Dynamic Distance Hereditary Graphs Using Split Decomposition*

Emeric Gioan and Christophe Paul**

CNRS - LIRMM, Université de Montpellier 2, France

Abstract. The problem of maintaining a representation of a dynamic graph as long as a certain property is satisfied has recently been considered for a number of properties. This paper presents an optimal algorithm for this problem on vertex-dynamic connected distance hereditary graphs: both vertex insertion and deletion have complexity $O(d)$, where d is the degree of the vertex involved in the modification. Our vertex-dynamic algorithm is competitive with the existing linear time recognition algorithms of distance hereditary graphs, and is also simpler. Besides, we get a constant time edge-dynamic recognition algorithm. To achieve this, we revisit the split decomposition by introducing graph-labelled trees. Doing so, we are also able to derive an intersection model for distance hereditary graphs, which answers an open problem.

1 Introduction

Motivated by their practical applications as well as by their related theoretical challenges, dynamic graph algorithms have received particular attention over the last few years [12]. Solving a problem on a dynamic graph consists of an algorithm that, under a series of graph modifications (vertex or edge modification), updates a data structure supporting elementary queries (e.g. adjacency). Let us note that the series of modifications to which the graph is submitted is not known in advance. To be of interest, such an algorithm should not recompute a solution from scratch. In order to ensure locality of the computation, most of the known dynamic graph algorithms are based on decomposition techniques. For example, the SPQR-tree data structure has been introduced in order to dynamically maintain the 3-connected components of a graph which allows on-line planarity testing [2].

This paper considers the *dynamic representation problem* which asks for the maintenance of a representation of a dynamic graph as long as a certain property Π is satisfied. Existing literature on this problem includes representation of chordal graphs [18], proper interval graphs [16], cographs [21], directed cographs [7], permutation graphs [8]. The data structures used for the last four results are strongly related to the modular decomposition tree [14]. The *split decomposition* (also called 1-join decomposition), introduced by Cunningham [9],

* Research supported by the French ANR project "*Graph Decompositions and Algorithms* (GRAAL)".

** Research conducted while C. Paul was on Sabbatical at School of Computer Science, McGill University, Montréal, Canada.

T. Tokuyama (Ed.): ISAAC 2007, LNCS 4835, pp. 41–51, 2007.

is a generalization of the modular decomposition. A natural question is to ask whether the split decomposition can be used to dynamically represent wider graph families? We answer positively to this question.

The algorithmic aspects of the split decomposition, unlike the modular decomposition, are not well understood. For instance, totally decomposable graphs are known to be the *distance hereditary graphs* [1,15], which form an interesting family of graphs for several reasons: they generalize the well-known cographs [5], which are totally decomposable by the modular decomposition; they are the graphs of rankwidth 1 [20] and are among the elementary graphs of clique-width 3 [4]; they also have various theoretical characterizations... Computing the split decomposition in linear time [10] is very complicated. It follows that most of the known algorithms (even recent ones) operating on distance hereditary graphs do not rely on the split decomposition but rather on a heavy breadth-first search layering characterization [1], or on some ad-hoc (rooted) tree decompositions [3,17,23]. Similarly the recent $O(1)$ edge-only dynamic recognition algorithm [6] is based on the BFS layering characterization.

In this paper, we revisit the split decomposition theory [9] under the new framework of *graph-labelled trees*, which formalize the (unrooted) tree decomposition underlying the split decomposition (see Section 2). As a by-product, we can derive two new characterizations of distance hereditary graphs (see Section 3). The first one is an intersection model for the family of distance hereditary graphs. The existence of such a model was known [19], but to our knowledge, the model itself was still not discovered [22]. Graph-labelled trees also yield an incremental characterization of distance hereditary graphs from which we can deduce an optimal vertex-dynamic algorithm to represent distance hereditary graphs (see Section 4). These are the first results obtained for the split decomposition in the framework of graph-labelled trees. Moreover the simplicity of the solutions we propose witnesses the elegance of graph-labelled trees: e.g. our $O(d)$ vertex insertion algorithm is not only competitive with the existing static linear time recognition algorithms [3,11,15], but is also simpler. We also get a constant time edge-dynamic recognition algorithm, different from [6] (see Section 5). We believe that new applications or generalizations of the split decomposition could be derived from graph-labelled trees.

2 Split Decomposition and Graph-Labelled Trees

Any graph $G = (V(G), E(G))$ we consider is simple and loopless. For a subset $S \subseteq V(G)$, $G[S]$ is the subgraph of G induced by S. If T is a tree and S a subset of leaves of T, then $T(S)$ is the smallest subtree of T spanning the leaves of S. If x is a vertex of G then $G - x = G[V(G) - \{x\}]$. Similarly if $x \notin V(G)$, $G + (x, S)$ is the graph G augmented by the new vertex x adjacent to $S \subseteq V(G)$. We denote $N(x)$ the neighbourhood of a vertex x. The neighborhood of a set $S \subseteq V(G)$ is $N(S) = \{x \notin S \mid \exists y \in S,\ xy \in E(G)\}$. The *clique* is the complete graph and the star is the complete bipartite graph $K_{1,n}$. The universal vertex of the star is called its *centre* and the degree one vertices its *extremities*.

Definition 2.1. [9] *A* split *of a graph G is a bipartition (V_1, V_2) of $V(G)$ such that 1) $|V_1| \geqslant 2$ and $|V_2| \geqslant 2$; and 2) every vertex of $N(V_1)$ is adjacent to every vertex of $N(V_2)$.*

Cliques and stars are called *degenerate* since for any such graph on at least 4 vertex, any non-trivial vertex bipartition is a split. A graph with no split that is not degenerate is called *prime*. The split decomposition of a graph G, as originally studied in [9], consists of: finding a split (V_1, V_2), decomposing G into $G_1 = G[V_1 \cup \{x_1\}]$, with $x_1 \in N(V_1)$ and $G_2 = G[V_2 \cup \{x_2\}]$ with $x_2 \in N(V_2)$, x_1 and x_2 being called the *marker vertices*; and then recursing on G_1 and G_2. In [9], Cunningham presents the idea of a tree decomposition. But its main result stating the uniqueness of lastly resulting graphs in a split decomposition focuses on the set of resulting graphs more than on the structure linking them together. To reformulate Cunningham's result in terms of tree, let us introduce some terminology.

Definition 2.2. *A* graph-labelled tree $(T, \mathcal{F})$ *is a tree in which any node v of degree k is labelled by a graph $G_v \in \mathcal{F}$ on k vertices such that there is a bijection ρ_v from the tree-edges incident to v to the vertices of G_v.*

We call *nodes* the internal vertices of a tree, and *leaves* the other ones. Let $(T, \mathcal{F})$ be a graph-labelled tree and l be a leaf of T. A node or a leaf u different from l is *l-accessible* if for any tree-edges $e = wv$ and $e' = vw'$ on the l, u-path in T, we have $\rho_v(e)\rho_v(e') \in E(G_v)$. By convention, the unique neighbor of l in T is also l-accessible. See Figure 1 for an example.

Definition 2.3. *The* accessibility graph $G(T, \mathcal{F})$ *of a* graph-labelled tree $(T, \mathcal{F})$ *has the leaves of T as vertices, and there is an edge between x and y if and only if y is x-accessible.*

Lemma 2.1. *Let $(T, \mathcal{F})$ be a graph-labelled tree and T_1, T_2 be the subtrees of $T - e$ where e is a tree-edge non-incident to a leaf. Then the bipartition (L_1, L_2) of the leaves of T, with L_i being the leaf set of T_i, defines a split in the graph $G(T, \mathcal{F})$.*

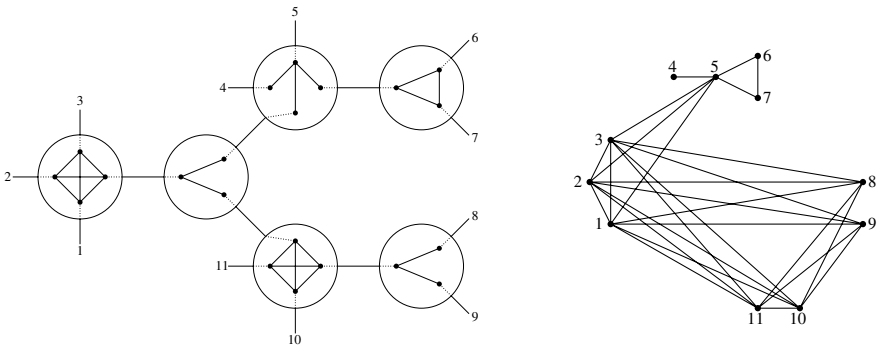

Fig. 1. A graph-labelled tree and its accessibility graph

We can naturally define the *split* and the converse *join* operations on a graph-labelled tree $(T, \mathcal{F})$ as follows (see Figure 2):

- *Split of* $(T, \mathcal{F})$*:* Let v be a node of T whose graph G_v has a split (A, B). Let G_A and G_B be the subgraphs resulting from the split (A, B) and a, b be the respective marker vertices. Splitting the node v consists in substituting v by two adjacent nodes v_A and v_B respectively labelled by G_A and G_B such that for any $x \in V(G_A)$ different from a, $\rho_{v_A}(x) = \rho_v(x)$ and $\rho_{v_A}(a) = v_A v_B$ (similarly for any $x \in V(G_B)$ different from b, $\rho_{v_B}(x) = \rho_v(x)$ and $\rho_{v_B}(b) = v_A v_B$).
- *Join of* $(T, \mathcal{F})$*:* Let uv be a tree-edge of T. Then joining the nodes u and v consists in substituting them by a single node w labelled by G_w the accessibility graph of the tree with only edge u, v and respective labels G_u and G_v. For any vertex x of G_w, $\rho_w(x) = \rho_v(x)$ or $\rho_w(x) = \rho_u(x)$ depending on which graph x belonged.

Observe that if $(T, \mathcal{F})$ is obtained from $(T', \mathcal{F}')$ by a join or a split operation, then it follows from the definitions that $G(T, \mathcal{F}) = G(T', \mathcal{F}')$.

Among the join operations, let us distinguish: the *clique-join*, operating on two neighboring nodes labelled by cliques, and the *star-join*, operating on star-labelled neighboring nodes u, v such that the tree-edge uv links the centre of one star to an extremity of the other. A graph-labelled tree $(T, \mathcal{F})$ is *reduced* if neither clique-join nor star-join can be applied, i.e. the clique nodes are pairwise non-adjacent and two star nodes u and v can be adjacent only if $\rho_u(uv)$ and $\rho_v(uv)$ are both centres or both extremities of their respective stars G_u and G_v.

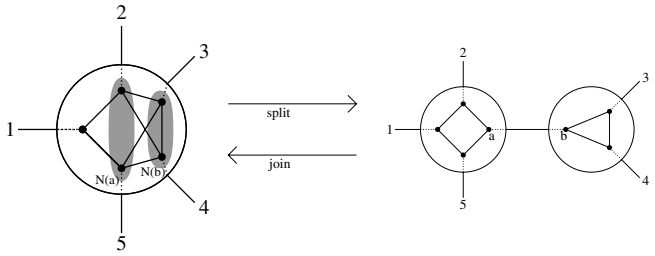

Fig. 2. The split and the join operations on a graph-labelled tree

Theorem 2.1 (Cunningham's Theorem reformulated). *For any connected graph G, there exists a unique reduced graph-labelled tree $(T, \mathcal{F})$ such that $G = G(T, \mathcal{F})$ and any graph of $\mathcal{F}$ is prime or degenerate.*

For a connected graph G, the *split tree* $ST(G)$ of G is the unique reduced graph-labelled tree $(T, \mathcal{F})$ in the above Theorem 2.1 (see Figure 1).

The next two lemmas are central to proofs of further theorems.

Lemma 2.2. *Let $ST(G) = (T, \mathcal{F})$ be the split tree of a connected graph G. Let l be a leaf of T, and $e = uv$, $e' = uv'$ be distinct tree edges such that u is a*

l-accessible and e is on the u, l path in T. Then $\rho_u(e)\rho_u(e') \in E(G_u)$ if and only if there exists a l-accessible leaf l' in the subtree of $T - e'$ containing v'.

The above easy lemma can be rephrased as follows: if u and v are two adjacent l-accessible nodes, then there exists a l-accessible leaf l' such that the l, l'-path contains the tree edge u, v. It follows that:

Lemma 2.3. *Let $ST(G) = (T, \mathcal{F})$ be the split tree of a connected graph G. For any vertex $x \in V(G)$, $T(N(x))$ has at most $2.|N(x)|$ nodes.*

3 Characterizations of Distance Hereditary Graphs

A graph is *distance hereditary* (DH for short) if the distance between any given pair of vertices remains the same in any connected induced subgraphs. By [15], a graph is DH if and only if it is totally decomposable by the split decomposition, i.e. its split tree is labelled by cliques and stars. Hence DH graphs are exactly accessibility graphs of clique-star labelled trees, *clique-star trees* for short. Among the possible clique-star trees, the split tree is the unique reduced one. Figure 1 gives an example. We mention that ternary clique-star trees were used in [13] to draw DH graphs. Another general example is given by cographs, which can be characterized as distance hereditary graphs whose stars in the split tree are all directed towards a root of the tree.

An intersection model. Given a family S of sets, one can define the intersection graph $\mathcal{I}(S)$ as the graph whose vertices are the elements of S and there is an edge between two elements if and only if they intersect. Many restricted graph families are defined or characterized as the intersection graphs (e.g. chordal graphs, interval graphs... see [19]). Graph families supporting an intersection model can be characterized without even specifying the model [19]. This result applies to DH graphs, and no model was known [22]. Based on clique-star trees, an intersection model can be easily derived. Note that it can be equivalently stated by considering only reduced clique-star trees, or even only ternary ones. We call *accessibility set* of a leaf l in a graph-labelled tree the set of pairs $\{l, l'\}$ with l' a l-accessible leaf.

Theorem 3.1 (Intersection model). *A graph is distance hereditary if and only if it is the intersection graph of a family of accessibility sets of leaves in a set of clique-star trees.*

Incremental characterization. Let G be a connected DH graph and let $ST(G) = (T, \mathcal{F})$ be its split tree. Given a subset S of $V(G)$ and $x \notin V(G)$, we want to know whether the graph $G + (x, S)$ is DH or not.

Definition 3.1. *Let $T(S)$ be the smallest subtree of T with leaves S. Let u be a node of $T(S)$.*

1. *u is* fully-accessible *if any subtree of $T - u$ contains a leaf $l \in S$;*

2. *u is* singly-accessible *if it is a star-node and exactly two subtrees of $T - u$ contain a leaf $l \in S$ among which the subtree containing the neighbor v of u such that $\rho_u(uv)$ is the centre of G_u;*
3. *u is* partially-accessible *otherwise.*

We say that a star node is *oriented towards* an edge (or a node) of T if the tree-edge mapped to the centre of the star is on the path between the edge (or node) and the star.

Theorem 3.2 (Incremental characterization). *Let G be a connected distance hereditary graph and $ST(G) = (T, \mathcal{F})$ be its split tree. Then $G + (x, S)$ is distance hereditary if and only if:*

1. *at most one node of $T(S)$ is partially accessible;*
2. *any clique node of $T(S)$ is either fully or partially-accessible.*
3. *if there exists a partially accessible node u, then any star node $v \neq u$ of $T(S)$ is oriented towards u if and only if it is fully accessible. Otherwise, there exists a tree-edge e of $T(S)$ towards which any star node of $T(S)$ is oriented if and only if it is fully accessible.*

4 A Vertex-Only Fully-Dynamic Recognition Algorithm

In this section we mainly compute the characterization given by Theorem 3.2 to obtain the following main result.

Main Theorem 4.1. *There exists a fully dynamic recognition algorithm for connected distance hereditary graphs with complexity $O(d)$ per vertex insertion or deletion operation involving d edges.*

The vertex insertion algorithm yields a linear time recognition algorithm of (static) DH graphs, thereby achieving the best known bound but also simplifying the previous non-incremental ones [15,11,3].

Corollary 4.1 (Static recognition). *The vertex insertion routine enables to recognize distance hereditary graphs in linear time.*

The following data structure is used for the clique-star tree $ST(G) = (T, \mathcal{F})$ of the given DH graph G: a (rooted) representation of the tree T; a single *clique-star marker* distinguishing the type of each node; a *centre-marker* distinguishing the centre of each star node; and the degree of each node. Let us notice that this data structure is an $O(n)$ space representation of the DH graphs on n vertices.

4.1 Vertex Insertion Algorithm

Computing the smallest subtree spanning a set of leaves. Given a set S of leaves of a tree T, we need to identify the smallest subtree $T(S)$ spanning S, and to store the degrees of its nodes. This problem is easy to solve on rooted (or directed) trees by a simple bottom-up marking process with complexity $O(|T(S)|)$. So an arbitrary root of T is fixed.

1. Mark each leaf of S. A node is *active* if its father is not marked.
2. Each marked node marked its father as long as: 1) the root is not marked and there is more than one active node, or 2) the root is marked and there is at least one active node.
3. As long as the root of the subtree induced by the marked nodes is a leaf not in S, remove this node and check again.

Testing conditions of Theorem 3.2. The first two conditions of Theorem 3.2 are fairly easy to check by following Definition 3.1: a node u is fully-accessible if its degrees in $T(S)$ and T are the same; u is singly-accessible if it is a star, if it has degree 2 in $T(S)$ and if the centre neighbor belongs to $T(S)$; and u is partially accessible otherwise, such a node having to be unique if it exists. This test costs $O(|T(S)|)$.

We now assume that the first two conditions of Theorem 3.2 are fulfilled. At first, the case is trivial if $|S| = 1$. So assume $|S| > 1$.

We define *local orientations* on vertices of a tree as the choice, for every vertex u, of a vertex $f(u)$ such that either $f(u) = u$ or $f(u)$ is a neighbor of u. Local orientations are called *compatible* if 1) $f(u) = u$ implies $f(v) = u$ for every neighbor v of u, and 2) $f(u) = v$ implies $f(w) = u$ for every neighbor $w \neq v$ of u. It is an easy exercise to see that if local orientations are compatible then exactly one of the two following properties is true: either there exists a unique vertex u with $f(u) = u$, in this case u is called *node-root*, or there exists a unique tree-edge uv with $f(u) = v$ and $f(v) = u$, in this case uv is called *edge-root*.

The test for the third condition of Theorem 3.2 consists of building, if possible, suitable compatible local orientations in the tree $T(S)$:

1. Let u be a leaf of $T(S)$. Then $f(u)$ is the unique neighbor of u.
2. Let u be a star node of $T(S)$. If u is partially-accessible, then $f(u) = u$. If u is singly-accessible, then $f(u) = v$ with v the unique neighbor v of u belonging to $T(S)$ such that $\rho_u(uv)$ is an extremity of the star. If u is fully-accessible, then $f(u) = v$ with v the neighbor of u such that $\rho_u(uv)$ is the centre of the star.
3. Let u be a clique node of $T(S)$. If u is partially-accessible, then $f(u) = u$. Otherwise, u is fully-accessible and its neighbors are leaves or star nodes. If $f(v) = u$ for every neighbor v of u then $f(u) = u$. If $f(v) = u$ for every neighbor v of u but one, say w, then $f(u) = w$. Otherwise u is an *obstruction*.

The third condition of Theorem 3.2 is satisfied if and only if 1) there is no obstruction and 2) local orientations of $T(S)$ are compatible. This test can be done in time $O(|T(S)|)$ by a search of $T(S)$. Hence, the conditions of Theorem 3.2 can be tested in $O(|T(S)|)$ time.

Updating the split-tree. We now assume that graph $G + (x, S)$ is DH. So by Theorem 3.2 the split tree has either a unique node-root or a unique edge-root. There are three cases (see Figure 3).

1. There is a node-root u being partially-accessible, or S is reduced to a unique vertex u. We may have to make a first update of T by splitting the node u

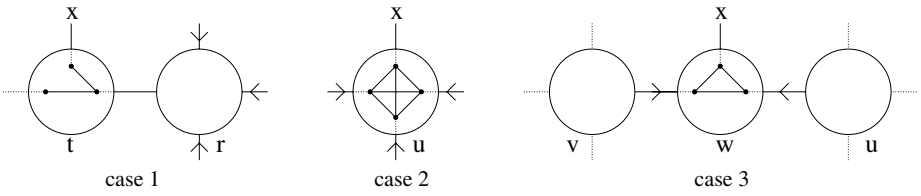

Fig. 3. Vertex insertion

under some conditions on the degrees. Let U, resp. A, be the set of tree-edges adjacent to u in T, resp. in $T(S)$.

(a) If u is a clique node with $|U \setminus A| \geq 2$, then u is replaced with an edge vw in T. Then v, resp. w, is labelled by a clique whose vertices correspond to A, resp. $U \setminus A$, except one which corresponds to vw. In this case, v is now the node-root.

(b) If u is a star node with centre mapped to the tree edge e and $|(U \setminus A) \setminus e)| \geq 1$, then u is replaced with an edge vw in T. Then v is labelled by a star whose extremities correspond to $A \setminus \{e\}$ and centre to vw (we have $|A \setminus \{e\}| > 1$ since u is not singly accessible), and w is labelled by a star whose extremities correspond to $(U \setminus A) \cup \{vw\}$ and centre to e. If $e \in A$, then the edge vw is now the edge-root. And if $e \notin A$, then the node v becomes the node-root.

If the tree still has a node-root $r = u$ or $r = v$, then let s be its neigbhor in T that does not belong to $T(S)$. Then the insertion edge is $e = rs$, and $ST(G + (x, S))$ is obtained by subdividing rs with a star node t of degree 3 whose centre is $\rho_w(rt)$ and making the leaf x adjacent to t. Then, in the case where s is a star with centre $\rho_v(st)$, we proceed a join operation on the tree-edge st .

2. There is a node-root u not being partially-accessible, Then u is a clique node, and $ST(G+(x, S))$ is obtained by adding the new leaf x adjacent to u whose degree thereby increases by one.
3. There is an edge-root uv. Then $ST(G + (x, S))$ is obtained by subdividing uv with a clique node w of degree 3 and making the leaf x adjacent to w.

Each above update operation can be done in $O(1)$ time, except the splitting in case 1 which requires $O(|T(S)|)$ time (by deleting A from u to get w, and adding A to a new empty node v). Any other update operation requires $O(1)$ time to maintain the data structure of the split tree (artificial root, degrees...). Then, the complexity for the whole insertion algorithm derives from from previous steps and the fact that $O(|T(S)|) = O(|S|)$ (Lemma 2.3).

Theorem 4.2 (Vertex insertion). *Let $G + (x, S)$ be a graph such that G is a connected distance hereditary graph. Given the data structure of split tree $ST(G)$, testing whether $G + (x, S)$ is distance hereditary and if so computing the data structure of $ST(G + (x, S))$ can be done in $O(|S|)$ time.*

4.2 Vertex Deletion Algorithm

Removing a vertex x from a distance hereditary graph G always yields a distance hereditary graph $G - x$. Let $ST(G)$ be the split tree of G. Updating the data structure of the split tree can be done as follows.

1. Remove the leaf x and update the degree of its neighbor v.
2. If v now has degree 2, then replace v with an edge between its neighbors, and, if needed, reduce the clique-star tree on this edge.
3. If v is a star node whose centre neighbor was x, then $G - x$ is no longer connected, and the split-trees of each connected component are the components of $T - \{v, x\}$.

It is easy to see that any operation costs $O(1)$ except the join operation which costs $min(d', d'')$ where d', d'' are degree of the concerned nodes. Since at least one of these nodes is fully accessible, this minimum degree is lower than d. Hence this join operation costs $O(d)$.

Lemma 4.1 (Vertex deletion). *Let G be a connected distance hereditary graph and x be a degree d vertex of G. Given the data structure of split tree $ST(G)$, testing whether $G-x$ is a connected distance hereditary graph and if so computing the data structure of $ST(G - x)$ can be done in $O(d)$ time.*

5 Other Applications and Concluding Remarks

The results in this Section, and the previous ones, will be fully developped, with proofs (omitted here for lack of space), in a forthcoming paper.

A constant time edge-only fully-dynamic recognition algorithm. We consider the problem of adding or deleting an edge to a connected distance hereditary graph and testing if the new graph is distance hereditary. We use again the split tree of the graph to do this test easily and maintain the split tree in constant time. Another constant time algorithm for this problen, with other technique, has been developped in [6].

Let G be a connected distance hereditary graph, and x and y two vertices of G. If $xy \notin E(G)$ resp. $xy \in E(G)$, then let $G' = G + e$ resp. $G' = G - e$ with $e = xy$. Let P be the path from x to y in $ST(G)$, which defines a word W on the alphabet $\{K, L, R, S\}$, where K stands for a clique node, R for a star node with center directed towards x, L for a star node with center directed towards y, and S for a star node with center not directed towards x or y. Note that $xy \in E(G)$ if and only if W contains no letter S.

Theorem 5.1. *The graph G' is distance hereditary if and only if W has one of the following forms, where letters in brackets can be deleted.*
Case $G' = G + e$: $(R)SS(L)$, $(R)SK(L)$, $(R)KS(L)$, $(R)S(L)$.
Case $G' = G - e$: $(R)LR(L)$, $(R)LK(L)$, $(R)KR(L)$, $(R)K(L)$, $(R)(L)$.
In the deletion case, if $W = (R)(L)$, then G' is no longer connected.

As a consequence, we get the following constant time algorithm:

1. Test if W has length at most 4 and satisfies conditions of Theorem 5.1.
2. Update the split tree. Nodes of letters in brackets are called *extreme*.
   (a) If the not-extreme nodes are not ternary, then make a split on these nodes to get ternary nodes instead, in the path from x to y.
   (b) Replace the not-extreme nodes with ternary nodes according to the following table, and, in the cases where there are two not-extreme nodes, exchange the edges of the graph-labelled tree adjacent to these nodes, which are not in the path from x to y, between these nodes. Extreme nodes are unchanged.

edge insertion $\longrightarrow$ $\longleftarrow$ edge deletion	
$(R)SS(L)$	$(R)LR(L)$
$(R)SK(L)$	$(R)LK(L)$
$(R)KS(L)$	$(R)KR(L)$
$(R)S(L)$	$(R)K(L)$

   (c) If necessary, make (at most two) join operations involving the nodes that have been changed to get a reduced graph-labelled tree.

Particular case of cographs. We mention that algorithms in this paper can be adaptated to the particular case of cographs, which are totally decomposable for the modular decomposition. Indeed, a connected cograph is a connected distance hereditary graph of which split tree has the property that every star is directed towards a same given edge. The previous known fully-dynamic recognition algorithms of cographs (see [5] for the vertex insertion and [21] for the other operations) can be restated in the framework of graph-labelled trees, and then turn out to be equivalent to special cases of the previous algorithms.

General split decomposition. We finally mention that a generalization of our insertion algorithm to arbitrary graphs would have a complexity no longer linear in the number of edges. Consider the example where a new vertex is attached to the extermities of a path on $n \geqslant 4$ vertices (whose nodes in the split tree form a path of ternary stars). The resulting cycle is prime, witnessing $\Omega(n)$ changes in the split-tree representation. So, even for circle graphs, such an algorithm would have $\Omega(n)$ time complexity.

References

1. Bandelt, H.-J., Mulder, H.M.: Distance hereditary graphs. J. Combin. Theory Ser. B 41, 182–208 (1986)
2. Di Battista, G., Tamassia, R.: On-line planarity testing. SIAM J. Comput. 25(5), 956–997 (1996)
3. Bretscher, A.: LexBFS based recognition algorithms for cographs and related families. PhD thesis, Dep. of Computer Science, University of Toronto (2005)

4. Corneil, D., Habib, M., Lanlignel, J.-M., Reed, B., Rotics, U.: Polynomial Time Recognition of Clique-Width ≤ 3 Graphs. In: Gonnet, G.H., Viola, A. (eds.) LATIN 2000. LNCS, vol. 1776, pp. 126–134. Springer, Heidelberg (2000)
5. Corneil, D., Perl, Y., Stewart, L.K.: A linear time recognition algorithms for cographs. SIAM J. Comput. 14(4), 926–934 (1985)
6. Corneil, D., Tedder, M.: An optimal, edges-only fully dynamic algorithm for distance-hereditary graphs. In: Proc. STACS 2007. LNCS (to appear, 2007)
7. Crespelle, C., Paul, C.: Fully-dynamic recognition algorithm and certificate for directed cographs. In: Hromkovič, J., Nagl, M., Westfechtel, B. (eds.) WG 2004. LNCS, vol. 3353, pp. 93–104. Springer, Heidelberg (2004) Discrete Appl. Math. 154(12), 1722–1741 (2006)
8. Crespelle, C., Paul, C.: Fully dynamic algorithm for recognition and modular decomposition of permutation graphs. In: Kratsch, D. (ed.) WG 2005. LNCS, vol. 3787, pp. 38–48. Springer, Heidelberg (2005)
9. Cunningham, W.H.: Decomposition of directed graphs. SIAM J. Alg. Disc. Meth. 3, 214–228 (1982)
10. Dahlhaus, E.: Parallel Algorithms for Hierarchical Clustering and Applications to Split Decomposition and Parity Graph Recognition. Journal of Algorithms 36(2), 205–240 (2000)
11. Damiand, G., Habib, M., Paul, C.: A simple paradigm for graph recognition: application to cographs and distance hereditary graphs. Th. Comp. Sci. 263, 99–111 (2001)
12. Eppstein, D., Galil, Z., Italiano, G.F.: Dynamic graph algorithms. In: Algorithms and Theory of Computation Handbook, ch. 8, CRC Press (1999)
13. Eppstein, D., Goodrich, M.T., Yu Meng, J.: Delta-confluent drawings. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS, vol. 3843, pp. 165–176. Springer, Heidelberg (2006)
14. Gallai, T.: Transitiv orientierbare graphen. Acta Mathematica Acad. Sci. Hungar. 18, 25–66 (1967)
15. Hammer, P., Maffray, F.: Completely separable graphs. Discrete Appl. Math. 27, 85–99 (1990)
16. Hell, P., Shamir, R., Sharan, R.: A fully dynamic algorithm for recognizing and representing proper interval graphs. SIAM J. Comput. 31(1), 289–305 (2002)
17. Hseih, S.-Y., Ho, C.-W., Hsu, T.-S., Ho, M.-T.: Efficient algorithms for the hamiltonian problem on distance hereditary graphs. In: Ibarra, O.H., Zhang, L. (eds.) COCOON 2002. LNCS, vol. 2387, pp. 77–86. Springer, Heidelberg (2002)
18. Ibarra, L.: Fully dynamic algorithms for chordal graphs. In: Proc. 10th ACM-SIAM Annual Symp. on Disc. Algorithm (SODA), pp. 923–924 (1999)
19. McKee, T., McMorris, F.R.: Topics in intersection graph theory. SIAM Monographs on Discrete Mathematics and Applications 2 (1999)
20. Oum, S.I.: Rank-width and vertex-minors. J. Combin. Th. Ser. B 95, 79–100 (2005)
21. Shamir, R., Sharan, R.: A fully dynamic algorithm for modular decomposition and recognition of cographs. Discrete Appl. Math. 136(2-3), 329–340 (2004)
22. Spinrad, J.: List of open problems. `www.vuse.vanderbilt.edu/~spin/open.html`
23. Uehara, R., Uno, Y.: Canonical tree representation of distance hereditary graphs and its applications. IEICE Technical Report COMP2005-61 (2006)