information

MDPI

Article

The Treewidth of Induced Graphs of Conditional Preference Networks Is Small

Jie Liu and Jinglei Liu *

School of Computer and Control Engineering, Yantai University, Yantai 264005, China; 1634034956@qq.com
* Correspondence: jinglei_liu@sina.com; Tel.: +86-136-8866-7725

Academic Editor: Willy Susilo
Received: 16 September 2015; Accepted: 2 February 2016; Published: 14 February 2016

Abstract: Conditional preference networks (CP-nets) are recently an emerging topic as a graphical model for compactly representing ordinal conditional preference relations on multi-attribute domains. As we know, the treewidth, which can decrease the solving complexity for many intractability problems, is exactly a fundamental property of a graph. Therefore, we can utilize treewidth to solve some reasoning tasks on induced graphs, such as the dominance queries on the CP-nets in the future. In this paper, we present an efficient algorithm for computing the treewidth of induced graphs of CP-nets; what we need is to make an assumption that the induced graph of a CP-net has been given. Then, we can leverage the *Bucket Elimination* technique to solve treewidth within polynomial time. At last, it is revealed that by our experiment, the treewidth of induced graphs of CP-nets is much smaller with regard to the number of vertices. For example, for an induced graph of CP-net with 1024 vertices, its treewidth is only 10. As far as we know, this is the first time, using the *Bucket Elimination*, to compute the treewidth of an induced graph of a CP-net. This approach for solving the treewidth may lay a good foundation for efficiently solving dominance queries on CP-nets in the future.

Keywords: conditional preference networks; induced graph; treewidth; bucket elimination; dominance queries

1. Introduction

Conditional preference networks (CP-nets) [1,2] have permeated our daily life as a specific graphical model. As a novel graphical tool, it has been used for representation and reasoning preference by the intuitive and compact preference statements since 1999. Recently, CP-nets have also played a crucial role in the areas of decision theory [3,4], recommender systems [5], and databases [6]. Many graph problems, however, are NP-hard [7]; the induced graphs of CP-nets are no exception. Parameter calculation theory [8,9] has provided strong support to solve NP-hard problems in recent years. Treewidth was coined by Robertson and Seymour on graph minors [10], which provides some new ideas for researching graphical models, such as probabilistic reasoning, constraint satisfaction, and knowledge representation and reasoning. CP-nets are a graphical model for compactly representing conditional qualitative preference relations [1,11]. Therefore, various queries and operations have been carried out on the induced graphs of CP-nets. It is well known that most combinatorial optimization problems are NP-complete problems. Luckily, however, these NP-complete problems will be solved in polynomial time if the treewidth of the input graph is bounded by a constant. Namely, ordering queries and dominance queries with respect to acyclic CP-nets will be answered efficiently in the number of variables if the treewidth of induced graphs of CP-nets have been obtained.

In this paper, we investigate how to compute the treewidth of induced graphs of CP-nets in Section 4.2. When the treewidth of induced graphs of CP-nets is computed by our algorithm, some reasoning tasks will become rather simple. For example, let us consider the evening dress case [1]—CP-nets express a user's preference for evening dress, we then work out the treewidth of its induced graph. Obviously, some operations, such as dominance queries [1], can be solved quickly. We apply a heuristic search method, *Bucket Elimination* [12], to find the treewidth of induced graphs of CP-nets. Though our algorithm seems to be applicable to undirected induced graphs of CP-nets, in Darwiche's textbook [13], the authors introduce the process of how to *moralize* a directed graph into a undirected graph. This method can be used in the induced graphs for CP-nets. The goal of our research, as well as the focus of this paper, is to apply heuristic search techniques to the problem of finding the treewidth of induced graphs of CP-nets of real world interest.

The main contributions of this work are threefold.

1 We explore the treewidth problem about the induced graphs of CP-nets— as far as we know, there is no literature availlable to study the treewidth characterization of induced graphs of CP-nets.
2 We design a more efficient algorithm to solve the treewidth of induced graphs of CP-nets utilizing a *Bucket Elimination* approach; this approach uses the randomness characteristics of input order to speed search.
3 We find that the treewidth of induced graphs of CP-nets is very small; this interesting discovery may lay the foundation for designing an algorithm for solving tractable reasoning tasks, such as dominance queries, in the future.

The remainder of this paper is organized as follows: Section 2 provides background on treewidth and some important notions—tree decomposition of a graph, CP-nets, and induced graphs of CP-nets. In Section 3, we introduce related work of treewidth and CP-nets. In Section 4, we present an algorithm to compute the treewidth of induced graphs of CP-nets, in the meantime providing a detailed example to illustrate the performing procedure of the algorithm. In Section 5, we explain our experimental setup and metrics, and give our experimental results in simulated data. Finally, we discuss a number of interesting directions for future theoretical research and applications in Section 6.

2. Background on Treewidth and CP-nets

Below, we first give a more detailed introduction of the necessary concepts, such as treewidth, CP-nets, and its induced graph.

2.1. Treewidth

The notion of treewidth was introduced by Robertson and Seymour in their work on graph minors [10].

Definition 1. *A tree decomposition of a graph $G = (V, E)$ is a pair (X, T), where $\{X_i \in X \mid i = 1, \cdots, n\}$ is a set of subsets of V, and T is a tree. Also, each node of T is exactly equal to an arbitrary subset of X. The (X, T) possess the following properties:*

(i) $\bigcup_{i \in T} X_i = V$*. That is, each graph vertex is contained in at least one tree node.*
(ii) *If vertices v and w both are connected in a graph G, then v and w are contained in at least one subset X_i.*
(iii) *Considering the three nodes i, j, k in the tree, when j exists in the path of i and k,$X_i \bigcap X_k \subseteq X_j$, that is, common vertices of i and k must appear in the j.*

Definition 2. *The width of tree decomposition $D = (X, T)$ equals $max_{X_i \in X} \mid X_i \mid -1$, and the minimum width of a tree decomposition of G is the treewidth of the graph G.*

Figure 1 shows a graph G and one of its tree decompositions; from it, we can draw the result that the width of the tree decomposition is two.

In fact, treewidth has a close relationship with chordal graphs [14], and they have a special tree decomposition called a clique tree. So, we can further give an equivalent definition of treewidth: the treewidth of G is one less than the size of the largest clique in the chordal graph containing G with the smallest clique number. A chordal graph with this clique size may be obtained by adding an edge between every two vertices that both belong to at least one of the sets X_i.

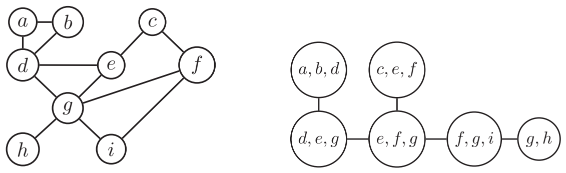


Figure 1. Graph G and its tree decomposition.

Actually, the treewidth of some graphs is determinate, and the graphs for which treewidth is easier to compute are those with a known constant treewidth. For example, tree structure is a special kind of graph, and its treewidth is 1. A complete graph, or a clique with n vertices, is the least tree-like type of graph; it has a treewidth of $n-1$ [15]. Moreover, series-parallel graphs (SPG) have a treewidth of 2, and a treewidth of $m-grid$ is m [16]. There are other types of graphs that do not have constant treewidth, but for which the treewidth can be found in polynomial time. One example is the chordal graph, or triangulated graph. Although determining the treewidth of an arbitrary graph is NP-complete [17], in practical application, we can use heuristic algorithms to construct tree decomposition, thus obtaining its approximate optimal treewidth. In past research, *Bucket Elimination* and *Separator* techniques [18] have often been used in these heuristic algorithms. In this paper, we will concentrate on computing the treewidth of induced graphs of CP-nets, and illustrate in detail how to use *Bucket Elimination* to obtain treewidth in the next section.

2.2. CP-nets

We assume that readers are familiar with standard notions of CP-nets; next, we review the basics of CP-nets briefly.

Let $V = \{X_1, \cdots, X_n\}$ be a set of variables. $Dom(X_i)$ indicates the sets of variables of X_i, and $|Dom(X_i)|$ denotes its size. $\Omega = \times_{i=1}^{n} Dom(X_i)$ denotes all combinations of variables. The size $|\Omega| = \prod_{i=1}^{n} Dom(X_i)$ denotes the total number of outcomes. Any two outcomes $o, o' \in \Omega$ are regarded as swap outcomes if and only if they have only one different attribute value. If X_i is preferred over X_j then, X_j is a parent node of X_i. We define $Pa(X_i)$ as the parents set of X_i.

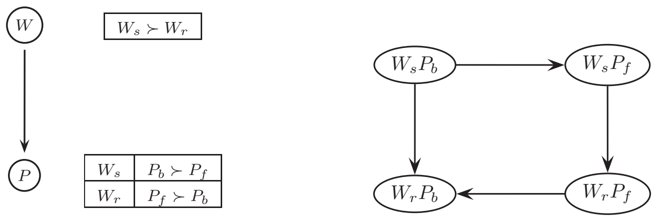
Definition 3. *A CP-net $N =< V, E >$ is a directed graph over variables $V = \{X_1, \cdots, X_n\}$, whose nodes are annotated with conditional preference tables $CPT(X_i)$ for each $X_i \in V$. Each conditional preference table $CPT(X_i)$ express the preference on $Dom(X_i)$ with respect to different assignments on the variable set $Pa(X_i)$.*

Definition 4. *Let $N =< V, E >$ be a CP-net, then the corresponding directed graph $N' =< \Omega, IE >$ is called the induced graph of N, where IE is the directed edges set indicating all the swapping relations. The induced graph N' has $|\Omega|$ nodes, considering the antisymmetry of the preference relation; there also are at most $n|\Omega|/2$ edges in the induced graph of N.*

Here we use a group of examples to illustrate the semantics of CP-net and its induced graph.

Example 1. *Consider the simple CP-net in Figure 2 that expresses my preference of weekend activities. This network consists of two variables, W and P, standing for the weather and plan, respectively. Now, I strictly prefer sunny (W_s) to rainy (W_r), while my preference between film (P_f) and ball (P_b) is conditioned on the weather to be served: I prefer playing basketball if served a sunny day, and seeing a film if served a rainy day.*

Figure 2 shows the preference graph over outcomes induced by this CP-net. An arrow in this graph directed from outcome o_i to o_j indicates that a preference for o_i over o_j can be determined directly from one of the CPT$_s$ in the CP-net. For example, the fact that W_sP_b is preferred to W_sP_f (as indicated by the direct arrow *between them) is a direct consequence of the semantics of $CPT(P)$. The top-left element (W_sP_b) is the best outcome, while the bottom-left element (W_rP_b) is the worst.*

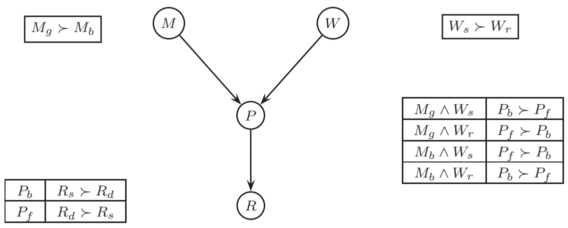


W_s	$P_b \succ P_f$
W_r	$P_f \succ P_b$

Figure 2. An example of CP-net named "My Weekend".

Next, we will give a complicated example of a CP-net.

Example 2. *Figure 3 illustrates another CP-net that expresses my preferences for the weekend. It consists of four variables, M, W, P, and R, standing for mood, weather, plan, and rest, respectively. With the underlying assumption that I always prefer a good mood to a bad mood, and good weather is preferred to bad weather. While my preference between the film and ball is conditioned on the combination of mood and weather, if I am happy and the weather is sunny, or I am not happy and the weather is rain, then playing basketball is my choice rather than film. Otherwise, I will go to a film. Finally, if I choose to play basketball, then I will take a bath to relax. On the contrary, I might have a big meal. Figure 4 shows the corresponding induced graph of the CP-net shown in Figure 3.*



$M_g \wedge W_s$	$P_b \succ P_f$
$M_g \wedge W_r$	$P_f \succ P_b$
$M_b \wedge W_s$	$P_f \succ P_b$
$M_b \wedge W_r$	$P_b \succ P_f$

P_b	$R_s \succ R_d$
P_f	$R_d \succ R_s$

Figure 3. An example of a CP-net named "My Weekend 1 ".

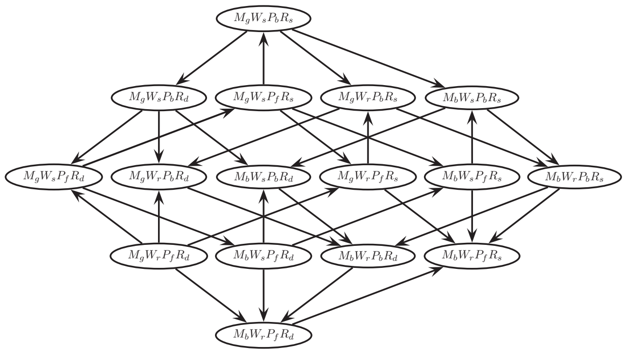


Figure 4. The induced graph of the CP-net "My Weekend 1".

Through the example above, we can see that the structure of the induced graph becomes more complex with the increase in the number of variables of CP-nets. At this moment, treewidth, as an important property, can naturally be used in the induced graphs of CP-nets.

3. Related Work

Given a set of attributes, CP-nets can not only express users' preferences, but also indicate the relationships between them. Pervious work has mainly focused on learning and reasoning CP-nets. For example, how to express a user's preferences in database queries [19], how to solve the expressive power on two kinds of specific CP-nets [2]. Guerin [20] presented an online heuristic algorithm for learning CP-nets from user queries, however, their algorithm does not always converge to the original CP-nets. Bigot *et al.* [21] studied how probabilistic conditional preference networks can be learnt, both in off-line and on-line settings. It is interesting to learn not a single model, but a probabilistic model that can compactly represent the preferences of a group of users. Liu *et al.* [22] studied conditional preference in recommender systems. Goldsmith *et al.* [23] researched the computational complexity of testing dominance and consistency in CP-nets, and proved that both dominance and consistency for general CP-nets are PSPACE-complete. Up to now, there has been little research investigating the structural properties of the induced graph of CP-nets, because its structure is rather complicated. As we know, the treewidth is interpreted as a significant property of arbitrary graphs in the study of Graph Minor. If the treewidth of a graph is obtained, we can perform some efficient reasoning over the induced graphs of CP-nets.

There are some papers working on determining treewidth and finding the optimal tree-decompositions. Given a graph G and an integer k, it is NP-complete to decide whether the treewidth of G is at most k or not [17]. Bodlaender *et al.* [24] proved that the treewidth problem is NP-complete for graphs with small maximum degree. The paper [25] gave a linear time algorithm that either outputs a tree-decomposition of G with treewidth at most k or determines that the treewidth of G is larger than k, for all constant k. For the Minimum Spanning Caterpillar Problem for bounded treewidth graphs, Dinneen and Khosravani [26] gave a linear time algorithm to solve it, but they assume that an approximate tree-decomposition of width bounded by a constant is given. Moreover, the treewidth problem of cographs [27], circular arc graphs [28], and chordal bipartite graphs [29], are polynomial solvable. However, so far, there are no papers to research the character of treewidth of the induced graphs of CP-nets.

As we all know, treewidth always has a parameterized complexity result on graphical models. Koster *et al.* [30] studied how to use tree decomposition to solve combinatorial optimization problems. In addition, Zhao *et al.* [31,32] published two papers that exploit tree decomposition to solve some problems about computational biology. In the area of graphical models, including knowledge representation and reasoning, probabilistic reasoning, and constraint satisfaction [33,34], all these studies focused on some graphs to represent abundant information about the real world. Since CP-nets can represent and reason users' preference represented by graphical model, treewidth can be applied in CP-nets naturally. In this paper, we study the treewidth problem on the induced graphs of CP-nets. Through the experiment in Section 5.2, we believe that any treewidth of the induced graph of a CP-net is small.

From Boutilier *et al.*'s paper [1], we know that there are four different paths from the outcome $\bar{a}\bar{b}c$ to the outcome abc in the induced graph. Our purpose is to find an optimal flipping sequence for $abc \succ \bar{a}\bar{b}c$; that is, to find the shortest path from these four paths. Fortunately, we can compute all-pairs shortest paths in induced graphs by leveraging low treewidth [35]. Since we have found that the treewidth of the induced graphs of CP-nets is small, we can quickly find the shortest path (optimal flipping sequence) in the induced graph. Therefore, solving dominance queries on the CP-nets is efficient.

4. Treewidth of the Induced Graphs of CP-nets

In this section, we mainly introduce the process of *Bucket Elimination* to solve treewidth, and a detailed algorithm pseudo-code.

4.1. Bucket Elimination Technique

The so-called elimination means that the vertices in a undirected graph are continuously eliminated, and how to get the *vertex elimination order* over the vertices is crucial in solving the treewidth. The notion of finding optimal vertex elimination orders is based on Bertele and Brioschi's [36] work on non-serial dynamic programming. This idea plays a role in a variety of algorithms in graphical models.

We begin with a series of definitions to explain the notion of *Bucket Elimination* in detail. First of all, we define the operation of eliminating a vertex from a graph as the process of adding an age between every pair of a vertex's neighbours that are not already adjacent, then removing the vertex and all edges incident to it.

A *vertex elimination order* over the vertices in a graph is a total order. When we eliminate some vertices, a group of graphs come into being. Clearly, the vertex elimination order is an total order on the given graph with n vertices. We would count an elimination order by its width, with the underlying assumption that we gain the elimination order $\pi = (C, A, B, D, E)$. The width of an elimination order is defined as the maximum degree of any vertex when it is eliminated from the graph. For example, the width of π is the maximal of these values in Figure 5; namely, $width(\pi) = \max(4, 2, 2, 1, 0) = 4$. An elimination order can determine only one width, the number of total vertex elimination orders of a graph with n vertices is $n!$.

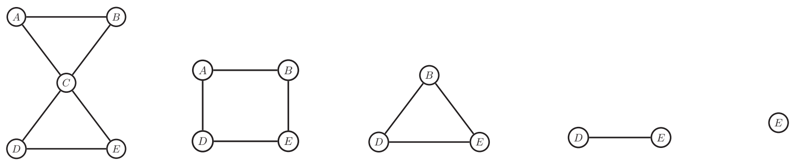


Figure 5. A five vertices graph for finding treewidth by eliminating vertices with the vertex elimination order $\pi = (C, A, B, D, E)$.

Importantly, the treewidth of a graph is the minimum width over all elimination orders, or, in other words, we must find an optimal vertex elimination order so that we can accurately conclude the treewidth of a graph. In Figure 6, the elimination order $\pi_{opt} = (A, B, C, D, E)$ is optimal. Notice that there is more than one optimal vertex elimination order. Finally, we have the following equation: $treewidth(G) = width(\pi_{opt}) = \max(2, 1, 2, 1, 0) = 2$.

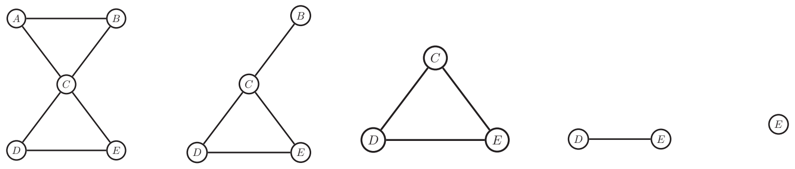


Figure 6. Finding the treewidth of a five vertices graph by eliminating vertices with the vertex elimination order $\pi_{opt} = (A, B, C, D, E)$.

We give the quantitative characterization of the number of attributes of CP-nets, the number of vertices of the induced graph, and the number of elimination orders in Figure 7. In fact, the elimination orders selected by our algorithm is a certain constant but exponential. Due to the fact that the induced graph of a CP-net is a special kind graph, and when the variable number is only 7, the number of total elimination orders is extremely large. So, we can not compute the width of all elimination orders to gain the treewidth of a graph in a practical application. According to the characteristics of randomness, we randomly select a certain number of vertex elimination orders to compute the treewidth of induced graphs. A simple example computing the treewidth of an induced graph with 4 vertices, is shown in Section 4.3.

$\|V\|$	$\|N\|$	$\|\pi\|$
1	2	2
2	4	24
3	8	40320
4	16	20922789888000
5	32	263130836933693530167218012160000000
6	64	126886932185884164103433389335161480802865516174545192198801894375214704230400000000000000
7	128	385620482362580421735677065923463640617493109590223590278828403276373402575165543560686168588507361534030051833058916347592172932262498857766114955245039357760034644709279247692495585280000000000000000000000000000000

Figure 7. $|V|$, $|N|$ and $|\pi|$ are the number of variables of CP-nets, vertices of induced graphs, and elimination orders, respectively.

While the number of vertices is large enough, the consumption of time and resources are also huge. However, there exists many specific graphs whose treewidth are constants or small [37]. Therefore, in practical application, we only need to solve the approximate optimal treewidth of graphs by some specific heuristic approach. Next, an example solving the treewidth of an induced graph will be given. Notice that, for the graph with five vertices in Figure 8, the number of total elimination orders is 120, Figure 9 only give one fifth of these 120 elimination order.

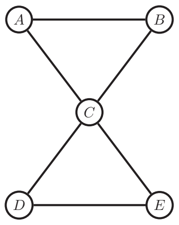


Figure 8. A graph on which we find the treewidth.

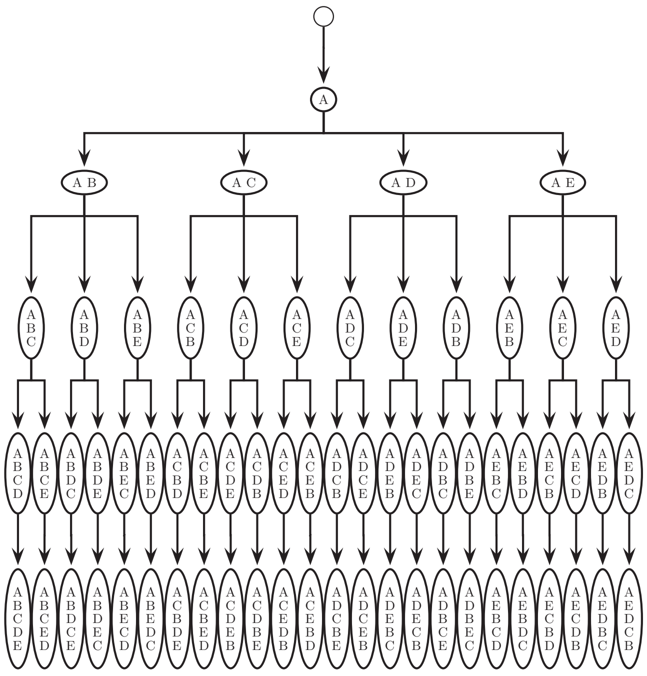


Figure 9. Fractional vertex elimination orders of Figure 8.

Given the number of elimination orders, we randomly select the vertices orders to solve the treewidth of this graph. In the next section, we describe this algorithm.

4.2. Algorithm for the Treewidth of an Induced Graph

Here we focus on our thought of the algorithm. First, Liu and Liao [2] gave the definition of induced graphs of CP-nets, which is derived from the CP-nets based on *ceteris paribus* semantics. So, it is assumed that, the induced graphs of CP-nets have not only been solved by the approach of literature [2] in advance, but has also been converted into an adjacency matrix of the corresponding underlying graph, so the adjacency matrix is the parameters of our algorithm. Since its vertex elimination orders are $n!$, in the program, we do not calculate the width of total vertex elimination orders. This reason has been explained in Section 4.1. Next, a certain amount of random vertex elimination orders are obtained. Then, this algorithm works by iterating through all sets of vertex elimination orders in order to find the approximate optimal treewidth. For instance, we in turn compute the degree of each vertex from the elimination order $\pi = (B, A, C, D, E)$. The function for solving the degree of the vertex is named $Deg()$. The adjacency matrix will be changed every time the degree of one vertex has been obtained, which will be accomplished in the $ChangeMatrix()$ function. At the same time, we take the maximum degree in the set of vertex elimination order as the width of this order. Finally, the minimum selected from all widths of vertex elimination orders is the treewidth of a graph. The pseudo-code of the proposed algorithm is summarized in Algorithm 1.

Algorithm 1: solving treewidth of included graphs of CP-nets N'.

```
   Input:  An adjacency matrix $A[i,j]$ of induced graph
   Output: The treewidth of $N'$
1  $treewidth \leftarrow 0$;
2  $Mdec \leftarrow 0$;
3  $mdec \leftarrow 0$;
   // Calculate the total number of vertex elimination order $sum$.
4  foreach $m \leftarrow 1$ to $sum$ do
       // Calculate the width of vertex elimination order in turn in term of the
          order produced randomly.
5      foreach $i \leftarrow 1$ to $N$ do
6          $mdec = Deg(i)$;
7          if $mdec > Mdec$ then
8              $Mdec \leftarrow mdec$;
9          end
10         $ChangeMatrix(i)$;
11     end
       // Find minimum width of all orders.
12     return $Treewidth$ ;
13 end
```

4.3. A Motivate Example

Now, we give a motivate example to demonstrate the solving procedure for Algorithm 1 in detail. In Figure 2, there are two binary-valued variables, and its induced graph has four vertices. For simplicity, let A, B, C, and D stand for these outcomes (W_sP_b, W_sP_f,W_rP_f, and W_rP_b), respectively. Initially, the induced graph was converted into a undirected graph stored in the 4 by 4 adjacency matrix. If there is a relationship between two vertices, the corresponding value in the adjacency matrix is set to 1. Otherwise, the value is set to 0, and the number of total elimination orders is 24. The program randomly selects half of the total amounts; namely, the practical number of elimination orders is 12. For example, $\pi_1 = (D, B, A, C)$ and $width(\pi_1) = \max(2, 2, 1, 0) = 2$, $\pi_4 = (B, A, D, C)$ and $width(\pi_4) = \max(2, 2, 1, 0) = 2$, and $\pi_{10} = (D, A, B, C)$, and $width(\pi_{10}) = \max(2, 2, 1, 0) = 2$, *etc.* Finally, the smallest value we work out as above is the treewidth of this induced graph. Table 1 shows the results with different elimination orders. Obviously, the treewidth is 2 in terms of our algorithm.

Table 1. The width in different elimination orders.

	π	$width$
π_1	$DBAC$	$max(2,2,1,0) = 2$
π_2	$ACDB$	$max(2,2,1,0) = 2$
π_3	$CDAB$	$max(2,2,1,0) = 2$
π_4	$BADC$	$max(2,2,1,0) = 2$
π_5	$DABC$	$max(2,2,1,0) = 2$
π_6	$CABD$	$max(2,2,1,0) = 2$
π_7	$CDAB$	$max(2,1,1,0) = 2$
π_8	$DACB$	$max(2,1,1,0) = 2$
π_9	$CBDA$	$max(2,1,1,0) = 2$
π_{10}	$DABC$	$max(2,1,1,0) = 2$
π_{11}	$BCAD$	$max(2,1,1,0) = 2$
π_{12}	$BACD$	$max(2,1,1,0) = 2$

In addition, the treewidth of induced graph of given CP-net is 5 in Figure 4 by Algorithm 1.

4.4. Time Complexity Analysis

Theorem 1. *The time complexity of Algorithm 1 is $O(c_1 n^2 + c_1 c_2 n)$.*

The variable n is the number of vertices of the induced graphs of CP-nets, c_1 is a certain constant of randomly generated but exponential elimination orders (please refer to Section 4.1 on how to choose c_1), and c_2 is the maximal in-degree of the induced graph.

Proof. In Algorithm 1, firstly, we arbitrary select some vertex elimination orders from c_1 orders. this has $\binom{c_1}{1} = c_1$ kinds of selection methods. Secondly, to compute the treewidth of the induced graph, we need to iterate through all the vertices set n. Then, we compute the degree of this vertex when eliminating each vertex, its time complexity is $O(n)$. At the same time, the change of adjacency matrix includes two parts. One iterates through all the vertices to judge whether there is a relationship between arbitrary vertex, the other forms a clique which consists of all the vertices having a relationship with the deleted vertex. The time complexity of this process is $O(n + c_2)$. Finally, the minimum selected from all widths of vertex elimination orders is the treewidth of the induced graph. Therefore, combined with the three steps, the execution time in Algorithm 1 is:

$$
\begin{aligned}
&O(c_1) \times [O(n) \times [O(n) + O(n + c_2)]] \\
&= O(c_1) \times [O(n) \times [O(2n) + O(c_2)]] \\
&= O(c_1)] \times [O(2n^2) + O(c_2 n)] \\
&= O(c_1 n^2) + O(c_1 c_2 n) \\
&= O(c_1 n^2 + c_1 c_2 n).
\end{aligned}
$$

According to the above analysis, the time complexity of Algorithm 1 is $O(c_1 n^2 + c_1 c_2 n)$.
□

5. Experimental Evaluation

5.1. Experimental Environment

Our algorithm was implemented in C++ and went through an intensive profiling phase. The experiments were run using Microsoft Visual Studio 2008, on a Lenovo V480 running 64-bit Windows 8 equipped with 4GB DDR3 memory and dominant frequency of 2.4GHz i5-Intel CPU. The run times are averaged over 10 runs for each unique problem instance. Moreover, we generated several different instances for each given induced graph model. Finally, each induced graph instances was ensured to compute the accurate treewidth.

5.2. Experimental Results on Simulated Data

In this section, we comment on the experiments we have carried out for the *Bucket Elimination* algorithm for computing the treewidth of the given induced graph in Table 2. Firstly, we define the size $|E_1|$ of a induced graph N' as the total number of edges in N', and we already know that $|E_1| = n|\Omega|/2$ by Definition 4. Moreover, we also define the size $|E_2|$ of a graph with n vertices as the total number of edges.

$$|E_2| = n(n-1)/2 \tag{1}$$

Table 3 shows the relationship between V, n, $|E_1|$, and $|E_2|$, where V and n denote the size of the variables of CP-nets and of the vertices of the induced graph, respectively. The size of $|E_2|$ is much more than $|E_1|$'s; thus, when computing the treewidth of the induced graphs of CP-nets, we need not consider too many edges. Apparently, it improves the running efficiency of the program.

Table 2. Experimental results of the induced graphs of CP-nets.

n	8	16	32	64	128	256	512	1024
$\|E\|$	12	32	80	192	448	1024	2304	5120
tw	1	2	6	7	8	9	9	10
$time(s)$	0.016	0.031	0.094	0.313	1.103	4.064	15.102	61.057

Table 3. The relationship between V, n, $|E_1|$, and $|E_2|$.

V	n	$\|E_1\|$	$\|E_2\|$
2	4	4	6
3	8	12	28
4	16	32	128
5	32	80	496
6	64	192	2016
7	128	448	8128

Table 2 lists the number of test cases, the range of the number of vertices n, edges $|E|$, *time*, as well as induced graphs with treewidth tw produced by Algorithm 1. More details on the different sets can be found below, but one thing that stands out immediately is that when the number of edges is different in the same induced graph, treewidth is not necessarily equal, and with an increase in the number of edges, treewidth gradually becomes larger. In Figure 10, it is more intuitive to express the relationship between the treewidth of an induced graph and the number of edges in the induced graph with 16 vertices and different edges.

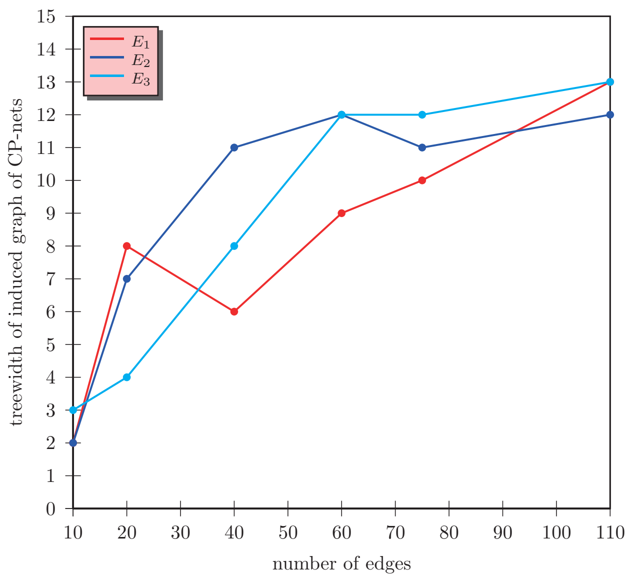


Figure 10. Treewidth corresponding to different edges in a given induced graph with 16 vertices.

For the induced graph of a CP-net with 1024 vertices, though we have randomly generated ten thousand elimination orders to solve treewidth, it is far less than the total elimination orders. Therefore, in this situation, the treewidth of the induced graph of a CP-net calculated by our algorithm could not be regarded as the accurate value, but it can be regarded as the upper bound of treewidth. It also comes to the conclusion that the treewidth of the induced graphs of CP-nets is still small when it has too many vertices.

6. Conclusions and Future Work

We know that some search algorithms can be used to traverse the improving search tree to find an improving sequence that supports dominance queries [1]— it means that we should find an optimal path over the induced graph. However, when the treewidth of a graph is small, an optimal path will be easier to find. Just as we have shown that the treewidth of induced graphs of CP-nets is much smaller in Section 5.2, so, dominance queries may be solved efficiently.

In this paper, we use a *Bucket Elimination* to compute the treewidth of the induced graphs of CP-nets. Although the time complexity of a *Bucket Elimination* is exponential in the treewidth of the given induced graphs of CP-nets, we randomly select a certain number of vertex elimination orders according to the characteristics of randomness, and the number of edges of the induced graphs of CP-nets is finite, which not only improves running time but also computes accurate treewidth of the induced graphs of CP-nets. Also, our research will make some operations over the induced graphs of CP-nets more feasible.

As everyone knows, program parallelization is getting more and more attention in recent years. Running on a multi-core computer is obviously much faster than on a single core. Then, we would implement the algorithm in parallel to improve the efficiency of computing the treewidth of the induced graphs of CP-nets in the near future.

Recently, a dynamic programming of the style of the classic Held-Karp algorithm has been given for the *Traveling Salesman Problem* [18], and Shoikhet and Geiger [38] have suggested that an algorithm of Arnborg *et al.* [38] can be used to compute treewidth. This algorithm builds a tree decomposition, and also leverages the dynamic programming idea. We would also like to compare our algorithms to the above dynamic algorithms experimentally in future work. Furthermore, in this paper, the induced graph of CP-net has been converted to undirected graph to compute its treewidth.

Our research would pay attention to how to compute the treewidth of directed induced graphs of CP-net—this will also be an interesting question in the future.

Acknowledgments: We gratefully acknowledge the detailed and helpful comments of the anonymous reviewers, who have enabled us to considerably improve this paper. This work was supported by Natural Science Foundation of China (61572419, 61403328, 61403329), and Shandong Province Natural Science Foundation (ZR2013FM011, 2015GSF115009, ZR2014FQ016, ZR2014FQ026).

Author Contributions: Jinglei Liu was the leader of this work and proposed the idea to solve the treewidth of induced graph of CP-nets. Jie Liu assisted with the data preprocessing, conducted the experiments, and prepared the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Boutilier, C.; Brafman, R.; Domshlak, C.; Hoos, H.; Poole, D. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res.* **2004**, *21*, 135–191.
2. Liu, J.L.; Liao, S.Z. Expressive efficiency of two kinds of specific CP-nets. *Inf. Sci.* **2015**, *295*, 379–394.
3. Majid, Z.A.L.; Emrouznejad, A.; Mustafa, A.; Al-Eraqi, A.S. Aggregating preference ranking with fuzzy Data Envelopment Analysis. *Knowl. Based Syst.* **2010**, *23*, 512–519.
4. Chen, H.; Zhou, L.; Han, B. On compatibility of uncertain additive linguistic preference relations and its application in the group decision making. *Knowl. Based Syst.* **2011**, *24*, 816–823.
5. Ha, V.; Haddawy, P. Toward Case-Based Preference Elicitation: Similarity Measures on Preference Structures. In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Madison, WI, USA, 24–26 July 1998; pp. 193–201.
6. Mindolin, D.; Chomicki, J. Contracting preference relations for database applications. *Artif. Intell.* **2009**, *175*, 1092–1121.
7. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W.H. Freeman: San Francisco, CA, USA, 1979.
8. Downey, R.G.; Fellows, M.R. *Parameterized Complexity*; Springer: Berlin/Heidelberg, Germany, 2012.
9. Chen, J.E. Parameterized computation and complexity: A new approach dealing with NP-hardness. *J. Comput. Sci. Technol.* **2005**, *20*, 18–37.
10. Robertson, N.; Seymour, P.D. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms* **1986**, *7*, 309–322.
11. Mattei, N.; Pini, M.S.; Rossi, F.; Venable, K.B. Bribery in voting with CP-nets. *Ann. Math. Artif. Intell.* **2013**, *68*, 135–160.
12. Dechter, R. Bucket elimination: A unifying framework for probabilistic inference. In *Learning in Graphical Models*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 75–104.
13. Darwiche, A. *Modeling and Reasoning With Bayesian Networks*; Cambridge University Press: Cambridge, UK, 2009.
14. Kosowski, A.; Li, B.; Nisse, N.; Suchan, K. k-chordal graphs: From cops and robber to compact routing via treewidth. In *Automata, Languages, and Programming*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 610–622.
15. Dow, P.A. Search Algorithms for Exact Treewidth. Ph.D. Thesis, University of California Los Angeles, Los Angeles, CA, USA, 2010.
16. Gao, W.Y.; Li, S.H. Tree decomposition and its Application in Algorithm: Survey. *Comput. Sci.* **2012**, *39*, 14–18.
17. Arnborg, S.; Corneil, D.G.; Proskurowski, A. Complexity of finding embeddings in ak-tree. *SIAM J. Algebraic Discret. Methods* **1987**, *8*, 277–284.
18. Bodlaender, H.L.; Fomin, F.V.; Koster, A.M.; Kratsch, D.; Thilikos, D.M. *On Exact Algorithms for Treewidth*; Springer: Berlin/Heidelberg, Germany, 2006.
19. Kießling, W. Foundations of preferences in database systems. In Proceedings of the 28th International Conference on Very Large Data Bases, Hong Kong, China, 20–23 August 2002; pp. 311–322.
20. Guerin, J.T.; Allen, T.E.; Goldsmith, J. Learning CP-net Preferences online from user queries. In *Algorithmic Decision Theory*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 208–220.

21. Bigot, D.; Mengin, J.; Zanuttini, B. Learning probabilistic CP-nets from observations of optimal items. In Proceedings of the 7th European Starting AI Researcher Symposium, Prague, Czech Republic, 18–19 August 2014; IOS Press: Amsterdam, The Netherlands, 2014; pp. 81–90.
22. Liu, W.Y.; Wu, C.H.; Feng, B.; Liu, J.T. Conditional preference in recommender systems. *Expert Syst. Appl.* **2015**, *42*, 774–788.
23. Goldsmith, J.; Lang, J.; Truszczynski, M.; Wilson, N. The computational complexity of dominance and consistency in CP-nets. *J. Artif. Intell. Res.* **2008**, *33*, 403–432.
24. Bodlaender, H.L.; Thilikos, D.M. Treewidth for graphs with small chordality. *Discret. Appl. Math.* **1997**, *79*, 45–61.
25. Bodlaender, H.L. A linear time algorithm for finding tree-decompositions of small treewidth. In Proceedings of the 25th Annual ACM Symposium on Theory of Computing, San Diego, CA, USA, 16–18 May 1993; ACM: New York, NY, USA, 1993; pp. 226–234.
26. Dinneen, M.J.; Khosravani, M. A linear time algorithm for the minimum spanning caterpillar problem for bounded treewidth graphs. In *Structural Information and Communication Complexity*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 237–246.
27. Bodlaender, H.L.; Möhring, R.H. The pathwidth and treewidth of cographs. *SIAM J. Discret. Math.* **1993**, *6*, 181–188.
28. Sundaram, R.; Singh, K.S.; Rangan, C.P. Treewidth of circular-arc graphs. *SIAM J. Discret. Math.* **1994**, *7*, 647–655.
29. Kloks, T.; Kratsch, D. Treewidth of chordal bipartite graphs. *J. Algorithms* **1995**, *19*, 266–281.
30. Koster, A.M.; van Hoesel, S.P.; Kolen, A.W. Solving partial constraint satisfaction problems with tree decomposition. *Networks* **2002**, *40*, 170–180.
31. Zhao, J.; Che, D.; Cai, L. Comparative pathway annotation with protein-DNA interaction and operon information via graph tree decomposition. *Pac. Symp. Biocomput.* **2007**, *12*, 496–507.
32. Zhao, J.; Malmberg, R.L.; Cai, L. Rapid ab initio prediction of RNA pseudoknots via graph tree decomposition. *J. Math. Biol.* **2008**, *56*, 145–159.
33. De Givry, S.; Schiex, T.; Verfaillie, G. Exploiting tree decomposition and soft local consistency in weighted CSP. In Proceedings of the 21st National Conference on Artificial Intelligence, Boston, MA, USA, 16–20 July 2006; Volume 6, pp. 1–6.
34. Jégou, P.; Ndiaye, S.; Terrioux, C. Dynamic Heuristics for Backtrack Search on Tree-Decomposition of CSPs. In Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, 6–12 January 2007; pp. 112–117.
35. Planken, L.; de Weerdt, M.; van der Krogt, R. Computing All-Pairs Shortest Paths by Leveraging Low Treewidth. *J. Artif. Intell. Res.* **2012**, *43*, 353–388.
36. Bertele, U.; Brioschi, F. *Nonserial Dynamic Programming*; Elsevier: Amsterdam, The Netherlands, 1972.
37. Zhang, C.; Naughton, J.; DeWitt, D.; Luo, Q.; Lohman, G. On supporting containment queries in relational database management systems. In Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, Santa Barbara, CA, USA, 21–24 May 2001; Volume 30, pp. 425–436.
38. Shoikhet, K.; Geiger, D. A practical algorithm for finding optimal triangulations. In Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, Providence, RI, USA, 27–31 July 1997; pp. 185–190.