Fully leafed induced subtrees (extended abstract)*

A. Blondin Massé[1], J. de Carufel[2], A. Goupil[2], M. Lapointe[1], É. Nadeau[1], É. Vandomme[1]

[1] Laboratoire de Combinatoire et d'Informatique Mathématique, Université du Québec à Montréal, Canada
[2] Laboratoire Interdisciplinaire de Recherche en Imagerie et en Combinatoire, Université du Québec à Trois-Rivières, Canada

Abstract

Subtrees of graphs, as well as their number of leaves, have been investigated by various communities: from discrete mathematics to data mining and information retrieval. We consider a variant where we require the subtrees to be induced and compute their maximal number of leaves. The problem, which is NP-complete in general, becomes polynomial in the case of trees. The leaf function associates to a number n the maximal number of leaves an induced subtree of size n can have. To compute the leaf function, we provide an efficient branch and bound algorithm. In the particular case of trees, we provide a polynomial algorithm using the dynamic programming paradigm.

Keywords : *graph theory, induced subtrees, optimization problem, number of leaves*

1 Introduction

In the past decades, many researchers coming from various communities extensively studied subtrees of graphs and their number of leaves. For instance in 1984, Payan *et al.* [9] discussed the maximum number of leaves, called the *leaf number*, that can be realized by a spanning tree of a given graph. This problem, called the *Maximum Leaf Spanning Tree problem* (MLST), is known to be NP-complete even in the case of regular graphs of degree 4 [8] and has attracted interest in the telecommunication network community [3, 4]. The *frequent subtree mining problem* [5] investigated in the data mining community, has applications in biology. The detection of subgraph patterns such as induced subtrees is useful in information retrieval [12] and requires efficient algorithms for the enumeration of induced subtrees. In this perspective, Wasa *et al.* [11] proposed an efficient parametrized algorithm for the generation of induced subtrees in a graph. Note that the *induced* property requirement brings an interesting constraint on subtrees, yielding distinctive structures with respect to other constraints such as in the MLST problem. A first result given by to Erdös *et al.* in 1986, showed that the problem of finding an induced subtree of a given graph G with more than i vertices is NP-complete [7].

Among induced subtrees of simple graphs, we focus in particular on those with a maximal number of leaves. We call these objects *fully leafed induced subtrees* (FLIS). Particular instances of the FLIS recently appeared in a paper of Blondin Massé *et al.* [2], where the authors considered the maximal number of leaves that can be realized by tree-like polyominoes, respectively polycubes. Their investigation led to the discovery of a new 3D tree-like polycube structure that realizes the maximal number of leaves constraint. The observation that tree-like polyominoes and polycubes are induced subgraphs of the lattices $\mathbb{Z}^2$ and $\mathbb{Z}^3$ respectively leads naturally to the investigation of FLIS in general simple graphs, either finite or infinite.

*This document is an extended abstract of the paper [1] available on arXiv.

2 Fully leafed induced subtrees

We introduce the decision problem called *Leafed Induced Subtree problem (*LIS) and its associated optimization problem MLIS:

LIS. Given a simple graph G and two positive integers i and ℓ, does there exist an induced subtree of G with i vertices and ℓ leaves?

MLIS. Given a simple graph G on n vertices, what is the maximum number of leaves, $L_G(i)$, that can be realized by an induced subtree of G with i vertices, for $i \in \{0, 1, \ldots, n\}$?

We believe that induced subtrees with the maximal number of leaves are interesting candidates for the representation of structures appearing in nature and in particular in molecular networks. Indeed, in chemical graph theory, subtrees are known to be useful in the computation of the *Wiener index* of chemical graph, that corresponds to a topological index of a molecule [10]. The results of [2] and [10] suggest that a thorough investigation of subtrees, and in particular induced subtrees with many leaves, could lead to the discovery of combinatorial structures relevant to chemical graph theory.

Definition 1 (Leaf function) *For a graph $G = (V, E)$, the* leaf function *of G, denoted by L_G, is the function with domain $\{0, 1, 2, \ldots, \text{size}(G)\}$ which associates to i the maximum number of leaves that can have an induced subtree of size i of G. As is customary, we set $\max \emptyset = -\infty$. An induced subtree T of G with i vertices is called* fully leafed *when its number of leaves is exactly $L_G(i)$.*

The following observations are immediate. Consider a graph G with at least 3 vertices. The sequence $(L_G(i))_{i=0,1,\ldots,|G|}$ is non-decreasing if and only if G is a tree. Moreover, we have $L_G(0) = 0 = L_G(1)$ and $L_G(2) = 2$ if G contains at least one edge. If G is connected and non-isomorphic to a complete graph, then $L_G(3) = 2$.

While it is easy to determine the leaf function for some well-known families of graphs (such as complete graphs, wheels etc.), in general the problem is much harder.

3 Complexity and algorithms

First, we prove that the problem LIS is NP-complete by reducing it from the *Independent Set* problem. To tackle the MLIS problem and compute the leaf function, we provide a non trivial branch and bound algorithm. The algorithm is based on a data structure that we call an *induced subtree configuration.*

Definition 2 *Let $G = (V, E)$ be a simple graph and $\Gamma = \{\text{green}, \text{yellow}, \text{red}, \text{blue}\}$ be a set of colors with coloring functions $c : V \to \Gamma$. An* induced subtree configuration *of G is an ordered pair $C = (c, H)$, where c is a coloring and H is a stack of colorings called the* history *of C.*

All colorings $c : V \to \Gamma$ must satisfy the following conditions for any $u, v \in V$:

(i) The subgraph induced by $c^{-1}(\text{green})$ is a tree;

(ii) If $c(u) = \text{green}$ and $\{u, v\} \in E$, then $c(v) \in \{\text{green}, \text{yellow}, \text{red}\}$;

(iii) If $c(u) = \text{yellow}$, then $|c^{-1}(\text{green}) \cap N(u)| = 1$, where $N(u)$ denotes the set of neighbors of u.

The initial induced subtree configuration *of a graph G is the pair (c_{blue}, H) where $c_{\text{blue}}(v) = \text{blue}$ for all $v \in G$ and H is the empty stack. When the context is clear, C is simply called a* configuration.

Roughly speaking, a configuration is an induced subtree enriched with information that allows one to generate other induced subtrees either by extension, by exclusion or by backtracking. The colors assigned to the vertices can be interpreted as follow. The *green* vertices are the confirmed vertices to be included in a subtree. Since each *yellow* vertex is connected to exactly one *green* vertex, any *yellow* vertex can be safely added to the *green* subtree to create a new induced subtree. The *red* vertices are those that are excluded from any possible tree extension. The exclusion of a *red* vertex is done either because it is adjacent to more than one *green* vertex and its addition would create a cycle or because it is explicitly excluded for generation purposes. Finally, the *blue* vertices are available vertices that have not been considered yet and that could be considered later. It is convenient to save in the stack H the colorations from which C was obtained.

Contrary to a naive algorithm that considers all induced subtrees to compute the maximal number of leaves, the strategy prunes the search space by discarding induced subtrees that cannot be extended to fully leafed subtrees (see Figure 1). Therefore, given an induced subtree configuration of n *green* vertices, we define a function $C.\textsc{LeafPotential}(n')$, for $n \leq n' \leq |V|$, which computes an upper bound on the number of leaves that can be reached by extending the current configuration C to a configuration of n' vertices. To compute this upper bound we consider an optimistic scenario in which all available *yellow* and *blue* vertices that are close enough can be safely colored in *green* without creating a cycle, whatever the order in which they are selected.

Proposition 1 *Let C be any configuration of a simple graph $G = (V, E)$ with $n \geq 3$* green *vertices and let n' be an integer such that $n \leq n' \leq |V|$. Then any extension of C to a configuration of n' vertices has at most $C.\textsc{LeafPotential}(n')$ leaves, where $C.\textsc{LeafPotential}(n')$ is the operator described above.*

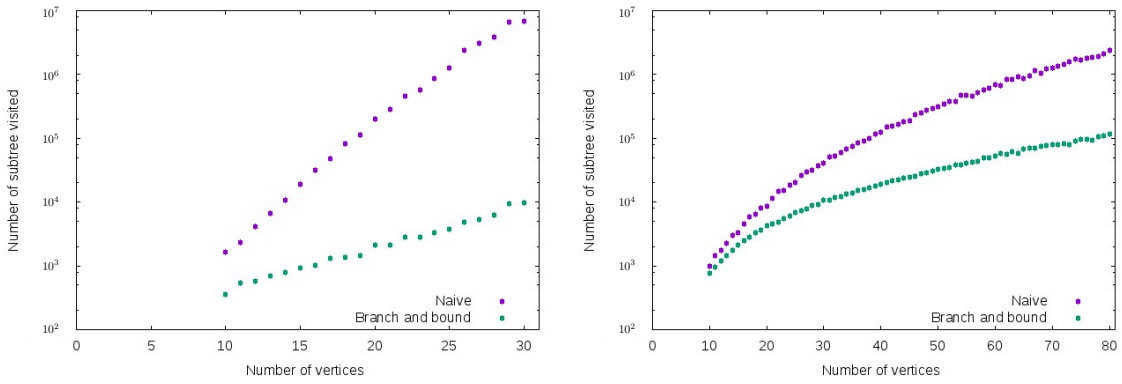

FIG. 1: Number of induced subtrees visited for samples of 10 random graphs with density 0.3 (on the left) and with density 0.8 (on the right).

When restricted to the case of trees, we show that the MLIS problem is polynomial using a dynamic programming strategy.

Theorem 1 *Let $T = (V, E)$ be a tree with $n \geq 2$ vertices. Then L_T can be computed in $\mathcal{O}(n^3\Delta)$ time and $\mathcal{O}(n^2)$ space where Δ denotes the maximal degree of a vertex in T.*

Notice that a naive greedy approach cannot work, even in the case of trees, because a fully leafed induced subtree with n vertices is not necessarily a subtree of a fully leafed induced subtree with $n + 1$ vertices. Both algorithms are available, with examples, in a public GitHub repository[1].

[1] `https://github.com/enadeau/fully-leafed-induced-subtrees`

4 Perspectives

There seems to be some room for improving and specializing the branch and bound algorithm. For example, we are able to speed up the computations for the hypercube Q_6 by taking into account some symmetries, but significant improvements could be done by discarding more configurations by exploiting the complete automorphism group of the hypercube. It seems reasonable to expect similar speed up for other highly symmetric graphs.

From a theoretical perspective, it is not clear if the algorithm described for the trees is optimal. As a last observation, we believe that it would be interesting to investigate the natural problems of counting and generating related to the concept of fully leafed induced subtrees.

References

[1] Alexandre Blondin Massé, Julien de Carufel, Alain Goupil, Mélodie Lapointe, Émile Nadeau, and Élise Vandomme. Fully leafed induced subtrees. `https://arxiv.org/abs/1709.09808` *arXiv preprint*.

[2] Alexandre Blondin Massé, Julien de Carufel, Alain Goupil, and Maxime Samson. Fully leafed tree-like polyominoes and polycubes. In *Combinatorial Algorithms*, LNCS 28th International Workshop, IWOCA 2017, New-Castle, Australia, Springer. To appear.

[3] Azzedine Boukerche, Xuzhen Cheng, and Joseph Linus. A performance evaluation of a novel energy-aware data-centric routing algorithm in wireless sensor networks. *Wireless Networks*, 11(5):619–635, 2005.

[4] Si Chen, Ivana Ljubiċ, and Subramanian Raghavan. The generalized regenerator location problem. *INFORMS Journal on Computing*, 27(2):204–220, 2015.

[5] Akshay Deepak, David Fernández-Baca, Srikanta Tirthapura, Michael J. Sanderson, and Michelle M. Evominer: frequent subtree mining in phylogenetic databases. *Knowledge and Information Systems*, 41(3):559–590, 2014.

[6] Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Heidelberg, fourth edition, 2010.

[7] Paul Erdős, Michael Saks, and Vera T. Sós. Maximum induced trees in graphs. *J. Combin. Theory Ser. B*, 41(1):61–79, 1986.

[8] Michael R. Garey and David S. Johnson. *Computers and intractability*. W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[9] Charles Payan, Maurice Tchuente, and Nguyen Huy Xuong. Arbres avec un nombre maximum de sommets pendants. *Discrete Math.*, 49(3):267–273, 1984.

[10] Lászlò A. Székely and Hua Wang. On subtrees of trees. *Advances in Applied Mathematics*, 34(1):138–155, 2005.

[11] Kunihiro Wasa, Hiroki Arimura, and Takeaki Uno. Efficient enumeration of induced subtrees in a K-degenerate graph. In *Algorithms and computation*, LNCS 8889, 94–102. Springer, Cham, 2014.

[12] Mohammed J. Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, 71–80, New York, NY, USA, 2002. ACM.