Expert Systems With Applications 86 (2017) 32–41

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Operators of preference composition for CP-nets

Xuejiao Sun*, Jinglei Liu, Kai Wang

School of Computer and Control Engineering, Yantai University, Yantai 264005, PR China

ARTICLE INFO

Article history:
Received 25 November 2016
Revised 20 April 2017
Accepted 16 May 2017
Available online 22 May 2017

Keywords:
Condition preference networks (CP-nets)
Preference composition
Preference query
Satisfiability sequence
Relational database

ABSTRACT

Now the handling of user preference is becoming an increasingly important issue in database fields where they capture soft criteria for queries. A broader category of qualitative preferences with dependent relations among multiple attributes is widely existing, which is CP-nets. In this article, we focus on designing the operators of preference composition for CP-nets. Firstly, we extend Pareto composition to our model by including equivalence relation $\approx$, incomparability relation $\|$ and conflicting relation $\perp$, which can preserve a strict partial order and conditional associativity. On this basis, two questions are solved: (a) the generation of satisfiability sequences for CP-nets, (b) the *top-k* queries of relational database with CP-nets preference. For (a), a CP-net is induced into multiple tables, consequently the strong dominance tests between outcomes can be solved by using preference composition instead of using induced preference graph of CP-nets. For (b), we adopt the concept of Query Lattice to provide a natural semantics for the block sequence answering a preference query, where two algorithms (called *QOCP* and *IQOCP*) are introduced. These questions are solved efficiently and effectively at the perspective of combination of graph model and relational database.

1. Introduction

Recently, the research on preferences has attracted broader interest in the database research community. Users would like to describe features of data that are potentially useful in some tasks, or in other words features that best suit their preferences. Modern database systems should then be able to process queries enhanced with preferences, and such queries are called preference queries (Arvanitis & Koutrika, 2012). There are already promising applications of preference queries in the area of personalized search engines and recommender systems (Liu, Wu, Feng, & Liu, 2015). Since in traditional databases or search engines, the user's preferences were not considered, where query conditions are considered as hard by default and a nonempty answer is returned only if it satisfies all query criteria. For example, the following database query clause to SQL from the relation *Dinner*

Where Drink='wine' and Staple='steak' expresses a very clear and specific query condition. It indicates that the user must know the complete information of the query object. Therefore, in this context, the user can face either of two problems: (1) the data can't exactly match the query (query criteria are too restrictive), so the result is empty set or (2) there are too many answers (query criteria are too weak). In fact, the user's preferences usually are vague, for example, I would rather express the following preferences over my dinner configurations:

"I prefer red wine to white wine if served a steak,
and white wine if served some seafood".

Where the constraint degree is no longer 0 or 1, it is described as soft constraint, and all outcomes are sorted according to how closely they match the query conditions, called "Personalized Query" of relational database (Arvanitis & Koutrika, 2012).

In the preceding work of preference applications, the preference modeling is central. There are quite a few approaches in the literature that deal with preference representation and try to reach meaningful conclusions regarding the desired answers of a database query from different perspectives. However, to the best of our knowledge, an important category of preference models is missing, for preferences hold unconditionally in the database query with preference studied so far, namely, there are no serious attempts made to realize queries of preference with dependencies (called conditional dependence preferences), which is necessary, because in reality there are two features in the decision making system:

(1) Multiple attributes. The decision objects are usually not described so simply, and preferences may be represented as a set of constraints over a set of decision variables. For example, when a user wants to buy a car, his preferences focus here on various attributes: make, type, price, color,

* Corresponding author.
E-mail addresses: sunxuejiao6@sina.com (X. Sun), jinglei_liu@sina.com (J. Liu), wangkai_bw@163.com (K. Wang).

http://dx.doi.org/10.1016/j.eswa.2017.05.040
0957-4174/© 2017 Elsevier Ltd. All rights reserved.

etc., and the degrees and orders of concerns over preference attributes are different, which will lead to different final selections.

(2) Conditional dependence preferences. The user preference for one attribute can be impacted by other attributes. For instance, the selection of drink type depends on the matching staple food, the type of the car determines the selection of its color.

Not only in daily life, but also in social choice, database query, collaborative filtering system and recommendation system, the qualitative preferences with dependencies among multiple attributes are widely existing. Thus, the way to describe a broader category of preferences is required. A commonly used graphical notation for their representation is CP-nets in artificial intelligence. CP-nets (Conditional Preference networks) can be used to describe the implied qualitative preference relations in the situational preference information with a relatively tight, intuitive, structured qualitative preference. The main advantage of this tool is that it is a qualitative graphical tool, and it can reflect the dependencies among preferences.

Observing previous studies on CP-nets were either based on induced preference graph of CP-nets (e.g., (Liu, Liao, & Zhang, 2012) solved the strong dominance test by Warshall algorithm and (Sun & Liu, 2015) presented the generation algorithm of satisfiability sequences) or by means of quantitative methods (e.g., (Mindolin & Chomicki, 2007)), clearly, there are some defects in these ways. As we all know, getting an induced preference graph itself is a very difficult work, involving its storage, construction, properties, and so forth. There are few works about these and no corresponding efficient generation algorithm is introduced so far, and classical methods for them result in exponential computational complexities and (Goldsmith, Lang, Truszczynski, & Wilson, 2008) showed that dominance test in CP-nets is PSPACE-complete. As to solving questions in qualitative model by using quantitative methods, it is a concession or unwilling choice, since the expressive power of quantitative way is weaker than qualitative one and, thus, will lose a lot of information.

Instead, our work is different from existing approaches on executing and optimizing queries with preferences, we study preference query based on CP-nets from the perspective of graph model. In our opinion, there is significant relationship between CP-nets and relational database, CP-nets are divided and induced accordingly into multiple database tables. Thus we extend the Pareto composition in this work (Section 4). We realize the generation of CP-nets satisfiability sequences by means of preference composition theory closer to (Georgiadis, Kapantaidakis, Christophides, Nguer, & Spyratos, 2008), instead of using induced preference graph (Sun & Liu, 2015) (Section 5). Furthermore, we also aim to realize the top-k queries in the context of databases with CP-nets preference, that is, to retrieve the best k outcomes (or tuples) from tables, in the most extreme cases, which is the Cartesian product of all the attribute domains (Section 6).

In summary, the main contributions of this article are as follows:

(1) An extended Pareto composition mechanism to CP-nets, which can preserve a strict partial order composition result and conditional associativity.
(2) How to store a CP-net (not mentioned in current CP-nets studies) is solved by inducing it into multiple database tables.
(3) An efficient means of computing the strong dominance test (i.e., to perform preferential comparison between outcomes) relying not on the induced preference graph but on the preference graph itself and the extended preference composition mechanism.
(4) We realize the top-k queries of relational database with conditional ceteris paribus preferences. The algorithms (*QOCP* and *IQOCP*) are proposed on the basis of query-rewriting algorithm, named *LBA* (Georgiadis, Kapantaidakis, Christophides, Nguer, & Spyratos, 2008) by adding two new operators: the composition operator of non-dependent relations and the composition operator of dependent relations.

2. Related work

Much work has gone into embedding the notion of preference in database systems from both the modeling and implementation aspects. Database also bring a whole fresh perspective to the study of preferences, both computational and representational (Stefanidis, Koutrika, & Pitoura, 2011). From a representational perspective, the key question is how to represent and incorporate preferences in database query. Chomicki (2003) and Kießling (2002) studied preference relations between database tuples using logical formulas or preference constructors, which are qualitative approaches; whereas Agrawal and Wimmers (2000) and Koutrika and Ioannidis (2004) specified preferences using scoring functions, every database tuple is assigned a numerical score, which is quantitative approach. However, not every intuitively plausible preference relation can be captured by scoring functions, so in terms of expressive power, the qualitative specification of preferences is more general than the quantitative one (Abbas & Bell, 2012; Choo, Schoner, & Wedley, 1999; Stefanidis, Koutrika, & Pitoura, 2011). From a computational perspective, the key question is how to efficiently process preferences in database queries. One way is by expanding regular database queries with conditions that express a predefined set of user preferences (user profile), adopted by van Bunningen, Feng, and Apers (2006); Georgiadis, Kapantaidakis, Christophides, Nguer, and Spyratos (2008); Kostas Stefanidis (2007); Miele, Quintarelli, and Tanca (2009), etc. Another way is by extending query languages with preference operators, such as, the Winnow operator (Chomicki, 2003; Torlone & Ciaccia, 2002a), and Preference Selection operator (Kießling, 2002), and so on.

For CP-nets, some detailed syntaxes, semantics, and applications have been described by Boutilier, Brafman, Domshlak, Hoos, and Poole (2004) and Goldsmith, Lang, Truszczynski, and Wilson (2008). Recently, there have also some researches on subclasses of CP-nets: TCP-nets (Tradeoff-enhanced CP-nets) (Brafman, Domshlak, & Shimony, 2006) and CI-nets (Conditional Important networks) (Bouveret, Endriss, & Lang, 2009), which mainly add some elements reflecting the importance of attributes. Cornelio, Goldsmith, Mattei, Rossi, and Venable (2013) presented a two-fold generalization of conditional preference networks that incorporates uncertainty. Ciaccia (2007) programmed a new SQL statement by using CP-nets as the preference database query language; Endres and Kießling (2006) realized how to turn the query of TCP-nets to a database query; Santhanam, Basu, and Honavar (2010) presented an earlier algorithm for testing dominance in TCP-nets by encoding the TCP-net semantics (i.e., the induced preference graph) into a Kripke structure and then analyzing the Kripke structure using a model checker. Bosc, Hadjali, and Pivert (2011) investigated how to realize preference query with CP-nets and other weighted Boolean query. As a model of qualitative preference, there are also some reasoning tasks, such as satisfiability, consistency, connectivity, boundedness, etc. Liu (2011); Liu and Liao (2015); Liu, Liao, and Zhang (2012) analyzed the expressive power of CP-nets, proved completeness and consistency theorems. Sun and Liu (2012) concluded that the consistency is equal to the satisfiability of CP-nets and did the consistency reasoning. There are also a few recent proposals of using CP-nets to represent database preferences, such as Hierarchical CP-nets extended CP-nets by defining different

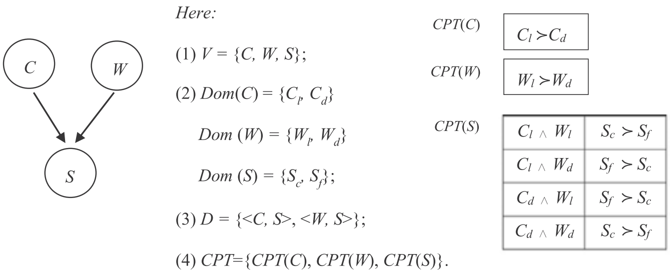


Fig. 1. CP-net for “My Living Room Decoration” .

priorities over descendant and ancestor attributes in Mindolin and Chomicki (2007). Ciaccia (2007) considered incomplete CP-nets where preferences were only partially specified resulting in pairs of tuples being incomparable or indifferent in some contexts. Endres and Kießling (2006) translated CP-nets into expressions in the formal preference language over strict partial orders.

3. Basic definitions

Definition 1. Given an attribute set $V = \{X_1 \cdots X_n\}$ over which the user has preferences, such that D_i, $1 \leq i \leq n$, is the domain of the attribute, the decision space $\Omega = D_1 \times \cdots \times D_n$ is a Cartesian product of all attribute domains, and if $o \in \Omega$, o is an outcome.

For o and o', if there is only one attribute value is different and all others are the same, they are called swap outcomes. And if there exist dependency relationships among these attributes, namely the user preference for X_i depends on attribute set X_s, we say that X_s is the father of X_i, denoted as $pa(X_i) = X_s$.

Definition 2. A relation $\succ$ is a strict partial order over Ω, if and only if, $\succ$ is anti-symmetric $(\forall o,\ o')(o,\ o' \in \Omega \wedge o \succ o' \wedge o' \neq o \rightarrow o' \nsucc o)$, and transitive $(\forall o,\ o',\ o'')(o,\ o',\ o'' \in \Omega \wedge o \succ o' \wedge o' \succ o'' \rightarrow o \succ o'')$.

Given a pair of outcomes o and o', if the relation $o \succ o'$ holds, we say that outcome o is preferred to o'; $o \| o'$ (i.e., o and o' are incomparable for lack of more information) iff, for $o,\ o' \in \Omega$, $o \nsucc o'$ and $o' \nsucc o$ hold. When $o \succ o'$ or $o \| o'$, it is written as $o \succ_N o'$. $\succ_N$ is the preference set expressed by CP-nets.

Definition 3. A CP-net is a directed graph C, formally defined as the quadruple:

$$C =< V, Dom, D, CPT > .$$

Where V is the set of attributes (vertices). Dom is the set of value domains for all attributes in V, for $X_i \in V$, $Dom(X_i)$ is the value domain of X_i, D is the dependency set (directed edges, i.e., for an directed edge, the values of starting point affect the preference of end point) among V. CPT is the set of conditional preference tables for all attributes in V, i.e., for $X_i \in V$, $CPT(X_i)$ is associates a total order $\succ_u^i$ with each instantiation u of X_i's parents $Pa(X_i) = U$, denoting under the different assignments of $pa(X_i)$, the user's preference orderings for $Dom(X_i)$. To simplify the work, we only discuss binary CP-nets (i.e., $|Dom(X_i)| = 2$) in this article.

Example 1. (My Living Room Decoration) Consider the simple CP-net in Fig. 1 that expresses my living room configurations. This network consists of three variables C, W and S, standing for the curtain, wall color and sofa, respectively. Now, I strictly prefer light color (C_l, W_l) to dark color (C_d, W_d) for curtain and wall, while my preference between coriaceous sofa (S_c) and fabric sofa (S_f) is conditioned on the curtain and wall selected: I prefer coriaceous sofa if the same color for curtain and wall selected, and fabric sofa if different colors selected for them. Fig. 1 shows the CP-net $C =< V, Dom, D, CPT >$.

Definition 4. Consider a CP-net C, the directed graph $G = \langle \Omega, IE \rangle$ is the preference graph over outcomes induced by C (i.e., induced preference graph), of which, Ω is the set of outcomes (vertexes), IE is the set of directed edges composed of all the swap outcomes such that $o \succ o'$ holds.

Note that for a binary CP-net with n variables, in its corresponding induced preference graph, there are 2^n vertices and $n * 2^{n-1}$ edges. Therefore, it is a hard work to obtain the induced preference graph, let alone use it to solve other problems in CP-nets.

4. Preference composition

In order to facilitate the next work, we define some preference relations over attribute domains, that is, a preference over an attribute A is defined by a preference expression P_A. Such statements can actually define binary relations in example 1 as follows:

(1) P_C: $pa(C) = \emptyset$, C_l is preferred to C_d;
(2) P_W: $pa(W) = \emptyset$, W_l is preferred to W_d;
(3) P_S: $pa(S) = (C, W)$, the preferred relations between S_c and S_f depend on the values of C and W.

Given a set of preferences, preference composition seeks to combine them. The preferences to be composed correspond to variant preferences of a user. With the increasing popularity of social networks, composition is central for the success of personalization. One of these qualitative mechanisms for composing preferences is Pareto preference composition (Stefanidis, Koutrika, & Pitoura, 2011), however, it is not associative when used in an n-ary composition, $n > 2$. Thus we extend the Pareto composition to our model as follows:

Definition 5. Given two preference relations P_X, P_Y over a CP-net, we define an induced relation $\succ P_{XY} = \ \succ P_X \ \otimes \ \succ P_Y$ over $Dom(X) \times Dom(Y)$, as:

$$(x, y) \succ_{XY} (x', y')\ \textit{iff}\ (x \nsucceq_X x' \wedge y \succ_Y y') \vee (x \succ_X x' \wedge y \nsucceq_Y y')$$
$$(x, y) \approx_{XY} (x', y')\ \textit{iff}\ (x \approx_X x') \wedge (y \approx_Y y')$$
$$(x, y)\ \|_{XY}\ (x', y')\ \textit{iff}\ (x\ \|_X\ x') \vee (y\ \|_Y\ y')$$
$$(x, y) \perp_{XY} (x', y')\ \textit{otherwise}.$$

In which, $\succeq$ denotes the ”at least as preferable” relation (Georgiadis, Kapantaidakis, Christophides, Nguer, & Spyratos, 2008), including $\succ$ and $\approx$. $\approx$ is an equal preference relation, in our study, $x \approx x'$ only means that x and x' are the same value for the attribute X. $\|$ denotes a kind of uncertain preference relation, $o \| o'$ means that we can not now compare the two outcomes (o and o') owing to lack of more information, in other words, we can not determine whether $o \succ o'$ or $o' \succ o$. At the same time, it is an unstable relation, since $o \| o'$ may be converted to $o \succ o'$ or $o' \succ o$ by taking the transitive closure of the direct relation $\succ$. While $\perp$ means a kind of conflicting relation, for the composition outcome o, preference of some attributes is prior to o', whereas preference of others is inferior to o'. Here our choice relies solely on partial orders, without any further assumptions.

Theorem 1. *This composition operator can preserve a strict partial order composition result.*

Proof. There are three kinds of preference relations $\succ$, $\|$ and $\perp$. They all satisfy the properties of a strict partial order: irreflexivity and transitivity. Therefore, strict partial order is preserved by extending Pareto composition. □

Theorem 2. *The associativity can also be guaranteed, if there are no dependencies among these preference attributes to be composed.*

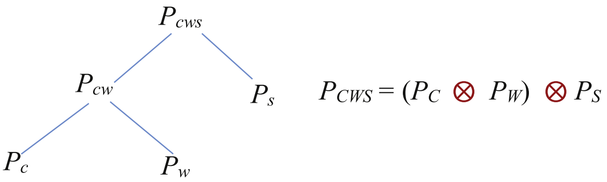


Fig. 2. Preference expression tree and composition operator.

Proof. Clearly, for preference relations P_X, P_Y, P_Z, if there are no dependencies among them, the following formula holds:

$(\succ P_X \otimes \succ P_Y) \otimes \ \succ P_Z \equiv \succ P_X \otimes (\succ P_Y \otimes \ \succ P_Z)$.

For example, consider preferences P_X, P_Y, P_Z, suppose we first apply Definition 5 on X with $x \succ_X x'$ and Y with $y \succ_Y y'$, the result would be $(x,\ y) \succ_{XY} (x',\ y')$, then we went on to compose this intermediate result with Z $(z \succ_Z z')$, the final result would be $(x,\ y,\ z) \succ_{XYZ} (x',\ y',\ z')$. Now we first composed P_Y and P_Z, then went on to compose this intermediate result with P_X, the final result would also be $(x,\ y,\ z) \succ_{XYZ} (x',\ y',\ z')$. it's similar for other composition relations, therefore, in this case, the associativity can be guaranteed.

However, for CP-nets, there usually exist dependencies among preference relations. For example, assuming $pa(Y) = X$, $pa(Z) = Y$, we can not compose P_Y, P_Z firstly. Therefore, the above formula does not hold.

Based on the above, we can draw a conclusion: For CP-nets preferences, the associativity can be guaranteed by the extending Pareto composition operator, if there are no dependencies among these preference attributes to be composed. Therefore, we call it as "conditional associativity". □

Associativity and closure of preorders under the extended Pareto composition enable a bottom-up evaluation of arbitrary preference expressions. In order to compose CP-nets preferences, the rule of composition is defined as:

Rule 1. All attributes are composed in this order: attributes without parents are finished firstly, and then, in turn, compose the attribute whose parent have been composed, until all attributes are finished. At the end of each composition, all the $\|$ relations should be adjusted according to the transitivity of other strict partial relations, namely $\succ$.

According to the extended preference composition mechanisms and rule 1, we define the preference expression tree and the preference composition operator $\otimes$ to carry out the pairwise compositions from leaf to root just like shown in Fig. 2.

5. CP-Nets satisfiability sequence

5.1. Definition of satisfiability sequence

As a language of preference representation, the natural question is whether a CP-net is satisfiable, that is, whether there exists an outcomes flip sequence satisfying all the preference statements.

Definition 6. Let Ω be the decision space of a CP-net C with n vertices, C is satisfiable iff there exists at least one collating sequence $\succ_N$ including all outcomes in Ω such that:

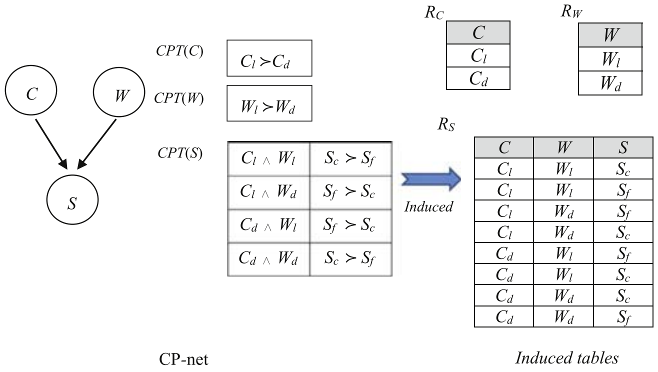
$o_1 \succ_N o_2 \succ_N o_3 \succ_N \cdots \succ_N o_i \cdots \succ_N o_k$.

In which, $o_i \in \Omega, k = 2^n$, this sequence is called satisfiability sequence l.

It is a well-known result that every acyclic CP-net is satisfiable, whereas the satisfiability of a cyclic CP-net can't be guaranteed, and for a satisfiable CP-net, the satisfiability sequence is not unique (Sun & Liu, 2012), $l \in L$, L is the set of all satisfiability sequences for a CP-net. It is necessary to obtain these sequences, especially it is the crux of the recommendation systems and personalized database query systems.

Table 1
Correspondences between CP-nets and relation tables .

CP-nets	Relation tables	Symbolism
Attribute	Field	$X_1, X_2, \cdots, X_n$
Attribute set	Field set	V
Domain of attribute X_i	Domain of field X_i	$Dom(X_i)$
All outcomes O	Tuple set	$O = Dom(V)$
Feasible outcomes O'	Relation	$O' \subseteq Dom(V)$



$C_l \wedge W_l$	$S_c \succ S_f$
$C_l \wedge W_d$	$S_f \succ S_c$
$C_d \wedge W_l$	$S_f \succ S_c$
$C_d \wedge W_d$	$S_c \succ S_f$

C
C_l
C_d

W
W_l
W_d

C	W	S
C_l	W_l	S_c
C_l	W_l	S_f
C_l	W_d	S_f
C_l	W_d	S_c
C_d	W_l	S_f
C_d	W_l	S_c
C_d	W_d	S_c
C_d	W_d	S_f

Fig. 3. A CP-net is induced into tables.

5.2. Induced tables

In the following, we shall present a new approach to generate the satisfiability sequences by preference composition mechanisms. In fact, observing CP-nets and relation tables in database, there are some correspondences between them, shown in Table 1.

Besides that, the user's preference ordering on $Dom(X_i)$ (i.e. $CPT(X_i)$) can be encoded by the appearing orders of tuples with the same assignment of $pa(X_i)$ in table. Based on these relations presented, we can induce a CP-net into multiple relation tables according to Rule 2.

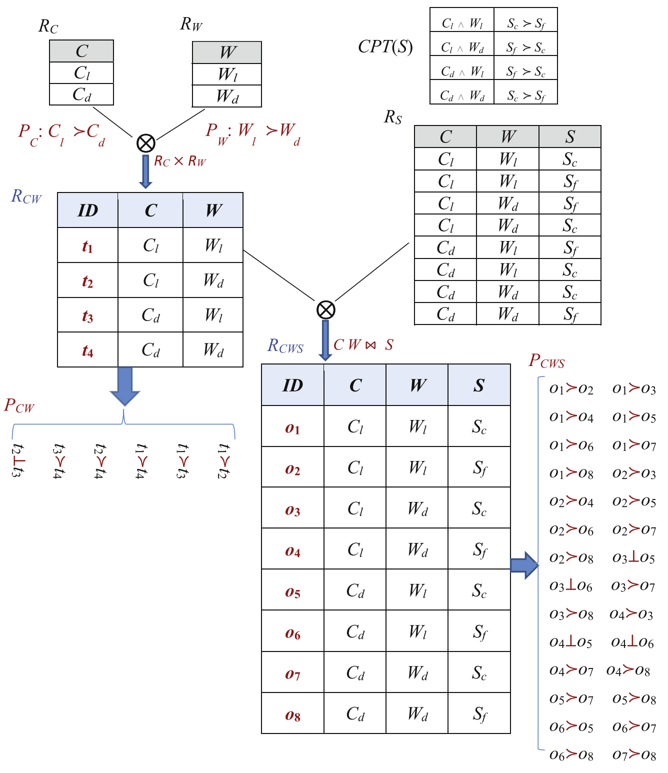
Rule 2. Every attribute X_i in a CP-net is induced into a table R_{X_i}, where the field set is $Pa(X_i) \cup X_i$, (that is, X_i and all its parent nodes) and the tuple set is the Cartesian product of its all field domains. Each row is attached a line number, for the same assignment of $Pa(X_i)$, the row containing the preferred value of $Dom(X_i)$ always appears ahead of the row containing the other one.

Examples of induced tables are depicted in Fig. 3. As it can be seen, the CP-net (in Example 1) is induced into three tables: R_C, R_W and R_S, where $V(R_C) = \{C\}$, $V(R_W) = \{W\}$, $V(R_S) = \{C, W, S\}$. For simplicity, line numbers in all induced tables are overlooked because we just want to explicate the process of preference composition.

5.3. Generation of satisfiability sequences

Next, we describe process of preference composition expressed by CP-nets in Fig. 4. For $P_C \otimes P_W$, let R_C and R_W be merged into table R_{CW} by Cartesian product operation, to better identify every tuple, we add a field ID to R_{CW}. Over R_{CW}, P_{CW} is induced using extended composition (in Definition 5), which is: $t_1 \succ t_2$, $t_1 \succ t_3$, $t_1 \succ t_4$, $t_2 \succ t_4$, $t_3 \succ t_4$ and $t_2 \perp t_3$. For instance, for tuples t_1 and t_2, $t_1(C) \succ_C t_2(C) \wedge t_1(W) \succ_W t_2(W)$, so $t_1 \succ_{CW} t_2$; for tuples t_2 and t_3, $t_2(C) \succ_C t_3(C) \wedge t_3(W) \succ_W t_2(W)$, so $t_2 \perp_{CW} t_3$.

Then, we merge R_{CW} and R_S into R_{CWS}, that is the nature of natural join due to the existence of common columns (C, W), and all



R_C

C
C_l
C_d

R_W

W
W_l
W_d

$CPT(S)$

$C_l \wedge W_l$	$S_c \succ S_f$
$C_l \wedge W_d$	$S_f \succ S_c$
$C_d \wedge W_l$	$S_f \succ S_c$
$C_d \wedge W_d$	$S_c \succ S_f$

R_S

C	W	S
C_l	W_l	S_c
C_l	W_l	S_f
C_l	W_d	S_f
C_l	W_d	S_c
C_d	W_l	S_f
C_d	W_l	S_c
C_d	W_d	S_c
C_d	W_d	S_f

R_{CW}

ID	C	W
t_1	C_l	W_l
t_2	C_l	W_d
t_3	C_d	W_l
t_4	C_d	W_d

R_{CWS}

ID	C	W	S
o_1	C_l	W_l	S_c
o_2	C_l	W_l	S_f
o_3	C_l	W_d	S_c
o_4	C_l	W_d	S_f
o_5	C_d	W_l	S_c
o_6	C_d	W_l	S_f
o_7	C_d	W_d	S_c
o_8	C_d	W_d	S_f

P_{CWS}

$o_1 \succ o_2$	$o_1 \succ o_3$
$o_1 \succ o_4$	$o_1 \succ o_5$
$o_1 \succ o_6$	$o_1 \succ o_7$
$o_1 \succ o_8$	$o_2 \succ o_3$
$o_2 \succ o_4$	$o_2 \succ o_5$
$o_2 \succ o_6$	$o_2 \succ o_7$
$o_2 \succ o_8$	$o_3 \perp o_5$
$o_3 \perp o_6$	$o_3 \succ o_7$
$o_3 \succ o_8$	$o_4 \succ o_3$
$o_4 \perp o_5$	$o_4 \perp o_6$
$o_4 \succ o_7$	$o_4 \succ o_8$
$o_5 \succ o_7$	$o_5 \succ o_8$
$o_6 \succ o_5$	$o_6 \succ o_7$
$o_6 \succ o_8$	$o_7 \succ o_8$

Fig. 4. Process of preference composition.

tuples from o_1 to o_8 constitute the decision space Ω for CP-nets. Over R_{CWS}, P_{CWS} also can be induced by the same way, which is:

$$\begin{array}{lllllll}
o_1 \succ o_2 & o_2 \parallel o_3 & o_3 \perp o_5 & o_4 \succ o_3 & o_5 \succ o_7 & o_6 \succ o_5 & o_7 \succ o_8 \\
o_1 \succ o_3 & o_2 \succ o_4 & o_3 \perp o_6 & o_4 \perp o_5 & o_5 \succ o_8 & o_6 \parallel o_7 & \\
o_1 \parallel o_4 & o_2 \parallel o_5 & o_3 \succ o_7 & o_4 \perp o_6 & & o_6 \succ o_8 & \\
o_1 \succ o_5 & o_2 \succ o_6 & o_3 \parallel o_8 & o_4 \parallel o_7 & & & \\
o_1 \parallel o_6 & o_2 \parallel o_7 & & o_4 \succ o_8 & & & \\
o_1 \succ o_7 & o_2 \succ o_8 & & & & & \\
o_1 \parallel o_8 & & & & & &
\end{array}$$

Now, we adjust all the $\parallel$ relations by taking the transitive closure of the direct relation $\succ$, and then obtain the final strong dominance tests (DT) as follows:

$$\begin{array}{lllllll}
o_1 \succ o_2 & o_2 \succ o_3 & o_3 \perp o_5 & o_4 \succ o_3 & o_5 \succ o_7 & o_6 \succ o_5 & o_7 \succ o_8 \\
o_1 \succ o_3 & o_2 \succ o_4 & o_3 \perp o_6 & o_4 \perp o_5 & o_5 \succ o_8 & o_6 \succ o_7 & \\
o_1 \succ o_4 & o_2 \succ o_5 & o_3 \succ o_7 & o_4 \perp o_6 & & o_6 \succ o_8 & \\
o_1 \succ o_5 & o_2 \succ o_6 & o_3 \succ o_8 & o_4 \succ o_7 & & & \\
o_1 \succ o_6 & o_2 \succ o_7 & & o_4 \succ o_8 & & & \\
o_1 \succ o_7 & o_2 \succ o_8 & & & & & \\
o_1 \succ o_8 & & & & & &
\end{array}$$

In addition to the explicit listing of dominance tests between tuples which could be obtained from P_{CWS}, a more meaningful work is by using P_{CWS} to construct the simplified induced graph of CP-nets (T) shown in Fig. 5.

Notice that this simplified induced graph is a Hasse diagram, it gets rid of some direct-connected edges from a vertex to another vertex because there are transitive paths between them. Hence, from the Hasse diagram, we can get the block sequences of satisfiability, that is,

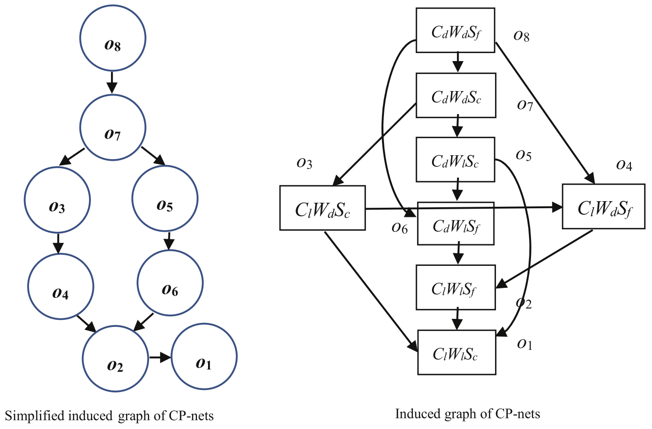


Fig. 5. Comparison of simplified graph and induced graph of CP-nets.

$$o_1 \succ o_2 \succ \left\{ \begin{array}{l} o_4 \succ o_3 \\ o_6 \succ o_5 \end{array} \right\} \succ o_7 \succ o_8$$

Even more specifically, according to the definition of CP-nets satisfiability, we have the following satisfiability sequences:

l_1: $o_1 \succ_N o_2 \succ_N o_4 \succ_N o_3 \succ_N o_6 \succ_N o_5 \succ_N o_7 \succ_N o_8$
l_2: $o_1 \succ_N o_2 \succ_N o_4 \succ_N o_6 \succ_N o_5 \succ_N o_3 \succ_N o_7 \succ_N o_8$
l_3: $o_1 \succ_N o_2 \succ_N o_4 \succ_N o_6 \succ_N o_3 \succ_N o_5 \succ_N o_7 \succ_N o_8$
l_4: $o_1 \succ_N o_2 \succ_N o_6 \succ_N o_4 \succ_N o_3 \succ_N o_5 \succ_N o_7 \succ_N o_8$
l_5: $o_1 \succ_N o_2 \succ_N o_6 \succ_N o_4 \succ_N o_5 \succ_N o_3 \succ_N o_7 \succ_N o_8$
l_6: $o_1 \succ_N o_2 \succ_N o_6 \succ_N o_5 \succ_N o_4 \succ_N o_3 \succ_N o_7 \succ_N o_8$

As is shown above, the algorithm of satisfiability sequences generation (called *SSG*) (Algorithm 1) is presented as follows.

Algorithm 1 Algorithm of satisfiability sequences generation (*SSG*).

```
Require: a CP-net $C$
Ensure: all satisfiability sequences $L$
1: $C$ is induced into multiple relation tables according to Rule 2;
2: Carry out the preference compositions to solve the dominance tests ($DT$) according to Rule 1;
3: Construct the simplified induced graph of CP-nets ($T$) by using $DT$;
4: Obtain all the satisfiability sequences $L$ from $T$;
5: Return $L$.
```

6. Top-*k* queries of database table

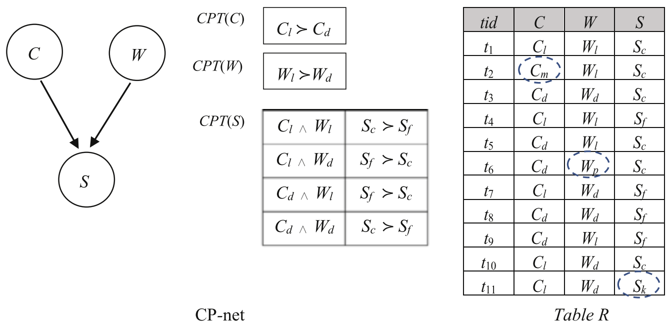
In the following section, we shall turn to the study of top-*k* queries of database table with CP-nets preference.

Definition 7. Given a table R with preference relation P (P is described by CP-nets), k is a positive integer, then the top-k tuples of R is defined as:

$$\text{top-k}(R) = \{t_i \mid t_i \in R \wedge (1 \le i \le k) \wedge (\forall t_i \in \text{top-k}) \\ (\forall t' \in R - \text{top-k}(R))\,(t_i \succ t')\}.$$

6.1. Motivating example

Example 2. We introduce here a relation table R with CP-nets preference shown in Fig. 6. Observing that tuples t_2, t_6 and t_{11} are



$C_l \wedge W_l$	$S_c \succ S_f$
$C_l \wedge W_d$	$S_f \succ S_c$
$C_d \wedge W_l$	$S_f \succ S_c$
$C_d \wedge W_d$	$S_c \succ S_f$

tid	C	W	S
t_1	C_l	W_l	S_c
t_2	C_m	W_l	S_c
t_3	C_d	W_d	S_c
t_4	C_l	W_l	S_f
t_5	C_d	W_l	S_c
t_6	C_d	W_p	S_c
t_7	C_l	W_d	S_f
t_8	C_d	W_d	S_f
t_9	C_d	W_l	S_f
t_{10}	C_l	W_d	S_c
t_{11}	C_l	W_d	S_k

Fig. 6. A relation table with CP-nets preference.

out of Ω which includes eight outcomes decided by its CP-net, because $C_m \notin dom(C)$, $W_p \notin dom(W)$, $S_k \notin dom(S)$. Thus we define $V(P, A_i)$ the set of active terms for preference P_{A_i}, over attribute A_i, i.e.$V(P, A_i) \subseteq dom(A_i)$, and $T(P, A)$ the set of active tuples of R featuring active terms for every attribute of P_A(all other tuples are called inactive), it holds that $\pi_A(T(P, A)) \subseteq V(P, A)$. For example, for table $R, V(P,C) = \{C_l, C_d\}, V(P,W) = \{W_l, W_d\}, V\{P,S) = \{S_c, S_f\}$;$T(P_{CWS}$, $\{C, W, S\}) = \{t_1, t_3, t_4, t_5, t_7, t_8, t_9, t_{10}\}$.

Let Q_A be the query condition, that is the union of $dom(A)$, and $Ans(Q_A)$ be the set of tuples matching the query Q_A. PQ_A denotes the preference P_A together with the query Q_A, the answer to PQ_A is the sequence $Ans(PQ_A)$. Now we sort all the tuples as the following processing steps:

- P_C
  $Q_C : (C = C_l) \vee (C = C_d)$,
  $Ans(Q_C) = \{t_1, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}\}$,
  $Ans(PQ_C) = \{t_1, t_4, t_7, t_{10}, t_{11}\} \succ \{t_3, t_5, t_6, t_8, t_9\}$.
- P_{CW}
  $Q_{CW} : (C = C_l \wedge W = W_l) \vee (C = C_l \wedge W = W_d) \vee (C = C_d \wedge W = W_l) \vee (C = C_d \wedge W = W_d)$,
  $Ans(Q_{CW}) = \{t_1, t_3, t_4, t_5, t_7, t_8, t_9, t_{10}, t_{11}\}$,
  $Ans(PQ_{CW}) = \{t_1, t_4\} \succ \{t_7, t_{10}, t_{11}\} \cup \{t_5, t_9\} \succ \{t_3, t_8\}$.
- P_{CWS}
  $Ans(Q_{CWS}) = \{t_1, t_3, t_4, t_5, t_7, t_8, t_9, t_{10}\}$,
  $Ans(PQ_{CWS}) = \{t_1\} \succ \{t_4\} \succ \{\{t_7\} \succ \{t_{10}\}\} \cup \{\{t_9\} \succ \{t_5\}\} \succ \{t_3\} \succ \{t_8\}$.

Note that the answer to a preference query is a sequence of data blocks, where each block contains data that are more preferred than those in the blocks after. Just like the resulting block sequence essentially linearizes the order of tuples induced by the preference P_{CWS} as depicted in $Ans(PQ_{CWS})$. In this way, the user can inspect the blocks by turn and stop at any point at which he feels satisfied by the data. In other words, we don't even need to construct and linearize this Cartesian product, instead, we can simply generate its block sequences, which rely solely on the number of necessary queries and avoid database rescans.

6.2. Query-ordering algorithms with CP-nets preference

As for block sequence, it is proposed by *Georgiadis* in *Efficient Rewriting Algorithms for Preference Queries* (Georgiadis, Kapantaidakis, Christophides, Nguer, & Spyratos, 2008), including *LBA* (Lattice Based Algorithm), *TBA* (Threshold Based Algorithm) and corresponding theorems. Algorithm *LBA* takes as input a relation *R* and a preference expression P_A involving a subset *A* of *R*'s attributes. Then, it outputs progressively successive blocks of *T*(*P*, *A*). Each time a block is computed, the user may signal to continue with the next one; alternatively, he may request to obtain the top-*k* tuples of *T*(*P*, *A*). However, unlike existing qualitative preference frameworks, our preference query is more sophisticated in consideration of dependencies among *A*, thus we expand query-ordering algorithm by introducing two operators (⊙, ⊕):

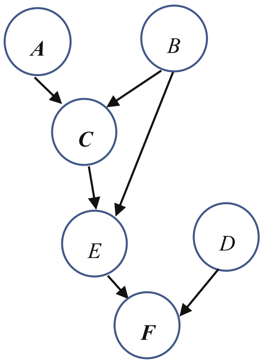


Fig. 7. A vertex relation diagram.

⊙—*—the composition operator for vertices which depend on no other vertex*,

⊕—*—the composition operator for vertices which depend on other vertex*, and two vertex sets (S_i, S_d):

S_i —*—the set of vertices which depend on no other vertex*,

S_d —*—the set of vertices which depend on other vertex.*

For example, observing the following vertex relation diagram in Fig. 7, we have:

$S_i = \{A, B, D\}$,

$S_d = \{C, E, F\}$, and the relations between S_i and S_d are:

$C \leftarrow \{A, B\}, E \leftarrow \{C, B\}, F \leftarrow \{E, D\}$.

Rule 3. The preference composition for attributes of relation *R* is a composition sequence, which should be carried out in this order: firstly, compose these attributes in S_i using operator ⊙ progressively, and then compose attributes in S_d using operator ⊕ in turn, each time, for the selected attribute to be composed, it must be sure that all its parents have been included in the preceding sequence.

Just like the attribute set in Fig. 7, the composition sequence of all attributes is:

$$\underbrace{P_A \odot P_B \odot P_D}_{S_i} \oplus \underbrace{P_C \oplus P_E \oplus P_F}_{S_d}$$

As for ⊙, it is the composition of two non-dependent attributes, we can obtain the sequence of blocks as the answer to the preference query P_Q by *LBA*. It is worth noticing that the resulting block sequence essentially linearizes the order of tuples after finishing all the ⊙ operations in S_i, just like PQ_C and PQ_{CW} shown in example 2. As for ⊕, it further subdivides these blocks derived from previous process, by using *CPT* of attributes in S_d. Hence, the Query-Ordering Algorithm with CP-nets Preference (*QOCP*) (Algorithm 2) is presented as follows.

ConstructQuerySets divides *A* into sets S_i and S_d according to whether the attribute has parents. *LBA* (lines 3–9) is realized by two steps: generating a query block sequence and outputting the computed block. In which, *ConstructQueryBlocks* (line 3) traverses recursively a preference expression tree P_A (from *P.root*) and computes bottom-up the number of blocks and their origin in *QB*. In our model, $\forall A, B \in S_i$, the equal preference relation $P_A \approx P_B$ holds, because there are no dependencies among attributes in S_i. Therefore, for each *QB* entry it generates the structure of the respective block sequence only when ≈ appears as a preference relation between expressions *P.left* and *P.right* (shown in Algorithm 3); *Evaluate* (line 7) executes each query q_i of its input set $\cup_{q_i}$, finally, it outputs the computed block and returns its size. By iteratively

Algorithm 2 Query-ordering algorithm with CP-nets preference (*QOCP*).

```
Require: a relation $R$, a CP-net with attribute set $A$ and a $k > 0$
Ensure: the top-$k$ tuples of $R$
 1: $ConstructQuerySets(S_i, S_d)$
 2: $totalsize = 0, j = 1$
 3: $QB = ConstructQueryBlocks(P_{S_i})$
 4: $i = 0$
 5: repeat
 6:     $\cup_{q_i} = GetBlockQueries(QB[i])$
 7:     $BS_s = Evaluate(\cup_{q_i})$
 8:     $i+ = 1$
 9: until ExitReq or $i = |QB|$
10: $P = BS_s$
11: $Q = S_d$
12: $N = num(P)$    // the number of blocks in block sequence
13: repeat
14:     $M = GetAttribute(Q)$
15:     repeat
16:         $P(j) = SubDividedSequ(P(j), CPT(M))$
17:         $j = j + 1$
18:     until $j > N$
19:     $Q = Q - M$
20: until $Q = \emptyset$
21: $Output(P, k)$    // output the top-$k$ tuples in $P$
```

Algorithm 3 *ConstructQueryBlocks*.

```
Require: a preference expression $P_A$
Ensure: a query block sequence $QB$
 1: if $P$ is a leaf then
 2:     $QB = PrefBlocks(V(P, A_i))$
 3: else
 4:     $QB\_left = ConstructQueryBlocks(P.left)$
 5:     $QB\_right = ConstructQueryBlocks(P.right)$
 6: end if
 7: for each $w \in [0, |QB\_left| + |QB\_right| - 1]$ do
 8:     $QB[w] = \bigcup\{QB\_left[i] * QB\_right[j] | i + j = w\}$
 9: end for
10: return $QB$
```

calling *GetBlockQueries* (line 6) to create the associated list of conjunctive queries and *Evaluate* (line 7) to output successive *T*(*P*, *A*) blocks, we can get the successive blocks of $T(P_{S_i}, S_i)$, that is the block sequence for S_i, namely BS_s.

GetAttribute (line 14) derives the attribute which has not been composed but whose all parent attributes have been composed from S_d. *SubDividedSequ* (line 16) further partitions these blocks into several subsets one by one by using the attribute's *CPT*. *GetAttribute* and *SubDividedSequ* are repeated until the compositions of all attributes in S_d are finished (lines 13 to 20). Thus, we can linearize all tuples of *T*(P_A, *A*) and compute the top-*k* tuples from this sequence.

However, observing *Ans*(PQ_{CW}) and *Ans*(PQ_{CWS}) in Example 2, we can find that if the total number of tuples in the top *i* BS_s blocks > *k* and the total number of tuples in the top($i - 1$) BS_s blocks < *k* (Algorithm 4 lines 13 to 16), we only need to call *SubDividedSequ* for the *i*th block (Algorithm 4 line 19). Therefore, the improved Query-Ordering Algorithm with CP-nets Preference (*IQOCP*) (Algorithm 4) is presented as follows.

Algorithm 4 Improved query-ordering algorithm with CP-nets preference (*IQOCP*).

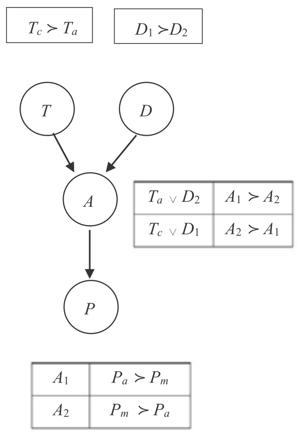
```
Require: a relation $R$, a CP-net with attribute set $A$ and a $k > 0$
Ensure: the top-$k$ tuples of $R$
 1: $ConstructQuerySets(S_i, S_d)$
 2: $totalsize = 0, i = 0$
 3: $QB = ConstructQueryBlocks(P_{S_i})$
 4: $i = 0$
 5: repeat
 6:     $\cup_{q_i} = GetBlockQueries(QB[i])$
 7:     $BS_s = Evaluate(\cup_{q_i})$
 8:     $i+ = 1$
 9: until ExitReq or $i = |QB|$
10: $P = BS_s$
11: $Q = S_d$
12: $N = num(P)$    // the number of blocks in block sequence
13: repeat
14:     $totalsize = totalsize + |P(i)|$    // the total number of tuples in the top i blocks
15:     $i = i + 1$
16: until $totalsize > k$ or $i > N$
17: repeat
18:     $M = GetAttribute(Q)$
19:     $P(i) = SubDividedSequ(P(i), CPT(M))$
20:     $Q = Q - M$
21: until $Q = \emptyset$
22: $Output(P, k)$    // output the top-$k$ tuples in $P$
```

The cost of *IQOCP* is mainly due to the number of conjunctive queries it has to execute in order to construct a block of the answer. A conjunctive query is usually evaluated by traversing the available indices on the involved attributes, intersecting the tids (tuple identifiers) and then fetching the matching tuples from the disk. Assuming that *n* queries (each query is *q*) are executed in total to construct the resulting block sequence, the algorithm cost will be $O(n * (log|R| + |ans(q)|))$(Georgiadis, Kapantaidakis, Christophides, Nguer, & Spyratos, 2008). In the best case, only one query (i.e., $n = 1$) is required and the number of returned tuples is very small, in particular, when |*V*(*P*, *A*)| < <|*T*(*P*, *A*)|, the practical cost drops to *O*(*log*|*R*|). In the worst case, all the lattice queries need to be executed to construct the entire block sequence as just a few of the leaf queries actually return almost all of the active tuples. Thus, the total cost of the index traversals will rise to *O*(|*V*(*P*, *A*)|**log*|*R*|) where $|V(P, A)| = \times_i |V(P, A_i)|$, In particular, when |*V*(*P*, *A*)| > >|*T*(*P*, *A*)|, the practical complexity of *IQOCP* in the worst case becomes *O*(|*V*(*P*, *A*)|).

6.3. An example

Here we give an example to demonstrate the solution process for Algorithm 4. Consider table *R*(*Numid*, *T*, *B*, *D*, *A*, *C*, *P*) in Fig. 8, describing part of the contents of an Insurance Model Selection Library (*IMSL*), where for simplicity each tuple is identified by a tuple identifier (*Numid*). A customer wishing to buy an insurance might state the following preferences over *IMSL* resources:

(1) critical illness cover (T_c) is preferred to individual personal accident insurance (T_a); (preference P_T)
(2) 10-year period (D_1) is preferred to 20-year period (D_2); (preference P_D)
(3) $200,000 ($A_1$) is preferred to $100,000 ($A_2$) insured amount if individual personal accident insurance (T_a) or 20-year insurance duration (D_2) selected, whereas $100,000 insured amount is preferred if critical illness cover (T_c) or 10-year insurance duration (D_1) selected; (preference P_A)



Numid	*T*	*B*	*D*	*A*	*C*	*P*
t_1	T_c	B_1	D_1	A_2	C_1	P_a
t_2	T_a	B_2	D_2	A_1	C_2	P_m
t_3	T_t	B_1	D_2	A_1	C_3	P_a
t_4	T_c	B_2	D_1	A_1	C_1	P_a
t_5	T_a	B_2	D_1	A_1	C_1	P_a
t_6	T_c	B_1	D_2	A_2	C_2	P_t
t_7	T_c	B_3	D_2	A_1	C_2	P_a
t_8	T_a	B_2	D_1	A_2	C_3	P_a
t_9	T_a	B_2	D_1	A_2	C_2	P_m
t_{10}	T_c	B_3	D_2	A_1	C_1	P_m
t_{11}	T_c	B_1	D_3	A_2	C_1	P_m
t_{12}	T_a	B_2	D_2	A_3	C_2	P_a
t_{13}	T_c	B_4	D_1	A_2	C_5	P_m
t_{14}	T_a	B_4	D_1	A_1	C_4	P_a
t_{15}	T_c	B_3	D_1	A_2	C_5	P_m
t_{16}	T_k	B_3	D_2	A_1	C_5	P_t
t_{17}	T_c	B_5	D_1	A_2	C_6	P_m
t_{18}	T_a	B_3	D_2	A_1	C_1	P_a
t_{19}	T_a	B_4	D_2	A_2	C_2	P_a
t_{20}	T_c	B_5	D_2	A_2	C_5	P_m

CP-net *Table R*

Fig. 8. Insurance recommendation with CP-nets preference.

(4) monthly payment (P_m) is preferred to yearly payment (P_a) if the insured amount is $100,000 ($A_2$), whereas yearly payment is a better selection if the insured amount is $200,000 ($A_1$). (preference P_P)

Such statements define actually binary relations: (1), (2), (3) and (4) defined over attribute domains, in which, relations (1) and (2) are independent, (3) depends on (1) and (2), whereas (4) depends on (3), therefore, the customer's preference is a CP-nets preference just like shown in Fig. 8.

Now we need to recommend 3 insurances with different attribute settings for this customer, that is, compute the top-3 tuples of *R*.

Observing table *R* and CP-net in Fig. 8, we can find that there are altogether six attributes (i.e., *T, B, D, A, C* and *P*) taken into account for an insurance configuration. However, the customer only focuses on four attributes (i.e., *T, D, A* and *P*) while deciding to select an insurance, which is the attribute set *V* and whose detailed preferences are depicted in CP-net. Therefore, we recommend the top-*k* outcomes to the customer from table *R* only relying on his CP-net.

Let us consider first the dependencies among attributes in CP-net, which partition the attribute set *V* into two subsets, such that:
$S_i = \{T, D\}$,
$S_d = \{A, P\}$,and the relations between S_i and S_d are:
$A \leftarrow \{T, D\}, P \leftarrow \{A\}$.

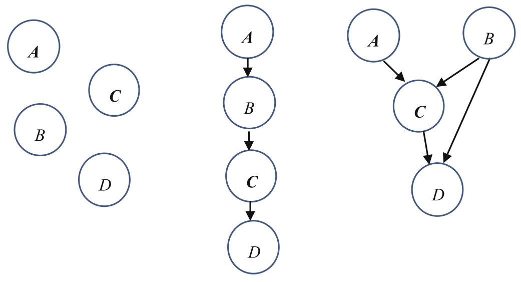
According to rule 3, the composition sequence of all attributes in *V* is:

$$\underbrace{P_T \odot P_D}_{S_i} \oplus \underbrace{P_A \oplus P_P}_{S_d}$$

Step 1, we will compute the operator $\odot$ in S_i:

(1) The preference P_T which relates two values of the attribute *T*, namely T_c and T_a. The underlying assumption here is that the only tuples that are of interest to the user are those containing one of these values. Therefore, the set of tuples matching P_T is the answer to the query $Q_T : (T = T_c) \vee (T = T_a)$. Referring to table *R*, the answer to Q_T is the following: $Ans(Q_T) = \{t_1, t_2, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{17}, t_{18}, t_{19}, t_{20}\}$, in the same way, the answer to Q_D is the following: $Ans(Q_D) = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}\}$. Consequently, the answer to $Q_{T \odot D}$ is the following: $Ans(Q_{T \odot D}) = Ans(Q_T) \bigcap Ans(Q_D) = \{t_1, t_2, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{12}, t_{13}, t_{14}, t_{15}, t_{17}, t_{18}, t_{19}, t_{20}\}$.

(2) Then, we compute the block sequence answering a preference query. The block sequences of P_T and P_D are $\{T_c\} \succ \{T_a\}$ and $\{D_1\} \succ \{D_2\}$ respectively. Thus, according to the preference composition theorems in (Georgiadis, Kapantaidakis, Christophides, Nguer, & Spyratos, 2008):
"Given the block sequences $X_0 \succ X_1 \succ \cdots \succ X_{n-2} \succ X_{n-1}$, and $Y_0 \succ Y_1 \succ \cdots \succ Y_{m-2} \succ Y_{m-1}$ of two preferences P_X and P_Y, the block sequence $Z_0 \succ Z_1 \succ \cdots \succ Z_{n+m-3} \succ Z_{n+m-2}$ of the preference induced by $P_X \odot P_Y$ over $X \times Y$, will consist of $n + m - 1$ blocks; each block Z_p will comprise elements only from blocks X_q and Y_r, such that $q + r = p$."
Thus, for $P_T \odot P_D$, the block sequence for preference query is:
$QB = T_cD_1 \succ T_cD_2 \bigcup T_aD_1 \succ T_aD_2$,
and the answer to this query block sequence is the following:
$BS_s = \{t_1, t_4, t_{13}, t_{15}, t_{17}\} \succ \{t_6, t_7, t_{10}, t_{20}\} \bigcup \{t_5, t_8, t_9, t_{14}\} \succ \{t_2, t_{12}, t_{18}, t_{19}\}$.



Separable-structured Chain-structured DAG

Fig. 9. Three kinds of CP-nets.

Note that there is only one block $BS_s(1)$ needing to be further subdivided owing to $|BS_s(1)| > 3$.

Step 2, we will compute the operator $\oplus$ in S_d:

(1) For P_A, by using *CPT*(*A*), $BS_s(1)$ is subdivided to the following sequence:
$P(1) = \{t_1, t_{13}, t_{15}, t_{17}\} \succ \{t_4\}$.

(2) For P_P, by using *CPT*(*P*), *P*(1) is further subdivided to the following sequence:
$P(1) = \{t_{13}, t_{15}, t_{17}\} \succ \{t_1\} \succ \{t_4\}$.

Now, we can recommend the following insurance outcomes to the customer:
top-3(R) = $\{t_{13}, t_{15}, t_{17}\}$.

7. Experiment

As a common ground for performance comparison of all algorithms, we conducted experiments on an 1.60GHz Intel(R) Core(TM) CPU with 4GB RAM running Windows 8.1 system, all implemented in Java on top of PostgreSQL 9.2. We used a simulation dataset just like shown in Fig. 8.

The effect of different structures of CP-nets: In Fig. 9 we identified three kinds of binary CP-nets: separable-structured, chain-structured and DAG CP-nets.

To carry out the strong dominance tests of different structured CP-nets, their database handling cost and preference composition cost were compared respectively in this three kinds of CP-nets

Table 2
Space performance for CP-nets with different structures .

Structures	Numbers of $\|V\|$					
	5	6	7	8	9	10
tree-structured	509	730	841	956	1075	1179
polytree	711	1056	1578	2351	3647	5980
directed-path singly connected	975	1572	2605	5774	9411	12325

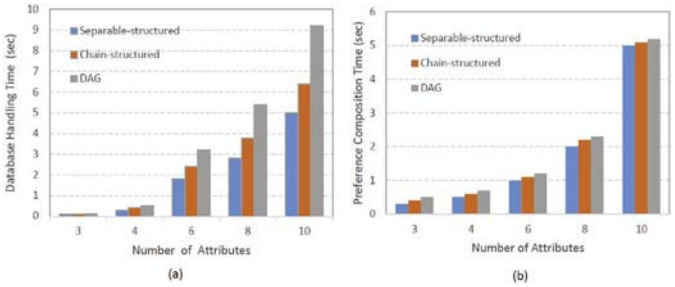


Fig. 10. Processing time vs number of attributes(1).

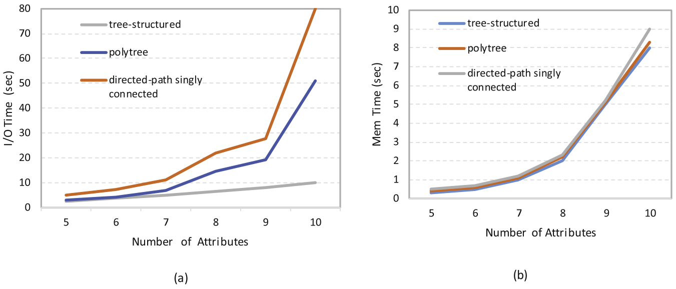


Fig. 11. Processing time vs number of attributes(2).

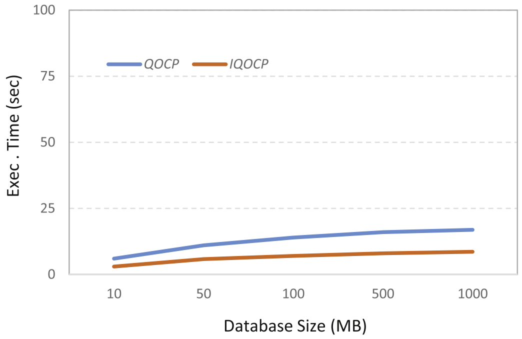


Fig. 12. Total processing time vs database size.

with 3, 4, 6, 8 and 10 attributes. As shown in Fig. 10(a), the database handling mainly includes inducing the CP-nets into tables and table joins operations. Despite there will be n induced tables for any CP-nets with n attributes, the amounts of data in these tables are significantly different, owning to the fact that the more dependencies among attributes, the more tuples there will be in induced tables. But for preference composition cost, we mainly take into account the comparisons among tuples according to extending Pareto composition. Note that the number of tuples in a composed table is fixed, i.e., for n attributes compositions, there will obtain 2^n tuples. Hence, there is approximately the same cost for three kinds of CP-nets in Fig. 10(b).

In particular, we randomly generated three kinds of DAG CP-nets: tree-structured CP-nets, polytree CP-nets, directed-path singly connected CP-nets, by varying the numbers of variable $|V|$, the structures of the network and the conditional preference tables, and see how they affect the performance of the proposed algorithm, here $|V|$ was set to 5, 6,7, 8, 9 and 10 respectively, $|Dom(X_i)|$ was set to 2. The response time for solving dominance querying was illustrated by Fig. 11. The experimental results are consistent with Fig. 10, that is, the complexity of dominance querying with respect to binary-valued CP-nets shows a connection between the structure of the CP-net graph and database handling time. This conclusion is justified as the *CPTs* are part of the problem description, and the size of a $CPT(X_i)$ is exponential in $|Pa(X_i)|$), hence, there are significant increases in the I/O cost of inducing CP-nets into Database tables (shown in Fig. 11(a)) and the size of the database storage space (shown in Table 2, the numbers of bytes vary with the changes of numbers of $|V|$ and CP-net structures), but there is approximately the same memory cost of preference composition (shown in Fig. 11(b)) with $|O|$ fixed.

The effect of database size: In this experiment, we scaled up the sizes of the database from 10 to 1,000 MB (or from 100K to 10,000K tuples) to perform *QOCP* and *IQOCP*, $V(P, A)$ is fixed. Fig. 12 depicts the total execution time of the 2 algorithms. As expected, *IQOCP* is significantly more efficient than *QOCP* owing to reducing the number of comparisons. But their performances are not rapidly increasing as the database grows. This is due to the fact that, as $|V(P, A)| < <|T(P, A)|$, queries of the first Query Lattice block (for *LBA*) most probably suffice for computing the answer, regardless the fact that their answer size has grown. Their performances fall gradually with database growth only when tuples need to be retrieved from the database.

8. Conclusion

In this article, we have mainly designed the operators of preference composition. The traditional Pareto composition is extended to preference modeled by CP-nets, preserving a strict partial order and non-associativity, on this basis, two questions have been solved: how to generate the satisfiability sequences of CP-nets and how to get the top-k tuples of relational table with CP-nets preference. The first question is solved by inducing a CP-net into multiple tables—realizes the storage of CP-nets, and natural join operations—gets the dominance tests of all outcomes. For the second question, to realize the top-k queries, existing algorithms like *Block Nested Loop* (BNL)(Borzsony, Kossmann, & Stocker, 2001) and Best (Torlone & Ciaccia, 2002b) are agnostic to preference expressions, whose semantics is captured only externally by the employed dominance testing functions. For this reason, they need to access all tuples of a relation R at least once and perform for every R tuple at least one dominance test. Hence, they are inadequate for large databases. Moreover, as both have to read the entire relation before returning the top block, they are not suitable for a progressive result computation, as *QOCP* and *IQOCP* are. *QOCP* and *IQOCP* both scale linearly based on Query Lattice, the sequence is divided into multiple blocks firstly, and then every block is subdivided by using *CPT*. *IQOCP* improves *QOCP* by avoiding to subdivide blocks which have already been contained in the top-k tuples, that is, only one block needs to be subdivided.

We also know the conclusion that for voluminous databases, *LBA* is best for queries with short standing preferences (typically resulting to small query lattices), while *TBA* wins when long standing preferences (typically resulting to larger query lattices) are used instead. So we are interested in extending corresponding queries by *TBA* for a long standing CP-nets preference over 5 attributes with 12 values each.

Acknowledgments

We gratefully acknowledge the detailed and helpful comments of the anonymous reviewers, who have enabled us to considerably improve this paper. This work is supported by Science and Technology Project of Higher Education of Shandong Province (No. J14LN23, No. J15LN09), Shandong Province Natural Science Foundation (No. ZR2014FL009, No. ZR2015PF010) and Natural Science Foundation of China (No. 61572419, No. 61572418).

References

Abbas, A. E., & Bell, D. E. (2012). One-switch conditions for multiattribute utility functions. *Operations Research, 60*(5), 1199–1212.

Agrawal, R., & Wimmers, E. L. (2000). A framework for expressing and combining preferences. In *Proceedings of the 2000 acm sigmod international conference on management of data, SIGMOD '00* (pp. 297–306). New York, NY, USA: ACM. doi:10.1145/342009.335423.

Arvanitis, A., & Koutrika, G. (2012). Towards preference-aware relational databases. In *Proceedings of the 2012 ieee 28th international conference on data engineering, ICDE '12* (pp. 426–437). Washington, DC, USA: IEEE Computer Society. doi:10.1109/ICDE.2012.31.

Borzsony, S., Kossmann, D., & Stocker, K. (2001). The skyline operator. In *Data engineering, 2001. proceedings. 17th international conference on* (pp. 421–430). doi:10.1109/ICDE.2001.914855.

Bosc, P., Hadjali, A., & Pivert, O. (2011). On database queries involving competitive conditional preferences. *International Journal of Intelligent Systems, 26*(3), 206–227. doi:10.1002/int.20463.

Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., & Poole, D. (2004). CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research, 21*(1), 135–191.

Bouveret, S., Endriss, U., & Lang, J. (2009). Conditional importance networks: A graphical language for representing ordinal, monotonic preferences over sets of goods. In *Proceedings of the 21st international jont conference on artifical intelligence, IJCAI'09* (pp. 67–72). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Brafman, R. I., Domshlak, C., & Shimony, S. E. (2006). On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research, 25*(1), 389–424.

van Bunningen, A., Feng, L., & Apers, P. (2006). A context-aware preference model for database querying in an ambient intelligent environment. In S. Bressan, J. Kng, & R. Wagner (Eds.), *Database and expert systems applications*. In *Lecture Notes in Computer Science: 4080* (pp. 33–43). Springer Berlin Heidelberg. doi:10.1007/11827405_4. http://dx.doi.org/10.1007/11827405_4.

Chomicki, J. (2003). Preference formulas in relational queries. *ACM Transactions on Database Systems, 28*(4), 427–466. doi:10.1145/958942.958946.

Choo, E. U., Schoner, B., & Wedley, W. C. (1999). Interpretation of criteria weights in multicriteria decision making. *Computers & Industrial Engineering, 37*(3), 527–541.

Ciaccia, P. (2007). Querying databases with incomplete cp-nets. In *Multidisciplinary workshop on advances in preference handling (m-pref 2007)* (pp. 1–8).

Cornelio, C., Goldsmith, J., Mattei, N., Rossi, F., & Venable, K. (2013). Updates and uncertainty in cp-nets. In S. Cranefield, & A. Nayak (Eds.), *Ai 2013: Advances in artificial intelligence*. In *Lecture Notes in Computer Science: 8272* (pp. 301–312). Springer International Publishing. doi:10.1007/978-3-319-03680-9_32. http://dx.doi.org/10.1007/978-3-319-03680-9_32.

Endres, M., & Kießling (2006). Transformation of tcp-net queries into preference database queries. In *ECAI 2006 - advances in preference handling* (pp. 1–8).

Georgiadis, P., Kapantaidakis, I., Christophides, V., Nguer, E. M., & Spyratos, N. (2008). Efficient rewriting algorithms for preference queries. In *Proceedings of the 2008 IEEE 24th international conference on data engineering, ICDE '08* (pp. 1101–1110). Washington, DC, USA: IEEE Computer Society. doi:10.1109/ICDE.2008.4497519.

Goldsmith, J., Lang, J., Truszczynski, M., & Wilson, N. (2008). The computational complexity of dominance and consistency in cp-nets. *Journal of Artificial Intelligence Research, 33*(1), 403–432.

Kießling, W. (2002). Foundations of preferences in database systems. In *Proceedings of the 28th international conference on very large data bases, VLDB '02* (pp. 311–322). VLDB Endowment.

Kostas Stefanidis, E. P. P. V. (2007). Adding context to preferences. In *2007 IEEE 23rd international conference on data engineering, ICDE '07* (pp. 846–855).

Koutrika, G., & Ioannidis, Y. (2004). Personalization of queries in database systems. In *Proceedings of the 20th international conference on data engineering, ICDE '04* (pp. 597–608). Washington, DC, USA: IEEE Computer Society.

Liu, J. L. (2011). Research on cp-nets and its expressive power. *ACTA AUTOMATICA SINICA, 37*(3), 290–302.

Liu, J. L., & Liao, S. Z. (2015). Expressive efficiency of two kinds of specific cp-nets. *Information Sciences, 295*(2), 379–394. http://dx.doi.org/10.1016/j.ins.2014.10.038

Liu, J. L., Liao, S. Z., & Zhang, W. (2012). On the completeness and consistency for cp-nets. *Journal of Software, 23*(6), 1531–1541.

Liu, W. Y., Wu, C. H., Feng, B., & Liu, J. T. (2015). Conditional preference in recommender systems. *Expert Systems with Applications, 42*(2), 774–788.

Miele, A., Quintarelli, E., & Tanca, L. (2009). A methodology for preference-based personalization of contextual data. In *12th international conference on extending database technology* (pp. 287–298).

Mindolin, D., & Chomicki, J. (2007). Hierarchical cp-networks. In *In m-pref* (pp. 1–8).

Santhanam, G. R., Basu, S., & Honavar, V. (2010). Dominance testing via model checking. In *AAAI 2010* (pp. 357–362). Atlanta, Georgia, USA.

Stefanidis, K., Koutrika, G., & Pitoura, E. (2011). A survey on representation, composition and application of preferences in database systems. *ACM Transaction on Database Systems, 36*(3), 19:1–19:45. doi:10.1145/2000824.2000829.

Sun, X. J., & Liu, J. L. (2012). On the satisfiability and consistency for cp-nets. *Journal of Computer Research and Development, 49*(4), 754–762.

Sun, X. J., & Liu, J. L. (2015). Research on algorithm of satisfiability ranking generation for cp-nets. *Computer Science, 42*(5), 270–273.

Torlone, R., & Ciaccia, P. (2002a). Finding the best when it's a matter of preference. In *Sistemi evoluti per basi di dati, 2002* (pp. 347–360).

Torlone, R., & Ciaccia, P. (2002b). Which are my preferred items?. In W. on Recommendation, & P. in E-commerce (Eds.), *Sistemi evoluti per basi di dati, 2002* (pp. 1–9).