**Applied Mathematical Sciences, Vol. 6, 2012, no. 16, 765 - 778**

# Maintaining Graph Properties of Weakly Chordal Graphs

**Mohamed-Amine Boutiche**

- University USTHB, Laboratory LAID3
Faculty of Mathematics, Bab Ezzouar, Algeria
- University Paul Verlaine
Laboratory LITA, Metz, France
boutiche@univ-metz.fr

**Hacène Ait Haddadène**

- University USTHB, Laboratory LAID3
Faculty of Mathematics, Bab Ezzouar, Algeria
- University Paul Verlaine
Laboratory LITA, Metz, France
aithaddadenehacene@yahoo.fr

**Hoai An Le Thi**

University Paul Verlaine
Laboratory LITA, Metz, France
lethi@univ-metz.fr

**Abstract**

We consider the problem of maintaining of graph properties that arise in its tree decomposition. We propose a dynamic algorithm that supports operations of deleting and/or inserting edges or vertices for weakly chordal graphs. Moreover, we update the changes on the parameters tree-width and tree-length for this class when the operations occurs. An implementation of the algorithm is presented in $O(m^2)$ computational complexity order when considering a worst-case analysis.

**Mathematics Subject Classification:** 05C85, 05C10, 05C83

**Keywords:** Tree decomposition, Tree-length, Tree-width, Dynamic algorithm, Weakly chordal

## 1 Introduction

The tree decomposition of a graph is a fundamental concept in graph theory and was introduced by Robertson and Seymour [20] to solve various problems. Tree-width is a widely investigated parameter [1,4,17,20]. Many intractable problems can be solved in polynomial time for graph classes of bounded tree-width [1,12]. Tree-length is a recent graph parameter introduced by Gavoille and Dourisboure [9]. This notion is motivated by the fact that graphs with bounded tree-length have sparse additive spanners and admits compact routing schemes [8,10,19].

Chordal graphs is a most investigated class of graphs. One of the reasons is that the class has a natural tree decomposition called a clique tree, constructed in linear time [8,11,13]. Since chordal graphs admits clique tree representation, their tree-width and tree-length are known (at most $\omega(G)$ for tree-width, where $\omega(G)$ is the size of the largest clique in $G$, and the tree-length is equal to one).

The class of weakly chordal graphs is also well studied and has many known applications [7,15,21]. This class of graph was introduced by Hayward [14], and is a subclass of perfect graphs that include chordal graphs and their complements.

A first dynamic algorithm that maintain a clique tree representation of a chordal graph was introduced by L. Ibarra [16]. This algorithm supports operations of deleting or inserting an arbitrary edges. Later, in [6] the author generalizes this algorithm to maintain some properties on any graph. In our paper, we present a dynamic algorithm that maintain the tree decomposition representation for weakly chordal graphs and supports operations of deleting or inserting edges or vertices. Moreover, we update the changes on the parameters tree-width and tree-length for this class when the operations occurs. The paper is organized as follow; In section 2, we give some important definitions about graphs and their tree decompositions. In section 3, some characterizations dealing with the maintaining of weakly chordal graphs are listed. The dynamic algorithm is presented in section 4. An implementation of the algorithm is presented in section 5, and we finish with a conclusion.

## 2 Main Definitions

### 2.1 General definitions

In this section, we give some basic definitions concerning graphs. Given a graph $G$, we denote by $V(G)$ the set of vertices of $G$, and by $E(G)$ the set of edges. Usually we use $n = |V(G)|$ and $m = |E(G)|$.

- We denote the neighbors of a vertex $v$ by $N_G(v)$ and the set $N_G(v) \cup v$ by $N_G[v]$.

- The distance $dist_G(u, v)$ between vertices $u$ and $v$ of a graph $G = (V, E)$ is the number of edges in a shortest path connecting $u$ and $v$.

- The induced diameter of $S$ denoted by $diam(S)$ is $max_{v,u\in S}\{dist_G(v, u)\}$.

- We denote by $C$ a cycle in a graph $G$.

- A chord of $C$ is an edge of $G$ joining two vertices of $C$ which are not consecutive.

**Definition 2.1** *A graph $G$ is called a chordal (or triangulated) graph if and only if every cycle in $G$ of length $4$ or more has a chord.*

**Definition 2.2** *In a connected graph $G$, a separator $S$ is a subset of vertices whose removal separates $G$ into at least two connected components. $S$ is called a $(a, b)-$separator if and only if it disconnects vertices $a$ and $b$. A $(a, b)-$separator is said to be a minimal separator if and only if it does not contain any other $(a, b)-$separator.*

The notion of tree decomposition was introduced by Robertson and Seymour [20] in their studies of the minors of graphs.

**Definition 2.3** *A tree decomposition of a graph $G$ is a tree $T$ whose vertices, called bags, are subsets of $V(G)$ such that:*

1. $\cup_{X\in V(T)} X = V(G)$
2. *for all $\{u, v\} \in E(G)$, there exist $X \subset V(T)$ such that $u, v \in X$*
3. *for all $X, Y, Z \subset V(T)$, if $Y$ is on the path between $X$ and $Z$ in $T$ then $X \cap Z \subset Y$ (also called the subtree property).*

A tree decomposition is reduced if no bags are contained in another one. A leaf of such decomposition contains necessarily a vertex contained in none other bags.

**Definition 2.4** *The width of a tree decomposition $T$ is denoted by $width(T)$ and we have $width(T)=\max\limits_{X\in V(T)}(|X|-1)$. The length of a tree decomposition $T$ is $length(T)=\max\limits_{X\in V(T)} diam_G(X)$.*

**Definition 2.5** *The tree-width and the tree-length of $G$, denoted by $tw(G)$ and $tl(G)$, are, respectively, $\min\limits_T(width(T))$ and $\min\limits_T(length(T))$, where the minimum is taken over all tree decompositions of $G$.*

We use the following conventions: we refer to the vertices of $G$ and the nodes of $T$ and use $s, t, u, v$ as vertex names and $W, X, Y, Z$ as node names respectively. A node $X \in T$ corresponds to an induced subgraph $B_X$ of $G$. We denote by $G + (x, y)$ (resp. $G - (x, y)$), the graph obtained by adding (resp. removing) an edge $(x, y)$. We denote by $G + x$ (resp. $G - x$), the graph obtained by adding (resp. removing) a vertex $x$.

## 2.2 Weakly Chordal graphs

In this section, we give some definitions yielding with weakly chordal graphs.

**Definition 2.6** *A graph G=(V,E) is said to be weakly chordal or weakly triangulated if no G nor $\overline{G}$ contain a chordless cycle of more than 4 vertices.*

**Definition 2.7** *A pair $x, y$ of distinct, non-adjacent vertices is a two-pair if every induced path from $x$ to $y$ consists of exactly two edges.*

The following result, due to Hayward, Hoàng, and Maffray [15], demonstrates the close relationship between two-pairs and weakly chordal graphs.

**Theorem 2.8 (15)** *If $G$ is a weakly chordal graph, then every induced subgraph of $G$ that is not a clique contains a two-pair.*

**Remark 2.9** *In a chordal graph $G = (V, E)$, for every minimal separator $S$, every component $C$ of $G[V - S]$ contains a confluence point. While, in a weakly chordal graph $G$, for every minimal separator $S$, every full component $C$ contains either a confluence point or a confluence edge [ie; an edge e such that $N(C) \subseteq N(e)$].*

These observations show that an edge in a weakly chordal graph plays a role similar to a vertex in a chordal graph.

Now, we explore the notion of an $S-$saturating edge, which is a stronger version of a confluence edge due to Berry et Al [2,3].

**Definition 2.10** *Given a set $S$ of vertices, an edge $e$ of $G[V - S]$ is said to be $S-$saturating if, for each connected components $S_j$ of $\overline{G}(S)$, at least one endpoint of $e$ see all vertices of $S_j$.*

**Definition 2.11** *An edge e is LB-simplicial if for each minimal separator $S$ included in the neighborhood of e, e is $S-$saturating.*

**Theorem 2.12** [18] *A graph $G = (V, E)$ is weakly chordal if and only if every edge of $E$ is LB-simplicial.*

**Lemma 2.13** [18] *In a given graph $G$, an edge that belong to a hole cannot be LB-simplicial.*

**Lemma 2.14** [18] *In a given graph $G$, each antihole contains an edge that is not LB-simplicial.*

**Theorem 2.15** [3] *Let $G$ be a weakly chordal graph, and let $S$ be a minimal separator of $G$, such that $\overline{G}(S)$ is connected. then, in each full component $C$ of $C(S)$, there is a vertex that see all the vertices of $S$.*

## 3 Maintaining Weakly Chordal Graphs

### 3.1 Edge Insertion

In this subsection, we give some observations where, when the insertion of an edge from a weakly chordal graph, keeps the graph weakly chordal.

**Lemma 3.1 (21)** *Given a weakly chordal graph $G = (V, E)$, a two pair $\{x, y\}$, then the graph obtained by inserting the edge $(x, y)$ to $G$ is weakly chordal.*

Now, suppose that $(x, y)$ extremities are not a two pair

**Lemma 3.2 (3)** *Given a weakly chordal graph $G = (V, E)$, then the graph obtained by adding the edge $(x, y)$ to $G$ is weakly chordal if and only if $(x, y)$ is not a middle edge of any $P_4$ in $G$.*

### 3.2 Edge Deletion

We give some obvious observations where, when the deletion of an edge from a weakly chordal graph, keeps the graph weakly chordal.

**Claim 3.3** *If $G - (x, y)$ contains a $C_5$ then $(x, y)$ is a unique chord of a $C_5$ in $G$.*

**Claim 3.4** *If $G - (x, y)$ contains a $C_6$ then $(x, y)$ is a unique chord of a $C_6$ in $G$ and belongs to at least two $C_4$.*

**Claim 3.5** *If $G - (x, y)$ contains a $C_k$ $k \geq 7$ then $(x, y)$ is a unique chord of a $C_7$ in $G$ which implies that $G$ contains a $C_5$ (contradiction with the fact that $G$ is weakly chordal.*

**Proposition 3.1** *Given a weakly chordal graph $G = (V, E)$, then the graph obtained by removing the edge $(x, y)$ to $G$ fail to be weakly chordal if and only $(x, y)$ is a unique chord of a $C_5$ or a $C_6$ in $G$.*

**Proof.** The necessary condition comes from Fact 1 and Fact 2. For prove the sufficient condition, let $e = (x, y)$ be an edge of $G$ which is the unique chord of a $C_5$ (respectively a $C_6$). Then from lemma 2.13, every edge of $C_5$ (resp.$C_6$) can not be LB-simplicial and from theorem 2.12, $G - (x, y)$ is not weakly chordal.

## 3.3 Vertex Insertion

Let $G = (V, E)$ be a weakly chordal graph. Given a vertex $x$ incident to a set $N(x) \subseteq V$. The insertion of $x$ to $G$ induce a graph $G + x = (V \cup \{x\}, E \cup \{xy/y \in N(x)\})$ .

We give a characterization where, when the insertion of a vertex to a weakly chordal graph, keeps the graph weakly chordal.

**Claim 3.6** *Let $E_x = \{xy/y \in N(x)\}$, then $G + x$ is weakly chordal if and only if every edge in $E_x$ is LB-simplicial.*

**Proof.** From theorem 2.12.

## 3.4 Vertex Deletion

Let $G = (V, E)$ be a weakly chordal graph. Given a vertex $x$ incident to a set $N(x) \subseteq V$. The deletion of $x$ from $G$ induce a graph $G - x = (V \setminus \{x\}, E \setminus \{xy/y \in N(x)\})$.

We give a characterization where, when the removal of a vertex from a weakly chordal graph, keeps the graph weakly chordal.

**Claim 3.7** *Let $x \in V$, then $G - x$ is weakly chordal.*

**Proof.** From heredity of weakly chordal graphs.

## 4 A Dynamic Algorithm

Authors in [4,23] have given an algorithm which compute the tree-width for graphs of polynomial number of minimal separators. They showed that weakly chordal graphs have a polynomial number of minimal separators and that this algorithm compute tree decomposition of minimum width. This tree decomposition has its bags labeled by either cliques or maximal potential cliques that have the following form $(N(x) \cap N(y)) \cup \{x\}$ or $(N(x) \cap N(y)) \cup \{y\}$ with $x, y$ a two pair.

**Proposition 4.1** [5] *The tree decomposition of minimum width for weakly chordal graphs has length at most* 2.

All graphs considered here are weakly chordal and connected. To maintain the graph's tree decomposition in a unified way in our dynamic algorithm, we can extend the definitions in a disconnected case as explained in the end of section 2.1.

## 4.1 Edge Insertion

We show how to update $T$ for $G + \{u, v\}$. Let $T'$ be the tree decomposition of $G + \{u, v\}$. We have three cases:

**Case 1**. $\exists B_X$ such that $uv \in B_X$

$B_X$ is of form $(N(x) \cap N(y)) \cup \{x\}$ or $(N(x) \cap N(y)) \cup \{y\}$ with $x, y$ a two pair. Then, no change $T = T'$, $tw(G) = tw(G')$ and $tl(G) = tl(G')$.

**case 2**. $u \in B_X$, $v \in B_Y$ and $\{X, Y\} \in E(T)$.

**Proposition 4.2** *Let $I = B_x \cap B_y$, then there is a tree decomposition $T'$ of $G + u, v$ where $I \cup \{u, v\}$ is a bag of maximum diameter 2.*

**Proof.** Let $T'$ be a tree defined by $T + (I \cup \{u, v\})$. Then, we have that $I \cup \{u, v\}$ contains only vertices of $G$. The edge $u, v \in G + u, v$ implies that conditions 1 and 2 of definition 2.3 are satisfied for $T'$ and the third condition of definition 2.3 is satisfied by definition of $I$. Then, $I \cup \{u, v\}$ is a bag in $T'$. Moreover, $u$ and $v$ are adjacent in $I \cup \{u, v\}$ and $\forall x, y \in I$ we have $d_I(x, y) \leq 2$. Hence $\forall x, y \in I \cup \{u, v\}$ we have $d_{I\cup\{u,v\}}(x, y) \leq 2$.

And we have the following result concerning the tree-width and tree-length of $G + \{u, v\}$.

**Proposition 4.3** *We have $tw(G') \leq tw(G)$ and $tl(G') \leq 2$.*

**Proof.** Since $width(T) = max_{X\in V(T)} |X| - 1$, if $B_z$ is the unique node that determine the tree-width of $T$, then $tw(G') = tw(G)$. $tl(G') \leq 2$ from prop 4.1.

**case 3**. $\{X, Y\} \notin T$.

We will explicitly use the fact that if $T_u$ and $T_v$ are the subtrees of $T$ induced by $B_G(u)$ and $B_G(v)$ respectively [1], then $T_u$ and $T_v$ do not intersect (because no bag contains $\{u, v\}$).

Let $T_{\overline{uv}}$ be a minimal $B_G(u)$,$B_G(v)$-separator, then $T_{\overline{uv}}$ is a minimal node separator in $T$. Moreover, $T_{\overline{uv}}$ is a subtree of $T$ induced by the nodes of the minimal separator.

**Proposition 4.4** *For $T'$ the tree decomposition of $G' = G + \{u, v\}$ with $\{X, Y\} \notin T$, then we have ; $tw(G') \leq tw(G) + 1$ and $tl(G') \leq 2$.*

**Proof.** If $T_u$ contains at least one bag with $(tw(G) + 1)$ vertices, then the insertion of a vertex $v$ to the bags of $T_u$ implies that $T_u$ will contain at least one bag with $(tw(G) + 2)$ vertices and hence

[1] $B_G(u)$ : bags that contain $u$ and $B_G(v)$ : bags that contain $v$.

$tw(G') = tw(G) + 1$, otherwise, $tw(G') = tw(G)$. Same reasoning for $T_v$. $tl(G') \leq tl(G)$ ; it is clear that the insertion of an edge makes the distances in the graph $G'$ reduce and consequently the length of $T'$, the bound of the tree-length of $G'$ is the same one for $G$.

```
Algorithm Insert Edge

Begin
If $\exists B_X$ such that $uv \in B_X$
    Then no change $T = T'$, $tw(G) = tw(G')$ and $tl(G) = tl(G')$.
If$\{X, Y\} \in T$
    Then
        - Find the closest nodes $X, Y \in T$ such that $u \in B_x$, $v \in B_y$
        - Replace edge $\{X, Y\}$ in $T$ with new node $Z$ representing $B_z = I \cup \{u, v\}$ and add edges $\{X, Z\}, \{Z, Y\}$
If $\{X, Y\} \notin T$
    Then
        - Find the closest nodes $X, Y \in T$ such that $u \in B_x$, $v \in B_y$;
        - Find $T_u$ and $T_v$ the subtrees of $T$ respectively induced by $B_G(u)$ and $B_G(v)$ and Find $T_{\overline{uv}}$ the minimal node separator of $X$ and $Y$ in $T$;
        - Compare between $|T_u|$ and $|T_v|$.
            If $|T_u| \leq |T_v|$ ;
                Then add the vertex $v$ to each bag of $T_{\overline{uv}}$ and $B_x$;
            Else Add the vertex $u$ to each bag of $T_{\overline{uv}}$ and $B_y$.
End
```

**Algorithm Insert Edge** updates $T$, so that $T'$ is a tree decomposition for $G + \{u, v\}$. Furthermore, $T'$ has the induced subtree property and both extremities of the edge $\{u, v\}$ belongs to $Z \in T'$, which satisfies all three conditions of definition 2.1.

## 4.2 Edge Deletion

We show now how we obtain $T'$ for $G - (u, v)$ starting from $T$ the tree decomposition of $G$. Here, we give an algorithm delete edge for this case.

**Algorithm Delete Edge**

```
Begin
If $uv \in B_X$ such that $B_X$ is a clique
Then No change, $T = T'$
If $uv \in B_X$ such that $B_X$ is a non-clique
Then
        - $B_X$ is of form $(N(x) \cap N(y)) \cup \{x\}$ or $(N(x) \cap N(y)) \cup \{y\}$ with $x, y$ a two pair.
        - Partition the set $N(X)$ of $X$'s neighbors into;
            $N_u = \{Y \in N(X)\ /\ u \in B_y\}$
            $N_v = \{Z \in N(X)\ /v \in B_z\}$
            $N_{\overline{uv}} = \{W \in N(X)/u, v \notin B_w\}$
        For every node $X$ of $T$ such that $\{u, v\} \in B_x$ let $B_x^u = B_x - \{v\}$, $B_x^v = B_x - \{u\}$ In $G - \{u, v\}$, every bag $B_x$ is split into two bags $B_x^u$ and $B_x^v$.
        - Replace node $X$ with new nodes $X_1$ and $X_2$ respectively representing $B_x^u$ and $B_x^v$ and add edge $\{X_1, X_2\}$
            If $Y \in N_u$, Replace $\{X, Y\}$ with $\{X_1, Y\}$
            If $Z \in N_v$, Replace $\{X, Z\}$ with $\{X_2, Z\}$
            If $W \in N_{\overline{uv}}$, Replace $\{X, W\}$ with $\{X_1, W\}$ or $\{X_2, W\}$, choosing arbitrarily.
End
```

We observe that $T$ has the induced subtree property. Thus we have the following proposition ;

**Proposition 4.5** *$tw(G') \leq tw(G)$ and $tl(G') \leq 2$.*

**Proof.** We have $width(T) = max_{X \in V(T)} |X| - 1$, if $B_x$ is the unique node that determine the tree-width of $T$, then, the fact that we split $B_x$ into two bags $B_x^u$ and $B_x^v$, reduce the width of $T$, and we have $tw(G') < tw(G)$. Else $tw(G') = tw(G)$. Furthermore, one has $tl(G') \leq 2$; indeed, it is clear that the removal of an edge create two bags $B_x^u$ and $B_x^v$ in such a way that the diameter of each bag do not exceed 2.

## 4.3 Vertex Insertion

We next show how to update$T$ for $G' = G + u$, where $u$ is a vertex such that $u \notin V(G)$.

Let $E' = E(G + u)$, then $E' = E \cup \{\{u, v_i\}; v_i \in N_{G'}(u)$ and $v_i \in V(G)$.

**Algorithm Insert Vertex**

```
Begin
If there exist $X \in T$ such that $v_i \in B_x, \forall i$
Then $X' = B_x \cup \{u\}$ is a bag is $T'$.
Else Insert $u$ to every node in $T$ ,
Then
        - For every node $X$ which is a leaf in $T$ do ; Remove $X$ while it exist at least one node that contain $u$ and $v_i, \forall i$
End
```

At the end of the procedure, we obtain a subtree denoted $T_u$ induced by $B_{G'}(u)$. Hence, $T'$ is obtained from $T$ by updating the bags of the subtree $T_u$ only, other bags remaining unchanged.

$T'$ is a tree decomposition for $G + \{u\}$ satisfying the property of induced subtree. And we have the following result ;

**Proposition 4.6** *We have ; $tw(G) \leq tw(G') \leq tw(G) + 1$ and $tl(G') \leq 3$.*

**Proof.** if $B_x$ is the unique node that determine the tree-width of $T$, then the fact that we add a new vertex to $B_x$ will imply that
$tw(G') = tw(G) + 1$, else, we have, $tw(G') = tw(G)$.

We have : $tl(G') \leq LengthT'$, and $LengthT' = Max_{B \in V(T')} diam_{G'} B$ and $LengthT'_x = Max_{B_x \in V(T')} diam_{G'} B_x$ with $B_x$ bag containing $u$ in $T'$, and $diam_{G''} B_x = max_{v,u \in B_x} \{dist_{G'}(v, u)\} \leq max_{v,u \in B_x} \{dist_G(v, u)\} + 1$ which implies that $tl(G') \leq tl(G) + 1 \leq 3$.

## 4.4 Vertex Deletion

We next show how to update$T$ for $G' = G - \{u\}$, ie, when we delete a vertex $u$ from the graph $G$.

For every node $X \in T$ such that $u \in B_x$, let $B_{x'} = B_x - \{u\}$. In $G - \{u\}$, every bag $B_x$ is replaced by $B_{x'}$.

**Algorithm Delete Vertex**

```
Begin
If there exist $B_{y_i}$ such that $B_{x'} \subset B_{y_i}$, for some $Y_i \in N(X')$,
Then Choose one such $Y_i$ arbitrarily,
        - Contract $\{X', Y_i\}$ and
        - Replace $X'$ with $Y_i$.
End
```

The tree decomposition obtained will be inevitably reduced.

**Proposition 4.7** *We have ; $tw(G) - 1 \leq tw(G') \leq tw(G)$ and $tl(G') \leq 2$*

**Proof.** Indeed, if all the bags of $T$ which contain $(tw(G) + 1)$ vertices are among the bags to which, one removed the vertex $u$, then ; $tw(G') = tw(G) - 1$. Else $tw(G') = tw(G)$. It is obvious that, $tl(G') \leq tl(G) \leq 2$ (by definition).

## 5 Implementation

### 5.1 Data Structure

Given an input weakly chordal graph $G$ represented with an adjacency list, as an initial preprocessing step, we can find the degrees of all vertices in linear time and store the values in a list. Since we suppose that the tree decomposition representation of a graph is given, the vertices can be partitioned into bags $X_i$ according to the tree decomposition $T$ in linear time. Each element in the degree list, indicates both its corresponding bag, and vertex of the graph.

In this way, finding out which set a given vertex belongs to or its degree, deleting a given vertex from a set, and adding a given vertex to a set take constant time. Consequently, moving a given set $A$ of vertices, takes at most $O(|A|)$ time.

Given this structure, to scan or return a node of the tree decomposition takes time linear in the size of the node, since the degree lists are all connected by pointers, thus we can just scan them one after the other, starting with the one corresponding to the minimum degree in the array, to which we have a pointer as well.

As noted before, the size of a degree list is $n$, but since we allow addition of vertices to the graph, for every such operation we should increase the size of the list by 1. This can be done in amortized constant time using dynamic lists, or allocating from the beginning a list of size $n + n\prime$, where $n\prime$ is an upper bound to the number of possible vertex additions given by the user of the algorithm. Notice that this is something every dynamic algorithm has to deal with, independently from the tree decomposition.

### 5.2 Operations

In this section we show how our data structure can be updated efficiently after a modification that maintains the tree decomposition representation of the input graph in a minimal way when adding or deleting a vertex, or adding or deleting an edge.

Before starting the implementation, we need to know the complexity of determining the minimal $a, b-$ separators in weakly chordal graphs which was done in a paper of (Bouchitté and Todinca [4] which have given an efficient algorithm listing all minimal vertex separators of any weakly chordal graph. An implementation of the algorithm is done in [22]. The algorithm determine all minimal $a, b-$separators in $O(m^2)$, where $m$ is the number of edges of $G$.

Recall however that, we suppose that the tree decomposition representation $T$ of a graph $G$ already exists.

We implement the procedure **Insert Edge**, by searching the closest node $X$ and $Y$ such that $u \in B_X$ and $v \in B_Y$ , and determining if $\{X, Y\} \in T$ or not. This takes polynomial time. Two cases arise: If $\{X, Y\} \notin T$, we compute $T_u$ and $T_v$ subtrees of $T$, and the minimal node $X, Y-$separator in $T$ , which is done in polynomial time, and if $\{X, Y\} \in T$, we replace the edge $\{X, Y\}$ with node $Z$ and we add the edges $\{X, Z\}$ and $\{Z, Y\}$. Since $T$ has at most $n$ nodes and each vector has length $n$, The procedures of insertion of an edge runs in $O(m^2)$ time.

We implement the procedure **Delete Edge** by finding the node $X$ and then examining $X$'s neighbors in polynomial time.

We implement the procedure **Insert Vertex** by finding the $X$ node, and adding $u$ to them, then to remove it when there exist at least one node that contain $u$ and $v_i$ , where $v_i \in E'$. This is done in polynomial time.

We implement the procedure of **Delete Vertex** by finding the node $X$, and to remove $u$ from them. This is done in polynomial time.

Finally, the total cost of the dynamic algorithm is $O(m^2)$.

## 6 Conclusion

A polynomial dynamic algorithm that maintains the tree decomposition representation of weakly chordal graphs is presented. The algorithm supports the updates on the graph parameters tree-length and tree-width for the cases of deletion/insertion of an arbitrary edge or node. Moreover, an implementation of the algorithm is presented in $O(m^2)$ computational complexity order when considering a worst-case analysis.

## References

[1] S. Arnborg and A. Prokurowski, A linear time algorithm for NP-Hard problems restricted to partial k-trees, *Discrete Applied Mathematics,* **23** (1989), 11 - 24.

[2] A. Berry, J-P. Bordat, P. Heggernes, G. Simonet, and Y. Villanger, A wide-range algorithm for minimal triangulation from an arbitrary ordering, *Journal of Algorithms,* **58** (2006), 33 - 66.

[3] A. Berry, J-P.Bordat and Pinar Heggernes, Recognizing Weakly Triangulated Graphs by Edge Separability, *Nordic Journal Computing,* **7(3)** (2000) 164 - 177.

[4] V. Bouchitté and I. Todinca, Tree-width and minimum fill -,in ; grouping the minimal separators, *SIAM. J on Computing,* **31 (1)** (2001) 212 - 232.

[5] M.A. Boutiche, H. Ait Haddadène and H. A. Le Thi, Tree-length of some graph classes and application to the routing problem, *Annales ROAD,* (2006).

[6] M.A. Boutiche, Topology Control and Maintaining of Graph Invariants, *Studia Informatica Universalis,* **9(2)**(2011) 38 - 49.

[7] K. Cameron, R. Sritharan and Y. Tang, Finding a maximum induced matching in weakly chordal graphs, *Discrete Mathematics,* **266** (2003) 133 - 142.

[8] Y. Dourisboure, Routage compact et longueur arborescente, *PhD Thesis LaBRI,* University of Bordeaux I (2003).

[9] Y. Dourisboure and C. Gavoille, Tree decomposition with bags of small diameter, *Discrete Math,* **307** (2007) 2008 - 2029.

[10] Y. Dourisboure and C. Gavoille, Improved compact routing scheme for chordal graphs, *Lecture Notes in Computer Science,* **2508** (2002) 252 - 264.

[11] F. Gavril, The intersection graphs of a path in a tree are exactly the chordal graphs, *Journal Comb Theory,* **16** (1974) 47 - 56.

[12] C. Gavoille, M. Katz, N. A. Katz, C. Paul and D. Peleg, Approximate distance labeling schemes, *Lecture Notes in Computer Science,* **2161** (2001) 476 - 488.

[13] M.C.Golumbic, M. Lypshteyn and M. Stern, Intersection Models of Weakly Chordal Graphs, *Discrete Applied Math,* **157 (9)** (2009) 2031 - 2047.

[14] R.Hayward, Weakly triangulated graphs, *J. Comb. Theory ser.B,* **39** (1985) 200 - 208.

[15] R.Hayward, C.T.Hoàng and F.Maffaray, Optimizing Weakly triangulated graphs, *Graphs and Combinatorics,* **5(1)** (1989) 339 - 349.

[16] L. Ibarra, Fully dynamic algorithms for chordal graphs and split graphs, *ACM Transactions on Algorithms,* **4(4)** 40 (2008) 1 20.

[17] T. Kloks, D. Kratsch, and J. Spinrad, On tree-width and minimum fill-in of asteroidal triple-free graphs, *Theoretical Computer Science,* **175** (1997) 309 - 335.

[18] C. G. Lekkerkerker and J. Ch. Boland, Representation of a finite graph by a set of intervals on the real line, *Fund. Math,* **51** (1962) 45 - 64.

[19] D. Lokshtanov, On the Complexity of Computing Treelength, *Lecure Notes in Computer Science,* **4708** (2007) 276 - 287.

[20] N. Robertson and P.D. Seymour, Graph minors. Algorithmic aspects of tree-width, *Journal of Algorithms,* **7** (1986) 309 - 322.

[21] J. Spinrad and R. Sritharan, Algorithms for weakly triangulated graphs, *Discrete Applied Mathematics,* **59** (1995) 181 - 191.

[22] L.S. Skeide, Recognizing weakly chordal graphs, *PhD Thesis,* (2002).

[23] I. Todinca, Aspests algorithmiques des triangulations minimales des graphes, *PhD thesis,* école normale supérieure de Lyon (1999).

**Received: September, 2011**