ELSEVIER

Information Processing Letters 58 (1996) 171-176

Information Processing Letters

Fixed-parameter tractability of graph modification problems for hereditary properties [1]

Leizhen Cai [2]

Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong

Received 3 February 1995; revised 29 March 1996
Communicated by M.J. Atallah

Abstract

This paper is concerned with the fixed-parameter tractability of the problem of deciding whether a graph can be made into a graph with a specified hereditary property by deleting at most i vertices, at most j edges, and adding at most k edges, where i, j, k are fixed integers. It is shown that this problem is fixed-parameter tractable whenever the hereditary property can be characterized by a finite set of forbidden induced subgraphs. Furthermore, the problem of deciding whether a graph can be made into a chordal graph by adding a fixed number k of edges is shown to be solvable in $O(4^k(k+1)^{-3/2}(m+n))$ time, and is thus fixed-parameter tractable.

Keywords: Design of algorithms; Graph algorithms; Fixed-parameter tractability; Graph modification problems

1. Introduction

All graphs in this paper are finite undirected simple graphs. A *graph property* is a set Π of graphs, and any graph in Π is a *Π-graph*. If for any graph $G \in \Pi$ every induced subgraph of G is also a Π-graph, then Π is a *hereditary property*. A property Π has a *forbidden set characterization* if there is a set F of graphs such that a graph is a Π-graph iff it does not contain any graph in F as an induced subgraph, and it has a *finite forbidden set characterization* if F is a finite set. For a property Π, the general *Π-graph modification problem* is to modify a graph G into a Π-graph by adding and deleting edges and vertices. For such a graph modification problem, one is primarily concerned with the number of edges and vertices involved in the modification process. Typical examples of the Π-graph modification problem include the following well-studied optimization problems in the literature [8,10,11,15]:

- The *edge deletion problem*, or equivalently, the *maximum spanning subgraph problem*: Find a set of edges of minimum cardinality in G whose deletion results in a Π-graph.
- The *vertex deletion problem*, or equivalently, the *maximum induced subgraph problem*: Find a set of vertices of minimum cardinality in G whose deletion results in a Π-graph.
- The *edge and vertex deletion problem*, or equivalently, the *maximum partial subgraph problem*: Find

[1] This work was partially supported by a Direct Grant for Research from the Chinese University of Hong Kong and an Earmarked Research Grant from the Research Grants Council of Hong Kong.

[2] Email: lcai@cs.cuhk.edu.hk.

PII S0020-0190(96)00050-6

a set of edges and vertices of minimum cardinality in G whose deletion results in a Π-graph.

- The *edge addition problem*, or equivalently, the *minimum spanning supergraph problem* (also called *graph augmentation problem* or *graph completion problem* in the literature): Find a set of new edges of minimum cardinality whose addition to G results in a Π-graph.

As one might imagine, the above problems are NP-hard in general. In fact, the first three problems are NP-hard for any nontrivial hereditary property Π [8,10,11,15].

In this paper, we consider a fixed-parameter version of the Π-graph modification problem for hereditary properties, namely, the following *$\Pi(i,j,k)$-graph modification problem* for fixed parameter (i,j,k), where E^c is the set of edges in the complement of G:

> Given a graph $G = (V,E)$, find subsets $V' \subseteq V$, $E' \subseteq E$ and $E^* \subseteq E^c$ with $|V'| \leqslant i$, $|E'| \leqslant j$ and $|E^*| \leqslant k$ such that the graph $G - V' - E' + E^*$ is a Π-graph.

When such subsets exist, G is called a *$\Pi(i,j,k)$-graph*, and the set $V' \cup E' \cup E^*$ is referred to as a *modifier* of G. Note that for any hereditary property Π, a non-Π-graph cannot be made into a Π-graph by vertex addition and thus there is no need to consider vertex addition for the Π-graph modification problem when Π is hereditary.

The above fixed-parameter problem is trivially polynomial-time solvable by the exhaustive search method for any polynomial-time recognizable property Π. On the other hand, the problem would be NP-hard for any nontrivial hereditary property Π if the parameter (i,j,k) were a part of the input. In this paper we will use a framework developed by Downey and Fellows [4–6] for dealing with fixed-parameter problems to refine the complexity status of the $\Pi(i,j,k)$-graph modification problem. An algorithm for a fixed-parameter problem is *uniformly polynomial* if it runs in time $O(n^\alpha)$ for every fixed parameter value, where α is a constant independent of the problem parameter and n is the size of the input. Note that the constant factor in the big-O notation can be an exponential function (or even worse) of the parameter. A fixed-parameter problem is *fixed-parameter tractable* if it admits a uniformly polynomial algorithm. This notion of fixed-parameter tractability attempts to distinguish tractable and intractable fixed-parameter problems, which is very akin to the notion of polynomial algorithms in distinguishing tractable and intractable problems. Readers interested in the general background of the theory of fixed-parameter complexity are referred to [4–6] for details.

In this paper, we show that the $\Pi(i,j,k)$-graph modification problem is fixed-parameter tractable for any hereditary property Π that has a finite forbidden set characterization. Furthermore, we prove that the problem of adding a fixed number k of new edges to a graph to make the graph a *chordal graph* (a graph in which any cycle of length greater than three contains a chord, i.e., an edge joining two non-consecutive vertices in the cycle) is also fixed-parameter tractable. Note that this problem is equivalent to the $\Pi(0,0,k)$-graph modification problem with Π being the set of chordal graphs. However, in this case, Π does not have a finite forbidden set characterization.

Throughout the paper, we use $V(G)$ and $E(G)$, respectively, to denote the vertex and edge sets of a graph G, and use m and n, respectively, to denote the number of edges and the number of vertices of the input graph.

2. Properties with finite forbidden set characterizations

We first notice that a property Π is hereditary iff it has a forbidden set characterization. To verify this, we only need to observe that any property that has a forbidden set characterization is clearly hereditary, and that for any hereditary property Π, the induced subgraph relation "$\trianglelefteq$" defines a partial order among all graphs not in Π and the set of minimal elements in this poset forms the forbidden set F of Π. Most interesting graph properties are hereditary and thus have forbidden set characterizations. Furthermore, many hereditary properties, such as line graphs [1], cographs [3], and split graphs [7], admit finite forbidden set characterizations.

Let Π be a property that has a finite forbidden set characterization. Let ν be the maximum number of vertices among all graphs in its forbidden set F. Then, given a graph G on n vertices, one can easily determine whether G is a Π-graph in $O(n^\nu)$ time by the ex-

haustive search method. Therefore for such a property Π, the $\Pi(i,j,k)$-graph modification problem can be solved in $O(n^{i+2j+2k+\nu})$ time by considering all possible ways to delete $\leqslant i$ vertices and $\leqslant j$ edges, and add $\leqslant k$ edges; and for each resulting graph checking whether the graph is a Π-graph. Of course, such an algorithm is neither uniformly polynomial nor efficient in practice. To obtain a uniformly polynomial-time algorithm, we use the finite set characterization of Π. The idea is to find a minimal forbidden induced subgraph H of Π in G, and then destroy it by deleting edges and vertices, and adding edges. For this method to work, we need to find a minimal forbidden induced subgraph of Π in G in uniformly polynomial time. Indeed, this can be done!

Theorem 1. *For any hereditary property Π, if Π is recognizable in time $T(m,n)$, then for any graph G that is not a Π-graph, a minimal forbidden induced subgraph of Π in G can be found in $O(nT(m,n))$ time.*

Proof. Let A be a recognition algorithm for Π that runs in time $T(m,n)$. Then a call $A(G)$ of A on G returns "true" if G is a Π-graph; otherwise it returns "false". We now give an algorithm that uses A as an oracle to find a minimal forbidden induced subgraph of Π in G. The idea is to check, for each vertex v of G, whether $G-v$ is a Π-graph. If it is then v belongs to a minimal forbidden induced subgraph, else $G-v$ must contain a minimal forbidden induced subgraph and thus we proceed to consider $G-v$ in the same manner. The following pseudocode describes the algorithm, where F is used to collect vertices of a minimal forbidden induced subgraph:

$F := \emptyset$;
$V := V(G)$;
while $V \neq \emptyset$ **do**
  Arbitrarily choose a vertex $v \in V$;
  $V := V - \{v\}$;
  if $A(G-v)$
  then $F := F \cup \{v\}$
  else $G := G - v$;
end while

Clearly, the above algorithm terminates after n iterations of the "while" loop. Let F' denote the set F after the termination of the above procedure. We claim that $G' = G[F']$ is a minimal forbidden induced subgraph of Π in G. Let v' be the last vertex deleted from G, and G^* be the graph just before the deletion of vertex v'. Then $G' = G^* - v'$. Since v' is deleted from G^*, G' is not a Π-graph. It remains to be shown that G' is minimal with respect to forbidden induced subgraphs, i.e., for any $u \in F'$, $G' - u$ is a Π-graph. Suppose to the contrary that there is a vertex $x \in F'$ such that $G' - x$ is not a Π-graph. Let G'' be the graph when vertex x is under consideration. Then G' is an induced subgraph of G'', and thus $G'' - x$ is not a Π-graph since $G' - x$ is an induced subgraph of $G'' - x$ and Π is hereditary. This implies that x would have been deleted from G'', i.e., $x \notin F'$, a contradiction. Therefore G' is a minimal forbidden induced subgraph of Π in G, and the algorithm is correct.

For the time complexity of the algorithm, we note that the oracle A is called n times, each taking at most $T(m,n)$ time. It is then clear that the algorithm takes $O(nT(m,n))$ time. □

With the above theorem in hand, we can now prove the following result:

Theorem 2. *The $\Pi(i,j,k)$-graph modification problem is fixed-parameter tractable for any hereditary property Π that admits a finite forbidden set characterization.*

Proof. To solve the $\Pi(i,j,k)$-graph modification problem, we repeat the following two steps until we either get a Π-graph or have used up the deletion of $\leqslant i$ vertices and $\leqslant j$ edges, and the addition of $\leqslant k$ edges. In the former case, G is a $\Pi(i,j,k)$-graph; otherwise it is not.

Step 1. Find a minimal forbidden induced subgraph H of Π in G.

Step 2. Modify G by either deleting an edge or a vertex from H, or adding an edge to H.

The correctness of the algorithm is obvious. When G is a $\Pi(i,j,k)$-graph, a modifier of G can be easily obtained by comparing the resulting graph with the original graph. We now estimate the complexity of the algorithm. As noted earlier, Π is recognizable in $O(n^\nu)$ time. Therefore, a minimal forbidden induced subgraph of Π in G can be found in $O(n^{\nu+1})$ time (by Theorem 1). Since ν is the maximum number of vertices in any minimal forbidden induced subgraph of Π,

there are at most $\binom{\nu}{2}$ different ways to add an edge to or delete an edge from H, and at most ν different ways to delete a vertex from H. This implies that the total number of graphs generated in the above procedure is at most $\binom{\nu}{2}^{j+k}\nu^i = O(\nu^{i+2j+2k})$. Therefore the overall running time of the algorithm is $O(\nu^{i+2j+2k}n^{\nu+1})$, and thus the algorithm is uniformly polynomial since ν, i, j and k are constants independent of m and n. □

3. Chordalization with k edges

Recall that a chordal graph is a graph in which any cycle of length greater than three contains a chord, i.e., an edge joining two non-consecutive vertices in the cycle. The *chordalization problem* (also known as the *minimum fill-in problem* and *chordal graph completion problem*) asks for a minimum number of edges whose addition to a given graph makes the graph a chordal graph. This problem corresponds to the problem of minimizing "fill-in" when applying Gaussian elimination to symmetric matrices (see [12]), and is known to be NP-hard [16]. Here, we consider the following fixed-parameter version of the chordalization problem, *the k-chordalization problem*:

> Given a graph G, find a set E^* of at most k new edges, where k is a fixed integer, so that the graph $G + E^*$ is a chordal graph.

When such a set E^* exists, the resulting graph $G + E^*$ is called a *k-chordalization* of G.

Let us refer to any chordless cycle on more than three vertices as a *hole*. Then from the definition of a chordal graph, it is clear that a graph is chordal iff it does not contain any hole as an induced subgraph. So if we take Π to be the set of all chordal graphs, then Π is clearly a hereditary property and the k-chordalization problem is exactly the $\Pi(0,0,k)$-graph modification problem. However, the general result in the previous section does not apply since Π's forbidden set $F = \{H \mid H \text{ is a hole}\}$ is not a finite set in this case. Nevertheless, we will show in this section that the k-chordalization problem is fixed-parameter tractable; in fact, the problem is solvable in linear time.

From the definition of a chordal graph, we see that to k-chordalize a graph G it suffices to destroy all holes in G by adding at most k edges. For a hole H with h vertices, there could be $\Omega(h^{2k})$ possible ways to destroy H by adding at most k edges. Therefore if we destroy holes by trying all possible ways for each hole, the running time could be $\Omega(n^{2k}(m+n))$ in the worst case, which is not uniformly polynomial. In order to obtain a uniformly polynomial-time algorithm, we explore the structure of chordal graphs.

First we fix some definitions. A spanning cycle of a graph is a *hamiltonian cycle*, and any graph that admits a hamiltonian cycle is a *hamiltonian graph*. A graph is an *outerplanar graph* if it can be embedded in the plane so that no two edges cross each other and all vertices lie on the same face (we usually choose this face to be the exterior face), and it is *maximal outerplanar* if no edge can be added without losing outerplanarity. For simplicity, we refer to a maximal outerplanar graph as a *mop*. Note that a *mop* is a chordal graph. The following property of a hamiltonian chordal graph will be useful in designing a uniformly polynomial-time algorithm.

Lemma 3. *Let G be a hamiltonian chordal graph. Then for any hamiltonian cycle C of G, there is a spanning mop of G that contains C.*

Proof. We use induction on the number of vertices of G. The only hamiltonian chordal graph with at most three vertices is a triangle and the theorem is clearly true in this case. Assume that the theorem is true for all hamiltonian chordal graphs with fewer than n vertices and let G be a hamiltonian chordal graph on $n \geqslant 4$ vertices.

Without loss of generality, we may assume that $C = v_0v_1 \ldots v_{n-1}v_0$ is a hamiltonian cycle of G. Since G is chordal and $n \geqslant 4$, there is a chord v_iv_j in G. Let $G_1 = G[\{v_i, v_{i+1}, \ldots, v_j\}]$ and $G_2 = G[\{v_j, v_{j+1}, \ldots, v_i\}]$ (indices are taken module n). Then $C_1 = v_iv_{i+1} \ldots v_jv_i$ and $C_2 = v_jv_{j+1} \ldots v_iv_j$ are hamiltonian cycles of G_1 and G_2 respectively. Therefore both G_1 and G_2 are hamiltonian chordal graphs. By the induction hypothesis, there is a spanning *mop* M_1 of G_1 that contains C_1, and a spanning *mop* M_2 of G_2 that contains C_2. Each of M_1 and M_2 admits an outerplanar embedding with its vertices lying on the exterior face. Let $M = M_1 \cup M_2$. Then we can merge the above two outerplanar embeddings of M_1 and M_2 by identifying edge v_iv_j to get an outerplanar embedding of M since v_iv_j is the only edge shared by M_1 and M_2. Therefore M is a spanning *mop* of G that contains C. □

```
Chordalization(G, k);
Input: A graph G and a positive integer k;
Output: A chordal graph G' such that G' is a
  supergraph of G and |E(G')| − |E(G)| ⩽ k
  if such a G' exists; otherwise return "No";
begin
if G is a chordal graph
then return G
else
  begin
  Find a hole H in G;
  if k < |V(H)| − 3
  then return "No"
  else for every mop H' on V(H) do
    G' := G + (E(H') − E(H));
    Chordalization(G', k − (|V(H)| − 3))
  endfor
  endif
endif
end.
```

Fig. 1.

Lemma 3 can be used to reduce the k-chordalization problem to a k'-chordalization problem for some $k' < k$.

Lemma 4. *Let H be a hole of a non-chordal graph G. Then G admits a k-chordalization iff there exists a mop H' on $V(H)$ such that the graph $G' = G + (E(H') - E(H))$ admits a $(k - (|V(H)| - 3))$-chordalization.*

Proof. Suppose that G^* is a $(k - (|V(H)| - 3))$-chordalization of G'. Then $|E(H')| - |E(H)| = |V(H)| - 3$ since H' is a *mop* on $V(H)$ and any *mop* on n vertices contains $2n - 3$ edges. Therefore G' has exactly $|V(H)| - 3$ more edges than G, implying that G^* is a k-chordalization of G.

Conversely, suppose that $\tilde{G}$ is a k-chordalization of G. Let $\tilde{H} = \tilde{G}[V(H)]$. Then $\tilde{H}$ is a hamiltonian chordal graph and, by Lemma 3, contains a *mop* H' on $V(H)$. Therefore H' contains $2|V(H)| - 3$ edges, and thus $|E(H')| - |E(H)| = |V(H)| - 3$. Let $G' = G + (E(H') - E(H))$. Then G' is a subgraph of $\tilde{G}$, and $|E(G')| - |E(G)| = |E(H')| - |E(H)| = |V(H)| - 3$, which implies that $\tilde{G}$ is a $(k - (|V(H)| - 3))$-chordalization of G'. □

In light of Lemma 4, we can k-chordalize a graph G by repeatedly finding a hole H in the graph and destroying it by adding edges to form all possible *mops* on $V(H)$ until we either get a chordal graph or have used up k edges. A pseudocode of the algorithm is given in Fig. 1.

The correctness of the algorithm follows from Lemma 4. Furthermore, a modifier of G can be obtained by subtracting the edges of G from the edges of its k-chordalization constructed by the algorithm.

To show that the algorithm is uniformly polynomial, we first estimate the total number of graphs generated in the algorithm. It is well known that the number of distinct *mops* on a chordless cycle of h vertices equals the number of distinct triangulations of a convex polygon on h vertices, which equals the $(h-2)$th Catalan number

$$C_{h-2} = \frac{1}{h-1}\binom{2(h-2)}{h-2}$$

(see, for example, [2, Chapter 16.4]). Let M_k denote the maximum number of graphs generated by the algorithm in k-chordalizing G. Let h denote the number of vertices in the hole H. Without loss of generality, we may assume $k \geqslant h - 3$. Then by Lemma 4,

$$\begin{aligned} M_k &= M_{k-(h-3)} \times \#\text{ distinct } \textit{mops} \text{ on } V(H) \\ &= M_{k-(h-3)} C_{h-2}. \end{aligned}$$

We will bound M_k by the $(k+1)$th Catalan number. To do so, we need the following property of Catalan numbers:

Lemma 5. *For any integers $i, j \geqslant 0$, $C_{i+1}C_{j+1} \leqslant C_{i+j+1}$.*

Proof. Fix i and use induction on j. The inequality clearly holds for $j = 0$. Assume it holds for j. Then

$$\begin{aligned} C_{i+(j+1)+1} &= \frac{2(i+j)+4}{i+j+3} \cdot \frac{2(i+j)+3}{i+j+2} C_{i+j+1} \\ &\qquad \left(\text{since } C_{n+1} = \frac{2n+2}{n+2} \cdot \frac{2n+1}{n+1} C_n\right) \\ &\geqslant \frac{2j+4}{j+3} \cdot \frac{2j+3}{j+2} C_{i+j+1} \\ &\geqslant C_{i+1}\left(\frac{2j+4}{j+3} \cdot \frac{2j+3}{j+2} C_{j+1}\right) \\ &\qquad \text{(by the induction hypothesis)} \\ &= C_{i+1}C_{j+2}. \end{aligned}$$

This establishes the inequality. □

To show $M_k \leqslant C_{k+1}$, we use induction on k. The claim is true for $k = 0$ since $M_0 = 1$ and $C_1 = 1$. Assume that the claim holds for k, then

$$\begin{aligned} M_{k+1} &= M_{(k+1)-(h-3)}C_{h-2} \\ &\leqslant C_{k-h+5}C_{h-2} \\ &\qquad \text{(by the induction hypothesis)} \\ &\leqslant C_{(k-h+4)+(h-3)+1} \\ &\qquad \text{(by Lemma 5)} \\ &= C_{k+2} \end{aligned}$$

Therefore the claim is true.

We now estimate the complexity of the algorithm. First of all, a chordal graph can be recognized in linear time [12] and furthermore a hole in a non-chordal graph can be found in linear time [14]. As discussed earlier, the algorithm generates at most C_{k+1} different graphs. These graphs correspond to triangulations of convex polygons. It is well known that there is a one-to-one correspondence between a triangulation of a convex polygon and a full binary tree (see, for example, cite[Chapter 16.4]cormen). Therefore these graphs can be generated in $O(C_{k+1})$ time since full binary trees can be generated in time proportional to the number of distinct full binary trees [13]. Therefore the algorithm runs in $O(C_{k+1}(m+n))$ time. Since

$$C_k = \frac{4^k}{\sqrt{\pi}k^{3/2}}\left(1 + O\left(\frac{1}{n}\right)\right)$$

(see [2, Ex. 13-4, p. 262]), we obtain the following theorem, which implies that the k-chordalization problem is fixed-parameter tractable:

Theorem 6. *A k-chordalization of a graph, if it exists, can be found in*

$$O\left(\frac{4^k}{(k+1)^{3/2}}(m+n)\right)$$

time.

Remark. After the submission of the paper, it was brought to the author's attention by Michael R. Fellows that Kaplan, Shamir and Tarjan [9] have also independently proved the fixed-parameter tractability of the k-chordalization problem by using a similar method. However, they only obtained $O(2^{4k}(m+n))$ as the time complexity of their algorithm.

References

[1] L.W. Beineke, On derived graphs and digraphs, in: H. Sachs, H.J. Voss and H. Walther, eds., *Beiträge zur Graphentheorie* (Teubner, Leipzig, 1968) 17–33.

[2] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms* (MIT Press, Cambridge, MA, 1990).

[3] D.G. Corneil, H. Lerchs and L.S. Burlingham, Complement reducible graphs, *Discrete Appl. Math.* **3** (1981) 163–174.

[4] R.G. Downey and M.R. Fellows, Parameterized complexity, Monograph in preparation.

[5] R.G. Downey and M.R. Fellows, Fixed-parameter intractability, in: *Proc. 7th Structure in Complexity Theory Conf.* (1992) 36–49.

[6] R.G. Downey and M.R. Fellows, Fixed parameter tractability and completeness, in: *Complexity Theory: Current Research* (Cambridge University Press, 1993) 36–49.

[7] S. Födes and P.L. Hammer, Split graphs, in: *Proc. 8th Southeastern Conf. on Combinatorics, Graph Theory and Computing*, Baton Rouge, LA (1977) 36–49.

[8] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness* (Freeman, San Francisco, 1979).

[9] H. Kaplan, R. Shamir and R.E. Tarjan, Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping (extended abstract), in: *Proc. 35th Ann. Symp. on Foundations of Computer Science*, Santa Fe, NM (1994) 36–49.

[10] M.S. Krishnamoorthy and D. Narsingh, Node-deletion NP-complete problems, *SIAM J. Comput.* **8** (1979) 619–625.

[11] J.M. Lewis, On the complexity of the maximum subgraph problem, in: *Proc. 10th ACM Symp. on Theory of Computing*, New York (1978) 265–274.

[12] D. Rose, R. Tarjan and G. Lueker, Algorithmic aspects of vertex elimination on graphs, *SIAM J. Comput.* **5** (1976) 266–283.

[13] F. Ruskey and A. Proskurowski, Generating binary trees by transpositions, *J. Algorithms* **11** (1990) 266–283.

[14] R. Tarjan and M. Yannakakis, Addendum: Simple linear-time algorithms to test chordality of graphs, text acyclicity of hypergraphs and selectively reduce acyclic hypergraphs, *SIAM J. Comput.* **14** (1985) 254–255.

[15] M. Yannakakis, Node- and edge-deletion NP-complete problems, in: *Proc. 10th ACM Symp. on Theory of Computing*, New York (1978) 253–264.

[16] M. Yannakakis, Computing the minimum fill-in is NP-complete, *SIAM J. Algebraic Discrete Methods* **2** (1981) 77–79.