The k-in-a-tree problem for chordal graphs[*]

Vinícius F. dos Santos Murilo V. G. da Silva
Jayme L. Szwarcfiter

Abstract

Algorithms for detecting particular induced subgraphs have been the focus of much research recently, mostly related to their connection to many classes of graphs defined by forbidden induced subgraphs. In this context, Chudnovsky and Seymour proposed a useful tool, called *three-in-a-tree algorithm* which solves the following problem in polynomial time: given a graph with three prescribed vertices, test if there is an induced tree containing these vertices. In our paper we deal with a generalization of this problem, known as *k-in-a-tree problem.* For the case where k is part of the input, the problem is known to be NP-complete. For fixed k, the complexity of this problem is open for $k \geq 4$, although there are polynomial time algorithms for restricted cases, such as claw-free graphs and graphs with girth at least k. In this paper we give a $\mathcal{O}(nm^2)$ time algorithm for the k-in-a-tree problem for chordal graphs, even in the case where k is part of the input. Furthermore, the algorithm outputs an induced tree containing the k prescribed vertices when there is such tree.

1 Introduction

A very useful tool for detecting induced subgraphs, known as *three-in-a-tree algorithm*, was proposed in 2006 by Chudnovsky and Seymour [3]. This algorithm deals with the following problem, known as the *three-in-a-tree problem*: Given a graph $G = (V, E)$ and three vertices $u, v, w \in V$ as input, find in polynomial time a tree T containing u, v, w such that T is an induced subgraph of G, or report that no such tree exists. In the original paper, the authors also used this algorithm as a tool for testing for certain forbidden induced subgraphs such as *thetas* and *pyramids* (these two structures play an important role in the recognition algorithm for perfect graphs and more generally the realm of classes of graphs defined by forbidden induced subgraphs [10]). Lévêque et al. [8] show that the three-in-a-tree algorithm can be used for testing for whole class of forbidden induced subgraphs and more recently Chang and Lu [2] also use the algorithm for testing for some structures in their algorithm for the recognition of even-hole-free graphs.

[*]Partially supported by CAPES and CNPq.
Keywords and phrases: k-in-a-tree, three-in-a-tree, induced subgraphs, chordal graphs.

The problem that we are dealing in this paper, which is known as the *k-in-a-tree problem* is a natural generalization of the three-in-a-tree problem in the case where the number of prescribed vertices is any natural $k \geq 3$ (note that for $k = 1$ and $k = 2$ the problem is trivial). For the sake of clarity, we may also assume that the input graph for this problem is connected.

The complexity of the k-in-a-tree problem for any $k \geq 4$ is still an open problem. Polynomial time algorithms for the k-in-a-tree problem for $k \geq 4$ are known only for the restricted cases of claw-free graphs, by Fiala et al. [6], and graphs with girth at least as large as k by Trotignon and Wei [9] (which is a generalization of a work by Derhy et al. [5]).

An important point to make is that in the k-in-a-tree problem, k is a fixed number. The version of this problem when k is part of the input is $\mathcal{NP}$-hard [4]. Although the complexity of k-in-a-tree is open for $k \geq 4$, many variations of this problem are known to be $\mathcal{NP}$-hard. Namely, it is $\mathcal{NP}$-hard the problem of testing for the following subgraphs containing k prescribed vertices: an induced cycle [1], for $k \geq 2$, a *minimum* induced tree [4], for $k \geq 3$, and an induced path [7], for $k \geq 3$ vertices.

In this paper we give a $\mathcal{O}(nm^2)$ time algorithm for the k-in-a-tree problem for chordal graphs, even in the case where k is part of the input. The algorithm also outputs an induced tree containing the k prescribed vertices when the input contains such induced tree.

2 The K-IN-A-TREE problem for chordal graphs

Throughout this section let $G = (V, E)$ be a connected chordal graph and $S \subseteq V$ be a set of k vertices for a fixed k. For any $A \in V(G)$, we denote the subgraph of G induced by A by $G[A]$. Let $\mathcal{C} = \{C_1, \ldots, C_l\}$ be the set of its maximal cliques. A clique-tree of G is a tree $\mathcal{T}$ with vertex set $V(\mathcal{T}) = \mathcal{C}$ such that $\mathcal{T}$ satisfies the following property: For every pair of distinct maximal cliques $C_i, C_j \in \mathcal{C}$, the set $C_i \cap C_j$ is contained in every clique in the path from C_i to C_j in the tree $\mathcal{T}$.

Definition: the K-IN-A-TREE problem is that of testing if G contains a set of vertices A such that $G[A]$ is a tree and $S \subseteq A$. If G contains such set of vertices, then we say that G contains an *induced-S-tree.* When the set S is clear from the context, we may say only that G contains an *induced tree.* When the input graph is a chordal graph, we call this problem K-IN-A-TREE-CHORDAL.

At the end of this section we present an algorithm for the K-IN-A-TREE-CHORDAL problem (Algorithm 1) and also provide an algorithm that outputs an induced-S-tree if the input chordal graph contains such tree (Algorithm 2). In fact our algorithms also solves the general case where k is part of the input. We give below an outline of the key ideas behind the algorithm.

(1) Initially we "trim" G removing simplicial vertices that are not in S since we do not need them in any possible induced-S-trees that could be present in G. This is almost straightforward and is the subject of Claim 1.

(2) We show that if this "trimmed" graph contains an induced-S-tree T, then every clique C in the clique-tree of G contains precisely one or two vertices of T. We prove that in Claim 2. We call these (one or two) vertices the *candidates* for clique C.

(3) The algorithm is conceptually simple and can be seen as a dynamic programming strategy going through every clique (always picking up cliques that are leaves in the clique tree) C trying to match the candidates of C to the candidates of its neighboring clique. However, in order for this strategy to work, we need to deal with many technicalities which are the subject of Claims 3, 4, 5 and 6. The main (inductive) idea of the algorithm going through every clique is the subject of Claim 7.

The following claims are used in the proof of correctness of the algorithm at the end of the paper. For space reasons, the proofs are omitted.

Claim 1 *Suppose that $x \in V \setminus S$ is a simplicial vertex of G. If G contains an induced-S-tree, then $G \setminus x$ also contains an induced-S-tree.*

Note that by Claim 1, if we are looking for an induced-S-tree in G, we can first repeatedly remove every simplicial of G in $V \setminus S$ and then look for the induced-S-tree in the remaining graph. For the next claims let $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ be a clique-tree of G where $\mathcal{C}$ is the set of its maximal cliques.

Claim 2 *Suppose that every simplicial vertex of G is in S and let A be any maximal clique of $\mathcal{T}$. Suppose that G contains an induced-S-tree T. Then $1 \leq |V(T) \cap A| \leq 2$.*

Let $A, X \in \mathcal{C}$ be maximal cliques of G such that A and X are adjacent in $\mathcal{T}$.

Claim 3 *Let ab (resp. xy) be an edge of A (resp. X). Suppose that G contains an induced-S-tree T. If $ab, xy \in E(T)$, then the following conditions hold:*

(i) $\{a, b\} \cap \{x, y\} \neq \emptyset$.

(ii) $G[\{a, b, x, y\}]$ does not contain a triangle.

Claim 4 *Let ab be an edge of A and x be a vertex of X. If $ab \in E(T)$ and x is the unique vertex of X in T, then $x \in \{a, b\}$.*

Claim 5 *Let $a \in A$ and $x \in X$. Suppose that G contains an induced-S-tree T. If $A \cap V(T) = \{a\}$ and $X \cap V(T) = \{x\}$, then $a = x$.*

In what follows, the symbol $\ominus$ denotes the symmetric difference of two sets.

Claim 6 *Let $A' \subseteq A$ and $X' \subseteq X$ be such that $1 \leq |A'| \leq 2$ and $1 \leq |X'| \leq 2$. Suppose that G contains an induced-S-tree T. If $V(T) \cap A = A'$ and $V(T) \cap X = X'$, then the following three conditions hold:*

(i) $A' \cap X' \neq \emptyset$.

(ii) $A' \ominus X' \subseteq A \ominus X$.

(iii) $G[A' \cup X']$ *does not contain a triangle.*

For every $C_i \in \mathcal{T}$, we define $L(C_i)$ as the list containing every subset $X \subseteq V(C_i)$ respecting the following two conditions:

Condition (1) $1 \leq |X| \leq 2$.

Condition (2) $V(C_i) \cap S \subseteq X$.

Let A be a leaf in $\mathcal{T}$ and X be its unique neighbor in $\mathcal{T}$. For every given $X' \in L(X)$ we define $F_A(X')$ as the list of every set $A' \in L(A)$ such that the following conditions are satisfied:

Case 1: If $|A'| = |X'| = 2$, then $A' \cap X' \neq \emptyset$ and $G[A' \cup X']$ does not contain a triangle.

Case 2: If $|A'| = 1$ and $|X'| = 2$, then $A' \subset X'$.

Case 3: If $|A'| = 2$ and $|X'| = 1$, then $X' \subset A'$.

Case 4: If $|A'| = |X'| = 1$, then $A' = X'$.

Let $B = A \setminus X$. Since $\mathcal{T}$ is a clique-tree, $B \neq \emptyset$. Let $G' = G[V \setminus B]$. Let $S' = S \setminus B$.

Claim 7 *G has an induced-S-tree T such that for every $C \in V(\mathcal{T})$, $1 \leq |V(C) \cap V(T)| \leq 2$ if and only if the following conditions hold:*

(i) G' has an induced-S'-tree T'.

(ii) $\exists X' \in L(X)$ such that $F_A(X') \neq \emptyset$.

(iii) In the induced-S'-tree T' we have $V(T') \cap X = X'$.

Theorem 1 *Algorithm 1 solves* k-IN-A-TREE-CHORDAL *problem.*

The proof of Theorem 1, which is omitted, relies on Claim 7. The proof of Claim 7 relies on the proof of Claims 1 – 6.

Algorithm 1: Decides if a chordal graph G contains an induced-S-tree

```
input : $G = (V, E)$ and $S \subseteq V(G)$ (set of $k$ prescribed vertices)
Remove iteratively every simplicial vertex of $G$ which is not in $S$;
$\mathcal{C} \leftarrow \{C_1, \ldots, C_l\}$ be the list of maximal cliques of $G$;
If $\exists C_i$ s.t. $|C_i \cap S| \geq 3$ then return No
for each $C_i$ do
    Let $L(C_i)$ be the list of every set $X \subseteq V(C_i)$ such that $1 \leq |X| \leq 2$;
end
for each $C_i$ do
    for each set $X \in L(C_i)$ do
        If $C_i \cap S \nsubseteq X$ then Remove $X$ from $L(C_i)$
    end
end
Let $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ be a clique-tree of $G$;
while $|\mathcal{C}| > 1$ do
    Let $A \in \mathcal{C}$ be a leaf of $\mathcal{T}$ and $X \in \mathcal{C}$ its unique neighbor in $\mathcal{T}$;
    $\forall X' \in L(X)$, let $F_A(X') = \emptyset$;
    for every $A'$ in $L(A)$ do
        for every $X'$ in $L(X)$ do
            If $A' \cap X' \neq \emptyset$, $A' \ominus X' \subseteq A \ominus X$ and $G[A' \cup X']$ does not
            contain a triangle then Insert $A'$ in the list $F_A(X')$;
        end
    end
    for every $X'$ in $L(X)$ do
        If $F_A(X') = \emptyset$ then Remove $X'$ from $L(X)$;
    end
    If $L(X)$ is empty then return No
    Remove $A$ from $\mathcal{T}$ and from $\mathcal{C}$;
end
return Yes
```

Algorithm 2: Builds an induced-S-tree.

```
input : A clique tree $\mathcal{T}$ of $G$, lists $F_C$, the last remaining clique $C$ of $\mathcal{C}$
        in Algorithm 1, $X \in L(C)$ and an empty graph $T = (\emptyset, \emptyset)$.
BuildTree($C$, $X$);
return $G[V(T)]$;
procedure BuildTree($C$, $X$);
Mark $C$;
$V(T) \leftarrow V(T) \cup \{X\}$;
for each unmarked neighbor $C'$ of $C$ do
    Let $Y \in F_{C'}(X)$;
    BuildTree($C'$, $Y$);
end
```

Let C be the last clique remaining in $\mathcal{C}$ after the execution of Algorithm 1. We call Algorithm 2 with parameters C and X, for any $X \in L(C)$. Algorithm 2 outputs an induced-S-tree T. Initially, $V(T) = E(T) = \emptyset$.

Theorem 2 *Let G be a chordal graph with n vertices and m edges and let S be a subset of $V(G)$. An induced-S-tree of G is found by Algorithm 2 in time $\mathcal{O}(nm^2)$ if such tree exists.*

References

[1] D. Bienstock, *On the complexity of testing for odd holes and induced odd paths.* Discrete Mathematics 90 (1991), 85-92

[2] H-C. Chang and H-I Lu, *A faster algorithm to recognize even-hole-free graphs.* Journal of Combinatorial Theory, Series B 113 (2015), 141161

[3] M. Chudnovsky and P. Seymour, *The three-in-a-tree problem.* Combinatorica 30 (2010), 387-417

[4] N. Derhy and C. Picouleau, *Finding induced trees.* Discrete Applied Mathematics 157 (2009), 3552-3557

[5] N. Derhy, C. Picouleau and N. Trotignon *The Four-in-a-Tree Problem in Triangle-Free Graphs* Graphs and Combinatorics 25 (2009), 489-502

[6] J. Fiala, M. Kamiński, B. Lidický and D. Paulusma *The k-in-a-path problem for claw-free graphs.* Symposium on Theoretical Aspects of Compuper Science (2010), 371-382

[7] R. Haas and M. Hoffmann, *Chordless paths through three vertices.* Theoretical Computer Science 351 (2006), 360-371

[8] B. Lévêque, D.Y. Lin, F. Maffray and N. Trotignon, *Detecting induced subgraphs.* Discrete Applied Mathematics 157 (2009), 3540-3551

[9] N. Trotignon and L. Wei, *The k-in-a-tree problem for graphs of girth at least k.* Discrete Applied Mathematics 158 (2010), 1644-1649

[10] K. Vušković, *The world of hereditary graph classes viewed through Truemper configurations.* Surveys in Combinatorics, London Math Soc, 2013.

DECOM, Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, Brazil.
vinicius.santos@gmail.com

DAINF, Universidade Tecnológica Federal do Paraná, Curitiba, Brazil.
murilo@utfpr.edu.br

IM, NCE, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil.
jayme@nce.ufrj.br