REPORTS IN INFORMATICS

ISSN 0333-3590

Linear-time certifying algorithms for recognizing trivially perfect graphs

Pinar Heggernes Dieter Kratsch

REPORT NO 339 December 2006

Department of Informatics
UNIVERSITY OF BERGEN
Bergen, Norway

This report has URL http://www.ii.uib.no/publikasjoner/texrap/pdf/2006-339.pdf

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at http://www.ii.uib.no/publikasjoner/texrap/.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

Linear-time certifying algorithms for recognizing trivially perfect graphs

Pinar Heggernes* Dieter Kratsch†

Abstract

We give the first linear-time certifying algorithms to recognize trivially perfect graphs, with sublinear certificates for negative output. In case of membership in the class of trivially perfect graphs, our algorithms provide as certificate a structure for the input graph that characterizes the class, and in case of non-membership they provide as certificate a forbidden induced subgraph of the class. The certificates of membership can be authenticated in time $O(n+m)$ and the certificates of non-membership can be authenticated in time $O(n)$.

Keywords: Algorithms and data structures, certifying graph algorithms, recognition of graph classes.

1 Introduction

The study of certifying algorithms is motivated by software engineering, software reliability and the insight that software is often not bug-free. Although an algorithm has been proven correct, its implementation may contain bugs. Thus it is desirable to have tools for knowing whether the output of an implementation of an algorithm is correct or returned due to a bug. The fundamental idea of certifying algorithms is that with its output the algorithm supplies a certificate. Then the output and the certificate are authenticated by a separate algorithm. This authentication algorithm takes the input, the output, and the certificate returned by the original algorithm, and verifies (independently of the original algorithm) whether the output is correct. Bug-free implementation of the authentication algorithm is crucial, and thus authentication should be simple. Certifying algorithms are highly desirable in practice to reduce the risk of erroneous answers caused by bugs in the implementation [14, 15]. For general discussions on result checking see also [18] and [15, section 2.14].

A *certifying algorithm* for a decision problem is an algorithm that provides a *certificate* with each answer. A certificate is an evidence that can be used to authenticate the correctness of the answer. An *authentication algorithm* is an algorithm that checks the validity of the certificate. A certificate is *sublinear* if its authentication algorithm has a tighter running time than a linear one; and a certificate is *weak* if it takes the same time to authenticate as it does to solve the original problem without the certificate [14].

A familiar example is a linear-time certifying algorithm to recognize bipartite graphs, computing a 2-coloring for bipartite input graphs and an odd cycle for non-bipartite input graphs. The authentication algorithm for the membership certificate checks in time $O(n+m)$ the validity of the 2-coloring by cycling through all edges, and thus the certificate is weak. The authentication algorithm for the non-membership certificate checks in time $O(n)$ that it is a cycle, it has odd length, and that the claimed edges occur in the input graph, and thus the certificate is sublinear.

*Department of Informatics, University of Bergen, N-5020 Bergen, Norway. Email: pinar.heggernes@ii.uib.no

†Laboratoire d'Informatique Théorique et Appliquée, Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France. Email: kratsch@univ-metz.fr

For certifying graph recognition algorithms, certificates of non-membership that are forbidden subgraphs of the recognized class are especially desirable. For example, parameterized algorithms for completing an arbitrary graph into a chordal graph [2] or an interval graph [10] by adding the minimum number of edges need to identify forbidden subgraphs and get rid of these recursively as long as the graph at hand is not in the class. For the recognition of probe split graphs [3] a forbidden subgraph is needed when the graph at hand is chordal and non-split.

A linear-time planarity test is part of the LEDA system [15, section 8.7]. It computes a planar embedding for planar input graphs and a subdivision of K_5 or $K_{3,3}$ for non-planar input graphs. Other graph classes having linear-time certifying recognition algorithms are chordal graphs [17], cographs [5], interval and permutation graphs [14], proper interval graphs [11, 16], proper interval bigraphs [11], proper circular-arc graphs, and unit circular-arc graphs [13].

Previously, we gave linear-time certifying algorithms for split, threshold, bipartite chain, and cobipartite chain graphs [9]. Here, we present two linear-time certifying algorithms to recognize trivially perfect graphs, one that provides a cotree and one that provides a so-called universal-in-a-component ordering as certificate of membership, and both of them provide as a certificate of non-membership a vertex subset inducing a P_4 or C_4 in the input graph. Thus, for both algorithms, the certificate of membership can be authenticated in time $O(n+m)$, and the certificate of non-membership can be authenticated in time $O(n)$.

2 Preliminaries

All graphs in this paper are simple and undirected. For a graph $G = (V, E)$, we let $n = |V|$ and $m = |E|$. An edge between vertices u and v is denoted by uv. If u and v are not adjacent we call uv a *non-edge*. The set of *neighbors* of a vertex $v \in V$ is the set of all vertices adjacent to v, denoted by $N(v)$. The *degree* of a vertex v is $d(v) = |N(v)|$. A *clique* is a set of vertices that are all pairwise adjacent, and an *independent set* is a set of vertices that are all pairwise non-adjacent. A vertex v is called *simplicial* if $N(v)$ is a clique. A vertex v satisfying $N(v) \cup \{v\} = V$ is called *universal*, and a vertex with no neighbors is called *isolated*. The subgraph of G induced by a vertex set $A \subseteq V$ is denoted by $G[A]$. All subgraphs in this text are induced subgraphs. The *complement* $\overline{G}$ of G is a graph that has the vertices of G as its vertex set and the non-edges of G as its edge set.

Let α be an ordering $(v_1, v_2, ..., v_n)$ of V. If α is such that $d(v_1) \geq d(v_2) \geq ... \geq (v_n)$, it is called a *non-increasing degree ordering*. If α has the property that v_i is universal in a connected component of $G[\{v_i, v_{i+1}, ..., v_n\}]$, for $1 \leq i \leq n$, it is called a *universal-in-a-component ordering (uco)*.

A cycle on k vertices is denoted by C_k, and a path on k vertices is denoted by P_k.

A graph recognition algorithm is called *certifying* if it can produce a certificate such that the membership or the non-membership of the graph in the class can be checked using this certificate. The following observation is used to establish authentication algorithms for certificates of non-membership.

Observation 2.1 *Let a vertex subset $A \subset V$ of constant size be a certificate of non-membership of a certifying recognition algorithm for graph class $\mathcal{G}$ on input graph $G = (V, E)$, where additionally for each vertex of A a pointer to a vertex of a graph H (indicating an isomorphism from $G[A]$ to H) is part of the certificate. Then there is an $O(n)$ time algorithm to authenticate whether $G[A]$ is isomorphic to the graph H.*

Proof. The set A can be provided by the certifying algorithm in a characteristic vector of size n or as a list of size $|A|$ containing pointers to the vertex list of G. Every vertex in A has at most $n-1$ neighbors and non-neighbors, so $G[A]$ can be computed in time $O(|A| \cdot n)$, which is $O(n)$ since $|A|$ is not dependent on the size of G. The pointers can be used to authenticate in time $O(|A|^2)$ that $G[A]$ is isomorphic to H. ∎

A graph G is a *trivially perfect graph* if for each induced subgraph H of G, the number of maximal cliques of H is equal to the maximum size of an independent set of H [6]. The following is a characterization of trivially perfect graphs through their forbidden subgraphs.

Theorem 2.2 [6] *A graph is trivially perfect if and only if it contains no vertex subset that induces P_4 or C_4.*

The class of *chordal graphs* is the class of graphs containing no induced cycle of length longer than 3. A *cograph* is a graph without a vertex subset that induces a P_4. Hence trivially perfect graphs form a subclass of cographs. In fact, it follows immediately that trivially perfect graphs exactly chordal cographs [1]. Since both chordal graphs and cographs have linear time certifying algorithms [17, 5, 8], obtaining a forbidden induced subgraph as a certificate of non-membership can be done by using the previous algorithms. However, the challenge is to give a certificate of membership that can be checked in $O(n+m)$ time. We will here give two linear-time certifying algorithms for trivially perfect graphs that both output as certificates of non-membership a vertex subset that induces a P_4 or a C_4. The first algorithm outputs as certificate of membership a universal-in-a-component ordering, and the second outputs a restricted cotree (a model for cographs).

3 Vertex orderings as certificates of membership

Each connected trivially perfect graph G has a universal vertex [19, 20], and consequently each connected induced subgraph of a trivially perfect graph has a universal vertex [7]. Hence for a trivially perfect graph G, we can find an ordering of its vertices $\alpha = (v_1, v_2, ..., v_n)$ such that v_i is universal in the connected component of $G[\{v_i, v_{i+1}, ..., v_n\}]$ that v_i belongs to, for $1 \leq i \leq n$. Thus trivially perfect graphs have universal-in-a-component orderings (uco). We first prove that such orderings characterize trivially perfect graphs.

Theorem 3.1 *A graph is trivially perfect if and only if it has a uco.*

Proof. If a graph is trivially perfect, it has a uco by the results mentioned above. For the opposite direction, let G be a graph that has a uco α and assume for a contradiction that G is not trivially perfect. Thus G has a set of vertices $\{w, x, y, z\}$ that induces a $C4$ or a $P4$. Let x be the vertex among these four that has the earliest occurrence in α. Thus $\{w, x, y, z\}$ is a connected subgraph in the remaining graph when all vertices of α prior to x are removed from G. Then x should be universal in a connected induced subgraph of G that contains $\{w, x, y, z\}$, but this is not possible since there is no vertex in a $C4$ or a $P4$ that is adjacent to all three other vertices. Thus α is not a uco. ∎

Lemma 3.2 *A graph is trivially perfect if and only if every non-increasing degree ordering is a universal-in-a-connected-component ordering.*

Proof. Let $G = (V, E)$ be a graph, and let $\alpha = (v_1, v_2, ..., v_n)$ be a non-increasing degree ordering of G, such that $d(v_1) \geq d(v_2) \geq ... \geq (v_n)$. If α is a uco, then by Theorem 3.1 G is trivially perfect. In the other direction, assume that G is trivially perfect, and assume for a contradiction that v_i is not universal in the connected component of $G[\{v_i, v_{i+1}, ...v_n\}]$ that it belongs to, for some i. Thus there is a vertex v_j with $j > i$ such that $v_i v_j \notin E$, and there is a path from v_i to v_j containing only vertices from $\{v_i, v_{i+1}, ...v_n\}$. Observe that there must be a vertex v_k on this path that is adjacent to both v_i and v_j because otherwise we have a P_4 with endpoints v_i and v_j. Since v_k is adjacent to v_j, and v_i is not adjacent to v_j, and the degree of v_i is at least the degree of v_k, v_i must have a neighbor x in G that is not adjacent to v_k (x might belong to $\{v_{i+1}, ...v_n\}$ or not). But, if x is adjacent to v_j, then $\{x, v_i, v_k, v_j\}$ induces a C_4, and if x is not adjacent to v_j, this set induces a P_4, contradicting that G is trivially perfect. ∎

Before we can conclude with a linear-time certifying recognition algorithm for trivially perfect graphs, we also have to show the following.

Lemma 3.3 *Given a graph G and an ordering α of the vertices of G, it can be checked in $O(n+m)$ time whether α is a uco of G.*

Proof. We process the vertices of G one by one in the order given by α. At the beginning all vertices are labeled with 0. At each step i, we check whether v_i and all its neighbors have the same label. If so, we change the labels of all neighbors of v_i to i, and delete v_i from the graph. If, at some step, there are several labels among the neighbors of v_i or if their labels are different than the label of v_i, then let k be the largest such label. This means that vertex v_k was not a universal vertex in the connected subgraph that it belonged to in the subgraph $G[\{v_k, v_{k+1}, ..., v_n\}]$, and thus α is not a uco. We claim that if we manage to iterate through all vertices without discovering such an anomaly, then α is a uco. To see this, observe that if α is not a uco then a vertex v_i must exist so that v_i does not label all vertices in the connected component that it belongs to in $G[\{v_i, ..., v_n\}]$ by i. Thus in this subgraph there is a vertex that is labeled by i that has a neighbor that is labeled with something smaller than i. Either the labels of these vertices remain different until the turn comes to one of them and then we detect the anomaly, or some other vertex changes both their labels to become the same. But since that other vertex cannot have the same label as both of them, again the anomaly will be detected. The running time is clearly $O(n+m)$ since we only check the neighborhood of each vertex once. ∎

Now we are ready to give the main result on trivially perfect graphs.

Theorem 3.4 *There is a linear-time certifying algorithm for recognizing trivially perfect graphs that outputs certificates which can be authenticated in $O(n+m)$ time for membership and in $O(n)$ time for non-membership.*

Proof. Given the input graph $G = (V, E)$, we start by computing a non-increasing degree order in $O(n+m)$ time. Then we check as described in the proof of Lemma 3.3 whether this order is a uco. If it is, we output YES and this ordering as a certificate of membership. If it is not, then whenever we detect an anomaly, the largest label k among the labels of neighbors of v_i and the label of v_i itself, gives a vertex v_k that was not universal in the connected component of $G[\{v_k, v_{k+1}, ..., v_n\}]$. Thus we can find one vertex in this component that is not adjacent to v_k, and this will give us the desired vertex set that induces a P_4 or a C_4 as described in the proof of Lemma 3.2, in $O(n+m)$ time. Then we output NO and this vertex set as a certificate of non-membership. The certificate of membership can be authenticated in $O(n+m)$ time by Lemma 3.3. The certificate for non-membership can be authenticated in $O(n)$ time by Observation 2.1. ∎

4 Cotrees as certificates of membership

We now give an alternative certifying algorithm for recognizing trivially perfect graphs that exploits structural and algorithmic properties of cographs. If G is a cograph then either G is disconnected, or its complement $\overline{G}$ is disconnected, or G consists of a single vertex. Using the corresponding decomposition rules, one obtains the modular decomposition tree of a cograph which is called a cotree. A *cotree* T of a cograph G is a rooted tree with two types of interior nodes: 0-nodes and 1-nodes. The vertices of G are assigned to the leaves of T in a one-to-one manner. Finally two vertices u and v are adjacent in G if and only if the lowest common ancestor of the leaves u and v in T is a 1-node [4]. The cotree of a cograph is uniquely determined. Note that due to the above mentioned decomposition rules, we may assume that each interior node of a cotree has at least two children, that no 0-node is a child of a 0-node, and that no 1-node is a child of a 1-node.

Combining Theorem 2.2 with the definition of cotrees, one establishes the following result.

Lemma 4.1 *A cograph G is a trivially perfect graph if and only if, in the cotree T of G, every 1-node has at most one child that is a 0-node.*

Proof. Let a be a 1-node of T with two 0-node children, say b and c. Then each of the subtrees rooted at b and rooted at c contains at least two leaves. These four leaves induce a C_4 in G. Thus G is not trivially perfect.

Assume that a cograph G is not trivially perfect. Since it is a cograph it does not contain P_4 as induced subgraph. By Theorem 2.2 there is a vertex subset $\{w, x, y, z\}$ that induces a C_4 in G. Let a be the root of the smallest possible rooted subtree of T containing the leaves assigned to w, x, y, z. Since $G[\{w, x, y, z\}]$ is connected a is a 1-node. If a has two 0-nodes among its children the proof is complete. Thus we may assume that a has precisely one 0-node as a child, say b. By the definition of cotrees, all other children of a are leaves of T. No vertex of $\{w, x, y, z\}$ could be assigned to a child of a, since each of these vertices is non-adjacent to one vertex of $\{w, x, y, z\}$. Thus all four leaves of T corresponding to these vertices are contained in the subtree rooted at b. This contradicts the choice of a. Consequently, a has at least two 0-node children. ∎

Thus a certifying algorithm for recognizing trivially perfect graphs can output such a cotree as a certificate of membership. However, then we have to argue that there is an $O(n + m)$ time authentication algorithm for checking this certificate, which also involves checking that the cotree output as certificate is indeed the cotree of G. Typically, certifying cograph recognition algorithms output a cotree if the input is a cograph, and a vertex subset that induces a P_4 if the input is not a cograph [5]. Despite the many cograph recognition algorithms we could not find an $O(n+m)$ time algorithm for checking whether a given cotree is the cotree of a given graph. Therefore we present here a cotree authentication algorithm, which is also the main motivation for giving an additional certifying algorithm for trivially perfect graphs.

Lemma 4.2 *Given a graph $G = (V, E)$ and a tree T, it can be checked in $O(n + m)$ time whether T is the cotree of G.*

Proof. It is easy to check that T is indeed a cotree, i.e. has 0-nodes, 1-nodes and leaves to which the vertices of G are assigned. To check whether T is indeed the cotree of G we verify the following property which is an immediate consequence of the definition of a cotree in a bottom-up fashion.

- If u and v are leaves and have the same 1-node of the cotree T as their parent then $N[u] = N[v]$, and thus $\{u, v\} \in E$.
- If u and v are leaves and have the same 0-node of the cotree T as their parent then $N(u) = N(v)$, and thus $\{u, v\} \notin E$.

Now let us provide the authentication algorithm.

1. For each vertex v, add v to the adjacency lists it belongs to. Then sort all adjacency lists of G such that vertices appear in the same order as that of the leaves in the cotree T.
2. In a bottom-up fashion check the interior nodes of T thereby updating the graph and the cotree. The current cotree is denoted by T', the current graph is denoted by $G' = G[V']$, where V' is the set of vertices not yet processed. Consequently, $N'[v] = N_{G'}[v] = N[v] \cap V'$ and $N'(v) = N_{G'}(v) = N(v) \cap V'$. Clearly T can only be accepted after a successful check of its root. We start with $G' := G$, $V' := V$ and $T' := T$. Interior nodes have to be checked as follows.

   (R0) Let a be a 0-node of T' all of whose children are leaves, and the corresponding vertices of G' are $u_1, u_2, \ldots, u_t$. Now compare the adjacency lists $N'[u_1], N'[u_2], \ldots, N'[u_t]$.

Conclude that T is not the cotree of G and terminate, if any list contains more than one vertex of $\{u_1, u_2, \ldots, u_t\}$, or if the lists $N'(u_1), N'(u_2), \ldots, N'(u_t)$ are not all equal. The comparison can be done in a standard fashion by passing simultaneously through all lists in time proportional to the total length of the inspected lists. If the algorithm has not terminated, conclude that T is the cotree of G if a is the root of T. Otherwise, remove the vertices $u_2, \ldots, u_t$ from G', the corresponding leaves from T', and all adjacency lists of G'. Remove the 0-node a from T' and put the leaf u_1 at its place.

(R1) Let b be a 1-node whose children are all leaves in T', and the corresponding vertices of G' are $v_1, v_2, \ldots, v_s$. Now compare the adjacency lists $N'[v_1], N'[v_2], \ldots, N'[v_s]$. in a standard fashion by passing simultaneously through all lists. If the lists are not all equal, conclude that T is not the cotree of G and terminate. Otherwise, if b is the root of T then conclude that T is the cotree of G and stop. If b is not the root of T, remove all vertices $v_2, \ldots, v_s$ from the graph G', all adjacency lists of G', and the corresponding leaves from the cotree T', remove the 1-node b from T' and put the leaf v_1 at its place.

Correctness follows immediately from the above mentioned properties.

To establish running time of $O(n + m)$, the deletion of vertices need to be supported. One approach would be to add pointers between the two occurrences of an edge, i.e. from x_i in $N[x_j]$ to x_j in $N[x_i]$ when sorting the lists. Then an easy way to see that the running time is indeed linear is by assigning the cost for passing $N'[u_1]$ respectively $N'[v_1]$ to one of the deleted vertices, say u_2 respectively v_2. Consequently each adjacency list is passed (amortized) at most three times: comparison, deletion and possibly the cost of some non deleted vertex. ∎

Our second certifying algorithm for recognizing trivially perfect graphs is heavily based on cograph recognition.

Theorem 4.3 *There is a linear-time certifying algorithm for recognizing trivially perfect graphs that outputs certificates which can be authenticated in $O(n+m)$ time for membership and in $O(n)$ time for non-membership.*

Proof. First we present the algorithm.

1. Run a linear time recognition algorithm for cographs that outputs a cotree in case of membership and that outputs a P_4 in case of non membership.

2. If the cograph recognition algorithm rejects G and outputs a set of vertices inducing a P_4, then conclude that G is not trivially perfect; output NO and the set of vertices inducing a P_4.

3. Otherwise check that in the cotree T each 1-node at most one 0-node child. If this is the case, then conclude that G is trivially perfect; output YES and T.

4. If none of the above, then T has a 1-node with two 0-node children, say a and b. Each of them has at least two children. Let u and v be vertices assigned to leaves of subtrees rooted at different children of b, let w and x be vertices assigned to leaves of the subtrees rooted at different children of b. Then output NO, and the vertex subset $\{u, v, w, x\}$ which induces a C_4.

An authentication algorithm can check in $O(n+m)$ time that T is the cotree of G by Lemma 4.2. To verify that each 1-node of T has at most one child that is a 0-node can easily be done within the same time limit The certificate for non-membership can be authenticated in $O(n)$ time by Observation 2.1. ∎

References

[1] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph classes : A survey*. Philadelphia, SIAM, 1999.

[2] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.

[3] M. S. Chang, T. Kloks, D. Kratsch, J. Liu, and S. L. Peng. On the Recognition of Probe Graphs of Some Self-Complementary Classes of Perfect Graphs. *Proceedings of COCOON 2005*, Springer LNCS 3595:808–817.

[4] D. G. Corneil, H. Lerchs, and L. Stewart-Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3:163–174, 1981.

[5] D. G. Corneil, Y. Perl, and L. K. Stewart. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 14:926–934, 1985.

[6] M.C. Golumbic. Trivially perfect graphs. *Discrete Math.* 24:105–107, 1978.

[7] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Second edition. Annals of Discrete Mathematics 57. Elsevier, 2004.

[8] M. Habib and C. Paul. A simple linear time algorithm for cograph recognition. *Discrete Applied Mathematics*, 145:183–197, 2005.

[9] P. Heggernes and D. Kratsch. Linear-time certifying algorithms for recognizing split graphs and related graph classes. *Reports in Informatics* 328, University of Bergen, 2006.

[10] P. Heggernes, C. Paul, J. A. Telle, and Y. Villanger. Interval Completion is Fixed Parameter Tractable. *Reports in Informatics* 336, University of Bergen, 2006.

[11] P. Hell and J. Huang. Certifying LexBFS recognition algorithms for proper interval graphs and proper interval bigraphs. *SIAM J. Discrete Math.*, 18:554–570, 2004.

[12] H. Ito and M. Yokoyama. Linear time algorithms for graph search and connectivity determination on complement graphs. *Inf. Process. Lett.*, 66:209-213, 1998.

[13] H. Kaplan and Y. Nussbaum. Certifying algorithms for recognizing proper circular-arc graphs and unit circular-arc graphs. *Proceedings of WG 2006*, Springer LNCS 4271:289–300.

[14] D. Kratsch, R. M. McConnell, K. Mehlhorn and J. P. Spinrad. Certifying algorithms to recognize interval and permutation graphs. *SIAM J. Computing*, 36:326-353, 2006.

[15] K. Mehlhorn and S. Näher. *LEDA : A Platform for Combinatorial and Geometric Computing*, Cambridge University Press, 1999.

[16] D. Meister. Recognition and computation of minimal triangulations for AT-free claw-free and co-comparability graphs. *Discrete Appl. Math.*, 146:193–218, 2005.

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

[18] H. Wassermann and M. Blum. Software reliability via run-time result-checking. *Journal of the ACM*, 44:826–849, 1997.

[19] E. S. Wolk. The comparability graph of a tree. *Proc. Amer. Math. Soc.*, 13:789–795, 1962.

[20] E. S. Wolk. A note on "The comparability graph of a tree". *Proc. Amer. Math. Soc.*, 16:17–20, 1965.