Preference Queries over Sets

Xi Zhang [1], Jan Chomicki [2]

Department of Computer Science and Engineering,
University at Buffalo, Buffalo, NY 14260, U.S.A.
[1]xizhang@acm.org
[2]chomicki@buffalo.edu

***Abstract*—We propose a “logic + SQL” framework for set preferences. Candidate *best* sets are represented using *profiles* consisting of scalar *features*. This reduces set preferences to tuple preferences over set profiles. We propose two optimization techniques: *superpreference* and *M-relation*. *Superpreference* targets dominated profiles. It reduces the input size by filtering out tuples not belonging to any best k-subset. *M-relation* targets repeated profiles. It consolidates tuples that are *exchangeable* with regard to the given set preference, and therefore avoids redundant computation of the same profile. We show the results of an experimental study that demonstrates the efficacy of the optimizations.**

I. Introduction

In recent years, *preferences* have been well studied in the database and AI literature [1], [2], [3], [4], [5]. The issues addressed in this research include, among others, preference specification, preference query languages, and preference query evaluation and optimization. However, the research on preferences, with a few exceptions [6], [7], [8], 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. For example, in university admission, the decision is made collectively on the whole incoming class. Factors such as preference for underprivileged groups are usually an important part of the decision-making process. In a board election, there might be co-existence or exclusivity relationships between candidates. A poll company usually needs to find a specific combination of candidates in order to obtain statistically representative results. Ex. 1 illustrates a book purchase scenario, where several *set preferences* are elaborated.

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

Book:

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
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 prefers to get one sci-fi book.
(C3) Ideally, she prefers that all three books be from the same vendor. If that is not possible, she prefers to deal with as few vendors as possible.

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. This preference is expressed as follows:
(C4) Alice's preference is a *prioritized composition* of (C2) and (C1).

The preference (C1) can be directly simulated by a *tuple preference* over $Book$, such that for any $t_1, t_2 \in Book$, t_1 is *preferred* to t_2 iff $t_1.price < t_2.price$. Then the top three books in $Book$ (according to this preference) constitute the *best* answer set. However, in the other cases, i.e., (C2-C4), such a simulation is not possible. □

Our goal is to develop a general framework for set preferences from the database perspective. Previous work on set preferences was predominantly done in AI research [7], [8]. To the best of our knowledge, the only work that addresses a related problem in the database context is [6]. However, the form of set preferences which can be expressed by the language in [6] is quite restricted. In this work, we emphasize the generality of preferences by providing a general formal framework for set preferences which combines first-order logic and SQL. We work with sets of fixed cardinality to focus on the composition of sets. Preferences involving fixed-cardinality sets emerge in many real applications, as outlined in Sec. V.

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 Ex. 1,

	Quantity of Interest	Desired Value or Order
(C1)	total cost	$<$
(C2)	number of sci-fi books	1
(C3)	number of distinct vendors	$<$

□

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

The main idea is to construct the profiles of 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*. Moreover, the best subsets under the set preference in the original relation correspond to the best profiles.

Our major contributions are:

- We propose a formal framework, combining SQL and first-order logic, for the specification of (second-order) set preferences via *profiling*.
- We define the notion of *superpreference relation* which helps to prune candidate *best* subsets. We show how to systematically construct first-order definitions of superpreference relations in restricted cases.
- We define the notion of *M-relation* that helps to consolidate candidate subsets. We show how to compute *optimal* M-relations using SQL.
- We report on an experimental study which demonstrates the efficacy of our optimization techniques and their synergistic character.

The rest of the paper is organized as follows. Sec. II provides the basic notions used throughout this work. We first elaborate our framework in Sec. III and Sec. IV, and then discuss the computational issues involved in computing the *best* subsets in Sec. V-A. We describe efficient optimizations to significantly reduce the computation effort in Sec. V-B (*superpreference*) and Sec. V-C (*M-relation*). In Sec. V-E, we show how to combine those two optimizations. We report on an empirical study of their performance in Sec. VI. Finally, we discuss the related work in Sec. VII and conclude in Sec. VIII.

II. Basic Notions

A binary relation $\succ$ is *irreflexive* iff $\forall x.\ x \nsucc x$. It is *transitive* iff $\forall x, y, z.\ (x \succ y \wedge y \succ z) \Rightarrow x \succ z$. It is *negatively transitive* iff $\forall x, y, z.\ (x \nsucc y \wedge y \nsucc z) \Rightarrow x \nsucc z$. It is *connected* iff $\forall x, y.\ x \succ y \vee y \succ x \vee x = y$. A *strict partial order* (SPO) is an irreflexive, transitive binary relation. A *weak order* (WO) is a negatively transitive SPO. A *total order* (TO) is a connected SPO. SPO conditions are generally considered to capture rationality of preferences. WOs arise when preferences are defined using a numeric scoring function. SPOs are more general than WOs, for example skyline, p-skyline and many other kinds of preferences [1], [2], [9] are SPOs but not WOs.

A *multiset* is a generalization of a set. In a multiset, each member may have more than one membership, in contrast to only one membership in a set. The *cardinality of a set (multiset)* s, denoted by $|s|$, is the total number of elements in s. A set (multiset) s' is a *subset (multisubset)* of the set (multiset) s iff each member of s' is also a member of s. For example, assume multisets $s_1 = \{a, b, b\}$, $s_2 = \{a, b\}$, $s_3 = \{a, a, b\}$. Then, $|s_1| = 3$, $|s_2| = 2$ and $|s_3| = 3$, respectively. Furthermore, $s_2 \subseteq s_1$, but $s_3 \nsubseteq s_1$.

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

Definition 1: **Tuple Preference [2], [4].** Given a relation schema $R = \langle A_1, \ldots, A_m \rangle$, a *tuple preference relation* $\succ$ over R is a subset of $[Dom(R)]^2$. If for a first order formula C, $C(t_1, t_2) \Leftrightarrow t_1 \succ t_2$, then the tuple preference is *defined* by the formula C. We denote the preference relation by $\succ_C$.

For a *tuple preference*, the computation of the *best* tuples is embedded into Relational Algebra (RA) in the form of the *winnow* operator. It is commonly assumed that the tuple preference is an SPO.

Definition 2: **Winnow Operator [4].** If R is a relation schema and $\succ_C$ a preference relation over R, then the winnow operator is written as $\omega_C(R)$, and for every instance r of R:

$$\omega_C(r) = \{t \in r \mid \nexists t' \in r. t' \succ_C t\}.$$

Tuples in $\omega_C(r)$ are not *dominated* by other tuples in r, and are thus the *best* tuples of r. Denote by *subsets*(r) the power set of the relation r. We capture the *quantities of interest* for subsets using *subset features*.

Definition 3: **Subset Feature.** Given a relation r, a *subset feature* $\mathcal{F}$ is a function which maps the subsets of r to exactly one of the two attribute domains. Denote by $Dom(\mathcal{F})$ the domain of $\mathcal{F}$, either $\mathcal{Q}$ or $\mathcal{D}$.

Definition 4: **Subset Profile Schema.** Given a relation r, a *subset profile schema* Γ is a schema $\langle \mathcal{F}_1, \ldots, \mathcal{F}_m \rangle$, where $\mathcal{F}_i$ is a subset feature, $i = 1, \ldots, m$.

Definition 5: **Subset Profile Relation.** Given a relation r and its subset profile schema $\Gamma = \langle \mathcal{F}_1, \ldots, \mathcal{F}_m \rangle$, the *subset profile relation* γ is defined as

$$\gamma = \{\langle \mathcal{F}_1(s), \ldots, \mathcal{F}_m(s) \rangle \mid s \in \textit{subsets}(r)\}$$

The tuple $\langle \mathcal{F}_1(s), \ldots, \mathcal{F}_m(s) \rangle$ is the *profile* of s under Γ, denoted by $\textit{profile}_\Gamma(s)$.

When the context is unambiguous, we omit *subset*, and refer to the above concepts as *feature*, *profile schema* and *profile relation*, respectively.

III. Aggregate Features

In this work, without loss of generality, we consider *single-valued* features whose values are rational numbers, as it is often the case in real applications [7]. This is achieved by defining features as aggregate values in Def. 6. Other possible single-valued features include boolean features, which will be discussed in Sec. VII.

Definition 6: **Aggregate Subset Features.** Given a relation r with a schema R, an aggregate *subset feature* $\mathcal{F}$ is defined by a parameterized SQL query of the form

```
SELECT expr FROM $S WHERE condition
```

where: (1) `$S` is a distinguished set parameter whose values can be instantiated to an arbitrary subset of r, i.e. $Dom($`$S`$) \subseteq$ *subsets*(r). (2) `expr` is of the form `aggr([DISTINCT] A)` where `aggr` $\in$ `{min,max,sum,count,avg}`, `A` is an attribute of R, or a function of constants and the above aggregates. (3) `FROM` list contains a single item `$S` or an alias for `$S`. (4) `WHERE` clause is a conjunction of atomic comparisons. As we will see, the cardinality of `$S` can be restricted.

Example 3: In Ex. 1, the quantity of interest in (C1), (C2) and (C3) is captured by the subset feature $\mathcal{F}_1, \mathcal{F}_2$ and $\mathcal{F}_3$, respectively.

$\mathcal{F}_1 \equiv$ `SELECT sum(price) FROM $S`
$\mathcal{F}_2 \equiv$ `SELECT count(title) FROM $S WHERE genre='sci-fi'`
$\mathcal{F}_3 \equiv$ `SELECT count(DISTINCT vendor) FROM $S`

where `$S` is a set parameter that can be substituted by any three-element subset of *Book*, as Alice decides to buy three books. Given any subset s of *Book*, we can evaluate the value of each feature by instantiating the set parameter `$S` in the feature definition with s. For example, assume $s = \{a_1, a_2, a_3\}$, then $\mathcal{F}_1(s)$ is the scalar result of the query `SELECT sum(price) FROM s`, which is \$15.00 + \$20.00 + \$25.00= \$60.00. Similarly, $\mathcal{F}_2(s) = 2$ due to the result of `SELECT count(title) FROM s WHERE genre='sci-fi'`, and $\mathcal{F}_3(s) = 2$ due to that of `SELECT count(DISTINCT vendor) FROM s`. Let the subset profile schema $\Gamma = \langle \mathcal{F}_1, \mathcal{F}_2 \rangle$. The subset profile relation γ (corresponding to *Book*) contains, among others, the following tuples: $(\$60, 2)$, which is the profile of the subsets $\{a_1, a_2, a_3\}$ and $\{a_2, a_3, a_5\}$; and $(\$61, 2)$, which is the profile of $\{a_3, a_7, a_8\}$. □

IV. Profile-based Set Preferences

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

Definition 7: **Set Preference.** Given a relation schema $R = \langle A_1, \ldots, A_m \rangle$, a *set preference relation* $\succ\!\!\succ$ is a finite subset of the product $[\textit{subsets}(Dom(R))]^2$.

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 based on profile preferences.*

Definition 8: **Profile-based Set Preference.** Let $\Gamma = \langle \mathcal{F}_1, \ldots, \mathcal{F}_m \rangle$ be a profile schema and $\succ_C$ a tuple preference relation, which is a subset of $[Dom(\mathcal{F}_1) \times \ldots \times Dom(\mathcal{F}_m)]^2$. For every sets s_1 and s_2,

$$s_1 \succ\!\!\succ_{(\Gamma,C)} s_2 \Leftrightarrow \textit{profile}_\Gamma(s_1) \succ_C \textit{profile}_\Gamma(s_2).$$

We say that the set s_1 is *preferred* to the set s_2 and the profile-based set preference relation is denoted by $\succ\!\!\succ_{(\Gamma,C)}$.

Proposition 4.1: If $\succ_C$ is an SPO, then for any profile schema Γ, the set preference relation $\succ\!\!\succ_{(\Gamma,C)}$ is an SPO, too.

Recall that essential components of set preferences are the *desired values or orders* of the *quantities of interest*, 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) Provide a subset profile schema by defining subset features $\mathcal{F}_1, \ldots, \mathcal{F}_m$. (2) Specify the profile preference using a tuple preference formula.

Def. 8 provides a general framework for set preferences. For the reasons we will shortly discuss in Sec. V, we restrict our interest to set preferences among subsets of *fixed cardinality*. Here, we only point out that in Ex. 1, Alice buys three books, and thus we work with subsets of fixed cardinality 3.

Example 4: Assume the profile schema $\Gamma = \langle \mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3 \rangle$ as in Ex. 3. We define the preference formula $Ci, (i = 1, 2, 3)$ over Γ, such that the individual set preference (Ci), $i = 1, 2, 3$ is based on Γ and $\succ_{Ci}$. For example,

$$s_1 \succ\!\!\succ_{(\Gamma,C1)} s_2$$
$$\Leftrightarrow \langle \mathcal{F}_1(s_1), \mathcal{F}_2(s_1), \mathcal{F}_3(s_1) \rangle \succ_{C1} \langle \mathcal{F}_1(s_2), \mathcal{F}_2(s_2), \mathcal{F}_3(s_2) \rangle$$
$$\Leftrightarrow \mathcal{F}_1(s_1) < \mathcal{F}_1(s_2).$$

□

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], [2].

Example 5: Consider (C4) in Ex. 1, i.e. the prioritized composition of (C2) and (C1) . Let the profile schema $\Gamma = \langle \mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3 \rangle$, and the preference formula $C4$ over Γ be the *prioritized composition* [4] of the preference formulas $C2$ and $C1$

$$\begin{aligned} s_1 \succ\!\!\succ_{(\Gamma,C4)} s_2 \Leftrightarrow\ & (\mathcal{F}_2(s_1) = 1 \wedge \mathcal{F}_2(s_2) \neq 1) \\ & \vee (\mathcal{F}_2(s_1) = 1 \wedge \mathcal{F}_2(s_2) = 1 \wedge \mathcal{F}_1(s_1) < \mathcal{F}_1(s_2)) \\ & \vee (\mathcal{F}_2(s_1) \neq 1 \wedge \mathcal{F}_2(s_2) \neq 1 \wedge \mathcal{F}_1(s_1) < \mathcal{F}_1(s_2)). \end{aligned}$$

We see that the preference formula $C4$ expresses precisely the set preference (C4). □

V. Computing the Best k-subsets

Recall that we make a decision to work with subsets of fixed cardinality. There are two reasons for this choice. First, fixed cardinality allows the user to focus on the composition of a *best* subset and the interactions between the tuples in it. Some set properties might have an inherent bias towards sets of certain cardinalities. For example, without limiting the cardinality in the set preference (C1), the user certainly prefers small subsets, since buying fewer books costs less. In the extreme case, the *best* subset would be the empty set, with cost \$0. In reality, this is rarely what the user intends. Likewise, many applications have an explicit or implicit requirement on cardinality. For example, a board election typically has a fixed number of seats to be filled. A university usually admits a class of a predetermined size. A poll company has limited resources for interviewing only a certain number of people.

Definition 9: k**-subset.** Given a relation r and a positive integer k, $k \leqslant |r|$, 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-subsets*(r) the set of all k-subsets of r.

Definition 10: k**-subset Profile Relation.** Given a relation r and its subset profile schema $\Gamma = \langle \mathcal{F}_1, \ldots, \mathcal{F}_m \rangle$, the *$k$-subset profile relation* γ_k is defined as

$$\gamma_k = \{\langle \mathcal{F}_1(s), \ldots, \mathcal{F}_m(s) \rangle \mid s \in k\textit{-subsets}(r)\}$$

We omit the subscript k in γ_k when the context is unambiguous.

A. Basic Algorithm

For a *tuple preference*, the computation of the *best* tuples is embedded into Relational Algebra (RA) in the form of a *winnow* operator (c.f., Def. 2). In addition to the universal *Nested Loops (NL)* algorithm, several other efficient evaluation algorithms for *winnow* have been proposed when the preference relation is an SPO, among others, *Block Nested Loops (BNL)* [1] and *Sort-Filter-Skyline (SFS)* [10]. In our framework, a set preference relation is formulated as a tuple preference relation $\succ_C$ over a profile schema Γ. Then, a *winnow* operator is defined over Γ, i.e. $\omega_C(\Gamma)$. The *best* k-subsets are computed by *winnowing* over the profile relation γ containing the profiles of all k-subsets of a given relation r. Alg. 1 applies *winnow* on a stream of profiles of all k-subsets.

Algorithm 1 (NAIVE) Basic Algorithm

```
Require: a profile schema $\Gamma$, an SPO profile preference relation $\succ_C$, a relation $r$ and a positive integer $k, k < |r|$
Ensure: the best $k$-subsets of $r$ under the set pref. $\ggcurly_{(\Gamma,C)}$
1: Generate all $k$-subsets of relation $r$ and compute the profiles of each $k$-subset based on the schema $\Gamma$, obtaining the profile relation $\gamma$.
2: Compute $\gamma' = \omega_C(\gamma)$ using any winnow evaluation algorithm, e.g. BNL [1].
3: Retrieve the subsets corresponding to the profiles in $\gamma'$.
```

For the generation of candidate k-subsets in Line 1 of Alg. 1, any sound and complete k-subset generator suffices. We arbitrarily choose a lexicographical k-subset generator [11] which produces k-subsets in the lexicographical order of the tuple indices.

Alg. 1 is suitable for a small k, while for a large k, the number of k-subsets $\binom{n}{k}$ can be intimidating, and exhaustive enumeration might not be acceptable. On the other hand, since the number of *best* sets can be as large as $\binom{n}{k}$ when the set preference relation $\ggcurly_{(\Gamma,C)}$ is empty, the worst case complexity $\Omega(n^k)$ is unavoidable. In the following sections, we identify redundant k-subsets generated by the basic algorithm, i.e., k-subsets whose profiles will be dominated by other profiles or k-subsets whose profiles are repeated. We propose two optimization techniques: *superpreference* and *M-relation*. Roughly speaking, *superpreference* targets the dominated profiles. It filters out tuples that do not contribute to any best k-subset. *M-relation* targets repeated profiles. It groups together tuples that are *exchangeable* with regard to the given set preference, and therefore avoids redundant computation of the same profile. Both techniques tend to reduce the number of candidate k-subsets and therefore speed up the computation of the best subsets.

B. Superpreference

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

Definition 11: **Superpreference Relation.** Given a relation r, a positive integer $k \leqslant |r|$ and a set preference relation $\ggcurly_{(\Gamma,C)}$, the corresponding superpreference relation, denoted by $\succ^+$, is such that

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

The *cover* of t is the set of tuples dominating t under $\succ^+$, i.e. $cover(t) = \{t' \in r \mid t' \succ^+ t\}$. Whenever $t_1 \succ^+ t_2 \Leftrightarrow t_1 \in r \wedge t_2 \in r \wedge C^+(t_1, t_2)$ and C^+ is a first-order formula, we say that $\succ^+$ *corresponds to* C^+.

Proposition 5.1: Given a relation r, a positive integer $k < |r|$ and a set preference relation $\ggcurly_{(\Gamma,C)}$, for every $s \in \textit{k-subsets}(r)$,

$$[\nexists s' \in \textit{k-subsets}(r), s' \ggcurly_{(\Gamma,C)} s] \Rightarrow [\forall t \in s, cover(t) \subseteq s]. \quad (1)$$

Eqn. (1) states that if a k-subset s is not dominated by any other k-subset, then s should contain the cover of each member. It is a necessary condition for a best k-subset. Prop. 5.1 enables two optimizations in Alg. 2: (1) Every tuple t whose $|cover(t)| \geqslant k$ is discarded, as it cannot belong to any best k-subset (Line 2); (2) During the candidate k-subset generation in Line 3, we skip over those candidate k-subsets *not* leading to a best k-subset according to Prop. 5.1. To be more specific, in Line 3 of Alg. 2, we use a modified version of the standard lexicographical k-subset generator, which applies a filter when enumerating k-subsets. It returns the first successive k-subset satisfying Eqn. (1) if any.

Algorithm 2 (SUPER) Superpreference Algorithm

```
Require: a profile schema $\Gamma$, an SPO profile preference relation $\succ_C$, a relation $r$, a positive integer $k, k < |r|$ and $\succ^+$ corresponding to $C^+$
Ensure: the best $k$-subsets of $r$ under the set pref. $\ggcurly_{(\Gamma,C)}$
1: Do pairwise comparisons between tuples in $r$ based on $\succ^+$, and determine $cover(t)$ for each $t \in r$.
2: Let $r' = \{t \in r \mid |cover(t)| < k\}$.
3: Use a modified version of the standard $k$-subset generator to obtain all $k$-subsets $s$ of $r'$ such that $\forall t \in s, cover(t) \subseteq s$ and compute the corresponding profile relation $\gamma'$ based on the schema $\Gamma$.
4: Compute $\gamma'' = \omega_C(\gamma')$ using any winnow evaluation algorithm.
5: Retrieve the subsets corresponding to the profiles in $\gamma''$.
```

Algorithm 3 Superpreference Algorithm under Weak Order Superpreference

```
Require: a profile schema $\Gamma$, a WO profile preference relation $\succ_C$, a relation $r$, a positive integer $k, k < |r|$ and $\succ^+$ corresponding to $C^+$
Ensure: the best $k$-subsets of $r$ under the set pref. $\ggcurly_{(\Gamma,C)}$
1: Let $r' = \omega_{C^+}(r)$.
2: If $|r'| \geqslant k$, generate all $k$-subsets of $r'$ and the corresponding profile relation $\gamma'$ based on the schema $\Gamma$, otherwise $r' = r' \cup \omega_{C^+}(r - r')$ and repeat this step.
3: Compute $\gamma'' = \omega_C(\gamma')$ using any winnow evaluation algorithm.
4: Retrieve the subsets corresponding to the profiles in $\gamma''$.
```

If the superpreference $\succ^+$ is a WO, Alg. 3 can further reduce the input (r') to the lexicographical k-subset generator, which leads to fewer candidate k-subsets. In order to illustrate the importance of the WO requirement in Alg. 3, let $r_1 = \omega_{C^+}(r)$, $r_2 = \omega_{C^+}(r - r_1)$, $r_3 = \omega_{C^+}(r - r_1 - r_2)$ ..., until all tuples in r are exhausted. Let $\cup r_i = r_1 \cup \ldots \cup r_i$, then $r_i = \omega_{C^+}(r - \cup r_{i-1})$. If the superpreference $\succ^+$ is a WO, then by the definition of WO every tuple in $\cup r_i$ is *superpreferred* to every tuple in $r - \cup r_i$. In other words, every tuple in $\cup r_i$ belongs to the cover of every tuple in $r - \cup r_i$. If $\cup r_i$ contains at least k tuples, we know that the cardinality of the cover of every tuple in $r - \cup r_i$ is at least k, and thus the tuple can be discarded. A general SPO does not guarantee such a relationship and thus we have to keep track of the covers of individual tuples (Alg. 2).

It still remains to show how to construct the formula C^+ given a profile schema Γ and a profile preference formula C. We show below that for restricted classes of profile schemas and profile preference formulas, C^+ can be constructed systematically. Def. 12 introduces an important class of features, namely *additive* features. As we will present shortly in this section as well as in Sec. V-C, additivity of features enables various optimization techniques.

Definition 12: **Additive Subset Features.** Given a relation r and a subset feature $\mathcal{F}$, $\mathcal{F}$ is *additive* iff (1) for every tuple $t \in r$, $\mathcal{F}(\{t\}) = f(t)$, and (2) for every $s \in$ *subsets*(r) and every $t \in r - s$, $\mathcal{F}(s \cup \{t\}) = \mathcal{F}(s) + f(t)$, where f is a function of t only, called the *base* of $\mathcal{F}$.

Proposition 5.2: If an aggregate feature $\mathcal{F}$ is of the form `SELECT expr FROM $S WHERE simple-condition` where: (1) `expr` is of the form `aggr(A)`, where `aggr` $\in$ {`sum`, `count`} and `A` is an attribute of the schema of r, or a linear combination of constants and the above aggregates; (2) `simple-condition` does not contain subqueries, then $\mathcal{F}$ is additive.

Theorem 1: If $k < |r| - 1$ is the chosen cardinality and a profile-based set preference is defined as a DNF formula of the following form

$$s_1 \ggcurly_{(\Gamma,C)} s_2 \Leftrightarrow \bigvee_{i=1}^{n}\left(\bigwedge_{j=1}^{m_i}(\mathcal{F}_{ij}(s_1)\ \theta \mathcal{F}_{ij}(s_2))\right) \qquad (2)$$

where $\theta \in \{=, \neq, <, >, \leqslant, \geqslant\}$ and $\mathcal{F}_{ij}$ is an additive aggregate subset feature, then C^+ is a first-order formula independent of k.

Proof: For every i and j, we have the rewriting

$$\begin{aligned} t_1 \succ^+ t_2 \Leftrightarrow\ & t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)}\text{-subsets}(r \backslash \{t_1, t_2\}), \\ & s' \cup \{t_1\} \ggcurly_{(\Gamma,C)} s' \cup \{t_2\}] \\ \Leftrightarrow\ & t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)}\text{-subsets}(r \backslash \{t_1, t_2\}), \\ & \textstyle\bigvee_{i=1}^{n}(\bigwedge_{j=1}^{m_i}(\mathcal{F}_{ij}(s' \cup \{t_1\})\ \theta \mathcal{F}_{ij}(s' \cup \{t_2\})))] \end{aligned} \cdot$$

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

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

where $c_{ij}(t)$ indicates whether the tuple t satisfies the `WHERE` condition of the feature $\mathcal{F}_{ij}$. Therefore,

$$\begin{aligned} t_1 \succ^+ t_2 \Leftrightarrow\ & t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)}\text{-subsets}(r \backslash \{t_1, t_2\}), \\ & \textstyle\bigvee_{i=1}^{n}(\bigwedge_{j=1}^{m_i}(D_{ij}(t_1, t_2)))] \end{aligned}$$

where $D_{ij}(t_1, t_2)$ is a formula with only variables t_1 and t_2. In particular, $D_{ij}(t_1, t_2)$ does not contain the set variable s'. By rewriting every conjunct in C, we have

$$t_1 \succ^+ t_2 \Leftrightarrow t_1 \in r \wedge t_2 \in r \wedge \textstyle\bigvee_{i=1}^{n}(\bigwedge_{j=1}^{m_i}(D_{ij}(t_1, t_2))),$$

and thus $C^+(t_1, t_2) = \bigvee_{i=1}^{n}(\bigwedge_{j=1}^{m_i}(D_{ij}(t_1, t_2)))$. ■

The additive subset features identified in Prop. 5.2 are eligible for the rewriting technique in Thm. 1. However, this rewriting does not work for non-additive features defined by `min`, or `max`, or `avg` with non-`TRUE WHERE` condition. In those cases, if we rewrite $\mathcal{F}(s)$ as an expression of s' and t, the term(s) containing variable s' cannot be canceled on both sides of θ. Intuitively, it states that we cannot determine which of t_1 and t_2 is superpreferred 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 Thm. 1 generates the following inequality:

$$\begin{aligned} & \mathcal{F}_{ij}(s' \cup \{t_1\})\ > \mathcal{F}_{ij}(s' \cup \{t_2\}) \Leftrightarrow \\ & \tfrac{(b_{ij}(s') \cdot \mathcal{F}_{ij}(s') + c_{ij}(t_1) \cdot t_1.A_{ij})}{(b_{ij}(s') + c_{ij}(t_1))} > \tfrac{(b_{ij}(s') \cdot \mathcal{F}_{ij}(s') + c_{ij}(t_2) \cdot t_2.A_{ij})}{(b_{ij}(s') + c_{ij}(t_2))}, \end{aligned}$$

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

When Thm. 1 is applicable, we can often use domain knowledge to significantly simplify the rewriting in Thm. 1. For the rewriting example in the proof of Thm. 1, if A_{ij} is *price*, which is always positive, then the rewriting result can be simplified to $c_{ij}(t_1) \wedge (t_1.A_{ij} > t_2.A_{ij} \vee \neg c_{ij}(t_2))$.

Example 6: In Ex. 1, consider the following preference: (C5) Alice wants to spend as little money as possible on sci-fi books. (C6) Alice wants the total rating of books to be as high as possible. The set preference is the intersection of (C5) and (C6). Therefore, we have $\Gamma = \langle \mathcal{F}_5, \mathcal{F}_6 \rangle$,

$\mathcal{F}_5 \equiv$ `SELECT sum(price) FROM $S`
`WHERE genre='sci-fi'`
$\mathcal{F}_6 \equiv$ `SELECT sum(rating) FROM $S`

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

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

Notice that the superpreference computed via Thm. 1 is by itself order-preserving. That is, if the profile preference $\succ_C$ is an SPO (WO, TO resp.), the superpreference $\succ^+$ computed via Thm. 1 is also an SPO (WO, TO resp.). However, the integration of any domain knowledge might change the order properties of $\succ^+$. For example, in Ex. 6, the profile preference $\succ_C$ is a WO while the superpreference $\succ^+$ is not. It is due to the fact that we integrate the domain knowledge *price* > 0 in $\succ^+$. Also notice that two important classes of preferences, skyline [1] and p-skyline [9], can be expressed with DNF formulas of the form used in Eqn. (2).

C. M-relation

Superpreference is a pruning technique used to filter out tuples not contributing to the best k-subsets. It reduces the size of the original relation, which leads to fewer candidate k-subsets. In other words, it prunes the *inferior* k-subsets even before the k-subset generation. In addition, we often observe another type of *redundant* candidate k-subsets during generation, as shown in Ex. 7. For example, if we have already generated the 3-subset $\{a_1, a_2, a_7\}$, we do not need to generate $\{a_1, a_2, a_9\}$ or $\{a_1, a_2, a_{10}\}$ as neither leads to a new profile given the set preference. It is therefore more efficient to consolidate a_7, a_9, a_{10} into a *meta-tuple*, i.e., an *M-tuple*, $m_{7,9,10}$, and consider only the M-tuple in the generation of candidate k-subsets.

Example 7: Add two more tuples to `Book` in Ex. 1

Book2:

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
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
a_9	romance	4.0	\$20.00	Amazon
a_{10}	history	4.0	\$19.00	Amazon

and use the same set preference $\succ_{(\Gamma,C)}$ as that in Ex. 6. The tuple a_7 and a_9 are *exchangeable* with regard to the set preference, because for every 2-subset s of `Book2`$\backslash\{a_7, a_9\}$, extending s with a_7 or a_9 leads to the same profile in the profile relation, i.e., $\textit{profile}_\Gamma(s \cup \{a_7\}) = \textit{profile}_\Gamma(s \cup \{a_9\})$. By the same argument, a_7, a_9 and a_{10} are mutually *exchangeable*. □

Based on the above idea, we can define an *exchangeability relation* among tuples. First, we classify the features in a profile schema into three categories, based on their additivity and their appearance in the definition of the set preference. Without loss of generality, the profile schema in a set preference $\succ_{(\Gamma,C)}$ is

$$\Gamma = \{\overbrace{\underbrace{\mathcal{F}_1, \ldots, \mathcal{F}_{m_a}}_{\text{additive}}, \mathcal{F}_{m_a+1}, \ldots, \mathcal{F}_{m_b}}^{\text{relevant}}, \mathcal{F}_{m_b+1} \ldots, \mathcal{F}_m\},$$

where: (1) The features $\mathcal{F}_1, \ldots, \mathcal{F}_{m_b}$ are the *relevant* features appearing in the profile preference formula C, denoted by Γ_Δ, while the features $\mathcal{F}_{m_b+1}, \ldots, \mathcal{F}_m$ do not appear in C; (2) The features $\mathcal{F}_1, \ldots, \mathcal{F}_{m_a}$ are additive (c.f., Def. 12), while $\mathcal{F}_{m_a+1}, \ldots, \mathcal{F}_{m_b}$ are not.

Definition 13: **Exchangeability Relation.** Given a relation r, a positive integer $k < |r|$ and a set preference relation $\succ_{(\Gamma,C)}$, an equivalence relation $\approx_{(\Gamma,C)}$ is an *exchangeability relation* over r if

$$\begin{aligned} & t_1 \approx_{(\Gamma,C)} t_2 \Rightarrow \\ & t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)}\text{-subsets}(r\backslash\{t_1, t_2\}), \quad (3) \\ & \textit{profile}_{\Gamma_\Delta}(s' \cup \{t_1\}) = \textit{profile}_{\Gamma_\Delta}(s' \cup \{t_2\})] \end{aligned}$$

in which case, we say that tuple t_1 and t_2 are *exchangeable*.

Given a set preference, there can be more than one exchangeability relation according to Def. 13. For example, the *equality relation* where each equivalence class contains exactly one tuple is a *trivial* exchangeability relation. In fact, any equivalence relation *contained* in an exchangeability relation is another exchangeability relation. However, Prop. 5.3 guarantees a unique *optimal* exchangeability relation.

Proposition 5.3: Given a relation r, a positive integer $k < |r|$ and a set preference relation $\succ_{(\Gamma,C)}$, the *optimal* exchangeability relation, which contains every possible exchangeability relation over r under k and $\succ_{(\Gamma,C)}$, exists and is unique.

Proof: (Sketch) Denote by P the set of all possible equivalence relations on r. It is well known that P, when ordered by *set containment* ($\subseteq$), form a *complete lattice* $(P, \subseteq)$, where any subset of P has a *supremum*. Since all exchangeability relations form a subset of P, they have a unique supremum. We can further show that the supremum of any two exchangeability relations is still an exchangeability relation. The supremum of all exchangeability relation is therefore the optimal one. ■

It is easy to see that the *optimal* exchangeability relation $\approx_{(\Gamma,C)}$ over the relation r is

$$\begin{aligned} & t_1 \approx_{(\Gamma,C)} t_2 \Leftrightarrow \\ & t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)}\text{-subsets}(r\backslash\{t_1, t_2\}), \quad (4) \\ & \textit{profile}_{\Gamma_\Delta}(s' \cup \{t_1\}) = \textit{profile}_{\Gamma_\Delta}(s' \cup \{t_2\})] \end{aligned}$$

Eqn. (4) differs from Eqn. (3) in that it requires a sufficient condition as well.

Example 8: Assume the same set preference $\succ_{(\Gamma,C)}$ in `Book2` of Ex. 6. By Def. 13, one exchangeability relation and its partition are $\approx_0 = \{(a_i, a_i)|i = 1, \ldots, 10\}$, $P_0 = \{\{a_i\}|i = 1, \ldots, 10\}$. Another exchangeability relation and its partition are $\approx_1 = \{(a_i, a_i)|i = 1, \ldots, 10\} \cup \{(a_7, a_9), (a_9, a_7), (a_7, a_{10}), (a_{10}, a_7), (a_9, a_{10}), (a_{10}, a_9)\}$, $P_1 = \{\{a_i\}|i = 1, \ldots, 6, 8\} \cup \{\{a_7, a_9, a_{10}\}\}$. The exchangeability relation $\approx_0$ is the equality relation. Since P_0 is a refinement of P_1, i.e., $P_0 \subset P_1$, the exchangeability relation $\approx_0$ is not optimal. It is easy to verify that $\approx_1$ is optimal since the merge of any equivalence classes in partition P_1 does not lead to an exchangeability relation over *Book2*. □

We introduce now a profile consolidation optimization using *M-relations*. Given an exchangeability relation $\approx_{(\Gamma,C)}$, an M-relation contains *M-tuples*, and there is a one-to-one mapping between its *M-tuples* and the equivalence classes of $\approx_{(\Gamma,C)}$. There are various ways to define an M-relation corresponding to an exchangeability relation. In Def. 14, we show how to define in SQL an M-relation corresponding, under certain conditions, to the *optimal* exchangeability relation.

In Def. 14, the M-relation is defined using a query on the distinct values of attributes corresponding to the relevant features $\mathcal{F}_1, \ldots, \mathcal{F}_{m_b}$. Moreover, for the additive relevant features $\mathcal{F}_1, \ldots, \mathcal{F}_{m_a}$, tuples are grouped by their contribution to the values of the additive features over sets: f_i is the *base* of the additive feature $\mathcal{F}_i$. The M-relation also keeps track of the number of tuples consolidated into an M-tuple, using the special attribute A_{cnt}.

Definition 14: ***M-relation* using SQL.** Given a relation r with a schema R, a non-negative integer k and a set preference $\succ_{(\Gamma,C)}$, define the M-relation schema $O = \langle A_1, \ldots, A_{m_a}, A_{m_a+1}, \ldots, A_{m_c}, A_{cnt}\rangle$, and the M-relation o by the following SQL query:

```
SELECT $f_1$(R) AS $A_1$, ..., $f_{m_a}$(R) AS $A_{m_a}$,
        attrs($\mathcal{F}_{m_a+1}, \ldots, \mathcal{F}_{m_b}$), count(*) AS $A_{cnt}$
FROM R GROUP BY $A_1$, ..., $A_{m_a}$, attrs($\mathcal{F}_{m_a+1}, \ldots, \mathcal{F}_{m_b}$)
```

where: (1) `R` is a tuple range variable. (2) f_i(`R`), $i = 1, \ldots, m_a$ is the base of the additive feature $\mathcal{F}_i$. (3) `attrs`$(\mathcal{F}_{m_a+1}, \ldots, \mathcal{F}_{m_b})$ is the set of attributes mentioned in the definitions of the features $\mathcal{F}_{m_a+1}, \ldots, \mathcal{F}_{m_b}$, say $\{A_{m_a+1}, \ldots, A_{m_c}\}$. (4) A_{cnt} is a special attribute in the

M-relation schema tracking the number of tuples consolidated into a single M-tuple in the M-relation.

Example 9: In Ex. 7, the M-relation schema is $\langle A_5, A_6, A_{cnt}\rangle$, which is determined by the set preference. In order to illustrate the case where there are *non-additive relevant* features, and more importantly the concept of *projection to relevant features*, we assume the non-additive feature $\mathcal{F}_3$ is also a relevant feature in the set preference. Thus, the M-relation schema becomes $\langle A_5, A_6, A_3, A_{cnt}\rangle$, and the M-relation is generated via the following SQL query:

```
SELECT
  CASE WHEN r.genre='sci-fi' THEN r.price
  ELSE 0 END AS $A_5$, r.rating AS $A_6$,
  r.vendor AS $A_3$, count(*) AS $A_{cnt}$
FROM Book2 r GROUP BY $A_5, A_6, A_3$
```

In the following M-relation o', the subscripts of each M-tuple are the indices of the tuples consolidated into it.

	A_5	A_6	A_3	A_{cnt}
m_1	\$15.00	5.0	Amazon	1
m_2	\$0.00	4.8	B&N	1
m_3	\$25.00	4.5	Amazon	1
m_4	\$0.00	4.4	B&N	1
m_5	\$15.00	4.3	Amazon	1
m_6	\$2.00	4.2	B&N	1
$m_{7,9,10}$	\$0.00	4.0	Amazon	3
m_8	\$18.00	3.5	Amazon	1

Notice that the actual M-relation o in Ex. 7 is the above o' relation with the columns A_5, A_6, A_{cnt} only. □

In Ex. 9, f_5, the base of the additive feature $\mathcal{F}_5$, is expressed via a CASE statement. In the CASE statement, the ELSE statement gives a *default* value to the new feature if the WHEN condition is not satisfied. The default value is 0 for A_5 in Ex. 9. In general, it can be any constant. M-relations identify the *exchangeable* tuples in Ex. 7.

Theorem 2: For a DNF profile preference formula C in the form of Eqn. (2), a relation r and $k < |r|$, if all relevant features are additive, then the exchangeability relation corresponding to the M-relation in Def. 14 is optimal over r.

Example 10: In Ex. 9, the partition generated by the M-relation defined in Def. 14 is $\{\{a_1\}, \ldots, \{a_6\}, \{a_7, a_9, a_{10}\}, \{a_8\}\}$, which is the exact partition corresponding to the optimal exchangeability relation defined by Eqn. (4). □

D. Computing Profiles via M-relations

Our goal is to *directly* compute the profiles from an M-relation. The subtlety lies in that if we compute profiles only from the k-subsets of the M-relation, we might miss some of the profiles in the original relation. Say $k = 2$, and an M-tuple $m_{1,2}$ corresponds to tuples t_1 and t_2 in the original relation r. We have a profile computed from the 2-subset $\{t_1, t_2\}$ in the profile relation γ. However, we cannot compute this profile from a k-subset of the M-relation, as any k-subset contains at most one M-tuple $m_{1,2}$ and $\{m_{1,2}, m_{1,2}\}$ is not a 2-subset of the M-relation. Hence, in order to compute the exact profiles of the original relation, we need to compute profiles from the *k-multisubsets* of an M-relation.

Definition 15: k-**multisubset.** Given an M-relation o and a positive integer k, $k < \sum_{m_i \in o} m_i.A_{cnt}$, a *k-multisubset* s of o is a multiset of o with cardinality k, and the number of occurrences of each M-tuple does not exceed its A_{cnt} value. Denote by *k-multisubsets*(o) the set of all k-multisubsets of o.

Note that Def. 15 is the multisubset version of Def. 9. We can easily extend Def. 3-12 to their multisubset counterparts (details omitted).

Theorem 3: Assume o is an M-relation corresponding to a relation r. For each $s \in$ *k-subsets*(r), there is a $s' \in$ *k-multisubsets*(o), such that $\textit{profile}_{\Gamma_\Delta}(s) = \textit{profile}_{\Gamma_\Delta}(s')$, and vice versa.

Thm. 3 guarantees that the projection of the original profile relation to relevant features computed from the M-relation is an onto function. Therefore, we can compute the projection by evaluating those features of k-multisubsets. Alg. 4 uses M-relations to compute the best k-subsets. Alg. 4 differs from Alg. 1 in the use of an M-relation and a k-multisubset generator. The number of occurrences of an M-tuple in a candidate k-multisubset is bounded from above by the number of tuples consolidated into it. It is therefore crucial to the k-multisubset generation that the M-relation records this number for each M-tuple.

Algorithm 4 (MREL) M-relation Algorithm

```
Require: a profile schema $\Gamma$, an SPO profile preference
    relation $>_C$, a relation $r$ and a positive integer $k, k < |r|$
Ensure: the best $k$-subsets of $r$ under the set pref. $\gg_{(\Gamma,C)}$
1: Compute the M-relation $o$ of $r$ based on $\Gamma$ and $>_C$.
2: Generate all $k$-multisubsets of $o$ and for each compute the
   profile features in $\Gamma$ relevant to $C$, i.e., in $\Gamma_\Delta$, to obtain
   $\gamma'$, i.e. the projection of the profile relation $\gamma$ onto $\Gamma_\Delta$.
3: Compute $\gamma'' = \omega_C(\gamma')$ using any winnow evaluation
   algorithm.
4: Retrieve the subsets whose profile projections correspond
   to the elements of $\gamma''$.
```

The profile of a k-multisubset in an M-relation is the projection of the profile of the corresponding k-subset(s) in the original relation to relevant features. Notice that the M-relation contains all the information needed to compute the projection of this profile. For an additive relevant feature, we keep each M-tuple's contribution to the value of this feature. For a non-additive relevant feature, we keep the values of every attribute involved in the evaluation of this feature. Therefore, in order to compute the projection of this profile, we simply add up each M-tuple's contribution for additive relevant features, and compute the values of non-additive relevant features as we would do for a k-subset.

Example 11: Continuing Ex. 9, the following table gives a few examples of candidate k-subsets ($k = 3$) in Book2 and their corresponding k-multisubsets in the M-relation o, together with their profiles in the profile relation γ.

k-subsets of Book2	k-multisubsets of o	profile in γ
$\{a_1, a_2, a_7\}$, $\{a_1, a_2, a_9\}$, $\{a_1, a_2, a_{10}\}$	$\{m_1, m_2, m_{7,9,10}\}$	(\$15.00, 13.8)
$\{a_1, a_7, a_9\}$, $\{a_1, a_7, a_{10}\}$, $\{a_1, a_9, a_{10}\}$	$\{m_1, m_{7,9,10}, m_{7,9,10}\}$	(\$15.00, 13.0)

In fact, a k-subset generator of Book2 will enumerate all $\binom{10}{3} = 120$ candidate k-subsets, while a k-multisubset generator of the M-relation will only enumerate $\binom{7}{3}+\binom{7}{2}+\binom{7}{1}+\binom{7}{0} = 64$ k-multisubsets, where $\binom{7}{i}$ stands for the case where the multisubset contains $3-i$ $m_{7,9,10}$, and i non-$m_{7,9,10}$ M-tuple(s). Since the feature $\mathcal{F}_5$ and $\mathcal{F}_6$ are both additive, the evaluation of $\mathcal{F}_5$ changes to SELECT sum(A_5) FROM $S and the evaluation of the feature $\mathcal{F}_6$ changes to SELECT sum(A_6) FROM $S. We can compute a profile from a k-multisubset by computing its features. For example, $s = \{m_1, m_2, m_{7,9,10}\}$, then $\mathcal{F}_5(s) = m_1.A_5 + m_2.A_5 + m_{7,9,10}.A_5 = \$15.00 + \$0.00 + \$0.00 = \$15.00$, and $\mathcal{F}_6(s) = m_1.A_6 + m_2.A_6 + m_{7,9,10}.A_6 = 5.0+4.8+4.0 = 13.8$. The profile of the multiset s is (\$15.00, 13.8). The profile relation Γ contains 64 distinct profiles. □

In Ex. 11, the k-multisubset generator eliminates all redundancy in the subset generation: each candidate k-multisubset returned by the generator leads to a distinct new profile. Though it is not always the case in general, a k-multisubset generator still reduces the number of subsets generated significantly in most cases. The benefit of M-relations comes in three folds: (1) Selectivity of attribute values. The M-relation is defined by a GROUP BY SQL query in Def. 14, which suggests that low selectivity of attribute values will lead to fewer M-tuples. (2) Selectivity of WHERE conditions in feature definitions. Since every feature definition is also a "mini-SQL" query, it is likely that a tuple not satisfying the WHERE condition of the feature definition will contribute a *default* value in the feature evaluation. In fact, such default values exist for all additive features. Recall that the M-relation definition query in Ex. 9 has default value 0 for attribute A_5. The default values fully exploit the selectivity of the WHERE condition. A highly selective WHERE condition will lead to more tuples with the default value, which are more likely to be consolidated into one tuple in the M-relation. This in turn suggests a smaller M-relation. (3) Non-relevant attributes. The projection to attributes relevant to the set preference reduces the redundancy in k-subset generation.

E. Hybrid Approaches

Superpreference and *M-relation* target two different aspects of pruning: *superpreference* filters out tuples which cannot be a part of any best k-subset, while *M-relation* consolidates exchangeable tuples and processes k-subsets in groups. It is natural to combine them to achieve synergistic pruning. We propose here two possible combinations of those two optimizations: SM and MS. Without loss of generality, assume we are given a relation r, a positive integer $k < |r|$ and a set preference $\succcurlyeq_{(\Gamma,C)}$.

(SM) Superpreference followed by M-relation. This algorithm is rather straightforward. We use the technique in *superpreference* to filter out tuples with cover size at least k, and then apply *M-relation* to the reduced input to compute the final results. The exact algorithm is illustrated in Alg. 5.

(MS) M-relation followed by Superpreference. In this algorithm (Alg. 6), we apply the superpreference technique to the M-relation. We compute the M-relation as before. In order to apply the superpreference optimization to an M-relation, we need to generalize the notions of *superpreference* and *cover* to M-relations. Recall that, in Def. 11, a tuple t_1 is superpreferred to a tuple t_2 iff it is more beneficial to extend every (k-1)-subset with t_1 instead of t_2 to get a k-subset. The superpreference relation $\succ_o^+$ in an M-relation o is based on the same idea: an M-tuple m_1 is superpreferred to an M-tuple m_2 iff it is more beneficial to extend every (k-1)-multisubset with m_1 instead of m_2 to get a k-multisubset. Similarly, the *cover* of an M-tuple m is the multiset of M-tuples, where each M-tuple m' dominates m under $\succ_o^+$, and has exactly $m'.A_{cnt}$ occurrences in the *cover*, i.e., $cover(m) = \{m', m' \text{ repeats } m'.A_{cnt} \text{ times} \mid m' \in o, m' \succ_o^+ m\}$.

Algorithm 5 (SM) Super-MRel Hybrid Algorithm

```
Require: a profile schema Γ, an SPO profile preference
    relation $\succ_C$, a relation $r$, a positive integer $k, k < |r|$
    and $\succ^+$ corresponding to $C^+$
Ensure: the best $k$-subsets of $r$ under the set pref. $\succcurlyeq_{(\Gamma,C)}$
 1: Do pairwise comparison between tuples in $r$ based on $\succ^+$,
    and determine $cover(t)$ for each $t \in r$.
 2: Let $r' = \{t \in r \mid |cover(t)| < k\}$.
 3: Return the result of the M-relation Algorithm (Alg. 4) with
    the input relation $r'$ and the profile preference relation $\succ_C$
    restricted to $r'$ instead.
```

Algorithm 6 (MS) MRel-Super Hybrid Algorithm

```
Require: a profile schema Γ, an SPO profile preference
    relation $\succ_C$, a relation $r$, a positive integer $k, k < |r|$
    and $\succ_o^+$ corresponding to $C^+$
Ensure: the best $k$-subsets of $r$ under the set pref. $\succcurlyeq_{(\Gamma,C)}$
 1: Compute the M-relation $o$ of the relation $r$.
 2: Compute $o' = \{m \in o \mid |cover(m)| < k\}$.
 3: Retrieve the relation $r' \subseteq r$ contributing to the M-relation
    $o'$.
 4: Return the result of the M-relation Algorithm (Alg. 4) with
    the input relation $r'$ and the profile preference relation $\succ_C$
    restricted to $r'$ instead.
```

VI. Experiments

We report here an experimental study of the *best-subset* generation algorithms proposed. We implemented all algorithms in C++ and ran experiments on a machine with Intel Core2 1.66GHz CPU running Cygwin on Windows XP with 2GB memory. We used a real dataset containing the information of 8000 book quotes from Amazon. The data schema is $\langle title, genre, rating, price, \ vendor\rangle$. We implemented five algorithms: NAIVE, SUPER, MREL, SM, MS. NAIVE is the basic algorithm in Alg. 1. SUPER and MREL are the implementation of Alg. 2 and Alg. 4, respectively. SM and MS are the hybrid algorithms in Sec. V-E.

The running time of each algorithm is composed of three major operations: (1) *Preprocessing*: superpreference filtering and/or the M-relation generation; (2) *Generation*: candidate k-subset (or k-multisubset) generation; (3) *Winnow*. Denote by g the number of sets generated in *Generation*. g is the determinant factor of the running time. It is a direct indicator of the *Generation* step and also the input of the *Winnow* step, which is an operation quadratic of its input size g. Our experiments demonstrate a strong correlation between g and the running time. In the set of experiments in Sec. VI-B, the correlation is 94.6%. Therefore, we focus on g for measuring the performance. Our experiments are practical to handle up to 8000 tuples. Further optimizations are needed for larger datasets. Recall the definitions of feature $\mathcal{F}_5$ and $\mathcal{F}_6$ in Ex. 6. Furthermore, we define the following features:

$\mathcal{F}_9 \equiv$ `SELECT sum(rating) FROM $S WHERE genre='sci-fi'`

$\mathcal{F}_{10} \equiv$ `SELECT sum(price) FROM $S`

$\mathcal{F}_{11} \equiv$ `SELECT count(title) FROM $S WHERE genre='sci-fi' and price<20.00`

$\mathcal{F}_{12} \equiv$ `SELECT sum(rating) FROM $S WHERE rating>=4.0`

The set preferences used in the experiments are listed in Table I. The corresponding superpreferences and M-relation generation SQL queries are listed in Table II, respectively. Notice that the only difference between SP1 and SP2 is that in SP1 we apply the `WHERE` condition in $\mathcal{F}_5$ which aggregates on `price`, while in SP2 we apply it to $\mathcal{F}_9$ which aggregates on `rating`. We intend to see the influence of the selectivity of attributes. In SP1, the attribue in $\mathcal{F}_5$, i.e., `price`, affects less tuples due to the `WHERE` condition, therefore the selectivity of the attribute in $\mathcal{F}_6$, i.e., `rating`, is dominating. Similarly, in SP2, the selectivity of `price` is dominating. Note that SP1 has been used extensively in our running examples.

Pref.	Prof. Schema Γ	Profile Pref. Formula C
SP1	$\langle \mathcal{F}_5, \mathcal{F}_6 \rangle$	$\mathcal{F}_5(s_1) < \mathcal{F}_5(s_2) \wedge \mathcal{F}_6(s_1) > \mathcal{F}_6(s_2)$
SP2	$\langle \mathcal{F}_9, \mathcal{F}_{10} \rangle$	$\mathcal{F}_9(s_1) > \mathcal{F}_9(s_2) \wedge \mathcal{F}_{10}(s_1) < \mathcal{F}_{10}(s_2)$
SP3	$\langle \mathcal{F}_{11}, \mathcal{F}_{12} \rangle$	$\mathcal{F}_{11}(s_1) > \mathcal{F}_{11}(s_2) \wedge \mathcal{F}_{12}(s_1) > \mathcal{F}_{12}(s_2)$

TABLE I
SET PREFERENCES

Summary of Experiments. We draw the following conclusions from the experimental results detailed below:

- SUPER, MREL, SM and MS are all effective in reducing the number of sets generated in *Generation*;
- The relative efficiency of the four optimization algorithms depends on the set preference in question, in particular, the selectivity of the attributes and the `WHERE` conditions in the feature definitions. In general, low attribute selectivity and high `WHERE` condition selectivity enhance the performance. The best algorithm for each set preference generates only 0.01% of the candidate k-subsets generated by NAIVE.
- The best of SUPER, MREL, SM and MS for each set preference also improve the scalability with input size n and k by several orders of magnitude.
- The hybrid algorithms SM and MS outperform the standalone optimizations SUPER and MREL.

A. Performance

In our first experiment, we wanted to study how much computation we could save by applying superpreference and/or M-relation. Let $k = 3$. We tested the dataset up to 1000 tuples as some algorithms do not scale up well with n.

Fig. 1(a), 1(b) and 1(c) illustrate the increase of the number of sets produced by *Generation* (g) with the increase of input size n for SP1, SP2 and SP3, respectively. For the six datasets of different sizes used, Fig. 1(d), 1(e) and 1(f) show the average ratio of the number of sets generated in each algorithm to that in NAIVE using the logarithmic y-scale.

As we can see from Fig. 1, for standalone optimization techniques SUPER and MREL, the relative efficiency depends on the set preference. MREL is more efficient in SP1 and SP3 (Fig. 1(d), 1(f)), while SUPER is more efficient in SP2 (Fig. 1(e)). If we compare SP1 and SP2 in juxtaposition, it is clear the different reactions to optimizations is caused by the different selectivity of the attributes `rating` and `price`: in our largest dataset, there are 9 distinct `rating` values and 406 distinct `price` values. For SUPER, the differences between the superpreferences of SP1 and SP2 are underlined.

$$\succ^+ \text{ in SP1} \equiv \begin{array}{l} (t_1.rating > t_2.rating \wedge t_2.genre = \textit{'sci-fi'} \\ \wedge t_1.price < t_2.price) \vee (\underline{t_1.rating > t_2.rating} \\ \underline{\wedge t_2.genre = \textit{'sci-fi'} \wedge t_1.genre \neq \textit{'sci-fi'}}) \end{array}$$

$$\succ^+ \text{ in SP2} \equiv \begin{array}{l} (t_1.rating > t_2.rating \wedge t_1.genre = \textit{'sci-fi'} \\ \wedge t_1.price < t_2.price) \vee (\underline{t_1.price < t_2.price} \\ \underline{\wedge t_1.genre = \textit{'sci-fi'} \wedge t_2.genre \neq \textit{'sci-fi'}}) \end{array}$$

SUPER is more effective in SP2 because the selectivity of `price` is higher than that of `rating` and more tuples are superpreferred in SP2. For MREL, recall that the selectivity of `rating` dominates in SP1, while the selectivity of `price` dominates in SP2. The selectivity of `rating` is much lower than that of `price`. We are thus able to consolidate on average more tuples into one M-tuple in SP1. Therefore, MREL is more effective in SP1. For SP3, the `WHERE` conditions in both features of SP3 are more restricted than that in SP1, i.e., the condition in $\mathcal{F}_{11}$ requires `price<20.00`, in addition to the condition `genre='sci-fi'` used in $\mathcal{F}_5$. MREL is able to benefit more from low selectivity and therefore achieves higher efficiency in SP3.

The hybrid algorithms SM and MS benefit from both superpreferences and M-relations, and in general have better performance than SUPER and MREL. An interesting phenomenon is that SM outperforms MS when MREL outperforms SUPER (c.f., Fig. 1(d), 1(f)), and MS outperforms SM when SUPER outperforms MREL (c.f., Fig. 1(e)). For all three set preferences (SP1, SP2 and SP3), the most efficient algorithm generates only about 0.01% of the candidate k-subsets generated by NAIVE.

Not only do the optimizations reduce the number of sets generated in *Generation*, but also improve the scalability with the input size n. In all three cases, i.e., SP1-3, (Fig. 1(a), 1(b) and 1(c)), the best algorithm displays a much slower growth compared to that of NAIVE.

Fig. 2 illustrates the increase of g with the increase of k when $n = 100$. By our theoretical analysis, increasing k by 1 raises the complexity of NAIVE from $O(n^k)$ to $O(n^{k+1})$. It is not surprising that g values are off the chart when $k > 4$ for NAIVE. The performances of the other four algorithms again depend on the specific set preference. The best algorithm

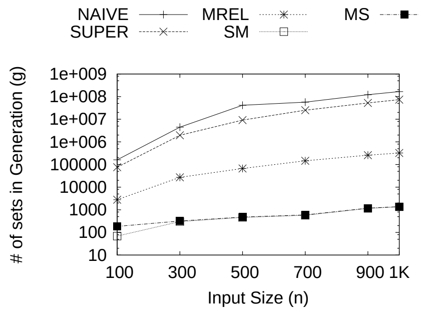
Pref.	Superpreference	M-relation Generating SQL

SP1	$t_1.rating > t_2.rating \wedge t_2.genre =$ *'sci-fi'*
	$\wedge(t_1.price < t_2.price \vee t_1.genre \neq$ *'sci-fi'*$)$
		`SELECT *, count(*) AS` A_{cnt} `FROM`
		`  (SELECT CASE WHEN r.genre='sci-fi' THEN r.price`
		`               ELSE 0 END AS` A_5`, r.rating AS` A_6 `FROM r)`
		`GROUP BY` A_5, A_6

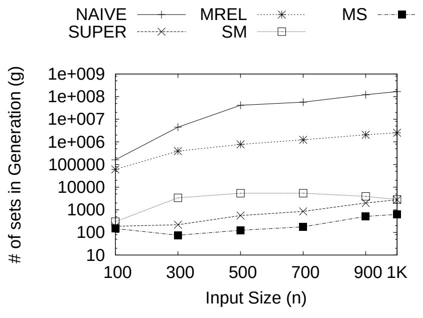
SP2	$t_1.price < t_2.price \wedge t_1.genre =$ *'sci-fi'*
	$\wedge(t_1.rating > t_2.rating \vee t_2.genre \neq$ *'sci-fi'*$)$
		`SELECT *, count(*) AS` A_{cnt} `FROM`
		`  (SELECT CASE WHEN r.genre='sci-fi' THEN r.rating`
		`               ELSE 0 END AS` A_9`, r.price AS` A_{10} `FROM r)`
		`GROUP BY` A_9, A_{10}

SP3	$t_1.genre =$ *'sci-fi'* $\wedge\, t_1.price < 20.00$
	$\wedge(t_2.genre \neq$ *'sci-fi'* $\vee\, t_2.price \geqslant 20.00)$
	$\wedge t_1.rating \geqslant 4.0$
	$\wedge(t_2.rating \leqslant 4.0 \vee t_1.rating > t_2.rating)$
		`SELECT *, count(*) AS` A_{cnt} `FROM`
		`  (SELECT CASE WHEN r.genre='sci-fi' and r.price<20.00`
		`               THEN 1  ELSE 0 END AS` A_{11}`,`
		`          CASE WHEN r.rating≥4.0 THEN r.rating`
		`               ELSE 0 END AS` A_{12}
		`  FROM r)`
		`GROUP BY` A_{11}, A_{12}

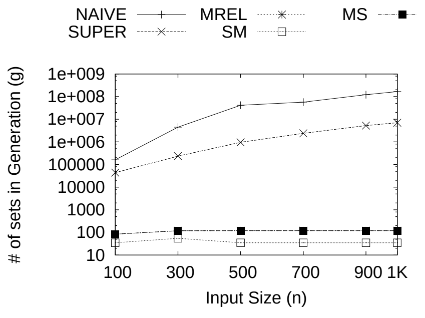
TABLE II
SUPERPREFERENCES AND M-RELATION CONSTRUCTION IN SQL



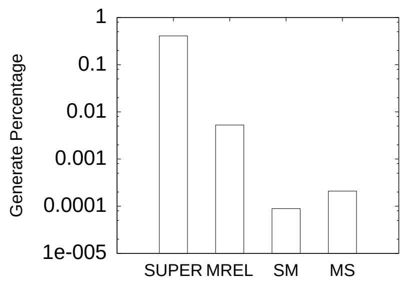
(a) # of sets v.s. n (SP1)



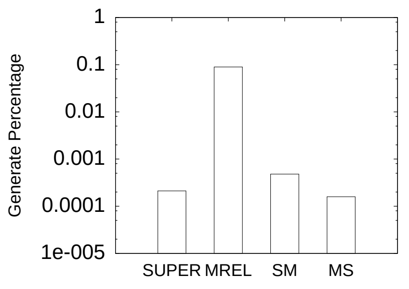
(b) # of sets v.s. n (SP2)



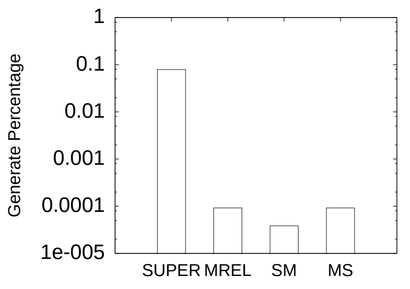
(c) # of sets v.s. n (SP3)



(d) Avg Generate Percentage (SP1)

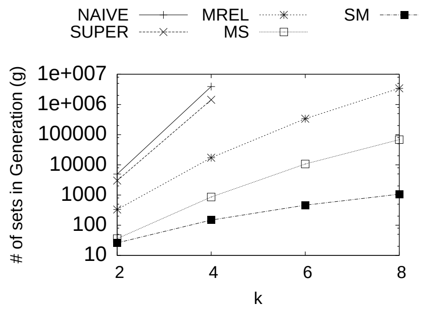


(e) Avg Generate Percentage (SP2)

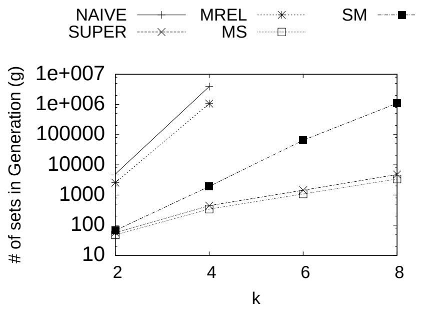


(f) Avg Generate Percentage (SP3)

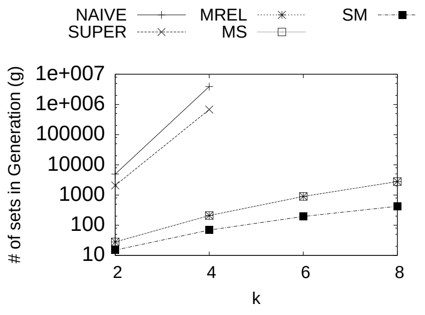
Fig. 1. Performance of Different Algorithms under Varying Input Size n



(a) # of sets v.s. k (SP1)

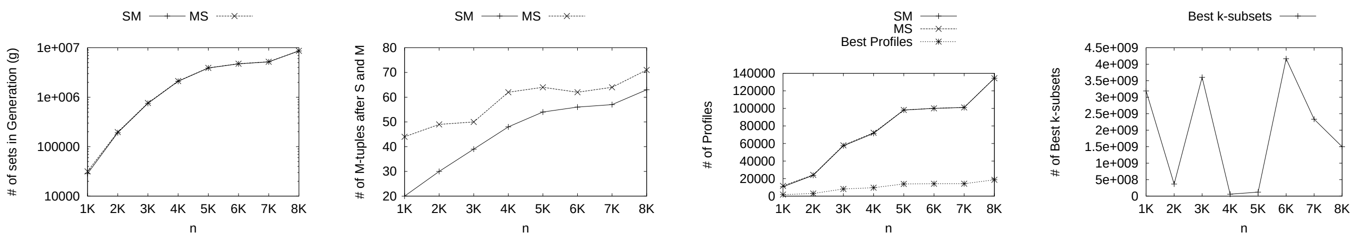


(b) # of sets v.s. k (SP2)



(c) # of sets v.s. k (SP3)

Fig. 2. Performance of Different Algorithms under Varying k



(a) Generate Size (g) (b) # of M-tuples after S and M (c) # of Profiles (d) # of Best k-subsets

Fig. 3. SM v.s. MS in SP1

for each set preference shows a between linear and quadratic scale-up with the increase of k. In particular, the exponential blowup predicted by the theoretical analysis does not happen for those best-performing algorithms.

It is worth noticing that regardless of which set preference is used, the performance of different algorithms is stable with the increase of n or k, as the curves of each algorithms do not cross each other in Fig. 1 or Fig. 2. The figures illustrate that it is crucial to choose the right algorithm given a set preference. A practical strategy would be to take a small sample of data, and try out the algorithms with a small k in order to pick the best. Notice that, even though the best of SM and MS is better than the best of SUPER and MREL, SUPER and MREL are simpler and involve less preprocessing, and thus could be desirable in some cases.

B. *SM v.s. MS*

In the second set of experiments, we took a closer look at the two best algorithms: SM and MS. In Fig. 1, SP2 and SP3 are biased towards the superpreference optimization and M-relation optimization, respectively. We therefore focused on SP1, which is less biased towards either optimization, when comparing SM and MS.

In Fig. 3, we compare the analytics of the hybrid algorithms SM and MS when $k = 5$ and n varies from 1000 to 8000 in SP1. Fig. 3(a) illustrates the number of sets generated in *Generation* for different parameter settings. Fig. 3(b) shows the number of M-tuples after the superpreference step and the M-relation step in SM and MS. Those are the M-tuples participating in the k-multisubset generation in both algorithms. This number has a positive correlation with the number of k-multisubsets generated (Fig. 3(a)). In Fig. 3(b), we can see that SM leads to $10\% \sim 50\%$ fewer M-tuples compared to those in MS. The saving decreases with the increase of n. For each parameter setting in Fig. 3(a), MS generates slightly more sets than SM does. The difference is relatively small compared to g. As a result, the two lines in Fig. 3(a) almost overlap with each other. Fig. 3(c) displays the number of profiles computed in SM and MS. The two lines also overlap with each other, since SM generates only slightly fewer profiles than MS does. As a reference, we also illustrate the number of best profiles in the profile relation in Fig. 3(c). This number grows slower than the number of profiles generated in both algorithms, which suggests room for further improvement.

For completeness, we show the result size: the number of best k-subsets in each parameter setting in Fig. 3(d). Both SM and MS correctly identify all the best k-subsets. The fluctuation in the number of best k-subsets can be explained by the existence of *super* tuples that are superpreferred to many tuples, and therefore lead to fewer best k-subsets. However, the number of such *super* tuples is not related to the size of the input. In general, the number of best k-subsets is huge, i.e., in the order of $10^8 \sim 10^9$. In fact, the result size in Fig. 3(d) is $2 \sim 5$ orders of magnitude larger than the number of candidate sets generated (Fig. 3(a)), which illustrates the compactness of the M-relation representation.

VII. 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], [7], [8].

Binshtok et al. [8] is conceptually the closest to our work. It addresses the problem of finding an optimal subset of a set of items according to given set preferences. The language for specifying such preferences uses 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 an integer value (Class 1), i.e., the number of items satisfying the predicate), or a boolean value (Class 2), i.e., whether the number of items satisfying the predicate is $> k$. Given a collection of set properties, a *set preference* is specified as either a TCP-net [12] or a scoring function. Binshtok et al. [8] gives heuristic search algorithms for finding an optimal subset. Binshtok et al. [8] considers subsets of any cardinality. For fixed-cardinality subsets, the language in [8] can easily be expressed in our framework with a simple extension of Def. 6 to boolean features. Each Class 1 *set property* in [8] can be translated to an *aggregate subset feature* with the `count` aggregate. Those features are additive.

For simplicity, we do not discuss boolean features in Def. 6. The extension to boolean features can be easily accomplished by introducing a relational operator (or an appropriate user-defined function) in the SQL definition: `SELECT expr` θ `constant FROM $S WHERE condition`, where $\theta \in \{=, \neq, <, >, \leqslant, \geqslant\}$. Each Class 2 *set property* in [8] can be translated to such a boolean feature. Such boolean features are additive when $\theta \in \{<, >, \leqslant, \geqslant\}$ and its numeric counterpart is additive. Otherwise, they are non-additive. The preference model in [8], i.e., either a scoring function or a TCP-net set preference, can be captured by our *set preference relation* as well. General aggregate features are not supported in [8]. The largest dataset experimented on in [8] is of a comparable size to those used in our experiments.

desJardins et al. [7] focuses on fixed-cardinality set preferences. It considers two 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 framework. The subset features *depth* and *diversity* are weighted sums of the *depth of attributes* and *diversity of attributes*, respectively. The *depth of an attribute* is a utility function of the most desirable attribute values of the set elements, which can be captured by a feature definition with a `WHERE` clause specifying the desirable condition and a `count` aggregate[1] in the `SELECT` clause. The *diversity of an attribute* is defined as 1 minus the *skew* of that attribute's values in the set. The computation of *skew* requires the values of *mean*, *mode* and *standard deviation* of that attribute. *Mean* and *mode* can be captured by aggregates `avg` and `max`, respectively. It is well-known that the *standard deviation* equals the square root of the second moment minus the square of the *mean*. We can capture the *second moment* of attribute A by specifying `sum(A*A)` in the `SELECT` clause. Features derived from *depth*, and features corresponding to the *mean* and the *second moment* of an attribute are additive, while features corresponding to the *mode* of an attribute are not. The preference model in [7] is maximizing an objective function composed of a linear combination of *depth* and *diversity*. This can be easily captured by our *set preference relation*.

We have just shown that we can express set preferences in both [8] and [7] using our current framework. In order to efficiently evaluate the set preferences derived from [8], [7], the M-relation optimization is always applicable. For the superpreference optimization, if the set preference formula derived can be written as a DNF in Eqn. (2), then we can apply the superpreference optimization in a systematic manner based on Thm. 1. Moreover, the features and the preference model in [7] is of a special form, so that a specialized optimization schema can be designed to achieve better optimization results.

Guha et al. [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. Guha et al. [6] considers subsets of any cardinality. The evaluation of general OPAC queries is NP-hard, and the specific form of the OPAC queries discussed in [6] corresponds to the well-known *multi-attribute knapsack* problem and is therefore NP-complete. Approximation algorithms are given for query evaluation. For fixed-cardinality subsets, again, in our framework the atomic aggregation constraints can be captured by k*-subset features*, and the parameters and the objective function by the *preference formula over profiles*.

[1]In cases where values outside the desirable range are penalized to varying degrees, we can define one feature for each degree of penalization.

VIII. Conclusions and Future Work

We demonstrated a "logic + SQL" framework for preference queries over sets, and designed a query rewriting technique based on *superpreference* and a consolidation technique based on *M-relation* to effectively optimize the generation of the *best* subsets of the given relation. For future directions, we see more opportunities in query optimization, especially for non-additive features. Also, a set preference with low selectivity, e.g., $\varnothing$, often leads to a large result set. We therefore consider the set preference elicitation an important topic whose study will enrich the picture of set preferences. Moreover, if the result set is large, we can adopt additional set ranking or browsing techniques to facilitate the navigation of results. In addition, one should study the embedding of the *best-subset* generation into relational query languages. For example, the best subsets could be returned non-deterministically. Another direction is designing more expressive set preference languages and studying their expressive power. An interesting problem is how to relax the fixed cardinality constraint in our framework. As we have pointed out in Sec. V, a complete absence of the cardinality constraint is likely to lead to problematic semantics. However, it is possible to relax this constraint in some applications: we might want to work with subsets of cardinality within a *range* instead. The *M-relation* optimization will continue to work as it is independent of the fixed cardinality assumption. On the other hand, the current version of *superpreference* relies on the fact that all subsets in competition are of the same cardinality. Extensions are needed to accommodate the relaxed cardinality assumption.

References

[1] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *ICDE*, 2001, pp. 421–430.

[2] W. Kießling, "Foundations of preferences in database systems," in *VLDB*, 2002, pp. 311–322.

[3] W. Kießling and G. Köstler, "Preference SQL - design, implementation, experiences," in *VLDB*, 2002, pp. 990–1001.

[4] J. Chomicki, "Preference formulas in relational queries," *ACM Trans. Database Syst.*, vol. 28, no. 4, pp. 427–466, 2003.

[5] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole, "CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements," *J. Artif. Intell. Res. (JAIR)*, vol. 21, pp. 135–191, 2004.

[6] S. Guha, D. Gunopulos, N. Koudas, D. Srivastava, and M. Vlachos, "Efficient approximation of optimization queries under parametric aggregation constraints," in *VLDB*, 2003, pp. 778–789.

[7] M. desJardins and K. Wagstaff, "DD-pref: A language for expressing preferences over sets," in *AAAI*, 2005, pp. 620–626.

[8] M. Binshtok, R. I. Brafman, S. E. Shimony, A. Mani, and C. Boutilier, "Computing optimal subsets," in *AAAI*, 2007, pp. 1231–1236.

[9] D. Mindolin and J. Chomicki, "Discovering relative importance of skyline attributes," in *VLDB*, 2009, pp. 610–621.

[10] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting: Theory and optimizations," in *Intelligent Information Systems*, 2005, pp. 595–604.

[11] D. L. Kreher and D. R. Stinson, *Combinatorial algorithms: generation, enumeration and search*. CRC Press LTC, 1998.

[12] R. I. Brafman, C. Domshlak, and S. E. Shimony, "On graphical modeling of preference and importance," *J. Artif. Intell. Res. (JAIR)*, vol. 25, pp. 389–424, 2006.