

Profiling Sets for Preference Querying

Xi Zhang and Jan Chomicki

Department of Computer Science and Engineering
University at Buffalo, SUNY, U.S.A.
{xizhang, chomicki}@cse.buffalo.edu

Abstract. We propose a logical framework for set preferences. Candidate sets are represented using *profiles* consisting of scalar *features*. This reduces set preferences to tuple preferences over set profiles. We also give a heuristic algorithm for the computation of the “best” sets.

1 Introduction

In recent years, *preferences* have been well studied in the database and AI literature [1–5]. The issues addressed in that research include preference specification, preference query languages, and preference query evaluation and optimization. However, the research on preferences has almost exclusively focused on *object (or tuple) preferences* which express preference relationships between individual objects or tuples in a relation.

We observe that in decision making a user sometimes needs to make a *group* decision based not only on the individual object properties but also on the properties of the group as a whole. Consider the following example.

Example 1. Alice is buying three books as gifts. Here is a list of book quotes collected from different vendors:

	title	genre	rating	price	vendor
	a_1	sci-fi	5.0	\$15.00	Amazon
	a_2	biography	4.8	\$20.00	B&N
	a_3	sci-fi	4.5	\$25.00	Amazon
Book:	a_4	romance	4.4	\$10.00	B&N
	a_5	sci-fi	4.3	\$15.00	Amazon
	a_6	romance	4.2	\$12.00	B&N
	a_7	biography	4.0	\$18.00	Amazon
	a_8	sci-fi	3.5	\$18.00	Amazon

Alice needs to decide on the three books to buy. She might have any of the following preferences:

- (C1) She wants to spend as little money as possible.
- (C2) She needs one sci-fi book.
- (C3) Ideally, she prefers that all three books are from the same vendor. If that is not possible, she prefers to deal with as few vendors as possible.
- (C4) She wants the second highest rating in the final choice set to be above 4.5.

In addition, Alice might have different combinations of the above preferences. For example, Alice might have both (C1) and (C2), but (C2) may be more important than (C1) to her.

The preference (C1) can be directly simulated by a *tuple preference* over *Book*, such that for any $t_1, t_2 \in \text{Book}$, t_1 is preferred to t_2 if and only if $t_1.\text{price} < t_2.\text{price}$. Then the top 3 books in *Book* (according to this preference) constitute the “best” answer set.

However, in the other cases, e.g. (C2-C4) and the prioritized composition of (C2) and (C1), such a simulation is not possible.

Example 1 motivates our framework for *set preferences*. We observe that a large class of set preferences has two components: (1) Quantities of interest; (2) Desired value or order of those quantities.

Example 2. In Example 1,

	Quantity of Interest	Desired Value or Order
(C1)	total cost	<
(C2)	number of sci-fi books	1
(C3)	number of distinct vendors	<
(C4)	the second highest rating	> 4.5

Consequently, our framework consists of two components: (1) *profiles*: collections of *features*, each of those capturing a *quantity of interest*; (2) *profile preference relations* to specify *desired values or orders*.

The main idea is to profile candidate subsets based on their *features*. Since each *profile* is a tuple of features, the original *set preference* can now be formulated as a *tuple preference* over the *profiles*.

In the rest of the paper, we elaborate our framework. We also discuss the computational issues involved in computing the “best” subsets. We show a heuristic algorithm for this task.

2 Basic Notions

We make the standard assumptions of the relational model of data. In particular, we assume that we have two attribute domains: rational numbers (\mathcal{Q}) and uninterpreted constants (\mathcal{D}).

We assume that set preferences are defined over sets of the same fixed cardinality.

Definition 1 (*k*-subset). Given a relation r and a positive integer k , $k \leq |r|$, a k -subset s of r is a subset of r with cardinality k , i.e. $s \subseteq r$ and $|s| = k$. Denote by $k\text{-subsets}(r)$ the set of all k -subsets of r .

We capture the *quantities of interest* for k -subsets using *k-subset features*.

Definition 2 (*k*-subset Feature). Given a relation r and a positive integer $k \leq |r|$, a k -subset feature $\mathcal{A}(\cdot)$ is a function: $k\text{-subsets}(r) \mapsto U$, where U (either \mathcal{Q} or \mathcal{D}) is the range of this function. The domain of feature \mathcal{A} , denoted by $\text{Dom}(\mathcal{A})$, is U .

Definition 3 (*k*-subset Profile Schema). Given a relation r and a positive integer $k \leq |r|$, a k -subset profile schema Γ is a schema $\langle \mathcal{A}_1, \dots, \mathcal{A}_m \rangle$, where \mathcal{A}_i is a k -subset feature, $i = 1, \dots, m$.

Definition 4 (*k*-subset Profile Relation). Given a relation r and its k -subset profile schema $\Gamma = \langle \mathcal{A}_1, \dots, \mathcal{A}_m \rangle$, the k -subset profile relation γ is defined as

$$\gamma = \{t \mid \exists s \in k\text{-subsets}(r), t = \langle \mathcal{A}_1(s), \dots, \mathcal{A}_m(s) \rangle\}.$$

The tuple $\langle \mathcal{A}_1(s), \dots, \mathcal{A}_m(s) \rangle$ is the profile of s under Γ , denoted by $\text{profile}_\Gamma(s)$.

3 SQL-based Feature Definition

Definition 5 (SQL-based k -subset Features). Given a relation r and a positive integer $k \leq |r|$, a SQL-based k -subset feature \mathcal{A} is defined by a parameterized SQL query of the following type T_0 :

T_0 : SELECT expr FROM S WHERE condition
where

- (1) S is a set parameter whose values can be arbitrary k -subsets of r , i.e. $\text{Dom}(S) = k\text{-subsets}(r)$;
- (2) expr is
 - (i) $\text{aggr}([\text{DISTINCT}] A)$, where $\text{aggr} \in \{\min, \max, \text{sum}, \text{count}, \text{avg}\}$, A is an attribute of r , or
 - (ii) any other legal expression in the SQL SELECT clause, which leads to a categorical query, i.e., a query returning a single value when evaluated over any k -subset s of r , substituted for S ;
- (3) the FROM list contains a single item which is S or an alias for S ;

Example 3. In Example 1, the quantity of interest in (C2) is captured by the k -subset feature \mathcal{A}_2 where $k = 3$, and S is a set parameter that can be substituted by any 3-subset of *Book*.

$\mathcal{A}_2 \equiv \text{SELECT count(title) FROM } S \text{ WHERE genre='sci-fi'$

Similarly, the quantities of interests in (C1), (C3) and (C4) are captured by the k -subset features \mathcal{A}_1 , \mathcal{A}_3 and \mathcal{A}_4 respectively.

$\mathcal{A}_1 \equiv \text{SELECT sum(price) FROM } S$

$\mathcal{A}_3 \equiv \text{SELECT count(DISTINCT vendor) FROM } S$

$\mathcal{A}_4 \equiv \text{SELECT t1.rating FROM } S \text{ t1}$

WHERE 1 = (SELECT count(t2.rating) FROM S t2
WHERE t2.rating > t1.rating)

The exprs of the features \mathcal{A}_1 , \mathcal{A}_2 and \mathcal{A}_3 are of type (i), and therefore we can guarantee syntactically that the feature definition query yields a single value over any k -subset. In the case of \mathcal{A}_4 , we can infer from its semantics that it is *categorical*. However, the syntax of \mathcal{A}_4 does not guarantee that. In fact, if we only slightly change the SELECT attribute in the outer query, for example to `SELECT t1.title`, then the query no longer yields a single value, because we could have multiple different books

with the second highest rating. As we can see now, if the `expr` in the SQL-based feature definition is of type (ii), we really need to study the semantics of the query to determine its *categoricity*.

The following example illustrates a k -subset profile schema and relation based on SQL-based k -subset features in Example 3.

Example 4. Continuing Example 3. Let the k -subset profile schema $\Gamma = \langle \mathcal{A}_1, \mathcal{A}_2 \rangle$. The profile relation γ contains, among others, the following tuples: $(\$60, 2)$, which is the profile of the 3-subsets $\{a_1, a_2, a_3\}$ and $\{a_2, a_3, a_5\}$; and $(\$61, 2)$, which is the profile of $\{a_3, a_7, a_8\}$.

4 Profile-based Set Preferences

Now we can define set preferences over k -subsets as tuple preferences over the corresponding profiles. Typically, a tuple preference relation is defined using a formula [4], as is the case for the tuple preference for (C1) in Example 1.

For a relation schema $R = \langle A_1, \dots, A_m \rangle$, we define the domain of R as the cross product of the domains of its attributes, i.e. $Dom(R) = Dom(A_1) \times \dots \times Dom(A_m)$.

Definition 6 (Tuple Preference [4]). *Given a relation schema $R = \langle A_1, \dots, A_m \rangle$, a tuple preference relation $>$ is a subset of $[Dom(R)]^2$. If for a first order formula C , $C(t_1, t_2) \Leftrightarrow t_1 > t_2$, then the tuple preference is defined by the formula C . We then denote the preference relation by $>_C$.*

Definition 7 (Set Preference). *Given a relation schema $R = \langle A_1, \dots, A_m \rangle$, a positive integer k , a set preference relation \succ is a subset of the product $[k\text{-subsets}(Dom(R))]^2$.*

In principle, set preferences could also be defined using logic formulas. However, *second-order* variables would be necessary. To avoid the conceptual and computational complexity associated with such variables, we consider only set preferences that are induced by profile preferences. Recall that we also restrict sets to be of the same fixed cardinality.

Definition 8 (Profile-based Set Preference). *Let $\Gamma = \langle \mathcal{A}_1, \dots, \mathcal{A}_m \rangle$ be a profile schema and $>_C$ a tuple preference relation, which is a subset of $[Dom(\mathcal{A}_1) \times \dots \times Dom(\mathcal{A}_m)]^2$. A set preference \succ is induced by Γ and $>_C$ if for every set s_1 and s_2 ,*

$$s_1 \succ s_2 \Leftrightarrow \text{profile}_\Gamma(s_1) >_C \text{profile}_\Gamma(s_2).$$

We then denote the set preference relation by $\succ_{(\Gamma, C)}$.

Proposition 1. *If the tuple preference relation $>_C$ is a strict partial order, then for any profile schema Γ , the set preference relation $\succ_{(\Gamma, C)}$ is a strict partial order as well.*

Recall that an essential component of set preferences consists of the *desired values* or *orders* of the *quantities of interests*, which are captured by a *preference relation over profiles*. In fact, in order to elaborate a set preference in our framework, a user needs to do the following:

1. Select an integer k and a relation r whose k -subsets are of interest.
2. Provide a k -subset profile schema by defining k -subset features $\mathcal{A}_1, \dots, \mathcal{A}_m$.
3. Specify the profile preference using a tuple preference formula.

Example 5. Assume $\Gamma = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4\}$. We can define the proper preference formula $C_i, i = 1, \dots, 4$ over Γ , such that individual set preference (C1-C4) is induced by Γ and $>_{C_i}$. For example, we define $\succcurlyeq_{(\Gamma, C_1)}$ as

$$\begin{aligned} s_1 \succcurlyeq_{(\Gamma, C_1)} s_2 &\Leftrightarrow \langle \mathcal{A}_1(s_1), \mathcal{A}_2(s_1), \mathcal{A}_3(s_1), \mathcal{A}_4(s_1) \rangle >_{C_1} \langle \mathcal{A}_1(s_2), \mathcal{A}_2(s_2), \mathcal{A}_3(s_2), \mathcal{A}_4(s_2) \rangle \\ &\Leftrightarrow \mathcal{A}_1(s_1) < \mathcal{A}_1(s_2). \end{aligned}$$

Individual preference formulas can be the building blocks of more complicated preferences, where formulas are assembled to express *union*, *intersection*, *prioritized composition* and *Pareto composition* of preferences [4].

Example 6. Consider the prioritized combination of (C2) and (C1) in Example 1. Let the profile schema $\Gamma = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4\}$, and the preference formula C_0 over Γ be such that

$$\begin{aligned} s_1 \succcurlyeq_{(\Gamma, C_0)} s_2 &\Leftrightarrow (\mathcal{A}_2(s_1) = 1 \wedge \mathcal{A}_2(s_2) \neq 1) \\ &\quad \vee (\mathcal{A}_2(s_1) = 1 \wedge \mathcal{A}_2(s_2) = 1 \wedge \mathcal{A}_1(s_1) < \mathcal{A}_1(s_2)) \\ &\quad \vee (\mathcal{A}_2(s_1) \neq 1 \wedge \mathcal{A}_2(s_2) \neq 1 \wedge \mathcal{A}_1(s_1) < \mathcal{A}_1(s_2)). \end{aligned}$$

Note that formula C_0 is the prioritized composition of C_2 and C_1 .

5 Computing the Best k -subsets

For a *tuple preference*, the computation of the “best” tuples is embedded into Relational Algebra (RA) in the form of a *winnow* operator.

Definition 9 (Winnow Operator [4]). *If R is a relation schema and $>_C$ a preference relation over R , then the winnow operator is written as $\omega_C(R)$, and for every instance r of R : $\omega_C(r) = \{t \in r \mid \neg \exists t' \in r. t' >_C t\}$.*

In our framework, a set preference is formulated as a tuple preference relation $>_C$ over a profile schema Γ , which in turn defines a *winnow* operator, i.e. $\omega_C(\Gamma)$. The “best” k -subsets are computed by *winnow* over the profile relation γ containing the profiles of all k -subsets of a given relation r . The algorithm and discussion of *winnow* in [4] are still valid here.

Algorithm 1 applies *winnow* on a stream of profiles of all k -subsets. The only difference is that *winnow* needs now to be aware of duplicates, since there may be different k -subsets with the same profile. In this case, *winnow* eliminates duplicates and keeps track of the relationship between a profile and its corresponding k -subsets.

This basic algorithm is only practical for small k ; for large k , the number of k -subsets, i.e. $\binom{n}{k}$, can be very large, and the exhaustive enumeration might not be acceptable. On the other hand, since the number of “best” sets can be as large as $\binom{n}{k}$

Algorithm 1 Basic Algorithm

Require: a profile schema Γ , a profile preference relation \succ_C , a relation r and a positive integer $k, k < |r|$

Ensure: the best k -subsets of r

- 1: Generate all k -subsets of relation r and for each compute its profile based on the schema Γ , obtaining the profile relation γ .
 - 2: Compute $\gamma' = \omega_C(\gamma)$ using a duplicate-aware version of any winnow evaluation algorithm, e.g. BNL [1, 4].
 - 3: Retrieve the subsets corresponding to the profiles in γ' .
-

when the set preference relation $\succ_{(\Gamma, C)}$ is empty, the worst case complexity $O(n^k)$ is unavoidable. In Algorithm 2, we use heuristics to guide the generation of k -subsets, which leads in some cases to an early stop.

The idea is that, given the set preference relation $\succ_{(\Gamma, C)}$, we are trying to find a “superpreference” relation \succ^+ such that if $t_1 \succ^+ t_2$, then any k -subset with t_1 is preferred (under $\succ_{(\Gamma, C)}$) to any k -subset with t_2 as long as these two k -subsets are otherwise identical, and vice versa.

Definition 10 (“Superpreference” Relation). *Given a relation r , a positive integer $k < |r|$ and a set preference relation $\succ_{(\Gamma, C)}$, the corresponding “superpreference” relation, denoted by \succ^+ , is such that*

$$t_1 \succ^+ t_2 \Leftrightarrow t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in (k-1)\text{-subsets}(r), \\ (t_1 \notin s' \wedge t_2 \notin s') \Rightarrow s' \cup \{t_1\} \succ_{(\Gamma, C)} s' \cup \{t_2\}].$$

When $t_1 \succ^+ t_2 \Leftrightarrow t_1 \in r \wedge t_2 \in r \wedge C^+(t_1, t_2)$, then we say that \succ^+ is *locally defined using C^+* , and the *cover* of t is defined as the set of tuples dominating t under \succ^+ , i.e. $\text{cover}(t) = \{t' \in r \mid t' \succ^+ t\}$.

Proposition 2. *Given a relation r , a positive integer $k < |r|$ and a set preference relation $\succ_{(\Gamma, C)}$, if its superpreference \succ^+ exists, then for every $s \in k\text{-subsets}(r)$,*

$$[\neg \exists s' \in k\text{-subsets}(r), s' \succ_{(\Gamma, C)} s] \Rightarrow [\forall t \in s, \text{cover}(t) \subseteq s].$$

Proof. If for some best k -subset s , there is a tuple $t' \in r$ such that $t' \in \text{cover}(t) - s$, then $(s \setminus \{t\}) \cup \{t'\} \succ_{(\Gamma, C)} s$, which is a contradiction.

Proposition 2 is a necessary condition for a best k -subset. It is used to prune the generation k -subsets in Algorithm 2 in the following fashion:

1. Any tuple t whose $|\text{cover}(t)| > k$ is discarded, as it cannot be in any best k -subsets;
2. k -subset are enumerated progressively based on the principle that if at any step, we are certain that Proposition 2 will be violated, then the corresponding k -subset will be discarded without computing its profile.

If the superpreference \succ^+ is a weak order¹, then Algorithm 3 can further simplify the k -subsets generation procedure.

¹ irreflexive, transitive, negatively transitive binary relation.

Algorithm 2 Heuristic Algorithm

Require: a profile schema Γ , a profile preference relation $>_C$, a relation r and a positive integer k , $k < |r|$, $>^+$ locally defined using C^+

Ensure: the best k -subsets of r

- 1: Do pairwise comparison between tuples in r , and determine $cover(t)$ for each $t \in r$.
 - 2: Let $r' = \{t \in r \mid |cover(t)| < k\}$.
 - 3: General all k -subsets s of r' such that $\forall t \in s, cover(t) \subseteq s$ and compute the corresponding profile relation γ' based on the schema Γ .
 - 4: Compute $\gamma'' = \omega_C(\gamma')$ using a duplicate-aware version of any winnow evaluation algorithm.
 - 5: Retrieve the subsets corresponding to the profiles in γ'' .
-

Algorithm 3 Heuristic Algorithm under Weak Order Superpreference

Require: a profile schema Γ , a profile preference relation $>_C$, a relation r and a positive integer k , $k < |r|$, $>^+$ locally defined using C^+

Ensure: the best k -subsets of r

- 1: Let $r' = \omega_{C^+}(r)$.
 - 2: If $|r'| \geq k$, generate all k -subsets of r' and the corresponding profile relation γ' based on the schema Γ , otherwise $r' = r' \cup \omega_{C^+}(r - r')$ and repeat this step.
 - 3: Compute $\gamma'' = \omega_C(\gamma')$ using a duplicate-aware version of any winnow evaluation algorithm.
 - 4: Retrieve the subsets corresponding to the profiles in γ'' .
-

It still remains to be shown how to construct a formula C^+ given the profile schema Γ and the profile preference formula C . Our preliminary study shows that for restricted classes of profile schemas and profile preference formulas, C^+ exists and can be constructed systematically.

Definition 11 (Additive k -subset Features). *Given a relation r , a positive integer $k \leq |r|$ and a k -subset feature \mathcal{A} , \mathcal{A} is additive if*

1. \mathcal{A} is well-defined for the domain of $(k-1)$ -subsets(r), and
2. for any subset $s' \in (k-1)$ -subsets(r), and any $t \in r \wedge t \notin s'$,

$$\mathcal{A}(s' \cup \{t\}) = \mathcal{A}(s') + f(t)$$

where f is a function of t only.

Notice that we use the same notation \mathcal{A} for the feature of original domain and the feature of the expanded domain, as the former should be a restriction of the latter to the original domain.

Proposition 3. *If a SQL-based k -subset feature \mathcal{A} is a subtype of (i) such that*

1. *there is no DISTINCT or subqueries, and*
2. *the aggregate is sum, count or avg with TRUE WHERE condition*

then \mathcal{A} is additive.

Proof. Let $Dom(S) = k\text{-subsets}(r) \cup (k-1)\text{-subsets}(r)$ in the definition of \mathcal{A} , \mathcal{A} is still well-defined. Under the above conditions, we can show by case study that function $f(t)$ in Definition 11 always exists. For example, if the aggregate is `sum`, $\mathcal{A}(s' \cup \{t\}) = \mathcal{A}(s') + c(t) \cdot t.A$, where A is the quantity of interest in \mathcal{A} , i.e. the attribute in `SELECT` clause, and $c(\cdot)$ is the indicator function of the `condition` in the definition of \mathcal{A} .

Theorem 1. *If the profile preference formula C can be rewritten as a DNF formula*

$$\bigvee_{i=1}^n \left(\bigwedge_{j=1}^m (\mathcal{A}_{ij}(s_1) \theta \mathcal{A}_{ij}(s_2)) \right)$$

where $\theta \in \{=, \neq, <, >, \leq, \geq\}$ and \mathcal{A}_{ij} is an additive SQL-based k -subset feature, then C^+ exists and can be constructed systematically.

Proof. Assume s' is a $(k-1)$ -subset of the relation r , we have the following rewriting

$$\begin{aligned} t_1 >^+ t t_2 &\Leftrightarrow t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in (k-1)\text{-subsets}(r), \\ &\quad (t_1 \notin s' \wedge t_2 \notin s') \Rightarrow s' \cup \{t_1\} \gg_{(\Gamma, C)} s' \cup \{t_2\}] \\ &\Leftrightarrow t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in (k-1)\text{-subsets}(r), \\ &\quad (t_1 \notin s' \wedge t_2 \notin s') \Rightarrow \bigvee_{i=1}^n \left(\bigwedge_{j=1}^m (\mathcal{A}_{ij}(s' \cup \{t_1\}) \theta \mathcal{A}_{ij}(s' \cup \{t_2\})) \right)] \end{aligned}$$

Since \mathcal{A}_{ij} is additive, we can show by case study that each $\mathcal{A}_{ij}(s' \cup \{t_1\}) \theta \mathcal{A}_{ij}(s' \cup \{t_2\})$ is equivalent to a formula of t_1, t_2 only. For example, assume `aggr` in \mathcal{A}_{ij} is `sum`, and θ is `>`, then with the abuse of the indicator function $c_{ij}(\cdot)$ as a boolean variable, we have

$$\begin{aligned} &\mathcal{A}_{ij}(s' \cup \{t_1\}) > \mathcal{A}_{ij}(s' \cup \{t_2\}) \\ \Leftrightarrow &\mathcal{A}_{ij}(s') + c_{ij}(t_1) \cdot t_1.A_{ij} > \mathcal{A}_{ij}(s') + c_{ij}(t_2) \cdot t_2.A_{ij} \\ \Leftrightarrow &(c_{ij}(t_1) \wedge c_{ij}(t_2) \wedge t_1.A_{ij} > t_2.A_{ij}) \\ &\vee (c_{ij}(t_1) \wedge \neg c_{ij}(t_2) \wedge t_1.A_{ij} > 0) \\ &\vee (\neg c_{ij}(t_1) \wedge c_{ij}(t_2) \wedge t_2.A_{ij} < 0) \end{aligned}$$

Therefore,

$$\begin{aligned} t_1 >^+ t t_2 &\Leftrightarrow t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in (k-1)\text{-subsets}(r), \\ &\quad (t_1 \notin s' \wedge t_2 \notin s') \Rightarrow \bigvee_{i=1}^n \left(\bigwedge_{j=1}^m (D_{ij}(t_1, t_2)) \right)] \end{aligned}$$

where $D_{ij}(t_1, t_2)$ is a formula defined in the same language as that of C and only with variables t_1 and t_2 . In particular, $D_{ij}(t_1, t_2)$ does not contain variable s' , in which case,

$$t_1 >^+ t t_2 \Leftrightarrow t_1 \in r \wedge t_2 \in r \wedge \bigvee_{i=1}^n \left(\bigwedge_{j=1}^m (D_{ij}(t_1, t_2)) \right)$$

By rewriting every conjunct in C , we obtain $C^+ = \bigvee_{i=1}^n (\bigwedge_{j=1}^m (D_{ij}(t_1, t_2)))$.

For the case of `avg` with `TRUE` condition, it is equivalent to the `sum` case for fixed-cardinality sets. \square

The k -subset features identified in Proposition 3 are eligible for the rewriting technique in Theorem 1. However, this rewriting does not work for features defined by `min`, or `max`, or `avg` with `non-TRUE WHERE` condition. In those cases, the feature is non-additive. Therefore, if we rewrite $\mathcal{A}(s)$ as an expression of s' and t , the term(s) containing variable s' cannot be cancelled on both sides of θ . Intuitively, it says that we cannot determine which of t_1 and t_2 is better without looking at the tuples in s' . For example, consider the case where `aggr` is `avg`, the `condition` is `non-TRUE`, and θ is $>$, the rewriting technique in Theorem 1 generates the following inequality:

$$\begin{aligned} & \mathcal{A}_{ij}(s' \cup \{t_1\}) > \mathcal{A}_{ij}(s' \cup \{t_2\}) \\ \Leftrightarrow & \frac{b_{ij}(s') \cdot \mathcal{A}_{ij}(s') + c_{ij}(t_1) \cdot t_1 \cdot \mathcal{A}_{ij}}{b_{ij}(s') + c_{ij}(t_1)} > \frac{b_{ij}(s') \cdot \mathcal{A}_{ij}(s') + c_{ij}(t_2) \cdot t_1 \cdot \mathcal{A}_{ij}}{b_{ij}(s') + c_{ij}(t_2)} \end{aligned}$$

where $b_{ij}(s') = |\{t | t \in s' \wedge c_{ij}(t) = 1\}|$. After simplifying the above inequality, we still have terms of variable s' .

In most cases, we can use domain knowledge to significantly simplify the rewriting approach described in Theorem 1. For the rewriting example in the proof, if \mathcal{A}_{ij} is `price`, which is always positive, then the rewriting is simplified to

$$c_{ij}(t_1) \wedge (t_1 \cdot \mathcal{A}_{ij} > t_2 \cdot \mathcal{A}_{ij} \vee \neg c_{ij}(t_2))$$

Example 7. In Example 1, consider the following preference

(C5) Alice wants to spend as little money as possible on sci-fi books.

(C6) Alice wants the average rating of books to be as high as possible.

and the set preference is the intersection of (C5) and (C6). Let $\Gamma = \langle \mathcal{A}_5, \mathcal{A}_6 \rangle$

$\mathcal{A}_5 \equiv \text{SELECT sum(price) FROM S WHERE genre='sci-fi'}$

$\mathcal{A}_6 \equiv \text{SELECT avg(rating) FROM S}$

and $s_1 \gg_{(\Gamma, C)} s_2$ iff $\mathcal{A}_5(s_1) < \mathcal{A}_5(s_2) \wedge \mathcal{A}_6(s_1) > \mathcal{A}_6(s_2)$. The “superpreference” formula C^+ obtained under the assumption that `price` > 0 is

$$\begin{aligned} C^+(t_1, t_2) \Leftrightarrow & t_1.\text{rating} > t_2.\text{rating} \wedge t_2.\text{genre} = \text{'sci-fi'} \\ & \wedge (t_1.\text{price} < t_2.\text{price} \vee t_1.\text{genre} \neq \text{'sci-fi'}). \end{aligned}$$

Proposition 4. For any set preference relation $\gg_{(\Gamma, C)}$, if the profile preference relation $>_C$ is a strict partial order (i.e. irreflexive and transitive), then the corresponding “superpreference” relation $>^+$ is a strict partial order as well.

Proof. If $>^+$ is empty, then it is trivially true. Otherwise, we need to show that $>^+$ is irreflexive and transitive. Given the irreflexivity of $>_C$, the irreflexivity of $>^+$ can be easily shown by contradiction. Here, we only prove the transitivity of $>^+$.

We need to show that $t_1 >^+ t t_2 \wedge t_2 >^+ t t_3 \Rightarrow t_1 >^+ t t_3$.

$$\begin{aligned} t_1 >^+ t t_2 \Leftrightarrow & t_1 \in r \wedge t_2 \in r \wedge [\forall s \in (k-1)\text{-subsets}(r), \\ & (t_1 \notin s \wedge t_2 \notin s) \Rightarrow s \cup \{t_1\} \gg_{(\Gamma, C)} s \cup \{t_2\}] \end{aligned} \quad (1)$$

$$\begin{aligned} t_2 >^+ t t_3 \Leftrightarrow & t_2 \in r \wedge t_3 \in r \wedge [\forall s \in (k-1)\text{-subsets}(r), \\ & (t_2 \notin s \wedge t_3 \notin s) \Rightarrow s \cup \{t_2\} \gg_{(\Gamma, C)} s \cup \{t_3\}] \end{aligned} \quad (2)$$

Therefore, consider any subset $s \in (k-1)\text{-subsets}(r)$, $t_1 \notin s \wedge t_3 \notin s$. We have the following two cases:

Case 1: $t_2 \notin s$

By (1), (2) and the transitivity of \succ_C , we have $s \cup \{t_1\} \succ_{(r,C)} s \cup \{t_3\}$.

Case 2: $t_2 \in s$

Let $s' = s - \{t_2\}$, by (1),

$$s' \cup \{t_3\} \cup \{t_1\} \succ_{(r,C)} s' \cup \{t_3\} \cup \{t_2\} \quad (3)$$

By (2),

$$s' \cup \{t_1\} \cup \{t_2\} \succ_{(r,C)} s' \cup \{t_1\} \cup \{t_3\} \quad (4)$$

By (3), (4) and the transitivity of \succ_C , we have $s \cup \{t_1\} \succ_{(r,C)} s \cup \{t_3\}$.

□

6 Related Work

There are many papers on preferences over tuples using either a qualitative or a quantitative approach. However, there are only a few works on preferences over sets [6–8].

[8] is conceptually the closest to our work. It addresses the problem of finding an optimal subset of items given a set of items. The language for specifying such set preferences is based on the attribute values of individual items within the set. Each *set property* is based on the number of items satisfying a certain predicate. It is either a boolean value (whether the number of items satisfying the predicate is $> k$), or an integer value (the number of items satisfying the predicate). Given a collection of set properties, a *set preference* is specified as either a TCP-net [9] or a scoring function. [8] gives heuristic search algorithms for finding an optimal subset. [8] considers subsets of any cardinality. For fixed-cardinality subsets, the language in [8] can easily be expressed in our approach: each *set property* being translated to a *k-subset feature* of the type (i) with the `count` aggregate; the boolean value and the TCP-net *set preference* being captured by a *preference formula over profiles*.

[7] focuses on fixed-cardinality set preferences. It considers two *k*-subset features: *diversity* and *depth*, and the set preference as an objective function of maximizing the linear combination of *diversity* and *depth*. Again, those cases can be expressed in our approach.

[6] considers a new class of queries called OPAC (optimization and parametric aggregation constraints) queries. Such queries aim at identifying sets of tuples that constitute the solutions of optimization problems. [6] considers subsets of any cardinality. The atomic parametric aggregation constraint is of the form $aggr(A) < parameter$ and the objective function is $\min / \max(aggr(atomic\ constraints))$. Approximation algorithms are given for query evaluation. For fixed-cardinality subsets, again, the atomic aggregation constraints can be captured by *k-subset features* and the parameters and the objective function can be captured by the *preference formula over profiles* in our framework.

7 Future Work

Our framework works for the set preferences induced by profiles and tuple preferences. We intend to study the expressive power of the framework for restricted classes of features and tuple preferences, and see if restrictions could lead to more efficient implementations. We also plan to adapt existing preference query optimization techniques [10] to set preferences, and develop new techniques. Finally, we will also investigate the query categoricity issue and the impact of integrity constraints.

Acknowledgment

Research supported by NSF grant IIS-0307434.

References

1. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE. (2001) 421–430
2. Kießling, W.: Foundations of preferences in database systems. In: VLDB. (2002) 311–322
3. Kießling, W., Köstler, G.: Preference SQL - design, implementation, experiences. In: VLDB. (2002) 990–1001
4. Chomicki, J.: Preference formulas in relational queries. *ACM Trans. Database Syst.* **28**(4) (2003) 427–466
5. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res. (JAIR)* **21** (2004) 135–191
6. Guha, S., Gunopulos, D., Koudas, N., Srivastava, D., Vlachos, M.: Efficient approximation of optimization queries under parametric aggregation constraints. In: VLDB. (2003) 778–789
7. desJardins, M., Wagstaff, K.: DD-pref: A language for expressing preferences over sets. In: AAAI. (2005) 620–626
8. Binshtok, M., Brafman, R.I., Shimony, S.E., Mani, A., Boutilier, C.: Computing optimal subsets. In: AAAI. (2007) 1231–1236
9. Brafman, R.I., Domshlak, C., Shimony, S.E.: On graphical modeling of preference and importance. *J. Artif. Intell. Res. (JAIR)* **25** (2006) 389–424
10. Chomicki, J.: Semantic optimization techniques for preference queries. *Inf. Syst.* **32**(5) (2007) 670–684