Information Processing Letters 74 (2000) 129–139

Information Processing Letters

www.elsevier.com/locate/ipl

Recognizing cographs and threshold graphs through a classification of their edges

Stavros D. Nikolopoulos [1]

Department of Computer Science, University of Ioannina, P.O. Box 1186, GR-45110 Ioannina, Greece

Received 24 September 1999; received in revised form 17 February 2000
Communicated by K. Iwama

Abstract

In this work, we attempt to establish recognition properties and characterization for two classes of perfect graphs, namely cographs and threshold graphs, leading to constant-time parallel recognition algorithms. We classify the edges of an undirected graph as either free, semi-free or actual, and define the class of A-free graphs as the class containing all the graphs with no actual edges. Then, we define the actual subgraph G_a of a non-A-free graph G as the subgraph containing all the actual edges of G. We show properties and characterizations for the class of A-free graphs and the actual subgraph G_a of a cograph G, and use them to derive structural and recognition properties for cographs and threshold graphs. These properties imply parallel recognition algorithms which run in O(1) time using O(nm) processors. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Cographs; Threshold graphs; Edge classification; Graph partition; Recognition; Parallel algorithms; Complexity

1. Introduction

Cographs (also called *complement reducible* graphs) are defined as the graphs which can be reduced to single vertices by recursively complementing all connected subgraphs. More precisely, the class of cographs can be defined recursively as follows:

(i) a single-vertex graph is a cograph;
(ii) the disjoint union of a cograph is a cograph;
(iii) the complement of a cograph is a cograph.

Cographs have arisen in many disparate areas of mathematics and computer science and have been independently rediscovered by various researchers under various names such as D^*-graphs, P_4 restricted graphs, 2-parity graphs and HD-graphs or Hereditary Dacey graphs. Cographs themselves were introduced in the early 1970s by Lerchs [12] who studied their structural and algorithmic properties. Lerchs has shown, among other properties (see [1,5,6,13]), the following two very nice algorithmic properties:

(P1) cographs are exactly the P_4 restricted graphs, and
(P2) cographs have a unique tree representation called cotree.

Threshold graphs, a well-known class of perfect graphs, are defined as those graphs where stable subsets of their vertex sets can be distinguished by using a single linear inequality. Equivalently, a graph $G = (V, E)$ is threshold if there exists a threshold assignment $[\alpha, t]$ consisting of a labeling α of the vertices by non-negative integers and an integer threshold t such that:

(i) S is a stable set if and only if $\alpha(v_1) + \alpha(v_2) + \cdots + \alpha(v_p) \leqslant t$,

[1] Email: stavros@cs.uoi.gr.

where $v_i \in S,\ 1 \leqslant i \leqslant p$, and $S \subseteq V$. Threshold graphs were introduced in 1973 by Chvátal and Hammer [4].

There are several recognition algorithms for the class of threshold graphs which run in linear sequential time. On the other hand, the class of cographs is known to have logarithmic-time parallel recognition algorithms. For the class of cographs, Adhar and Peng [1] presented a parallel recognition algorithm which requires $O(\log^2 n)$ time and uses $O(nm)$ processors on a CRCW PRAM model of computation, where n and m are the number of vertices and edges in the graph, respectively. Dahlhaus [7] proposed a nearly optimal parallel recognition algorithm which runs in $O(\log^2 n)$ time with $O(n+m)$ processors on a CREW PRAM model. Recently, He [10] published a cograph recognition algorithm working in $O(\log^2 n)$ time with $O(n+m)$ processors on a CREW PRAM model. It is worth noting that all previously known parallel algorithms use the fact that cographs can be represented by a unique tree (so-called cotree). This representation forms the base for the logarithmic time parallel recognition [5,7,10]. As far as threshold graphs are concerned, De Agostino and Petreschi [2] presented a parallel algorithm derived from a characterization based on degrees that runs in $O(\log n)$ time with $O(n/\log n)$ processors on a EREW PRAM. The main technique used for recognizing threshold graphs is the degree partition of the vertex set [8]. (We note that the degree partition of the vertex set V of a graph $G=(V,E)$ is given by $V = D_0 + D_1 + \cdots + D_k$ where D_i is the set of all the vertices of degree $\delta_i,\ 0 \leqslant i \leqslant k$.)

In this paper we attempt to establish recognition properties and characterization for two classes of perfect graphs, namely cographs and threshold graphs, leading to efficient $O(1)$-time parallel recognition algorithms. Most of the previously proposed recognition algorithms for cographs are based on a unique tree representation, while recognition algorithms for threshold graphs are based on a degree partition. Here, we take a different approach relying on a classification of the edges of an undirected graph and on the fact that cographs contain no induced subgraph isomorphic to P_4 (see Lerchs [12]) and threshold graphs contain no induced subgraphs isomorphic to $2K_2$, P_4, or C_4 (see Chvátal and Hammer [4]). To this end, we classify the edges of a graph G as either *free, semi-free* or *actual*, according to the relationship of the closed neighborhoods of the endpoints (or end-vertices) of their edges and obtain the actual subgraph G_a which is the subgraph of G containing all the actual edges. Based on this classification, we define the class of *A-free* graphs as the class containing all the undirected graphs with no actual edges. Consequently, we prove that any A-free graph does not contain P_4 or C_4 as induced subgraphs. This implies that

(i) A-free graphs are a special kind of cographs, and
(ii) a graph is threshold if and only if it is an A-free graph and has no induced subgraphs isomorphic to $2K_2$.

We show that G has no induced subgraph isomorphic to $2K_2$ if and only if the complement G^c of G is an A-free graph. Moreover, we show certain properties and characterizations for the actual subgraph G_a of a cograph G. Based on these results, we propose $O(1)$-time algorithms for recognizing A-free graphs, cographs and threshold graphs using $O(nm)$ processors on a CRCW PRAM.

Throughout the paper $\log n$ denotes logarithm to the base two, n denotes the number of vertices and m denotes the number of edges in a graph.

2. Edge classification and graph partition

Let $G=(V,E)$ be an undirected simple graph with n vertices and m edges. Following the notation and terminology in [9], the *neighbourhood* of a vertex u of G is the set $N(u) = N_G(u)$ consisting of all vertices v which are adjacent with u. The *closed neighbourhood* is $N[u] = N_G[u] := \{u\} \cup N(u)$. The subgraph of G induced by a subset $S \subseteq V$ is denoted by $G[S]$. We shall use the notation $N_{G[S]}(u)$ (respectively $N_{G[S]}[u]$) to denote the neighbourhood (respectively closed neighbourhood) of a vertex u of the graph $G[S]$.

Given a graph $G=(V,E)$, we define three classes of edges in G, denoted by *AE*, *FE* and *SE* according to relationship of the closed neighborhoods of the endpoints of its edges [14]. Let $x=(u,v)$ be an edge of G. Then,

$$(u,v) \in FE \quad \text{if } N[u] = N[v],$$

$$(u,v) \in SE \quad \text{if } N[u] \subset N[v],$$

$$(u,v) \in AE \quad \text{if } N[u] - N[v] \neq \emptyset \text{ and } N[v] - N[u] \neq \emptyset.$$

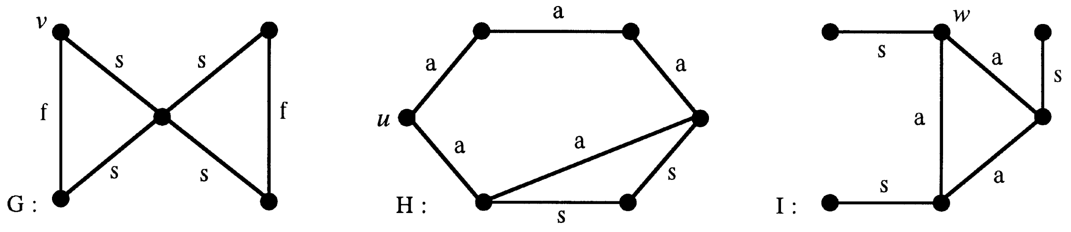

Fig. 1. Three undirected graphs. Free, semi-free and actual edges are denoted by f, s and a, respectively.

Obviously the edge set E of G can be partitioned into the three subsets of free edges, semi-free edges and of actual edges, respectively; that is, $E = FE + SE + AE$. We illustrate with three graphs G, H and I shown in Fig. 1.

Having classified the edges of a graph $G = (V, E)$ as either free, semi-free or actual, let us now define the class of *A-free* graphs as follows:

Definition 1. An undirected graph $G = (V, E)$ is called *A-free* if every edge of G is either free or semi-free edge.

By definition, a graph G is an A-free graph if and only if for every edge (x, y) of G, we have $N[x] \subseteq N[y]$ or $N[x] \supseteq N[y]$. The graph G in Fig. 1 is an A-free graph, while the graphs H and I in the same figure are not A-free graphs.

Let $G = (V, E)$ be a graph which contains actual edges; that is, G is not an A-free graph. We define the *actual subgraph* $G_a = (V_a, E_a)$ of the graph G as follows:

Definition 2. The graph $G_a = (V_a, E_a)$ is called *actual subgraph of* a graph $G = (V, E)$ if $E_a = AE$ and $V_a = \{v \in V \mid v$ is an endpoint of some edge of $E_a\}$, where AE is the set of the actual edges of G and $AE \neq \emptyset$.

The actual subgraph $G_a = (V_a, E_a)$ of the graph H of Fig. 1 is a C_5 (chordless cycle on 5 vertices), while the actual subgraph of the graph I of the same figure is a K_3 (complete graph on 3 vertices).

We now extend the notion of the neighbourhood $N(v)$ of a vertex v so that for any vertex u we define the i-distance neighbourhood of u denoted by $N(v, i)$, $i \geqslant 1$. The set $N(v, i)$, $i \geqslant 1$, contains all the vertices u such that the length of the shortest path from v to u is equal to i.

Given a connected graph $G = (V, E)$ and a vertex $v \in V$, we define a partition $\mathcal{L}(G, v)$ of the vertex set V (we shall frequently use the term *partition of the graph G*), with respect to the vertex v as follows:

$$\mathcal{L}(G, v) = \bigl\{N(v, i) \mid v \in V,\ 0 \leqslant i \leqslant L_v,\ 1 \leqslant L_v < |V|\bigr\},$$

where $N(v, i)$, $0 \leqslant i \leqslant L_v$, are the *adjacency-level sets*, or simply the *adjacency-levels*, and L_v is the *length* of the partition $\mathcal{L}(G, v)$ [15]. The adjacency-level sets of the partition $\mathcal{L}(G, v)$ of the graph G, are formally defined as follows:

$$N(v, i) = \bigl\{u \in V \mid d(v, u) = i\bigr\},$$

where $d(v, u)$ denotes the *distance* between vertices v and u in G. We point out that $d(v, u) \geqslant 0$, and $d(v, u) = 0$ when $v = u$, for every $v, u \in V$. (In the case where G is a disconnected graph, $d(v, u) = \infty$ when v and w do not belong to the same connected component.) Obviously,

$$L_v = \max\bigl\{d(v, u) \mid u \in V\bigr\},$$

$$N(v, 0) = \{v\} \quad \text{and} \quad N(v, 1) = N(v).$$

The adjacency-level sets $N(v, i)$, $0 \leqslant i \leqslant L_v$, of partition $\mathcal{L}(G, v)$, can easily be computed recursively as follows:

$$N(v, i) = \bigl\{u \mid (x, u) \in E \text{ and } x \in N(v, i-1)\bigr\} - \bigl\{N(v, i-1) \cup N(v, i-2)\bigr\}, \quad 2 \leqslant i \leqslant L_v < n.$$

We note that, these sets can also be computed by considering first the distance matrix of the graph G and then extracting all set information that is necessary. This computation can be efficiently done by using matrix multiplication; see [3].

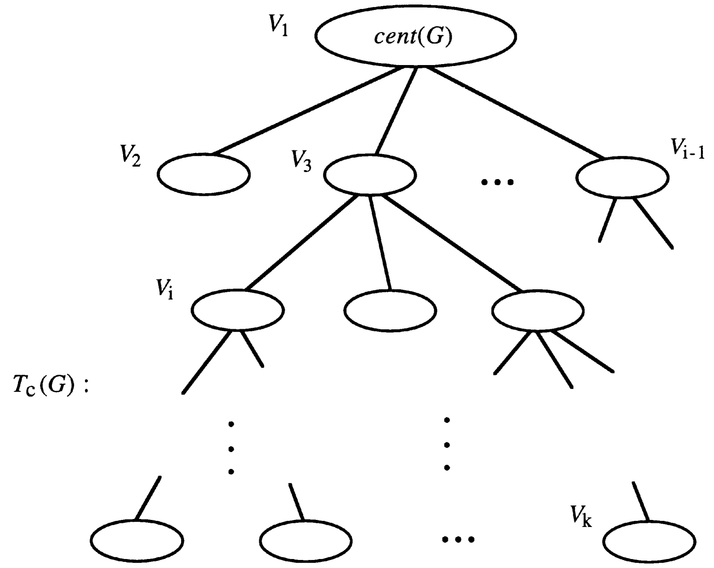

Fig. 2. The typical structure of an A-free graph. A line between nodes V_i and V_j indicates that each vertex of node V_i is adjacent to each vertex of node V_j.

3. Structural properties of A-free graphs

The following results provide algorithmic properties for the class of A-free graphs—that is, the class of all the undirected graphs with no actual edges. A typical structure of an A-free graph is shown in Fig. 2.

Let P_4 and C_4 denote the chordless path and the chordless cycle on four vertices, respectively. Based on the definition of the actual edges of a graph, we can easily show that the A-free graphs are exactly the graphs not having a P_4 or C_4 as an induced subgraph. Thus, the following lemma holds.

Lemma 3.1. *A graph $G = (V, E)$ is an A-free graph if and only if it contains no induced subgraph isomorphic to P_4 or C_4.*

Let $G = (V, E)$ be a connected A-free graph. For convenience, we here denote by $V(G)$ and $E(G)$ the sets V and E, respectively, and define

$$cent(G) = \{x \in V(G) \mid N[x] = V(G)\}.$$

Theorem 1 (see [11]). *The following two statements hold.*

(i) *A graph G is an A-free if and only if $G - cent(G)$ is an A-free graph.*
(ii) *Let G be a connected A-free graph. Then $cent(G) \neq \emptyset$. Moreover, if $G - cent(G) \neq \emptyset$, then $G - cent(G)$ contains at least two components.*

Then $V_1 := cent(G)$ is not an empty set. Put $G_1 = G$, and $G - V_1 = G_2 \cup G_3 \cup \cdots \cup G_r$, where each G_i is a component of $G - V_1$ and $r \geqslant 3$. Then since each G_i is an induced subgraph of G, G_i is also an A-free graph, and so let $V_i := cent(G_i) \neq \emptyset$ for $1 \leqslant i \leqslant r$. Since each component G_j of $G_i - cent(G_i)$ is also an A-free graph, we can continue this procedure until we get an empty graph. Then we finally obtain the following partition of $V(G)$.

$$V(G) = V_1 + V_2 + \cdots + V_k,$$

where $V_i = cent(G_i)$. Moreover we can define a partial order $\leqslant$ on $\{V_1, V_2, \ldots, V_k\}$ as follows:

$$V_i \leqslant V_j \quad \text{if } V_i = cent(G_i) \text{ and } V_j \subseteq V(G_i).$$

It is easy to see that this partition possesses has the following properties.

Theorem 2 (see [11]). *Let G be a connected A-free graph, and let $V(G) = V_1 + V_2 + \cdots + V_k$*

be the partition defined above, in particular, $V_1 :=$ cent(G). Then this partition and the partially ordered set $(\{V_i\}, \leqslant)$ have the following properties:

(P1) *If $V_i \leqslant V_j$, then every vertex of V_i and every vertex of V_j are joined by an edge of G.*
(P2) *For every V_i, cent$(G[\{\bigcup V_i \mid V_j \geqslant V_i\}]) = V_i$.*
(P3) *For every two V_s and V_t such that $V_s \leqslant V_t$, $G[\{\bigcup V_i \mid V_s \leqslant V_i \leqslant V_t\}]$ is a complete graph. Moreover, for every maximal element V_t of $(\{V_i\}, \leqslant)$, $G[\{\bigcup V_i \mid V_1 \leqslant V_i \leqslant V_t\}]$ is a maximal complete subgraph of G.*
(P4) *Every edge with both endpoints in V_i is a free edge.*
(P5) *Every edge with one endpoint in V_i and the other endpoint in V_j, where $V_i \neq V_j$, is a semi-free edge.*

We shall refer to the structure which meets the properties of Theorem 2 as *cent-tree $T_c(G)$*. We shall call *nodes* the elements of a cent-tree $T_c(G)$; that is, the vertex sets $V_1, V_2, \ldots, V_k$ of the partition of $V(G)$ of an A-free graph G. Fig. 2 shows the *cent-tree $T_c(G)$* of an A-free graph G; all edges in a V_i are free edges, while all edges between nodes are semi-free edges.

The cent-tree is a rooted tree with root V_1; every node V_i of $T_c(G)$ is either a leaf or has at least two children. Moreover, $V_s \leqslant V_t$ if and only if V_s is an ancestor of V_t.

If V_i and V_j are disjoint vertex sets of an A-free graph G, we say that V_i and V_j are *clique-adjacent* and denote $V_i \approx V_j$ if $V_i \leqslant V_j$ or $V_j \leqslant V_i$.

We point out that the property (P3) of Theorem 2 ensures that all the edges with both endpoints in a vertex set V_i are free edges, $1 \leqslant i \leqslant k$. A consequence of this property is that the vertex set $V_1 \cup V_i$ is not always a maximal clique. We can easily see that $V_1 \cup V_i$ is not a maximal clique if there exists a vertex set V_j such that $V_i \approx V_j$, $2 \leqslant j \leqslant k$.

4. Recognition properties of cographs

Based on the definition of cographs we can easily see that they can be obtained from a single vertex by performing a finite number of operations involving union and complementation (the definition is given in the introduction). Next, we present the fundamental theorem on cographs.

Theorem 3 (see [6]). *Let $G = (V, E)$ be an undirected graph. Then, the following statements are equivalent*:
(i) *G is a cograph*;
(ii) *G does not contain P_4 as a subgraph.*

By Theorem 3, a graph G is a cograph or D^*-graph if for every path in G with edges (v_1, v_2), (v_2, v_3), (v_3, v_4) the graph also contains the edges (v_1, v_3) or (v_2, v_4) or (v_1, v_4) [5]. A strict subset of cographs form the class of *diagonal graphs* or *D-graphs* (a strict subset of cographs) [16]. It is important to point out that Wolk [16] showed that the D-graphs are precisely the comparability graphs of rooted trees. This result was later quoted incorrectly as “A graph without induced subgraph isomorphic to P_4, that is, a cograph, is the comparability graph of rooted trees” [2]. The graph C_4 is a counter-example to this statement.

A graph is a *diagonal graph* or D-graph if for every path in G with edges (v_1, v_2), (v_2, v_3), (v_3, v_4), the graph also contains the edges (v_1, v_3) or (v_2, v_4) (see also [6]). Thus, the following result directly follows from Lemma 3.1 and Theorem 3.

Theorem 4.1. *Let $G = (V, E)$ be an A-free graph. Then, G is a cograph.*

Let us now consider the case where the graph G contains actual edges. Based on certain properties of the actual subgraph $G_a = (V_a, E_a)$, we show some characterizations of cographs leading to an efficient (constant-time) parallel recognition algorithm. Hereafter, the adjacency-level sets of the partition $\mathcal{L}(G_a, x)$ of the graph G_a are denoted by $N_a(x, i)$, $x \in V_a$ and $i \geqslant 0$.

The following theorem addresses a characterization of cographs which is at the heart of our parallel recognition algorithm.

Lemma 4.1. *Let $G = (V, E)$ be a graph and let $G_a = (V_a, E_a)$ be the actual subgraph of G. The graph G is a cograph if and only if*
(i) *$\bigcup_{0 \leqslant i \leqslant 2} N_a(x, i) = V_a$, and*
(ii) *there exists no actual edge $(y, z) \in E_a$ such that $y, z \in N_a(x, 2)$, and*
(iii) *there exists no semi-free edge $(x, y) \in SE$ such that $y \notin V_a$, for every vertex $x \in V_a$.*

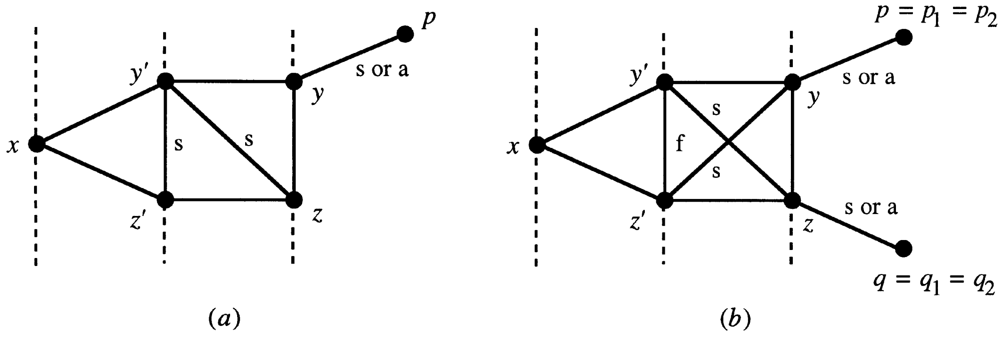

Fig. 3. The vertical dashed lines indicate the adjacency-levels. Free and semi-free edges are denoted by f and s, respectively; all the other edges are actual edges.

Proof. ($\Rightarrow$) Let $G = (V, E)$ be a cograph. Suppose that $V_a \neq \emptyset$; otherwise G is an A-free graph and, thus, G is a cograph (Theorem 4.1).

(i) By Theorem 2, G does not contain P_4 as a subgraph. Obviously then $N(x, 3) = \emptyset$ for every $x \in V$. Since G_a is a subgraph of G, $N_a(x, 3) = \emptyset$ for every $x \in V_a$. This implies, $\{x\} \cup N_a(x, 1) \cup N_a(x, 2) = V_a$ for every $x \in V_a$.

(ii) Suppose that there exists an actual edge $(y, z) \in E_a$ such that $y, z \in N_a(x, 2)$. Then, there exist vertices $y', z' \in N_a(x, 2)$ such that $(y, y') \in E_a$, $(z, z') \in E_a$. There are three cases to consider.

Case I: $(y, z') \notin E$ and $(z, y') \notin E$. Then, there exists a P_4 path $[z, y, y', x]$; a contradiction.

Case II: $(y, z') \notin E$ and $(z, y') \in E$ (see Fig. 3(a)). Since (y, z) is an actual edge, there exists a semi-free or actual edge (y, p) such that $(p, z) \notin E$. Then, there exists a P_4 path $[p, y, z, z']$; a contradiction.

Case III: $(y, z') \in E$ and $(z, y') \in E$ (see Fig. 3(b)). Since (y, z) is an actual edge, there exist semi-free or actual edges (y, p_1) and (z, q_1) such that $(p_1, z) \notin E$ and $(q_1, y) \notin E$. Moreover, since (y, y') and (z, z') are actual edges, there exist semi-free or actual edges (y, p_2) and (z, q_2) such that $(p_2, y') \notin E$ and $(q_2, z') \notin E$. Without loss of generality, we assume that $p = p_1 = p_2$ and $q = q_1 = q_2$. If $(x, p) \notin E$ then there exists a P_4 path $[p, y, y', x]$; a contradiction. If $(x, p) \in E$ then there exists a P_4 path $[z, y, p, x]$; a contradiction. The cases where $(x, q) \notin E$ and $(x, q) \in E$ follow similarly.

(iii) Suppose that there exists a semi-free edge $(x, y) \in SE$ such that $y \notin V_a$. Let (x, z) be an actual edge. Then, there exists a vertex z' such that $(z, z') \in E$ and $(x, z') \notin E$. There are two cases to consider.

Case I: There is no vertex x' such that $(x', x) \in E$ and $(x', z) \notin E$ (see Fig. 4(a)). Since (x, z) is an actual edge, we have $(y, z) \notin E$. Obviously, $(y, z') \notin E$; otherwise $C_4 = G[\{y, x, z, z'\}]$ and, thus, $y \in V_a$. Then, there exists a P_4 path $[y, x, z, z']$; a contradiction.

Case II: There is a vertex x' such that $(x, x') \in E$ and $(x', z) \notin E$ (see Fig. 4(b)). Vertices x' and z' are connected by an edge; otherwise, $[x', x, z, z']$ forms a P_4. Vertices y, z' are not connected by an edge; otherwise, $y \in V_a$. If both (y, z) and (y, x') are edges, then $y \in V_a$. Therefore, we have that either $[y, x, z, z']$ or $[y, x, x', z']$ forms a P_4; a contradiction.

($\Leftarrow$) Suppose that $G = (V, E)$ is not a cograph. By Theorem 3 G contains a $P_4 = [v_1, v_2, v_3, v_4]$. Therefore (v_2, v_3) is an actual edge, and $(v_1, v_3) \notin E$, $(v_2, v_4) \notin E$ and $(v_1, v_4) \notin E$. There are two cases to consider.

Case I: At least one of the edges (v_1, v_2), (v_3, v_4) is a semi-free edge. Let $(v_1, v_2) \in SE$. In this case, there exists a semi-free edge $(v_2, v_1) = (x, y) \in SE$ such that $y \notin V_a$ (see Fig. 5(a)).

Case II: Both (v_1, v_2) and (v_3, v_4) are actual edges. Then, there are vertices v_0 and v_5 such that $(v_1, v_0) \in E$, $(v_4, v_5) \in E$, $(v_0, v_2) \notin E$ and $(v_5, v_3) \notin E$. We distinguish two alternatives:

(A1) $v_0 \neq v_5$ (see Fig. 5(b)). Since $P_4 = [v_1, v_2, v_3, v_4]$, it is easy to see that $v_3 \in N_a(x, 2)$ and $v_4 \notin N_a(x, 1)$, where $x = v_1$. It follows that exactly one of the following alternatives holds:
 (i) $v_3, v_4 \in N_a(x, 2)$,
 (ii) $v_4 \in N_a(x, 3)$.

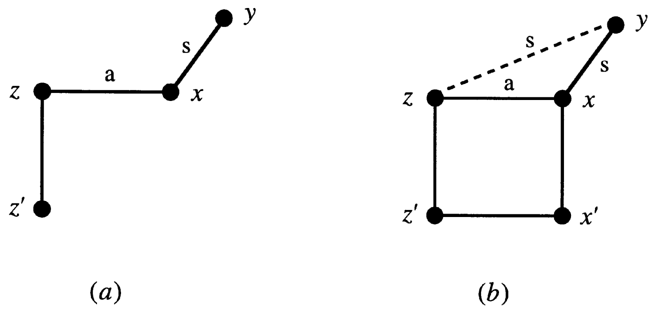

Fig. 4. Semi-free and actual edges are denoted by s and a, respectively.

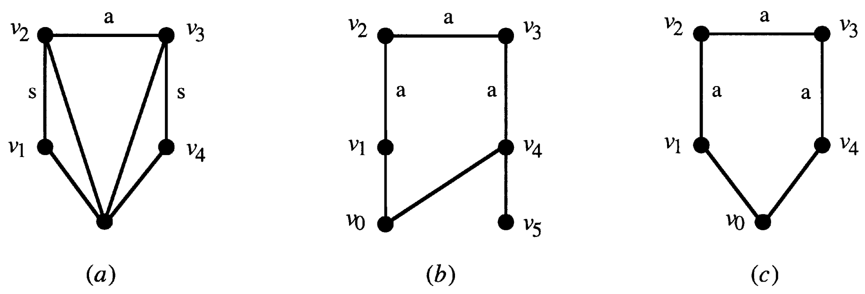

Fig. 5. Semi-free and actual edges are denoted by s and a, respectively.

If (i) holds then there exists an actual edge $(v_3, v_4) \in E_a$ such that $v_3, v_4 \in N_a(x, 2)$. If (ii) holds then $\{x\} \cup N_a(x, 1) \cup N_a(x, 2) \neq V_a$.

(A2) $v_0 = v_5$ (see Fig. 5(c)). In this case, both (v_1, v_0) and (v_0, v_4) are actual edges, and $(v_0, v_2) \notin E$ and $(v_0, v_3) \notin E$ (see (A1) in the case where either $(v_0, v_2) \in E$ or $(v_0, v_3) \in E$). Thus, there exists an actual edge $(v_2, v_3) = (y, z) \in E_a$ such that $y, z \in N_a(x, 2)$, where $x = v_0$. □

5. Recognizing threshold graphs

As stated in Section 1, there exists a sequential recognition algorithm for threshold graphs running in $O(n)$ time. The algorithm is based on the degree partition of an undirected graph and on some important characterizations of threshold graphs presented by Chvátal and Hammer [4]. The main idea of the recognition algorithm can be summarized as follows: Given an undirected graph $G = (V, E)$, first, it brings together all the vertices with the same degree. That is, the vertex set V of the graph is partitioned into k disjoint vertex sets $V = D_0 + D_1 + \cdots + D_k$ satisfying the property $v \in D_i$ if and only if $\delta(v) = i$, where $v \in V$ and $1 \leqslant i \leqslant k$. Then, it uses the following result: G is a threshold graph if and only if the recursions below are satisfied:

$$\delta_{i+1} = \delta_i + |D_{k-i}| \quad (i = 0, 1, \ldots, \lfloor k/2 \rfloor - 1),$$
$$\delta_i = \delta_{i+1} - |D_{k-i}| \quad (i = k, k-1, \ldots, \lfloor k/2 \rfloor + 1).$$

The approach used in this paper is different from the previous algorithm(s). Specifically, since our main objective is to achieve constant-time complexity for the recognition problem, we focus on forbidden subgraphs. In other words, we focus on the problem of recognizing induced subgraphs which are forbidden for threshold graphs. Chvátal and Hammer [4] have proved the following:

Theorem 4 (see [4]). *Let $G = (V, E)$ be a undirected graph. Then, the following statements are equivalent*:
(i) *G is a threshold graph.*
(ii) *G has no induced subgraph isomorphic to $2K_2$, P_4, or C_4.*

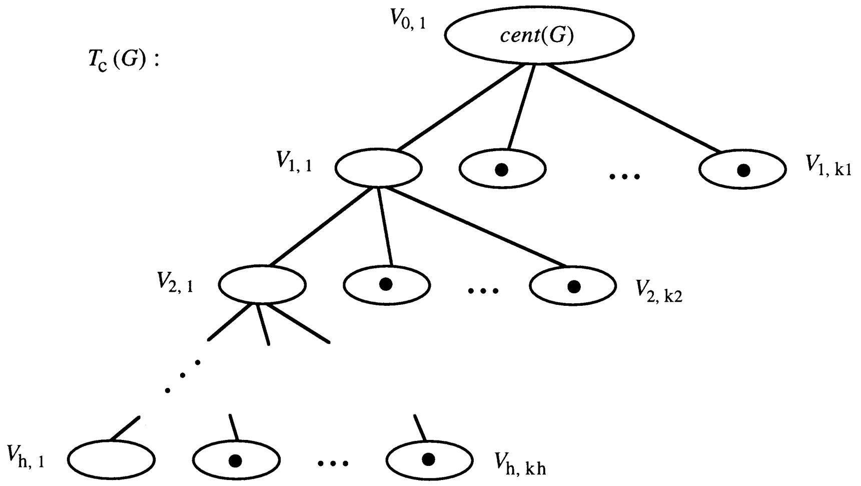

Fig. 6. The typical structure of an A-free graph which contains no induced subgraph isomorphic to $2K_2$. Nodes $V_{i,j}$ with $j = 1$ contain only one vertex.

We have proved that an A-free graph contains no induced subgraph isomorphic to P_4 or C_4 (see Lemma 3.1). By combining these results with the results of Theorem 4, we obtain the following theorem.

Theorem 5.1. *The threshold graphs are precisely those A-free graphs containing no induced subgraph isomorphic to* $2K_2$.

The following lemma gives us the conditions under which an A-free graph G contains no induced subgraph isomorphic to $2K_2$. We shall use $V_{h,k}$ to denote the kth node of the cent-tree $T_c(G)$ at level $h \geqslant 0$; $V_{0,1}$ denotes the root of $T_c(G)$ (see Fig. 6).

Lemma 5.1. *Let* $G = (V, E)$ *be an A-free graph. The following statements are equivalent*:
(i) G *has no induced subgraph isomorphic to* $2K_2$.
(ii) *The complement* G^c *of the graph* G *is an A-free graph.*

Proof. (i) $\Rightarrow$ (ii) Let G be an A-free graph and let $T_c(G)$ be the cent-tree of G rooted at $V_{0,1} := cent(G)$. Then, $level(x) = 0$ for every $x \in V_{0,1}$. Let $V_{1,1}, V_{1,2}, \ldots, V_{1,k1}$ be the children of the node $V_{0,1}$. Since G contains no induced subgraph isomorphic to $2K_2$, there exists at least one child of $V_{0,1}$ which may contain more that one node. Let $V_{1,1}$ be such a child; that is $|V_{1,1}| \geqslant 1$. If the subgraph induced by $V_{1,1}$ is an A-free graph (or equivalently $V_{1,1}$ is not a clique) then there exists at least one child of $V_{1,1}$ which may contain more that one node. Let $V_{2,1}$ be such a child. We continue in this way until all the children of the node $V_{h-1,1}$ are cliques (an isolate vertex is a clique); see Fig. 6. By construction, the cent-tree $T_c(G)$ of an A-free graph G has the following properties:

- The vertex set $K = V_{0,1} \cup V_{1,1} \cup \cdots \cup V_{h,1}$ is a clique.
- The vertex set $S = V - \{V_{0,1} \cup V_{1,1} \cup \cdots \cup V_{h,1}\}$ is an independent set.
- For every pair of nodes $x, y \in S$ such that $level(x) < level(y)$, $N(x) \subseteq N(y)$.

Let $G^c = (V, E^c)$ be the complement of the graph G. Then, K is an independent set and S is a clique in G^c. Let (x, y) be an arbitrary edge of G^c.

Case I. The edge $(x, y) \in E^c$ has one endpoint in the set K and the other endpoint in the set S. Then (x, y) is a semi-free edge.

Case II. The edge $(x, y) \in E^c$ has both endpoints in the set S. Since $N(x) \subseteq N(y)$ in G, we have that $N[y] \subseteq N[x]$. This implies that every edge in G^c is either free or semi-free. Thus, G^c is an A-free graph.

(ii) $\Rightarrow$ (i) Suppose that G contains an induced subgraph isomorphic to $2K_2$. Then, G^c contains an induced subgraph isomorphic to C_4; a contradiction. □

From the preceding Lemma 5.1 and Theorem 5.1 we may state the following characterization of threshold graphs.

Theorem 5.2. *An undirected graph $G = (V, E)$ is a threshold graph if and only if G and its complement G^c are A-free graphs.*

The above theorem enables us to design a constant-time parallel recognition algorithm for threshold graphs.

6. Recognition algorithms

Based on the results of the previous sections, we present here parallel recognition algorithms for A-free graphs, cographs and threshold graphs. The models of parallel computation used for analyzing the computational complexity of each algorithm are the well-known Concurrent-Read, Concurrent-Write PRAM (CRCW PRAM).

6.1. Free graphs

It is easy to formulate and analyze a parallel algorithm for computing the classes of free, semi-free and actual edges of a graph and, therefore, recognizing whether or not a undirected graph G is an A-free graph or a D-graph. We present the following results.

Theorem 6.1. *The free, semi-free and actual edges of an undirected graph G with n vertices and m edges can be computed in $O(1)$ time on a CRCW PRAM using $O(nm)$ processors.*

Corollary 6.1. *A-free graphs and D-graphs can be recognized in $O(1)$ time with $O(nm)$ processors on a CRCW PRAM model.*

Remark 6.1. We assume that the input graph is connected. Otherwise, the processor complexity of the CRCW implementation of the algorithm is $O(n^2 + nm)$. It is obvious that if the input graph is represented by the adjacency matrix, the constant-time recognition of free graphs and D-graphs (also, cographs and threshold graphs) requires at least $\Omega(n^2)$ processors, regardless of how many edges the graphs contain.

6.2. Cographs

We now present a parallel algorithm for recognizing cographs. The algorithm is based on the results of Theorem 4.1 and Lemma 4.1. Let $G = (V, E)$ be an undirected graph. By Theorem 4.1, we obtain that G is a cograph if it is an A-free graph. Therefore, we have to test whether or not the edge set *AE* is empty. If so, then G is a cograph. The case where G is not an A-free graph is handle by Lemma 4.1.

Next, we give a more formal listing of the recognition algorithm.

```
Step 1. **Compute** the edge sets $FE$, $SE$ and $AE$.
Step 2. **If** $AE \neq \emptyset$ **then** compute the actual subgraph
        $G_a = (V_a, E_a)$ of $G$;
        otherwise, $G$ is A-free and, thus, $G$ is a
        cograph; **exit**.
Step 3. **For** every vertex $x \in V_a$, **do** in parallel
        (3.1) compute the adjacency-level sets
              $N_a(x, 1)$ and $N_a(x, 2)$;
Step 4. **For** every vertex $x \in V_a$ **do** in parallel
        (4.1) if $\{x\} \cup N_a(x, 1) \cup N_a(x, 2) \neq V_a$ then
              $C_x \leftarrow$ false;
        (4.2) if there exists an actual edge $(y, z) \in$
              $E_a$: $y, z \in N_a(x, 2)$ then $C_x \leftarrow$ false;
        (4.3) if there exists a semi-free edge $(y, z) \in$
              $SE$: $y \notin V_a$, then $C_x \leftarrow$ false;
Step 5. **If** $C_x =$ true, for all vertices $x \in V_a$ **then** $G$ is
        a cograph;
        otherwise it is not;
```

Let us now compute the time-processor complexity of the proposed parallel algorithm for recognizing cographs.

We shall use a step-by-step analysis.

Steps 1, 2: These steps are executed in $O(1)$ time with $O(nm)$ processors on a CRCW PRAM (see Theorem 6.1).

Step 3: Let G be the input graph. We recall that the adjacency-level set $N(v, i)$ of G can be computed in $O(1)$ time, if the set $N(v, i-1)$ is given, $0 < i \leqslant L_v$. For the computation of $N(v, 2)$, δ_v sets of length δ are needed, where $\delta = \max\{\delta_u \mid u \in N(v, 1)$ and $v \in V\}$. Therefore, the adjacency-level set $N(v, 2)$ is computed in $O(1)$ time by using $\delta \sum_{v \in V} \delta_v = O(nm)$ processors. Since G_a is a subgraph of G, the adjacency-level sets $N_a(x, 1)$ and $N_a(x, 2)$ are computed within the same time and processor bounds, for all $x \in V_a$.

Step 4: This step consists of substeps 4.1, 4.2 and 4.3.

Substep 4.1: Obviously, this substep can be executed in constant time with n processors.

Substep 4.2: The operation of testing whether an edge is an actual edge is executed in constant time with one processor. Here, $m' \leqslant m$ edges are tested, where m' is the number of actual edges of G.

Substep 4.3: In this substep $m'' \leqslant m$ edges are tested, where m'' is the number of semi-free edges of G.

In total, Step 4 is executed in constant time with $\mathrm{O}(nm)$ processors (see Remark 6.1).

Step 5: Here, the logical AND of $\mathrm{O}(n)$ Boolean variables (bits) is computed. This computation takes constant time with $\mathrm{O}(n)$ processors.

From the previous step-by-step analysis, we conclude that the algorithm runs in $\mathrm{O}(1)$ time with $\mathrm{O}(nm)$ processors. Thus, we have the following result.

Theorem 6.2. *Cographs can be recognized in* $\mathrm{O}(1)$ *time by using* $\mathrm{O}(nm)$ *processors on a CRCW PRAM model.*

6.3. Threshold graphs

From Theorem 5.2 it is clear that we can obtain a constant-time recognition algorithm for threshold graphs. This theorem tells us that a graph G is a threshold graph if and only if G and its complement G^c are A-free graphs. Therefore, we obtain the following recognition algorithm:

Step 1. **Compute** the complement G^c of the input graph G;
Step 2. **Test** whether or not both G and G^c are A-free graphs; if so, G is a threshold graph.

The complement of a graph G with n vertices can be computed in constant time with n^2 processors. We have shown that the decision whether or not a graph G is an A-free graph can be made in constant time with $\mathrm{O}(nm)$ processors on a CRCW PRAM model (see Corollary 6.1). Thus, we can present the following result.

Theorem 6.3. *Threshold graphs can be recognized in* $\mathrm{O}(1)$ *time by using* $\mathrm{O}(nm)$ *processors on a CRCW PRAM model.*

Remark 6.1. It is easy to see that all the proposed recognition algorithms can also be executed in $\mathrm{O}(\log n)$ time by using $\mathrm{O}(nm/\log n)$ processors on a Concurrent-Read, Exclusive-Write PRAM (CREW PRAM) model.

7. Conclusions

We have presented parallel recognition algorithms for the classes of perfect graphs known as cographs and threshold graphs. For both classes we have proposed $\mathrm{O}(1)$-time recognition algorithms working with $\mathrm{O}(nm)$ processors on a CRCW PRAM model. This result implies that the following classes of perfect graphs have now constant-time parallel recognition algorithms:

- split graphs [14];
- D-graphs and A-free graphs (this paper, see also [14]);
- cographs (this paper);
- threshold graphs (this paper).

We point out that all the above $\mathrm{O}(1)$-time recognition algorithms are executed with $\mathrm{O}(nm)$ processors. We are currently studying other graph recognition properties and characterizations, still using the edge classification proposed in this paper (see also [14]). We hope our study will enable us to further extend classes of perfect graphs whose members can be recognized in $\mathrm{O}(1)$ parallel time.

References

[1] G.S. Adhar, S. Peng, Parallel algorithms for cographs and parity graphs with applications, J. Algorithms 11 (1990) 252–284.
[2] S. De Agostino, R. Petreschi, Parallel recognition algorithms for graphs with restricted neighborhoods, Internat. J. Found. Comput. Sci. 1 (1990) 123–130.
[3] A. Coppersmith, S. Winogrand, Matrix multiplication via arithmetic progression, J. Symbolic Comput. 9 (1990) 251–280.
[4] V. Chvátal, P.L. Hammer, Set-patching and threshold graphs, Res. Report CORR-73-21, University of Waterloo, 1973.
[5] D.G. Corneil, H. Lerchs, L. Burlingham, Complement reducible graphs, Discrete Appl. Math. 3 (1981) 163–174.

[6] D.G. Corneil, Y. Perl, L.K. Stewart, A linear recognition algorithm for cographs, SIAM J. Comput. 14 (1985) 926–934.
[7] E. Dahlhaus, Efficient parallel recognition algorithms of cographs and distance hereditary graphs, Discrete Appl. Math. 57 (1995) 29–44.
[8] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, Inc., New York, 1980.
[9] F. Harary, Graph Theory, Addison-Wesley, Reading, MA, 1969.
[10] X. He, Parallel algorithms for cographs with applications, in: O. Nurmi, E. Ukkonen (Eds.), Algorithm Theory—SWAT '92, Lecture Notes in Comput. Sci., Vol. 621, Springer, Berlin, 1992, pp. 94–105.
[11] M. Kano, S.D. Nikolopoulos, On the structure of A-free graphs. Part II, TR-25-99, Department of Computer Science, University of Ioannina, 1999.
[12] H. Lerchs, On cliques and kernels, Department of Computer Science, Universtity of Toronto, March 1971.
[13] R. Lin, S. Olariu, An NC recognition algorithm for cographs, J. Parallel Distrib. Comput. 13 (1991) 76–90.
[14] S.D. Nikolopoulos, Constant-time parallel recognition of split graphs, Inform. Process. Lett. 54 (1995) 1–8.
[15] S.D. Nikolopoulos, S.D. Danielopoulos, Parallel computation of perfect elimination schemes using partition techniques on triangulated graphs, Comput. Math. Appl. 29 (1995) 47–57.
[16] E.S. Wolk, The comparability graph of a tree, Proc. Amer. Math. Soc. 13 (1965) 789–795.