Database Preference Queries Revisited

Extended abstract

Ronen I. Brafman BRAFMAN@CS.BGU.AC.IL
Dept. of Computer Science, Ben-Gurion University, Israel

Carmel Domshlak DCARMEL@CS.CORNELL.EDU
Dept. of Computer Science, Cornell University, Ithaca, NY 14853
tel: 607-2578903, fax: 607-2554428

1. Introduction

The problem of preference elicitation and preference management has generated much interest in the database systems community in recent years. This interest stems from a rapidly growing class of untrained, lay users browsing extremely large databases accessible through the Internet. Typically these users do not have clear knowledge about the particular items that these databases contain, nor do they have a particular result in mind. Rather, they are attempting to identify items that are useful for them in some manner, or in other words, items that *suit their preferences best*. Examples include users looking for gifts in the databases of online merchants, users searching for attractive vacation packages, users looking for vendors/competitors in a particular area, etc. To support such users, database systems must be able to process preference queries, *i.e.*, queries that describe desirable properties of the end result of the search process. Such queries must be intuitive to formulate by the user, correctly interpreted by the system, computationally efficient to process, and enable the user to quickly home-in on desirable elements.

The need to support preference queries has not gone unnoticed by the database community, and a number of general frameworks emerged in the past decade (see [6] for a survey). Support for preference-queries raises two primary concerns: semantic clarity and adequacy, and computational efficiency. The semantic issue is particularly thorny in this context: When asked for their preferences, most users should be expected to supply only simple statements such as: "I prefer Continental to Delta" or "In a minivan I prefer automatic transmission to manual transmission", and we need to interpret these statements properly. Therefore, the first step to supporting preference queries is to clearly define the meaning of this, very circumscribed class of natural language statements. This semantics must ensure that the set of most-preferred items induced by it will reasonably match users' expectations, for otherwise, users will turn away from these systems. Some recent work on preference queries in database systems skirt the semantic issue, instead considering general frameworks that support multiple interpretations (*e.g.*, see [7, 16]). The scientific value of such work is clear, but any actual implementation must make a concrete commitment. Authors in the DB community who indirectly considered such concrete semantics seem to uniformly favor what we call the "totalitarian intersection" approach. The main conceptual contribution of this paper is: (1) To show that the "totalitarian intersection" approach is unsuitable as it fails to meet the basic demand of semantic adequacy and yields unintuitive or empty results even for very simple and natural queries; (2) To provide a different interpretation, the "*ceteris paribus* union" interpretation. This semantics is considered the most natural by philosophers (see, *e.g.*, [14]), but has not been considered in the DB community.

Semantically, the *ceteris paribus* interpretation is by far superior, but computationally, it is problematic. The operator BEST that extracts from the database the set of all most preferred

	category	ext-color	int-color
t_1	minivan	red	bright
t_2	minivan	red	dark
t_3	minivan	white	bright
t_4	minivan	white	dark
t_5	SUV	red	bright
t_6	SUV	red	dark
t_7	SUV	white	bright
t_8	SUV	white	dark

(a)

s_1	*I prefer red minivans to white minivans.*
s_2	*I prefer white SUVs to red SUVs.*
s_3	*In white cars I prefer a dark interior.*
s_4	*In red cars I prefer a bright interior.*
s_5	*I prefer minivans to SUVs.*

(b)

Figure 1: (a) Instance of the `Car` schema; (b) A schema-dependent preference query over `Car`.

items (independently proposed in [7, 16, 22] as the ultimate operator for evaluating preference queries) can be computationally intractable within this semantics: The worst-case time complexity of BEST is $O(\exp(n) \cdot D^2)$, where n is the arity and D is the size of the database relation. The main technical contribution of this paper is to introduce a relaxation of BEST, an operator that we call ORD, and to show a significant class of preference queries for which ORD is computable in time $O(nD \log D)$.

The technical results of this paper with respect to the *ceteris paribus* semantics use and extend some recent results presented in [3, 8]. The rest of the paper is structured as follows: In Section 2 we discuss the basic issues that arise in interpreting preference queries and some key technical assumptions. In Section 3 we discuss the totalitarian semantics and explain why it is inadequate. In Section 4 we discuss the *ceteris paribus* interpretation, showing that it works much better when we attempt to combine preference statements. In Section 5 we explain how we can efficiently answer preference queries in the context of the *ceteris paribus* semantics by using the ORD operator. We conclude the paper in Section 6.

2. Main Concepts and Issues

In this paper we focus on the *qualitative* approach to database preference queries (adopted, *e.g.*, in [7, 18, 12, 16]), in which user preferences are represented by a general binary preference relation over a relation schema[1]. Formally, given a relation schema $\mathbb{R}$, a *preference query* $\mathcal{Q}$ over $\mathbb{R}$ consists of a set $\mathcal{Q} = \{s_1, \cdots, s_m\}$ of *preference statements*. These preference statements define preference relations $\{\succ_1, \cdots, \succ_m\}$, respectively, from which one should derive the global preference relation $\succ_{\mathcal{Q}}$. Subsequently, given a relation instance R of $\mathbb{R}$, the database system should return a subset of R containing, *e.g.*, those tuples that are "optimal" according to $\succ_{\mathcal{Q}}$. To illustrate these concepts, consider a relation schema `Car` (`category`, `ext-color`, `int-color`), and an instance R of `Car` described in Figure 1(a). Suppose that the user expresses the preference statement s: "I prefer red minivans with bright interior to white minivans with dark interior". The preference relation induced by this statement over the tuples of R is $\succ_s = \{t_1 \succ_s t_4\}$ (while other pairs of tuples from R are incomparable in $\succ_s$).

2.1 Interpretation and Evaluation of Preference Queries

The interpretation of statement s above poses no serious difficulties because it explicitly compares tuples. However, this is the exception, rather than the rule. Preference statements typi-

1. More quantitative forms of specifications are possible, too (*e.g.*, see [1, 15]), though they require more user effort, and typically amount to the specification of an additive value function.

cally mention only a subset of attributes, as in s': "I prefer red minivans to white SUVs". This immediately leads to the first fundamental semantic question: *What preference order over the database tuples is induced by a preference statement over the values of a (strict) subset of the attribute set?* Two conflicting ways of addressing this question immediately present themselves: ignore the other attributes or fix their values. These interpretations correspond to the *totalitarian* semantics and the *ceteris paribus* semantics, respectively. According to the totalitarian semantics – a term we use following Hansson's notion of *totality account* [14] – s' implies that any tuple in which `category` = *minivan* and `ext-color` = *red* is preferred to any tuple in which `category` = *SUV* and `ext-color` = *white*. In particular, in our example we will have $\succ_{s'} = \{t_1 \succ_{s'} t_7,\ t_1 \succ_{s'} t_8,\ t_2 \succ_{s'} t_7,\ t_2 \succ_{s'} t_8\}$. On the other hand, according to the *ceteris paribus* semantics, s' implies that a tuple in which `category` = *minivan* and `ext-color` = *red* is preferred to a tuple in which `category` = *SUV* and `ext-color` = *white*, *provided* that both tuples agree on the value of all other attributes. (Hence the name *ceteris paribus*, which stands for "all else being equal" in Latin.) Under the *ceteris paribus* semantics, in our example we will have $\succ_{s'} = \{t_1 \succ_{s'} t_7,\ t_2 \succ_{s'} t_8\}$.

Authors in the DB community seem to implicitly favor the totalitarian semantics (*e.g.*, see the query examples in [7, 16, 22]), perhaps because it provides a much stronger preference order than the *ceteris paribus* semantics, making the set of "optimal" tuples smaller, and perhaps because it appears to have attractive computational properties. The *ceteris paribus* semantics, on the other hand, seems more faithful to the actual content of the preference statement, and appears to be almost uniformly favored by philosophers [13, 14], economists [2], and AI researchers [3, 11, 21].

As illustrated in example Figure 1(b), preference queries typically consist of a number of preference statements. This begs the second fundamental semantic question: *How should a set of preference orders be aggregated into a single preference order?* Previous work considered two types of aggregation operators, namely boolean and prioritized compositions (*e.g.*, see [7, 16]). Boolean composition considers all the preference statements in $\mathcal{Q}$ as equally important and combines the corresponding preference relations using one of the set operators such as intersection or union. In contrast, prioritized composition considers preference statements in $\mathcal{Q}$ according to some hierarchy of importance $\triangleright \subset \mathcal{Q} \times \mathcal{Q}$, where $s_i \triangleright s_j$ means that statement s_i is more important to the user than statement s_j. According to the prioritized composition rule, for every pair of tuples $t, t' \in R$, we have $t \succ_{\mathcal{Q}} t'$ if and only if $[\exists i\ :\ t \succ_i t'] \wedge [\forall j\ :\ [s_j \triangleright s_i] \rightarrow [t' \not\succ_j t]]$. The importance hierarchy $\triangleright$ must either be a part of the inputed preference query or must be derived from the given statements somehow. In the first case, additional strain is placed on the user. In the second case, we run into non-trivial semantic and computational issues very much similar to those that arise in non-monotonic reasoning, and to which there is no agreed-upon solution. Thus, in this paper, we focus on boolean composition.

After constructing the composed preference relation $\succ_{\mathcal{Q}}$, the database system should evaluate the query and extract those tuples that fit the requirements. Unlike standard, concrete database queries, the result of evaluating a preference query is not well defined. However, a consensus on this issue seems to have been reached: The query evaluation operator that has been (independently) proposed in [7, 16, 22] retrieves all the undominated tuples from R. In what follows, we refer to this operator as BEST. It is formally defined as follows: $\mathsf{BEST}(R, \succ_{\mathcal{Q}}) = \{t \in R \mid \forall t' \in R\ :\ t' \not\succ_{\mathcal{Q}} t\}$. In [6] it is shown that, depending on various properties of $\succ_{\mathcal{Q}}$, several algorithms can be used to implement $\mathsf{BEST}(R, \succ_{\mathcal{Q}})$. The main point to note is that all these algorithms incrementally eliminate every tuple, t, for which there is a dominating tuple $t' \succ_{\mathcal{Q}} t$. This seemingly innocuous step has a dramatic impact on the computational complexity of the BEST operator, as we later show.

2.2 The Scope of Preference Queries

To formalize the notion of a preference query, we introduce the following notation: Let $\mathcal{A}(\mathbb{R}) = \{a_1, \cdots, a_n\}$ denote the set of attributes specifying the relation schema $\mathbb{R}$, and let $\mathcal{D}(a_1), \cdots, \mathcal{D}(a_n)$ denote the domains of these attributes, respectively. The *tuple space* of the schema $\mathbb{R}$ is the set $\mathcal{D}(\mathbb{R}) = \times\mathcal{D}(a_i)$. (We use $\mathcal{D}(\cdot)$ denote the domain of a set of attributes as well.)

In this paper we restrict our attention to *schema-dependent queries over single attributes*, which seem to cover all the examples we have seen in the DB literature. The core requirement will be that preferences are expressed in terms of the values of the schema's attributes only. A preference statement s over the relation schema $\mathbb{R}$ is called *schema-dependent* if and only if there exist two disjoint subsets of attributes $\mathcal{A}^c(s), \mathcal{A}^r(s) \subseteq \mathcal{A}(\mathbb{R})$ (the subscripts stand for "conditioning" and "reference", respectively), such that $\mathcal{A}^r(s) \neq \emptyset$ and s can be presented as: $\alpha \Rightarrow \langle\{\beta_1 \succ \beta_1'\}, \cdots, \{\beta_l \succ \beta_l'\}\rangle$, where $\alpha \in \mathcal{D}(\mathcal{A}^c(s))$, and, for $1 \leq j \leq l$, we have $\beta_j, \beta_j' \in \mathcal{D}(\mathcal{A}^r(s))$ and $\beta_j \neq \beta_j'$. A preference query $\mathcal{Q} = \{s_1, \cdots, s_m\}$ is called *schema-dependent* if and only if each statement $s_i \in \mathcal{Q}$ is schema-dependent.

To illustrate the notion of schema-dependent queries, assume we have a database schema containing the attributes `category, ext-color, sunroof`, and `price`. Consider the statement s_1: "In red sports car, I prefer a sunroof." This statement can be written as:

$$[\texttt{category} = \text{Sport}] \wedge [\texttt{ext\_color} = \text{Red}] \Rightarrow \langle\{[\texttt{sunroof} = \text{Yes}] \succ [\texttt{sunroof} = \text{No}]\}\rangle$$

where $\mathcal{A}^c(s_1) = \{\texttt{category}, \texttt{ext\_color}\}$, and $\mathcal{A}^r(s_1) = \{\texttt{sunroof}\}$. Now, consider the statement s_2: "I prefer to pay less." Here we have $\mathcal{A}^r(s_2) = \{\texttt{price}\}$ and, since s_2 is unconditional, we have $\mathcal{A}^c(s_2) = \emptyset$. Notice that s_2 induces a preference relation between all possible values of `price`, making the size of the relation $\succ_2$ extremely large. To work with such preference relations, we must represent them implicitly, as in: $\Rightarrow \langle\{\text{p} \succ \text{p}' \mid \text{p} < \text{p}'\}\rangle$. To simplify the presentation, in the rest of the paper we assume that the domain size of each attribute is bounded by some constant. This assumption causes no loss of generality, and all the results and discussion in the paper apply to naturally ordered infinite domains as in the example above.

Our second restriction is with respect to the size of the set $\mathcal{A}^r$ describing the number of attributes being varied. Consider the following three preference statements:

s_1	*I prefer to have a sunroof.*
s_2	*In sport cars, I prefer to have a sunroof.*
s_3	*I prefer sports car with a sunroof to family sedans without a sunroof.*

The first statement expresses (unconditional) preference over the values of a single attribute – `sunroof`. The second statement expresses a conditional preference over the value of a single attribute: when `category` $= Sport$, I prefer `sunroof` $= Yes$. Although two attributes appear in this preference statement, one only serves to constrain the set of tuples to which the preference for sunroof apply. Thus, in both s_1 and s_2, we have $|\mathcal{A}^r| = 1$. The third statement expresses a preference between two assignments to *two* attributes: `category` and `sunroof`. Here, $|\mathcal{A}^r| = 2$.

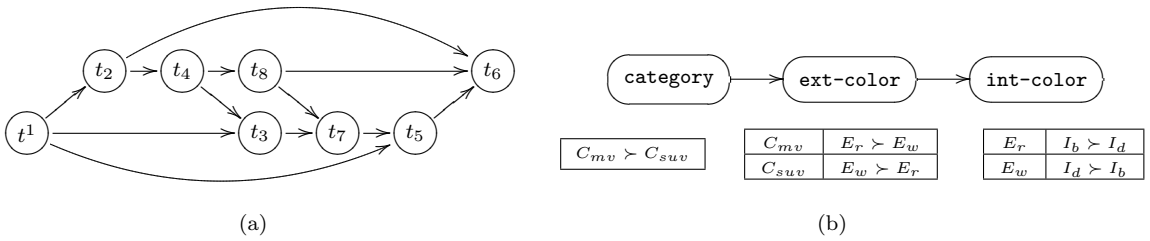
¿From a semantic point of view, the size of $|\mathcal{A}^r|$ is inconsequential. However, it does affect the complexity of various operations. Generally speaking, preference statements where $|\mathcal{A}^r| > 1$ are not very natural for users to express. Indeed, all the examples that we have seen in the DB literature deal with preferences over the value of a single attribute, possibly conditioned by concrete assignments to one or more additional attributes. This is why we restricted our attention to preference statements in which $|\mathcal{A}^r| = 1$. Finally, we assume that the preference relation induced by a preference query $\mathcal{Q}$ forms a strict partial order over $\mathcal{D}(\mathbb{R})$ (*i.e.* we assume that $\succ_{\mathcal{Q}}$ is irreflexive, asymmetric, and transitive).

3. The Totalitarian Semantics

The totalitarian semantics together with aggregation using intersection appears to be the implicit choice of past authors in the DB community [7, 16, 22]. The goal of this section is to show that this semantics is inadequate. We demonstrate this with a number of examples. First, consider the single preference statement: "I prefer choclate ice-cream to vanilla ice-cream." If expressed in the context of selecting ice-cream flavors only, this statement is innocuous. But suppose it is expressed in the context of selecting a meal. The implications according to the totalitarian semantics is that any bizarre choice for main course is preferred to any other choice provided the first comes with choclate ice-cream and the second comes with vanilla ice-cream. This should already disqualify this semantics in the eye of many readers because it is unlikely to match the user's intentions. Of course, one could argue that if this is the only preference the user expressed, then this result is warranted. That is not unreasonable, so let us consider what happens when we have more than one preference statement.

Consider the two statements: (s_1) "I prefer red to white as the color of my car," and (s_2) "I prefer minivans to family sedans." If we compose the preference orders induced by the totalitarian semantics for these statements using union aggregation, we get an *inconsistent* preference relation: According to s_1 a red family sedan is preferred to a white minivan, whereas according to s_2 a white minivan is preferred to a red family sedan. Since users are highly likely to express preferences for values of different schema attributes separately, we are likely to face this problem often. This is perhaps why authors have looked into the intersection operator instead. Hence, consider a third example: (s_1) – "In minivans, I prefer Ford to Chrysler," and (s_2) – "In SUVs, I prefer Toyota to Isuzu." These two natural statements describe preferences in different contexts. The intersection of the corresponding preference relations is *empty.* This illustrates an odd aspect of intersection composition: the addition of new preference statements may cause us to ignore the information in previous statements.

Of course, more complex aggregation operators that are semantically attractive may exist. For example, we could use union on disjoint contexts and intersection on similar contexts. Unfortunately, this suggestion is not trivial to operationalize: the contexts of two statements can overlap in various ways, and the nature of overlaps become more and more complicated the more statements we have. Another option is to use union with priorities. If we require priorities from the user, we are making the preference elicitation process more complex. If the user specifies similar priorities for two preferences on disjoint attributes, we obtain an inconsistent relation again. Trying to infer priorities automatically is both conceptually non-trivial, and can have serious computational cost. Indeed, such attempts are reminiscent of the problem of agreeing on the meaning of default statements in non-monotonic reasoning. The accepted semantics for default statements is in terms of a preference order over truth assignments [17, 20] – much like the type of structures we consider. The literature in that area is full of alternative interpretations, some of which employ explicit and implicit priorities. Moreover, the computational complexity of inference in these formalism is often prohibitive. The only redeeming aspect of the totalitarian semantics with intersection is that it is computationally cheap to compare tuples. But given its serious semantic problems, this provides little consolation. In the next section, we study an alternative approach based on the *ceteris paribus* semantics and the union operator, which, because of its conservative nature, evades the semantic pitfalls of the totalitarian semantics.



(a) (b)

Figure 2: (a) Preference relation between the tuples from Figure 1(a), induced by the query in Figure 1(b) under *ceteris paribus* semantics; (b) the CP-net for this query.

4. The *Ceteris Paribus* Union Semantics

"When discussing with my wife what table to buy to our living room, I said: 'A round table is better than a square one.' By this I did not mean that irrespectively of their other properties, any round table is better than a square-shaped table. Rather, I meant that any round table is better (for our living room) than any square table that does not differ significantly in its other characteristics, such as height, sort of wood, finishing, price, etc. This is preference *ceteris paribus* or "everything else being equal". Most of the preferences that we express or act upon seem to be of this type." This passage from [13] concisely expresses the motivation shared by authors in diverse areas for adopting the *ceteris paribus* semantics for preference statements.

This more conservative semantics does not lead to controversial conclusions. Those who may view this as a drawback because fewer tuples are now comparable must remember that meaningful preference queries usually involve multiple preference statements. And while union-based aggregation is problematic with totalitarianism, it works just fine with the *ceteris paribus* semantics. Thus, as we would expect, the more preference statements we have, the more tuples that are comparable and the more intricate the preference order obtained. And without unintended side-effects. ¿From now on, when we use the term *ceteris paribus* semantics we mean the *ceteris paribus* semantics with union-based aggregation.

To illustrate the *ceteris paribus* semantics, consider our running example in Figure 1. The union of the preference relations $\succ_1, \cdots, \succ_5$ under *ceteris paribus* semantics is graphically depicted in Figure 2(a): The nodes stand for the tuples $t_1, \ldots, t_8$, and there is a directed edge from t to t' if for one of the statements s_i, $1 \leq i \leq 5$, we have $t \succ_i t'$. The resulting preference relation is specified by the transitive closure of this graph.

In the past few years our understanding of the *ceteris paribus* semantic and its computational properties has considerably improved. In particular, [4] introduced CP-nets, a graph-based formalism for describing the relationship between preference statements over single attributes in the context of the *ceteris paribus* semantics, capturing the preferential independence assertion inherent in these statements, and [8] showed how the topology of CP-nets affects the complexity of various types of preference queries. A CP-net is a directed graph that is induced by a set of preference statements. In terms of the database applications, the CP-net nodes correspond to the schema's attributes, and a directed edge exists from v to v' if there is a preference statement s such that $v \in \mathcal{A}^c(s)$ and $v' = \mathcal{A}^r(s)$. That is, s describes the preference over the domain of v' conditioned on the value of v and possibly other attributes. Finally, the preference information induced by the statements of $\mathcal{Q}$ on the attributes of $\mathbb{R}$ is annotated with the corresponding nodes in the graph. Formally, CP-nets are defined as follows:

Definition 1 A *CP-net* N over variables $\mathbf{V} = \{X_1, \ldots, X_n\}$ is a directed graph G over the nodes $X_1, \ldots, X_n$, and there is a directed edge from X_i to X_j if the preference over the value

of X_j is conditioned on the value of X_i. Each node $X_i \in \mathbf{V}$ is annotated with a conditional preference table ($CPT(X_i)$) that associates a strict (possibly empty) partial order $\succ_{\mathbf{u}_i}$ with each instantiation $\mathbf{u}_i$ of X_i's parents $\mathbf{U}_i$.

For example, the CP-net $N^{\mathcal{Q}}$ induced by query $\mathcal{Q}$ from our running example in Figure 1 is depicted in Figure 2(b); The tables are the CPTs, and the values $\{C_{mv}, C_{suv}\}$, $\{E_r, E_w\}$ and $\{I_b, I_d\}$ shortly represent the domains $\{minivan, suv\}$, $\{red, white\}$, $\{bright, dark\}$ of the attributes `category`, `ext-color`, and `int-color`, respectively. Since in this particular example we have $R = \mathcal{D}(\mathbb{R})$, Figure 2(a) can be considered as a graphical illustration of a relation, the transitive closure of which is exactly the preference relation induced by $N^{\mathcal{Q}}$ over $\mathbb{R}$: An arc in this graph directed from tuple t_i to tuple t_j indicates that a preference for t_i over t_j can be determined directly from one of the CPTs in the CP-net. For example, the fact that $C_{mv} \wedge E_w \wedge I_d$ is preferred to $C_{mv} \wedge E_w \wedge I_b$ (as indicated by a directed arc between them) is a direct consequence of the *ceteris paribus* semantics of $CPT(I)$.

An important property of the *ceteris paribus* semantics is that acyclic CP-nets always induce strict partial preference orders. When the CP-net is cyclic, such preference cycles may exist, depending on both the network's structure and the particular preferences [9][2]. Of course, cyclic preferences are no less problematic with the totalitarian semantics. Finally, in what follows we call a CP-net N *completely specified* if for each variable X_i, each assignment $\mathbf{u}$ on $\mathbf{U}_i$, we have that $\succ_{\mathbf{u}}$ is a total order over the domain of X_i.

5. The BEST and ORD Operators

Our ultimate goal is to provide quick and appropriate responses to a preference query made by a user. At this point, we hope to have convinced the reader that, semantically, there is only one adequate option – the *ceteris paribus* semantics. Now, we take a closer look at the computational cost associated with evaluating queries under this semantics.

Consider a preference relation $\succ_{\mathcal{Q}}$ defined by a schema-dependent query $\mathcal{Q} = \{s_1, \cdots, s_m\}$. The complexity of answering preference queries should be measured as a function of $|R|, n$ (the number of attributes), and m. Recall that the BEST operator was defined as: $\mathsf{BEST}(R, \succ_{\mathcal{Q}}) = \{t \in R \mid \forall t' \in R \ : \ t' \not\succ_{\mathcal{Q}} t\}$, and that a basic sub-routine in all algorithms for computing BEST is *dominance-testing*, *i.e.* comparing between two tuples to determine whether one is better than the other. Unfortunately, here we ran into a problem: In [3, 8, 19] it is shown that the Achilles heel of all but the most simplistic qualitative preference representation models is exactly the complexity of dominance testing. For instance, the results in [3, 8] show that even for schema-dependent queries forming acyclic CP-nets over binary-valued attributes, the problem is NP-hard, and for non-binary attributes it is even not in NP. Our conclusion at this point is somewhat pessimistic: at least theoretically, the worst-case complexity of BEST is not appropriate for database systems. In the rest of this section we describe an alternative to BEST, the ORD operator, which can be computed in low polynomial time for many queries that are problematic for BEST.

ORD is an alternative to the BEST operator that immediately presents itself: It is based on sorting the given data set R according to the query preference relation $\succ_{\mathcal{Q}}$, and providing the user with the top k tuples of R in a *non-increasing* order of preference, possibly extending the presentation on demand. ORD is closely related to the standard ORDER BY operator of SQL in which a qualitative preference query is used as the metric for comparison between tuples. Formally, ORD is defined as follows:

2. A weaker notion of consistency discussed in [5, 10] can be used to answer certain queries in cyclic nets.

Definition 2 Given a relation schema $\mathbb{R}$, let $\mathcal{Q}$ be a schema-dependent preference query inducing a strict partial preference order $\succ_{\mathcal{Q}}$ over $\mathcal{D}(\mathbb{R})$. Given a relation instance R of $\mathbb{R}$, $\mathsf{ORD}(R, \succ_{\mathcal{Q}})$ contains all the tuples of R, totally ordered such that, for every $t, t' \in R$, if t appears on $\mathsf{ORD}(R, \succ_{\mathcal{Q}})$ before t', then we have $t' \not\succ_{\mathcal{Q}} t$.

Informally, ORD provides us with a total-order over the database relation that is consistent with $\succ_{\mathcal{Q}}$: If $t \succ_{\mathcal{Q}} t'$, then we know that ORD will show t prior to t', but if t and t' are incomparable according to $\succ_{\mathcal{Q}}$, then ORD will order them arbitrarily. Observe that there is no apparent computational difference between the operators BEST and ORD, since both correspond to this or another form of sorting, and comparing between the elements of the list is a basic part of any sorting algorithm. Hence, using ORD instead of its stronger counterpart BEST seems to be a bad idea in the first place. However, below we show that there is slight difference between BEST and ORD that turns out to be (complexity-wise) crucial.

In [3] it is shown how one can efficiently order a set of tuples with respect to a completely specified CP-net, and this despite the NP-hardness of dominance-testing in these networks. In terms of database queries, this result can be stated as follows:

Theorem 1 (based on [3]) *Let $\mathcal{Q}$ be a schema-dependent preference query over a relation schema $\mathbb{R}$ with n attributes, R be an instance of $\mathbb{R}$, and $|R| = D$. If $\mathcal{Q}$ induces an acyclic, completely specified acyclic CP-net, then $\mathsf{ORD}(R, \succ_{\mathcal{Q}})$ can be computed in time $O(nD^2)$.*

¿From the perspective of database preference queries, this is a key result for the *ceteris paribus* semantics. Unfortunately, this result requires the CP-net in question to be completely specified. That is, for each attribute, we must provide a total order over its values for every possible value assignment to its parents in the CP-net. This is not a very attractive requirement from the database perspective, as it requires the user to supply much information. Moreover, quadratic complexity in the size of the database relation, especially for the sort of very large online databases we have in mind, can be problematic. Below we show how to obtain a similar result, but with an incompletely specified acyclic CP-net, *i.e.*, with any schema-dependent query over single attributes that induces an acyclic CP-net graph.[3] Moreover, the computational complexity is reduced to $O(nD \log D)$.

Let us begin with a simple but very important observation that provides a key distinction between ORD and BEST. To order a pair of tuples t and t' consistently with a preference relation $\succ_{\mathcal{Q}}$, we can be satisfied by knowing only that $t \not\succ_{\mathcal{Q}} t'$ or $t' \not\succ_{\mathcal{Q}} t$. Note that this information is weaker than knowing the exact preference relationship between t and t'. Now consider the following important auxiliary lemma:

Lemma 2 *Let N be an acyclic CP-net, and $t \neq t'$ be a pair of complete assignments on the variables of N. Let X_i be a variable in N such that t and t' assign the same values to all ancestors of X_i in N, and different values to X_i. If, given the assignment $\mathbf{u}$ provided by t (and t') to $\mathbf{U}_i$, we have $t[X_i] \succ_{\mathbf{u}} t'[X_i]$, then we have $N \not\models t' \succ t$. Otherwise, if $t[X_i]$ and $t'[X_i]$ are incomparable given $\mathbf{u}$, we have both $N \not\models t' \succ t$ and $N \not\models t \succ t'$.*

It is not hard to see that the condition presented by Lemma 2 can be verified in time $O(n)$ by a top-down traversal of the CP-net. In what follows, we refer to this procedure as *ordering operator*. The only problematic point is that Lemma 2 presents a condition that is sufficient but not necessary for the truth of the query $N \not\models t' \succ t$, *i.e.* our *ordering operator* is incomplete.

3. We note again, cyclic preferential dependencies are semantically and computationally complicated for both totalitarian and *ceteris paribus* semantics.

order–pair (N, t, t')

1. Let $\mathbf{V}_{t,t'}$ be the set of all variables X_i, such that t and t' assign different values to X_i but the same values to all ancestors of X_i in N (in particular, $\mathbf{u}_t = \mathbf{u}_{t'}$ for all such X_i). Identify $\mathbf{V}_{t,t'}$ by a top-down traversal of N.
2. Fix a total topological ordering $\triangleright$ of the variables of N. For each variable $X_i \in N$, and each assignment $\mathbf{u}$ to $\mathbf{U}_i$, fix a total order $>_{\mathbf{u}}$, consistent with the strict partial order $\succ_{\mathbf{u}}$ specified by $CPT(X_i)$.
3. For each $X_i \in \mathbf{V}_{t,t'}$, let $\mathbf{u}^*$ be the assignment to $\mathbf{U}_i$ made by t (and t'). If, for each $X_i \in \mathbf{V}_{t,t'}$, we have $t[X_i] >_{\mathbf{u}^*} t'[X_i]$, then return $t \gg t'$. Otherwise, if for each $X_i \in \mathbf{V}_{t,t'}$, we have $t'[X_i] >_{\mathbf{u}^*} t[X_i]$, then return $t' \gg t$.
4. Otherwise, order $\mathbf{V}_{t,t'}$ with respect to $\triangleright$, and pick the first variable X_i in the sorted $\mathbf{V}_{t,t'}$. If $t[X_i] >_{\mathbf{u}^*} t'[X_i]$, then return $t \gg t'$, otherwise return $t' \gg t$.

Figure 3: A complete procedure for ordering a pair of complete assignments consistently with a given CP-net.

For instance, consider the tuples $t_3 = C_{mv} \wedge E_w \wedge I_b$ and $t_8 = C_{suv} \wedge E_w \wedge I_d$ in our running example from Figures 1 and 2. According to the CP-net of the query in question, these two assignments are incomparable (*i.e.*, neither can be proven to be preferred to the other). However, $N \not\models t_3 \succ t_8$ cannot be deduced using the condition of Lemma 2, because `category` is the only root variable of this CP-net, and t_3 assigns it a more preferred value than that assigned by t_8. Hence, the only conclusion so far (and not very useful one) is that some queries of the form $N \not\models t' \succ t$ can be answered efficiently. Fortunately, the following result shows that the ordering operator *is* complete in a weaker, yet sufficiently strong sense.

Lemma 3 *Given an acyclic CP-net N, and two complete assignments t and t' on the variables of N, the truth of at least one of the queries $N \not\models t' \succ t$ or $N \not\models t \succ t'$ can be determined using a pair of the corresponding ordering operators.*

Using this "partial completeness" of the algorithm for paired queries stated by Lemma 3, we can provide an enhanced version of the ordering operator that defines a complete extension $\gg$ of the preference ordering $\succ$ induced by the CP-net. The enhanced ordering operator order–pair is specified in Figure 5, and its correctness is asserted in Theorem 4.

Theorem 4 *Given an acyclic CP-net N over the variable set $\mathbf{V}$, the preference relation $\gg$ induced by the* order–pair *operator is a total order (*i.e.*, complete, irreflexive, anti-symmetric, and transitive) over $\mathcal{D}(\mathbf{V})$, consistent with the preference relation $\succ$ induced by N.*

Given the set-theoretic properties of $\gg$ described in Theorem 4, we can now proceed with our key result for the *ceteris paribus* semantics.

Theorem 5 *Let $\mathcal{Q}$ be a schema-dependent preference query over a relation schema $\mathbb{R}$ with n attributes, R be an instance of $\mathbb{R}$, and $|R| = D$. If $\mathcal{Q}$ induces an acyclic CP-net, then $\mathsf{ORD}(R, \succ_{\mathcal{Q}})$ can be computed in time $O(nD \log D)$.*

Using the previous results, the proof of Theorem 5 is straightforward: First, it is easy to see that the complexity of order–pair is $O(n)$. Second, since $\gg$ forms a total order, (*i.e.* every two tuples in $\mathcal{D}(\mathbb{R})$ are comparable with respect to $\gg$), we can use any sorting mechanism to implement ORD.

6. Conclusions

We showed that the totalitarian intersection semantics for database preference queries is inappropriate. The *ceteris paribus* semantics is much more appealing, but it computationally prohibitive when we attempt to compute the BEST operator. Instead, we proposed the use of the ORD operator, a relaxation of BEST that can be implemented efficiently for wide classes of preference queries. A closer inspection of the ORD operator shows interesting relationship to the totalitarian semantics. In a nut-shell, ORD leads to a flexible totalitarian-like approximation of BEST, with an implicit priority over preferences. For further discussion on this issue, as well as for analysis of the interplay between ORD and standard relational operators, and a discussion of the relative merits of qualitative and quantitative models for handling preferences in database systems, we refer the reader to the full paper.

References

[1] R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences. In *SIGMOD-00*, pages 297–306, 2000.

[2] H. Bierens and N. Swanson. The econometric consequences of the ceteris paribus condition in economic theory. *Journal of Econometrics*, 95(2):223–253, 2000.

[3] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning about conditional *ceteris paribus* preference statements. *Journal of Artificial Intelligence Research*, 2003. to appear.

[4] C. Boutilier, R. Brafman, H. Hoos, and D. Poole. Reasoning with conditional *ceteris paribus* preference statements. In *UAI-99*, pages 71–80, 1999.

[5] R. I. Brafman and Y. Dimopoulos. A new look at the semantics and optimization methods of CP-networks. In *IJCAI-03*, pages 1033–1038, 2003.

[6] J. Chomicki. Preference queries in relational databases. *ACM Transactions on Database Systems*. to appear.

[7] J. Chomicki. Querying with intristic preferences. In *EDBT-02*, pages 34–51, 2002.

[8] C. Domshlak. *Modeling and Reasoning about Preferences with CP-nets*. PhD thesis, Ben-Gurion University, 2002.

[9] C. Domshlak and R. Brafman. CP-nets - reasoning and consistency testing. In *KR-02*, pages 121–132, 2002.

[10] C. Domshlak, F. Rossi, C. Venable, and T. Walsh. Reasoning about soft constraints and conditional preferences: Complexity results and approximation techniques. In *IJCAI-03*, pages 215–220, 2003.

[11] J. Doyle and M. Wellman. Representing preferences as ceteris paribus comparatives. In *Proceedings of the AAAI Spring Symposium on Decision-Theoretic Planning*, pages 69–75, March 1994.

[12] K. Govindarajan, B. Jayaraman, and S. Mantha. Preference queries in deductive databases. *New Generation Computing*, pages 57–86, 2001.

[13] S. O. Hansson. What is ceteris paribus preference. *Journal of Philosophical Logic*, 25(3):307–332, 1996.

[14] S. O. Hansson. Preference logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 4, pages 319–394. Kluwer, 2 edition, 2001.

[15] V. Hristidis, N. Koudas, and Y. Papakonstantinou. PREFER: A system for the efficient execution of multi-parametric ranked queries. In *SIGMOD-01*, pages 259–269, 2001.

[16] W. Kießling. Foundations of preferences in database systems. In *VLDB-02*, 2002.

[17] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.

[18] M. Lacroix and P. Lavency. Preferences: Putting more knowledge into queries. In *VLDB-87*, pages 217–225, 1987.

[19] J. Lang. From preference representation to combinatorial vote. In *KR-02*, pages 277–288, 2002.

[20] Y. Shoham. A semantics approach to non-monotonic logics. In *IJCAI-87*, pages 388–392, 1987.

[21] S. W. Tan and J. Pearl. Qualitative decision theory. In *AAAI-94*, pages 928–933, 1994.

[22] R. Torlone and P. Ciaccia. Why are my preferred items? In *Workshop on Recommendation and Personalization in E-Commerce*, 2002.

Appendix A. Proofs

Lemma 2 Let N be an acyclic CP-net, and $t \neq t'$ be a pair of complete assignments on the variables of N. Let X_i be a variable in N such that t and t' assign the same values to all ancestors of X_i in N, and different values to X_i. If, given the assignment $\mathbf{u}$ provided by t (and t') to $\mathbf{U}_i$, we have $t[X_i] \succ_{\mathbf{u}} t'[X_i]$, then we have $N \not\models t' \succ t$. Otherwise, if $t[X_i]$ and $t'[X_i]$ are incomparable given $\mathbf{u}$, we have both $N \not\models t' \succ t$ and $N \not\models t \succ t'$.

Proof: Given such a variable X_i, suppose that we have $t[X_i] \succ_{\mathbf{u}} t'[X_i]$, and assume to the contrary that $N \models t' \succ t$. The semantics of CP-nets entails that there exists a sequence of outcome improvements from t to t', where each improvement is sanctioned by one of the CPTs in N (for a formal definition, see the notion of flipping sequences in [3]). Since in the current context $\mathbf{u}$, $t[X_i]$ is not improvable to $t'[X_i]$, one of the variables in $\mathbf{U}_i$ will have to be improved first. However, changing an assignment on a rooted subgraph of N, and restoring it back (recall how we picked X_i at the first place) is impossible. Otherwise, it will entail that acyclic CP-net may represent inasymmetric orders, and this will violate the very basic Theorem 1 in [3]. Hence, we proved that $N \not\models t' \succ t$. The proof for the case of $t[X_i]$ and $t'[X_i]$ incomparable given $\mathbf{u}$ is by a similar demonstration that there are no sequences of local improvements neither from t to t', nor from t' to t. ∎

Lemma 3 Given an acyclic CP-net N, and two complete assignments t and t' on the variables of N, the truth of at least one of the queries $N \not\models t' \succ t$ or $N \not\models t \succ t'$ can be determined using a pair of the corresponding ordering operators.

Proof: Due to the acyclicity of N, a variable X satisfying the conditions of Lemma 2 has to exist for at least one of the queries $N \not\models o' \succ o$ and $N \not\models o \succ o'$ (and possibly for both). Otherwise, it has to be the case that t is identical to t'. ∎

Theorem 4 Given an acyclic CP-net N over the variable set $\mathbf{V}$, the preference relation $\gg$ induced by the order–pair operator is a total order (*i.e.*, complete, irreflexive, anti-symmetric, and transitive) over $\mathcal{D}(\mathbf{V})$, consistent with the preference relation $\succ$ induced by N.

Proof: The completeness of $\gg$ is immediate from the definition of order–pair. Therefore, we proceed with showing that the transitive closure of the relation $\gg$ is asymmetric. Assume to the contrary that there exists a set of assignments $t_1, \ldots, t_k$ such that:

$$t_1 \gg t_2 \gg \cdots \gg t_k \gg t_1 \tag{1}$$

For $1 \leq i \leq k$, let $V(t_i)$ be the set of all variables X such that, given the assignment $\mathbf{u}$ provided by t_i to $\mathbf{U}_X$, the value $t_i[X]$ can be improved with respect to $>_{\mathbf{u}}$ used by order–pair. Let N_i be the subgraph of N consisting of the variables in $V(t_i)$ and their descendants in N.

By construction of order–pair, we have $N_i \not\supset N_{i+1}$ for $1 \leq i < k$, and $N_k \not\supset N_1$. To see this, notice that if, for some i, we have $N_i \supset N_{i+1}$, then:

1. There exists a variable X such that: (i) all ancestors of X are assigned by both t_i and t_{i+1} to their highest values with respect to $>_{\mathbf{u}}$, where $\mathbf{u} = t_i[\mathbf{U}_X] = t_{i+1}[\mathbf{U}_X]$; and (ii) according to $>_{\mathbf{u}}$, X is assigned to its highest value by t_{i+1} and one of the other values by t_i.

2. There is no variable X such that: (i) all ancestors of X are assigned by both t_i and t_{i+1} to their highest values with respect to $>_{\mathbf{u}}$; and (ii) according to $>_{\mathbf{u}}$, X is assigned to its highest value by t_i and one of the other values by t_{i+1}.

However, this contradicts our assumption that (order–pair will return) $t_i \gg t_{i+1}$.

For $1 \leq i \leq k$, let X^i be the highest variable in $V(t_i)$ according to the total order $\triangleright$ used by order–pair. Recall that, for $1 \leq i \leq k$, we have either $N_i \subseteq N_{i+1}$, or both $N_i \setminus N_{i+1} \neq \emptyset$ and $N_{i+1} \setminus N_i \neq \emptyset$ (*i.e.* the sets of root nodes of N_i and N_{i+1} are not included one in the other). Now, if $N_i \subseteq N_{i+1}$, then each variable node in N_i is either a root node in N_{i+1}, or is a descendant of one of these roots. Therefore, we have either $X^{i+1} \triangleright X^i$, or $X^{i+1} = X^i$. In the second case of mutual non-inclusion of N_i and N_{i+1}, the same relationship between X^i and X^{i+1} holds by the definition of order–pair. (All the above holds for $(X^1 \triangleright X^k) \vee (X^1 = X^k)$).

Now, if for some $1 \leq i \leq k$ we have $X^{i+1} \triangleright X^i$ (and not $X^{i+1} = X^i$), the initial assumption (1) is trivially contradicted. Therefore, we are left with the case of:

$$X^1 = X^2 = \cdots = X^k = X$$

Now, by definition of $X^1, \ldots, X^k$, we have:

$$t_1[\mathbf{U}_X] = t_2[\mathbf{U}_X] = \cdots = t_k[\mathbf{U}_X] = \mathbf{u}$$

This must be the case since all the ancestors of X are assigned to their unique assignment (of which $\mathbf{u}$ is a part) that is not improvable with respect to the set of total orderings $>_{\mathbf{u}}$ defined by order–pair. This entails

$$t_1[X] >_{\mathbf{u}} t_2[X] >_{\mathbf{u}} \cdots >_{\mathbf{u}} t_k[X] >_{\mathbf{u}} t_1[X],$$

which is inconsistent with the definition of CP-nets and step 2 of order–pair. Hence, we have accomplished the proof that $\gg$ is a total order over $\mathcal{D}(\mathbf{V})$. Finally, if $N \models t \succ t'$, then, by definition of order–pair, we must have $t \gg t'$. Therefore, the total order $\gg$ is consistent with the relation $\succ$ induced by N. ∎