**REPORTS
IN
INFORMATICS**

ISSN 0333-3590

Linear-time certifying algorithms for recognizing split graphs and related graph classes

Pinar Heggernes Dieter Kratsch

REPORT NO 328 June 2006

Department of Informatics

UNIVERSITY OF BERGEN

Bergen, Norway

Linear-time certifying algorithms for recognizing split graphs and related graph classes[*]

Pinar Heggernes[†] Dieter Kratsch[‡]

Abstract

We give linear-time certifying algorithms to recognize split graphs, threshold graphs, bipartite chain graphs and cobipartite chain graphs. All our algorithms provide a certificate: in case of membership this is a model for the input graph, in case of non-membership this is 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 $O(n)$.

1 Introduction

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. Certifying algorithms are highly desirable in practice to reduce the risk of erroneous answers caused by bugs in the implementation [12, 13]. For general discussions on result checking see also [17] and [13, section 2.14].

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. Another example is the linear-time planarity test which is part of the LEDA system [13, 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 [16], cographs [4], interval and permutation graphs [12], proper interval graphs [8, 15], proper interval bigraphs [8], proper circular-arc graphs, and unit circular-arc graphs [10].

Although linear-time recognition algorithms for split graphs and threshold graphs have been known [7], none of them provides a certificate for non-membership.

We give linear-time certifying algorithms to recognize split graphs, threshold graphs, bipartite chain graphs and cobipartite chain graphs. All non-membership certificates are based on a characterization of the graph class by forbidden induced subgraphs [1, 7]. The linear-time certifying algorithm for split graphs either provides a partition of the vertex set into a clique and an independent set as certificate of membership, or it provides as a certificate of non-membership a subset of vertices inducing a $2K_2$, C_4, or C_5 in the input graph. The linear-time certifying algorithm for threshold graphs either provides a partition of the vertex set into a clique and an

[*]The research was done while the second author was visiting the Algorithms Group at the University of Bergen, supported by The Research Council of Norway through grant 160778/V30.

[†]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, 57045 Metz Cedex 01, France. Email: kratsch@univ-metz.fr

independent set and a so-called nested neighborhood ordering as certificate of membership, or it provides as a certificate of non-membership a vertex subset inducing a $2K_2$, C_4, or P_4 in the input graph. The linear-time certifying algorithm for bipartite chain graphs either provides a bipartition and a nested neighborhood ordering as certificate of membership, or it provides as a certificate of non-membership a vertex subset inducing a C_3, $2K_2$, or C_5 in the input graph. The linear-time certifying algorithm for cobipartite chain graphs either provides a partition of the vertex set into two cliques and a nested neighborhood ordering as certificate of membership, or it provides as a certificate of non-membership a vertex subset inducing $\overline{K_3}$, C_4 or C_5 in the input graph.

For all algorithms, the certificate of membership can be authenticated in time $O(n + m)$. Furthermore, since each certificate of non-membership is an induced subgraph of constant (small) size of the input graph, they can be authenticated in time $O(n)$.

Up to our knowledge there is no published linear-time certifying recognition algorithm for any of the considered graph classes.

2 Preliminaries

All graphs in this paper are simple and undirected, and all input graphs to our algorithms are connected. If the input graph is not connected, then each algorithm can be run separately on each connected component for graph classes containing disconnected graphs. 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, and it is denoted by $N(v)$. The *degree* of a vertex v is denoted by $d(v) = |N(v)|$. 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 a graph 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 $d(v_1) \leq d(v_2) \leq ... \leq d(v_n)$, then it is called a *non-decreasing degree ordering*. If α has the property that v_i is simplicial in $G[\{v_i, vi + 1, ..., v_n\}]$, for $1 \leq i < n$ then it is called a *perfect elimination ordering (peo)*. An ordering $\beta = (x_1, x_2, ..., x_k)$ of a subset $X \subseteq V$ of vertices is called a *nested neighborhood ordering* if it has the property that $(N(x_1) \setminus X) \subseteq (N(x_2) \setminus X) \subseteq ... \subseteq (N(x_k) \setminus X)$.

Observation 2.1 *Given a graph $G = (V, E)$ and a vertex subset $X \subseteq V$ which is a clique or an independent set, X has a nested neighborhood ordering if and only if any non-decreasing degree ordering of the vertices in X is a nested neighborhood ordering.*

Proof. Let X have a nested neighborhood ordering. If there are two vertices in X with the same degree, then we can conclude that they have the same neighborhood outside of X. This is because they have the same number of neighbors in X since X is an independent set or a clique, and no pair of neighborhoods outside of X are incomparable or disjoint. Thus any non-decreasing degree ordering of X is a nested neighborhood ordering. For the opposite direction, assume that X does not have a nested neighborhood ordering, then no ordering (either non-decreasing degree or not) can be a nested neighborhood ordering. ∎

The class of *chordal graphs* is the class of graphs containing no induced cycle of length longer than 3. A graph is chordal if and only if its vertex set can be ordered by a peo [6]. A cycle on k vertices is denoted by C_k, and a path on k vertices is denoted by P_k. The following graph is called $2K_2$: $(\{a, b, c, d\}, \{ab, cd\})$.

The following result is trivial. We include it for the sake of completeness.

Lemma 2.2 *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. ■

Note that whenever we use Lemma 2.2, the graph H is a forbidden induced subgraph of graph class $\mathcal{G}$.

3 Split graphs

A graph is a *split graph* if its vertex set can be partitioned into a clique K and an independent set I, in which case we call (I, K) a *split partition.* Consequently, the split graph property is preserved under the graph complement operation. In fact, the class of split graphs consists of exactly those graphs that are both chordal and co-chordal [5]. A characterization of split graphs through forbidden induced subgraphs is the following.

Theorem 3.1 [5] *A graph is split if and only if it contains no vertex subset that induces $2K_2$, C_4, or C_5.*

For any split graph G, a non-decreasing degree ordering is a perfect elimination ordering, since in each such ordering, all vertices having neighbors only within the clique are ordered before all vertices having neighbors outside the clique.

Based on these properties of split graphs, we give in Figure 1 a certifying algorithm for recognizing split graphs.[1]

Theorem 3.2 *Algorithm Certifying-split is correct.*

Proof. If the algorithm returns YES then definitely the graph is split, since every induced subgraph of each iteration is split. If the graph is split, then a non-decreasing degree sequence is a peo and it makes sure that the vertices of the clique are ordered in the first k places, so the NO cases will never occur, and the algorithm will give a YES reply. Thus the algorithm outputs the correct YES or NO reply.

We need to argue that in the case of a NO reply, the output vertex subset indeed induces a forbidden subgraph of G. The algorithm outputs a NO and returns in three cases. These are numbered (1), (2), and (3) in the algorithm, and we will consider each case separately.

[1]In the algorithm, by **return** we mean that the algorithm terminates at that point, and the following lines are not executed.

Algorithm Certifying-Split
Input: A graph $G = (V, E)$.
Output: A reply YES if G is split together with a partition of V into an independent set I and a clique K, or a reply NO if G is not split together with a vertex subset inducing $2K_2$, C_4, or C_5.

```
Compute a non-decreasing degree ordering $\alpha = (v_1, v_2, ..., v_n)$ of $G$;
**if** $\alpha$ is not a peo of $G$ **then**
    Find two neigbors $v_j$ and $v_k$ of $v_i$ such that $v_j v_k \notin E$ and $i < j < k$;
    **if** $v_j$ and $v_k$ have a common neighbor $z$ with $v_i z \notin E$ **then**
        $\{v_i, v_j, z, v_k\}$ induces a $C_4$;
    **else**
        Find two vertices $x$ and $y$ such that $v_j x, v_k y \in E$ and $v_j y, v_k x \notin E$;
        **if** $xy \in E$ **then** $\{v_i, v_j, x, v_k, y\}$ induces a $C_5$;
        **else** $\{v_j, x, v_k, y\}$ induces a $2K_2$;
        **end-if**
    **end-if**
    Output NO and the vertex set inducing $C_4$, $C_5$ or $2K_2$, found above;
    **return**; **(1)**
**end-if**
Compute the largest size of a clique in $G$, and denote it $k$;
$K = \emptyset$;  $I = \emptyset$;   $i = n$;
**while** $|K| \leq k - 1$ **do**
    $A = N(v_i) \cap K$;
    **if** $|A| = |K|$ **then**
        $K = K \cup \{v_i\}$;
    **else**
        Find a neighbor $x \notin K$ and a non-neighbor $y \in K$ of $v_i$;
        Find a neighbor $z$ of $y$ such that $v_i z, v_i x \notin E$;
        Output NO and $\{v_i, x, y, z\}$ which induces a $2K_2$;
        **return**; **(2)**
    **end-if**
    $i = i - 1$;
**end-while**
**while** $i \geq 1$ **do**
    $A = N(v_i) \cap (K \cup I)$;
    **if** $A \subseteq K$ **then**
        $I = I \cup \{v_i\}$;
    **else**
        Denote by $x$ the single vertex in $A \cap I$;
        Find a common neighbor $y \in K$ of $x$ and $v_i$;
        Find a neighbor $z$ of $y$ with $xz \notin E$;
        Output NO and $\{v_i, x, y, z\}$ which induces a $2K_2$;
        **return**; **(3)**
    **end-if**
    $i = i - 1$;
**end-while**
Output YES and $K$ and $I$;
```

Figure 1: Algorithm Certifying-Split

(1) In this case, there exists a vertex v_i with neighbors v_j and v_k such that $v_jv_k \notin E$, and $i < j < k$. Since the degrees of v_j and v_k are at least as high as that of v_i, both v_j and v_k must have at least one neighbor other than v_i. For any common neighbor z of v_j and v_k, either edge $v_iz \in E$, or G has a chordless 4-cycle induced by $\{v_i, v_j, z, v_k\}$. Assume that every common neighbor of v_j and v_k is also a neighbor of v_i. In this case each of v_j and v_k must have a neighbor other than v_i that is not adjacent to the other or to v_i, because of the above degree argument. Thus there are edges v_jx and v_ky such that $x \neq y \neq v_i$, and neither x nor y is adjacent to v_i. Now, either $xy \in E$ and we have a C_5 induced by $\{v_i, v_j, x, y, v_k\}$, or $xy \notin E$ and we have a $2K_2$ induced by $\{v_j, x, v_k, y\}$.

For the rest of the proof we know that α is a peo, since the algorithm did not terminate at (1). Therefore, we also know that G is chordal, and the largest size k of a clique of G can be computed in time $O(n + m)$ (see e.g. [7]).

(2) In this case, $|K| \leq k - 1$. We know that there are k vertices of degree at least $k - 1$ in the graph, and since v_i is picked before some of these, $d(v_i) \geq k - 1$. Thus, since v_i has at least one non-neighbor $y \in K$, it has at least one neighbor $x \notin K$. Notice that no neighbor x of v_i outside of K can be adjacent to y. This is because every such neighbor has an earlier position in α, and its higher numbered neighbors in $K \cup \{v_i\}$ induce a clique, since α is a peo. Thus $v_iy, xy \notin E$. Since y has degree at least as high as that of v_i, $d(y) \geq k - 1$, and since there are at most $k - 2$ vertices in K in addition to y, vertex y must have a neighbor z outside of K. The positions of x and z in α are earlier than v_i and y. Therefore xz cannot be an edge in G, otherwise v_iy would be an edge in G by the property of a peo. Thus $\{v_i, x, y, z\}$ induces a $2K_2$.

(3) In this case, v_i has exactly one neighbor x in I, since I is an independent set, and the higher numbered neighbors of v_i induce a clique by the peo property of α. Observe that no non-neighbor of x in K can be a neighbor of v_i, since x has a higher position in α, which is a peo. So the set of neighbors of v_i in K is a subset of neighbors of x in K. In addition, x has at most $k - 1$ neighbors in K, because otherwise we get a clique of size $k + 1$. Thus there is a vertex y in K such that $xy, v_iy \notin E$. Now we will argue that y has a neighbor z which is not adjacent to x or v_i. Assume first that y has a neighbor z outside of K. Vertex z cannot be adjacent to x or v_i since either z is already safely placed in I, or it has a smaller position in α and this would violate the peo property of α. Assume for the other case that y has no neighbor outside of K. This means that $d(y) = k - 1$. Since y has higher position in α than x, we can conclude that $d(x) \leq k - 1$. Since v_i is a neighbor of x, x has at most $k - 2$ neighbors in K, and thus there are at least two vertices in K that are both non-adjacent to x, and consequently to v_i. Therefore, in this case, y has a neighbor z in K which is not adjacent to x or v_i. In either case, we get the desired $2K_2$ induced by $\{v_i, x, y, z\}$. ■

Theorem 3.3 *The running time of Algorithm Certifying-Split is $O(n + m)$.*

Proof. Golumbic [7] describes a linear-time procedure that checks whether a given ordering is a peo, and if not, provides the vertices v_i, v_j, v_k that are used in Algorithm Certifying-Split. Furthermore, in [7] it is explained how the size of a largest clique in a chordal graph can be computed in linear time.

In the rest of the algorithm, for each v_i, the time required is linear in the number of edges between v_i and the already processed vertices, as long as v_i is placed in K or I. Thus the total time required by all steps that place a vertex in K or I is $O(n + m)$. Every time we start looking for a forbidden induced subgraph we are already sure that the output is NO. Thus we search for a forbidden subgraph at most once during an execution of the algorithm. Finding such a forbidden subgraph requires $O(n)$ time, since it only concerns checking the neighborhoods of at most 4 vertices. ■

Theorem 3.4 *The certificates returned by Algorithm Certifying-Split can be authenticated in $O(n+m)$ time for input graphs that are split, and $O(n)$ time for input graphs that are not split.*

Proof. When the algorithm concludes that the graph is split, it returns K and I. It can be checked in $O(n+m)$ time that K is indeed a clique and I is indeed an independent set, and that there are no other vertices in the graph. When the algorithm concludes that the graph is not split, a vertex subset of 4 or 5 vertices is returned as a certificate. By Lemma 2.2, it can be checked in $O(n)$ time that the vertex subset indeed induces a forbidden subgraph. ■

4 Threshold graphs

A graph $G = (V, E)$ is a *threshold graph* if there exists an assignment of non-negative integers to the vertices of G and an integer threshold t such that for every vertex subset $X \subseteq V$ the following holds: X is an independent set if and only if the sum of the labels of the vertices in X is at most t.

Some important characterizations of threshold graphs that will be used for our certifying algorithm for recognizing them are gathered in the following theorem.

Theorem 4.1 [3, 7, 14] *For a connected graph $G = (V, E)$, the following are equivalent:*

- *G is a threshold graph.*
- *G contains no vertex subset that induces $2K_2$, P_4, or C_4.*
- *G is split and the vertices of the independent set (in any split partition) have a nested neighborhood ordering.*
- *When we repeatedly delete a universal vertex and resulting isolated vertices until no universal vertices remain, the resulting graph is empty.*

Based on these characterizations, we give a certifying algorithm for recognizing threshold graphs in Figure 2. In particular, the last point of Theorem 4.1 [7, Exercise 10-4] constitutes the recognition part of our algorithm.

Theorem 4.2 *Algorithm Certifying-Threshold is correct.*

Proof. If the input graph is not split the algorithm returns a forbidden subgraph. If it is split, then we have a partition of its vertices into K and I.

If there are any universal vertices in a graph then any vertex of highest degree must be universal. Note that the non-decreasing degree order of the vertices does not change by deleting universal or isolated vertices. Thus at every step of the while-loop, v_i is universal if and only if the remaining graph contains universal vertices. Note also that, as soon as we reach the first deleted vertex, all remaining vertices are already deleted, since they have smaller degree. Hence at the end of the while-loop we can conclude that G is threshold if and only if the value of *threshold* is TRUE, by the last point of Theorem 4.1. The ordering β that is output is a non-decreasing degree ordering of the vertices in I. Since we already know that these vertices have a nested neighborhood ordering, then by Observation 2.1, β is a nested neighborhood ordering.

```
Algorithm Certifying-Threshold;
Input: A graph $G = (V, E)$.
Output: A reply YES if $G$ is threshold together with a nested neighborhood
ordering $\beta$ of the vertices in the independent set, or a reply NO if $G$ is not
threshold together with a vertex set inducing $2K_2$, $C_4$, or $P_4$.

Run Algorithm Certifying-Split to recognize whether $G$ is split;
if $G$ is not split then
    Output NO, and the vertex set inducing $2K_2$, $C_4$, or $P_4$ (contained in $C_5$)
    that is returned by the above call to Algorithm Certifying-Split;
else
    Let $K$ and $I$ be as returned by the above call to Algorithm Certifying-Split;
    Let $\alpha = (v_1, v_2, ..., v_n)$ be a non-decreasing degree ordering of $G$;
    $\beta = (v_1, v_2, ..., v_{|I|})$;
    $threshold$ =TRUE;  $i = n$;
    while $v_i$ is undeleted do
        if $v_i$ is universal then
            delete $v_i$;
            for all neighbors $x$ of $v_i$ do
                $N(x) = N(x) \setminus \{v_i\}$;
                if $d(x) = 0$ then
                    delete $x$;
                end-if
            end-for
        else
            $threshold$ = FALSE;
            continue; (abort the while-loop)
        end-if
        $i = i - 1$;
    end-while
    if $threshold$ then
        Output YES, and $\beta$;
    else
        repeat
            Delete all remaining vertices of $K$ that have no neighbors in $I$;
            Delete all remaining vertices of $I$ that are adjacent to all vertices of $K$;
        until no such vertices remain;
        Choose a vertex $v$ of $I$ of highest degree;
        Find a non-neighbor $y$ of $v$ in $K$;
        Find a neighbor $z$ of $y$ in $I$;
        Find a a vertex $w \in K$ that is a neighbor $v$ and a non-neighbor of $z$;
        Output NO and $\{v, w, y, z\}$ which induces a $P_4$;
    end-if
end-if
```

Figure 2: Algorithm Certifying-Threshold

If we have concluded that G is not threshold, we know that the remaining graph must contain a P_4. Note that any remaining vertex of K that has no neighbors outside of K cannot belong to a P_4. Similarly, any remaining vertex of I that is adjacent to all remaining vertices of K cannot belong to a P_4. Therefore we delete all such vertices. The remaining graph cannot be complete, because otherwise the repeat-loop would not have stopped. It cannot be empty, since this would mean that there was an undeleted universal vertex after the while-loop. There cannot be any isolated vertices either, since we never delete any neighbor of a vertex in I in the repeat-loop. Thus, in the remaining graph after the repeat-loop, vertex v of highest degree in I has at least one non-neighbor y in K, and y has at least one neighbor z in I. Since the degree of v is at least the degree of z, v must have a neighbor w in K which is not adjacent to z. Hence we have a set $\{v, w, y, z\}$ which induces a P_4. ∎

Theorem 4.3 *The running time of Algorithm Certifying-Threshold is $O(n + m)$.*

Proof. Trivially, the while-loop requires $O(n + m)$ time. As argued above, the non-decreasing degree order never changes. Thus in the repeat-loop it is enough to check the vertices of I starting from the highest degree and the vertices of K starting with the lowest degree. The last steps require linear time. ∎

Theorem 4.4 *The certificates returned by Algorithm Certifying-Threshold can be authenticated in $O(n + m)$ time for input graphs that are threshold, and $O(n)$ time for input graphs that are not threshold.*

Proof. A certificate of membership for input graph G consists of K, I and β. It can be checked in linear time whether β is a nested neighborhood ordering of I as follows. The first vertex of β can mark its neighbors, and initialize a variable that keeps the number of marked vertices. Starting from the second vertex of β, each vertex can first check if it is adjacent to all marked vertices by counting the number of marked neighbors and comparing to the counter, and then mark its neighbors and update the counter. From β, I and K are implicit, and we can check in linear time that I is an independent set and K is a clique. Thus the membership certificate can be authenticated in $O(n + m)$ time. The argument for $O(n)$ authentication time of non-membership certificates is given in Lemma 3.4. ∎

5 Chain graphs

A graph $G = (V, E)$ is a *difference graph* if there exist real numbers a_v for each $v \in V$ and a real number t, such that $|a_v| < t$ for $v \in V$ and $uv \in E$ if and only if $|a_u - a_v| \geq t$. A graph whose vertex set can be partitioned into two independent sets is called a *bipartite graph.* This partition of the vertex set into two independent sets is called *bipartition*, and it is unique if and only if the graph is connected. A graph is bipartite if and only if contains no induced cycle of odd length [11]. It follows from the definition of difference graphs that they are bipartite.

Yannakakis gave difference graphs another name. He called them *chain graphs*, and defined a bipartite graph to be a chain graph if one of the sides of the bipartition has nested neighborhood property. He also showed that one side has this property if and only if both sides have the property [19].

We will refer to difference graphs as *bipartite chain graphs*, because we also define a graph class called *co-bipartite chain graphs.* A co-bipartite graph is a graph whose complement is bipartite. A bipartition into two independent sets in a bipartite graph is a bipartition into two cliques in its complement. We will say that a graph is *co-bipartite chain* if it is co-bipartite and the neighborhood of one of the sides of the bipartition has a nested neighborhood ordering. In this case both sides have a nested neighborhood ordering.

Theorem 5.1 [14] *A graph $G = (V, E)$ is a bipartite chain graph if and only if there is a bipartition of V into independent sets X and Y such that adding all possible edges with both endpoints in X yields a threshold graph.*

Theorem 5.2 [14] *A graph is a bipartite chain graph if and only if it contains no vertex subset that induces $2K_2$, C_3, or C_5.*

Based on the above results, we can obtain a linear-time certifying algorithm for bipartite chain graphs.

Theorem 5.3 *There is a linear-time certifying algorithm for recognizing bipartite chain 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 check whether the input graph is bipartite. It is well known that simple modifications of breadth-first search can be used to find an odd cycle in a graph or give a bipartition of it into two independent sets in linear time. Thus, after this test, if the graph is not bipartite, we get a certificate in the form of an odd cycle. Hence, we output a vertex set inducing C_3, C_5, or $2K_2$ (contained in a larger odd cycle) as a certificate.

If the reply is yes, we get the bipartition. We proceed to compute a non-decreasing degree ordering of each side of the bipartition. By Observation 2.1 the ordering of each side is a nested neighborhood ordering if and only if the input graph is a chain graph. We check this in linear time as described in the proof of Theorem 4.4. If we conclude that the graph is a chain graph, we output YES and the certificates for membership: the bipartition and the ordering of each side, which can be authenticated in $O(n + m)$ time.

If we conclude that the input graph is bipartite but not chain, then we know that it contains a $2K_2$. To find a vertex set inducing $2K_2$, we repeatedly delete from either side a vertex that is adjacent to all vertices of the other side and the resulting isolated vertices, since no such vertex can be a part of a $2K_2$. After this, we take a vertex v of highest degree. We know that v has a non-neighbor y on the other side. Since y is not isolated, it has a neighbor z on the same side as v. Since the degree of v is at least the degree of z, v has at least one neighbor x which z is not adjacent to. We can return $\{v, x, y, x\}$ which induces a $2K_2$.

By Lemma 2.2, the certificates for non-membership can be authenticated in $O(n)$ time. Furthermore, breadth-first search is linear, and all remaining steps require $O(n + m)$ time, by straight forward analysis. ∎

Consequently, there is an alternative linear-time certifying algorithm for threshold graphs, using a linear-time certifying algorithm for bipartite chain graphs: First we check whether the input graph is split. If not, we return a certificate. If it is split, then we delete all edges within the clique, and we run the above described certifying algorithm for recognizing bipartite chain graphs.

Now we want to show that there is a linear-time certifying algorithm for recognizing co-bipartite chain graphs.

Observation 5.4 *A graph is bipartite chain if and only if its complement is co-bipartite chain.*

Proof. Let $G = (V, E)$ be a bipartite chain graph, with bipartition $V = X + Y$. There is an ordering $(x_1, x_2, ..., x_k)$ of the vertices in X, such that $(N(x_1) \cap Y) \subseteq$

$(N(x_2) \cap Y) \subseteq ... \subseteq (N(x_k) \cap Y)$. Consequently, $(Y \setminus N(x_k)) \subseteq (Y \setminus N(x_{k-1}) \subseteq ... \subseteq (Y \setminus N(x_1))$. Let $\overline{N}(x_i)$ denote the set of vertices in G that are not adjacent to x_i, for $1 \leq i \leq k$. Then the given chain of set inclusions is equivalent to $(\overline{N}(x_k) \cap Y) \subseteq (\overline{N}(x_{k-1}) \cap Y) \subseteq ... \subseteq (\overline{N}(x_1) \cap Y)$, and hence X has a nested neighborhood ordering in the complement of G. Since both X and Y are cliques in $\overline{G}$, the result follows in one direction, and the other direction is completely symmetric. ∎

We can thus characterize co-bipartite chain graphs through their forbidden subgraphs. In the following, we denote an edgeless graph on 3 vertices by I_3.

Theorem 5.5 *A graph is a co-bipartite chain graph if and only if it contains no vertex subset that induces I_3, C_4, or C_5.*

Proof. Follows from Theorem 5.2 and Observation 5.4, since the complement of C_3 is I_3, the complement of $2K_2$ is C_4, and the complement of C_5 is C_5. ∎

Theorem 5.6 *There is a linear-time certifying algorithm for recognizing co-bipartite chain 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 study the complement $\overline{G}$ of G. It is well known that breadth-first search on $\overline{G}$, and related operations like computing connected components of $\overline{G}$, can be performed in $O(n+m)$ time using the adjacencies in G, without actually computing $\overline{G}$ explicitly [2, 9]. In case $\overline{G}$ is disconnected with at least two connected components containing at least one edge each, then we can easily find a set of vertices inducing $2K_2$ in $\overline{G}$, which induces C_4 in G, and hence G cannot be co-bipartite chain. In this case we return a reply NO and the set of vertices inducing C_4 as a certificate. In the opposite case, we simply check whether $\overline{G}$ is bipartite. If $\overline{G}$ is not bipartite, then we find a set of vertices inducing C_3, C_5, or $2K_2$ (contained in a larger odd cycle) in $\overline{G}$. These sets of vertices induce I_3, C_5, or C_4, respectively, in G. Again, we reply NO, and return the found set of vertices. Until this point, we have only used breadth-first search on $\overline{G}$, and thus spent $O(n+m)$ time.

After the above steps, if $\overline{G}$ is bipartite, then the bipartition is unique (except any isolated vertices), and gives a bipartition of G into two cliques, and we know that G is co-bipartite. The remaining work is done on G. The isolated vertices of $\overline{G}$ are universal in G, and can safely be placed on either side of the bipartition.

Now we remove each edge of G with both endpoints belonging to the same side of the bipartition. Definitely, the resulting graph G' is bipartite. Next, we check that each side of the bipartition has a nested neighborhood ordering, as described in the proof of Theorem 4.4. If we conclude that G' is not bipartite chain, then we get a set of vertices inducing a $2K_2$ in G', and we return this set as a certificate for a NO reply, since this set induces a C_4 in G. If we conclude that G' is bipartite chain, then we return a reply YES and the ordering of the vertices of each side.

The work described in the above paragraph requires $O(n+m)$ time. The certificates can be authenticated in $O(n+m)$ time for membership, and in $O(n)$ time for non-membership. ∎

6 Concluding remarks

We have given the first linear-time certifying algorithms for recognizing split, threshold, bipartite chain, and co-bipartite chain graphs, where the certificates for membership and non-membership can be authenticated in $O(n+m)$ and $O(n)$ time, respectively.

References

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

[2] K. W. Chong, S. D. Nikolopoulos, and L. Palios. An optimal parallel co-connectivity algorithm. *Theory of Computing Systems*, 37:527–546, 2004.

[3] V. Chvátal and P. L. Hammer. Set-packing and threshold graphs. Univ. Waterloo Res. Report, CORR 73-21, 1973.

[4] D. G. Corneil, Y. Perl, and L.K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14:926–934, 1985.

[5] S. Földes and P. L. Hammer. Split graphs. *Congressus Numerantium*, 19:311–315, 1977.

[6] D.R. Fulkerson and O.A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Math.*, 15:835–855, 1965.

[7] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.

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

[9] H. Ito and M. Yokoyama. Linear time algorithms for graph search and connectivity determination on complement graphs. *Information Processing Letters*, 66:209-213, 1998.

[10] H. Kaplan and Y. Nussbaum. Certifying algorithms for recognizing proper circular-arc graphs and unit circular-arc graphs. *Proceedings of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science* (WG 2006), Lecture Notes in Computer Science, to appear.

[11] D. König. *Theorie der Endlichen und Unendlichen Graphe*, Akademische Verlagsgesellschaft, 1936.

[12] D. Kratsch, R. M. McConnell, K. Mehlhorn and J. P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *Proceedings of the 14th annual ACM-SIAM Symposium on Discrete Algorithms* (SODA 2003), pp. 158–167, 2003.

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

[14] N. V. R. Mahadev and U. N. Peled. *Threshold Graphs and Related Topics.* Annals of Discrete Mathematics 56. North-Holland, 1995.

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

[16] 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 Journal on Computing*, 14:254–255, 1985.

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

[18] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Meth.*, 2:77–79, 1981.

[19] M. Yannakakis. Node deletion problems on bipartite graphs. *SIAM J. Comput.*, 10:310-327, 1981.