Parameterized Algorithms

Lecture 5: Kernelization II

June 05, 2020

Max-Planck Institute for Informatics, Germany.

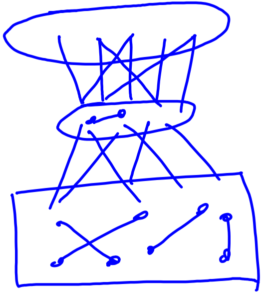
More Kernelization techniques

Crown Decomposition

Crown Decomposition

A partition of the vertex set of a graph into 3 parts (crown)C, (head)H and (the rest) R, such that:

- C is non-empty and an independent set, with edges to vertices of H alone.
- The bipartite graph between C and H in G contains a matching of size $|H|$.



Crown Decomposition

Lemma

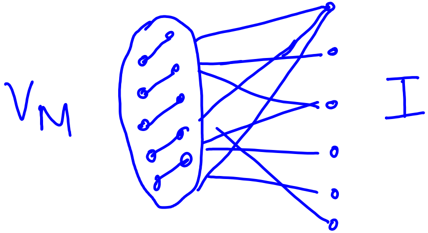
Let G be a graph on at least $3k+1$ vertices. Then in polynomial time, either we can find a matching of size $k+1$ or a Crown Decomposition of G.

Crown Decomposition

Lemma

Let G be a graph on at least $3k+1$ vertices. Then in polynomial time, either we can find a matching of size $k+1$ or a Crown Decomposition of G.

- Find a greedy matching M of G, if $|M| \geq k+1$ we are done
- Else V_M be the endpoints of M and $I = V(G) \setminus V_M$
- Consider the bipartite graph G' between V_M and I, compute a minimum vertex cover X of G'



Crown Decomposition

Lemma

Let G be a graph on at least $3k+1$ vertices. Then in polynomial time, either we can find a matching of size $k+1$ or a Crown Decomposition of G.

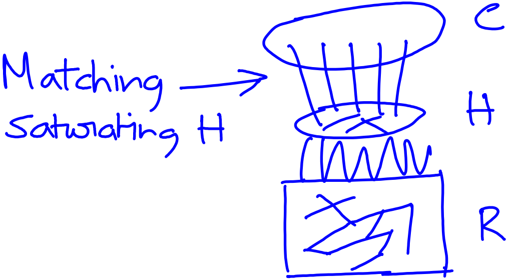
- Find a greedy matching M of G, if $|M| \geq k+1$ we are done
- Else V_M be the endpoints of M and $I = V(G) \setminus V_M$
- Consider the bipartite graph G' between V_M and I, compute a minimum vertex cover X of G'
- If $X \cap V_M = \emptyset$, then $|I| \leq k$, and hence $|V(G)| \leq 3k$
- Else, M' be a maximum matching in G', and M^* is subset of edges with exactly one endpoint in X.
- Crown Decomposition:

$$C = V(M^*) \cap I,\ H = V(M^*) \cap X, R$$

Crown Decomposition

VERTEX COVER kernel on $3k$ vertices.

- Remove all isolated vertices in G
- Find a Crown Decomposition (C, H, R) or a $k+1$ matching
- In the former case, the reduced instance is ~~$(G - C, k - |C|)$~~ $(G - C \cup H, k - |H|)$
- In the latter case, a trivial no instance



Linear Programming

Linear Programming

$$\min \sum_{v \in V(G)} x_v$$

$$x_u + x_v \geq 1 \;\; \forall (u,v) \in E(G)$$

$$x_v \geq 0 \;\; \forall v \in V(G)$$

Consider a (fractional) optimal solution x

$$V_0 = \{v \mid x_v < \frac{1}{2}\}, V_{\frac{1}{2}} = \{v \mid x_v = \frac{1}{2}\}, V_1 = \{v \mid x_v > \frac{1}{2}\}$$

Theorem (Nemhauser-Trotter)

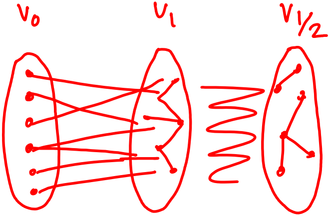
There is an optimum vertex cover S *such that* $V_1 \subseteq S \subseteq V_1 \cup V_{\frac{1}{2}}$

Linear Programming

- Observe: V_0 is an independent set, and has edges only to V_1.
- Given an optimum vertex cover S', let $S = (S' \setminus V_0) \cup V_1$.
- If $|S| > |S'|$ then $|S' \cap V_0| < |V_1 \setminus S'|$
- Let $\epsilon = \min_{v \in V_1 \cup V_0} |\frac{1}{2} - x_v|$
- Consider the LP solution where,

$$\begin{array}{c}\text{we decrease } x_v \text{ by } \epsilon \text{ for } v \in V_1 \setminus S' \\ \text{we increase } x_v \text{ by } \epsilon \text{ for } v \in V_0 \cap S'\end{array}$$

- It is feasible, and smaller than LP-Opt! Hence, $|S| = |S'|$



Linear Programming

Consider a (fractional) optimal solution x

$$V_0 = \{v \mid x_v < \frac{1}{2}\}, V_{\frac{1}{2}} = \{v \mid x_v = \frac{1}{2}\}, V_1 = \{v \mid x_v > \frac{1}{2}\}$$

Theorem (Nemhauser-Trotter)

There is an optimum vertex cover S such that $V_1 \subseteq S \subseteq V_1 \cup V_{\frac{1}{2}}$

- Reduction Rule: Delete $V_0 \cup V_1$, and reduce k by $|V_1$.
- When reduction doesn't apply, every vertex is in $V_{\frac{1}{2}}$, i.e. there are $2k$ Vertices.

Linear Programming

Consider a (fractional) optimal solution x

$$V_0 = \{v \mid x_v < \frac{1}{2}\}, V_{\frac{1}{2}} = \{v \mid x_v = \frac{1}{2}\}, V_1 = \{v \mid x_v > \frac{1}{2}\}$$

Theorem (Nemhauser-Trotter)

There is an optimum vertex cover S *such that* $V_1 \subseteq S \subseteq V_1 \cup V_{\frac{1}{2}}$

- Reduction Rule: Delete $V_0 \cup V_1$, and reduce k by $|V_1$.
- When reduction doesn't apply, every vertex is in $V_{\frac{1}{2}}$, i.e. there are $2k$ Vertices.

Vertex Cover has a kernel on $2k$ vertices

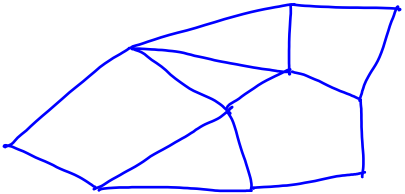
Planar Graphs

Connected Vertex Cover in Planar Graphs

- Planar Graphs: Graphs that can be drawn on a plane, without crossing edges.
- Euler's formula: $f = |E(G)| - |V(G)| + 2$

Lemma

Let G be a planar graph and A be a subset of vertices. Then $G - A$ has at most $2|A|$ connected components that see 3 vertices of A.



Connected Vertex Cover in Planar Graphs

Connected Vertex Cover: Find a vertex cover X of size k such that $G[X]$ is connected.

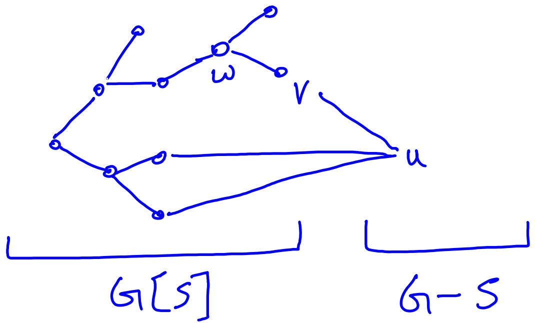
- Remove all isolated vertices, and G must be connected with at least 3 vertices
- Keep at most one degree-1 neighbors of a vertex
- If v is a degree-2 cut vertex, contract it; k drops by 1.

Lemma

If v is a degree-2 vertex, but not a cut vertex, then there is an optimum CVC S that excludes v

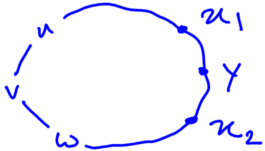
Connected Vertex Cover in Planar Graphs

- If $v \in S$, then one of it's two neighbors u, w in S
- Suppose $v, w \in S$, and $u \notin S$, then consider $S' = S - v + u$
- S' is a connected vertex cover
  Consider a spanning tree of $G[S]$, v is a leaf there, and all other neighbors of u are present in S.



Connected Vertex Cover in Planar Graphs

- If $v \in S$, then one of it's two neighbors u, w in S
- Suppose $v, w \in S$, and $u \notin S$, then consider $S' = S - v + u$
- S' is a connected vertex cover
  Consider a spanning tree of $G[S]$, v is a leaf there, and all other neighbors of u are present in S.
- Otherwise $u, v, w \in S$. Then $S - v$ is a vertex cover but perhaps not connected. Let X_1, X_2 be two components of $G[S] - v$.
- Consider a cycle C in G contain u, v, w.
  There are 3 consecutive vertices in $C - v$, say $x_1 y x_2$ such that $x_1 \in X_1, x_2 \in X_2$ and $y \notin S$
- $S - v + y$ is a connected vertex cover



Connected Vertex Cover in Planar Graphs

Connected Vertex Cover: Find a vertex cover X of size k such that $G[X]$ is connected.

- Remove all isolated vertices, and G must be connected with at least 3 vertices
- Keep at most one degree-1 neighbors of a vertex
- If v is a degree-2 cut vertex, contract it; k drops by 1.
- If v is a degree-2 non-cut vertex, split v into two vertices.

Connected Vertex Cover in Planar Graphs

Connected Vertex Cover: Find a vertex cover X of size k such that $G[X]$ is connected.

- Remove all isolated vertices, and G must be connected with at least 3 vertices
- Keep at most one degree-1 neighbors of a vertex
- If v is a degree-2 cut vertex, contract it; k drops by 1.
- If v is a degree-2 non-cut vertex, split v into two vertices.

Lemma

When no reduction rules apply, if G has a CVC of size k, then $|V(G)| \leq 4k$.

Recall, $G - S$ can have at most $2k$ vertices that see 3 or more vertices of S. Any other vertex is a degree-2 vertex, which can be reduced, or a degree-1 vertex, of which there are at most k. Hence, at most $4k$ vertices.

Kernelization Lower Bounds

Intuition

- k-PATH: Decide if G contains a path of length k.
- Suppose that k-PATH has a kernel of size k^3.

  It can be encoded in k^6 bits.
- Consider a collection of t instances of k-PATH, for $t = k^7$.

$$(G_1, k), (G_2, k) \dots (G_t, k)$$

- $G = G_1 \cup G_2 \dots G_t$ has a path of length k if and only if one of G_i does.

  i.e. (G, k) is an OR of $(G_1, k) \dots, (G_t, k)$
- Let (H, k') be the kernel for (G, k).

  (H, k) has "lost information" about some of the t instances!
- The Kernelization algorithm must have "solved" these forgotten instances.

  NP-hard problem in polytime!

Distillation

◦ Let $L, R \subseteq \Sigma^*$ be two languages. An OR-distillation of L into R is an **algorithm** that given a sequence of strings $x_1, x_2, \ldots, x_t \in \Sigma^*$ each of maximum length ℓ, runs in polynomial time in the total length of these strings and produces a string $y \in \Sigma^*$ such that $|y| = poly(\ell)$ and $y \in R$ if and only if some $x_i \in L$.

◦ A language L is in the complexity class coNP/poly if there is a Turing machine M and for each integer n, there is a string α_n of length $poly(n)$, called *advice* such that given any string $x \in \Sigma^n$, using α_n M can decide if $x \in L$ in *non-deterministic polynomial time.*

Theorem

Let $L, R \subseteq \Sigma^*$ be two languages. If there is an OR-distillation of L into R, then $L \in$ coNP/poly.

If L were NP-hard, then NP $\subseteq$ coNP/Poly

Kernelization + Composition $\Longrightarrow$ Distillation

∘ An equivalence relation R on the set Σ^* is called a polynomial equivalence relation if (i) there exists an algorithm that, given strings $x, y \in \Sigma^*$, resolves whether $x \equiv_R y$ in time polynomial in $|x| + |y|$; and (b) Relation R restricted to the set Σ^n has at most $poly(n)$ equivalence classes.

∘ Let $L \subseteq \Sigma^*$ be a language and $Q \subseteq \Sigma^* \times N$ be a parameterized language. We say that L cross-composes into Q if there exists a polynomial equivalence relation R and an algorithm A that takes as input a sequence of strings $x_1, x_2, \ldots, x_t \in \Sigma^*$ that are equivalent with respect to R, runs in time polynomial in total length of the strings, and outputs one instance $(y, k') \in \Sigma^* \times N$ such that:
(a) $k' \leq poly(k + t)$ where k is the max length a string x_i, and
(b) $(y, k') \in Q$ if and only if some $x_i \in L$

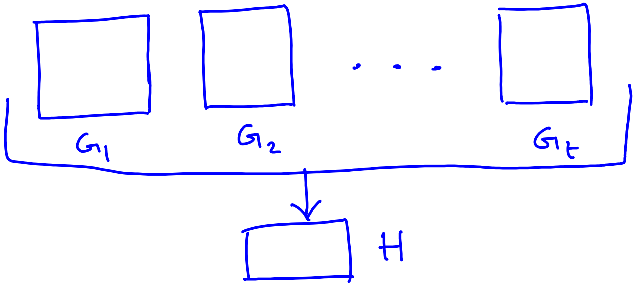
Theorem (Main Tool)

If an NP-hard language L cross-composes into a parameterized language Q, then Q does not admit a polynomial compression, unless $NP \subseteq coNP/\ poly$.

k-Path

Ham-Path cross-composes into k-Path

- Equiv Relation R: all malformed instances (in Σ^*) in one-class, and all well-formed instances in another.
- Given t instances of Ham-Path $G_1, G_2, \ldots, G_n$ on n-vertices, let (G, k) where $G = G_1 \cup \ldots G_t$ and $k = n$ be an instance of k-Path.
- Therefore k-Path has no polynomial kernel (or compression).



k-Path

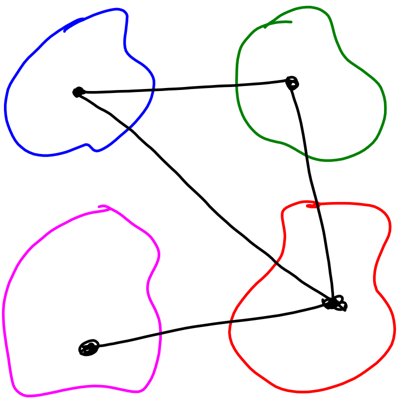
Ham-Path cross-composes into k-Path

- Equiv Relation R: all malformed instances (in Σ^*) in one-class, and all well-formed instances in another.
- Given t instances of Ham-Path $G_1, G_2, \ldots, G_n$ on n-vertices, let (G, k) where $G = G_1 \cup \ldots G_t$ and $k = n$ be an instance of k-Path.
- Therefore k-Path has no polynomial kernel (or compression).

Similarly we have AND-Distillation and AND-Composition

Graph Motif

Graph Motif: Given a graph G, integer k and a coloring c of $V(G)$ using k colors, find a connected subgraph H on k vertices with exactly one vertex of each color.



Graph Motif

Graph Motif: Given a graph G, integer k and a coloring c of $V(G)$ using k colors, find a connected subgraph H on k vertices with exactly one vertex of each color.

OR-Composition: t instances with same number of colors k

$$(G_1, k, c_1), (G_2, k, c_2), \ldots, (G_t, k, c_t)$$

Define (G, k, c) via disjoint union

(G, k, c) has a colorful motif H if and only if some (G_i, k, c_i) does

Lemma

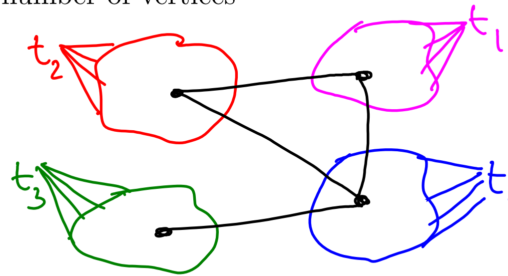
Graph Motif *has no polynomial kernel parameterized by the number of colors* k.

Steiner Tree par. by tree-size

Polynomial Parameter Transform: A polynomial time reduction that preserves the parameter value upto a polynomial factor, (i.e. k becomes $poly(k)$).

Graph Motif to Steiner Tree par. by tree-size

- Given (G, k, c), construct G' by add k new terminal vertices adjacent to each color class. Consider (G', T, ℓ) as the Steiner Tree instance where $\ell = 2k$.
- Note: Tree-size ℓ = number of vertices



Steiner Tree par. by tree-size

Polynomial Parameter Transform: A polynomial time reduction that preserves the parameter value upto a polynomial factor, (i.e. k becomes $poly(k)$).

Graph Motif to Steiner Tree par. by tree-size

- Given (G, k, c), construct G' by add k new terminal vertices adjacent to each color class. Consider (G', T, ℓ) as the Steiner Tree instance where $\ell = 2k$.
- Note: Tree-size ℓ = number of vertices

Theorem

Steiner Tree *parameterized by the tree-size, has no polynomial kernel.*

Thank You.