KAIST, School of Computing, Spring 2024
Algorithms for NP-hard problems (CS492)
Lecture: Eunjung KIM

Scribed By: Eunjung KIM
Lecture #12
4 March 2024

Contents

1 Algorithm for approximating treewidth

Reed [1] proposed an elegant algorithm for approximating computing treewidth of a graph, which is the subject of this section.

Theorem 1. *There is an algorithm which, given a graph G and a positive integer w, either outputs a tree decomposition of width at most $4w+2$ or correctly concludes that $\mathsf{tw}(G) > w$ in time $O(27^w \cdot wnm)$.*

Let G be a graph and $S \subseteq V(G)$ be a vertex subset. A vertex subset $X \subseteq V(G)$ is called a *α-balanced S-separator* if for every connected component C of $G - X$, we have $|C \cap S| \leq \alpha \cdot |S|$. We will take $\alpha := \frac{1}{2}$ and simply refer to a balanced S-separator without mentioning the coefficient α.

A *separation* of G is a pair (A, B) with $A \cup B = V(G)$ such that there is no edge between $A \setminus B$ and $B \setminus A$. The *order* of a separation (A, B) is the size $|A \cap B|$. A *α-balanced S-separation* is a separation (A, B) which additionally satisfies $|(A \setminus B) \cap S| \leq \alpha \cdot |S|$ and $|(B \setminus A) \cap S| \leq \alpha \cdot |S|$.

We use three key lemmas. The proofs of Lemmata 1 and 2 can be found in the textbook by Cygan et al. [2] as Lemma 7.19 and Lemma 7.20

Lemma 1. *Let G be a graph of treewidth at most w and let S be an arbitrary vertex subset of G. Then there is a $\frac{1}{2}$-balanced S-separator of G consisting of at most $w+1$ vertices.*

Lemma 2. *Let G be a graph and let $S \subseteq V(G)$. If $X \subseteq V(G)$ is $\frac{1}{2}$-balanced S-separator of G, then there is a $\frac{2}{3}$-balanced S-separation (A, B) such that $A \cap B = X$.*

Lemma 3. *There is an algorithm which, given a graph G, a vertex set $S \subseteq V(G)$ and an integer $w \geq 1$, either finds a $\frac{2}{3}$-balanced S-separation of order $w+1$ or correctly concludes that there is no such separation of G in time $O(3^{|S|} \cdot wm)$.*

Proof: Consider a tripartition of S into (S_A, S_X, S_B) and pick an arbitrary (S_A, S_B)-separator Y of $G - S_X$. Then for a connected component C of $G - (S_X \cup Y)$, observe that C intersects at most one of S_A and S_B because $S_X \cup Y$ is a (S_A, S_B)-separator in G. Now, let (A', B') be a bipartition of the connected components of $G - (S_X \cup Y)$ so that $\mathcal{A}'$ contains all connected components of $G - (S_X \cup Y)$ intersecting S_A and $\mathcal{B}'$ contains the remaining components. Those connected components intersecting neither of S_A and S_B are put arbitrarily. Define $A := \bigcup_{C \in \mathcal{A}'} C \cup (S_X \cup Y)$ and $B := \bigcup_{C \in \mathcal{B}'} C \cup (S_X \cup Y)$, and note that (A, B) is a separation with $A \cap B = (S_X \cup Y)$ such that $S \cap (A \setminus B) = S_A$, $S \cap (B \setminus A) = S_B$ and $S \cap (S_X \cup Y) = S_X$.

We summarize it in the next claim.

Claim 1. *Let (S_A, S_X, S_B) be a tripartition of S and Y be a (S_A, S_B)-separator of $G - S_X$. Then there is a separation (A, B) of G such that $S \cap (A \setminus B) = S_A$, $S \cap (B \setminus A) = S_B$ and $S \cap (A \cap B) = S_X$. Moreover, one can construct such a separation in polynomial time.*

The algorithm works as follows. We guess all possible tripartitions of S into (S_A, S_X, S_B) such that $|S_A| \leq \frac{2}{3} \cdot |S|$ and $|S_B| \leq \frac{2}{3} \cdot |S|$. For each guess (S_A, S_X, S_B), we invoke max-flow min-cut algorithm to find a (S_A, S_B)-separator of Y minimum size in $G - S_X$. If $|S_X \cup Y| \leq w+1$, then by Claim 1 we can obtain a $\frac{2}{3}$-balanced S-separation of order at most $w+1$. We output such a separation (A, B).

Suppose for every guess (S_A, S_X, S_B) and for every minimum-size (S_A, S_B)-separator of Y in $G - S_X$ found with max-flow min-cut algorithm, we have $|S_X \cup Y| > w+1$. We argue that G has no $\frac{2}{3}$-balanced S-separation of order at most $w+1$.

Indeed, if there is a $\frac{2}{3}$-balanced S-separation (A, B) of order $w+1$, then let $X = A \cap B$. Let $S_A := S \cap (A \setminus B)$, $S_X := S \cap X$ and $S_B := S \cap (B \setminus A)$, and observe that (S_A, S_X, S_B) forms a tripartition of S with $|S_A| \leq \frac{2}{3} \cdot |S|$ and $|S_B| \leq \frac{2}{3} \cdot |S|$. Moreover, X separates S_A and S_B in G, and thus $X \setminus S_X$ is an (S_A, S_B)-separator in $G - S_X$ of size at most $w+1-|S_X|$. Now, another (S_A, S_B)-separator Z in $G - S_X$ could have been returned by the max-flow min-cut algorithm and notice that $|S_X \cup Z| \leq |S_X| + |Z| \leq |S_X| + (w+1-|S_X|) \leq w+1$. This contradicts the assumption that no such separator was found for any considered tripartition of S.

The factor $2^{|S|}$ in the running time comes for the upper bound on the number of tripartitions on S. The polynomial part comes from the running time of Ford-Fulkerson algorithm for solving max-flow min-cut. □

The algorithm for treewidth is described below. Let $\mathcal{A}(G, S, w)$ be an algorithm as depicted in Lemma 3. To compute a tree decomposition of G, we execute the procedure **compute-tw** on $(G, \emptyset, w)$.

Algorithm 1 Algorithm for computing treewidth

```
1:  procedure compute-tw(G, S, w)
2:      if |V(G)| ≤ 4w + 3 then return ({z}, {(z, V(G)}).  ▷ Tree on a single node z with χ(z) = V(G)
3:      Expand S up to size 3w + 2 by adding arbitrary vertices of V(G) \ S.
4:                                                  ▷ |V(G)| ≥ 4w + 4 and |S| = 3w + 2.
5:      if A(G, S, w) = NO then return "tw > w."
6:      else
7:          Let (A, B) be the separation returned by A(G, S, w) and let X := A ∩ B.
8:          if compute-tw(G[A], (A ∩ S) ∪ X, w)="tw > w" then return "tw > w."
9:          if compute-tw(G[B], (B ∩ S) ∪ X, w)="tw > w" then return "tw > w."
10:         else
11:             Let (T_A, χ_A) = compute-tw(G[A], (A ∩ S) ∪ X, w)
12:             Let (T_B, χ_B) = compute-tw(G[B], (B ∩ S) ∪ X, w)
13:             Create a bag Z := S ∪ X.
14:             T is the tree obtained by creating a node z and connecting z with root(T_A) and root(T_B).
15:             Let χ be the 'canonical' mapping from V(T) to 2^{V(G)} with χ(z) = Z.
16:             return (T, χ)
```

Correctness. To begin with, we argue that $|S|$ when **compute-tw**(G, S, w) is called has size at most $3w+1$ and the execution at line 3 is valid. We prove it by induction on the length of the path from the root call to the current recursive call. For the root call, this is trivially true because $S = \emptyset$. Assume that at the point **compute-tw**(G, S, w) is called $|S| \leq 3w+1$. When $\mathcal{A}$ returns a separation (A, B) on the instance (G, S, w) *after* expanding S up to size $3w+2$, that (A, B) is a $\frac{2}{3}$-balanced S-separation implies $|(A \cap S) \cup X| = |((A \setminus B) \cap S) \cup X| \leq 2w + (w+1) \leq 3w+1$, as desired. Therefore, at line 11-12, the calls are made with the second argument having size at most $3w+1$.

If **compute-tw**(G, S, w) returns "tw $> w$", there is a triple (G', S', w) for some induced subgraph G' of G and a vertex subset S' of G' such that $\mathcal{A}(G', S', w) =$ NO. In this case $\mathsf{tw}(G') > w$ since otherwise, by Lemmas 1 and 2, the procedure $\mathcal{A}$ on (G', S', w) would have detected a balanced S'-separation of order at most $w+1$. From $\mathsf{tw}(G) \geq \mathsf{tw}(G') > w$, the output of **compute-tw**(G, S, w) is correct.

It remains to show that when **compute-tw**(G, S, w) does not return "tw $> w$", that is, output some pair (T, χ), then it is a tree decomposition of G of width at most $4w+2$. We prove a stronger claim by induction

on $|V(G)|$.

($\star$) If **compute-tw**(G, S, w) does not return "tw $> w$", then it outputs a tree decomposition of G such that the width is at most $4w$ and the root bag contains S.

When $|V(G)| \leq 4w + 3$, the output pair (T, χ) trivially satisfies ($\star$).

Note that $1 \leq |(A \setminus B) \cap S| \leq 2w$ and $1 \leq |(B \setminus A) \cap S| \leq 2w$ hold as (A, B) is a $\frac{2}{3}$-balanced S-separation and $|S| = 3w + 2$ in line 4, This implies $|A| < |V(G)|$ and $|B| < |V(G)|$. Assume that (T_A, χ_A) and (T_B, χ_B) be tree decompositions of $G[A]$ and $G[B]$ respectively satisfying ($\star$). Let z_A and z_B be the roots of the former and the latter tree decomposition respectively.

Let us check that (T, χ) is a tree decomposition of G. For any vertex v or an edge e of G the vertex and edge coverage property hods for v or e by induction hypothesis; that is, for $V(G) = A \cup B$ without loss of generality we have $v \in A$ and the vertex coverage of the (sub)tree decomposition $(T_{z_A}, \chi|_{V(T_{z_A})}) = (T_A, \chi_A)$ for v implies the vertex coverage of (T, χ) for v. For an edge e of G, notice that either both endpoints of e are in A or both are in B because there is no edge between $(A \setminus B, B \setminus A)$ and the same argument works to show the edge coverage of e in (T, χ).

To verify the connectivity condition, we consider a few cases. When $w \notin S \cup X$, without loss of generality we may assume $w \in A \setminus B \setminus S$. Then the bags containing w are only contained in the subtree decomposition of (T, χ) rooted at z_A and the induction hypothesis ensure the connectivity of w. For $v \in A \cap B$, note that $A \cap B = X$ are contained in the root bags of both (T_A, χ_A) and (T_B, χ_B) as well as the bag $Z = S \cup X$, thus the bags of T containing v are connected. For $v \in A \cap S$ (the case when $v \in B \cap S$ is symmetric), note that v must be contained in the bag of z_A and in the bag of z. By induction hypothesis, the nodes of T_A whose bags containing v form a subtree of T_A rooted at z_A, and thus the nodes of T whose bags containing v form a subtree of T.

That the root bag of (T, χ) contains S is valid by construction. It remains to verify that (T, χ) has width at most $4w + 2$. This is trivial from the induction hypothesis and that $|S \cup X| \leq 3w + 2 + w + 1 \leq 4w + 3$.

Runtime. At each execution of **compute-tw**(G, S, w), the most expensive part is the execution of the procedure $\mathcal{A}$ in line 5, which takes $O(3^{|S|} \cdot wm) = O(27^w \cdot wm)$ by Lemma 3. Therefore, if we prove that the constructed tree decomposition has at most n nodes, the claimed running time of the main theorem is settled. The key observation is that at line 3, we add at least one vertex to S (otherwise, we terminated the current call at line 2) because the previous recursive call sets $|S| \leq 3w + 1$ as discussed above. Moreover, any newly added vertex at line 3 does not appear in the parent node (why?). Therefore, with the tree decomposition (T, χ) returned at the end of the full execution of the algorithm, one can define the mapping $\mathsf{top} : V(G) \to V(T)$ so that $\mathsf{top}(v)$ is the topmost node whose bag contains v. Now that this mapping is surjective, it holds that $V(T) \leq V(G) = n$. This settles the claimed running time.

References

[1] Bruce A. Reed. Surveys in combinatorics, 1997: Tree width and tangles: A new connectivity measure and some applications. 1997.

[2] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.