Theoretical Computer Science 573 (2015) 63–70

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Edge deletion problems: Branching facilitated by modular decomposition ☆

Yunlong Liu [a,b], Jianxin Wang [a], Jie You [a], Jianer Chen [a,c], Yixin Cao [d,*]

[a] School of Information Science and Engineering, Central South University, Changsha, China
[b] College of Mathematics and Computer Science, Key Laboratory of High Performance Computing and Stochastic Information Processing (Ministry of Education of China), Hunan Normal University, Changsha, China
[c] Department of Computer Science and Engineering, Texas A&M University, College Station, TX, United States
[d] Department of Computing, Hong Kong Polytechnic University, Hong Kong, China

ARTICLE INFO

Article history:
Received 18 August 2014
Received in revised form 22 January 2015
Accepted 30 January 2015
Available online 7 February 2015
Communicated by P. Widmayer

Keywords:
Graph modification problems
Edge deletion problems
Chain graphs
Trivially perfect graphs
Modular decomposition
Branching

ABSTRACT

Edge deletion problems ask for a minimum set of edges whose deletion makes a graph have a certain property. When this property can be characterized by a finite set of forbidden induced subgraphs, the problem can be solved in fixed-parameter time by a naive bounded search tree algorithm. Sometimes deleting an edge to break an erstwhile forbidden induced subgraph might introduce new ones, which may involve the neighbors of the original forbidden induced subgraph. Therefore, in considering possible ways to break a forbidden induced subgraph one naturally takes its neighborhood into consideration. This observation easily yields more efficient branching rules, but a naive implementation will require too many tedious case analyses. Here we take advantage of modular decomposition, which allows us to focus on far simpler quotient graphs instead of the original graphs. They together yield simple improved algorithms for the edge deletion problems to chain graphs and trivially perfect graphs.

1. Introduction

Edge deletion problems ask for a minimum set of edges whose deletion makes a graph have a certain property, i.e., to belong to some specific graph class. As an important sub-collection of graph modification problems, they have wide applications, and most of them are known to be NP-complete [27,13,23,20]. Cai [3] observed that if the desired graph class can be characterized by a finite number of forbidden induced subgraphs, then it can be solved in time $c^k \cdot n^{O(1)}$ by a straightforward bounded search tree algorithm—as usual, n and m denote the numbers of vertices and edges respectively in the input graph, and k is the number of edges to be deleted. Recall that a graph problem, with a nonnegative parameter k, is *fixed-parameter tractable (FPT)* if there is an algorithm solving it in time $f(k) \cdot (n+m)^{O(1)}$, where f is a computable function depending only on k [7]. Therefore, edge deletion problems to chain graphs ($\{K_3, 2K_2, C_5\}$-free) and trivially perfect graphs ($\{P_4, C_4\}$-free) are FPT.

☆ Supported in part by the National Natural Science Foundation of China under Grants 61070224, 61232001, and 61420106009, and the China Postdoctoral Science Foundation under Grant 2012M521551.

* Corresponding author. Tel.: +852 3400 3195.
E-mail address: yixin.cao@polyu.edu.hk (Y. Cao).

http://dx.doi.org/10.1016/j.tcs.2015.01.049
0304-3975/

These graph classes can be recognized in linear time, and if a given graph is not a chain graph or not a trivially perfect graph, the algorithms of Heggernes et al. [16] can find in linear time a minimal forbidden induced subgraph. Using their algorithms, the aforementioned bounded search tree algorithm can be implemented in time $O(c^k \cdot (n+m))$. As far as the polynomial factor is concerned, this is surely optimum, and what remains is to decrease the constant c. This decrement, if possible, would be more meaningful when the polynomial factor on graph size remains linear—in literature, it is not uncommon that the improvement of the exponential factor is achieved at a cost of increasing the polynomial factor. For the trivially perfect edge deletion problem, Nastos and Gao [22] managed to decrease the constant c from 4 to 2.45, which we now further decrease to 2.42. Our main results include also the first nontrivial algorithm for the chain edge deletion problem.

Theorem 1.1. *Problems chain edge deletion and trivially perfect edge deletion can be solved in time* $O(2.57^k \cdot (n+m))$ *and* $O(2.42^k \cdot (n+m))$ *respectively.*

Related work. Yannakakis [27] initiated the study of edge deletion problems by establishing NP-completeness of several of them. His results, as well as some earlier ones, were collected by Garey and Johnson in their magnum opus [13]. The general observation on *easy* graph classes (those having characterization of finite forbidden subgraphs) is due to Cai [3]. Bounded search tree algorithms using refined branching rules have been considered by Nastos and Gao [21,22] and a subset of the authors [19]. Modular decomposition has been the major technical tool, used in a far extensive way, in solving the edge deletion problem to interval graphs [4].

More often than not, the study of edge deletion problems is associated with their edge addition variations (also known as completion problems) [18,8]. In particular, the edge deletion problem to a graph class is polynomially equivalent to the completion problem to its complement graph class (containing the complement of all graphs in the original graph class). It is worth mentioning that some graph classes, e.g., threshold graphs ($\{2K_2, P_4, C_4\}$-free), are self-complementary, on which the edge deletion problem and the completion problem are equivalent. Also of intensive interest is the edge editing problems, where the modifications can be both edge additions and deletions, while the most studied is probably the cluster graphs (P_3-free) [5].

An $O(k^2)$-vertex kernel for the chain edge deletion problem was reported by Bessy and Perez [2].[1] Very recently, Drange and Pilipczuk [9] announced an $O(k^7)$-vertex kernel for the trivially perfect edge deletion problem.

Our technique. The deletion of an edge to break an erstwhile forbidden induced subgraph might introduce new one(s). For example, deleting any edge from a C_5 introduces a $2K_2$ (chain edge deletion), and deleting any edge from a C_4 introduces a P_4 (trivially perfect edge deletion). Therefore, in either case, we need to delete at least two edges from the subgraph, which implies a branching rule far better than the trivial one. A more common situation is when the newly introduced forbidden subgraph is *not* induced by the same set of vertices. Therefore, instead of branching on a forbidden induced subgraph, one may want to take also its neighbors into consideration. We remark that a similar observation has been used by Aravind et al. [1] in designing kernelization algorithms for edge deletion problems.

Our main technical idea appears in the way we carry out this simple strategy. The observation above is straightforward and easily yields more efficient branching rules, but a naive implementation of it requires us to consider dozens, if not hundreds, of different cases, quickly going beyond the capability of manual verification. To avoid such tedious case analyses, we take advantage of modular decomposition, which allows us to focus on far simpler quotient graphs instead of the original graphs. Together with the symmetry of forbidden induced subgraphs that concern us, this substantially reduces the chore in verifying the correctness of our branching rules. Since trivially perfect graphs are a subclass of cographs, the application of modular decomposition to them is natural; it is interesting that modular decomposition helps for the chain graphs as well.

2. Preliminaries

All graphs discussed in this paper are undirected and simple. A graph G is given by its vertex set $V(G)$ and edge set $E(G)$, whose cardinalities will be denoted by n and m respectively. For a vertex $v \in V(G)$, let $N_G(v)$ denote its neighbors and $N_G[v] := N(v) \cup \{v\}$. This is extended to a subset $X \subseteq V(G)$ by $N_G[X] := \bigcup_{v \in X} N[v]$ and $N_G(X) := N[X] \setminus X$. These subscripts are omitted when there is no ambiguities. For a subset $X \subseteq V(G)$, denote by $G[X]$ the subgraph induced by X. For a subset $E_- \subseteq E(G)$ of edges, we use $G - E_-$ to denote the subgraph $(V(G), E(G) \setminus E_-)$. When E_- is a singleton set consisting of only edge e, we write $G - e$ instead. The *complement graph* of a graph G is defined to be $\overline{G} = \left(V(G), \binom{V(G)}{2} \setminus E(G)\right)$, i.e., it has the same vertex set as G, and for each pair of distinct vertices $u, v \in V(G)$, there is an edge uv in $E(\overline{G})$ if and only if $uv \notin E(G)$.

[1] Since in earlier literature, "chain completion" and "chain [edge] deletion" have been used to refer to the special cases where input graphs are limited to bipartite graphs, to avoid ambiguity, Bessy and Perez [2] chose to use "bipartite chain deletion" for the problem concerning us. It seems that their kind consideration did not pay off, so we decide to abide by the terminology that is now standard, i.e., without an explicit modifier, the input graph to the CHAIN EDGE DELETION problem can be any graph.