A simple linear-time recognition algorithm for weakly quasi-threshold graphs*

Stavros D. Nikolopoulos and Charis Papadopoulos

Department of Computer Science, University of Ioannina
P.O.Box 1186, GR-45110 Ioannina, Greece
{stavros, charis}@cs.uoi.gr

Abstract: Weakly quasi-threshold graphs form a proper subclass of the well-known class of cographs by restricting the join operation. In this paper we characterize weakly quasi-threshold graphs by a finite set of forbidden subgraphs: the class of weakly quasi-threshold graphs coincides with the class of $\{P_4, \text{co-}(2P_3)\}$-free graphs. Moreover we give the first linear-time algorithm to decide whether a given graph belongs to the class of weakly quasi-threshold graphs, improving the previously known running time. Based on the simplicity of our recognition algorithm, we can provide certificates of membership (a structure that characterizes weakly quasi-threshold graphs) or non-membership (forbidden induced subgraphs) in additional $\mathcal{O}(n)$ time. Furthermore we give a linear-time algorithm for finding the largest induced weakly quasi-threshold subgraph in a cograph.

Keywords: weakly quasi-threshold graphs, cographs, forbidden induced subgraphs, recognition, linear-time algorithms.

1 Introduction

The well-known class of cographs is recursively defined by using the graph operations of 'union' and 'join' [4]. Bapat et al. [1], introduced a proper subclass of cographs, namely the class of *weakly quasi-threshold* graphs, by restricting the join operation and studied their *Laplacian spectrum* . In the same work they proposed a quadratic-time algorithm for recognizing such graphs. Here we characterize the class of weakly quasi-threshold graphs by the class of graphs having no P_4 (chordless path on four vertices) or co-$(2P_3)$ (the complement of two disjoint P_3's). This characterization also shows that the complement of a weakly quasi-threshold graph is not necessarily weakly quasi-threshold graph. Moreover we give a tree representation for such graphs, similar to the cotrees for cographs, and propose a linear-time recognition algorithm.

The class of cographs coincides with the class of graphs having no induced P_4 [5]. There are several subclasses of cographs. *Trivially-perfect* graphs, also known as *quasi-threshold* graphs, are characterized as the subclass of cographs having no induced C_4 (chordless cycle on four vertices), that is, such graphs are $\{P_4, C_4\}$-free graphs, and are recognized in linear time [3, 6]. Another interesting subclass of cographs are the $\{P_4, C_4, 2K_2\}$-free graphs known as *threshold* graphs, for which there are several linear-time recognition algorithms [3, 6]. Clearly every threshold graph is trivially-perfect but the converse is not true. Gurski introduced the class of $\{P_4, \text{co-}(2P_3), 2K_2\}$-free graphs in his study of characterizing graphs of certain restricted clique-width [7]. Together with the class of weakly quasi-threshold graphs (that are exactly the class of $\{P_4, \text{co-}(2P_3)\}$-free graphs as

*This research work is co-financed by E.U.-European Social Fund (75%) and the Greek Ministry of Development-GSRT (25%).

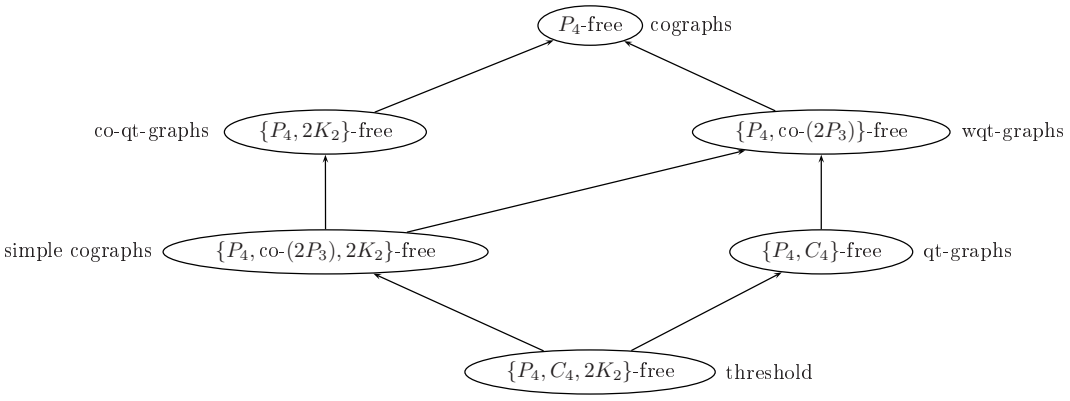

Figure 1: Subclasses of cographs.

we show in this paper), we obtain the inclusion properties for the above families of graphs that we depict in Figure 1.

Firstly, we characterize weakly quasi-threshold graphs by a finite set of forbidden subgraphs. We exhibit properties of the cotree of a weakly quasi-threshold graph similar to those of a cograph. Based on those structural properties we design a simple linear-time recognition algorithm for such graphs. We also show how our algorithm can be extended to support certificates within the same time bound. In case of membership in the weakly quasi-threshold graph class our algorithm provides as certificate a structure for the input graph that characterizes the class. In case where the input graph G is not weakly quasi-threshold graph we show that our algorithm can support an evidence of non-membership by reporting a forbidden subgraph of G. In addition we describe a linear-time algorithm for finding an induced weakly quasi-threshold subgraph of maximum number of vertices in a given cograph using our characterizations.

2 Preliminaries

We consider undirected finite graphs with no loops or multiple edges. For a graph G, we denote its vertex and edge set by $V(G)$ and $E(G)$, respectively, with $n = |V(G)|$ and $m = |E(G)|$. For a vertex subset $S \subseteq E(G)$, the subgraph of G induced by S is denoted by $G[S]$. Moreover, we denote by $G - S$ the graph $G[V(G) - S]$ and by $G - v$ the graph $G[V(G) - \{v\}]$. The complement of a graph G is denoted by $\overline{G}$.

The *neighborhood* $N_G(x)$ of a vertex x of the graph G is the set of all the vertices of G which are adjacent to x. The *closed neighborhood* of x is defined as $N_G[x] = N_G(x) \cup \{x\}$. If $S \subseteq V(G)$, then the neighbors of S, denoted by $N_G(S)$, are given by $\bigcup_{x \in S} N_G(x) - S$.

A vertex x of G is *universal* if $N_G[x] = V(G)$ and is *isolated* if it has no neighbors in G. Two vertices x, y of G are called *false twins* if $N_G(x) = N_G(y)$. A *clique* is a set of pairwise adjacent vertices while an *independent set* is a set of pairwise non-adjacent vertices. A chordless cycle on k vertices is denoted by C_k and a chordless path on k vertices is denoted by P_k. The complement of the graph consisting of two disjoint P_3's is denoted by co-$(2P_3)$.

A graph is *connected* if there is a path between any pair of vertices. A *connected component* of a disconnected graph is a maximal connected subgraph of it. The *largest* induced subgraph refers to the induced subgraph having the maximum number of vertices.

Given two vertex-disjoint graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, their *union* is $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. Their *join* $G_1 + G_2$ is the graph obtained from $G_1 \cup G_2$ by adding all the edges between the vertices of V_1 and V_2.

For a given fixed graph H, any graph is called *H-free* if it does not contain an induced subgraph isomorphic to H. For a set of graphs $\mathcal{H}$, a graph that is H-free for all $H \in \mathcal{H}$ is called *$\mathcal{H}$-free*.

Cographs and cotrees: The class of cographs, also known as *complement reducible graphs*, is defined recursively as follows:

(c1) a single vertex is a cograph;

(c2) if G_1 and G_2 are cographs, then $G_1 \cup G_2$ is also a cograph;

(c3) if G_1 and G_2 are cographs, then $G_1 + G_2$ is also a cograph.

The class of cographs coincides with the class of P_4-free graphs [5].

Along with other properties, it is known that cographs admit a unique tree representation, called a *cotree* [4]. For a cograph G its cotree, denoted by $T(G)$, is a rooted tree having $O(n)$ nodes. The vertices of G are precisely the leaves of $T(G)$ and every internal node of $T(G)$ is labelled by either 0 (0-node) or 1 (1-node). Two vertices are adjacent in G if and only if their least common ancestor in $T(G)$ is a 1-node. Moreover, if G has at least two vertices then each internal node of the tree has at least two children and any path from the root to any node of the tree consists of alternating 0- and 1-nodes. The complement of any cograph G is a cograph and the cotree of the complement of G is obtained from $T(G)$ with inverted labeling on the internal nodes of $T(G)$. Note that we distinguish between vertices of a graph and nodes of a tree. Cographs can be recognized and their cotrees can be computed in linear time [5, 8, 2].

3 A characterization of weakly quasi-threshold graphs

Bapat et al., introduced in [1] the class of *weakly quasi-threshold graphs* (or *wqt graphs* for short) and defined the given class as follows:

(w1) a single vertex is a wqt graph;

(w2) if G_1 and G_2 are wqt graphs then $G_1 \cup G_2$ is a wqt graph;

(w3) if G is a wqt graph then adding a universal vertex in G results in a wqt graph;

(w4) if G is a wqt graph then adding a vertex in G having the same neighborhood with a vertex of G results in a wqt graph.

By definition the class of cographs and wqt graphs have certain similarities. Clearly every wqt graph is a cograph but the converse is not true. Properties c1,c2 and w1,w2 completely coincide, whereas properties w3–w4 correspond to a restricted version of c3. Moreover it follows that in a connected wqt graph there is either a universal vertex or a false twin. Then it is not difficult to see that the class of wqt graphs is closed under taking induced subgraphs, that is, the class of wqt graphs is *hereditary*.

In the next lemma we prove an alternative definition of wqt graphs.

Lemma 3.1. *The class of wqt graphs can be defined recursively as follows:*

(a1) an edgeless graph is a wqt graph;

(a2) if G_1 and G_2 are wqt graphs then $G_1 \cup G_2$ is a wqt graph;

(a3) if G is a wqt graph and H is an edgeless graph then $G + H$ is a wqt graph.

Proof. Properties w2 and a2 are exactly the same. By properties w1 and w2 we have that edgeless graphs are wqt graphs. We need to show that property a3 can substitute both properties w3–w4. If G is a wqt graph and H is an edgeless graph then the graph $G + H$ is obtained by first adding a universal vertex in G and then by the addition of false twins. Hence $G + H$ is a wqt graph.

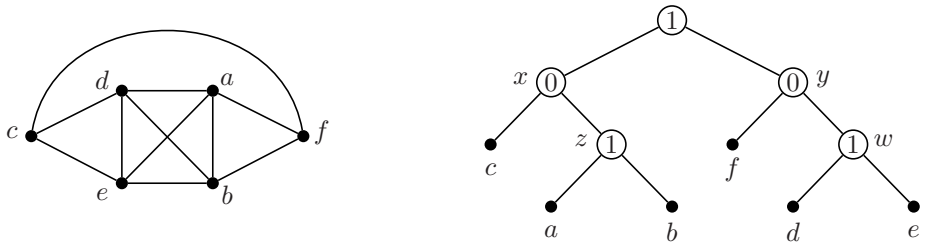

Figure 2: A co-($2P_3$) and its cotree.

For the converse let G be a connected wqt graph. First observe that G can be reduced to a disconnected wqt graph $G[A]$ by repeatedly removing a universal vertex or a false twin vertex. Let S be a set of the removal vertices. Let $x_n, \ldots, x_k$ be an order of S where x_i is either universal or false twin in $G_i = G[\{x_i, \ldots, x_k\} \cup A]$, $n \leq i \leq k$. We show that there is such an order of $\{x_n, \ldots, x_k\}$ where all the false twin vertices appear consecutive. If there is a universal vertex x_j between two false twin vertices x_i and x_k then swapping the positions of x_j and x_k keeps the same property for the resulting order. We apply this operation for every universal vertex between two false twin vertices and obtain an order of the vertices of S where the false twin vertices appear consecutive. Observe that the set of the false twin vertices induces an edgeless graph in G. Thus the join operation between a wqt graph and an edgeless graph is sufficient to construct a connected wqt graph. □

Next we give a characterization of weakly quasi-threshold graphs through forbidden subgraphs.

Theorem 3.2. *A graph G is weakly quasi-threshold if and only if G does not contain any P_4 or* co-($2P_3$) *as induced subgraphs.*

Proof. First observe that a P_4 or a co-($2P_3$) do not have a universal vertex or a false twin. Thus every graph containing a P_4 or a co-($2P_3$) as an induced subgraph is not wqt graph, since the class of wqt graphs is hereditary. Therefore every $\{P_4, \text{co-}(2P_3)\}$-free graph is wqt graph.

For the converse we need to show that every graph that is not wqt contains either a P_4 or a co-($2P_3$) as induced subgraph. It is not difficult to see that any graph on three vertices is a wqt graph. Let G be a non-wqt graph on at least four vertices that does not have a universal or false twin vertex but every connected proper induced subgraph of G has such a vertex. We distinguish two cases: (i) G is not a cograph, and (ii) G is a cograph. For the first case G contains a P_4 by the results of [5].

Now we consider the second case. Since G is a cograph, it admits a cotree $T(G)$. By properties a2 and c2, G must be connected and, thus, the root of $T(G)$ is a 1-node. If the 1-node root of $T(G)$ has at least one leaf-child then G contains a universal vertex. We thus assume that the root of $T(G)$ has no leaf-children. By the properties of the cotree, the root of $T(G)$ has at least two children, say x and y, both being 0-nodes. Note that both x and y have at least two children in $T(G)$. If one of x or y has only leaves as children in $T(G)$ then those vertices are false twin and induce an edgeless graph. Thus both x and y have at least one non-leaf child, say z and w, respectively. Observe that z and w are 1-nodes, since 0- and 1-nodes alternate in each path in $T(G)$. Now there exist three vertices a, b, c in G corresponding to the subtree rooted at x such that: a and b are adjacent corresponding to the subtree rooted at z and vertex c non-adjacent to both a and b corresponding to the subtree rooted at x; see Figure 2 for a schematic illustration. It suffices to consider three more vertices corresponding to the subtree rooted at y with similar properties. Those six vertices induce a co-($2P_3$) in G and thus it completes the proof. □

4 A linear-time recognition algorithm

In this section we give a linear-time algorithm for deciding whether an arbitrary graph is wqt. Our basic idea relies on the linear-time construction of the cotree [5] and then test the characterization of Theorem 3.2 translated into the structural properties of the cotree.

Let G be the input graph. We first apply the linear-time recognition algorithm for checking whether G is a cograph [5]. If G is not a cograph then we know that G is not a wqt graph as it contains a P_4. Otherwise G admits a cotree $T(G)$ that can be constructed in linear time [5, 8]. Now it suffices to efficiently check an induced co-$(2P_3)$ on G by using the cotree $T(G)$. For that purpose, we modify $T(G)$ and obtain T^* from $T(G)$ by applying the following two operations:

(i) delete the subtree rooted at a 0-node having only leaves as children;

(ii) remove a leaf that has 1-node as parent.

Next we check if every 1-node in T^* has at most one child. In case of an affirmative answer we output that G is a wqt graph; otherwise, we output that G is not a wqt graph.

Correctness of the algorithm is based on the following lemma.

Lemma 4.1. *Let G be a cograph and let T^* be its modified cotree. Then G is wqt graph if and only if every 1-node of T^* has at most one child.*

Proof. By Theorem 3.2 we need to prove that a 1-node of T^* has at least two children if and only if G has an induced co-$(2P_3)$. If there exists such a 1-node in T^* then it has at least two 0-nodes, say x and y, as children in $T(G)$ since we only remove leaves from a 1-node. By operation (i) both nodes x and y have at least one non-leaf child. Such a child is a 1-node in $T(G)$, since x and y are 0-nodes. This then shows an induced co-$(2P_3)$ in G because every internal node of $T(G)$ has at least two children.

For the converse, let T^* be the modified tree for which every internal 1-node has at most one child. We need to prove that in that case G is a wqt graph. We doing so by showing that adding vertices corresponding to each of the two modified operations results in a wqt graph. Notice that if there exists a 1-node in T^* then its possible child corresponds to a non-edgeless graph. If operation (i) is applied then adding a set of vertices that induces an edgeless graph or being false twins results in a wqt graph by properties (w2)–(w3). If operation (ii) applies then adding a universal vertex results in a wqt graph by property (w3). Therefore G is a wqt graph and we conclude the proof. □

Next we show the running time of the above algorithm.

Theorem 4.2. *Weakly quasi-threshold graphs can be recognized in $\mathcal{O}(n+m)$ time.*

Proof. A wqt-graph recognition algorithm is described above; its correctness follows by Theorem 3.2 and Lemma 4.1. We next show that the algorithm can be implemented in time linear in the size of the input graph G. Cographs can be recognized and their cotrees can be computed in linear time [5]. Since the number of nodes in the cotree $T(G)$ is $\mathcal{O}(n)$, it follows that the modified tree T^* can be obtained in $\mathcal{O}(n)$ time by traversing $T(G)$ from the leaves to the root. Moreover checking the number of children for every internal node of T^* takes $\mathcal{O}(n)$ time, since the number of nodes in T^* may only decrease. Therefore the overall running time of the aforementioned algorithm is bounded by the running time of the cograph recognition and the cotree construction, that is, $\mathcal{O}(n+m)$. □

Let us now show that if the input graph G is not wqt graph then we can provide in the same running time a set of vertices that induces either a P_4 or a co-$(2P_3)$ in G. In case G is not a cograph an induced P_4 is given as an output by the results of [5]. So we next proceed by assuming that G is a cograph.

After constructing the cotree $T(G)$ of G, we assign pointers to the vertices of G corresponding to the leaves of $T(G)$. For every internal node u of $T(G)$ we assign pointers to the leaves of

the subtree rooted at u. The latter assignment can be done in $\mathcal{O}(n)$ as there are $\mathcal{O}(n)$ nodes in $T(G)$. Constructing T^* from $T(G)$ can be done as follows. Every time we remove a subtree we do not actually remove it from the data structure but we maintain it as a single tree rooted at the node/vertex that is removed from $T(G)$. Thus after constructing T^* we maintain a forest of disjoint cotrees where each root of a cotree points to the node of T^* that used to be its parent in $T(G)$.

Since G is not wqt graph, there must be a 1-node in T^* having at least two children by Lemma 4.1. Let u and v be two of its children in T^*. By the construction of T^* both u and v are 0-nodes having at least two children in T^*; for otherwise, they would have been deleted from $T(G)$. Moreover in T^* there is at least one child of x that is a non-leaf and, thus, a 1-node. Let t_u be a 1-node child of u and let t_v be a 1-node child of v. In T^* nodes t_u and t_v may have no child but by definition in $T(G)$ they have at least two children. Thus looking at the children of t_u and using their pointers to the other cotrees we find two vertices that belong in two different subtrees rooted as children of t_u. Let a, b and d, e be such vertices of the subtrees rooted at t_u and t_v, respectively. Furthermore let c and f be two vertices of the subtrees of u and v not belonging to the subtrees rooted at t_u and t_v, respectively (note that both c and f exist by the properties of a cotree). Then the set $\{a, b, c, d, e, f\}$ induces a co-$(2P_3)$ in G. By keeping a simple data structure for T^* we need $\mathcal{O}(n)$ time for finding the required set. Therefore in the same running time of our algorithm for recognizing wqt graph we either output the modified tree T^* for G having the property described in Lemma 4.1 or report an induced P_4 or co-$(2P_3)$ of G.

Theorem 4.3. *Given a graph G there is an $\mathcal{O}(n+m)$ algorithm that reports "*`yes`*" if G is weakly quasi-threshold graph or either an induced P_4 or* co-$(2P_3)$ *of G otherwise.*

Largest weakly quasi-threshold graphs in cographs: As already mentioned every wqt graph is a cograph but the converse is not necessarily true. With the following theorem we show that the problem of removing the minimum number of vertices from a cograph so that the resulting graph is wqt can be done in linear time. Note that the proposed algorithm can serve as a recognition algorithm as well.

Theorem 4.4. *Given a cograph G there is an $\mathcal{O}(n+m)$ algorithm that finds a largest induced weakly quasi-threshold subgraph of G.*

Proof. We first describe such an algorithm. Let $T(G)$ be the cotree of G and let T^* be the modified cotree. For a node u of $T(G)$ we define the following parameters with respect to the induced subgraph H_u of G corresponding to the leaves of the subtree rooted at u: (i) the number of vertices of H_u, denoted by $n(u)$, (ii) the maximum clique of H_u, denoted by $\mathrm{MC}(u)$, and (iii) the maximum independent set of H_u, denoted by $\mathrm{MI}(u)$. For every node u those values are denoted by a triple $(n(u), \mathrm{MC}(u), \mathrm{MI}(u))$.

Our algorithm starts by traversing both $T(G)$ and T^* from the leaves to the root and computes for each node of $T(G)$ a largest induced wqt subgraph; the one computed at the root of $T(G)$ provides the largest induced wqt subgraph of G. The computed graph is represented by a cotree T' that we construct during the traversal of $T(G)$. Furthermore every time the algorithm visits a node u of $T(G)$ it computes the triple $(n(u), \mathrm{MC}(u), \mathrm{MI}(u))$. As an initialization step we let $(1, \{v\}, \{v\})$ be the triple of each leaf v in $T(G)$. Let u be an internal node of $T(G)$ and let $u_1, u_2, \ldots, u_k$ be the children of u in $T(G)$. If u is a 0-node then the algorithm assigns to u the triple $(\sum n(u_i), \mathrm{MC}(u_i'), \bigcup \mathrm{MI}(u_i))$ where u_i' is the child having the maximum $|\mathrm{MC}(u_i)|$, and copies node u in T'. If u is a 1-node and u has at most one child in T^* then according to Lemma 4.1 we do not modify its subtree which means that we assign to u the triple $(\sum n(u_i), \bigcup \mathrm{MC}(u_i), \mathrm{MI}(u_i'))$ where u_i' is the child having the maximum $|\mathrm{MI}(u_i)|$, and we copy u in T'. We are left with the case that u is a 1-node and u has at least two children in T^*. In such a case we need to modify the subtree rooted at u. Let $u_1^*, u_2^*, \ldots, u_\ell^*$ be the children of u in T^*; note that each child u_1^* is a 0-node, $1 \leq i \leq \ell$. Based on Lemma 4.1 we modify every subtree in $T(G)$ rooted at u_i^* except that

u_p^* having the maximum value among $\min\{n(u_i^*) - |\mathrm{MC}(u_i^*)|, n(u_i^*) - |\mathrm{MI}(u_i^*)|\}$. For every other node $u_j^* \neq u_p^*$ we do the following operations: if $|\mathrm{MC}(u_j^*)| > |\mathrm{MI}(u_j^*)|$ then we delete the subtree rooted at u_j^* and add the vertices of $\mathrm{MC}(u_j^*)$ as children of u; otherwise we remove the nodes of the subtree rooted at u_j^* and add the vertices of $\mathrm{MI}(u_j^*)$ as children of u_j^*. In the last case we assign to u_j^* the triple $(|\mathrm{MI}(u_j^*)|, \{v\}, \mathrm{MI}(u_j^*))$, where $v \in \mathrm{MI}(u_j^*)$. The modifications are applied in T' by copying u with its new children and the assigned triple for u is exactly the same as in the previous case for the 1-node with respect to the triples of its new children.

For the correctness of the algorithm we use Lemma 4.1. Observe first that each triple $(n(u), \mathrm{MC}(u), \mathrm{MI}(u))$ of a node u is corrected computed according to the results in [4]. Clearly the computed tree T' is by definition a cotree. Note also that in the modified tree of T' every 1-node has at most one child by construction since every modified subtree in $T(G)$ corresponds to an independent set or a clique meaning that such nodes are deleted in the modified tree of T'. Hence the computed subgraph is a wqt graph by Lemma 4.1. Furthermore notice that the algorithm removes vertices only if there is a 1-node u having at least two children in T'. By Lemma 4.1 at most one child must remain in T'. Let u_p^* be such a child. For every child $u_j^* \neq u_p^*$ of u we need to make the corresponding subgraph either an independent set or a clique since by property (a3) of Lemma 3.1 only edgeless subgraphs or universal vertices can be added to the subgraph corresponding to u_p^*. Thus the maximum independent set or clique minimizes the number of the removal vertices in every subgraph corresponding to a node u_j^*. For every child u_i^* of u in T^* let $k_i = \max\{|\mathrm{MC}(u_i^*)|, |\mathrm{MI}(u_i^*)|\}$. The minimum number of vertices that needs to be removed from each subgraph corresponding to a node u_i^* is given by $n(u_i^*) - k_i$. Observe that the algorithm chooses the node u_p^* with the maximum value $n(u_i^*) - k_i$. Assume that a node $u_q^* \neq u_p^*$ gives a largest subgraph, that is, $\sum_{i \neq q}(n(u_i^*) - k_i) < \sum_{i \neq p}(n(u_i^*) - k_i)$. But then we obtain that $n(u_p^*) - k_p < n(u_q^*) - k_q$ which is contradiction since we pick node u_p^* having the maximum such value. Therefore the computed graph for T' is the largest induced wqt subgraph of G.

Regarding the running time of the described algorithm, note that the number of nodes of $T(G)$ and T^* is $\mathcal{O}(n)$ meaning that all steps can be executed in $\mathcal{O}(n)$ time. Constructing the cotree $T(G)$ and the modified tree T^* takes $\mathcal{O}(n+m)$ time which concludes the proof. □

5 Concluding remarks

We characterized weakly quasi-threshold graphs by two forbidden induced subgraphs, namely P_4 and co-$(2P_3)$. Based on this characterization we proposed a simple linear-time recognition algorithm for such graphs. We also showed that the proposed algorithm can be extended to provide certificates of non-membership in linear time. As an open question we state an efficient fully dynamic recognition algorithm for the class of weakly quasi-threshold graphs (especially for edge modifications since vertex additions have the property that a vertex v can be added in a wqt graph G if and only if v is either universal or v has a false twin in a connected component of G).

References

[1] R.B. Bapat, A.K. Lal, and S. Pati. Laplacian spectrum of weakly quasi-threshold graphs. *Graphs and Combinatorics*, 24:273–290, 2008.

[2] A. Bretscher, D. Corneil, M. Habib, and C. Paul. A simple linear time LexBFS cograph recognition algorithm. *SIAM Journal on Discrete Mathematics*, 22:1277–1296, 2008.

[3] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey*. SIAM Monographs on Discrete Mathematics and Applications, 1999.

[4] D.G. Corneil, H. Lerchs, and L.K. Stewart. 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 Journal on Computing*, 14:926–934, 1985.

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

[7] F. Gurski. Characterizations for co-graphs defined by restricted NLC-width or clique-width operations. *Discrete Mathematics*, 306:271–277, 2006.

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