CS762: Graph-Theoretic Algorithms
Lecture 15: Solving **Independent Set** on a tree-decomposition of small tree-width. Recognition of k-trees
February 8, 2002

Scribe: Vlad Ciubotariu

Abstract

We show how to solve **Independent Set** efficiently, using dynamic programming, on a tree-decomposition and provide a correctness proof and time analysis. Then we briefly discuss the related notion of *fixed parameter tractability* and show how to recognize k-trees in linear time. Unfortunately, recognizing partial k-trees is NP-hard.

1 Introduction

We continue with the bulk of the algorithm that solves **Independent Set** on a graph G with a given tree-decomposition. We consider a 'regular' tree-decomposition $T = (I, F)$ which satisfies the following:

- for all $i \in I$, $|X_i| = k + 1$
- for all $(i, j) \in F$, $|X_i - X_j| = 1$

This is achievable for every graph with a tree-decomposition of tree-width k. Furthermore, we consider the tree decomposition to be rooted at one arbitrary node r. Every node will fall in exactly one of the two categories: interior node or leaf.

2 An algorithm for solving Independent Set

We use dynamic programming to solve **Independent Set** on graphs G that have a tree-decomposition $T = (I, F)$ of tree-width k. The notion of subproblem is defined as follows:

- for a node i of T, we define Y_i to be the set $\{v \in X_j \mid j \text{ is } i \text{ or } j \text{ is a descendant of i}\}$.
- $Z \subseteq X_i$ is called a configuration in node i.
- The subproblem $is_i(Z)$ is to find the size of a maximum independent set IS in $G[Y_i]$ so that $IS \cap X_i = Z$. We denote the optimal solution to this subproblem by $is_i(Z)$.

The optimal solution to the initial problem is obtained by picking a configuration in the root node that maximizes the size of the independent sets over all 2^{k+1} possible configurations in that node:

$$\alpha(G) = \max_{Z \subseteq X_r} \{is_r(Z)\} \tag{1}$$

We proceed now to identify the recursive relation that holds between a problem and its subproblems.

2.1 Base Case

Let the base case be when the node i is a leaf.

$$is_i(Z) = \begin{cases} |Z|, & \text{if } |Z| \text{ is an independent set} \\ -\infty, & \text{otherwise} \end{cases} \tag{2}$$

In other words, solving a subproblem in a leaf node amounts to checking whether the respective configuration is independent in G.

2.2 Induction Step

Node i is an interior node. We split this case in two sub-cases:

2.2.1 Case 1

Node i has exactly one child. Remember that the tree-decomposition satisfies the following:

- for all $i \in I$, $|X_i| = k + 1$
- for all $(i, j) \in F$, $|X_i - X_j| = 1$

Let j be the child of i. There exist $v, w \in V(G)$ uniquely determined so that

$$X_j = X_i - v + w \tag{3}$$

Claim 1 *Given a configuration Z in node i the following holds:*

$$is_i(Z) = \begin{cases} -\infty, & \textit{if } Z \textit{ is not independent} \\ \max\{is_j(Z), is_j(Z + w)\}, & \textit{if } v \notin Z \\ 1 + \max\{is_j(Z - v), is_j(Z - v + w)\}, & \textit{if } v \in Z \end{cases} \tag{4}$$

Proof: Suppose Z is independent. Assume $v \in Z$, the remaining case $v \notin Z$ is similar. We'll first show that

$$is_i(Z) \leq 1 + \max\{s_j(Z - v), is_j(Z - v + w)\}$$

Let IS be a solution to the configuration Z in i. According to definition, $IS \cap X_i = Z$. Because of equation 3, $X_j \cap (IS - v)$ is either $Z - v$ or $Z - v + w$ depending on whether w belongs to IS. At the same time, $IS - v \subseteq Y_j$, and so it is an independent set in $G[Y_j]$. And so, the previous inequality follows immediately.

We now show that

$$is_i(Z) \geq 1 + \max\{s_j(Z - v), is_j(Z - v + w)\}$$

Consider now node j, in either of the following configurations: $Z - v$ or $Z - v + w$. Solutions to these subproblems correspond to independent sets in $G[Y_j]$. We show that to any such independent set IS node v can be added with the consequence that $IS + v$ is independent and $(IS + v) \cap X_i = Z$.

Let IS so that $IS \cap X_j = Z - v$ or $IS \cap X_j = Z - v + w$. Given that $w \notin X_i$ and equation 3, $(IS + v) \cap X_i = Z$. Naturally, $IS + v \subseteq Y_i$. We only need show $IS + v$ is independent. Suppose to the contrary that there exists $u \in IS$ so that $(u, v) \in E(G)$. Because $v \in X_i$ and $v \notin X_j$, $v \notin Y_j$. Let k be the node such that $u, v \in X_k$. Because X_k contains v, k is either i or a node reachable from i. In any case, since u appears in Y_j, i lies on a path between two nodes that contain u in their labels. It follows that $u \in X_i$. $u \in IS$ and $(IS + v) \cap X_i = Z$ imply that u is in Z. But that cannot be because Z is independent. The desired inequality follows as an immediate consequence. $\square$

2.2.2 Case 2

Assume i has $s \geq 2$ descendants: $j_1, \ldots, j_s$. Let $is_i^l(Z)$ be the would-be value of $is_i(Z)$ computed by the recurrence relation 4 given in section 2.2.1 if j_l were the only descendant of i.

Claim 2

$$is_i(Z) = \sum_{l=1}^{s} is_i^l(Z) - (s-1)|Z| \tag{5}$$

Proof: We follow an argument similar to the one in section 2.2.1. Equation 3 becomes in this case

$$X_{j_l} = X_i - v + w_l, l \in \{1, \ldots, s\} \tag{6}$$

An optimum solution for i in configuration Z induces an optimum solution for i in isolation with j_l, $l \in \{1, \ldots, s\}$ and vice-versa. We'll prove only one direction of the equivalence. Let IS_l be a solution to the subproblem in node i taken in isolation with j_l, with configuration Z. Now

$$IS_l \cap X_i = Z, l \in \{1, \ldots, s\}$$

and so, given $l_1 \neq l_2 \in \{1 \ldots s\}$, $u \in IS_{l_1} \cap IS_{l_2}$ implies $u \in X_i$ which in turn implies $u \in Z$. Therefore,

$$|\bigcup_{l=1}^{s} IS_l| = \sum_{l=1}^{s} |IS_l| - (s-1)|Z|.$$

It remains to show that $\bigcup_{l=1}^{s} IS_l$ is an independent set. Let $l_1 \neq l_2 \in \{1, \ldots, s\}$ and assume to the contrary that there are $u \in IS_{l_1}, w \in IS_{l_2}$ such that $(u, w) \in E(G)$. Let T_u be the subtree rooted at i and containing i, j_{l_1} and its descendants, and symmetrically, T_w. There must exist a node p in the tree-decomposition T such that $\{u, w\} \subseteq X_p$. Suppose p is in T_u. Then w appears in a label in T_u and so w must also appear in X_i. Since $w \in IS_{l_2} \cap X_i$, $w \in Z$. And so, $w \in IS_{l_1}$. Contradiction because IS_{l_1} is independent. Similarly if $p \in T_w$. Otherwise, if $p \notin T_u$ and $p \notin T_w$ then it will follow that i lies on a path between two vertices that include in their labels u (it also holds for w). It follows that u is in Z and again a contradiction is derived.

We proved that

$$is_i(Z) \geq \sum_{l=1}^{s} is_i^l(Z) - (s-1)|Z|.$$

□

2.3 Complexity

We assume the solution is computed bottom-up.

2.3.1 Leaves

In a leaf we resolve all possible configurations, and so perform

$$\sum_{i=0}^{k+1} \binom{k+1}{i} \binom{i}{2} = 2^{k-2}k(k+1)$$

operations. Because k is constant and we have $O(n)$ leaves it follows that in leaves we spend $O(2^{k+1}n)$ computation time.

2.3.2 Interior Nodes

In an interior node i, for a given configuration we need $O(deg(i))$ time to compute equation 4 or equation 5, as the case may be. It follows that we'll need $O(\sum_{i \text{ interior}} 2^{k+1} deg(i)) = O(2^{k+1} n)$ time.

2.4 Related Work

Many other problems, otherwise NP-complete, can be solved efficiently on graphs G with a given tree-decomposition of small tree-width, among them ([2]):

- **Vertex Cover**
- **Maximum Matching**
- **Dominating Set**
- **Hamiltonian Cycle**

Bodlaender also mentions **Graph Isomorphism** ([3], [4]), but says that this is done with a different technique.

3 Fixed Parameter Tractability

Many problems are defined in terms of one or more parameters. One example is solving **Independent Set** on a tree-decomposition of *tree-width* k; k here is the parameter.

Definition 1 *A problem is* fixed parameter tractable *in k if there is an algorithm that takes $O(f(k) \cdot n^c)$ time to solve an instance of size n.*

Remark 1 *$f(k)$ depending only on k is independent of n and so the respective problem is polynomial or* tractable *when k is* fixed*.*

4 Recognizing partial k-trees

Now we can turn to recognizing partial k-trees. We study first k-trees and recall their definition:

Definition 2 *A graph G is called a* k-tree *if*

- *G is chordal*
- *G has a partial elimination order $v_1, \ldots, v_n$ such that*

$$indeg(v_i) = k, \textit{for all } i \geq k+1 \tag{7}$$

A *partial k-tree* is defined as a spanning subgraph of a k-tree.

The connection between graphs with tree-width at most k and partial k-trees is given by the following result:

Theorem 1 (Scheffler [6], Wimer [7]) *G has tree-width at most k if and only if G is a partial k-tree.*

One can show the following result (this was left as an exercise):

Lemma 1 *If G is a k-tree then every partial elimination order of G satisfies condition 7.*

Therefore, recognizing k-trees can be done in linear time ($O(m+n)$) by computing a perfect elimination order and then testing for condition 7.

On the other hand, recognizing partial k-trees is NP-hard.

Theorem 2 (Arnborg, Corneil, Proskurowski [1]) *Given a graph G and value k, testing whether G has tree-width at most k is NP-complete.*

However, we can get an approximation of the tree-width.

Theorem 3 (Reed [5]) *For every constant k, there exists an $O(n \log n)$ algorithm, that given a graph $G = (V, E)$, either outputs that the tree-width of G is larger than k, or outputs a tree-decomposition of G with tree-width at most $3k + 2$.*

References

[1] S. Arnborg, D.G. Corneil, A. Proskurowski, *Complexity of finding embeddings in a k-tree*, SIAM, J. Alg. Discrete Methods, 8 (1987), 277-284.

[2] S. Arnborg, A. Proskurowski, *Linear time algorithms for NP-hard problems restricted to partial k-trees*, Discrete Appl. Math., 23 (1989), 11-24.

[3] H.L. Bodlaender, *A tourist guide through tree-width*, Acta Cybernet., 11 (1993), 1-23.

[4] H.L. Bodlaender, *Dynamic programming on graphs with bounded tree-width*, in Proc. 15th International Colloqu. on Automata, Languages and Programming, *Lecture Notes in Comput. Sci.*, 317 (1988), 105-119.

[5] B. Reed, *Finding approximate separators and computing the tree-width quickly*, in Proc. of the 24th Annual Symposium on Theory of Computing, (1992), 221-228.

[6] P. Scheffler, *The graphs of tree-width k are exactly the partial k-trees*, manuscript (1986).

[7] T.V. Wimer, *Linear algorithms on k-terminal graphs*, Ph.D. thesis, Dept. of Computer Science, URI-030, Clemson University, Clemson, SC (1987).