Theoretical Computer Science 535 (2014) 25–30

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

An improved parameterized algorithm for the independent feedback vertex set problem

Yinglei Song [1]

School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang, Jiangsu 212003, China

A R T I C L E   I N F O

Article history:
Received 2 August 2013
Received in revised form 14 March 2014
Accepted 27 March 2014
Communicated by V.Th. Paschos

Keywords:
Independent feedback vertex set
Parameterized algorithm
Dynamic programming

A B S T R A C T

In this paper, we develop a new parameterized algorithm for the Independent Feedback Vertex Set problem. Given a graph $G = (V, E)$, the goal of the problem is to determine whether there exists a vertex subset $F \subseteq V$ such that $V - F$ induces a forest in G and F is an independent set. We show that there exists a parameterized algorithm that can determine whether a graph contains an IFVS of size k or not in time $O(4^k n^2)$. To our best knowledge, this result improves the known upper bound for this problem, which is $O(5^k n^{O(1)})$.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

A *feedback vertex set* (FVS) in a graph $G = (V, E)$ is a vertex subset $F \subseteq V$ such that vertices in $V - F$ induce a forest in G. Feedback vertex sets have important applications in deadlock recovery in the development of operating systems [23] and database management systems [17]. Moreover, F is an *independent feedback vertex set* (IFVS) if it induces an independent set in G. In contrast to the fact that a graph always has an FVS, a graph (such as a clique) may not always have an IFVS.

In the past two decades, the FVS problem has been intensively studied. The goal of the problem is to determine whether a graph contains an FVS of size k or not. In [14], it is shown that the problem is NP-complete. So far, a few exact algorithms have been developed to compute a minimum FVS in a graph. For example, in [21], an algorithm that needs $O(1.9053^n)$ time was developed to find a minimum FVS in a graph, this algorithm is also the first one that breaks the trivial $O(2^n n)$ bound. In [9], an elegant algorithm that can enumerate all induced forests in a graph is used to find the minimum FVS. This algorithm needs $O(1.7548^n)$ computation time. Recently, the upper bound of this problem is improved to $O(1.7347^n)$ [10].

In practice, efficient solutions exist for some NP-hard problems when some or all of the parameters in these problems are small integers. A well known example is the Vertex Cover problem [7,18]. Given a parameter k and a graph $G = (V, E)$, a simple parameterized algorithm can decide whether G contains a vertex cover of size at most k in time $O(2^k|V|)$ [7,18]. This algorithm is efficient and can be used in practice when the parameter k is a fixed small integer.

An NP-hard problem is *fixed parameter tractable* if it can be solved in time $O(g(p_1, p_2, \dots, p_s)n^c)$, where n is the size of the problem, $p_1, p_2, \dots, p_s$ are the parameters in the problem, g is a function of $p_1, p_2, \dots, p_s$ and c is a constant that does not depend on any of the parameters. In the past decades, a large number of NP-hard problems have been shown to be fixed parameter tractable [7,18]. However, not all NP-hard problems are known to be fixed parameter tractable, the

E-mail address: yingleisong@gmail.com.

[1] The majority of this work is done when the author was working in the Department of Mathematics and Computer Science at University of Maryland Eastern Shore, USA. This work is supported in part by the University Research Fund of Jiangsu University of Science and Technology, under the Grant Number 635301202.

http://dx.doi.org/10.1016/j.tcs.2014.03.031
0304-3975/© 2014 Elsevier B.V. All rights reserved.

Independent Set problem is one of them [8]. Many of these problems are believed to be parameterized intractable and a number of complexity classes have been identified to describe their parameterized complexity [8].

The parameterized FVS problem uses the size of an FVS as the parameter k. In [1,5–8,11–13,19,20], it is shown that the problem is fixed parameter tractable and a few algorithms that can solve the problem in time $O(2^{O(k)}n^{O(1)})$ are developed. In [3], it is shown that the problem can be solved in time $O(5^k kn^2)$. In [2], the upper bound of this problem is further improved to $O(3.83^k kn^2)$. Recently, a faster algorithm that needs $O^*(3.592^k n^2)$ time is developed in [15]. This algorithm is currently the best algorithm known for the FVS problem. In [4], a new technique is developed for connectivity problems parameterized by tree width and it is shown that the FVS problem can be solved by a randomized algorithm in time $O^*(3^k n^2)$.

Although the FVS problem has attracted a lot of attention from researchers in computer science, little effort has been made toward the IFVS problem. Recently, it is shown that the FVS problem can be reduced to the IFVS problem in polynomial time while preserving the parameter k [16]. The reduction is simply subdividing each graph edge once. The IFVS problem is thus more general than the FVS problem.

In [16], an algorithm that needs $O(5^k n^{O(1)})$ time is developed for the IFVS problem. The algorithm starts by applying two reduction rules to remove all vertices of degree at most one from the graph and then uses the algorithm developed in [2] to obtain an FVS of size k. The algorithm returns “no” if no such FVS exists. Based on such an FVS, a branching approach is used to analyze the graph and determine whether it contains an IFVS of size k or not. It is also shown in [16] that the IFVS problem has a kernel of size $O(k^3)$.

In this paper, we develop a new parameterized algorithm that can determine whether a graph contains an IFVS of size k or not in time $O(4^k n^2)$. Our algorithm follows the *iterative compression* approach developed in [22]. This technique has been used to develop parameterized algorithms for many problems [5,6,12]. Specifically, given a graph $G = (V, E)$, where $V = \{v_1, v_2, v_3, \dots, v_n\}$, the approach starts with an empty graph and adds vertices one by one in the order of $v_1, v_2, v_3, \dots, v_n$ to reconstruct G. When v_i ($1 \leqslant i \leqslant n$) is added, edges are also added to join v_i with its neighbors in $\{v_1, v_2, \dots, v_{i-1}\}$. We denote the graph induced on vertices $v_1, v_2, \dots, v_i$ in G by G_i. In other words, $G_i = G[\{v_1, v_2, \dots, v_i\}]$. It is clear that a series of graph $G_0, G_1, \dots, G_n$ can be constructed as the vertices are added to the graph, where G_0 is an empty graph and $G_n = G$. The approach starts with G_k since G_k trivially contains an IFVS of size k, each G_i ($k+1 \leqslant i \leqslant n$) is processed to determine whether it contains an IFVS of size k or not. If the answer is “yes”, the approach finds such a set F_i in G_i and continues to process G_{i+1}, otherwise the approach outputs “no” since G does not contain an IFVS of size k if one of its subgraph does not contain such an IFVS. The approach outputs “yes” if eventually G is found to contain an IFVS of size k. A detailed description of the approach can be found in [22] and some other related work such as [5,6,12].

It is not difficult to see that, the key part of the above approach is to develop an algorithm that can determine whether G_{i+1} contains an IFVS of size k or not, given an FVS L_{i+1} of size $k+1$ in G_{i+1}, where $L_{i+1} = F_i \cup \{v_{i+1}\}$. The major contribution of this paper is the development of such an algorithm. We need to emphasize here that L_{i+1} may not be an IFVS of G_{i+1}. However, we show later that our algorithm only needs the fact that it is an FVS of G_{i+1}. Other than using a branching approach, the algorithm uses a dynamic programming based approach to accomplish the task in time $O(4^k n)$. Since G contains n vertices in total, the total amount of computation time needed by the approach is thus at most $O(4^k n^2)$. Based on the simple parameter preserving polynomial time reduction that reduces the FVS problem to IFVS problem, we also immediately obtain a parameterized algorithm that solves the FVS problem in time $O(4^k n^2)$.

2. Notations and preliminaries

The graphs in this paper are undirected graphs without loops. For a given graph $G = (V, E)$ and a vertex subset $V' \subseteq V$, $G[V']$ is the subgraph induced on the vertices in V'. A vertex u is a *neighbor* of V' if it is joined to one of the vertices in V' by an edge in G'. We use $N_G[V']$ to denote the set of neighbors of V' in G. We may omit the subscript G when the underlying graph is clear in the context. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we use $G_1 \cup G_2$ to denote the graph $(V_1 \cup V_2, E_1 \cup E_2)$. Furthermore, to simplify the notation, we use $G - V'$ to denote subgraph $G[V - V']$. A vertex subset V' is a *feedback vertex set* in G if $G - V'$ is a forest. The objective of our algorithm is to find a feedback vertex set that contains the minimum number of vertices and also induces an independent set.

3. The algorithm

Our algorithm is based on the following lemma.

Lemma 3.1. *Given a graph $G = (V, E)$ and an FVS F in G, a minimum IFVS in G can be computed in time $O(4^{|F|}|V|)$.*

Proof. Since F is an FVS, $G[V - F]$ is a forest. As the first step of the algorithm, we arbitrarily choose a vertex in each tree in $G[V - F]$ as the root of the tree. We color each vertex in $V - F$ with black. We then check the number of children of each vertex in the tree and find those that have more than two children. For each such vertex u, we remove the edges that join it with its children $c_1, c_2, \dots, c_d$, where d is the number of children in the tree. We then create a path of white vertices $w_1, w_2, \dots, w_{d-1}$. w_{d-1} is joined to u with an edge. To construct a binary tree, w_1 is joined to c_1 and c_2 with two edges