Fixed-Parameter Algorithms, IA166

Sebastian Ordyniak

Faculty of Informatics
Masaryk University Brno

Spring Semester 2013

Outline

Introduction/Motivation

- Kernelization is a technique to obtain FPT algorithms.
- Kernelization algorithms are preprocessing algorithms that can be used to enhance any algorithmics method.
- Kernelization also gives a theoretical framework for mathematically evaluating preprocessing algorithms.
- Kernelization algorithms are related to approximation algorithms.

Outline

A Simple Kernel for VERTEX COVER

k-VERTEX COVER (k-VC) **Parameter: k**

Input: A graph G and a natural number k.
Question: Does G have a vertex cover S with $|S| \leq k$?

Some Observations

Observation (1)

Let (G, k) be a k-VC instance and let v be an isolated vertex of G. Then (G, k) and $(G \setminus \{v\}, k)$ are equivalent instances of k-VC.

Observation (2)

Let (G, k) be a k-VC instance and let v be a vertex of G with degree greater than k. Then (G, k) and $(G \setminus \{v\}, k - 1)$ are equivalent instances of k-VC.

Observation (3)

Let G be a graph with maximum degree k that admits a vertex cover with at most k vertices. Then $|E(G)| \leq k^2$.

The Kernel

Theorem

Let (G, k) be a k-VC instance. In polynomial time we can obtain an equivalent k-VC instance (G', k') with $|E(G')| \leq O(k^2)$.

Proof:

Iteratively remove isolated vertices and vertices with degree greater than k. By Obervations (1) and (2) the resulting instance (G', k') is equivalent to the original instance and $k' \leq k$.
If $|E(G')| > k'^2$ then by Observation (3) (G', k') is a NO-instance and we may return any trivial and small NO-instance of k-VC. Otherwise we return (G', k'). $\square$

Remarks

Theorem

Let (G, k) be a k-VC instance. In polynomial time we can obtain an equivalent k-VC instance (G', k') with $|E(G')| \leq O(k^2)$.

Remark:

The above theorem is easily extended to an FPT-algorithm:

- Compute the reduced instance (G', k') from the above theorem. This takes only a polynomial amount of time.
- Solve k-VC by brute-force on (G', k'). Because $|E(G')| \leq k'^2 \leq k^2$ this takes time at most 2^{k^2}.

Hence, the running time for the whole algorithm is $O^*(2^{k^2})$.

Remarks

- This preprocessing algorithm used a parameter dependent preprocessing rule: not so nice, i.e., not immediately applicable to non-parameterized optimization problems.
- Preprocessing algorithms of this type (kernelization algorithms) always give FPT-algorithms with nice additive complexities.

Outline

Definition

Definition

A kernelization algorithm A for a parameterized problem (Q, κ) is a **polynomial time** algorithm that for every instance X of (Q, κ) returns an equivalent instance X' with $|X'| \leq f(\kappa(X))$ for some arbitrary but computable function $f : \mathbb{N} \to \mathbb{N}$.
This is also called an $f(\kappa)$-kernel for (Q, κ).

Remark

Usually, $\kappa(X') \leq \kappa(X)$. This property is sometimes added to the definition.

Remarks

- The above algorithm for k-VC is a kernelization algorithm that returns an instance G' with $|E(G')| \in O(k^2)$ and $|V(G)| \in O(k^2)$.
- We will sometimes (sloppily) ignore logarithmic factors and call this an $O(k^2)$-kernel; note however that at least $|E(G')| \log |V(G')| = O(k^2 \log k)$ bits may be needed to encode G'.
- For graph problems, *vertex kernels* are important: e.g. suppose a graph G' is returned with $|E(G')| \leq k^2$ and $|V(G')| \leq ck$: this is an $O(k^2)$-kernel but a ck-vertex kernel. Edge kernels are defined similarly.

Equivalence between FPT-algorithms and Kernelization

Theorem

A parameterized problem (P, κ) admits an FPT algorithm iff there is a kernelization algorithm for (P, κ) (and (P, κ) is decidable).

Equivalence between FPT-algorithms and Kernelization

Proof ($\rightarrow$):

Suppose A is an FPT-algorithm for (P, κ) with running time $O(f(\kappa(X))(|X|)^c)$ for an instance X of (P, κ). If $|X| \leq f(\kappa(X))$ then X itself is a $f(\kappa)$-kernel. Hence, we can assume that $|X| > f(\kappa(X))$. Note that in this case the algorithm A runs in polynomial time because $O(f(\kappa(X)(|X|)^c)) \subseteq O((|X|)^{c+1})$. Hence, we can modify A into a kernelization algorithm as follows: If A returns YES then we return a trivial YES-instance for (P, κ) and if A returns NO we return a trivial NO-instance for (P, κ). Hence, we obtain an constant size kernel in this case and altogether a $f(\kappa)$-kernel for (P, κ).

Equivalence between FPT-algorithms and Kernelization

Proof ($\leftarrow$):

For the reverse direction suppose we are given a kernelization algorithm A for the decidable problem (P, κ). Hence, running A on an instance X of (P, κ) gives us a $f(\kappa)$-kernel X' for some arbitrary but computable function $f : \mathbb{N} \to \mathbb{N}$. Because (P, κ) is decidable we can then solve X' by brute-forth in time $O(g(f(\kappa(X')))) \subseteq O(g(f(\kappa(X))))$ for some arbitrary but computable function $g : \mathbb{N} \to \mathbb{N}$. Altogether we obtain the required FPT-algorithm for (P, κ) with running time $O^*(g(f(\kappa(X))))$. $\square$

Outline

Definition

MAXIMUM SATISFIABILITY (MAX SAT) **Parameter:** k

Input: A boolean CNF-formula $F := \bigvee_{i=1}^{m} C_i$ and a natural number k.
Question: Is there a truth assignment for F that satisfies at least k clauses?

Remark

The size of a CNF-formula is the sum of clause lengths, i.e., the number of literals. That means we ignore logarithmic factors again!

Trivial Clauses

Definition

A clause of F is trivial if it contains both a positive and a negative literal of the same variable.

Observation

Let F' be the CNF-formula obtained from F after removing all t trivial clauses. Then (F, k) and $(F', k - t)$ are equivalent.

Long Clauses

Definition

For an instance (F, k) a clause of F is long if it contains at least k literals and short otherwise.

Theorem

If F contains at least k long clauses then (F, k) is a YES-instance.

Proof:

Because every non-trival long clause contains at least k variables you can choose a unique variable for each of the k long clauses and satisfy the clauses by setting the choosen unique variable accordingly. □

Long Clauses

Theorem

Let (F, k) be an instance of Max Sat where F contains no trivial clauses and exactly $l \leq k$ long clauses and let F' be the CNF-formula obtained from F by deleting the l long clauses. Then (F, k) and $(F', k - l)$ are equivalent.

Proof:

A truth assignment for F which satisfies at least k clauses, satisfies at least $k - l$ clauses of F'. Furthermore, in a truth assignment for F' which satisfies $k - l$ clauses, all expect at most $k - l$ variables are free to be changed. This allows us to satisfy the remaining l long clauses. □

Theorem

Let (F,k) be an instance of MAX SAT where F does not contain trivial or long clauses. If F contains at least $2k$ clauses then (F,k) is a YES-instance.

Proof:

Take an arbitrary truth assignment τ and its complement $\bar{\tau}$. Because every clause is satisfied either by τ or by $\bar{\tau}$ one of them satisfies at least $\frac{2k}{2} = k$ clauses. $\square$

An $O(k^2)$-kernel for MAX SAT

The kernelization algorithm for MAX SAT on instance (F, k):

1. Let F contain exactly t trivial clauses. If $t \geq k$ return a trivial YES-instance. Otherwise, let F' be the formula obtained from F by removing the t trivial clauses and let $k' = k - t$.
2. Let F' contain exactly l long clauses. If $l \geq k'$ return a trivial YES-instance. Otherwise, let F'' be the formula obtained from F' after removing the l long clauses and let $k'' = k' - l$.
3. If F'' contains at least $2k''$ clauses return a trivial YES-instance. Otherwise, F'' contains at most $2k''$ clauses with at most k' literals each. Hence (F'', k'') is a $O(k''k') = O(k^2)$-kernel.

Outline

Definition

k-d-Hitting Set **Parameter:** k

Input: A hypergraph $H = (V, E)$ with $|e| \leq d$ for every $e \in E$.
Question: Does H have a hitting set S with $|S| \leq k$, i.e., a set S of vertices of H such that $S \cap e \neq \emptyset$ for every $e \in E$?

Remarks

- Because k-VC is equivalent to k-2-Hitting Set, the kernel for k-VC is a special case of the kernel for k-d-Hitting set.
- The more general problem Hitting Set is W[2]-complete!

Sunflowers

Definitions

Let $H = (V, E)$ be a hypergraph. A k-subflower in H consists of a set $S = \{e_1, \ldots, e_k\} \subseteq E$ and a core $C \subseteq V$ such that $e_i \cap e_j = C$ for every $1 \leq i < j \leq k$. A hypergraph is d-uniform if $|e| = d$ for every $e \in E$.

Sunflower Lemma

Let $H = (V, E)$ be a d-uniform hypergraph with more than $(k-1)^d d!$ edges. Then H has a k-sunflower which can be found in polynomial time.

Sunflower Lemma

Sunflower Lemma

Let $H = (V, E)$ be a d-uniform hypergraph with more than $(k-1)^d d!$ edges. Then H has a k-sunflower which can be found in polynomial time.

Proof:

By induction over d. If $d = 1$ then H has more than $k - 1$ disjoint edges which gives a k-sunflower. For $d > 1$ we use the following induction hypothesis:

IH: Every $(d-1)$-uniform hypergraph with more than $(k-1)^{d-1}(d-1)!$ edges contains a k-sunflower.

Sunflower Lemma

Proof, continued:

IH: Every $(d-1)$-uniform hypergraph with more than $(k-1)^{d-1}(d-1)!$ edges contains a k-sunflower.

Let $F = \{f_1, \ldots, f_l\}$ be a maximal set of disjoint hyperedges in H. If $l \geq k$ then F is a sunflower with core $\emptyset$.
Otherwise, let $W = \bigcup_{i=1}^{l} f_i$ then $|W| \leq (k-1)d$. H contains more than $(k-1)^d d!$ edges and every edges of H is hit by W.

Sunflower Lemma

Proof, continued:

IH: Every $(d-1)$-uniform hypergraph with more than $(k-1)^{d-1}(d-1)!$ edges contains a k-sunflower.

Hence, there is an element $w \in W$ that hits more than $\frac{(k-1)^d d!}{(k-1)d} = (k-1)^{d-1}(d-1)!$ edges. Taking all of these edges and removing w from them yields a $(d-1)$-uniform hypergraph H' with more than $(k-1)^{d-1}(d-1)!$ edges. By induction, H' contains a k sunflower S. Let C be its core. Taking the corresponding edges in H yields a k-sunflower in H with core $C \cup \{w\}$. $\square$

Sunflower Lemma

Remark

The proof of the Sunflower Lemma can be easily modified to a polynomial time algorithm to find a k-sunflower in a hypergraph H.

A kernel for k-d-HITTING SET

Let F be a $(k+1)$-sunflower with core C in hypergraph H and let S be a hitting set of H.

- If $S \cap C = \emptyset$ then C has to hit all pedals of F so $|S| \geq k+1$.
- Therefore, H has a hitting set of size k iff the hypergraph H' with edge set $(E(H) \setminus F) \cup \{C\}$ has a hitting set of size k.
- Reduction rule: replace (H, k) by (H', k).

By the subflower lemma, a reduced hypergraph H contains:

- at most $(k-1)$ edges of size 1;
- at most $(k-1)^2 2!$ edges of size 2;
- ...
- at most $(k-1)^d d!$ edges of size d.

Hence, it contains at most $(k-1)^d d! d$ edges in total.

A kernel for k-d-HITTING SET

Theorem

The above algorithm is a $(k-1)^d d! d$-edge kernelization for k-d-HITTING SET.

Outline

A Kernel for Maximum Leaves Spanning Tree

Let G be a graph and $v \in V(G)$:

Definitions

- A subgraph H of G is spanning if $V(H) = V(G)$.
- G is a tree if it is connected and has no cycles.
- A leaf of a (tree) is a vertex v with degree 1.
- We denote by $\deg(v)$ the degree of the vertex v in G.

k-Max Leaves Spanning Tree (k-LST) **Parameter:** k

Input: A connected graph G and a natural number k.
Question: Does G have a spanning tree with at least k leaves?

Some further notions

Let G be a graph and $\{u, v\} \in E(G)$.

Definition (Contraction)

$G/\{u, v\}$ is the graph obtained from G after contracting the edge $\{u, v\}$ into a new vertex, i.e., $G/\{u, v\}$ has vertex set $(V(G) \setminus \{u, v\}) \cup \{n\}$ and edge set

$$\begin{aligned}
&\{\, \{x, y\} \in [V(G) \setminus \{u, v\}]^2 : \{x, y\} \in E(G)\,\} \cup \\
&\{\, \{x, n\} : x \in V(G) \setminus \{u, v\} \text{ and} \\
&\qquad\qquad (\{x, u\} \in E(G) \text{ or } \{x, v\} \in E(G))\,\}.
\end{aligned}$$

Some further notions

Let G be a graph and $\{u, v\} \in E(G)$.

Definitions

- $G - \{u, v\}$ is the graph $(V(G), E(G) \setminus \{u, v\})$.
- If G is connected then the edge $\{u, v\}$ is a bridge if the graph $G - \{u, v\}$ is disconnected.

Reduction Rules

Degree 2 Rule

Let (G, k) be k-LST instance and let $\{u, v\} \in E(G)$ with $\deg(u) = \deg(v) = 2$. If $G - \{u, v\}$ is connected, then $(G - \{u, v\}, k)$ is an equivalent instance.

Bridge Rule

Let (G, k) be k-LST instance and let $\{u, v\} \in E(G)$ with $\deg(u) \geq \deg(v) \geq 2$. If $\{u, v\}$ is a bridge, then $(G/\{u, v\}, k)$ is an equivalent instance.

Consequently, a reduced instance (G, k) contains no adjacent vertices of degree 2 and no bridges between vertices of degree at least 2.

Reduction Rules

Theorem

A reduced connected graph G contains a spanning tree with at least $\frac{|V(G)|}{5}$ leaves.

Proof:

Let T be a (possible non-spanning) subgraph of G that is a tree. We define: $n(T)$=|V(T)|, $l(T)$ is the number of leaves of F, and $d(T)$ is the number of dead leaves of T, i.e., the leaves of T that have no neighbor outside of T. We first show that G contains a subtree T with $4l(T) + d(T) \geq n(T)$. W.l.o.g. G contains a vertex v with degree at least 3. Then v together with its neighbors is such a tree.

Reduction Rules

Theorem

A reduced connected graph G contains a spanning tree with at least $\frac{|V(G)|}{5}$ leaves.

Proof, continued:

Given a tree T with $4l(T) + d(T) \geq n(T)$, a larger tree T' with $4l(T') + d(T') \geq n(T')$ exists if:

(A) T contains a vertex with at least 2 neighbors not in T, or a non-leaf with at least 1 neighbor not in T;

(B) If (A) does not apply but there is a vertex outside of T with either at least 2 neighbors in T, or with degree 1.

(C) If there is a vertex outside of T with exactly one neighbor in T and degree at least 3.

Reduction Rules

Theorem

A reduced connected graph G contains a spanning tree with at least $\frac{|V(G)|}{5}$ leaves.

Proof, continued:

(D) If (B) and (C) do not apply but T is not yet spanning. Then there is u inside of T with at least 1 neighbor inside and 1 neighbor v outside of T and with degree exactly 2. Furthermore, the degree of v cannot be 1 (otherwise u and its other neighbor would form a bridge) and also not 2 (otherwise u and v would be degree 2 neighbors). Hence, v has degree at least 3 and no neighbors in T ((C)). Consequently, we can add v and its neighbors to T.

Reduction Rules

Theorem

A reduced connected graph G contains a spanning tree with at least $\frac{|V(G)|}{5}$ leaves.

Proof, continued:

Hence, a spanning tree with $4l(T) + d(T) \geq n(T)$ exists.
Because $d(T) = l(T)$ in any spanning tree we get $l(T) \geq \frac{n}{5}$.

A 5k-vertex kernel for k-Leaf Spanning Tree

The following algorithm gives a 5k-vertex kernel for a k-LST instance (G, k):

Apply the degree 2 rule and the bridge rule until an equivalent irreducible instance (G', k') is obtained.

If G' has more than 5k vertices we can return a trivial YES-instance and otherwise G' is the kernel.

Outline

MINIMUM VERTEX COVER as an Integer Programm

Let G be an undirected graph with vertices $V(G) = \{v_1, \dots, v_n\}$ and k a natural number. Then the k-VERTEX COVER problem for (G, k) can be written as follows:

VC-ILP: $\min \sum_{i=1}^{n} x_i$

$$\begin{aligned} s.t.\ x_i + x_j \geq 1 \quad & \forall \{v_i, v_j\} \in E(G) \\ x_i \in \{0, 1\} \quad & \forall i \in \{1, \dots, n\} \end{aligned}$$

Here a $0/1$-variable x_i determines whether the vertex v_i is taken into the vertex cover.

VC-IPL Relaxation

Let G be an undirected graph with vertices $V(G) = \{v_1, \ldots, v_n\}$ and k a natural number. Then the Half-Integer Relaxation of the k-VERTEX COVER problem for (G, k) can be written as follows:

VC-REL: $\quad \min \sum_{i=1}^{n} x_i$

$$\text{s.t. } x_i + x_j \geq 1 \quad \forall \{v_i, v_j\} \in E(G)$$
$$x_i \in \{0, \tfrac{1}{2}, 1\} \; \forall i \in \{1, \ldots, n\}$$

Here a $0/1$-variable x_i determines whether the vertex v_i is taken into the vertex cover.

VC-REL

How does VC-REL help us to construct a kernel for VC? We need to answer the following questions:

Question (1)

How can we find an optimal solution to VC-REL?

Question (2)

How can an optimal solution for VC-REL be used to construct a $2k$-vertex kernel for k-VC?

We start by answering Question (2).

Some Properties of VC-REL

Let $\eta : \{x_1, \ldots, x_n\} \to \{0, \frac{1}{2}, 1\}$ be an optimal solution to VC-REL on the graph G and define $V_j = \{\, v_i : \eta(x_i) = j \,\}$ for every $j \in \{0, \frac{1}{2}, 1\}$.

Property (1)

If C is a vertex cover for $G[V_{\frac{1}{2}}]$, then $C \cup V_1$ is a VC for G.

Property (2)

$G[V_{\frac{1}{2}}]$ has no VC of size less than $\frac{|V_{\frac{1}{2}}|}{2}$.

Property (3)

There is a minimum VC C of G with $V_1 \subseteq C$.

Some Properties of VC-REL

Let $\eta : \{x_1, \ldots, x_n\} \to \{0, \frac{1}{2}, 1\}$ be an optimal solution to VC-REL on the graph G and define $V_j = \{\, v_i : \eta(x_i) = j \,\}$ for every $j \in \{0, \frac{1}{2}, 1\}$.

Property (1)

If C is a vertex cover for $G[V_{\frac{1}{2}}]$, then $C \cup V_1$ is a VC for G.

Proof:

Clearly all edges with at least 1 endpoint in V_1 and all edges with both endpoints in $V_{\frac{1}{2}}$ are covered by $C \cup V_1$. Hence, the only edges remaining are the edges with both endpoints in V_0 or 1 endpoint in V_0 and the other in $V_{\frac{1}{2}}$. However, because η is a solution of VC-REL such edges can not exist. $\square$

Some Properties of VC-REL

Let $\eta : \{x_1, \ldots, x_n\} \to \{0, \frac{1}{2}, 1\}$ be an optimal solution to VC-REL on the graph G and define $V_j = \{ v_i : \eta(x_i) = j \}$ for every $j \in \{0, \frac{1}{2}, 1\}$.

Property (2)

$G[V_{\frac{1}{2}}]$ has no VC of size less than $\frac{|V_{\frac{1}{2}}|}{2}$.

Proof:

Suppose not and let C be a VC of $G[V_{\frac{1}{2}}]$ of size less than $\frac{|V_{\frac{1}{2}}|}{2}$. Because of Property (1) $C \cup V_1$ is a vertex cover of G. Hence, η' with $\eta'(x_i) = 1$ if $v_i \in C \cup V_1$ and $\eta'(x_i) = 0$ otherwise is a solution to VC-REL and $\sum_{i=0}^{n} \eta'(x_i) < |V_1| + \frac{1}{2}|V_{\frac{1}{2}}| = \sum_{i=0}^{n} \eta(x_i)$ contradicting the minimality of η.

Some Properties of VC-REL

Let η be an optimal solution to VC-REL on the graph G and define $V_j = \{ v_i : \eta(x_i) = j \}$ for every $j \in \{0, \frac{1}{2}, 1\}$.

Property (3)

There is a minimum VC C of G with $V_1 \subseteq C$.

Proof:

Let C be a minimum VC of G. We first show that $|C \cap V_0| \geq |V_1 \setminus C|$. Let $\eta' : \{x_1, \ldots, x_n\} \to \{0, \frac{1}{2}, 1\}$ such that $\eta'(x_i) = \frac{1}{2}$ if $v_i \in (C \cap V_0) \cup (V_1 \setminus C)$ and $\eta'(x_i) = \eta'(x_i)$, otherwise. We claim that η' is a solution to VC-REL.

Some Properties of VC-REL

Let η be an optimal solution to VC-REL on the graph G and define $V_j = \{\, v_i : \eta(x_i) = j \,\}$ for every $j \in \{0, \frac{1}{2}, 1\}$.

Property (3)

There is a minimum VC C of G with $V_1 \subseteq C$.

Proof, continued:

Consider an edge $\{v_i, v_j\}$. If $\{v_i, v_j\} \subseteq V_{\frac{1}{2}} \cup V_1$ then $\eta'(x_i) + \eta'(x_j) \geq \frac{1}{2} + \frac{1}{2} = 1$. Hence, w.l.o.g. we can assume that $v_i \in V_0$. Then $v_j \in V_1$ (otherwise η would not be feasable). If $v_j \in C$ then $\eta'(x_j) = 1$. Otherwise, because C is a vertex cover $v_i \in C$ and hence $\eta'(v_i) + \eta'(v_j) = \frac{1}{2} + \frac{1}{2} = 1$.

Some Properties of VC-REL

Let η be an optimal solution to VC-REL on the graph G and define $V_j = \{\, v_i : \eta(x_i) = j \,\}$ for every $j \in \{0, \frac{1}{2}, 1\}$.

Property (3)

There is a minimum VC C of G with $V_1 \subseteq C$.

Proof, continued:

Because η is an optimal solution to VC-REL, we obtain:

$$0 \leq \sum_i \eta'(x_i) - \sum_i \eta(x_i) = \frac{1}{2}|C \cap V_0| - \frac{1}{2}|V_1 \setminus C|$$

Hence, $|V_1 \setminus C| \leq |C \cap V_0|$, as required.
Consider the set $C' := (C \setminus V_0) \cup V_1$. It follows that $|C'| \leq |C|$. It is now easy to see that C' is a VC of G which concludes the proof. $\square$

A $2k$-Vertex Kernel for Vertex Cover

Hence, we have:

(1) If C is a vertex cover for $G[V_{\frac{1}{2}}]$, then $C \cup V_1$ is a VC for G.

(2) $G[V_{\frac{1}{2}}]$ has no VC of size less than $\frac{|V_{\frac{1}{2}}|}{2}$.

(3) There is a minimum VC C of G with $V_1 \subseteq C$.

This allows for the following $2k$-Vertex Kernelization Algorithm:

Let (G, k) be a k-VC instance and let η be an optimal solution to the corresponding VC-REL problem. Consider (G', k') with $G' = G[V_{\frac{1}{2}}]$ and $k' = k - |V_1|$. Because of Property (1) and Property (3) the two instances are equivalent. Furthermore, because of Property (2) (G', k') contains at most $2k$ vertices, otherwise we can return a trivial NO-instance.

Solving VC-REL

Question (1)

How can we find an optimal solution to VC-REL?

There are (at least) 2 answers to this question.

Answer (1)

Relax VC-REL further to a linear program that can be solved in polynomial time. One can now show that such a real-valued solution can be efficiently transformed to a VC-REL solution of the same value.

Answer (2)

Using matchings in bipartite graphs.

Solving VC-REL using Matchings

Definition

A graph G is bipartite if there is a partition $\{A, B\}$ of $V(G)$ such that all edges of G have 1 endpoint in A and 1 endpoint in B. A and B are the sides or parts of G.

Let G be a graph. We denote by B(G) the bipartite graph obtained from G that has vertex set $\{\, v, v' : v \in V(G) \,\}$ and edge set $\{\, (u, v'), (u', v) : \{u, v\} \in E(G) \,\}$.

Lemma

VC-REL on G has a solution η with $\sum_i \eta(x_i) = z$ iff B has a vertex cover C with $|C| = 2z$.

Solving VC-REL using Matchings

Lemma

VC-REL on G has a solution η with $\sum_i \eta(x_i) = z$ iff $B(G)$ has a vertex cover C with $|C| = 2z$.

Proof:

Let η be a solution for VC-REL on G. We set $C := \{\, v_i, v_i' : \eta(x_i) = 1 \,\} \cup \{\, v_i : \eta(x_i) = \frac{1}{2} \,\}$. To show that C is a VC of G consider an edge $\{v_i, v_j'\} \in E(B(G))$. Clearly, C covers this edge as long as $\eta(x_i) \neq 0$. Furthermore, if $\eta(x_i) = 0$ then $\eta(x_j) = 1$ (because η is a solution and $\{v_i, v_j\} \in E(G)$). Hence, $v_j' \in C$.

Solving VC-REL using Matchings

Lemma

VC-REL on G has a solution η with $\sum_i \eta(x_i) = z$ iff B has a vertex cover C with $|C| = 2z$.

Proof:

For the reverse direction let C be a vertex cover of $B(G)$. We define η such that $\eta(x_i) = 1$ if $v_i, v_i' \in C$, $\eta(x_i) = \frac{1}{2}$ if either $v_i \in C$ or $v_i' \in C$ and $\eta(x_i) = 0$, otherwise.

Solving VC-REL using Matchings

Lemma

VC-REL on G has a solution η with $\sum_i \eta(x_i) = z$ iff B has a vertex cover C with $|C| = 2z$.

Proof, continued:

We claim that η is a solution to VC-REL of G. Consider an edge $\{v_i, v_j\} \in E(G)$. Because C covers both $\{v_i, v_j'\}$ and $\{v_i', v_j\}$ one of the following holds:

- $v_i \in C$ and $v_i' \in C$. Then $\eta(x_i) = 1$.
- $v_j \in C$ and $v_j' \in C$. Then $\eta(x_j) = 1$.
- $v_i \in C$ and $v_j \in C$. Then $\eta(x_i) \geq \frac{1}{2}$ and $\eta(x_j) \geq \frac{1}{2}$.
- $v_i' \in C$ and $v_j' \in C$. Then $\eta(x_i) \geq \frac{1}{2}$ and $\eta(x_j) \geq \frac{1}{2}$.

König's Theorem

Theorem

Let G be a bipartite graph. Then the size of a minimum vertex cover equals the size of a maximum matching, and both can be found in polynomial time.

The previous Lemma now allows us to compute an optimal solution to VC-REL for a graph G by computing a maximum matching (minimum vertex cover) in the bipartite graph $B(G)$.

König's Theorem

Definition

A matching in a graph G is a set of edges $M \subseteq E(G)$ that share no end vertices (every $v \in V(G)$ is incident with at most 1 edge of M). A vertex $v \in V(G)$ is saturated by M if it is incident with an edge of M.

Definition

Let B be a graph with a matching M. A path P in B is alternating if its edges are alternatingly in M and not in M. An alternating path is augmenting if its end vertices are not saturated by M.

Berge's Theorem

Let G be a graph with matching M. Then M is maximum iff G contains no augmenting path.

Proof of König's Theorem

Theorem

The size of a MVC equals the size of a MM on a bipartite graph.

Proof:

Because every edge of a matching needs to be covered by any vertex cover it trivially holds that $|M| \leq |C|$ for any matching M and any VC C.

Hence, it remains to show that $|C| \leq |M|$.

Proof of König's Theorem

Proof, continued:

The following algorithm finds a MVC C and a MM M with $|C| = |M|$ on a bipartite graph B with parts V and V':

(1) Start with $C = V$ and $M = \emptyset$.

(2) If $|C| = |M|$ then return C and M, halt.

(3) Choose an unsaturated vertex $v \in C$ and construct an alternating search tree subgraph T of B, rooted at v.

(4) If T contains an augmenting path P, then augment M using P, goto (2).

(5) Otherwise, find a vertex set S with $v \in S$ such that: $N(S)$ is saturated and $|N(S)| < |S|$. Then $C' := (C \setminus S) \cup N(S)$ is a VC with $|C'| < |C|$. Set $C := C'$, goto (2).

Hall's Theorem

The following theorem also follows:

Hall's Theorem

A bipartite graph B with sides V and V' has a matching saturating V iff there is no $S \subseteq V$ with $|N(S)| < |S|$.

Summary

- In polynomial time we can find a matching M and a VC C with $|M| = |C|$, which therefore are maximum resp. minimum.
- By applying this procedure to the bipartite graph B constructed from G, we can solve VC-REL on G in polynomial time.
- This concludes the $2k$-vertex kernelization for k-VC.

A Kernel for VC using Crowns

Definition

A crown in a graph G is a triple (I, H, M) such that:

C1 $I \subseteq V(G)$ is an independent set;

C2 $N(I) \subseteq H$;

C3 M is a matching (between I and H) that saturates H.

Theorem

Let G be a graph and η an optimal solution to VC-REL of G where $V_1 \neq \emptyset$. Then there is a matching M between V_0 and V_1 such that (V_0, V_1, M) is a crown of G.

A Kernel for VC using Crowns

Theorem

Let G be a graph and η an optimal solution to VC-REL of G where $V_1 \neq \emptyset$. Then there is a matching M between V_0 and V_1 such that (V_0, V_1, M) is a crown of G.

Proof:

Clearly, Properties C1 and C2 are trivially satisfied by V_0 and V_1. Furthermore, as we have shown before (Property (3)) that V_1 is a minimum vertex cover for $G[V_1 \cup V_0] - E[V_1]$ which by Koenig's Theorem implies Property C3.

A Kernel for VC using Crowns

Theorem

Let G be a graph containing a crown (I, H, M) with $|H| < |I|$. Then an optimal solution for VC-REL to G gives us a crown for G.

Proof:

The crown (I, H, M) with $|H| < |I|$ ensures that VC-REL has a solution η with value at most $|H| + \frac{1}{2}|V(G) \setminus (I \cup H)| < \frac{1}{2}|V(G)|/2$. Hence (unless G contains isolated vertices), we obtain that $V_1 \neq \emptyset$ for an optimal solution η which gives us a crown for G.

A Kernel for VC using Crowns

- We have seen before that if (I, H, M) is a crown of G, then (G, k) and $(G \setminus (I \cup H), k - |I|)$ are equivalent k-VC instances.
- Furthermore, if G contains no crown (I, H, M) with $|H| < |I|$, then every VC S of G has size at least $\frac{|V|}{2}$.

Conclusion

A different way to express the $2k$-vertex kernel for k-VC: find crowns (I, H, M) with $|H| < |I|$ in polynomial time if they exist and reduce them. A crownless graph is a $2k$-kernel.

Remark

Crown reductions have also been used to find kernelizations for other problems.

An Alternative $3k$-Vertex Kernel for VC

Lemma

Let G be a graph without isolated vertices and k be a natural number. Then in polynomial time we can either:

- find a matching of size $k + 1$;
- find a crown decomposition;
- or conclude that the graph has at most $3k$ vertices.

An Alternative $3k$-Vertex Kernel for VC

Lemma

Let G be a graph without isolated vertices and k be a natural number. Then in polynomial time we can either:

- find a matching of size $k+1$; $\rightarrow$ No solution!
- find a crown decomposition; $\rightarrow$ Reduce!
- or conclude that the graph has at most $3k$ vertices. $\rightarrow$ $3k$-vertex kernel!

An Alternative $3k$-Vertex Kernel for VC

Proof:

Greedily find a maximal matching M of G. If $|M| > k$ then we are done. Consider the bipartite graph B with partition $\{I := G \setminus V[M], H := V[M]\}$ obtained from G after deleting all edges between vertices in $V[M]$. Because M is maximal in G it follows that I is an independent set in G (and also in B). Find a minimum vertex cover C of B (using Koenig's Theorem this can be done in polynomial time). If C contains a vertex from H then we obtain a crown decomposition. Otherwise C contains all vertices of I hence G contains at most $2k + k$ vertices.

Dual of Vertex Coloring

SAVING k-COLORS **Parameter: k**

Input: A graph G and a natural number k.
Question: Does G have a vertex coloring with $|V(G)| - k$ colors?

Dual of Vertex Coloring

Lemma

Let $\mathcal{I} := (G, k)$ be an instance of SAVING k-COLORS and (I, H, M) be a crown decomposition of the complement of G. Then $\mathcal{I}$ and $\mathcal{I}' := (G \setminus (I \cup H), k - |H|)$ are equivalent instances of SAVING k-COLORS.

Proof:

Let $\mathcal{I}$ and (I, H, M) be as above. Then I is a clique in G and hence every vertex in I has to be colored with a different color. Furthermore, because of the matching M the vertices in H can be colored using only these $|I|$ colors and none of these colors can be used for $G \setminus (I \cup H)$.

Dual of Vertex Coloring

Lemma

Let (G, k) be an instance of SAVING k-COLORS and let $\bar{G}$ be the complement of G. Then in polynomial time we can either:

- find a matching of size $k + 1$ in $\bar{G}$;
- find a crown decomposition of $\bar{G}$;
- or conclude that the graph $\bar{G}$ has at most $3k$ vertices.

This gives a $3k$-vertex kernel for SAVING k-COLORS.

Dual of Vertex Coloring

Lemma

Let (G, k) be an instance of SAVING k-COLORS and let $\bar{G}$ be the complement of G. Then in polynomial time we can either:

- find a matching of size $k+1$ in $\bar{G}$; $\rightarrow$ Yes, we can save k colors!
- find a crown decomposition of $\bar{G}$; $\rightarrow$ Reduce!
- or conclude that the graph $\bar{G}$ has at most $3k$ vertices. $\rightarrow$ $3k$-vertex kernel!

This gives a $3k$-vertex kernel for SAVING k-COLORS.

Outline

Using Kernelization for Approximation

Definition

An α-approximation algorithm for a minimization (maximization) problem P is a polynomial time algorithm that returns a solution S for P such that value(S) $\leq \alpha$value(OPT) (value(S) $\geq \frac{1}{\alpha}$value(OPT)), where OPT is an optimal solution.

Observation

The $2k$-kernelization algorithm for a graph G gives a 2-approximation algorithm for VERTEX COVER as follows: Compute an optimal solution η to VC-REL of G. Then $V_{\frac{1}{2}} \cup V_1$ is a vertex cover of G of size at most 2 times the optimal solution.

Using Kernelization for Approximation

Remark

ck-vertex kernels for "vertex subset" problems usually yield c-approximation algorithms for the corresponding optimization problem.

Example

For MAXIMUM LEAVES SPANNING TREE, the $5k$-vertex kernelization gives a 5-approximation algorithm.

Outline

Combining Search Tree and Kernelization

Let P be a parameterized problem such that:

- P has a bounded search algorithm with branching number α and $P_S(|X|)$ is the time spend at each node of the search tree.
- P has a $p(k)$-kernelization algorithm with running time $P_K(|X|)$ where $p(k)$ is some polynomial in k.

for every instance (X, k) of P. Then the bounded search tree algorithm has a running time of $O(\alpha^k P_S(|X|))$. Furthermore, if we first apply kernelization and then run the bounded search tree algorithm on the kernel we obtain a running time of $O(P_K(|X|) + P_S(q(k))\alpha^k)$.

Combining Search Tree and Kernelization

Theorem

Let P be a parameterized problem such that:

- P has a bounded search algorithm with branching number α and $P_S(|X|)$ is the time spend at each node of the search tree.
- P has a $p(k)$-kernelization algorithm with running time $P_K(|X|)$ where $p(k)$ is some polynomial in k.

for every instance (X, k) of P. Then an algorithm that runs the kernelization algorithm at each node of the search tree has running time $O(P_K(|X|) + \alpha^k)$

Combining Search Tree and Kernelization

Proof, sketch:

For the combination of search tree and kernelization as outlined above, the recurrence function becomes:

$$T(k) = T(k-d_1) + \cdots + T(k-d_l) + P_K(p(k)) + P_S(p(k))$$

Because $p(k)$ is polynimimial in k we obtain:

$$T(k) = T(k-d_1) + \cdots + T(k-d_l) + P(k)$$

for some arbitrary polynomial $P(k)$. It can be shown that $T(k)$ is bounded by α^k.

Outline

Kernelization: Summary

- Kernelization algorithms are a method to obtain FPT algorithms.
- Every problem in FPT has a kernelization algorithm. One is hence mostly interested in finding small (polynomial) kernels.
- Kernelization algorithms are preprocessing algorithms that can add to any algorithmic method (e.g. approximation algorithms).
- Kernelization algorithms usually consist of reduction rules which reduce simple local structures and a bound $f(k)$ for irreducible instances that allows us to return No or Yes depending on the size of the instance.

Designing Kernelization Algorithms

- What are the trivial substructures, where an optimal solution of a certain form can be guaranteed?
- Is there a reduction rule reflecting this?
- Can a bound be proved for irreducible instances? If not, which structures are problematic? ...