Chapter 7

The Polytree Algorithm

The goal of this chapter is to introduce the polytree algorithm, a message passing algorithm which falls as a special case of the jointree algorithm that we discussed in the previous chapter. The importance of this special case stems from a number of historical and technical reasons that we also discuss in this chapter.

7.1 Introduction

One of the classical algorithms for inference in Bayesian networks is known as the polytree algorithm. As the name suggests, the algorithm is applicable only to networks having a polytree structure, which are also known as singly–connected networks. The significance of the polytree algorithm stems from a number of reasons:

1. It is one of the first algorithms for inference in Bayesian networks and, as such, has introduced some very influential concepts and notations.

2. It gives a cognitive dimension to its computations as it can attribute a specific probabilistic meaning to each of the subcomputations it performs.

3. It is the basis for a more general class of algorithms, known as conditioning algorithms, which apply to arbitrary Bayesian networks.

4. It is the basis of an influential class of algorithms for approximate inference in Bayesian networks, known as loopy belief propagation algorithms.

The polytree algorithm was discovered before the jointree algorithm that we discussed in the previous chapter. Yet, since it falls as a special case of the jointree algorithm, we will find it more revealing to introduce it from that perspective. In particular, we will first show that the polytree algorithm corresponds to TabElm of Chapter 6, given a specific choice of an elimination tree and an evidence map. We will then discuss the mentioned extensions to the polytree algorithm for handling multiply–connected networks and for performing approximate inference.
7.2 Message Passing on Polytrees

As we mentioned earlier, the polytree algorithm corresponds to TabElm of Chapter 6, given a specific choice of elimination tree and evidence map. The only difference between these two algorithms given these specific choices is the notation used to denote messages and the order in which messages are combined together. We will next introduce the notions of a natural elimination tree, and a natural evidence map, which give rise to the polytree algorithm. We will then explain the notational conventions employed by the polytree algorithm in order to establish its correspondence to TabElm.

7.2.1 Natural Elimination Trees

Consider the polytrees in Figure 7.1 and suppose that we want to answer queries about these networks using TabElm from Chapter 6. We must then specify an elimination tree and an evidence map for each of these networks, which must be passed as input to TabElm, in addition to the queries of interest. Consider now the elimination trees in Figure 7.2, which correspond to the polytrees in Figure 7.1. These elimination trees were obtained from the corresponding polytrees by simply dropping the directionality of edges. We shall call such trees natural elimination trees. Formally, if \( N \) is a Bayesian network with a polytree structure, then a natural elimination tree for \( N \) is a triple \( \langle N, A, \phi \rangle \), where \( N \) are the variables of \( N \), \( A \) are the undirected edges of \( N \), and \( \phi_X \) is the CPT of variables \( X \) in network \( N \).

Figure 7.3 depicts the separators, clusters and widths of each elimination tree in Figure 7.2. We have a number of observations about this figure:

1. The separator \( S_{XY} \) contains only a single node \( X \), where \( X \) is the parent of \( Y \) in the corresponding polytree.

2. \( \text{cluster}(X) \) contains only the family of \( X \) in the corresponding polytree.

3. The width of each elimination tree is the maximum number of parents per node in its corresponding polytree (the polytree treewidth). Therefore, each of the elimination trees is optimal in this case.

These properties are not accidental. In fact, each of them is a property that holds for every natural elimination tree which is constructed for a polytree.

The above results have a number of implications on TabElm when applied to a natural elimination tree. First, any message sent from one node to another will always be defined over a single variable. In particular, the message sent from node \( U \) to its child \( X \) will be defined over variable \( U \) only. Moreover, the message sent from node \( Y \) to its parents \( X \) will be defined over variable \( X \) only. Second, the worst case complexity of TabElm in this case will be \( O(n \exp(w)) \), where \( n \) is the number of nodes in the polytree and \( w \) is its treewidth.
Figure 7.1: Polytrees.

Figure 7.2: Elimination trees for the polytrees in Figure 7.1.

Figure 7.3: Separators and clusters of elimination trees in Figure 7.2.
7.2.2 Natural Evidence Maps

We have mentioned earlier that the polytree algorithm corresponds to TabElm when applied to the natural elimination tree and natural evidence map. We will now define a natural evidence map.

Given a natural elimination tree \( \langle N, A, \phi \rangle \) for polytree \( N \), an evidence map \( \epsilon \) is said to be natural iff it maps each variable \( X \) in polytree \( N \) into the node \( X \) in the elimination tree: \( \epsilon(X) = X \). In the first elimination tree in Figure 7.2, the natural elimination map would be

\[
\epsilon(A) = A, \quad \epsilon(B) = B, \quad \epsilon(C) = C, \quad \epsilon(D) = D.
\]

Note that by definition of a natural elimination tree, each node \( X \) in such a tree will host the conditional probability table of \( X, \Theta_{X|U} \). Therefore, the natural evidence map ensures that evidence about variable \( X \) will always be incorporated into the conditional probability table of \( X \).

7.2.3 Notating Messages on a Polytree

As we mentioned earlier, when using the natural elimination tree and natural evidence map of a polytree, TabElm degenerates to what is known as the polytree algorithm. This latter algorithm, however, utilizes a different notation for denoting messages, which we will now introduce. Figure 7.4 depicts the notational conventions for denoting messages in the polytree algorithm. Specifically:

- A message which is sent from parent \( U_i \) to child \( X \) is denoted by \( \pi_X(U_i) \), and is referred to as the causal support for \( X \) from its parent \( U_i \).
A message from child $Y_i$ to its parent $X$ is denoted by $\lambda_{Y_i}(X)$, and is referred to as the diagnostic support for node $X$ from its child $Y_i$.

Note that the message from parent $U_i$ to child $X$ is a probability table over variable $U_i$ and, hence, each entry in the table $\pi_X(U_i)$ corresponds to a particular value $u_i$ of the parent $U_i$, denoted $\pi_X(u_i)$. Similarly, the message from child $Y_i$ to its parent $X$ is a probability table over variable $X$. Therefore, each entry in this message corresponds to a particular value $x$ of the parent $X$, denoted $\lambda_{Y_i}(x)$.

A key point here is that messages received from parents are notated differently from messages received from children. This distinction—which was not made in TabElm where the message from node $i$ to node $j$ was simply denoted by $M_{ij}$—has compelling reasons in the context of the polytree algorithm which will become apparent later.

The polytree algorithm employs three further conventions for denoting the combination of messages received from the children of node $X$; the combination of messages received from the parents of node $X$; and the combination of messages received from all neighbors of node $X$. Specifically,

$$\lambda(X) \overset{\text{def}}{=} \prod_i \lambda_{Y_i}(X)$$

is used to denote the multiplication of all messages received from children $Y_i$ by node $X$, and is called the diagnostic support for node $X$. Note that $\lambda(X)$ is a probability table over variable $X$ since each of the messages $\lambda_{Y_i}(X)$ is also a probability table over variable $X$.

The polytree algorithm also uses the notation

$$\pi(X) \overset{\text{def}}{=} \sum_{\mathbf{U}} \Theta_{X|\mathbf{U}} \mathbf{e}_X \prod_i \pi_X(U_i),$$

to denote the above combination of messages obtained from parents $U_i$ by node $X$, and refers to this as the causal support for node $X$. Here, $\mathbf{U}$ are the parents of node $X$, $\Theta_{X|\mathbf{U}}$ is the CPT of $X$, and $\mathbf{e}_X$ is the evidence pertaining to variable $X$.\(^1\)

Finally, the notation

$$BEL(X) \overset{\text{def}}{=} \pi(X)\lambda(X)$$

is used to denote the multiplication of causal and diagnostic supports for node $X$, which is referred to as the belief in node $X$. The polytree algorithm computes this belief table for each node $X$ in the polytree.

As we shall see next, $BEL(X)$ is indeed equal to the probability table returned by calling $\text{pull}(X, X, \mathbf{e})$ of TabElm. Therefore, one run of the polytree algorithm with respect to evidence $\mathbf{e}$ corresponds to calling $\text{pull}(X, X, \mathbf{e})$ for each node $X$ in the polytree.

\(^1\)The derivation of the original polytree algorithm assumes no evidence on variable $X$ as such evidence is emulated by evidence on a virtual child of node $X$. Under that assumption, the notation would be: $\pi(X) \overset{\text{def}}{=} \sum_{\mathbf{U}} \Theta_{X|\mathbf{U}} \prod_i \pi_X(U_i)$.
7.2.4 The Correspondence

If \( \text{TabElm} \) is applied to a polytree \( \mathcal{N} \), using the natural elimination tree and the natural evidence map, then \( \text{TabElm} \) will correspond to the polytree algorithm. In particular, for a node \( X \) in the polytree, its parents \( U_i \), and children \( Y_i \), the query \( \text{pull}(X, X, e) \) to \( \text{TabElm} \) returns the table

\[
\text{project}(\Theta_{X|U}^{e_X} \prod_i \pi_X(U_i) \prod_i \lambda_Y(X), X).
\]

This table which provides the joint marginal \( \Pr(X, e) \) is equal to:

\[
\sum_{U} \Theta_{X|U}^{e_X} \prod_i \pi_X(U_i) \prod_i \lambda_Y(X) = \prod_i \lambda(Y_i) \sum_{U} \Theta_{X|U}^{e_X} \prod_i \pi_X(U_i)
\]

\[
\pi(X) \cdot \lambda(X) = \text{BEL}(X).
\]

This is the belief in node \( X \) as computed by the polytree algorithm.

Therefore, given the natural elimination tree and the natural evidence map, a call to \( \text{pull}(X, X, e) \) in \( \text{TabElm} \) will compute the belief in node \( X \) as defined by the polytree algorithm. Moreover, the messages passed from parents to children in this computation correspond to \( \pi \)-messages in the polytree algorithm, and those passed from children to parents correspond to \( \lambda \)-messages. Note that \( \text{TabElm} \) does not explicitly combine the diagnostic supports for node \( X \) into \( \lambda(X) \). Neither does it explicitly combine the causal supports for node \( X \) into \( \pi(X) \). Instead, it directly combines all diagnostic and causal supports to compute the belief \( \text{BEL}(X) \). The algorithm can be easily modified, however, to perform these intermediate combinations if necessary.

7.2.5 An Example

Consider the Bayesian network in Figure 7.5, which has a polytree structure. Figure 7.5 depicts the natural elimination tree for this network, together with the messages computed by calling \( \text{pull}(D, D, \text{true}) \) of \( \text{TabElm} \). We have three \( \pi \)-messages in this case, which are given below with their polytree notations:

<table>
<thead>
<tr>
<th>( A )</th>
<th>( \pi_B(A) )</th>
<th>( B )</th>
<th>( \pi_B(B) )</th>
<th>( C )</th>
<th>( \pi_B(C) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>.01</td>
<td>true</td>
<td>.00199</td>
<td>true</td>
<td>.001</td>
</tr>
<tr>
<td>false</td>
<td>.99</td>
<td>false</td>
<td>.99801</td>
<td>false</td>
<td>.999</td>
</tr>
</tbody>
</table>

We also have two \( \lambda \)-messages:

<table>
<thead>
<tr>
<th>( D )</th>
<th>( \lambda_F(D) )</th>
<th>( D )</th>
<th>( \lambda_F(D) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>1</td>
<td>true</td>
<td>1</td>
</tr>
<tr>
<td>false</td>
<td>1</td>
<td>false</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 7.5: Message passing in a polytree.
Again, note that each of the messages is defined over a single variable. Moreover, any message which relates to directed edge $U \rightarrow X$ is defined over the single variable $U$.

The diagnostic support for node $D$ in this case is:

<table>
<thead>
<tr>
<th>$D$</th>
<th>$\lambda(D) = \lambda_F(D)\lambda_F(D)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>1</td>
</tr>
<tr>
<td>false</td>
<td>1</td>
</tr>
</tbody>
</table>

Moreover, the causal support for node $D$ is:

| $D$  | $\pi(D) = \sum_{BC} \Theta_{D|BC}\pi_D(B)\pi_D(C)$ |
|------|---------------------------------------------------|
| true | .0127094085                                      |
| false| .9872905915                                      |

If we multiply tables $\lambda$ and $\pi$ in this case, we obtain the belief in node $D$:

<table>
<thead>
<tr>
<th>$D$</th>
<th>$BEL(D) = \pi(D)\lambda(D)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>.0127094085</td>
</tr>
<tr>
<td>false</td>
<td>.9872905915</td>
</tr>
</tbody>
</table>

which is also the answer returned by pull($D, D, true$) of TabElm.

### 7.3 The Meaning of Polytree Messages

One of the most important attributes of the polytree algorithm is that each of its subcomputations has a very specific probabilistic meaning, which we will discuss next. We must first, however, introduce the specific notation used by the algorithm for denoting different pieces of evidence pertaining to nodes in a polytree. These conventions are depicted in Figure 7.6 and can be summarized as follows:

- $e^+_X$ denotes evidence pertaining to node $X$ and its ancestors in the polytree.\(^2\)
- $e^-_X$ denotes evidence pertaining to descendants of node $X$ in the polytree.
- $e^+_XY$ denotes evidence pertaining to nodes on the $X$–side of directed edge $X \rightarrow Y$ in the polytree.
- $e^-_{XY}$ denotes evidence pertaining to nodes on the $Y$–side of directed edge $X \rightarrow Y$ in the polytree.

For example, in Figure 7.7, we have

- $e^+_D$: $A = false$.

\(^2\)The original polytree algorithm assumes that there is no evidence about node $X$ in its derivation, as it assumes that such evidence can be incorporated as evidence on a virtual child of node $X$. 
\begin{itemize}
  \item $e_D^-$: $E=\text{true}$, $F=\text{false}$.
  \item $e_{DF}^+$: $A=\text{false}$, $E=\text{true}$.
  \item $e_{DF}^-$: $F=\text{false}$.
\end{itemize}

As we will now show, each message passed by the polytree algorithm has a very specific probabilistic meaning. The same also applies to each combination of messages performed by the algorithm. For example, the message $\pi_X(U_i)$ that node $X$ receives from its parent $U_i$ has the following probabilistic meaning:

$$
\pi_X(u_i) = \Pr(u_i, e_{U_i,X}^+) = \Pr(u_i, e_{U_i,X}^-).
$$

That is, it represents the probability of $u_i$ and evidence $e_{U_i,X}$ which pertains to nodes connected to $X$ through its parent $U_i$. Consider Figure 7.8 for example, which depicts the messages computed by TabElm as a result of calling pull($D, D, A=\text{false}, E=\text{true}, F=\text{false}$). The message $\pi_D(B)$ which node $D$ receives from its parent $B$ has the following meaning:

$$
\begin{array}{c|c}
B & \pi_D(B) \\
\hline
\text{true} & .00099 = \Pr(B=\text{true}, A=\text{false}) \\
\text{false} & .98901 = \Pr(B=\text{false}, A=\text{false}) \\
\end{array}
$$

since evidence $e_{BD}^+$ is $A=\text{false}$ in this case.

Moreover, the message $\lambda_{Y_i}(X)$ that node $X$ receives from its child $Y_i$ has the following probabilistic meaning:

$$
\lambda_{Y_i}(x) = \Pr(e_{XY_i}^- | x).
$$

That is, it represents the probability of evidence $e_{XY_i}^-$, which pertains to nodes connected to $X$ through its child $Y_i$, given $x$. In Figure 7.8, the message $\lambda_E(D)$ which node $D$ receives from its child $E$ has the following probabilistic meaning:

$$
\begin{array}{c|c}
D & \lambda_E(D) \\
\hline
\text{true} & .9 = \Pr(E=\text{true} | D=\text{true}) \\
\text{false} & .3 = \Pr(E=\text{true} | D=\text{false}) \\
\end{array}
$$

since evidence $e_{DE}^+$ is $E=\text{true}$ in this case.

The causal support that node $X$ receives from its parents has the following probabilistic meaning:

$$
\pi(x) = \Pr(x, e_X^+).
$$

That is, it represents the probability of $x$ and all evidence pertaining to $X$ and its ancestors in the polytree. In Figure 7.8, the causal support $\pi(D)$ for node $D$ has the following meaning:

$$
\begin{array}{c|c}
D & \pi(D) \\
\hline
\text{true} & .0117108585 = \Pr(D=\text{true}, A=\text{false}) \\
\text{false} & .9782891415 = \Pr(D=\text{false}, A=\text{false}) \\
\end{array}
$$
Figure 7.6: Notational conventions for denoting evidence in a polytree.

Figure 7.7: Example evidence in a polytree.
since evidence $e_D^e$ is $A$=false in this case.

The diagnostic support that node $X$ receives from its children has the following probabilistic meaning:

$$\lambda(x) = \Pr(e_X^c | x).$$

That is, it represents the probability of all evidence pertaining to descendants of $X$, given $x$. In Figure 7.8, the diagnostic support $\lambda(D)$ for node $D$ has the following meaning:

<table>
<thead>
<tr>
<th>$D$</th>
<th>$\lambda(D)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>.72 = $\Pr(E=\text{true}, F=\text{false}</td>
</tr>
<tr>
<td>false</td>
<td>.27 = $\Pr(E=\text{true}, F=\text{false}</td>
</tr>
</tbody>
</table>

since evidence $e_D^e$ is $E=\text{true}, F=\text{false}$ in this case.

### 7.4 The Original Polytree Algorithm

As we mentioned earlier, the original polytree algorithm was derived independently of the jointree algorithm. Specifically, the following quantities were initially defined:

$$\pi_X(u_i) \stackrel{def}{=} \Pr(u_i, e_{V_i \setminus X}^e)$$
Given these definitions, the following relationships were then proven based on the independencies that hold in a polytree:

\[
\begin{align*}
\pi(x) & \overset{\text{def}}{=} \Pr(x, e_X^+) \\
\lambda_{Y_i}(x) & \overset{\text{def}}{=} \Pr(e_{X_{Y_i}} | x) \\
\lambda(x) & \overset{\text{def}}{=} \Pr(e_X^- | x) \\
\text{BEL}(x) & \overset{\text{def}}{=} \Pr(x, e).
\end{align*}
\]

The above equations were then taken as the basis of the polytree algorithm. Our treatment of the polytree algorithm was different in this chapter, where we derived it as a special case of the jointree algorithm. Specifically, all of the above equations fall immediately from \(\text{TabElm}\) given a natural elimination tree, a natural evidence map, and the polytree conventions for denoting messages.

We also note here that there are two versions of the polytree algorithm. The one we described in this chapter was proposed by Peot and Shachter. The original version was proposed by Pearl and it defines \(\pi_X(u_i)\) as \(\Pr(u_i | e_{U_i}X)\) instead of \(\Pr(u_i, e_{U_i}X)\). As a result, \(\text{BEL}(x)\) becomes \(\Pr(x | e)\) instead of \(\Pr(x, e)\).

The version we described in this chapter, however, is the more commonly used version since it computes joint marginals instead of conditional marginals, which allows one to obtain the probability of evidence for free.

## 7.5 Cutset Conditioning

The polytree algorithm was among the first algorithms for inference in Bayesian networks. Although it applies to singly–connected networks only, it can be easily extended to handle multiply–connected networks using a method known as \textit{loop–cutset conditioning}.

Consider Figure 7.9 which depicts a multiply–connected Bayesian network. Suppose further that we want to answer the query \(\Pr(E, D=\text{true}, B=\text{true})\). As we know from Chapter 5, we can orient this network to the given query, leading to the network in Figure 7.10. This network is singly–connected and is equivalent to the one in Figure 7.9 as far as answering the query \(\Pr(E, D=\text{true}, B=\text{true})\).
Figure 7.9: A multiply-connected Bayesian network.
Figure 7.10: A singly-connected Bayesian network which results from orienting the multiply-connected network in Figure 7.9 towards the query $\Pr(E, D=true, B=true)$ or $\Pr(E, D=true, B=false)$. 

| $C$     | $D$     | $\sum_B \Theta_{D|BC}^{B=true}$ | $C$     | $D$     | $\sum_B \Theta_{D|BC}^{B=false}$ |
|---------|---------|----------------------------------|---------|---------|----------------------------------|
| true    | true    | .95                              | true    | true    | .8                               |
| true    | false   | .05                              | true    | false   | .2                               |
| false   | true    | .9                               | false   | true    | 0                                |
| false   | false   | .1                               | false   | false   | 1                                |
Therefore, the polytree algorithm is all we need to answer this query, even though the original Bayesian network is multiply-connected.

The ability in this case to use the polytree algorithm with respect to a multiply-connected network \( N \) is due to a special property of the query \((Q, e)\): orienting the network \( N \) towards this query leads to a singly-connected network \( N' \). But what if the query \((Q, e)\) does not satisfy this property? For example, what if the query we want to answer was, say, \( \Pr(E, D=\text{true}) \)?

In such a case, we can consider two queries, \( \Pr(E, D=\text{true}, B=\text{true}) \) and \( \Pr(E, D=\text{true}, B=\text{false}) \), which can be answered using polytrees \( N^1 \) and \( N^2 \). Here, polytree \( N^1 \) results from orienting the original network towards query \( \Pr(E, D=\text{true}, B=\text{true}) \), and polytree \( N^2 \) results from orienting the network towards \( \Pr(E, D=\text{true}, B=\text{false}) \); see Figure 7.10. Specifically, the first query with respect to polytree \( N^1 \) leads to the following joint marginal:

<table>
<thead>
<tr>
<th></th>
<th>( \Pr(E, D=\text{true}, B=\text{true}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>.077</td>
</tr>
<tr>
<td>false</td>
<td>.217</td>
</tr>
</tbody>
</table>

and the second query with respect to polytree \( N^2 \) leads to

<table>
<thead>
<tr>
<th></th>
<th>( \Pr(E, D=\text{true}, B=\text{false}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>.226</td>
</tr>
<tr>
<td>false</td>
<td>.097</td>
</tr>
</tbody>
</table>

Adding up the corresponding entries in the above table, we get

<table>
<thead>
<tr>
<th></th>
<th>( \Pr(E, D=\text{true}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>.303</td>
</tr>
<tr>
<td>false</td>
<td>.314</td>
</tr>
</tbody>
</table>

which is the joint marginal for our original query.

In general, if we have a Bayesian network \( N \) and query \( \Pr(Q, e) \), we can assemble the answer to this query from answers to the more detailed queries \( \Pr(Q, ec) \) as follows:

\[
\Pr(q, e) = \sum_c \Pr(q, ec),
\]

where \( C \) is any set of nodes in the Bayesian network \( N \). We can always choose the set of nodes \( C \) to ensure that the orientation of network \( N \) to query \( (Q, ec) \) is a singly-connected network \( N' \). We simply have to choose the nodes in \( C \) so that deleting their outgoing edges from \( N \) leads to a singly-connected network \( N' \). We then have

\[
\Pr(q, e) = \sum_c \Pr(q, ec) = \sum_c \Pr'(q, ec),
\]
where $\Pr'$ is the joint probability distribution with respect to the polytree $\mathcal{N}'$. A set of nodes $C$ which satisfies the above property is known as a loop-cutset for the network $\mathcal{N}$. Moreover, the method we outlined above is known as loop-cutset conditioning.

Figure 7.11 depicts a few networks and their corresponding loop cutsets. Given a Bayesian network $\mathcal{N}$ with $n$ nodes, and a corresponding loop cutset $C$ of size $s$, the method of loop-cutset conditioning requires $O(\exp(s))$ invocations to the polytree algorithm. Moreover, each of these invocations takes $O(n \exp(k))$ time, where $k$ is the maximum number of parents per node in the polytree $\mathcal{N}'$ (which results from removing the edges outgoing from cutset $C$ in network $\mathcal{N}$). Therefore, loop-cutset conditioning takes $O(n \exp(k + s))$ time, which is exponential in the size of used cutset. Computing a loop-cutset of minimal size is hence an important task in the context of cutset conditioning, but such computation is known to be NP-hard.

Note, however, that we only need to store the accumulated sum of tables $\Pr'(Q, ec)$ across the different calls to the polytree algorithm. Therefore, the space complexity of loop-cutset conditioning is only $O(n \exp(k))$, which is the space complexity of the polytree algorithm. This is clearly not exponential in the cutset size and is quite important as the elimination algorithms we discussed in Chapters 5 and 6 have time and space complexities which are both exponential in the treewidth.