

Vertex Cover of size $\leq k$? Data Reduction

A general GOOD idea: Data reduction

Find simple rules to reduce the size of the input

From input G, compute (in polynomial-time) another instance G' s.t.

$|G'| < |G|$ and a solution for G can be deduced from a solution for G'.

Hence, it is sufficient to solve the problem on the (smaller) instance G'

Back to k-Vertex Cover:

Lemma 3: Let $G = (V, E)$ and $v \in V$ with degree $> k$.
Then v belongs to any vertex cover S of size at most k

proof: Indeed, if $v \notin S$, all its neighbors must belong to it and $|S| > k$.

Rule: If G has a vertex v of degree $> k$, $vc(G) \leq k \Leftrightarrow vc(G \setminus v) \leq k - 1$.

Lemma 4: $G = (V, E)$. If $vc(G) \leq k$ and no vertex of degree $> k$
Then $|E| \leq k^2$

Vertex Cover of size $\leq k$? Data Reduction

A general GOOD idea: Data reduction

Find simple rules to reduce the size of the input

From input G, compute (in polynomial-time) another instance G' s.t.

$|G'| < |G|$ and a solution for G can be deduced from a solution for G'.

Hence, it is sufficient to solve the problem on the (smaller) instance G'

Back to k-Vertex Cover:

Lemma 3: Let $G = (V, E)$ and $v \in V$ with degree $> k$.
Then v belongs to any vertex cover S of size at most k

proof: Indeed, if $v \notin S$, all its neighbors must belong to it and $|S| > k$.

Rule: If G has a vertex v of degree $> k$, $vc(G) \leq k \Leftrightarrow vc(G \setminus v) \leq k - 1$.

Lemma 4: $G = (V, E)$. If $vc(G) \leq k$ and no vertex of degree $> k$
Then $|E| \leq k^2$

Vertex Cover of size $\leq k$? Data Reduction

A general GOOD idea: Data reduction

Find simple rules to reduce the size of the input

From input G, compute (in polynomial-time) another instance G' s.t.

$|G'| < |G|$ and a solution for G can be deduced from a solution for G'.

Hence, it is sufficient to solve the problem on the (smaller) instance G'

Back to k-Vertex Cover:

Lemma 3: Let $G = (V, E)$ and $v \in V$ with degree $> k$. Then v belongs to any vertex cover S of size at most k

proof: Indeed, if $v \notin S$, all its neighbors must belong to it and $|S| > k$.

Rule: If G has a vertex v of degree $> k$, $vc(G) \leq k \Leftrightarrow vc(G \setminus v) \leq k - 1$.

Lemma 4: $G = (V, E)$. If $vc(G) \leq k$ and no vertex of degree $> k$ Then $|E| \leq k^2$

proof: Each of the $\leq k$ vertices of a Vertex Cover covers at most k edges.

Vertex Cover of size $\leq k$? First Kernelization algorithm

Alg3: Kernelization Algorithm for deciding if $vc(G) \leq k$

input: graph $G = (V, E)$, integer $\ell \leq k$.

Remove isolated vertices
If $|E| = 0$, *Return TRUE*. **Else if** $\ell = 0$, *Return FALSE*
Else if no vertex of degree $> \ell$ and $|V| > \ell^2$, *Return FALSE*
Else if no vertex of degree $> \ell$, Apply $Alg2(G, \ell)$
Else Let v be a vertex of degree $> \ell$. Apply $Alg3(G \setminus v, \ell - 1)$.

While there is a “high" degree node, add it to the solution. When there are no such nodes, either it remains too much edges to have a small vertex cover. Otherwise, apply brute force algorithm (e.g., $Alg2$) to the remaining “small" graph

Time-complexity: $O(2^k \cdot k^2 + |V| \cdot k)$ (It is a FPT algorithm!!)

$O(|V| \cdot k)$: find at most k vertices of “high" degree **Reduction Rule**
$O(2^k \cdot k^2)$: application of $Alg2$ to a graph with $O(k^2)$ edges **“Brute Force”**

Kernelization: Apply reduction rule(s) until the instance has constant (only dependent on k) size. Then apply “brute force"

Vertex Cover of size $\leq k$? First Kernelization algorithm

Alg3: Kernelization Algorithm for deciding if $vc(G) \leq k$

input: graph $G = (V, E)$, integer $\ell \leq k$.

Remove isolated vertices
If $|E| = 0$, *Return TRUE*. **Else if** $\ell = 0$, *Return FALSE*
Else if no vertex of degree $> \ell$ and $|V| > \ell^2$, *Return FALSE*
Else if no vertex of degree $> \ell$, Apply $Alg2(G, \ell)$
Else Let v be a vertex of degree $> \ell$. Apply $Alg3(G \setminus v, \ell - 1)$.

While there is a "high" degree node, add it to the solution. When there are no such nodes, either it remains too much edges to have a small vertex cover. Otherwise, apply brute force algorithm (e.g., $Alg2$) to the remaining "small" graph

Time-complexity: $O(2^k \cdot k^2 + |V| \cdot k)$ (It is a FPT algorithm!!)

$O(|V| \cdot k)$: find at most k vertices of "high" degree **Reduction Rule**
$O(2^k \cdot k^2)$: application of $Alg2$ to a graph with $O(k^2)$ edges **"Brute Force"**

Kernelization: Apply reduction rule(s) until the instance has constant (only dependent on k) size. Then apply "brute force"

Vertex Cover of size $\leq k$? First Kernelization algorithm

Alg3: Kernelization Algorithm for deciding if $vc(G) \leq k$

input: graph $G=(V,E)$, integer $\ell \leq k$.

Remove isolated vertices
If $|E|=0$, *Return TRUE*. **Else if** $\ell=0$, *Return FALSE*
Else if no vertex of degree $>\ell$ and $|V|>\ell^2$, *Return FALSE*
Else if no vertex of degree $>\ell$, Apply $Alg2(G,\ell)$
Else Let v be a vertex of degree $>\ell$. Apply $Alg3(G\setminus v,\ell-1)$.

While there is a “high" degree node, add it to the solution. When there are no such nodes, either it remains too much edges to have a small vertex cover. Otherwise, apply brute force algorithm (e.g., *Alg*2) to the remaining “small" graph

Time-complexity: $O(2^k \cdot k^2 + |V| \cdot k)$ (It is a FPT algorithm!!)

$O(|V| \cdot k)$: find at most k vertices of “high" degree **Reduction Rule**
$O(2^k \cdot k^2)$: application of *Alg*2 to a graph with $O(k^2)$ edges **“Brute Force"**

Kernelization: Apply reduction rule(s) until the instance has constant (only dependent on k) size. Then apply “brute force"

Vertex Cover Comparison of previous algorithms

Problem: Let $k \in \mathbb{N}$ be a fixed integer. Given $G = (V, E)$, $vc(G) \leq k$?

	time-complexity	numerical example $\lvert V\rvert = 10^4$ and $k = 10$
brute-force for Min. Vertex Cover	$O(\lvert E\rvert \cdot 2^{\lvert V\rvert})$	$>> 10^{3000}$
brute-force, k fixed (*Alg*1)	$O(\lvert E\rvert\lvert V\rvert^k)$	10^{48}
bounded Branch & Bound (*Alg*2)	$O(2^k \cdot k\lvert V\rvert)$	10^8
first kernelization (*Alg*3)	$O(2^k \cdot k^2 + k\lvert V\rvert)$	$2 \cdot 10^5$

Outline

Kernelization

Problem Kernel

Let L be a parameterized problem, that is, L consists of (I, k), where I is the problem instance and k is the parameter.
Reduction to a problem kernel then means to replace instance (I, k) by a "reduced" instance (I', k') (called problem kernel) such that

1. $k' \leq k$, $|I'| \leq g(k)$ for some function g only depending on k,
2. $(I, k) \in L$ if and only if $(I', k') \in L$, and
3. reduction from (I, k) to (I', k') has to be computable in polynomial time.

A *Kernelization* algorithm consists in

1. reduce the size of the instance I in time polynomial in $|I| = n$
2. solve the problem on the reduced instance I' with size $O(g(k))$

Time-complexity: $O(f(g(k)) + n^{O(1)})$
where function f is the time-complexity for solving the problem on I' (e.g., brute force)

It is a FPT algorithm!!

Kernelization

Problem Kernel

Let L be a parameterized problem, that is, L consists of (I, k), where I is the problem instance and k is the parameter.
Reduction to a problem kernel then means to replace instance (I, k) by a "reduced" instance (I', k') (called problem kernel) such that

1. $k' \leq k$, $|I'| \leq g(k)$ for some function g only depending on k,
2. $(I, k) \in L$ if and only if $(I', k') \in L$, and
3. reduction from (I, k) to (I', k') has to be computable in polynomial time.

A *Kernelization* algorithm consists in

1. reduce the size of the instance I in time polynomial in $|I| = n$
2. solve the problem on the reduced instance I' with size $O(g(k))$

Time-complexity: $O(f(g(k)) + n^{O(1)}$
where function f is the time-complexity for solving the problem on I' (e.g., brute force)

It is a FPT algorithm!!

FPT vs. Kernelization

Theorem: [Bodlaender et al. 2009]

A parameterized problem is FPT if and only if
it is decidable and has a kernelization algorithm.

proof: $\Leftarrow$ see previous slide ("decidable" implies that function f exists)
$\Rightarrow$ Kernelization: Apply the FPT algorithm. The kernel is the answer $\in \{YES, NO\}$.

It is desirable (if possible) to compute "small" kernel, e.g.,

- linear kernel $g(k) = O(k)$
- quadratic kernel $g(k) = O(k^2)$

Example: *Alg*3 for Vertex Cover

- ...

In what follows: kernelization algorithm for Vertex Cover with linear kernel

FPT vs. Kernelization

Theorem: [Bodlaender et al. 2009]

A parameterized problem is FPT if and only if
it is decidable and has a kernelization algorithm.

proof: $\Leftarrow$ see previous slide (“decidable" implies that function f exists)
$\Rightarrow$ Kernelization: Apply the FPT algorithm. The kernel is the answer $\in \{YES, NO\}$.

It is desirable (if possible) to compute “small" kernel, e.g.,

- linear kernel $g(k) = O(k)$
- quadratic kernel $g(k) = O(k^2)$
  Example: *Alg*3 for Vertex Cover
- ...

In what follows: kernelization algorithm for Vertex Cover with linear kernel

FPT vs. Kernelization

Theorem: [Bodlaender et al. 2009]

A parameterized problem is FPT if and only if
it is decidable and has a kernelization algorithm.

proof: $\Leftarrow$ see previous slide (“decidable" implies that function f exists)
$\Rightarrow$ Kernelization: Apply the FPT algorithm. The kernel is the answer $\in \{YES, NO\}$.

It is desirable (if possible) to compute “small" kernel, e.g.,

- linear kernel $g(k) = O(k)$
- quadratic kernel $g(k) = O(k^2)$
  Example: $Alg3$ for Vertex Cover
- ...

In what follows: kernelization algorithm for Vertex Cover with linear kernel

FPT vs. Kernelization

Theorem: [Bodlaender et al. 2009]

A parameterized problem is FPT if and only if it is decidable and has a kernelization algorithm.

proof: $\Leftarrow$ see previous slide ("decidable" implies that function f exists)
$\Rightarrow$ Kernelization: Apply the FPT algorithm. The kernel is the answer $\in \{YES, NO\}$.

It is desirable (if possible) to compute "small" kernel, e.g.,

- linear kernel $g(k) = O(k)$
- quadratic kernel $g(k) = O(k^2)$

  Example: $Alg3$ for Vertex Cover
- ...

In what follows: kernelization algorithm for Vertex Cover with linear kernel

Outline

Back to Fractional Relaxation for Vertex Cover

Let $G=(V,E)$ be a graph

Integer Linear programme (ILP) for Vertex Cover:

$$\begin{array}{lrcll} \text{Min.} & \sum_{v\in V} x_v & & & \\ \text{s.t.:} & x_v + x_u & \geq & 1 & \forall \{u,v\} \in E \\ & x_v & \in & \{0,1\} & \forall v \in V \end{array}$$

Fractional relaxation (LP) for Vertex Cover:

$$\begin{array}{lrcll} \text{Min.} & \sum_{v\in V} x_v & & & \\ \text{s.t.:} & x_v + x_u & \geq & 1 & \forall \{u,v\} \in E \\ & x_v & \geq & 0 & \forall v \in V \end{array}$$

Theorem: From Fractional to Integral Solution

Let $(x_v)_{v\in V}$ be a fractional optimal solution.
$V_0 = \{v \in V \mid x_v < 1/2\}$, $V_1 = \{v \in V \mid x_v > 1/2\}$ and $V_{1/2} = \{v \in V \mid x_v = 1/2\}$
There exists a Minimum (Integral) vertex cover S such that $V_1 \subseteq S \subseteq V_1 \cup V_{1/2}$

Corollary: reduction Rule using LP for Vertex Cover

Let $(x_v)_{v\in V}$ be a fractional optimal solution.
Then $vc(G) \leq k$ if and only if $vc(G \setminus V_1) \leq k - |V_1|$.

Back to Fractional Relaxation for Vertex Cover

Let $G = (V, E)$ be a graph

Integer Linear programme (*ILP*) for Vertex Cover:

$$\begin{array}{llcll} \text{Min.} & \sum_{v \in V} x_v & & & \\ \text{s.t.:} & x_v + x_u & \geq & 1 & \forall \{u,v\} \in E \\ & x_v & \in & \{0,1\} & \forall v \in V \end{array}$$

Fractional relaxation (*LP*) for Vertex Cover:

$$\begin{array}{llcll} \text{Min.} & \sum_{v \in V} x_v & & & \\ \text{s.t.:} & x_v + x_u & \geq & 1 & \forall \{u,v\} \in E \\ & x_v & \geq & 0 & \forall v \in V \end{array}$$

Theorem: From Fractional to Integral Solution

Let $(x_v)_{v \in V}$ be a fractional optimal solution.
$V_0 = \{v \in V \mid x_v < 1/2\}$, $V_1 = \{v \in V \mid x_v > 1/2\}$ and $V_{1/2} = \{v \in V \mid x_v = 1/2\}$
There exists a Minimum (Integral) vertex cover S such that $V_1 \subseteq S \subseteq V_1 \cup V_{1/2}$

Corollary: reduction Rule using LP for Vertex Cover

Let $(x_v)_{v \in V}$ be a fractional optimal solution.
Then $vc(G) \leq k$ if and only if $vc(G \setminus V_1) \leq k - |V_1|$.

Back to Fractional Relaxation for Vertex Cover

Let $G = (V, E)$ be a graph

Integer Linear programme (ILP) for Vertex Cover:

$$\begin{array}{llcll} \text{Min.} & \sum_{v\in V} x_v & & & \\ \text{s.t.:} & x_v + x_u & \geq & 1 & \forall \{u,v\} \in E \\ & x_v & \in & \{0,1\} & \forall v \in V \end{array}$$

Fractional relaxation (LP) for Vertex Cover:

$$\begin{array}{llcll} \text{Min.} & \sum_{v\in V} x_v & & & \\ \text{s.t.:} & x_v + x_u & \geq & 1 & \forall \{u,v\} \in E \\ & x_v & \geq & 0 & \forall v \in V \end{array}$$

Theorem: From Fractional to Integral Solution

Let $(x_v)_{v\in V}$ be a fractional optimal solution.
$V_0 = \{v \in V \mid x_v < 1/2\}$, $V_1 = \{v \in V \mid x_v > 1/2\}$ and $V_{1/2} = \{v \in V \mid x_v = 1/2\}$
There exists a Minimum (Integral) vertex cover S such that $V_1 \subseteq S \subseteq V_1 \cup V_{1/2}$

proof: S^* be an optimal (integral) solution of Vertex Cover. Let $S = (S^* \setminus V_0) \cup V_1$.
Clearly S is a vertex cover. By contradiction, if S is not optimal, $|S^* \cap V_0| < |V_1 \setminus S^*|$.
Let $v \in V_1 \cup V_0$ be a vertex with x_v as close as possible from $1/2$ (exists by assumption). Let $\varepsilon = |x_v - 1/2|$. Remove ε to x_w for any $w \in V_1 \setminus S^*$ and add ε to x_w for any $w \in V_0 \cap S^*$. We get a smaller feasible fractional solution, a contradiction.

Back to Fractional Relaxation for Vertex Cover

Let $G = (V, E)$ be a graph

Integer Linear programme (*ILP*) for Vertex Cover:

$$\begin{array}{lrcrr} \text{Min.} & \sum_{v \in V} x_v & & & \\ \text{s.t.:} & x_v + x_u & \geq & 1 & \forall \{u,v\} \in E \\ & x_v & \in & \{0,1\} & \forall v \in V \end{array}$$

Fractional relaxation (*LP*) for Vertex Cover:

$$\begin{array}{lrcrr} \text{Min.} & \sum_{v \in V} x_v & & & \\ \text{s.t.:} & x_v + x_u & \geq & 1 & \forall \{u,v\} \in E \\ & x_v & \geq & 0 & \forall v \in V \end{array}$$

Theorem: From Fractional to Integral Solution

Let $(x_v)_{v \in V}$ be a fractional optimal solution.
$V_0 = \{v \in V \mid x_v < 1/2\}$, $V_1 = \{v \in V \mid x_v > 1/2\}$ and $V_{1/2} = \{v \in V \mid x_v = 1/2\}$
There exists a Minimum (Integral) vertex cover S such that $V_1 \subseteq S \subseteq V_1 \cup V_{1/2}$

Corollary: reduction Rule using LP for Vertex Cover

Let $(x_v)_{v \in V}$ be a fractional optimal solution.
Then $vc(G) \leq k$ if and only if $vc(G \setminus V_1) \leq k - |V_1|$.

Linear Kernel for Vertex Cover

Alg4: Linear Kernel for $vc(G) \leq k$

input: graph $G = (V, E)$, integer $\ell \leq k$.

- **If** $|E| = 0$, *Return TRUE*
- Remove isolated vertices
- Let $(x_v)_{v \in V}$ be an optimal solution obtained by LP
- **If** optimal fractional solution $> \ell$, *Return FALSE*
- **Else** let $V_1 = \{v \in V \mid x_v > 1/2\}$.
  - **If** $V_1 \neq \emptyset$ then *Return* $Alg4(G \setminus V_1, \ell - |V_1|)$.
  - **Else** Apply $Alg2(G, \ell)$

While possible, apply LP and add to the solution the vertices w with $x_w > 1/2$.
When it is not possible anymore, then all vertices v are such that $x_v = 1/2$ (check it).
Hence $|V| \leq 2k$ (Linear kernel).
Then, apply brute force algorithm (e.g., $Alg2$) to the remaining "small" graph

Linear Kernel for Vertex Cover

Alg4: Linear Kernel for $vc(G) \leq k$

input: graph $G = (V, E)$, integer $\ell \leq k$.

- **If** $|E| = 0$, *Return TRUE*
- Remove isolated vertices
- Let $(x_v)_{v \in V}$ be an optimal solution obtained by LP
- **If** optimal fractional solution $> \ell$, *Return FALSE*
- **Else** let $V_1 = \{v \in V \mid x_v > 1/2\}$.
  - **If** $V_1 \neq \emptyset$ then *Return* $Alg4(G \setminus V_1, \ell - |V_1|)$.
  - **Else** Apply $Alg2(G, \ell)$

While possible, apply LP and add to the solution the vertices w with $x_w > 1/2$.
When it is not possible anymore, then all vertices v are such that $x_v = 1/2$ (check it).
Hence $|V| \leq 2k$ (Linear kernel).
Then, apply brute force algorithm (e.g., $Alg2$) to the remaining “small" graph

Outline

Take Aways

- Parameterized Problem: input (size n) $+$ parameter k
- FPT algorithm: in time $f(k)n^{O(1)}$
- Kernelization: Data reduction
- Kernelization $\Leftrightarrow$ FPT
- Linear Kernel for Vertex Cover