Discrete Applied Mathematics 236 (2018) 42–56

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

A faster parameterized algorithm for Pseudoforest Deletion[☆]

Hans L. Bodlaender[a,b,*], Hirotaka Ono[c], Yota Otachi[d]

[a] Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, Netherlands
[b] Department of Mathematics and Computer Science, University of Technology Eindhoven, P.O. Box 513, 5600 MB Eindhoven, Netherlands
[c] Graduate School of Informatics, Nagoya University, Furo-cho, Chikusa-ku, Nagoya, 464-8601, Japan
[d] Faculty of Advanced Science and Technology, Kumamoto University, 2-39-1 Kurokami, Chuo-ku, Kumamoto, 860-8555, Japan

A R T I C L E I N F O

Article history:
Received 17 January 2017
Received in revised form 14 August 2017
Accepted 12 October 2017
Available online 6 December 2017

Keywords:
Graph algorithms
Fixed parameter tractability
Pseudoforest
Feedback vertex set
Treewidth

A B S T R A C T

A pseudoforest is a graph where each connected component contains at most one cycle, or alternatively, a graph that can be turned into a forest by removing at most one edge from each connected component. In this paper, we show that the following problem can be solved in $O(3^k nk^{O(1)})$ time: given a graph G and an integer k, can we delete at most k vertices from G such that we obtain a pseudoforest? The result improves upon an earlier result by Philip et al. (2015) who gave a (nonlinear) $7.56^k n^{O(1)}$-time algorithm both in the exponential factor depending on k as well as in the polynomial factor depending on n.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

In this paper, we consider the Pseudoforest Deletion problem. A pseudoforest is an undirected graph that is obtained from a forest by adding at most one edge to each connected component. In the Pseudoforest Deletion problem, we are given a graph $G = (V, E)$ and an integer k, and ask if there is a set of at most k vertices in G, that, when deleted from G, turns G into a pseudoforest.

The Pseudoforest Deletion problem derives its interest by its relation to the well studied Feedback vertex set problem, where we want to delete at most k vertices from a graph so that the graph becomes a forest. Allowing one more edge per connected component of the graph after removing the deletion set changes the problem significantly, and, somewhat surprisingly, seems to make it simpler in the parameterized setting, as the running time of our algorithm is smaller than that of the best known parameterized algorithms for Feedback Vertex Set.

The Pseudoforest Deletion problem was first studied by Philip et al. [14], together with the generalization where each connected component is a tree plus at most ℓ edges. They showed that for each ℓ, the problem to delete at most k vertices such that we obtain such an ℓ-*pseudoforest* has a kernel with $O(k^2)$ vertices, with the constant factor growing with ℓ. For the Pseudoforest Deletion problem, i.e., the case that $\ell = 1$, they give a deterministic algorithm with running time $7.56^k n^{O(1)}$.[1]

[☆] This research was partially supported by the Networks project, funded by the Dutch Ministry of Education, Culture and Science through NWO and by MEXT/JSPS KAKENHI grant numbers 24106004, 24220003, 25730003, 26540005. The third author was partially supported by FY 2015 Researcher Exchange Program between JSPS and NSERC. An extended abstract of this paper appeared in the proceedings of IPEC 2016 (Bodlaender, 2016) [7].

* Corresponding author at: Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, Netherlands.
E-mail addresses: h.l.bodlaender@uu.nl (H.L. Bodlaender), ono@nagoya-u.jp (H. Ono), otachi@cs.kumamoto-u.ac.jp (Y. Otachi).

[1] They did not specify the exact dependency in n, which is at least quadratic.

https://doi.org/10.1016/j.dam.2017.10.018
0166-218X/© 2017 Elsevier B.V. All rights reserved.

In this paper, we improve upon the latter result, both with respect to the exponential factor in k, as well as in the polynomial factor in n, which is, in our case, linear.

It is easy to see that the PSEUDOFOREST DELETION problem belongs to the class of problems studied by Fomin et al. [10], and thus, by these results, the problem has a constant-factor polynomial-time approximation algorithm, a polynomial kernel (improved to quadratic by the results of Philip et al. [14]), and a randomized algorithm that runs in time $O(c^k n)$ for some constant c. The randomized algorithm is a generalization of an algorithm by Becker et al. [2] for the FEEDBACK VERTEX SET problem and a related problem called the LOOP CUTSET problem. Fomin et al. [10] consider a large class of problems that includes PSEUDOFOREST DELETION; they show that each problem in this class has a deterministic algorithm that runs in time $O(2^{O(k)} n \log^2 n)$, and a constant-factor approximation algorithm that runs in time $O(nm)$. If one looks closely at the randomized algorithm by Becker et al. [2] and the generalization by Fomin et al. [10], it follows that one can solve the PSEUDOFOREST DELETION problem with a randomized algorithm in $O(4^k nk^{O(1)})$ time.

Our improvement on these two algorithms is based upon the combination of a few different insights and techniques, in particular:

- Positive instances, i.e., graphs that can be turned into a pseudoforest by deleting at most k vertices have treewidth at most $k + 2$.
- The notion of *pseudoforest* has the following *local characterization*: a graph is a pseudoforest if and only if it has an edge orientation such that each vertex has outdegree at most one.
- The local characterization allows us to solve the problem with dynamic programming on a tree decomposition in time that is linear in the number of vertices and single exponential in the treewidth, without the need to use advanced techniques like the cut and count method [9] or the rank based approach [6]. With help of *convolutions* [17] (see also [4]), the running time of the dynamic programming algorithm is reduced to $O(3^t nt^{O(1)})$ on tree decompositions of width t.
- What remains is the need to find an initial tree decomposition to run the dynamic programming algorithm on. For this, we use a modification of the $O(f(t)n)$ algorithm for TREEWIDTH by Bodlaender [5]. The modification includes the use of *iterative compression* inside one of the subroutines.

It is interesting to contrast our result with the currently best known parameterized algorithms for FEEDBACK VERTEX SET: for the PSEUDOFOREST DELETION problem we have a deterministic $O(3^k nk^{O(1)})$ algorithm, while FEEDBACK VERTEX SET can be solved in $O(3^k n^{O(1)})$ time with a randomized algorithm [9] and $O(3.63^k n^{O(1)})$ time with a deterministic algorithm [12]; in both cases, the running time is not linear in n.

This paper is organized as follows. In Section 2, we give some preliminary definitions. Section 3 contains a few graph theoretic observations. The main algorithm is given in Section 4. It uses as a subroutine a dynamic algorithm that solves the PSEUDOFOREST DELETION problem when a tree decomposition of bounded width is available. The details of this subroutine are given in the Appendix. In Section 5, we discuss a corollary of our result and an ILP formulation of the problem. Some conclusions are given in Section 6.

2. Preliminaries

When not specified otherwise, a graph $G = (V, E)$ is considered to be undirected, but possibly with self-loops and parallel edges. Allowing self-loops and parallel edges makes the description of the main algorithm easier. An *orientation* of a graph $G = (V, E)$ is a directed graph obtained by giving each edge in G a direction. For a graph $G = (V, E)$ and vertex set $W \subseteq V$, the *subgraph of* G *induced by* W is denoted by $G[W] = (W, \{e \in E \mid \text{ both endpoints of } e \text{ belong to } W\})$. The subgraph of $G = (V, E)$ induced by all vertices except those in W is denoted by $G \setminus W$, i.e., $G \setminus W = G[V \setminus W]$.

A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$ with T a tree, and $\{X_i \mid i \in I\}$ a collection of subsets (called *bags*) of V, such that

1. $\bigcup_{i \in I} X_i = V$;
2. for all $\{v, w\} \in E$, there is an $i \in I$ with $\{v, w\} \subseteq X_i$;
3. for all $v \in V$, the set of nodes $\{i \in I \mid v \in X_i\}$ forms a connected subtree of T.

The *width* of a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G.

For the definition above, if there are parallel edges or self-loops, we can just ignore them, i.e., a tree decomposition of a graph with parallel edges and self-loops is a tree decomposition of the associated simple graph (obtained by keeping only one of each set of parallel edges and removing all self-loops).

In this paper, we also use the related notion of *nice* tree decomposition. In the literature, there are a few variants of this notion that differ in details. In this case, we use the variant with *edge introduce nodes* and leaf bags of size one.

A *nice tree decomposition* is a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ where T is a *rooted* tree, and nodes are of one of the following five different types. With each bag/node in the tree decomposition, we also associate a subgraph of G; the subgraph associated with node i is denoted $G_i = (V_i, E_i)$. We give each type together with how the corresponding subgraph is formed.

Download English Version:

https://daneshyari.com/en/article/6871506

Download Persian Version:

https://daneshyari.com/article/6871506

Daneshyari.com