Probabilities and Sets in Preference Querying

by

Xi Zhang

April 16, 2010

A dissertation submitted to the
Faculty of the Graduate School of
the University at Buffalo, State University of New York
in partial fulfillment of the requirements for the
degree of
Doctor of Philosophy

Department of Computer Science and Engineering

Acknowledgments

I am deeply indebted to my adviser Dr. Jan Chomicki for his continuous encouragement, guidance and support in every stage of my PhD study. His enthusiasm, knowledge and diligence have been a constant source of inspiration for me. Through him, I see what a great researcher should be. His vision and experience guided me consistently through the dissertation writing and made this experience enjoyable.

I am very grateful to the other brilliant faculty members who served on my committee: Dr. Hung Q. Ngo and Dr. Michalis Petropoulos. Dr. Hung Q. Ngo, whose lectures were by far my favorites, enlightened me on numerous math puzzles, one of which later turned out to be very useful in this dissertation. Dr. Michalis Petropoulos provided warm encouragement and valuable comments throughout the course of writing this dissertation.

I would also like to thank the distinguished database researchers at AT&T Labs, Dr. Graham Cormode, Dr. Lukasz Golab, Dr. Flip Korn and Dr. Divesh Srivastava, with whom I had the honor to work with. Their passion, curiosity and spirits are contagious. They generously shared their vision and provided insightful comments on early drafts of my work and other research problems.

This dissertation is dedicated to my beloved husband Chao Chen, my most loving parents Ling Wei and Zhemin Zhang, and my most loyal friends Lan, Jing, Michelle, Sixia and Slawek. It simply would not have been possible without their unconditional love and support.

This research has been supported by NSF Grant IIS-0307434 and NSF Grant IIS-0119186.

Contents

List of Figures

List of Tables

Probabilities and Sets in Preference Querying

by

Xi Zhang

Submitted to the Department of Computer Science and Engineering
on April 16, 2010, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

User preferences in databases are attracting increasing interests with the boom of information systems and the trend of personalization. In literature, there are two different frameworks dealing with this topic, namely *quantitative* approaches and *qualitative* approaches. The former assume the availability of a *scoring function*, while the latter do not. In qualitative approaches, preferences are expressed using *preference formulas*. We investigate three advanced topics on preferences stemming from those frameworks.

First, we study *top-k queries over uncertain data* in the quantitative framework. We formulate three intuitive semantic properties for top-k queries in probabilistic databases, and propose Global-Top$k$ query semantics which satisfies them to a great degree. We also design efficient dynamic programming algorithms for query evaluation.

Second, we observe that all work on top-k queries in probabilistic database focus on *ordinal scores*, however there are applications where *cardinal scores* are more appropriate. This motivates our work on *preference strength*, where we consider the magnitude of score in addition to the order it establishes over tuples.

Finally, as a counterpart to the top-k query in the quantitative framework, we explore the *set preference* problem in the qualitative framework. Observing the fact that preferences can also be collective in this case, our goal is to tackle this second-order problem with first-order tools. 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 propose algorithms to compute the "best" sets.

Chapter 1

Introduction

1.1 Motivations

1.1.1 Top-k Queries over Uncertain Databases

The study of incompleteness and uncertainty in databases has long been of interest to the database community [35, 12, 31, 1, 27, 61, 41]. Recently, this interest has been rekindled by an increasing demand for managing rich data, often incomplete and uncertain, emerging from scientific data management, sensor data management, data cleaning and information extraction. Dalvi et al. [20] focuses on query evaluation in traditional probabilistic databases; Benjelloun et al. [4] supports uncertain data and data lineage in Trio [55]; Olteanu et al. [49] uses the vertical World-Set representation of uncertain data in MayBMS [46]. The standard semantics adopted in most works is the *possible worlds* semantics [35, 27, 61, 4, 20, 49].

On the other hand, since the seminal papers of Fagin et al. [23, 25], the top-k problem has been extensively studied in multimedia databases [48], middleware systems [45], data cleaning [30] and core technology in relational databases [33, 34]. In the top-k problem, each tuple is given a *score*, and users are interested in k tuples with the highest scores.

More recently, the top-k problem has been studied in probabilistic databases [52, 53, 51]. Those papers, however, are solving two essentially different top-k problems. Soliman et al. [52, 53] assumes the existence of a scoring function to rank tuples. Probabilities provide information on how likely tuples will appear in the database. In contrast, in [51], the ranking criterion for top-k is the probability associated with each query answer. In many applications, it is necessary to deal with tuple probabilities and scores at the same time. Thus, we use the model of [52, 53]. Even in this model, different semantics for top-k queries are possible, so a part of the challenge is to categorize different semantics.

As a motivating example, let us consider the following graduate admission example.

Example 1. *A graduate admission committee needs to select two winners of a fellowship. They narrow the candidates down to the following short list:*

Name	*Overall Score*	*Prob. of Acceptance*
Aidan	0.65	0.3
Bob	0.55	0.9
Chris	0.45	0.4

where the overall score *is the normalized score of each candidate based on their qualifications, and the* probability of acceptance *is derived from historical statistics on candidates with similar qualifications and background.*

The committee want to make offers to the best two candidates who will take the offer. This decision problem can be formulated as a top-k query over the above probabilistic relation, where $k = 2$.

In Example 1, each tuple is associated with an *event*, whether the candidate will accept the offer or not. The probability of the event is shown next to each tuple. In this example, all the events of tuples are independent, and tuples are therefore said to be *independent*. Such a relation is said to be *simple*. In contrast, Example 2 illustrates a more general case.

Example 2. *In a sensor network deployed in a habitat, each sensor reading comes with a confidence value* Prob*, which is the probability that the reading is valid. The following table shows the temperature sensor readings at a given sampling time. These data are from two sensors, Sensor 1 and Sensor 2, which correspond to two* parts *of the relation, marked C_1 and C_2, respectively. Each sensor has only one* true *reading at a given time, therefore tuples from the same part of the relation correspond to exclusive events.*

	Temp.°F (Score)	*Prob*
C_1	22	0.6
	10	0.4
C_2	25	0.1
	15	0.6

Our question is: "What is the temperature of the warmest spot?"

The question can be formulated as a top-k query, where $k = 1$, over a probabilistic relation containing the above data. The scoring function is the temperature. However, we must take into consideration that the tuples in each part $C_i, i = 1, 2$, are exclusive.

1.1.2 Set Preferences

In the previous section, *preferences* are modeled as *scoring functions*. An alternative to this *quantitative* approach is the *qualitative* approach, where *preferences* are expressed using *preference formulas* instead. In recent years, research adopting this approach flourishes in the database and AI communities [7, 36, 37, 16, 8]. The issues addressed in that research include preference specification, preference query languages, and preference query evaluation and optimization. However, the research on qualitative 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 3. *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 are 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, i.e., 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 *if and only if* $t_1.price < t_2.price$*. Then the top* 3 *books in* $Book$ *(according to this preference) constitute the* best *answer set.*

However, in the other cases, e.g. (C2-C3) and the prioritized composition of (C2) and (C1), such a simulation is not possible.

Example 3 motivates our framework for *set preferences*, which is based on the observation that a large class of set preferences has two components: (1) Quantities of interest; (2) Desired value or order of those quantities.

Example 4. *Here is a summary of those two components in Example 3.*

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

1.2 Our Contributions

In this section, we describe our research in several directions relevant to probabilities and sets in preference querying.

Top-k Queries over Uncertain Databases

In this topic, we assume the existence of a *scoring function* for top-k queries in probabilistic databases. First, in Chapter 3, we formulate three intuitive semantic postulates and use them to analyze and categorize different top-k semantics in probabilistic databases. We then propose a new semantics for top-k queries in probabilistic databases, called Global-Top$k$, which satisfies the above postulates to a large degree.

Second, we exhibit polynomial algorithms for evaluating top-k queries under the Global-Top$k$ semantics in *simple* probabilistic databases and general probabilistic databases, under *injective* scoring functions. We design efficient heuristics to improve the performance of the basic algorithms. Experiments are carried out to demonstrate the efficacy of those optimizations.

Third, in Chapter 4, we generalize Global-Top$k$ semantics to general scoring functions, where ties are allowed, by introducing the notion of *allocation policy*. We propose dynamic-programming based algorithms for query evaluation under the *Equal* allocation policy.

Preference Strength in Probabilistic Ranking Queries

In Chapter 5, we study the problem of *preference strength* in top-k queries in probabilistic databases. First, we formulate two more semantic postulates of top-k queries in probabilistic databases, which are related to *sensitivity*. Then, we propose a parameterized semantics of top-k queries in probabilistic databases, called Global-Top$k^{\gamma,\beta}$, which considers *cardinal scores* and satisfies the sensitivity postulates. Finally, we exhibit a polynomial algorithm to elicit the parameters in the Global-Top$k^{\gamma,\beta}$ semantics. Experiments are carried out to illustrate the influence of preference strength in the semantics.

Set Preferences

In Chapter 6, we first elaborate our set preference framework based on the qualitative tuple preference framework in the literature [16]. Our set preference framework consists of two components: (1) *profiles*: tuples 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 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.

In Chapter 6, we discuss the computational issues involved in computing the *best* subsets. Furthermore, we design efficient optimizations to significantly reduce the computation effort, and carry out empirical study on their performance.

Chapter 2

Preliminaries

In this chapter, we review the standard notions of order theory and set theory.

2.1 Binary Relation

A binary relation $>$ is

- irreflexive: $\forall x.\ x \not> x$,
- asymmetric:$\forall x, y.\ x > y \Rightarrow y \not> x$,
- transitive: $\forall x, y, z.\ (x > y \land y > z) \Rightarrow x > z$,
- negatively transitive: $\forall x, y, z.\ (x \not> y \land y \not> z) \Rightarrow x \not> z$,
- connected: $\forall x, y.\ x > y \lor y > x \lor x = y$.

- A *strict partial order* is an irreflexive, transitive (and thus asymmetric) binary relation.
- A *weak order* is a negatively transitive strict partial order.
- A *total order* is a connected strict partial order.

2.2 Set and Multiset

The *cardinality of a set* s, denote by $|s|$, is the total number of members in s.

A *multiset* is a generalization of a set. In a set, each member has only one occurrence, while in a multiset, each member can have more than one occurrence.

The *cardinality of a multiset* s, also denoted by $|s|$, is the total number of occurrences in s.

A set s' is a *subset* of set s if each member of s' is also a member of s.

A multiset s' is a *multisubset* of set s if each member of s' is also a member of s.

Chapter 3

Top-k Queries in Uncertain Databases under Injective Scoring Functions

In this chapter, we address the problem of top-k queries in probabilistic databases, semantically and computationally. Most of the discussion focuses on the scenario when the scoring function involved is *injective*, i.e. no ties allowed. We defer the discussion of general scoring functions to Chapter 4.

3.1 Basic Notions

3.1.1 Probabilistic Relations

To simplify the discussion, we assume that a probabilistic database contains a single *probabilistic relation*. We refer to a traditional database relation as a *deterministic relation*. A *partition* $\mathcal{C}$ of R is a collection of non-empty subsets of R such that every tuple belongs to one and only one of the subsets. That is, $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ such that $C_1 \cup C_2 \cup \ldots \cup C_m = R$ and $C_i \cap C_j = \emptyset, 1 \leqslant i \neq j \leqslant m$. Each subset $C_i, i = 1, 2, \ldots, m$ is a *part* of the partition $\mathcal{C}$. A *probabilistic relation* R^p has three components, a *support (deterministic) relation* R, a probability function p and a partition $\mathcal{C}$ of the support relation R. The probability function p maps every tuple in R to a probability value in $(0, 1]$. The partition $\mathcal{C}$ divides R into subsets such that the tuples within each subset are exclusive and therefore their probabilities sum up to at most 1. In the graphical presentation of R, we use horizontal lines to separate tuples from different parts.

Definition 3.1.1 (Probabilistic Relation)**.** *A probabilistic relation R^p is a triplet $\langle R, p, \mathcal{C}\rangle$, where R is a support deterministic relation, p is a probability function $p : R \mapsto (0, 1]$ and $\mathcal{C}$ is a partition of R such that $\forall C_i \in \mathcal{C}, \sum_{t \in C_i} p(t) \leqslant 1$.*

In addition, we make the assumption that tuples from different parts of of $\mathcal{C}$ are independent, and tuples within the same part are exclusive. Soliman et al. [52, 53] and Ré et al. [51] provide more general probabilistic database models. Definition 3.1.1 is equivalent to the model used in Soliman et al. [52, 53] with exclusive tuple generation rules.

Example 2 shows an example of a probabilistic relation whose partition has two parts. Generally, each part corresponds to a real world entity, in this case, a sensor. Since there is only one true state of an entity, tuples from the same part are exclusive. Moreover, the probabilities of all possible states of an entity sum up to at most 1. In Example 2, the sum of the probabilities of tuples from Sensor 1 is 1, while that from Sensor 2 is 0.7. This can happen for various reasons. In the above example, we might encounter a physical difficulty in collecting the sensor data, and end up with partial data.

Definition 3.1.2 (Simple Probabilistic Relation)**.** *A probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$ is simple iff the partition $\mathcal{C}$ contains only singleton sets.*

The probabilistic relation in Example 1 is *simple* (individual parts not illustrated). Note that in this case, $|R| = |\mathcal{C}|$.

We adopt the well-known *possible worlds* semantics for probabilistic relations [35, 27, 61, 4, 20, 49].

Definition 3.1.3 (Possible World)**.** *Given a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, a deterministic relation W is a* possible world *of R^p iff*

1. *W is a subset of the support relation, i.e., $W \subseteq R$;*

2. *For every part C_i in the partition $\mathcal{C}$, at most one tuple from C_i is in W, i.e., $\forall C_i \in \mathcal{C}, |C_i \cap W| \leqslant 1$;*

3. The probability of W (defined by Equation (3.1)) is positive, i.e., $Pr(W) > 0$.

$$Pr(W) = \prod_{t \in W} p(t) \prod_{C_i \in \mathcal{C}'} (1 - \sum_{t \in C_i} p(t)) \tag{3.1}$$

where $\mathcal{C}' = \{C_i \in \mathcal{C} | W \cap C_i = \varnothing\}$.

Denote by $pwd(R^p)$ *the set of all possible worlds* of R^p.

3.1.2 Scoring function

A *scoring function over a deterministic relation* R is a function from R to real numbers, i.e., $s : R \mapsto \mathbb{R}$. The function s induces a *preference relation* $\succ_s$ and an *indifference relation* $\sim_s$ on R. For any two distinct tuples t_i and t_j from R,

$$t_i \succ_s t_j \text{ iff } s(t_i) > s(t_j);$$
$$t_i \sim_s t_j \text{ iff } s(t_i) = s(t_j).$$

A *scoring function over a probabilistic relation* $R^p = \langle R, p, \mathcal{C} \rangle$ is a scoring function s over its support relation R. In general, a scoring function establishes a *weak order* over R, where tuples from R can tie in score. However, when the scoring function s is *injective*, $\succ_s$ is a *total order*. In such a case, no two tuples tie in score.

3.1.3 Top-k Queries

Definition 3.1.4 (Top-k Answer Set over a Deterministic Relation)**.** *Given a deterministic relation R, a non-negative integer k and a scoring function s over R, a top-k answer set in R under s is a set T of tuples such that*

1. $T \subseteq R$;

2. If $|R| < k$, $T = R$, otherwise $|T| = k$;

3. $\forall t \in T \; \forall t' \in R - T.\; t \succ_s t'$ or $t \sim_s t'$.

According to Definition 3.1.4, given k and s, there can be more than one top-k answer set in a deterministic relation R. The evaluation of a top-k query over R returns one of them nondeterministically, say S. However, if the scoring function s is injective, S is unique, denoted by $top_{k,s}(R)$.

3.2 Semantics of Top-k Queries

In the following two sections, we restrict our discussion to *injective* scoring functions. We will discuss the generalization to general scoring functions in Chapter 4.

3.2.1 Semantic Postulates for Top-k Answers

Probability opens the gate for various possible semantics for top-k queries. As the semantics of a probabilistic relation involves a set of possible worlds, it is to be expected that there may be multiple top-k answer set obtained from different worlds, even under an injective scoring function. The answer to a top-k query over a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$ should clearly be a set of tuples from its support relation R. We formulate below three intuitive *postulates*, which serve as benchmarks to categorize different semantics.

In the following discussion, denote by $Ans_{k,s}(R^p)$ the collection of all top-k answer sets of R^p under the function s.

Postulates

- **Static Postulates**

  1. *Exact* k: When R^p is sufficiently large ($|\mathcal{C}| \geqslant k$), the cardinality of every top-k answer set S is exactly k;

$$|\mathcal{C}| \geqslant k \Rightarrow [\forall S \in Ans_{k,s}(R^p).\ |S| = k].$$

  2. *Faithfulness*: For every top-k answer set S and any two tuples $t_1, t_2 \in R$, if both the score and the probability of t_1 are higher than those of t_2 and $t_2 \in S$, then $t_1 \in S$;

$$\forall S \in Ans_{k,s}(R^p)\ \forall t_1, t_2 \in R.\ s(t_1) > s(t_2) \wedge p(t_1) > p(t_2) \wedge t_2 \in S \Rightarrow t_1 \in S.$$

- **Dynamic Postulate**

  $\cup\, Ans_{k,s}(R^p)$ denotes the union of all top-k answer sets of $R^p = \langle R, p, \mathcal{C}\rangle$ under the function s. For any $t \in R$,

  $$t \text{ is a } \textit{winner} \text{ iff } t \in \cup\, Ans_{k,s}(R^p)$$
  $$t \text{ is a } \textit{loser} \text{ iff } t \in R - \cup\, Ans_{k,s}(R^p)$$

3. *Stability*:

   – Raising the score/probability of a winner will not turn it into a loser;

   (a) If a scoring function s' is such that $s'(t) > s(t)$ and for every $t' \in R - \{t\}$, $s'(t') = s(t')$, then

   $$t \in \cup\, Ans_{k,s}(R^p) \Rightarrow t \in \cup\, Ans_{k,s'}(R^p).$$

   (b) If a probability function p' is such that $p'(t) > p(t)$ and for every $t' \in R - \{t\}$, $p'(t') = p(t')$, then

   $$t \in \cup\, Ans_{k,s}(R^p) \Rightarrow t \in \cup\, Ans_{k,s}((R^p)'),$$

   where $(R^p)' = \langle R, p', \mathcal{C}\rangle$.

   – Lowering the score/probability of a loser will not turn it into a winner.

   (a) If a scoring function s' is such that $s'(t) < s(t)$ and for every $t' \in R - \{t\}$, $s'(t') = s(t')$, then

   $$t \in R - \cup\, Ans_{k,s}(R^p) \Rightarrow t \in R - \cup\, Ans_{k,s'}(R^p).$$

   (b) If a probability function p' is such that $p'(t) < p(t)$ and for every $t' \in R - \{t\}$, $p'(t') = p(t')$, then

   $$t \in R - \cup\, Ans_{k,s}(R^p) \Rightarrow t \in R - \cup\, Ans_{k,s}((R^p)'),$$

where $(R^p)' = \langle R, p', \mathcal{C} \rangle$.

All of those postulates reflect certain requirements of top-k answers from the user.

Exact k expresses user expectations about the size of the result. Typically, a user issues a top-k query in order to restrict the size of the result and get a subset of cardinality k (cf. Example 1). Therefore, k can be a crucial parameter specified by the user that should be complied with.

Faithfulness reflects the significance of score and probability in a static environment. It plays an important role in designing efficient query evaluation algorithms. The satisfaction of *Faithfulness* admits a set of pruning techniques based on *monotonicity*.

Stability reflects the significance of score and probability in a dynamic environment, where it is common that users might update score/probability on-the-fly. *Stability* requires that the consequences of such changes should not be counterintuitive.

A more in-depth discussion on postulates can be found in Section 3.2.3.

3.2.2 Global-Top$k$ Semantics

We propose here a new top-k answer semantics in probabilistic relations, namely **Global-Top**k, which satisfies the postulates formulated in Section 3.2.1 to a large degree:

- **Global-Top**k: return k highest-ranked tuples according to their probability of being in the top-k answers in possible worlds.

Considering a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$ under an injective scoring function s, any $W \in pwd(R^p)$ has a unique top-k answer set $top_{k,s}(W)$. Each tuple from the support relation R can be in the top-k answer set (in the sense of Definition 3.1.4) in zero, one or more possible worlds of R^p. Therefore, the sum of the probabilities of those possible worlds provides a global ranking criterion.

Definition 3.2.1 (Global-Top$k$ Probability)**.** *Assume a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, a non-negative integer k and an injective scoring function s over R^p. For any tuple t in R, the Global-Top$k$ probability of t, denoted by $P^{R^p}_{k,s}(t)$, is the sum of the probabilities of all possible worlds of R^p whose top-k answer set contains t.*

$$P_{k,s}^{R^p}(t) = \sum_{\substack{W \in pwd(R^p) \\ t \in top_{k,s}(W)}} Pr(W). \tag{3.2}$$

For simplicity, we skip the superscript in $P_{k,s}^{R^p}(t)$, i.e., $P_{k,s}(t)$, when the context is unambiguous.

Definition 3.2.2 (Global-Top$k$ Answer Set in a Probabilistic Relation)**.** *Given a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, a non-negative integer k and an injective scoring function s over R^p, a Global-Top$k$ answer set in R^p under s is a set T of tuples such that*

1. $T \subseteq R$;

2. If $|R| < k$, $T = R$, otherwise $|T| = k$;

3. $\forall t \in T, \forall t' \in R - T, P_{k,s}(t) \geqslant P_{k,s}(t')$.

Notice the similarity between Definition 3.2.2 and Definition 3.1.4. In fact, the probabilistic version only changes the last condition, in which the Global-Top$k$ probability plays the role of the scoring function. This semantics preserves the nondeterministic nature of Definition 3.1.4. For example, if two tuples are of the same Global-Top$k$ probability, and there are $k - 1$ tuples with a higher Global-Top$k$ probability, Definition 3.2.2 allows one of the two tuples to be added to the top-k answer set nondeterministically. Example 5 gives an example of the Global-Top$k$ semantics.

Example 5. *Consider the top-2 query in Example 1. Clearly, the scoring function here is the* Overall Score*. The following table shows all the possible worlds and their probabilities. For each world, the names of the people in the top-2 answer set of that world are underlined.*

Possible World	*Prob*
$W_1 = \varnothing$	0.042
$W_2 = \{\underline{Aidan}\}$	0.018
$W_3 = \{\underline{Bob}\}$	0.378
$W_4 = \{\underline{Chris}\}$	0.028
$W_5 = \{\underline{Aidan}, \underline{Bob}\}$	0.162
$W_6 = \{\underline{Aidan}, \underline{Chris}\}$	0.012
$W_7 = \{\underline{Bob}, \underline{Chris}\}$	0.252
$W_8 = \{\underline{Aidan}, \underline{Bob}, Chris\}$	0.108

Chris is in the top-2 answer of W_4, W_6, W_7, so the top-2 probability of Chris is $0.028 + 0.012 + 0.252 = 0.292$. Similarly, the top-$2$ probability of Aidan and Bob are 0.9 and 0.3, respectively. $0.9 > 0.3 > 0.292$, therefore Global-Top$k$ will return $\{Aidan, Bob\}$.

Note that top-k answer sets may be of cardinality less than k for some possible worlds. We refer to such possible worlds as *small* worlds. In Example 5, $W_{1...4}$ are all small worlds.

3.2.3 Other Semantics

We present here the most well-established top-k semantics in the literature.

Soliman et al. [52] proposes two semantics for top-k queries in probabilistic relations.

- *U-Top$k$*: return the most probable top-k answer set that belongs to possible world(s);
- *U-kRanks*: for $i = 1, 2, \ldots, k$, return the most probable i^{th}-ranked tuples across all possible worlds.

Hua et al. [32] independently proposes PT-k, a semantics based on Global-Top$k$ probability as well. PT-k takes an additional parameter: probability threshold $p_\tau \in (0, 1]$.

- *PT-k*: return every tuple whose probability of being in the top-k answers in possible worlds is at least p_τ.

Example 6. *Continuing Example 5, under U-Top$k$ semantics, the probability of top-2 answer set $\{Bob\}$ is 0.378, and that of $\{Aidan, Bob\}$ is $0.162 + 0.108 = 0.27$. Therefore,*

$\{Bob\}$ is more probable than $\{Aidan, Bob\}$ under U-Top$k$. In fact, $\{Bob\}$ is the most probable top-2 answer set in this case, and will be returned by U-Top$k$.

Under U-kRanks semantics, Aidan is in 1^{st} place in the top-2 answer of W_2, W_5, W_6, W_8, therefore the probability of Aidan being in 1^{st} place in the top-2 answers in the possible worlds is $0.018 + 0.162 + 0.012 + 0.108 = 0.3$. *However, Aidan is not in 2^{nd} place in the top-2 answer of any possible world, therefore the probability of Aidan being in 2^{nd} place is* 0. *This is summarized in the following table.*

	Aidan	*Bob*	*Chris*
Rank 1	0.3	$\underline{0.63}$	0.028
Rank 2	*0*	$\underline{0.27}$	0.264

U-kRanks selects the tuple with the highest probability at each rank (underlined) and takes the union of them. In this example, Bob wins at both Rank 1 and Rank 2. Thus, the top-2 answer returned by U-kRanks is $\{Bob\}$.

PT-k returns every tuple with its Global-Top$k$ probability above the user specified threshold p_τ, therefore the answer depends on p_τ. Say $p_\tau = 0.6$, then PT-k return $\{Aidan\}$, as it is the only tuple with a Global-Top$k$ probability at least 0.6.

The postulates introduced in Section 3.2.1 lay the ground for analyzing different semantics. In Table 3.1, a single "✓" (resp. "×") indicates that postulate is (resp. is not) satisfied under that semantics. "✓/×" indicates that, the postulate is satisfied by that semantics in *simple* probabilistic relations, but not in the general case.

Semantics	Exact k	Faithfulness	Stability
Global-Top$k$	✓	✓/×	✓
PT-k	×	✓/×	✓
U-Top$k$	×	✓/×	✓
U-kRanks	×	×	×

Table 3.1: Postulate Satisfaction for Different Semantics

For *Exact k*, Global-Top$k$ is the only semantics that satisfies this postulate. Example 6 illustrates the case where U-Top$k$, U-kRanks and PT-k violate this postulate. It is not satisfied by U-Top$k$ because a *small* possible world with a high probability could dominate other worlds. In this case, the dominating possible world might not have enough tuples. It is also violated by U-kRanks because a single tuple can win at multiple ranks in U-kRanks. In PT-k, if the threshold parameter p_τ is set too high, then less than k tuples will be returned (as in Example 6). As p_τ decreases, PT-k return more tuples. In the extreme case when p_τ approaches 0, every tuple with a positive Global-Top$k$ probability will be returned.

For *Faithfulness*, Global-Top$k$ violates it when exclusion rules lead to a highly restricted distribution of possible worlds, and are combined with an unfavorable scoring function (*see* Appendix A (5)). PT-k violates *Faithfulness* for the same reason (*see* Appendix A (6)). U-Top$k$ violates *Faithfulness* since it requires all tuples in a top-k answer set to be compatible. This postulate can be violated when a high-score/probability tuple could be dragged down arbitrarily by its compatible tuples which are not very likely to appear (*see* Appendix A (7)). U-kRanks violates both *Faithfulness* and *Stability*. Under U-kRanks, instead of a set, a top-k answer is an ordered vector, where ranks are significant. A change in a tuple's probability/score might have unpredictable consequence on ranks, therefore those two postulates are not guaranteed to hold (*see* Appendix A (8)(12)).

Faithfulness is a postulate which can lead to significant pruning in practice. Even though it is not fully satisfied by any of the four semantics, some degree of satisfaction can still be beneficial, as it will help us find pruning rules. For example, our optimization in Section 3.3.2 explores the *Faithfulness* of Global-Top$k$ in simple probabilistic databases. Another example: one of the pruning techniques in [32] explores the *Faithfulness* of exclusive tuples in general probabilistic databases as well.

See Appendix A for the proofs of the results in Table 3.1.

It worths mentioning here that the intention of Table 3.1 is to provide a list of semantic postulates, so that users would be able to choose the appropriate postulates for an application. For example, in a government contract bidding, only k companies from the first round will advance to the second round. The score is inverse to the price offered by a company, and the probability is the probability that company will complete the task on time. The constraint of k is hard, and thus *Exact k* is a must for the top-k semantics chosen. In contrast, during college admission, where the score reflects the qualifications of an applicant and the probability is the probability of offer acceptance, while we intend to have a class of k students, there is usually room for fluctuation. In this case, *Exact k* is not a must. It is the same story with *Faithfulness* and *Stability*: *Faithfulness* is required in applications such as auctions, where the score is the value of an item and the probability is the availability of the item. In this case, it is natural to aim at the "best deals", i.e., items with high value and high availability. *Stability* is a common postulate required by many dynamic applications. For example, we want to maintain a list of k best sellers, where the score is inverse to the price of an item and the probability is its availability. It is to be expected that a discounted price and improved availability of an item should not have an adverse influence on the item's stand on the list of k best sellers[1].

In short, we are not advertising that a specific semantics is superior/inferior to any other semantics using Table 3.1. Rather, with the help of Table 3.1, we hope that users will be able to search for the most appropriate semantics based on the right combination of postulates for their applications.

Recently, Cormode et al. [19] proposed a new semantics, *Expected Rank*, and showed that it satisfies *Exact k* and *Stability*, but not *Faithfulness*. They also introduced new postulates: *containment*, *unique-rank* and *value-invariance*, and studied the postulate satisfaction of different semantics including Global-Top$k$. Roughly speaking, *containment* stipulates the containment relationship between top-k answers when increasing k in top-k queries. *Unique-rank* demands a uniquely ranked list of tuples as the top-k answer. *Value-invariance* requires that the top-k answer stays the same as long as the change in the scoring function is order-preserving. Some of those three postulates are more intuitive than others. However, again, it is up to the application to decide on the desirable subset of postulates.

[1]In real life, we sometimes observe cases when *stability* does not hold: a cheaper Wii console with improved availability does not make it more popular than it was. The reason could be psychological.

3.3 Query Evaluation under Global-Top$k$

3.3.1 Simple Probabilistic Relations

We first consider a *simple* probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$ under an injective scoring function s.

Proposition 3.3.1. *Given a simple probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$ and an injective scoring function s over R^p, if $R = \{t_1, t_2, \ldots, t_n\}$ and $t_1 \succ_s t_2 \succ_s \ldots \succ_s t_n$, the following recursion on Global-Top$k$ queries holds:*

$$q(k,i) = \begin{cases} 0 & k = 0 \\ p(t_i) & 1 \leqslant i \leqslant k \\ (q(k,i-1)\dfrac{\bar{p}(t_{i-1})}{p(t_{i-1})} + q(k-1,i-1))p(t_i) & \textit{otherwise} \end{cases} \tag{3.3}$$

where $q(k,i) = P_{k,s}(t_i)$ and $\bar{p}(t_{i-1}) = 1 - p(t_{i-1})$.

Proof. See Appendix B.

Notice that Equation (3.3) involves probabilities only, while the scores are used to determine the order of computation.

Example 7. *Consider a simple probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, where $R = \{t_1, t_2, t_3, t_4\}$, $p(t_i) = p_i, 1 \leqslant i \leqslant 4$, $\mathcal{C} = \{\{t_1\}, \{t_2\}, \{t_3\}, \{t_4\}\}$, and an injective scoring function s such that $t_1 \succ_s t_2 \succ_s t_3 \succ_s t_4$. The following table shows the Global-Top$k$ probability of t_i, where $0 \leqslant k \leqslant 2$.*

k	t_1	t_2	t_3	t_4
0	0	0	0	0
1	p_1	$\bar{p}_1p_2$	$\bar{p}_1\bar{p}_2p_3$	$\bar{p}_1\bar{p}_2\bar{p}_3p_4$
2	$\mathbf{p_1}$	$\mathbf{p_2}$	$(\bar{\mathbf{p}}_\mathbf{2} + \bar{\mathbf{p}}_\mathbf{1}\mathbf{p_2})\mathbf{p_3}$	$((\bar{\mathbf{p}}_\mathbf{2} + \bar{\mathbf{p}}_\mathbf{1}\mathbf{p_2})\bar{\mathbf{p}}_\mathbf{3}$ $+\bar{\mathbf{p}}_\mathbf{1}\bar{\mathbf{p}}_\mathbf{2}\mathbf{p_3})\mathbf{p_4}$

Row 2 (bold) is each t_i's Global-Top2 probability. Now, if we are interested in a top-2 answer in R^p, we only need to pick the two tuples with the highest value in Row 2.

Theorem 3.3.1 (Correctness of Algorithm 1)**.** *Given a simple probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, a non-negative integer k and an injective scoring function s, Algorithm 1 correctly computes a Global-Top$k$ answer set of R^p under the scoring function s.*

Proof. Algorithm 1 maintains a priority queue to select the k tuples with the highest Global-Top$k$ value. Notice that the nondeterminism is reflected in Line 6 of Algorithm 1 as the maintenance of the priority queue in the presence of tying elements. As long as Line 2 of Algorithm 1 correctly computes the Global-Top$k$ probability of each tuple in R, Algorithm 1 returns a valid Global-Top$k$ answer set. By Proposition 3.3.1, Algorithm 2 correctly computes the Global-Top$k$ probability of tuples in R. □

Algorithm 1 is a one-pass computation over the probabilistic relation, which can be easily implemented even if secondary storage is used. The overhead is the initial sorting cost (not shown in Algorithm 1), which would be amortized by the workload of consecutive top-k queries.

Algorithm 1 (Ind_Topk) Evaluate Global-Top$k$ Queries in a Simple Probabilistic Relation under an Injective Scoring Function

```
Require: $R^p = \langle R, p, \mathcal{C} \rangle, k$
   (tuples in $R$ are sorted in the decreasing order based on the scoring function $s$)
1: Initialize a fixed cardinality $(k+1)$ priority queue $Ans$ of $\langle t, prob \rangle$ pairs, which compares pairs on $prob$, i.e., the Global-Top$k$ probability of $t$;
2: Calculate Global-Top$k$ probabilities using Algorithm 2, i.e.,
       $q(0 \ldots k, 1 \ldots |R|) = \text{Ind\_Topk\_Sub}(R^p, k);$
3: for $i = 1$ to $|R|$ do
4:     Add $\langle t_i, q(k,i) \rangle$ to $Ans$;
5:     if $|Ans| > k$ then
6:         remove a pair with the smallest $prob$ value from $Ans$;
7:     end if
8: end for
9: return $\{t_i | \langle t_i, q(k,i) \rangle \in Ans\}$;
```

Algorithm 2 takes $O(kn)$ to compute the dynamic programming (DP) table. In addition, Algorithm 1 uses a priority queue to maintain the k highest values, which takes $O(n \log k)$. Altogether, Algorithm 1 takes $O(kn)$ time.

Algorithm 2 (Ind_Topk_Sub) Compute Global-Top$k$ Probabilities in a Simple Probabilistic Relation under an Injective Scoring Function

```
Require: $R^p = \langle R, p, \mathcal{C} \rangle, k$
    (tuples in $R$ are sorted in the decreasing order based on the scoring function $s$)
 1: $q(0,1) = 0$;
 2: for $k' = 1$ to $k$ do
 3:     $q(k',1) = p(t_1)$;
 4: end for
 5: for $i = 2$ to $|R|$ do
 6:     for $k' = 0$ to $k$ do
 7:         if $k' = 0$ then
 8:             $q(k',i) = 0$;
 9:         else
10:             $q(k',i) = p(t_i)(q(k',i-1)\frac{\bar{p}(t_{i-1})}{p(t_{i-1})} + q(k'-1,i-1))$;
11:         end if
12:     end for
13: end for
14: return $q(0 \ldots k, 1 \ldots |R|)$;
```

The major space use in Algorithm 1 is the bookkeeping of the DP table in Line 2 (Algorithm 2). A straightforward implementation of Algorithm 1 and Algorithm 2 takes $O(kn)$ space. However, notice that in Algorithm 2, the column $q(0 \ldots k, i)$ depends on the column $q(0 \ldots k, i-1)$ only, and only the kth value $q(k, i-1)$ in the column $q(0 \ldots k, i-1)$ will be used in updating the priority queue in Line 4 of Algorithm 1 later. Therefore, in practice, we can reduce the space complexity to $O(k)$ by moving the update of the priority queue in Algorithm 1 to Algorithm 2, and using a vector of size $k+1$ to keep track of the previous column in the DP table. To be more specific, in Algorithm 2, each time we finish computing the current column based on the previous column in the DP table, we update the priority queue with the kth value in the current column and then replace the previous column with the current column. For readability, we present here the original algorithms without this optimization for space.

3.3.2 Threshold Algorithm Optimization

Fagin et al. [25] proposes *Threshold Algorithm (TA)* for processing top-k queries in a middleware scenario. In a middleware system, an *object* has m attributes. For each attribute,

Algorithm 1^{TA} (TA_Ind_Topk)

(1) Go down T list, and fill in entries in the DP table. Specifically, for $\underline{t} = t_j$, compute the entries in the j^{th} column up to the k^{th} row. Add t_j to the top-k answer set Ans, if any of the following conditions holds:

   (a) Ans has less than k tuples, i.e., $|Ans| < k$;

   (b) The Global-Top$k$ probability of t_j, i.e., $q(k, j)$, is greater than the lower bound of Ans, i.e., LB_{Ans}, where $LB_{Ans} = \min_{t_i \in Ans} q(k, i)$.

   In the second case, we also need to drop a tuple with the lowest Global-Top$k$ probability in order to preserve the cardinality of Ans.

(2) After we have seen at least k tuples in T, we go down P list to find the first p whose tuple t has not been seen. Let $\underline{p} = p$, and we can use $\underline{p}$ to estimate the *threshold*, i.e., upper bound (UP) of the Global-Top$k$ probability of any unseen tuple. Assume $\underline{t} = t_i$,

$$UP = (q(k, i)\frac{\bar{p}(t_i)}{p(t_i)} + q(k-1, i))\underline{p}.$$

(3) If $UP > LB_{Ans}$, Ans might be updated in the future, so go back to (1). Otherwise, we can safely stop and report Ans.

there is a sorted list ranking objects in the decreasing order of its score on that attribute. An *aggregation function* f combines the individual attribute scores x_i, i=$1, 2, \ldots, m$ to obtain the overall object score $f(x_1, x_2, \ldots, x_m)$. An aggregation function is *monotonic* iff $f(x_1, x_2, \ldots, x_m) \leqslant f(x'_1, x'_2, \ldots, x'_m)$ whenever $x_i \leqslant x'_i$ for every i. Fagin et al. [25] shows that TA is optimal in finding the top-k objects in such a system.

Denote T and P for the list of tuples in the decreasing order of score and probability, respectively. Following the convention in [25], $\underline{t}$ and $\underline{p}$ are the last value seen in T and P, respectively.

Theorem 3.3.2 (Correctness of Algorithm 1^{TA}). *Given a simple probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, a non-negative integer k and an injective scoring function s over R^p, Algorithm 1^{TA} correctly finds a Global-Top$k$ answer set.*

Proof. See Appendix B.

The optimization above aims at an early stop. Bruno et al. [10] carries out an extensive experimental study on the effectiveness of applying TA in RDMBS. They consider various aspects of query processing. One of their conclusions is that if at least one of the indices available for the attributes[2] is a *covering index*, that is, it is defined over all other attributes and we can get the values of all other attributes directly without performing a primary index lookup, then the improvement by TA can be up to two orders of magnitude. The cost of building a useful set of indices once would be amortized by a large number of top-k queries that subsequently benefit form such indices. Even in the lack of covering indices, if the data is highly correlated, in our case, that means high-score tuples having high probabilities, TA would still be effective.

TA is guaranteed to work as long as the aggregation function is monotonic. For a simple probabilistic relation, if we regard *score* and *probability* as two special attributes, Global-Top$k$ probability $P_{k,s}$ is an aggregation function of *score* and *probability*. The closeness between *Faithfulness* and the monotonicity of Global-Top$k$ probability makes the satisfaction of *Faithfulness* a strong indication of a possible TA optimization. Theorem 3.3.2 confirms that TA is applicable to the computation of Global-Top$k$. Consequently, assuming that we have an index on probability as well, we can guide the dynamic programming (DP) in Algorithm 2 by TA. Now, instead of computing all kn entries for DP, where $n = |R|$, the algorithm can be stopped as early as possible. A subtlety is that Global-Top$k$ probability $P_{k,s}$ is *only* well-defined for $t \in R$, unlike in [25], where an aggregation function is well-defined over the domain of all possible attribute values. Therefore, compared to the original TA, we need to achieve the same behavior without referring to virtual tuples which are not in R.

U-Top$k$ satisfies *Faithfulness* in simple probabilistic relations. An adaptation of the TA algorithm in this case is available in [53]. TA is not applicable to U-kRanks. Even though we can define an aggregation function per $rank$, $rank = 1, 2, \ldots, k$, for tuples under U-kRanks, the violation of *Faithfulness* (cf. Appendix A (8)) suggests a violation of monotonicity of those k aggregation functions. PT-k computes Global-Top$k$ probabilities as well, and is therefore a natural candidate for TA in simple probabilistic relations.

[2]Probability is typically supported as a special attribute in DBMS.

3.3.3 Arbitrary Probabilistic Relations

Induced Event Relation

In the general case of probabilistic relations (Definition 3.1.1), each part of the partition $\mathcal{C}$ can contain more than one tuple. The crucial *independence* assumption in Algorithm 1 no longer holds. However, even though tuples from one part of the partition $\mathcal{C}$ are not independent, tuples from different parts are. In the following definition, we assume an identifier function id. For any tuple t, $id(t)$ identifies the part where t belongs. We show how to reduce the arbitrary case to the simple case using the notion of *induced event relation*.

Definition 3.3.1 (Induced Event Relation)**.** *Given a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, an injective scoring function s over R^p and a tuple $t \in C_{id(t)} \in \mathcal{C}$, the event relation induced by t, denoted by $E^p = \langle E, p^E, \mathcal{C}^E\rangle$, is a probabilistic relation whose support relation E has only one attribute, $Event$. The relation E and the probability function p^E are defined by the following two generation rules:*

- *Rule 1:* $t_{e_t} \in E$ *and* $p^E(t_{e_t}) = p(t)$;
- *Rule 2:* $\forall C_i \in \mathcal{C} \wedge C_i \neq C_{id(t)}$.

$$(\exists t' \in C_i \wedge t' >_s t) \Rightarrow (t_{e_{C_i}} \in E) \text{ and } p^E(t_{e_{C_i}}) = \sum_{\substack{t' \in C_i \\ t' >_s t}} p(t').$$

No other tuples belong to E. The partition $\mathcal{C}^E$ is defined as the collection of singleton subsets of E. An induced event relation is a simple probabilistic relation.

Except for one special tuple generated by *Rule 1*, each tuple in the induced event relation (generated by *Rule* 2) represents an event e_{C_i} associated with a part $C_i \in \mathcal{C}$. Given the tuple t, the *event* e_{C_i} is defined as "there is a tuple from the part C_i with a score higher than that of t". The probability of this event, denoted by $p(t_{e_{C_i}})$, is the probability that e_{C_i} occurs.

The role of the special tuple t_{e_t} and its probability $p(t)$ will become clear in Proposition 3.3.2. Let us first look at an example of an induced event relation.

Example 8. *Given R^p as in Example 2, we would like to construct the induced event relation $E^p = \langle E, p^E, \mathcal{C}^E \rangle$ for tuple t=(Temp: 15) from C_2. By Rule 1, we have $t_{e_t} \in E$, $p^E(t_{e_t}) = 0.6$. By Rule 2, since $t \in C_2$, we have $t_{e_{C_1}} \in E$ and $p^E(t_{e_{C_1}}) = \sum_{\substack{t' \in C_1 \\ t' >_s t}} p(t') = p((\text{Temp: } 22)) = 0.6$. Therefore,*

Event (E)	Prob (p^E)
t_{e_t}	0.6
$t_{e_{C_1}}$	0.6

Evaluating Global-Top$k$ Queries

With the help of *induced event relations*, we can reduce Global-Top$k$ in the general case to Global-Top$k$ in simple probabilistic relations.

Proposition 3.3.2. *Given a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$ and an injective scoring function s, for any $t \in R^p$, the Global-Top$k$ probability of t equals the Global-Top$k$ probability of t_{e_t} when evaluating top-k in the induced event relation $E^p = \langle E, p^E, \mathcal{C}^E \rangle$ under the injective scoring function $s^E : E \to \mathbb{R}$, $s^E(t_{e_t}) = \frac{1}{2}$ and $s^E(t_{e_{C_i}}) = i$:*

$$P_{k,s}^{R^p}(t) = P_{k,s^E}^{E^p}(t_{e_t}).$$

Proof. See Appendix B.

In Proposition 3.3.2, the choice of the function s^E is rather arbitrary. In fact, any injective function giving t_{e_t} the lowest score will do. Every tuple other than t_{e_t} in the induced event relation corresponds to an event that a tuple with a score higher than that of t occurs. We want to track the case that at most $k-1$ such events happen. Since any induced event relation is simple (Definition 3.3.1), Proposition 3.3.2 illustrates how we can reduce the computation of $P_{k,s}^{R^p}(t)$ in the original probabilistic relation to a top-k computation in a simple probabilistic relation, where we can apply the DP technique described in Section 3.3.1. The complete algorithms are shown as Algorithm 3 and Algorithm 4.

In Algorithm 4, we first find the part $C_{id(t)}$ where t belongs. In Line 2, we initialize the support relation E of the induced event relation with the tuple generated by Rule 1 in Definition 3.3.1. For any part C_i other than $C_{id(t)}$, we compute the probability of the event

Algorithm 3 (IndEx_Topk) Evaluate Global-Top$k$ Queries in a General Probabilistic Relation under an Injective Scoring Function

```
Require: $R^p = \langle R, p, \mathcal{C} \rangle, k, s$
    (tuples in $R$ are sorted in the decreasing order based on the scoring function $s$)
 1: Initialize a fixed cardinality $k+1$ priority queue $Ans$ of $\langle t, prob \rangle$ pairs, which compares
    pairs on $prob$, i.e., the Global-Top$k$ probability of $t$;
 2: for $t \in R$ do
 3:   Calculate $P^{R^p}_{k,s}(t)$ using Algorithm 4, i.e.,
          $$P^{R^p}_{k,s}(t) = \text{IndEx\_Topk\_Sub}(R^p, k, s, t);$$
 4:   Add $\langle t, P^{R^p}_{k,s}(t) \rangle$ to $Ans$;
 5:   if $|Ans| > k$ then
 6:     remove a pair with the smallest $prob$ value from $Ans$;
 7:   end if
 8: end for
 9: return $\{t | \langle t, P^{R^p}_{k,s}(t) \rangle \in Ans\}$;
```

Algorithm 4 (IndEx_Topk_Sub) Calculate $P^{R^p}_{k,s}(t)$ using an induced event relation

```
Require: $R^p = \langle R, p, \mathcal{C} \rangle, k, s, t \in R$
    (tuples in $R$ are sorted in the decreasing order based on the scoring function $s$)
 1: Find the part $C_{id(t)} \in \mathcal{C}$ such that $t \in C_{id(t)}$;
 2: $E = \{t_{e_t}\}$, where $p^E(t_{e_t}) = p(t)$;
 3: for $C_i \in \mathcal{C}$ and $C_i \neq C_{id(t)}$ do
 4:          $$p(e_{C_i}) = \sum_{\substack{t' \in C_i \\ t' >_s t}} p(t');$$
 5:   if $p(e_{C_i}) > 0$ then
 6:     $E = E \cup \{t_{e_{C_i}}\}$, where $p^E(t_{e_{C_i}}) = p(e_{C_i})$;
 7:   end if
 8: end for
 9: Use Algorithm 2 to compute Global-Top$k$ probabilities in $E^p = \langle E, p^E, \mathcal{C}^E \rangle$ under the
    scoring function $s^E$, i.e.,
          $$q(0 \ldots k, 1 \ldots |E|) = \text{Ind\_Topk\_Sub}(E^p, k)$$
10: $P^{R^p}_{k,s}(t) = P^{E^p}_{k,s^E}(t_{e_t}) = q(k, |E|)$;
11: return $P^{R^p}_{k,s}(t)$;
```

e_{C_i} according to Definition 3.3.1 (Line 4), and add it to E if its probability is non-zero (Lines 5-7). Since all tuples from the same part are exclusive, this probability is the sum of the probabilities of all qualifying tuples in that part. If no tuple from C_i qualifies, this probability is zero. In this case, we do not care whether any tuple from C_i will be in the possible world or not, since it does not have any influence on whether t will be in top-k or not. The corresponding event tuple is therefore excluded from E. Note that, by default, any probabilistic database assumes that any tuple not in the support relation has probability zero. Line 9 uses Algorithm 2 to compute $P^{E^p}_{k,s^E}(t_{e_t})$. Note that Algorithm 2 requires all tuples be sorted on score. Since we already know the scoring function s^E, we simply need to organize tuples based on s^E when generating E. No extra sorting is necessary.

Theorem 3.3.3 (Correctness of Algorithm 3)**.** *Given a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, a non-negative integer k and an injective scoring function s, Algorithm 3 correctly computes a Global-Top$k$ answer set of R^p under the scoring function s.*

Proof. The top-level structure of Algorithm 3 resembles that of Algorithm 1. Therefore, as long as Line 3 in Algorithm 3 correctly computes the Global-Top$k$ probability of each tuple in R, Algorithm 3 returns a valid Global-Top$k$ answer set. Lines 1-8 in Algorithm 4 compute the event relation induced by the tuple t. By Proposition 3.3.2, Lines 9-10 in Algorithm 4 correctly compute the Global-Top$k$ probability of t. $\square$

In Algorithm 4, Lines 3-8 take $O(n)$ time to build E (we need to scan all tuples within each part). The call to Algorithm 2 in Line 9 takes $O(k|E|)$, where $|E|$ is no more than the number of parts in partition $\mathcal{C}$, which is in turn no more than n. So Algorithm 4 takes $O(kn)$. Algorithm 3 make n calls to Algorithm 4 to compute $P^{R^p}_{k,s}(t)$ for every tuple $t \in R$. Again, Algorithm 3 uses a priority queue to select the final answer set, which takes $O(n \log k)$. The entire algorithm takes $O(kn^2 + n \log k) = O(kn^2)$.

A straightforward implementation of Algorithm 3 and Algorithm 4 take $O(kn)$ space, as the call to Algorithm 2 in Algorithm 4 could take up to $O(k|E|)$ space. However, by using a spatially optimized version of Algorithm 2 mentioned in Section 3.3.1, this DP table computation in Algorithm 4 can be completed in $O(k)$ space. Algorithm 4 still needs

$O(|E|)$ space to store the induced event relation computed between Lines 3-8. Since $|E|$ has an upper bound n, the total space is therefore $O(k + n)$.

3.3.4 Optimizations for Arbitrary Probabilistic Relations

In the previous section, we presented the basic algorithms to compute Global-Top$k$ probabilities in general probabilistic relations. In this section, we provide two heuristics, *Rollback* and *RollbackSort*, to speed up this computation. Our optimizations are similar to *prefix sharing* optimizations in [32], although the assumptions and technical details are different. In our terminology, the *aggressive* and *lazy* prefix sharing in [32] assume the ability to "look ahead" in the input tuple stream to locate the next tuple belonging to every part. In contrast, *Rollback* assumes no extra information, and *RollbackSort* assumes the availability of aggregate statistics on tuples.

Rollback and *RollbackSort* take advantage of the following two facts in the basic algorithms:

Fact 1: The overlap of the event relations induced by the consecutive tuples in the original relation;

Rollback and *RollbackSort* are based on the following "incremental" computation of induced event relations for tuples in R. By Definition 3.3.1, for any tuple $t \in R$, only tuples with a higher score will have an influence on t's induced event relation. Given a scoring function s, consider two adjacent tuples t_i, t_{i+1} in the decreasing order of scores, which is the processing order in Algorithm 3. Denote by E_i and E_{i+1} their induced event relations under the function s, respectively.

Case 1: t_i and t_{i+1} are exclusive.

Then t_i and t_{i+1} have the same induced event relation except for the one tuple generated by Rule 1 in each induced event relation.

$$E_i - \{t_{e_{t_i}}\} = E_{i+1} - \{t_{e_{t_{i+1}}}\}. \tag{3.4}$$

Case 2: t_i and t_{i+1} are independent, and t_{i+1} is independent of $t_1, \ldots, t_{i-1}$ as well.

Recall that any tuple t_j exclusive to t_i, even though it has a score higher than that of t_i, i.e., $1 \leqslant j \leqslant i-1$, does not contribute to E_i due to the existence of $t_{e_{t_i}}$ in E_i. However, tuple t_{i+1} is independent of such tuple t_j, and therefore both t_i and such tuple t_j contribute to t_{i+1}'s induced event relation. In E_{i+1}, instead of $t_{e_{t_i}}$, there is an event tuple $t_{e_{C_{id(t_i)}}}$, which corresponds to the event that one tuple from $C_{id(t_i)}$ appears. The fact that t_{i+1} is independent of $t_1, \dots, t_{i-1}$ guarantees that there is no tuple in $E_i - \{t_{e_{t_i}}\}$ incompatible with the event tuple $t_{e_{t_{i+1}}}$ generated by Rule 1 in E_{i+1}. Therefore, all event tuples in $E_i - \{t_{e_{t_i}}\}$ should be retained in E_{i+1}. Consequently,

$$E_i - \{t_{e_{t_i}}\} = E_{i+1} - \{t_{e_{C_{id(t_i)}}}, t_{e_{t_{i+1}}}\}. \tag{3.5}$$

Case 3: t_i and t_{i+1} are independent, and t_{i+1} is incompatible with at least one tuple from $t_1, \dots, t_{i-1}$.

In this case, like in Case 2, the first part of the condition guarantees the existence of $t_{e_{C_{id(t_i)}}}$ in E_{i+1}. However, the second part of the condition essentially states that some tuple from $C_{id(t_{i+1})}$ has a score higher than that of t_i. Thus, there is an event tuple $t_{e_{C_{id(t_{i+1})}}}$ in E_i, which is incompatible with $t_{e_{t_{i+1}}}$ generated by Rule 1 in E_{i+1}, and therefore should not appear in E_{i+1}. As a result, besides the one tuple generated by Rule 1 in each induced event relation, E_{i+1} and E_i also differ in the event tuple $t_{e_{C_{id(t_i)}}}$ and $t_{e_{C_{id(t_{i+1})}}}$.

$$E_i - \{t_{e_{C_{id(t_{i+1})}}}, t_{e_{t_i}}\} = E_{i+1} - \{t_{e_{C_{id(t_i)}}}, t_{e_{t_{i+1}}}\}. \tag{3.6}$$

Fact 2: The arbitrary choice of the scoring function s^E in Proposition 3.3.2.

Recall that in Proposition 3.3.2, the event tuple t_{e_t} has the same Global-Top$k$ probability in the induced event relation under any scoring functions s^E giving t_{e_t} the lowest score.

Rollback

In *Rollback*, we use an annotated $(k+1) \times n$ table T^a to support two major operations for each induced event relation: (1) the creation of the induced event relation, and (2) the computation in the dynamic programming (DP) table to calculate the Global-Top$k$ probability of the tuple inducing it. T^a is analogous to the DP table that we have been using so far. In addition, each column in T^a is annotated with $(part\_id, prob)$ of an event tuple in the current induced event relation.

By *Fact 1*, it is clear that the creation of induced event relations is incremental if we do it for tuples in the processing order in Algorithm 3, i.e., the decreasing order of scores. *Rollback* exploits this order and piggybacks the creation of the induced event relation to the computation in the DP table.

By *Fact 2*, we can reuse the scoring function to the greatest extent between two consecutive induced event relations, and therefore avoid the recomputation of a part of the DP table.

Without loss of generality, assume $t_1 \succ t_2 \succ \ldots \succ t_n$, and the tuple just *processed* is t_i, $1 \leqslant i \leqslant n$. By "processed", we mean that there is a DP table for computing the Global-Top$k$ probability, denoted by DP_i, where each column is associated with an event tuple in t_i's induced event relation E_i. Assume $|E_i| = l_i$, then there are l_i columns in DP_i. Obviously, $l_i \leqslant i$ since only $t_1, t_2, \ldots, t_i$ can contribute to E_i. In fact, $l_i = i$ when all i tuples are independent. In this case, each tuple corresponds to a distinct event tuple in E_i. When there are exclusive tuples in E_i, $l_i < i$. In this case, if a tuple from $t_1, t_2, \ldots, t_{i-1}$ is incompatible with t_i, it is ignored due to the existence of $t_{e_{t_i}}$ in E_i. For other exclusive tuples, the tuples from the same part collapse into a single event tuple in E_i. Moreover, the probability of such event tuple is the sum of the probabilities of all exclusive tuples contributing to it.

Now, consider the next tuple to be processed, t_{i+1}, its induced event relation E_{i+1}, and the DP table DP_{i+1} to compute the Global-Top$k$ probability in E_{i+1}. If the current situation is of Case 1, then E_i and E_{i+1} only differ in the event tuple generated by Rule 1. Recall that the only requirement on the scoring function used in an induced event relation is to assign the lowest score to the event tuple generated by Rule 1. This requirement is translated into associating the last column in the DP table with the tuple generated by Rule 1. Therefore, we can take the first $l_i - 1$ columns from DP_i and reuse them in DP_{i+1}. In other words, by reusing the scoring function in DP_i as much as possible based on *Fact 2*, the resulting DP_{i+1} table differs from DP_i only in the last column. In practice, DP_{i+1} is computed incrementally by modifying the last column of DP_i in place. In other word, we have the content of DP_i stored in the data structure T^a used in *Rollback*. By modifying the last column of T^a and changing the annotation, we can have DP_{i+1} in T^a. Denoted by col_{cur} the current last column in DP_i. In DP_{i+1}, col_{cur} should be reassociated with the event tuple $t_{e_{t_{i+1}}}$, i.e.,

$$col_{cur}.part\_id = id(t_{i+1}),$$
$$col_{cur}.prob = p(t_{i+1}).$$

It is easy to see that the incremental computation cost is the cost of computing $k+1$ entries in col_{cur}.

Similarly for Case 2, the first $l_i - 1$ columns in DP_i can be reused. The two new event tuples in DP_{i+1} are $t_{e_{C_{id(t_i)}}}$ and $t_{e_{t_{i+1}}}$. To compute DP_{i+1}, we need to change the association of two columns, col_{cur} and col_{cur+1}. The last column in DP_i (col_{cur}) is reassociated with $t_{e_{C_{id(t_i)}}}$:

$$col_{cur}.part\_id = id(t_i),$$
$$col_{cur}.prob = \sum_{\substack{t_{j''} \in C_{id(t_i)} \\ 1 \leqslant j'' \leqslant i}} p(t_{j''}).$$

The last column in DP_{i+1} (col_{cur+1}) is associated with $t_{e_{t_{i+1}}}$:

$$col_{cur+1}.part\_id = id(t_{i+1}),$$
$$col_{cur+1}.prob = p(t_{i+1}).$$

Example 9. *Consider the following data*[3]*, and a top-*2 *query.*

	Part	*Score*	*Prob.*
t_1	C_1	0.9	0.3
t_2	C_2	0.8	0.1
t_3	C_3	0.7	0.2
t_4	C_1	0.6	0.4
t_5	C_2	0.5	0.7

Tuples are processed in the decreasing order of their scores, i.e., $t_1, t_2, \ldots, t_5$*. Figure 3-1 illustrates each* DP_i *table, i.e., the content of* T^a*, after the processing of tuple* t_i*. The annotation* $(part\_id, prob)$ *of each column is also illustrated. The entry in bold is the Global-Top*k *probability of the corresponding tuple inducing the event relation.*

Take the processing of t_3 *for example. Since* t_3 *is independent of* t_2 *and* t_1*, this is Case 2. Therefore, the last column in* DP_2 *(*col_2*) needs to be reassociated with* $t_{e_{C_{id(t_2)}}} = t_{e_{C_2}}$ *in* E_3*. In* DP_3*,*

$$col_2.part\_id = id(t_2) = 2,$$
$$col_2.prob = \sum_{\substack{t_{j''} \in C_2 \\ 1 \leqslant j'' \leqslant 2}} p(t_{j''}) = p(t_2) = 0.1.$$

The last column in DP_3 *(*col_3*) is associated with the event tuple* $t_{e_{t_3}}$ *generated by Rule 1 in* E_3*:*

$$col_3.part\_id = id(t_3) = 3,$$
$$col_3.prob = p(t_3) = 0.2.$$

Compared to DP_2*, the first column with an annotation change in* DP_3 *is* col_3*. The DP table needs to be recomputed from* col_3 *(inclusive) upwards. In this case, it is only* col_3*. Notice that, even though the annotation of* col_2 *does not change from* DP_2 *to* DP_3*, its meaning*

[3]We explicitly include partition information into the representation, and thus the horizontal lines do not represent partition here.

col / k	col_1 $(1, 0.3)$
0	0
1	0.3
2	**0.3**

(a) DP_1

col / k	col_1 $(1, 0.3)$	col_2 $(2, 0.1)$
0	0	0
1	0.3	0.07
2	0.3	**0.1**

(b) DP_2

col / k	col_1 $(1, 0.3)$	col_2 $(2, 0.1)$	col_3 $(3, 0.2)$
0	0	0	0
1	0.3	0.07	0.126
2	0.3	0.1	**0.194**

(c) DP_3

col / k	col_1 $(2, 0.1)$	col_2 $(3, 0.2)$	col_3 $(1, 0.4)$
0	0	0	0
1	0.1	0.18	0.288
2	0.1	0.2	**0.392**

(d) DP_4

col / k	col_1 $(3, 0.2)$	col_2 $(1, 0.7)$	col_3 $(2, 0.7)$
0	0	0	0
1	0.2	0.56	0.168
2	0.2	0.7	**0.602**

(e) DP_5

Figure 3-1: DP table evolution in *Rollback*

changes: in DP_2, col_2 is associated with $t_{e_{t_2}}$ in E_2; in DP_3, col_2 is associated with $t_{e_{C_{id(t_2)}}}$ in E_3 instead.

In Case 1 and Case 2, the event tuple which we want to "erase" from E_i, i.e., $t_{e_{t_i}}$, is associated with the last column in DP_i. In Case 3, by Equation (3.6), we want to "erase" from E_i the event tuple $t_{e_{C_{id(t_{i+1})}}}$ in addition to $t_{e_{t_i}}$. Assume $t_{e_{C_{id(t_{i+1})}}}$ is associated with col_j in DP_i, and the columns in DP_i are

$$col_1, \ldots, col_{j-1}, col_j, col_{j+1}, \ldots, col_{cur-1}, col_{cur}$$

which correspond to

$$t_{e_{C_{i_1}}}, \ldots, t_{e_{C_{i_{j-1}}}}, t_{e_{C_{i_j}}}, t_{e_{C_{i_{j+1}}}}, \ldots, t_{e_{C_{i_{cur-1}}}}, t_{e_{t_i}}$$

in E_i, respectively. Obviously, $i_j = id(t_{i+1})$. By Equation (3.6),

$$E_{i+1} = \{t_{e_{C_{i_1}}}, \ldots, t_{e_{C_{i_{j-1}}}}, t_{e_{C_{i_{j+1}}}}, \ldots, t_{e_{C_{i_{cur-1}}}}, t_{e_{C_{id(t_i)}}}, t_{e_{t_{i+1}}}\}.$$

By *Fact 2*, as long as $t_{e_{t_{i+1}}}$ is associated with the last column in DP_{i+1}, the column association order of other tuples in E_{i+1} does not matter in computing the Global-Top$k$ probability of t_{i+1}. By adopting a column association order such that

$$t_{e_{C_{i_1}}}, \ldots, t_{e_{C_{i_{j-1}}}}$$

are associated with

$$col_1, \ldots, col_{j-1}$$

respectively in DP_{i+1}, we can reuse the first $j-1$ columns already computed in DP_i. Recall that in our DP computation, the values in a column depend on the values in its previous column. Once we change the values in col_j, every $col_{j'}$, $j' > j$, needs to be recomputed regardless. Therefore, the recomputation cost is the same for every column association order of event tuples

$$t_{e_{C_{i_{j+1}}}}, \ldots, t_{e_{C_{i_{cur-1}}}}, t_{e_{C_{id(t_i)}}}$$

In *Rollback*, we simply use this order above as the column association order. In fact, the name of this optimization, *Rollback*, refers to the fact that we are "rolling back" the computation in the DP auxiliary data structure T^a until we hit col_j and recompute all the columns with an index equal to or higher than j.

Example 10. *Continuing Example 9, consider the processing of* t_5. t_5 *is independent of* t_4*, while* t_5 *and* t_2 *are exclusive. Therefore, this is Case 3. We first locate* col_j *associated with* $t_{e_{C_{id(t_5)}}} = t_{e_{C_2}}$ *in* DP_4*. In this case, it is* col_1*. Then, we roll all the way back to* col_1 *in* DP_4*, erasing every column on the way including* col_1*. Since* $col_j = col_1$*, there is no column from* DP_4 *that we can reuse in* DP_5*. We move on to recompute* $col_{j'}$*,* $j \leqslant j'$*, in* DP_5 *that are associated with* $t_{e_{C_3}}$ *and* $t_{e_{C_{id(t_4)}}} = t_{e_{C_1}}$*. In particular,* col_2 *in* DP_5 *is associated with* $t_{e_{C_1}}$*. Thus,*

$$\begin{aligned} col_2.part\_id &= 1, \\ col_2.prob &= \sum_{t_{j''} \in C_1, 1 \leqslant j'' \leqslant 4} p(t_{j''}) \\ &= p(t_1) + p(t_4) \\ &= 0.3 + 0.4 \\ &= 0.7. \end{aligned}$$

The last column in DP_5 *is again associated with the event tuple* $t_{e_{t_5}}$ *generated by Rule 1 in* E_5*.*

Out of the five tuples, the processing of t_1, t_2, t_3 *is of Case 2, and the processing of* t_4, t_5 *is of Case 3. Whenever we compute/recompute the DP table, the event tuples associated with the columns are from the induced event relation, and therefore independent. Thus, every DP table computation progresses in the same fashion as that with the DP table in Example 7.*

Finally, we keep the Global-Top2 *probability of each tuple (from the original probabilistic relation) in a priority queue. When we finish processing all the tuples, we get the top-*2 *winners. In this example, the priority queue is updated every time we get an entry in bold. The winners are* t_5 *and* t_4 *with the Global-Top*2 *probability* 0.602 *and* 0.392*, respectively.*

RollbackSort

For the rollback operation in Case 3 of *Rollback*, define its *depth* as the number of columns recomputed in rolling back excluding the last column. For example, when processing t_5 in Example 10, col_1, col_2 and col_3 are recomputed in DP_5. Therefore the *depth* of this rollback operation is $3 - 1 = 2$.

Recall that in Case 3 of *Rollback*, we adopt an arbitrary order

$$t_{e_{C_{i_{j+1}}}}, \ldots, t_{e_{C_{i_{cur-1}}}}, t_{e_{C_{id(t_i)}}}$$

to process those event tuples in DP_{i+1}. The Global-Top$k$ computation in E_{i+1} does not stipulate any particular order over those tuples. Any permutation of this order is equally valid. The intuition behind *RollbackSort* is that we will be able to find a permutation that will reduce the *depth* of future rollback operations (if any), given additional statistics on the probabilistic relation R^p, namely the count of the tuples in each part of the partition. Theoretically, it requires an extra pass over the relation to compute the statistics. In practice, however, this extra pass is often not needed because this statistics can be precomputed and stored.

In *RollbackSort*, if the current situation is Case 3, we do a stable sort on

$$t_{e_{C_{i_{j+1}}}}, \ldots, t_{e_{C_{i_{cur-1}}}}, t_{e_{C_{id(t_i)}}}$$

in the non-decreasing order of the number of unseen tuples in its corresponding part, and then use the resulting order to process those event tuples. The intuition is that each unseen tuple has the potential to trigger a rollback operation. By pushing the event tuple with the most unseen tuples close to the end of the current DP table, we could reduce the depth of future rollback operations. In order to facilitate this sorting, we add one more component $unseen$ to the annotation of each column in T^a.

Example 11. *We redo the problem in Example 9 and Example 10 using* RollbackSort. *Now, the annotation of each column becomes* $(part\_id, prob, unseen)$. *The evolution of the DP table is shown in Figure 3-2. The statistics on all parts are:* 2 *tuples in* C_1*,* 2 *tuples in* C_2 *and* 1 *tuple in* C_3*.*
Consider the processing of t_1 *in* DP_1*. As we just see one tuple* t_1 *from* C_1*, there is one more unseen tuple from* C_1 *coming in the future. Therefore,* $col_1.unseen = 1$*. All the other* $unseen$ *annotations are computed in the same way.*

When processing t_4 *(Case 3), the column associated with* $t_{e_{C_{id(t_4)}}}$ *in* DP_3 *is* col_1*. We roll back to* col_1 *as before and recompute all the columns upwards in* DP_4*. Notice that, the recomputation is performed in the order* $t_{e_{C_3}}$*,* $t_{e_{C_2}}$*, in contrast to the order* $t_{e_{C_2}}$*,* $t_{e_{C_3}}$ *used in Example 10 (Figure 3-1(d)).* C_2 *has one more unseen tuple which can trigger the rollback operation while there are no more unseen tuples from* C_3*. The benefit of this order becomes clear when we process* t_5*. Now, we only need to rollback to* col_2 *in* DP_4*. The* $depth$ *of this rollback operation is* 1*. Recall that the* $depth$ *of the same rollback operation is* 2 *in Example 10. In other words, we save the computation of* 1 *column by applying* RollbackSort.

Rollback and *RollbackSort* significantly improve the performance in practice, as we will see in Section 3.4. The price we pay for this speedup is an increase in the space usage. The space complexity is $O(kn)$ for both optimization. The quadratic theoretic bound on running time remains unchanged.

3.4 Experiments

We report here an empirical study on various optimization techniques proposed in Section 3.3.2 and Section 3.3.4, as the behavior of the straightforward implementation of our algorithms is pretty much predicted by the aforementioned theoretical analysis. We implement

col / k	col_1 $(1, 0.3, 1)$
0	0
1	0.3
2	**0.3**

(a) DP_1

col / k	col_1 $(1, 0.3, 1)$	col_2 $(2, 0.1, 1)$
0	0	0
1	0.3	0.07
2	0.3	**0.1**

(b) DP_2

col / k	col_1 $(1, 0.3, 1)$	col_2 $(2, 0.1, 1)$	col_3 $(3, 0.2, 0)$
0	0	0	0
1	0.3	0.07	0.126
2	0.3	0.1	**0.194**

(c) DP_3

col / k	col_1 $(3, 0.2, 0)$	col_2 $(2, 0.1, 1)$	col_3 $(1, 0.4, 0)$
0	0	0	0
1	0.2	0.08	0.288
2	0.2	0.1	**0.392**

(d) DP_4

col / k	col_1 $(3, 0.2, 0)$	col_2 $(1, 0.7, 0)$	col_3 $(2, 0.7, 0)$
0	0	0	0
1	0.2	0.56	0.168
2	0.2	0.7	**0.602**

(e) DP_5

Figure 3-2: DP table evolution in *RollbackSort*

all the algorithms in C++ and run experiments on a machine with Intel Core2 1.66G CPU running Cygwin on Windows XP with 2GB memory.

Each synthetic dataset has a uniform random score distribution and a uniform random probability distribution. There is no correlation between the score and the probability. The size (n) of the dataset varies from 5K up to 1M. In a dataset of a general probabilistic relation, x is the percentage of exclusive tuples and s is the max number of exclusive tuples in a part from the partition. In other words, in a general probabilistic relation of size n, there are $\lceil nx \rceil$ tuples involved in a non-trivial part from the partition. The size of each part is a random number from $[2, s]$. Unless otherwise stated, x defaults to 0.1 and s defaults to 20. The default value of k in a top-k query is 100.

For simple relations, the baseline algorithm *Basic* is the space optimized version of Algorithm 1 and 2 mentioned in Section 3.3.1. *TA* integrates the TA optimization technique in Section 3.3.2. For general relations, the baseline algorithm *Reduction* is a straightforward implementation of Algorithm 3 and 4. *Rollback* and *RollbackSort* implements the two optimization techniques in Section 3.3.4, respectively.

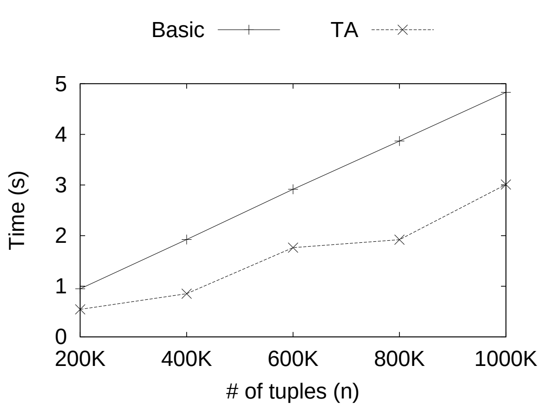
Summary of experiments

We draw the following conclusions from the forthcoming experimental results:

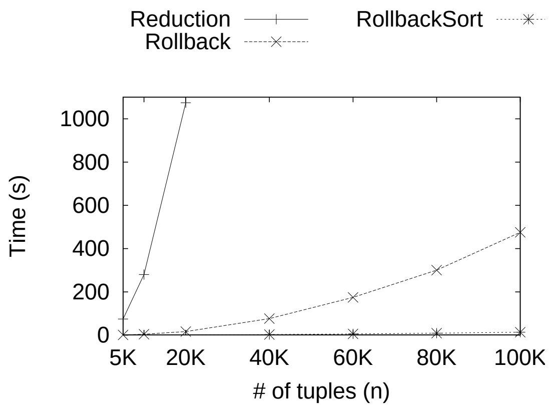
- Optimizations such as *TA*, *Rollback* and *RollbackSort* are effective and significantly reduce the running time. On average, *TA* saves about half of the computation cost in simple relations. Compared to *Reduction*, *Rollback* and *RollbackSort* improve the running time up to 2 and 3 orders of magnitude, respectively.

- Decreasing the percentage of exclusive tuples (x) improves the running time of *Rollback* and *RollbackSort*. When x is fixed, increasing the max number of tuples in each part (s) improves the running time of *Rollback* and *RollbackSort*.

- For general probabilistic relations, *RollbackSort* scales well to large datasets.

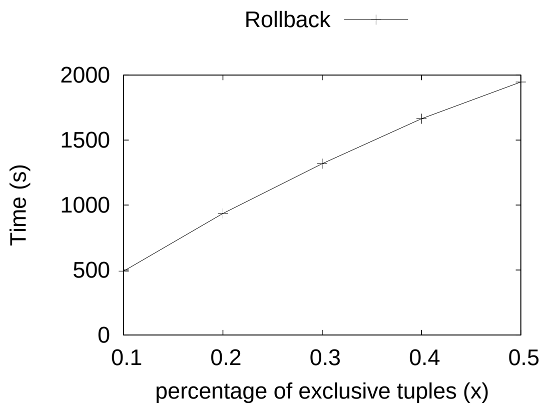


(a) Simple Prob. Relation

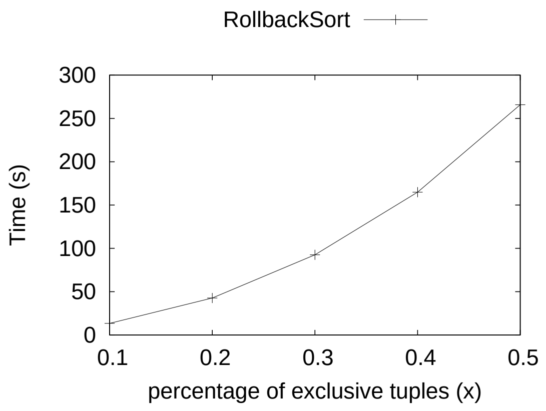


(b) General Prob. Relation

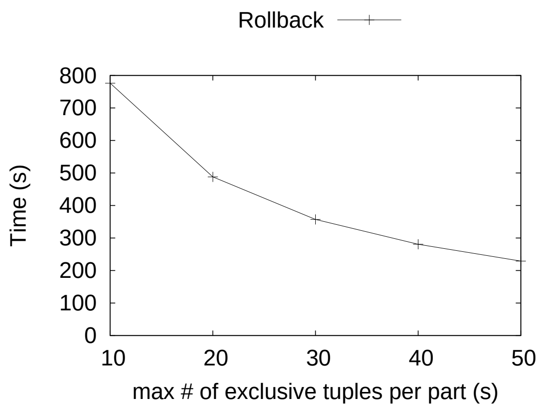
Figure 3-3: Performance of Optimizations



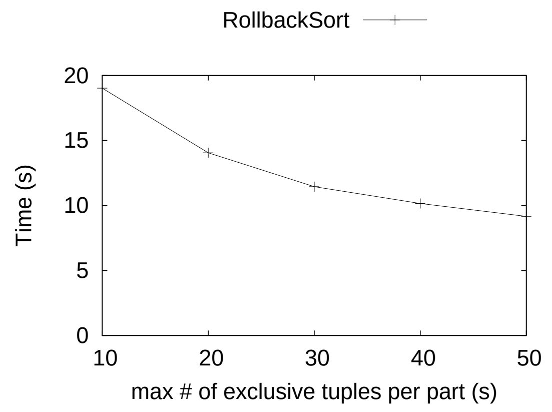
(a) Running time vs x (*Rollback*)



(b) Running time vs x (*RollbackSort*)



(c) Running time vs s (*Rollback*)



(d) Running time vs s (*RollbackSort*)

Figure 3-4: Sensitivity to Parameters

3.4.1 Performance of Optimizations

Figure 3-3(a) illustrates the improvement of *TA* over *Basic* for simple probabilistic relations. While *Basic* is already linear in terms of n, *TA* still saves a significant amount of computation, i.e., a little less than half. It worths emphasizing that there is no correlation between the score and the probability in our datasets. It is well-known that TA optimization has a better performance when there is a positive correlation between attributes, and a worse performance when there is a negative correlation between attributes. Therefore, the dataset we use, i.e., with no correlation, should represent an average case.

For general probabilistic relations, Figure 3-3(b) illustrates the performance of *Reduction*, *Rollback* and *RollbackSort* when n varies from 5K to 100K. For the baseline algorithm *Reduction*, we show only the first three data points, as the rest are off the chart. The curve of *Reduction* reflects the quadratic theoretical bound. From Figure 3-3(b), it is clear that the heuristic *Rollback* and *RollbackSort* greatly reduce the running time over the quadratic bound. The improvement is up to 2 and 3 orders of magnitude for *Rollback* and *RollbackSort*, respectively.

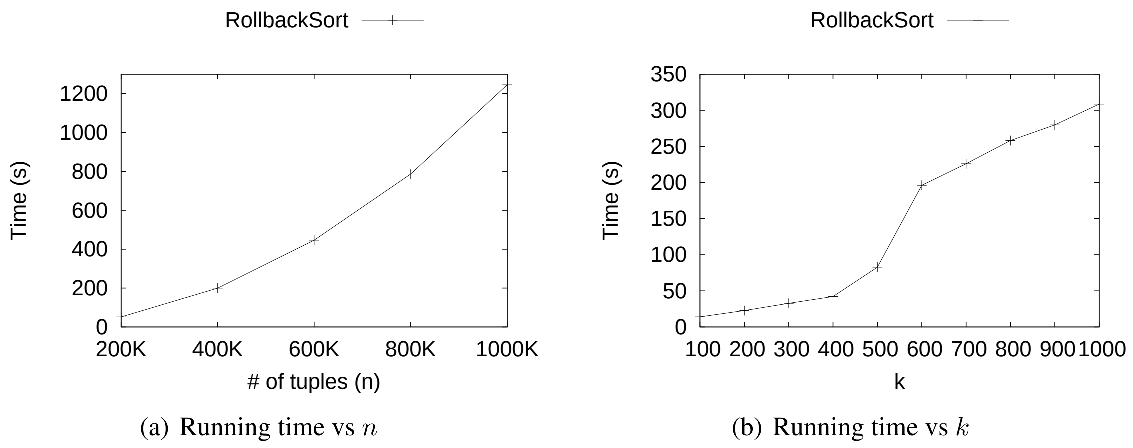
3.4.2 Sensitivity to Parameters

Our second set of experiments studies the influence of various parameters on *Rollback* and *RollbackSort*. The results are shown in Figure 3-4. Notice the difference between the scale of y-axis of Figure 3-4(a) (resp. Figure 3-4(c)) and that of Figure 3-4(b) (resp. Figure 3-4(d)). *RollbackSort* outperforms *Rollback* by one order of magnitude.

Figure 3-4(a) and 3-4(b) show the impact of varying the percentage of exclusive tuples (x) in the dataset. It is to be expected that with the increase of the percentage of exclusive tuples, more rollback operations are needed in both *Rollback* and *RollbackSort*. However, *Rollback* shows a linear increase, while *RollbackSort* shows a trend more than linear but less than quadratic.

Figure 3-4(c) and 3-4(d) illustrate the impact of the size of the parts in the partition. In these two sets of experiments, we fix the total number of exclusive tuples, and vary the max size of a part (s). A large s suggests fewer but relatively larger parts in the partition, as compared to a small s. For both *Rollback* and *RollbackSort*, we see a similar trend that as s increases, the running time decreases. The relative decrease in *Rollback* is larger than that in *RollbackSort*, which can be explained by the fact that *RollbackSort* is already optimized for repetitive occurrences of tuples from the same part, and therefore it should be less subjective to the size of parts.

3.4.3 Scalability



(a) Running time vs n (b) Running time vs k

Figure 3-5: Scalability of *RollbackSort*

As we have already seen analytically in Section 3.3.1 and empirically in Figure 3-3(a), the algorithm for simple probabilistic relations scales linearly to large datasets. *TA* can further improve the performance.

For general probabilistic databases, Figure 3-5 shows that *RollbackSort* scales well to large datasets. Figure 3-5(a) illustrates the running time of *RollbackSort* when n increases to 1M tuples. The trend is more than linear, but much slower than quadratic. Figure 3-5(b) shows the impact of k on the running time. Notice that, the general trend in Figure 3-5(b) is linear except there is a “step-up” when k is about 500. We conjecture that this is due to the non-linear maintenance cost of the priority queue used in the algorithm.

3.5 Complexity

For completeness, we list in Table 3.2 the complexity of the best known algorithm for the semantics in the literature.

Semantics	Simple Probabilistic DB	General Probabilistic DB
Global-Top$k$	$O(kn)$	$O(kn^2)$
PT-k	$O(kn)$	$O(kn^2)$
U-Top$k$	$O(n \log k)$	$O(n \log k)$
U-kRanks	$O(kn)$	$O(kn^2)$
Expected Rank	$O(n \log k)$	$O(n \log k)$

Table 3.2: Time Complexity of Different Semantics

Chapter 4

Top-k Queries in Uncertain Databases under General Scoring Functions

In Chapter 3, we assume that the scoring function is *injective*. It is a simplification which can be lifted. In fact, we often encounter scoring functions where *ties* are common in reality, e.g., hotel ratings. In this chapter, we study the semantics and the computational problem of top-k queries in probabilistic databases under general scoring functions, i.e., with ties. To the best of our knowledge, this is the first attempt to address the tie problem in a rigorous manner.

4.1 Semantics and Postulates

4.1.1 Global-Top$k$ Semantics with Allocation Policy

Under a general scoring function, the Global-Top$k$ semantics remains the same. However, the definition of Global-Top$k$ probability in Definition 3.2.1 needs to be generalized to handle *ties*.

Recall that under an injective scoring function s, there is a unique top-k answer set S in every possible world W. When the scoring function s is non-injective, there may be multiple top-k answer sets $S_1, \ldots, S_d$, each of which is returned nondeterministically. Therefore, for any tuple $t \in \cap S_i, i = 1, \ldots, d$, the world W contributes $Pr(W)$ to the

Global-Top$k$ probability of t. On the other hand, for any tuple $t \in (\cup S_i - \cap S_i), i = 1 \ldots, d$, the world W contributes only a *fraction* of $Pr(W)$ to the Global-Top$k$ probability of t. The *allocation policy* determines the value of this fraction, i.e., the *allocation coefficient*. Denote by $\alpha(t, W)$ the allocation coefficient of a tuple t in a world W. Let $all_{k,s}(W) = \cup S_i, i = 1, \ldots, d$.

Definition 4.1.1 (Global-Top$k$ Probability under a General Scoring Function)**.** *Assume a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, a non-negative integer k and a scoring function s over R^p. For any tuple t in R, the Global-Top$k$ probability of t, denoted by $P^{R^p}_{k,s}(t)$, is the sum of the (partial) probabilities of all possible worlds of R^p whose top-k answer set may contain t.*

$$P^{R^p}_{k,s}(t) = \sum_{\substack{W \in pwd(R^p) \\ t \in all_{k,s}(W)}} \alpha(t, W) Pr(W). \tag{4.1}$$

With no prior bias towards any tuple, it is natural to assume that each of $S_1, \ldots, S_d$ is returned nondeterministically with an *equal* probability. Notice that this probability has nothing to do with tuple probabilities. Rather, it is determined by the number of equally qualified top-k answer sets. Hence, we have the following *Equal* allocation policy.

Definition 4.1.2 (Equal Allocation Policy)**.** *Assume a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, a non-negative integer k and a scoring function s over R^p. For a possible world $W \in pwd(R^p)$ and a tuple $t \in W$, let $a = |\{t' \in W | t' >_s t\}|$ and $b = |\{t' \in W | t' \sim_s t\}|$*

$$\alpha(t, W) = \begin{cases} 1 & \text{if } a < k \text{ and } a + b \leqslant k \\ \dfrac{k-a}{b} & \text{if } a < k \text{ and } a + b > k \end{cases}$$

This notion of *Equal* allocation policy is in the spirit of *uniform allocation policy* introduced in [11] to handle imprecision in OLAP, although the specified goals are different. Note that [11] also introduces other allocation policies based on additional information. In our application, it is also possible to design other allocation policies given additional information.

4.1.2 Satisfaction of Postulates

The semantic postulates in Section 3.2.1 are directly applicable to Global-Top$k$ with allocation policy. In Appendix A, we show that the *Equal* allocation policy preserves the semantic postulates of Global-Top$k$.

4.2 Query Evaluation in Simple Probabilistic Relations

Definition 4.2.1. *Let $R^p = \langle R, p, \mathcal{C} \rangle$ be a probabilistic relation, k a non-negative integer and s a general scoring function over R^p. Assume that $R = \{t_1, t_2, \ldots, t_n\}$, $t_1 \succeq_s t_2 \succeq_s \ldots \succeq_s t_n$. Let $T^{R^p}_{k,[i]}$, $k \leqslant i$, be the sum of the probabilities of all possible worlds of exactly k tuples from $\{t_1, \ldots, t_i\}$:*

$$T^{R^p}_{k,[i]} = \sum_{\substack{W \in pwd(R^p) \\ |W \cap \{t_1, \ldots, t_i\}| = k}} Pr(W)$$

As usual, we omit the superscript in $T^{R^p}_{k,[i]}$, i.e., $T_{k,[i]}$, when the context is unambiguous. Remark 4.2.1 shows that in a simple probabilistic relation $T_{k,[i]}$ can be computed efficiently.

Remark 4.2.1. *Let $R^p = \langle R, p, \mathcal{C} \rangle$ be a simple probabilistic relation, k a non-negative integer and s a general scoring function over R^p. Assume that $R = \{t_1, t_2, \ldots, t_n\}$, $t_1 \succeq_s t_2 \succeq_s \ldots \succeq_s t_n$. For any i, $1 \leqslant i \leqslant n-1$, $T^{R^p}_{k,[i]}$ can be computed using the DP table for computing the Global-Top$k$ probabilities in R^p under an order-preserving injective scoring function s' such that $t_1 \succ_{s'} t_2 \succ_{s'} \ldots \succ_{s'} t_n$.*

Proof. By case study,

- Case 1: If $k = 0$, $1 \leqslant i \leqslant n-1$, then

$$T^{R^p}_{k,[i]} = \prod_{1 \leqslant j \leqslant i} \bar{p}(t_j) = \frac{P^{R^p}_{1,s'}(t_{i+1})}{p(t_{i+1})} \tag{4.2}$$

- Case 2: For every $1 \leqslant k \leqslant i \leqslant n-1$, by the definition of $T^{R^p}_{k,[i]}$, we have

$$T^{R^p}_{k,[i]} = \sum_{\substack{W \in pwd(R^p) \\ |W \cap \{t_1,\ldots,t_i\}| \leqslant k}} Pr(W) - \sum_{\substack{W \in pwd(R^p) \\ |W \cap \{t_1,\ldots,t_i\}| \leqslant k-1}} Pr(W)$$

In the DP table computing the Global-Top$k$ probabilities of tuples in R^p under the injective scoring function s', we have

$$\begin{aligned} P^{R^p}_{k+1,s'}(t_{i+1}) &= \sum_{\substack{W \in pwd(R^p) \\ t_{i+1} \in top_{k+1,s'}(W)}} Pr(W) && (s' \text{ is injective}) \\ &= \sum_{\substack{W \in pwd(R^p) \\ |W \cap \{t_1,\ldots,t_i\}| \leqslant k \\ t_{i+1} \in W}} Pr(W) \\ &= p(t_{i+1}) \sum_{\substack{W \in pwd(R^p) \\ |W \cap \{t_1,\ldots,t_i\}| \leqslant k}} Pr(W) && (\text{tuples are independent}) \end{aligned}$$

Therefore,

$$T^{R^p}_{k,[i]} = \frac{P^{R^p}_{k+1,s'}(t_{i+1})}{p(t_{i+1})} - \frac{P^{R^p}_{k,s'}(t_{i+1})}{p(t_{i+1})} \tag{4.3}$$

□

Remark 4.2.2 shows that we can compute Global-Top$k$ probability under a general scoring function in polynomial time for an extreme case, where the probabilistic relation is simple and all tuples tie in scores. As we will see shortly, this special case plays an important role in our major result in Proposition 4.2.1.

Remark 4.2.2. *Let $R^p = \langle R, p, \mathcal{C} \rangle$ be a simple probabilistic relation, k a non-negative integer and s a general scoring function over R^p. Assume that $R = \{t_1, \ldots, t_m\}$ and $t_1 \sim_s t_2 \sim_s \ldots \sim_s t_m$. For any tuple t_i, $1 \leqslant i \leqslant m$, the Global-Top$k$ probability of t_i, i.e., $P^{R^p}_{k,s}(t_i)$, can be computed using Remark 4.2.1.*

Proof. If $k > m$, it is trivial that $P^{R^p}_{k,s}(t_i) = p(t_i)$. Therefore, we only prove the case when $k \leqslant m$. According to Equation (4.1), for any i, $1 \leqslant i \leqslant m$,

$$
\begin{aligned}
P_{k,s}^{R^p}(t_i) &= \sum_{j=1}^{m} \sum_{\substack{W \in pwd(R^p) \\ t_i \in all_{k,s}(W), |W|=j}} \alpha(t_i, W) Pr(W) \\
&= \sum_{j=1}^{m} \sum_{\substack{W \in pwd(R^p) \\ t_i \in W, |W|=j}} \alpha(t_i, W) Pr(W) \quad \text{(Since all tuple tie , } all_{k,s}(W) = W\text{)} \\
&= \sum_{j=1}^{k} \sum_{\substack{W \in pwd(R^p) \\ t_i \in W, |W|=j}} \alpha(t_i, W) Pr(W) + \sum_{j=k+1}^{m} \sum_{\substack{W \in pwd(R^p) \\ t_i \in W, |W|=j}} \alpha(t_i, W) Pr(W) \\
&= \sum_{j=1}^{k} \sum_{\substack{W \in pwd(R^p) \\ t_i \in W, |W|=j}} Pr(W) + \sum_{j=k+1}^{m} \frac{k}{j} \sum_{\substack{W \in pwd(R^p) \\ t_i \in W, |W|=j}} Pr(W)
\end{aligned}
$$

With out loss of generality, assume $i = m$, then the above equation becomes

$$
\begin{aligned}
P_{k,s}^{R^p}(t_m) &= \sum_{j=1}^{k} \sum_{\substack{W \in pwd(R^p) \\ t_m \in W, |W|=j}} Pr(W) + \sum_{j=k+1}^{m} \frac{k}{j} \sum_{\substack{W \in pwd(R^p) \\ t_m \in W, |W|=j}} Pr(W) \\
&= p(t_i)(\sum_{j=1}^{k} T_{j-1,[m-1]}^{R^p} + \sum_{j=k+1}^{m} \frac{k}{j} T_{j-1,[m-1]}^{R^p}) \qquad (4.4)
\end{aligned}
$$

By Remark 4.2.1, every $T_{j-1,[m-1]}^{R^p}$, $0 \leqslant j-1 \leqslant m-1$, can be computed by the DP table computing Global-Top$k$ probabilities in R^p under an order-preserving injective scoring function s' via Equation (4.2) or (4.3). Therefore, Equation (4.4) can be computed using Remark 4.2.1. □

Based on Remark 4.2.1 and Remark 4.2.2, we design Algorithm 5 and prove its correctness in Theorem 4.2.1 using Proposition 4.2.1.

Assume $R^p = \langle R, p, \mathcal{C} \rangle$ where $R = \{t_1, t_2, \ldots, t_n\}$ and $t_1 \succeq_s t_2 \succeq_s \ldots \succeq_s t_n$. For any $t_l \in R$, i_l is the largest index such that $t_{i_l} \succ_s t_l$, and j_l is the largest index such that $t_{j_l} \succeq_s t_l$. Note that t_1, …, t_{i_l} are tuples with a score higher than that of t_l, and t_{i_l+1}, …, t_{j_l} are tuples tying with t_l.

Intuitively, Algorithm 5 and Proposition 4.2.1 convey the idea that, in a simple probabilistic relation, the computation of Global-Top$k$ probabilities under the *Equal* allocation policy can be simulated by the following procedure:

(S1) Independently flip a biased coin with probability $p(t_j)$ for each tuple $t_j \in R = \{t_1, t_2, \ldots, t_n\}$, which gives us a possible world $W \in pwd(R^p)$;

(S2) Return a top-k answer set S of W nondeterministically (with equal probability in the presence of multiple top-k sets). The Global-Top$k$ probability of t_l is the probability that $t_l \in S$.

The above Step (S1) can be further refined into:

(S1.1) Independently flip a biased coin with probability $p(t_j)$ for each tuple $t_j \in R_A = \{t_1, t_2 \ldots, t_{i_l}\}$, which gives us a collection of tuples W_A;

(S1.2) Independently flip a biased coin with probability $p(t_j)$ for each tuple $t_j \in R_B = \{t_{i_l+1}, \ldots, t_n\}$, which gives us a collection of tuples W_B. $W = W_A \cup W_B$ is a possible world from $pwd(R^p)$;

In order for t_l to be in S, W_A can have at most $k-1$ tuples. Let $|W_A| = k'$, then $k' < k$. Every top-k answer set of W contains all k' tuples from W_A, plus the top-$(k-k')$ tuples from W_B. For t_l to be in S, it has to be in the top-$(k-k')$ set of W_B. Consequently, the probability of $t_l \in S$, i.e., the Global-Top$k$ probability of t_l, is the joint probability that $|W_A| = k' < k$ and t_l belongs to the top-$(k-k')$ set of W_B. The former is $T_{k',[i_l]}$ and the latter is $P^{R^p_B}_{k-k',s}(t_l)$, where R^p_B is R^p restricted to R_B. Again, due to the independence among tuples, Step (S1.1) and Step (S1.2) are independent, and their joint probability is simply the product of the two.

Further notice that since t_l has the highest score in R_B and all tuples are independent in R_B, and any tuple with a score lower than that of t_l does not have an influence on $P^{R^p_B}_{k-k',s}(t_l)$. In other words, $P^{R^p_B}_{k-k',s}(t_l) = P^{R^p_s(t_l)}_{k-k',s}(t_l)$, where $R^p_s(t_l)$ is R^p restricted to all tuples tying with t_l in R. Notice that the computation of $P^{R^p_s(t_l)}_{k-k',s}(t_l)$ is the extreme case addressed in Remark 4.2.2.

Algorithm 5 elaborates the algorithm based on the idea above, where $m = j_l - i_l$ is the number of tuples tying with t_l (including t_l).

Furthermore, Algorithm 5 exploits the overlapping among DP tables and makes the following two optimizations:

1. Use a single DP table to collect the information needed to compute all $T_{k',[i_l]}$, $k' = 0, \ldots, k-1$, $l = 1, \ldots, n$ and $k' \leqslant i_l$ (Line 2).

   Notice that by definition, when $1 \leqslant l \leqslant n$, $1 \leqslant i_l \leqslant n-1$. It is easy to see that the DP table computing $T_{k-1,[n-1]}$ subsumes all other DP tables.

2. Use a single DP table to compute all $P^{R^p_s(t_l)}_{k-k',s}(t_l)$, $k' = 0, \ldots, k-1$, for a tuple t_l (Lines 8-14).

   Notice that by Equation (4.4), for different k', the computation of $P^{R^p_s(t_l)}_{k-k',s}(t_l)$ requires the same set of $T^{R^p_s(t_l)}_{j-1,[m-1]}$ values (Lines 9-11). In Line 13, $P^{R^p_s(t_l)}_{k-k',s}(t_l)$ is abbreviated as $P_l(k'')$, where $k'' = k - k'$, to emphasize the changing parameter k'.

Each DP table computation uses a call to Algorithm 2 (Line 2 in Algorithm 5, Line 3 in Algorithm 6).

Algorithm 6 (Ind_Topk_Gen_Sub) Compute the DP table for Global-Top$k$ probabilities in a Simple Probabilistic Relation under an All-Tie Scoring Function

```
Require: $R^p_s(t_{target}) = \langle R, p, \mathcal{C} \rangle, t_{target}, m$
    (make sure that $|R| = m, t_{target} \in R$)
1: Rearrange tuples in $R$ such that $R = \{t_1, \ldots, t_{m-1}, t_m\}$ and $t_m = t_{target}$;
2: Assume the injective scoring function $s'$ is such that $t_1 \succ_{s'} \ldots \succ_{s'} t_{m-1} \succ_{s'} t_{target}$;
3: Get the DP table
       $q_{tie}(0 \ldots m, 1 \ldots m) = \text{Ind\_Topk\_Sub}(R^p_s(t_{target}), m);$
4: return $q_{tie}(0 \ldots m, 1 \ldots m)$;
```

Proposition 4.2.1. *Let $R^p = \langle R, p, \mathcal{C} \rangle$ be a simple probabilistic relation where $R = \{t_1, \ldots, t_n\}$, $t_1 \succeq_s t_2 \succeq_s \ldots \succeq_s t_n$, k a non-negative integer and s a scoring function.*

Algorithm 5 (Ind_Topk_Gen) Evaluate Global-Top$k$ Queries in a Simple Probabilistic Relation under a General Scoring Function

```
Require: $R^p = \langle R, p, \mathcal{C}\rangle, k$
    (tuples in $R$ are sorted in the non-increasing order based on the scoring function $s$)
1:  Initialize a fixed cardinality $(k+1)$ priority queue $Ans$ of $\langle t, prob\rangle$ pairs, which compares pairs on $prob$, i.e., the Global-Top$k$ probability of $t$;
2:  Get the DP table for computing $T_{k',[i]}, k' = 0, \ldots k-1, i = 1, \ldots, n-1, k' \leqslant i$ using Algorithm 2, i.e.,
        $q(0 \ldots k, 1 \ldots |R|) = \text{Ind\_Topk\_Sub}(R^p, k);$
3:  for $l = 1$ to $|R|$ do
4:      $m = j_l - i_l;$
5:      if $m == 1$ then
6:          Add $\langle t_l, q(k, l)\rangle$ to $Ans$;
7:      else
8:          Get the DP table for computing $P^{R^p_s(t_l)}_{k-k',s}(t_l)$, i.e., $P_l(k-k')$, $k' = 0, \ldots, k-1$
                $q_{tie}(0 \ldots m, 1 \ldots m) = \text{Ind\_Topk\_Gen\_Sub}(R^p_s(t_l), t_l, m);$
9:          for $k'' = 0$ to $m-1$ do
10:             $$T^{R^p_s(t_l)}_{k'',[m-1]} = \frac{q_{tie}(k''+1, m) - q_{tie}(k'', m)}{p(t_l)};$$
11:         end for
12:         for $k'' = 1$ to $k$ do
13:             $$P_l(k'') = p(t_l)(\sum_{j=1}^{k''} T^{R^p_s(t_l)}_{j-1,[m-1]} + \sum_{j=k''+1}^{m} \frac{k''}{j} T^{R^p_s(t_l)}_{j-1,[m-1]});$$
14:         end for
15:         $P^{R^p}_{k,s}(t_l) = 0;$
16:         for $k' = 0$ to $k-1$ do
17:             $$T_{k',[i_l]} = \frac{q(k'+1, i_l+1) - q(k', i_l+1)}{p(t_{i_l+1})};$$
18:             $$P^{R^p}_{k,s}(t_l) = P^{R^p}_{k,s}(t_l) + T_{k',[i_l]} \cdot P_l(k-k');$$
19:         end for
20:         Add $\langle t_l, P^{R^p}_{k,s}(t_l)\rangle$ to $Ans$;
21:     end if
22:     if $|Ans| > k$ then
23:         remove a pair with the smallest $prob$ value from $Ans$;
24:     end if
25: end for
26: return $\{t_i | \langle t_i, prob\rangle \in Ans\}$;
```

For every $t_l \in R$, the Global-Top$k$ probability of t_l can be computed by the following equation:

$$P^{R^p}_{k,s}(t_l) = \sum_{k'=0}^{k-1} T_{k',[i_l]} \cdot P^{R^p_s(t_l)}_{k-k',s}(t_l) \tag{4.5}$$

where $R^p_s(t_l)$ is R^p restricted to $\{t \in R | t \sim_s t_l\}$.

Proof. *See* Appendix B.

Theorem 4.2.1 (Correctness of Algorithm 5)**.** *Given a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, a non-negative integer k and a general scoring function s, Algorithm 5 correctly computes a Global-Top$k$ answer set of R^p under the scoring function s.*

Proof. In Algorithm 5, by Remark 4.2.1, Line 2 and Line 17 correctly compute $T_{k',[i]}$ for $0 \leqslant k' \leqslant k-1$, $1 \leqslant i \leqslant n-1$, $k' \leqslant i$. The entries in Line 8 serve to compute Line 10 by Equation (4.2) and (4.3). Recall that $R^p_s(t_l)$ is R^p restricted to all tuples tying with t_l, which is the extreme case addressed in Remark 4.2.2. By Remark 4.2.2, Line 8 collects the information to compute $P^{R^p_s(t_l)}_{k-k',s}(t_l)$, i.e., $P_l(k'')$, $1 \leqslant k'' = k-k' \leqslant k$. Lines 12-14 correctly compute those values by Equation (4.4). Here, any non-existing $T^{R^p_s(t_l)}_{j-1,[m-1]}$, i.e., $j-1 \notin [0, m-1]$, is assumed to be zero. By Proposition 4.2.1, Lines 15-19 correctly compute the Global-Top$k$ probability of t_l. Also notice that in Line 6, the Global-Top$k$ probability of a tuple without tying tuples is retrieved directly. It is an optimization as the code handling the general case (i.e., $m > 1$, Lines 8-20) works for this special case as well. Again, the top-level structure with the priority queue in Algorithm 5 ensures that a Global-Top$k$ answer set is correctly computed. □

In Algorithm 5, Line 2 takes $O(kn)$, and for each tuple, there is one call to Algorithm 6 in Line 8, which takes $O(m^2_{\max})$, where $m_{\max}$ is the maximal number of tying tuples. Lines 9-11 take $O(m_{\max})$. Lines 12-14 take $O(km_{\max})$. Therefore, Algorithm 5 takes $O(n \max(km_{\max}, m^2_{\max}))$ altogether.

As before, the major space use is the computation of the two DP tables in Line 2 and Line 8. A straightforward implementation leads to $O(kn)$ and $O(m^2_{\max})$ space, respectively. Therefore, the total space is $O(n \max(k, m_{\max}))$. Using a similar space optimization in

Section 3.3.1, the space use for the two DP tables can be reduced to $O(k)$ and $O(m_{\max})$, respectively. Hence, the total space is $O(\max(k, m_{\max}))$.

4.3 Query Evaluation in General Probabilistic Relations

Recall that under an injective scoring function, every tuple t in a general probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$ induces a *simple* event relation E^p, and we reduce the computation of t's Global-Top$k$ probability in R^p to the computation of t_{e_t}'s Global-Top$k$ probability in E^p.

In the case of general scoring functions, we use the same reduction idea. However, now for each part $C_i \in \mathcal{C}, C_i \neq C_{id(t)}$, tuple t induces in E^p two *exclusive* tuples $t_{e_{C_i,\succ}}$ and $t_{e_{C_i,\sim}}$, corresponding to the *event* $e_{C_i,\succ}$ that "there is a tuple from the part C_i with a score *higher than* that of t" and the *event* $e_{C_i,\sim}$ that "there is a tuple from the part C_i with a score *equal to* that of t", respectively. In addition, in Definition 4.3.1, for the ease of description, we allow the existence of tuples with probability 0, which does not affect the technical results. This is an artifact whose purpose will become clear in Theorem 4.3.1.

Definition 4.3.1 (Induced Event Relation under General Scoring Functions)**.** *Given a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, a scoring function s over R^p and a tuple $t \in C_{id(t)} \in \mathcal{C}$, the event relation induced by t, denoted by $E^p = \langle E, p^E, \mathcal{C}^E\rangle$, is a probabilistic relation whose support relation E has only one attribute, $Event$. The relation E and the probability function p^E are defined by the following four generation rules and the postprocess step:*

- *Rule 1.1:* $t_{e_{t,\sim}} \in E$ *and* $p^E(t_{e_{t,\sim}}) = p(t)$;
- *Rule 1.2:* $t_{e_{t,\succ}} \in E$ *and* $p^E(t_{e_{t,\succ}}) = 0$;
- *Rule 2.1:*

$$\forall C_i \in \mathcal{C} \wedge C_i \neq C_{id(t)}.(t_{e_{C_i,\succ}} \in E) \text{ and } p^E(t_{e_{C_i,\succ}}) = \sum_{\substack{t' \in C_i \\ t' \succ_s t}} p(t');$$

- *Rule 2.2:*

$$\forall C_i \in \mathcal{C} \wedge C_i \neq C_{id(t)}.(t_{e_{C_i},\sim} \in E) \text{ and } p^E(t_{e_{C_i},\sim}) = \sum_{\substack{t' \in C_i \\ t' \sim_s t}} p(t').$$

$\{t_{e_t,\sim}, t_{e_t,>}\} \in \mathcal{C}^E$ *and* $\{t_{e_{C_i},\sim}, t_{e_{C_i},>}\} \in \mathcal{C}^E$.

Postprocess step: only when $p^E(t_{e_{C_i},>})$ *and* $p^E(t_{e_{C_i},\sim})$ *are both* 0*, delete both* $t_{e_{C_i},>}$ *and* $t_{e_{C_i},\sim}$.

Proposition 4.3.1. *Given a probabilistic relation* $R^p = \langle R, p, \mathcal{C}\rangle$ *and a scoring function* s*, for any* $t \in R^p$*, the Global-Top*k *probability of* t *equals the Global-Top*k *probability of* $t_{e_t,\sim}$ *when evaluating top-*k *in the induced event relation* $E^p = \langle E, p^E, \mathcal{C}^E\rangle$ *under the scoring function* $s^E : E \to \mathbb{R}$, $s^E(t_{e_t,>}) = \frac{1}{2}$, $s^E(t_{e_t,\sim}) = \frac{1}{2}$, $s^E(t_{e_{C_i},\sim}) = \frac{1}{2}$ *and* $s^E(t_{e_{C_i},>}) = i$*:*

$$P^{R^p}_{k,s}(t) = P^{E^p}_{k,s^E}(t_{e_t,\sim}).$$

Proof. See Appendix B.

Notice that the induced event relation E^p in Definition 4.3.1, unlike its counterpart under an injective scoring function, is not simple. Therefore, we cannot utilize the algorithm in Proposition 4.2.1. Rather, the induced relation E^p is a special general probabilistic relation, where each part of the partition contains *exactly* two tuples. Recall that we allow tuples with probability 0 now. For this special general probabilistic relation, the recursion in Theorem 4.3.1 (Equation (4.6), (4.7)) collects enough information to compute the Global-Top$k$ probability of $t_{e_t,\sim}$ in E^p (Equation (4.8)).

Definition 4.3.2 (Secondary Induced Event Relations)**.** *Let* $E^p = \langle E, p^E, \mathcal{C}^E\rangle$ *be the event relation induced by tuple* t *under a general scoring function* s*. Without loss of generality, assume*

$$E = \{t_{e_{C_1},>}, t_{e_{C_1},\sim}, \ldots, t_{e_{C_{m-1}},>}, t_{e_{C_{m-1}},\sim}, t_{e_t,>}, t_{e_t,\sim}\},$$

and we can split E *into two non-overlapping subsets* $E_>$ *and* $E_\sim$ *such that*

$$E_> = \{t_{e_{C_1},>}, \ldots, t_{e_{C_{m-1}},>}, t_{e_t,>}\},$$
$$E_\sim = \{t_{e_{C_1},\sim}, \ldots, t_{e_{C_{m-1}},\sim}, t_{e_t,\sim}\}.$$

The two secondary induced event relation $E^p_>$ *and* $E^p_\sim$ *are* E^p *restricted to* $E_>$ *and* $E_\sim$*, respectively. They are both simple probabilistic relations which are mutually related. For every* $1 \leqslant i \leqslant m-1$*, the tuple* $t_{i,>}$ *(*$t_{i,\sim}$ *resp.) refers to* $t_{e_{C_i},>}$ *(*$t_{e_{C_i},\sim}$ *resp.). The tuple* $t_{m,>}$ *(*$t_{m,\sim}$ *resp.) refers to* $t_{e_t,>}$ *(*$t_{e_t,\sim}$ *resp.).*

In spirit, the recursion in Theorem 4.3.1 is close to the recursion in Proposition 3.3.1, even though they are not computing the same measure. The following table does a comparison between the measure q in Proposition 3.3.1 and the measure u in Theorem 4.3.1:

Measure	$= \sum Pr(W)$	$\lvert\{t_j \mid t_j \in W,$ $j \leqslant i, t_j \sim_s t\}\rvert$
$q(k,i)$	(1) W contains t_i (2) W has *no more than* k tuples from $\{t_1, t_2, \ldots, t_i\}$	1
$u_>(k,i,b)$	(1) W contains t_i (2) W has *exactly* k tuples from $\{t_{1,>}, t_{2,>}, \ldots, t_{i,>}\}$	b
$u_\sim(k,i,b)$	(1) W contains t_i (2) W has *exactly* k tuples from $\{t_{1,\sim}, t_{2,\sim}, \ldots, t_{i,\sim}\}$	b

Under the general scoring function s^E, a possible world of an induced relation E^p may partially contribute to the tuple $t_{m,\sim}$'s Global-Top$k$ probability. The allocation coefficient depends on the combination of two factors: the number of tuples that are strictly better than $t_{m,\sim}$ and the number of tuples tying with $t_{m,\sim}$. Therefore, in the new measure u, first, we add one more dimension to keep track of b, i.e., the number of tying tuples of a subscript no more than i in a world. Second, we keep track of distinct (k,b) pairs. Furthermore, the recursion on the measure u differentiates between two cases: a non-tying tuple (handled by $u_>$) and a tying tuple (handled by $u_\sim$), since those two types of tuples have different influences on the values of k and b.

Formally, let $u_>(k',i,b)$ ($u_\sim(k',i,b)$ resp.) be the sum of the probabilities of all the possible worlds W of E^p such that

1. $t_{i,>} \in W$ ($t_{i,\sim} \in W$ resp.)

2. i is the k'th smallest tuple subscript in world W

3. the world W contains b tying tuples, i.e., tuples from $E_{\sim}$, with subscript less than or equal to i.

The equations (4.6) and (4.7) resemble Equation (3.3), except that now, since we introduce tuples with probability 0 to ensure that each part of $\mathcal{C}^E$ has exactly two tuples, we need to address the special cases when a divisor can be zero. Notice that, for any $i, 1 \leqslant i \leqslant m$, at least one of $p^E(t_{i,>})$ and $p^E(t_{i,\sim})$ is non-zero, otherwise, they are not in E^p by definition.

Theorem 4.3.1. *Given a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, a scoring function s, $t \in R^p$, and its induced event relation $E^p = \langle E, p^E, \mathcal{C}^E\rangle$, where $|E| = 2m$, the recursion in Table 4.1 on $u_>(k', i, b)$ and $u_\sim(k', i, b)$ holds, where $b_{\max}$ is the number of tuples with a positive probability in $E^p_\sim$. The Global-Top$k$ probability of $t_{e_t,\sim}$ in E^p under the scoring function s^E can be computed by the following equation:*

$$\begin{aligned} P^{E^p}_{k,s^E}(t_{e_t,\sim}) &= P^{E^p}_{k,s^E}(t_{m,\sim}) \\ &= \sum_{b=1}^{b_{\max}} \Big(\sum_{k'=1}^{k} u_\sim(k', m, b) + \sum_{k'=k+1}^{k+b-1} \frac{k-(k'-b)}{b} u_\sim(k', m, b) \Big) \end{aligned} \tag{4.8}$$

Proof. See Appendix B.

Recall that we designed Algorithm 1 based on the recursion in Proposition 3.3.1. Similarly, a DP algorithm based on the mutual recursion in Theorem 4.3.1 can be designed. We are going to skip the details. Instead, we show how the algorithm works using Example 12 below.

The time complexity of the recursion in Theorem 4.3.1 determines the complexity of the algorithm. It takes $O(b_{\max} n^2)$ for one tuple, and $O(b_{\max} n^3)$ for computing all n tuples. Recall that $m_{\max}$ is the maximal number of tying tuples in R, and thus $b_{\max} \leqslant m_{\max}$. Again, the priority queue takes $O(n \log k)$. Altogether, the algorithm takes $O(m_{\max} n^3)$ time.

The space complexity of this algorithm is $O(b_{\max} n^2)$ in a straightforward implementation and $O(b_{\max} n)$ if space optimized as in Section 3.3.1.

When $i = 1, 0 \leqslant k' \leqslant m$ and $0 \leqslant b \leqslant b_{\max}$,

$$u_{>}(k', 1, b) = \begin{cases} p^E(t_{1,>}) & k' = 1, b = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$u_{\sim}(k', 1, b) = \begin{cases} p^E(t_{1,\sim}) & k' = 1, b = 1 \\ 0 & \text{otherwise} \end{cases}$$

For every i, $2 \leqslant i \leqslant m$, $0 \leqslant k' \leqslant m$ and $0 \leqslant b \leqslant b_{\max}$,

$u_{>}(k', i, b) =$ (4.6)

Condition	Formula
$k' = 0$	0
$1 \leqslant k' \leqslant m, p^E(t_{i-1,>}) > 0$	$(u_{>}(k', i-1, b)\frac{1 - p^E(t_{i-1,>}) - p^E(t_{i-1,\sim})}{p^E(t_{i-1,>})}$ $+u_{>}(k'-1, i-1, b)$ $+u_{\sim}(k'-1, i-1, b))p^E(t_{i,>})$
$1 \leqslant k' \leqslant m, p^E(t_{i-1,>}) = 0$ and $0 \leqslant b < b_{\max}$	$(u_{\sim}(k', i-1, b+1)\frac{1 - p^E(t_{i-1,>}) - p^E(t_{i-1,\sim})}{p^E(t_{i-1,\sim})}$ $+u_{>}(k'-1, i-1, b)$ $+u_{\sim}(k'-1, i-1, b))p^E(t_{i,>})$
$1 \leqslant k' \leqslant m, p^E(t_{i-1,>}) = 0$ and $b = b_{\max}$	$(u_{>}(k'-1, i-1, b) + u_{\sim}(k'-1, i-1, b))p^E(t_{i,>})$

$u_{\sim}(k', i, b) =$ (4.7)

Condition	Formula
$k' = 0$ or $b = 0$	0
$1 \leqslant k' \leqslant m, 1 \leqslant b \leqslant b_{\max}$ and $p^E(t_{i-1,\sim}) > 0$	$(u_{\sim}(k', i-1, b)\frac{1 - p^E(t_{i-1,>}) - p^E(t_{i-1,\sim})}{p^E(t_{i-1,\sim})}$ $+u_{>}(k'-1, i-1, b-1)$ $+u_{\sim}(k'-1, i-1, b-1))p^E(t_{i,\sim})$
$1 \leqslant k' \leqslant m, 1 \leqslant b \leqslant b_{\max}$ and $p^E(t_{i-1,\sim}) = 0$	$(u_{>}(k', i-1, b-1)\frac{1 - p^E(t_{i-1,>}) - p^E(t_{i-1,\sim})}{p^E(t_{i-1,>})}$ $+u_{>}(k'-1, i-1, b-1)$ $+u_{\sim}(k'-1, i-1, b-1))p^E(t_{i,\sim})$

Table 4.1: Recursion in Theorem 4.3.1

Example 12. *When evaluating a top-*2 *query in* $R^p = \langle R, p, \mathcal{C}\rangle$*, consider a tuple* $t \in R$ *and its induced event relation* $E^p = \langle E, p^E, \mathcal{C}^E\rangle$

$E_{\succ}$	$t_{e_{C_1},\succ}$ (t_1)	$t_{e_{C_2},\succ}$ (t_3)	$t_{e_{C_3},\succ}$ (t_5)	$t_{e_t,\succ}$ (t_7)
p^E	0.6	0.5	0.2	0

$E_{\sim}$	$t_{e_{C_1},\sim}$ (t_2)	$t_{e_{C_2},\sim}$ (t_4)	$t_{e_{C_3},\sim}$ (t_6)	$t_{e_t,\sim}$ (t_8)
p^E	0	0.25	0.6	0.4

In order to compute the Global-Topk probability of t_8 *(i.e.,* $t_{e_t,\sim}$*) in* E^p*, Theorem 4.3.1 leads to the following DP tables, each for a distinct combination of a* b *value and a secondary induced relation, where* $b_{\max} = 3$*.*

$k\backslash t$	t_1	t_3	t_5	t_7
0	0	0	0	0
1	0.6	0.2	0.02	0
2	0	0.3	0.07	0
3	0	0	0.06	0
4	0	0	0	0

(a) ($b = 0$, $E^p_{\succ}$)

$k\backslash t$	t_1	t_3	t_5	t_7
0	0	0	0	0
1	0	0	0	0
2	0	0	0.02	0
3	0	0	0.03	0
4	0	0	0	0

(b) ($b = 1$, $E^p_{\succ}$)

$k\backslash t$	t_1	t_3	t_5	t_7
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

(c) ($b = 2$, $E^p_{\succ}$)

$k\backslash t$	t_1	t_3	t_5	t_7
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

(d) ($b = 3$, $E^p_{\succ}$)

$k\backslash t$	t_2	t_4	t_6	t_8
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

(e) ($b = 0$, $E^p_{\sim}$)

$k\backslash t$	t_2	t_4	t_6	t_8
0	0	0	0	0
1	0	0.1	0.06	**0.008**
2	0	0.15	0.21	**0.036**
3	0	0	0.18	0.052
4	0	0	0	0.024

(f) ($b = 1$, $E^p_{\sim}$)

$k\backslash t$	t_2	t_4	t_6	t_8
0	0	0	0	0
1	0	0	0	**0**
2	0	0	0.06	**0.032**
3	0	0	0.09	**0.104**
4	0	0	0	0.084

(g) ($b = 2$, $E^p_{\sim}$)

$k\backslash t$	t_2	t_4	t_6	t_8
0	0	0	0	0
1	0	0	0	**0**
2	0	0	0	**0**
3	0	0	0	**0.024**
4	0	0	0	**0.036**

(h) ($b = 3$, $E^p_{\sim}$)

Figure 4-1: Mutual Recursion in Example 12

The computation of each entry follows the mutual recursion in Theorem 4.3.1, for example,

$$\begin{aligned} u_{\succ}(2, t_5, 0) &= (u_{\succ}(1, t_3, 0) + u_{\sim}(1, t_4, 0) + u_{\succ}(2, t_3, 0)\frac{1 - p^E(t_3) - p^E(t_4)}{p^E(t_3)})p^E(t_5) \\ &= (0.2 + 0 + 0.3\frac{1 - 0.5 - 0.25}{0.5})0.2 = 0.07 \end{aligned}$$

$$
\begin{aligned}
u_{\sim}(2,t_6,1) &= (u_{>}(1,t_3,0)+u_{\sim}(1,t_4,0)+u_{\sim}(2,t_4,1)\frac{1-p^E(t_3)-p^E(t_4)}{p^E(t_4)})p^E(t_6) \\
&= (0.2+0+0.15\frac{1-0.5-0.25}{0.25})0.6 = 0.21
\end{aligned}
$$

Finally, under the scoring function s^E defined in Proposition 4.3.1

$$
\begin{aligned}
P^{E^p}_{k,s^E}(t_{e_t,\sim}) &= P^{E^p}_{2,s^E}(t_8) \\
&= \sum_{b=1}^{3}(\sum_{k'=1}^{2} u_{\sim}(k',8,b) + \sum_{k'=2+1}^{2+b-1}\frac{2-(k'-b)}{b}u_{\sim}(k',8,b)) \\
&= u_{\sim}(1,t_8,1)+u_{\sim}(2,t_8,1) \\
&\quad +u_{\sim}(1,t_8,2)+u_{\sim}(2,t_8,2)+\frac{1}{2}u_{\sim}(3,t_8,2) \\
&\quad +u_{\sim}(1,t_8,3)+u_{\sim}(2,t_8,3)+\frac{2}{3}u_{\sim}(3,t_8,3)+\frac{1}{3}u_{\sim}(3,t_8,4) \\
&= 0.008+0.036+0+0.032+\frac{1}{2}0.104+0+0+\frac{2}{3}0.024+\frac{1}{3}0.036 \\
&= 0.156
\end{aligned}
$$

Bold entries in Figure 4-1 are involved in the above equation.

4.4 Conclusion

We study the semantic and computational problems for top-k queries in probabilistic databases in Chapter 3 and Chapter 4.

In Chapter 3, we propose three postulates to categorize top-k semantics in probabilistic databases and discuss their satisfaction by the semantics in the literature. Those postulates are the first step to analyze different semantics. We do not think that a single semantics is superior/inferior to other semantics just because of postulate satisfaction. Rather, we deem that the choice of the semantics should be guided by the application. The postulates help to create a profile of each semantics.

We propose a new top-k semantics, Global-Top$k$, which satisfies the postulates to a large degree. We study the computational problem of query evaluation under Global-Top$k$

semantics for simple and general probabilistic relations when the scoring function is injective. For simple probabilistic relations, we propose a dynamic programming algorithm and effectively optimize it with Threshold Algorithm. For general probabilistic relations, we show a polynomial reduction to the simple case, and design *Rollback* and *RollbackSort* optimizations to speed up the computation. We conduct an empirical study to verify the effectiveness of those optimizations.

In Chapter 4, we extend the Global-Top$k$ semantics to general scoring functions and introduce the concept of *allocation policy* to handle ties in score. To the best of our knowledge, this is the first attempt to address the tie problem rigorously. Previous work either does not consider ties or uses an arbitrary tie-breaking mechanism. Advanced dynamic programming algorithms are proposed for query evaluation under general scoring functions for both simple and general probabilistic relations.

We provide theoretical analysis following every algorithm proposed.

4.5 Future Work

Recently, several new semantics and variants of the existing semantics have been proposed in the literature [53, 19]. So far, the research reported in the literature has primarily focused on independent and exclusive relationships among tuples [52, 53, 32, 56, 19]. It will be interesting to investigate other complex relationships between tuples. Other possible directions include top-k aggregate queries [53], top-k evaluation in other uncertain database models proposed in the literature [49] and more general preference queries in probabilistic databases.

Chapter 5

Preference Strength in Probabilistic Ranking Queries

In probabilistic database research, there have been elevated interests in top-k queries [52, 32, 58, 19, 28, 42, 60], skylines [50, 3, 44, 57], nearest neighbor queries [15, 14, 5, 13] and other forms of ranking queries [43]. More often than not, the semantics of those queries is about the trade-off between probability and score. However, an interesting phenomenon is that we tend to overlook the importance of score in designing the semantics for various queries. Take top-k queries for example. Example 13 illustrates one shortcoming of the Global-Top$k$ semantics. Other semantics of top-k queries, e.g. U-Top$k$ [52], U-kRanks [52] and PT-k [32], suffer from the same problem.

Example 13. *A travel agent is buying two tickets. The choices are as follows.*

Flight(duration, price, . . .)	*Score*	*On-time*
FL10(5.1h, $99)	0.9	0.3
FL20(5h, $200)	0.6	0.4
FL30(5.1h, $205)	0.59	0.1
FL40(5.2h, $210)	0.58	0.7

where the Score *is the normalized score of each flights based on their duration, price, service etc., and* On-time *is the probability that flight is on-time based on historical statistics.*

This decision problem can be formulated as a top-k query over the above probabilistic relation, where $k = 2$. The Global-Top$k$ return $\{FL40(0.5838), FL20(0.4)\}$. However, the agent is under the pressure of saving cost, and since FL20 is not significantly better *on time compared to FL10 and FL10 is* a lot cheaper *than FL20, he might prefer FL10 instead.*

In Example 13, Global-Top$k$ is not able to reflect the preference based on *how much* better/worse each alternative is when compared to other alternatives, since the original Global-Top$k$ semantics uses *ordinal* scores. In *ordinal* scores, the preference among tuples remain unchanged as long as the order induced by scores over tuples does not change. In other words, the magnitude of scores does not matter as long as the order induced remains unchanged. While this property could be desirable in some applications, it might be counterintuitive in others, e.g., Example 13.

Example 13 illustrates an application where *cardinal* scores are desirable. While Example 13 exemplifies the semantics of Global-Top$k$ of top-k queries in probabilistic databases, the *ordinal* score is a prevalent problem in various kinds of ranking problems in probabilistic databases.

In general, the use of scores is a natural way to express the “degree of preference” when the alternatives, i.e., tuples, are certain. Notice that *preference strength* is not an issue in ranking in deterministic databases: as long as t_1 is better than t_2 by some standard, we have to choose t_1 before choosing t_2. However, when we start considering uncertain data, we might want to trade in a little probability for a large margin in score. There is a natural trade-off between *probability* and *score*.

In this chapter, we investigate the integration of *preference strength* into the semantics of top-k queries in probabilistic databases.

5.1 Preference Strength in Probabilistic Top-k Queries

5.1.1 Sensitivity Postulates

Recall that we introduce three postulates, namely *Exact k*, *Faithfulness* and *Stability*, in Chapter 3 to facilitate the selection of the desirable semantics based on the user’s application. Now, we introduce two more postulates: *Sensitivity to Probability* and *Sensitivity to Score*.

Recall the notions used in Chapter 3. For a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, denote by $Ans_{k,s}(R^p)$ the collection of all top-k answer sets of R^p under the scoring function s. Denote by $\cup\, Ans_{k,s}(R^p)$ the union of all top-k sets, i.e., $\cup\, Ans_{k,s}(R^p) = \cup_{S \in Ans_{k,s}(R^p)} S$.

- *Sensitivity to Probability*: When $R^p = \langle R, p, \mathcal{C} \rangle$ is sufficiently large ($|\mathcal{C}| > |\cup\, Ans_{k,s}(R^p)|$) and $k > 0$, then for every $t \in R - \cup\, Ans_{k,s}(R^p)$, there exists a probability function $p' \neq p$, such that $(R^p)' = \langle R, p', \mathcal{C} \rangle$ and $t \in \cup\, Ans_{k,s}((R^p)')$.

- *Sensitivity to Score*: When $R^p = \langle r, p, \mathcal{C} \rangle$ is sufficiently large ($|\mathcal{C}| > |\cup\, Ans_{k,s}(R^p)|$) and $k > 0$, then for every $t \in \cup\, Ans_{k,s}(R^p)$, there exists a scoring function $s' \neq s$, such that $t \in \cup\, Ans_{k,s'}(R^p)$.

Intuitively, those two sensitivity postulates require that the semantics should be dominated by neither *probability* nor *score*. Rather, the semantics should reflect the *trade-off* between those two factors. The use of ordinal scores makes it rather difficult to satisfy *Sensitivity to Score*. In fact, all the semantics considered so far (U-Top$k$, U-kRanks, Global-Top$k$ and PT-k) do not satisfy *Sensitivity to Score*. On the other hand, all of them satisfy *Sensitivity to Probability*. This clearly indicates that we weigh *probability* and *score* unequally in designing those semantics.

5.1.2 Global-Top$k^{\gamma,\beta}$ Semantics

We propose Global-Top$k^{\gamma,\beta}$ semantics, which integrates *preference strength* and satisfies both *Sensitivity to Probability* and *Sensitivity to Score* postulates.

Definition 5.1.1 (Global-Top$k^{\gamma,\beta}$ Value)**.** *Assume a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, a non-negative integer k and a scoring function $s : R \mapsto (0, 1]$. For every tuple t in R, the Global-Top$k^{\gamma,\beta}$ value of t, denoted by $v_{k,s}^{\gamma,\beta,R^p}(t)$, is the product of its Global-Top$k$ probability raised to γ and its score raised to β, where $\gamma, \beta \in [0, 1]$.*

$$v_{k,s}^{\gamma,\beta,R^p}(t) \quad = \quad P_{k,s}^{R^p}(t)^{\gamma} \cdot s(t)^{\beta} \tag{5.1}$$

Following the notion in Chapter 3, $P_{k,s}^{R^p}(t)$ is the Global-Top$k$ probability of the tuple t in the probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$ under the scoring function s. In the rest of this chapter, we omit the superscripts and subscripts in Definition 5.1 when the context is unambiguous. That is, denote by $P(t)$ the Global-Top$k$ probability of the tuple t, and the Global-Top$k^{\gamma,\beta}$ value of the tuple t is therefore

$$v^{\gamma,\beta}(t) \quad = \quad P(t)^{\gamma} \cdot s(t)^{\beta} \tag{5.2}$$

A parameterized semantics based on Global-Top$k^{\gamma,\beta}$ values is as follows:

- **Global-Top**$k^{\gamma,\beta}$: return k highest-ranked tuples with the highest Global-Top$k^{\gamma,\beta}$ value.

where $\gamma, \beta \in [0, 1]$.

Notice that in Definition 5.1.1, we require the scores to be normalized in order to have the same range as probabilities, i.e., $(0, 1]$.

Corollary 5.1.1. *The Global-Top$k^{\gamma,\beta}$ semantics is a generalization of the Global-Top$k$ semantics.*

Proof. When $\gamma = 1$ and $\beta = 0$, the Global-Top$k^{\gamma,\beta}$ value degenerates to the Global-Top$k$ probability. □

Theorem 5.1.1. *When $\gamma_1/\beta_1 = \gamma_2/\beta_2$, the Global-Top$k^{\gamma_1,\beta_1}$ semantics induces the same order over the tuples in the probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$ under the scoring function s as the Global-Topk^{γ_2,β_2} semantics does.*

Proof. We show that if $\gamma_1/\beta_1 = \gamma_2/\beta_2$, then for every $t_i, t_j \in R$,

$$v^{\gamma_1,\beta_1}(t_i) > v^{\gamma_1,\beta_1}(t_j) \Leftrightarrow v^{\gamma_2,\beta_2}(t_i) > v^{\gamma_2,\beta_2}(t_j)$$

which is a sufficient condition for the theorem to hold.

$$
\begin{aligned}
& & & v^{\gamma_1,\beta_1}(t_i) > v^{\gamma_1,\beta_1}(t_j) \\
& & \Leftrightarrow\ & P(t_i)^{\gamma_1} s(t_i)^{\beta_1} > P(t_j)^{\gamma_1} s(t_j)^{\beta_1} \\
& & \Leftrightarrow\ & (P(t_i) s(t_i)^{\frac{\beta_1}{\gamma_1}})^{\gamma_1} > (P(t_j) s(t_j)^{\frac{\beta_1}{\gamma_1}})^{\gamma_1} \\
& (\text{since } 0 < \gamma_1 \leqslant 1) & \Leftrightarrow\ & P(t_i) s(t_i)^{\frac{\beta_1}{\gamma_1}} > P(t_j) s(t_j)^{\frac{\beta_1}{\gamma_1}} && (5.3) \\
& (\text{since } \frac{\gamma_1}{\beta_1} = \frac{\gamma_2}{\beta_2}) & \Leftrightarrow\ & P(t_i) s(t_i)^{\frac{\beta_2}{\gamma_2}} > P(t_j) s(t_j)^{\frac{\beta_2}{\gamma_2}} \\
& (\text{since } 0 < \gamma_2 \leqslant 1) & \Leftrightarrow\ & (P(t_i) s(t_i)^{\frac{\beta_2}{\gamma_2}})^{\gamma_2} > (P(t_j) s(t_j)^{\frac{\beta_2}{\gamma_2}})^{\gamma_2} && (5.4) \\
& & \Leftrightarrow\ & P(t_i)^{\gamma_2} s(t_i)^{\beta_2} > P(t_j)^{\gamma_2} s(t_j)^{\beta_2} \\
& & \Leftrightarrow\ & v^{\gamma_2}\beta_2(t_i) > v^{\gamma_2}\beta_2(t_j)
\end{aligned}
$$

Equation 5.3 and Equation 5.4 are due to the fact that the function $y = x^a$ is strictly increasing on $x > 0$ for any constant $a \in (0, 1]$. □

Theorem 5.1.1 shows that the Global-Top$k^{\gamma,\beta}$ is *order-preserving* as long as the ratio between γ and β remains a constant. Notice that in the Global-Top$k^{\gamma,\beta}$ semantics, we only care about the order induced by the Global-Top$k^{\gamma,\beta}$ values over tuples. In other words, the exact Global-Top$k^{\gamma,\beta}$ value does not matter. Therefore, by Theorem 5.1.1, we can fix $\gamma = 1$ and only vary the β parameter in the Global-Top$k^{\gamma,\beta}$ semantics. We refer to the resulting semantics as the Global-Topk^{β} semantics.

Example 14. *In Example 13, by setting $\beta = 1$, the top-2 flights returned by Global-Topk^{β} are $\{FL10, FL40\}$.*

Corollary 5.1.2. *The Global-Topk^{β} semantics is equivalent to the Global-Top$k$ semantics when $\beta = 0$.*

In the Global-Topk^{β} semantics, the Global-Topk^{β} value of a tuple t is therefore denoted by $v^{\beta}(t)$.

5.1.3 Postulate Satisfaction

Corollary 5.1.2 illustrates that Global-Topk^{β} coincides with Global-Top$k$ when $\beta = 0$. Therefore, we are interested in the postulate satisfaction when $\beta > 0$. Table 5.1 summarizes the results. In addition, we also list in Table 5.1 the postulate satisfaction of the original Global-Top$k$ semantics, especially for the sensitivity postulates.

Semantics	Exact k	Faithfulness	Stability	Sens. to Prob.	Sens. to Score
Global-Topk^{β}	✓	✓/×	✓	✓	✓
Global-Top$k$	✓	✓/×	✓	✓	×

Table 5.1: Postulate Satisfaction for Global-Topk^{β} ($\beta > 0$)

See Appendix C for the proofs of Table 5.1.

As we can see in Table 5.1, Global-Topk^{β} inherits most postulates of Global-Top$k$ except for *Sensitivity to Score*. The importance of this postulates lies in the additional flexibility to specify the trade-off between probabilities and scores.

5.1.4 The Elicitation of the β Parameter

As true for any parameterized semantics, the parameter elicitation is important when the user is unable to supply the parameters directly. If the user wants to use an ordinal score, he/she should set $\beta = 0$. Generally speaking, a relatively smaller β emphasizes the *probability* factor while a relatively larger β emphasizes the *score* factor.

If the user is not able to specify the parameter, in most cases, he/she will be able to supply training data for parameter elicitation.

Definition 5.1.2 (Training Dataset). *A training dataset A is a collection of pairwise comparisons between tuples. If $(t_i, t_j) \in A$, then the tuple t_i is preferred to the tuple t_j by the user, denoted by $t_i \succ_u t_j$.*

For each pairwise comparison in the training dataset, we can compute a *feasible region* of the parameter β, such that every β value within that region leads to a Global-Topk^{β}

semantics inducing an order over tuples that is consistent with the pairwise comparison. Therefore, the elicitation of β from the training dataset becomes the computation of the intersection of feasible regions derived from each pairwise comparison.

Definition 5.1.3 (Feasible Region)**.** *Assume a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, a non-negative integer k, an injective scoring function s and a training dataset A. For every $t_i, t_j \in R$ and $(t_i, t_j) \in A$, the corresponding feasible region I_{ij} is the union of intervals on the real line such that for every $\beta \in I_{ij}$, $v^\beta(t_i) > v^\beta(t_j)$.*

The feasible region of the training dataset A, denoted by I_A, is the intersection of the feasible regions of every pairwise comparison in A.

$$I_A = \bigcap_{(t_i,t_j)\in A} I_{ij}$$

For each pairwise comparison in the training set, the elicitation can be categorized into the following four cases.

Case 1: If $t_1 >_s t_2$, $P_{k,s}(t_1) > P_{k,s}(t_2)$ and $t_1 >_u t_2$.

Feasible region: $\beta > 0$

Case 2: If $t_1 >_s t_2$, $P_{k,s}(t_1) > P_{k,s}(t_2)$ and $t_2 >_u t_1$.

Feasible region: $\varnothing$. This pairwise comparison is inconsistent to our model.

Case 3: If $t_1 >_s t_2$, $P_{k,s}(t_1) < P_{k,s}(t_2)$ and $t_1 >_u t_2$.

Feasible region: $\beta > \frac{\log P_{k,s}(t_2) - \log P_{k,s}(t_1)}{\log s(t_1) - \log s(t_2)}$

Case 4: If $t_1 >_s t_2$, $P_{k,s}(t_1) < P_{k,s}(t_2)$ and $t_2 >_u t_1$.

Feasible region: $0 < \beta < \frac{\log P_{k,s}(t_2) - \log P_{k,s}(t_1)}{\log s(t_1) - \log s(t_2)}$

Proposition 5.1.1. *The feasible region of a pairwise comparison is a finite union of intervals.*

Proof. The feasible region of a pairwise comparison can only be one of the aforementioned four cases. Each is a finite union of intervals. □

Proposition 5.1.2. *The feasible region of a training dataset is a finite union of intervals.*

Proof. The training dataset contains finitely many pairwise comparisons. Therefore, the conclusion follows Proposition 5.1.1. □

Definition 5.1.4 (Elicitation Problem). *Assume a simple probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, an injective scoring function s and a training dataset A. For every real number β, let $f(\beta)$ be the number of pairwise comparisons in A whose feasible regions contain β. The elicitation problem is to find $\{\beta | \max_{\beta>0} f(\beta)\}$.*

The purpose of elicitation is to find intervals on the real line which satisfy the largest number of pairwise comparisons in the training set A.

Algorithm 7 illustrates the elicitation algorithm for the parameter β. From the aforementioned case study, we know that the interval derived from a single pairwise comparison is of the form $(0, a)$ or (b, ∞). Therefore, in Line 1 of Algorithm 7, those two types of intervals are ordered on a and b, respectively. Notice that the total number of intervals are no more than the cardinality of the training dataset A, i.e., $n+m \leqslant |A|$, since each pairwise comparison produces at most one interval, and the empty intervals are filtered out. In Line 2, we mark the real line $(0, \infty)$ with points from $S_a = \{a_1, \ldots, a_n\}$ and $S_b = \{b_1, \ldots, b_m\}$, and rename each point with $c_i, i = 1, \ldots, m+n$ in non-decreasing order. $c_i, 1 \leqslant i \leqslant n+m$ partition $(0, \infty)$ into $m + n + 1$ intervals. For each end of those intervals, whether it is an open or close end depends on whether it is a left or right end, and whether it is from S_a or S_b. Line 7 to Line 19 determines whether each end is open or close for all intervals. ∞ is always an open end of an interval.

Theorem 5.1.2. *Assume a simple probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, an injective scoring function s and a training dataset A. For every $\beta \in I^*$ returned by Algorithm 7, $f(\beta) = \max_{\beta>0} f(\beta)$.*

Proof. We show that

1. Intervals created by c_i in Line 6-Line 27 cover the entire range $(0, \infty)$;

   The intervals in I are disjoint and their union covers $(0, \infty)$. For example, if the current c_i, $i = 1, \ldots, m + n$, is the right end of the current interval and $c_i \in S_a$,

Algorithm 7 β **Elicitation Algorithm**

```
Require: a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, a non-negative integer $k$, a scoring function $s$, a training set $A$
Ensure: a collection of intervals where the $\beta$ value satisfies as many pairwise comparisons in $A$ as possible
 1: Compute the feasible region for each pair in $A$ and sort the resulting non-empty intervals into
        $$\{(0, a_1), \ldots, (0, a_n)\} \cup \{(b_1, \infty), \ldots, (b_m, \infty)\}$$
    such that $0 < a_1 \leqslant \ldots \leqslant a_n$ and $0 < b_1 \leqslant \ldots \leqslant b_m$.
 2: Let $S_a = \{a_1, \ldots, a_n\}$ and $S_b = \{b_1, \ldots, b_m\}$. Let $c_i \in S_a \cup S_b$, $i = 1, \ldots, m + n$ and
        $$0 < c_1 \leqslant \ldots \leqslant c_{m+n}$$
 3: $c_0 = 0$, $c_{m+n+1} = \infty$
 4: $w_a = n$, $w_b = 0$
 5: $I = \varnothing$
 6: for $i = 1$ to $m + n + 1$ do
 7:    if $c_{i-1} \in S_a$ then
 8:       if $c_i \in S_a$ or $c_i == \infty$ then
 9:          $I_{cur} = [c_{i-1}, c_i)$
10:       else
11:          $I_{cur} = [c_{i-1}, c_i]$
12:       end if
13:    else
14:       if $c_i \in S_a$ or $c_i == \infty$ then
15:          $I_{cur} = (c_{i-1}, c_i)$
16:       else
17:          $I_{cur} = (c_{i-1}, c_i]$
18:       end if
19:    end if
20:    $I = I \cup \{I_{cur}\}$
21:    $f(I_{cur}) = w_a + w_b$
22:    if $c_i \in S_a$ then
23:       $w_a = w_a - 1$
24:    else if $c_i \in S_b$ then
25:       $w_b = w_b + 1$
26:    end if
27: end for
28: return
        $$I^* = \max_{f(I_i)}\{I_i | I_i \in I\}$$
```

then c_i is a close right end (Line 9, Line 15). In the next iteration, c_i becomes the left end of the next interval, which is open (Line 9, Line 11). For $c_i \in S_b$, similar reasoning applies. For the boundary case $c_0 = 0$, since $c_0 \notin S_a$, either Line 15 or Line 17 is applicable to this case, which results in an open left end. For the case that $c_{m+n+1} = \infty$, either Line 9 or Line 15 applies, which leads to an open right end.

2. For every interval $I_i \in I$, every Global-Topk^{β} semantics where $\beta \in I_i$ satisfies $f(I_i)$ number of pairwise comparisons in the training dataset A. Line 21 computes the $f(I_i)$ number for every interval in $I_i \in I$.

   For every $\beta > 0$, the total number of intervals from $S_a \cup S_b$ which contain β is the number of pairwise comparisons satisfied by the corresponding Global-Topk^{β} semantics. It is easy to verify that every β value from the interval $[a_j, a_{j+1})$, $1 \leqslant j < n$ is contained in $n - j$ intervals $(0, a_{j+1}), \ldots, (0, a_n)$ from S_a. Every β value from the interval $(0, a_1)$ is contained in all the n intervals $(0, a_1), \ldots, (0, a_n)$ from S_a, while every β value from the interval (a_n, ∞) is not contained in any interval from S_a. Similarly, every β value from $(b_l, b_{l+1}]$, $1 \leqslant l < m$ is contained in l intervals $\{(b_1, \infty), \ldots, (b_l, \infty)\}$ from S_b. Every β from $(0, b_1)$ is not contained in any interval from S_b, while every β from (b_m, ∞) is contained in all the m intervals $(b_1, \infty), \ldots, (b_m, \infty)$ from S_b. In Algorithm 7, $f(I_i) = n - j + l$ if and only if $I_i \subseteq [a_j, a_{j+1})$ and $I_i \subseteq (b_l, b_{l+1}]$ (Line 3-Line 27), which is the number of intervals from $S_a \cup S_b$ containing β for every $\beta \in I_i$.

3. Algorithm 7 returns every interval which contains β values that lead to the largest number of satisfied pairwise comparisons in the training dataset A.

   Line 28 in Algorithm 7 returns all and only those intervals with the highest f values, which is the number of pairwise comparisons satisfied in A by using any β from those intervals.

$\square$

The sorting in Line 1 and Line 2 takes $O(|A| \log |A|)$ time, where $|A|$ is the size of the training dataset. Both the computation of f values (Line 3-Line27) and finding the intervals with the maximal f value (Line 28) take a linear sweep over the $O(|A|)$ intervals. Altogether, Algorithm 7 runs in $O(|A| \log |A|)$ time.

The Generalized Elicitation of the Parameter β A generalized situation with β elicitation is when the user is only able to provide pairwise comparison training data with uncertainty. In such cases, each pair in the training dataset is associated with a probability.

Definition 5.1.5 (Probabilistic Training Dataset). *A probabilistic training dataset A is a collection of probabilistic pairwise comparisons between tuples. If $(t_i, t_j, p_{ij}) \in A$, then the tuple t_i is preferred to the tuple t_j by the user with a probability p_{ij}, i.e., $Pr(t_i \succ_u t_j) = p_{ij}$, where $0 < p_{ij} \leqslant 1$.*

Such uncertainty can be a result of:

1. different confidence levels of pairwise comparisons given as feedback, where the probability indicates the confidence level, or

2. an aggregation of feedbacks collected from multiple users, where the probability is the percentage of users confirming the corresponding preference.

In either case, it is possible for two incompatible pairwise comparisons to coexist, i.e., $(t_i, t_j, p_{ij}) \in A$ and $(t_j, t_i, p_{ji}) \in A$. In this case, $p_{ij} + p_{ji} \leqslant 1$.

It is possible to generalize the β elicitation problem (Definition 5.1.4) and techniques in Algorithm 7 to incorporate uncertainty in the training dataset. In the generalized elicitation problem in Definition 5.1.6, the intervals derived from pairwise comparisons are now associated with a weight, which is the probability of the corresponding pairwise comparison. The elicitation goal is to find intervals on the real line, where every β value maximizes the weighted total number of satisfiable pairwise comparisons in the training dataset A.

Definition 5.1.6 (Generalized Elicitation Problem). *Assume a simple probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, an injective scoring function s and a probabilistic training dataset A. For every real number β, let $f(\beta)$ be the weighted sum of the number of pairwise comparisons in A whose feasible regions contain β, where the weights are the probabilities associated with the pairwise comparisons. The elicitation problem is to find $\{\beta | \max_{\beta > 0} f(\beta)\}$.*

Algorithm 8 Generalized β Elicitation Algorithm

```
Require: a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, a non-negative integer $k$, a scoring function $s$, a training set $A$ of pairwise comparisons with a probability
Ensure: a collection of intervals where the $\beta$ value maximizes the weighted sum of pairwise comparisons in $A$
 1: Compute the feasible region for each pairwise comparison in $A$ and sort the resulting non-empty intervals into
    $$\{\langle (0, a_1), p_{a_1} \rangle, \ldots, \langle (0, a_n), p_{a_n} \rangle\} \cup \{\langle (b_1, \infty), p_{b_1} \rangle, \ldots, \langle (b_m, \infty), p_{b_m} \rangle\}$$
    such that $0 < a_1 \leqslant \ldots \leqslant a_n$ and $0 < b_1 \leqslant \ldots \leqslant b_m$. Each interval is associated with the probability of the pairwise comparison giving rise to it.
 2: Let $S_a = \{a_1, \ldots, a_n\}$ and $S_b = \{b_1, \ldots, b_m\}$. Let $c_i \in S_a \cup S_b$, $i = 1, \ldots, m + n$ and
    $$0 < c_1 \leqslant \ldots \leqslant c_{m+n}$$
 3: $c_0 = 0$, $c_{m+n+1} = \infty$
 4: $w_a = \sum_{i=1}^{n} p_{a_i}$, $w_b = 0$, $i_a = 1$, $i_b = 1$
 5: $I = \varnothing$
 6: for $i = 1$ to $m + n + 1$ do
 7:   if $c_{i-1} \in S_a$ then
 8:     if $c_i \in S_a$ or $c_i == \infty$ then
 9:       $I_{cur} = [c_{i-1}, c_i)$
10:     else
11:       $I_{cur} = [c_{i-1}, c_i]$
12:     end if
13:   else
14:     if $c_i \in S_a$ or $c_i == \infty$ then
15:       $I_{cur} = (c_{i-1}, c_i)$
16:     else
17:       $I_{cur} = (c_{i-1}, c_i]$
18:     end if
19:   end if
20:   $I = I \cup \{I_{cur}\}$
21:   $f(I_{cur}) = w_a + w_b$
22:   if $c_i \in S_a$ then
23:     $w_a = w_a - p_{a_{i_a}}$, $i_a = i_a + 1$
24:   else if $c_i \in S_b$ then
25:     $w_b = w_b + p_{b_{i_b}}$, $i_b = i_b + 1$
26:   end if
27: end for
28: return $I^* = \max_{f(I_i)}\{I_i | I_i \in I\}$
```

Algorithm 8 illustrates the algorithm for eliciting β via a probabilistic training dataset. Algorithm 8 differs from Algorithm 7 in

1. Line 1, where each interval is associated with a weight, which is the probability of the pairwise comparison where it is derived, and

2. Line 4, Line 23 and Line 25, $f(I_i)$ computes the weighted sum of the number of pairwise comparisons satisfied.

In particular, Algorithm 7 is a special case of Algorithm 8 when $p_{a_i} = p_{b_j} = 1$ for every $i = 1, \dots, n$ and $j = 1, \dots, m$, which suggests the pairwise comparisons in the training set are *certain*, i.e., of probability 1, in the case of Algorithm 7. Notice that the incompatible pairwise comparisons in the training dataset A is not a problem. Their feasible regions do not overlap, and there will not be an interval where we count both incompatible pairwise comparisons.

The runtime of Algorithm 8 follows that of Algorithm 7. It is $O(|A| \log |A|)$ including the sorting.

5.1.5 Experiments

We conduct an experimental study on Global-Topk^β semantics and the parameter β elicitation. We implementation the Global-Topk^β semantics in C++ and run experiments on a machine with Intel Core2 1.66G CPU running Cygwin on Windows XP with 2GB memory.

Each synthetic dataset is a *simple* probabilistic database containing n tuples, where n varies from 100 up to 100K. The size n defaults to 10^4. Each dataset has a uniform random score distribution and a uniform random probability distribution. Scores are normalized so that their values are between 0 and 1. The probabilities and the scores can be *uncorrelated* ($corr = 0$), *negatively* correlated ($corr < 0$) or *positive* correlated ($corr > 0$). The correlation $corr$ between probabilities and scores is a real number in $[-1, 1]$.

The parameter β ranges from 0.1 to 10. The default value of k in a top-k query is 50.

Summary of experiments

We draw the following conclusions from the forthcoming experimental results:

- The top-k result evolves when the β value changes. When β increases, the discrepancy in the top-k results first increases and then decreases. It changes most rapidly when β is at a *peak* value greater than 1. The *peak* shifts to the right when the dataset size increases.

- The top-k result evolves more smoothly when the ratio k/n is high.

- The discrepancy of the top-k results is more significant when the dataset has a high negative correlation. It grows sublinearly with k, and the growth gets slower when the data correlation becomes more positive.

- We experiment with a metric to compare top-k results as sets, as well as a metric to compare top-k results as vectors. By comparing the two metrics, we see that the change in the tuple membership in the top-k result contributes more to the top-k result discrepancy when β increases. This trend is more obvious for datasets of a high negative correlation.

- The elicitation of β value is more accurate when the user has a strong bias in the trade-off of probabilities and scores, which is interpreted as either a very low or a very high β value.

- Both the lack of correlation and a strong negative/positive correlation in the dataset is helpful to the β elicitation.

Distance Measure for Comparing Two Top-k lists

Normalized Minimizing Kendall Distance We need a distance measure to compare two top-k results. The Kendall distance, or sometimes referred to as the *Kemeny* distance in the literature, has been shown to be desirable in comparing rankings in the IR research [22].

We adopt the normalized version of *minimizing Kendall distance* in [24] for this task. Fagin et al. [24] proves that it is in an equivalent class of many other popular measures.

Definition 5.1.7 (Minimizing Kendall Distance [24])**.** *Given a relation R, and two top-k lists τ_1 and τ_2, $K(\tau_1, \tau_2)$ is the* minimizing Kendall distance *between τ_1 and τ_2. Let*

$\tau_{1,2} = \tau_1 \cup \tau_2$.

$$K(\tau_1, \tau_2) = \sum_{(t_i,t_j)\in\tau_{1,2}\times\tau_{1,2}, i<j} \overline{K}_{i,j}(\tau_1, \tau_2)$$

where $\overline{K}_{i,j} = 1$ if it falls in one of the following two cases, $\overline{K}_{i,j} = 0$ otherwise.

Case 1 Both t_i and t_j appear in τ_1 and τ_2, but in opposite order;

Case 2 Both t_i and t_j appear in exactly one top-k list, and only the one ranked lower appears in the other top-k list.

Case 3 Exactly one of t_i and t_j appears in exactly one of τ_1 and τ_2.

Intuitively, $K(\tau_1, \tau_2)$ counts the number of distinct ordered pairs of $\tau_{1,2}$ which are ranked in the opposite order by τ_1 and τ_2. Here, ranking in the opposite order can be either *explicit* (Case 1), or *implicit* (Case 2 and Case 3). In all cases, we can infer that t_i and t_j are ranked in the opposite order in every two permutations of R extending τ_1 and τ_2, respectively. It is called *minimizing* Kendall distance because it is the lower bound of the number of pairs ranked in the opposite order in any two permutations of R extending τ_1 and τ_2, respectively.

The maximum value of $K(\tau_1, \tau_2) = k^2$, which happens when the two top-k lists are disjoint. This is also the denominator used in the *normalized version* of K. We use the *normalized minimizing Kendall distance* as the distance measure in our experiments. K stands for the normalized distance from now on. It is a real number from $[0, 1]$. It reaches 0 when the two top-k lists are identical and reaches 1 when the they are disjoint. Notice that if the two top-k lists contain the same set of tuples but are in the reverse order to each other, then the Kendall distance is $(k(k-1)/2)/(k^2) = (k-1)/(2k) \approx 1/2$. Intuitively, the larger the Kendall distance, the smaller the intersection of the two lists and/or more reverse pairs in the two lists. Similarly, a small Kendall distance suggests a large overlap of the two lists and less reverse pairs in the two lists.

The degree of overlap of two top-k lists can be captured by the classical *Jaccard distance*. The Jaccard distance of two top-k list τ_1 and τ_2 is defined by

$$J(\tau_1, \tau_2) = \frac{|\tau_1 \cup \tau_2| - |\tau_1 \cap \tau_2|}{|\tau_1 \cup \tau_2|}$$

Proposition 5.1.3. *Given two top-k lists τ_1 and τ_2, their normalized Kendall distance $K(\tau_1, \tau_2)$ and their Jaccard distance $J(\tau_1, \tau_2)$ satisfies $J(\tau_1, \tau_2) \leqslant \sqrt{K(\tau_1, \tau_2)}$.*

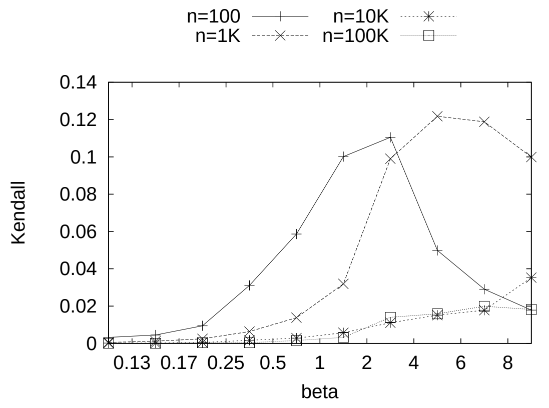
Proof. Assume the two lists τ_1 and τ_2 differ in n_{diff} tuples, then each top-k list has n_{diff} tuples not in the other list. Every ordered pair $(t_i, t_j) \in (\tau_1 - \tau_2) \times (\tau_2 - \tau_1)$ falls in Case 3 of Definition 5.1.7, and therefore has $\overline{K}_{i,j}(\tau_1, \tau_2) = 1$. Therefore, $k^2 K(\tau_1, \tau_2) \geqslant n_{diff}^2$. On the other hand, the Jaccard distance $J(\tau_1, \tau_2) = \frac{n_{diff}}{k}$. Altogether, we have $J(\tau_1, \tau_2) \leqslant \sqrt{K(\tau_1, \tau_2)}$. □

In some experiments, we calculate the Jaccard distance $J(\tau_1, \tau_2)$ as well as the Kendall distance $K(\tau_1, \tau_2)$ in comparing top-k results. This helps us to understand the composition of the Kendall distance better. Roughly speaking, $J^2(\tau_1, \tau_2)$ out of $K(\tau_1, \tau_2)$ is caused by memberships of the top-k list only, while the rest is caused by pairwise orders together with the memberships.

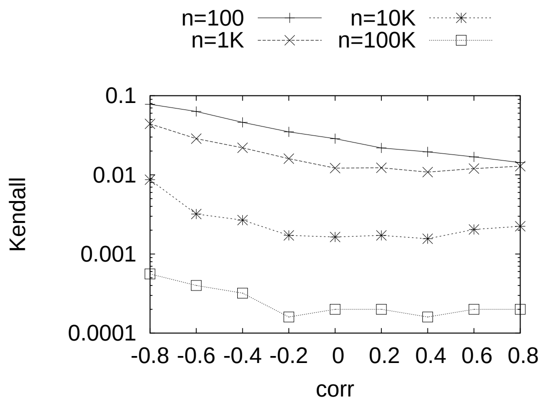
Evolvement of Top-k Results with β

Figure 5-1(a) illustrates the evolvement of top-k results with increasing β values. We compute the Kendall distance of two top-k lists returned by the Global-Topk^β semantics parameterized by consecutive β values from $\{1/10, 1/8, 1/6, 1/4, 1/2, 1, 2, 4, 6, 8, 10\}$. In order to be unbiased with regard to the data correlation, each data point in Figure 5-1(a) is the average Kendall distance of 13 datasets with correlations evenly ranging from -1 to 1.

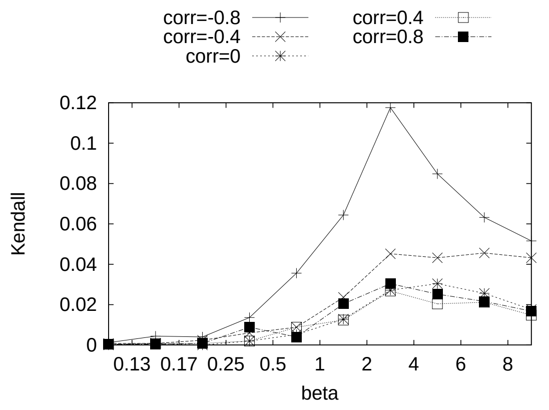
Each curve in Figure 5-1(a) corresponds to a group of datasets of the same size. We observe in Figure 5-1(a) that for datasets of size 100,$1K$ and $10K$, with the increase of β, the Kendall distance increases to a *peak* value and then decreases. The peak value of each curve is a β value to the right of 1, and it shifts to the right with the increase in the data size ($n = 100$, $n = 1K$, $n = 10K$). It seems that the decrease does not happen for $n = 10K$. However, noticing the shift of peak values, it may well be that the peak value of curve $n = 100K$ locates at the further right of the figure and is therefore not displayed. Therefore, Figure 5-1(a) shows that the change in top-k result is most significant when the β value is close to the peak value for datasets of the same size. Both extremely small or large β values have little influence over the top-k results.



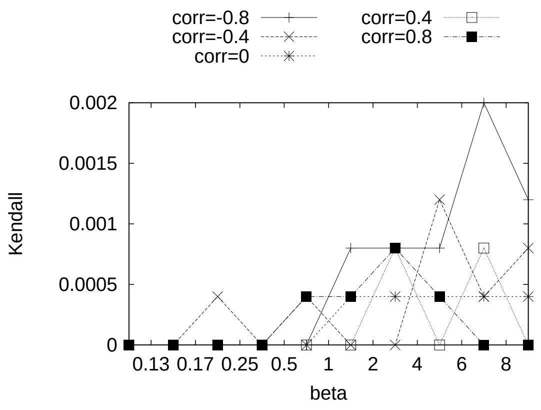
(a) Avg Kendall v.s. β



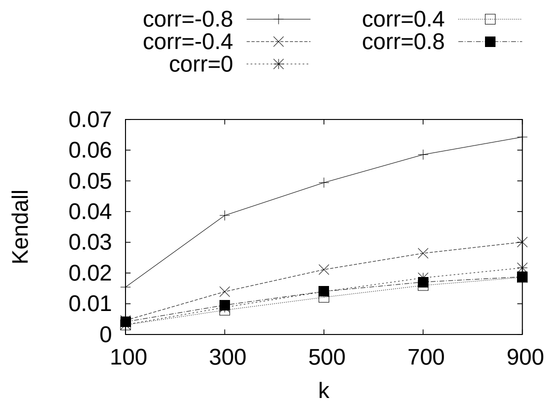
(b) Avg Kendall v.s. corr



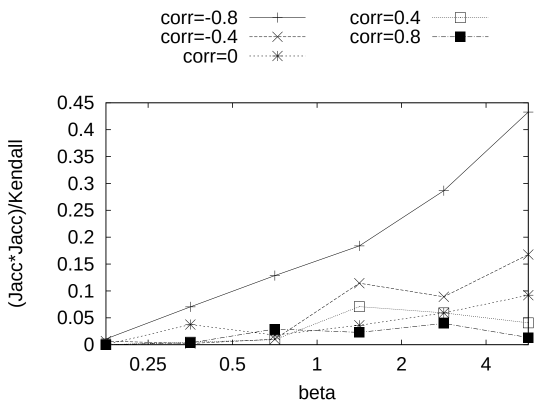
(c) Kendall v.s. β ($n = 10^3$)



(d) Kendall v.s. β ($n = 10^5$)



(e) Avg Kendall v.s. k



(f) Kendall Distance Composition

Figure 5-1: Global-Topk^β Semantics Comparison

We observe in Figure 5-1(a) that the curve corresponding to datasets of a large size is rougher than that corresponding to datasets of a small size. As a proof, we zoom in on the curve $n = 1K$ and the curve $n = 100K$ in Figure 5-1(a). The results are shown in Figure 5-1(c) and Figure 5-1(d), respectively. In both figures, we show only 5 out of the 11 datasets of different correlations for clarity. It is clear now that change in the Kendall distance is smooth in Figure 5-1(c), while it is rough in Figure 5-1(d), where the data size is larger. One explanation is that, as we fix k to 50 in this experiments, the percentage of the top-k result decreases with the increase in data size. For example, in a dataset of size $1K$, the top-k result contains 0.5% of the population, while in a dataset of size $100K$, the top-k result contains merely 0.005% of the population. Therefore, the change in the top-k result is more significant in a large dataset.

Evolvement of Top-k Results v.s. Data Correlation

Figure 5-1(b) illustrates average Kendall distances for datasets of data correlations ranging from -0.8 to 0.8. For each data point in Figure 5-1(b), we average over the Kendall distances of two top-k lists returned by the Global-Topk^{β} semantics parameterized by consecutive β values from $\{1/10, 1/8, 1/6, 1/4, 1/2, 1, 2, 4, 6, 8, 10\}$. The general trend is that the change in top-k results is less significant for datasets where the probability and the score have a higher positive correlation. This trend fits our semantic design nicely. Recall that the Global-Topk^{β} semantics satisfies the *Faithfulness* postulate. In its proof (c.f., Appendix C), we can see that if a tuple t_1 is of both a higher probability and a higher score than those of a tuple t_2, then the Global-Topk^{β} probability of t_1 is higher than that of t_2. In other words, their relative order in the Global-Topk^{β} ranking is predetermined. On the other hand, a high positive correlation between probabilities and scores suggests more such pairs of tuples, and therefore it is less flexibility in producing different top-k results in such datasets.

Evolvement of Top-k Results v.s. k

Figure 5-1(e) illustrates average Kendall distances for top-k queries with k ranging from 100 to 900. We use five datasets, each contains 10^4 tuples and is with correlation evenly ranging from -0.8 to 0.8. Same as before, each data point in Figure 5-1(e) corresponds to the average Kendall distances of two top-k lists returned by the Global-Topk^β semantics parameterized by consecutive β values from $\{1/10, 1/8, 1/6, 1/4, 1/2, 1, 2, 4, 6, 8, 10\}$. In Figure 5-1(e), the average Kendall distance increases with k. All five datasets display a similar sublinear increasing trend, however, the increase gets slower when the positive correlation in the dataset increases.

Evolvement of Top-k Composition

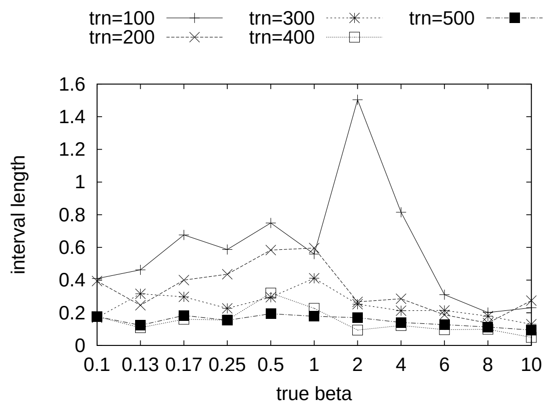
This set of experiments study the composition top-k results. We use five datasets with a correlation ranging from -0.8 to 0.8. For each dataset, we compute the top-k lists returned by the Global-Topk^β with consecutive β values from $\{1/6, 1/4, 1/2, 1, 2, 4, 6\}$. Then, we compute the Kendall distance and the Jaccard distance for every two consecutive top-k lists returned. The result is shown in Figure 5-1(f). The quantity reported in Figure 5-1(f) is J^2/K, i.e. the percentage of the Kendall distance caused by the memberships of the top-k list only. We use datasets of size 10^4 and set $k = 500$ in this set of experiments.

We can see from Figure 5-1(f) that in general the quantity J^2/K increases with the β value regardless of the data correlation. On the other hand, the increase in β implies that more weight is given to the *score* value. Therefore, Figure 5-1(f) shows that a larger fraction of the Kendall distance is derived from the memberships of top-k lists when more weight is given to the score value. In Figure 5-1(f), this trend is more obvious when the probability and the score are of a high negative correlation.

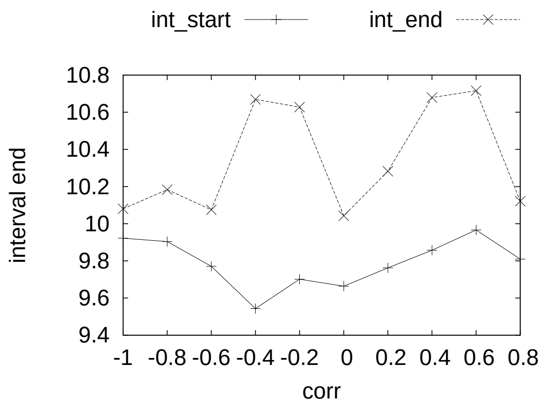
Parameter β Elicitation

Our final set of experiments is on the elicitation of the parameter β. We focus on the generalized parameter β elicitation problem in Algorithm 8, where each pairwise comparison is associated with a probability. The β elicitation problem in Algorithm 7 is a special case of this problem. In this set of experiment, we use a dataset of 1000 tuples and correlation -1. Parameter k is fixed to 20. The elicitation process is set up as follows.

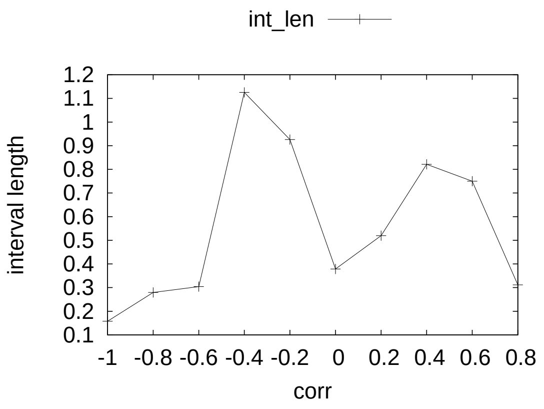
1. Assume the ground truth β value and compute the Global-Topk^β value for each tuple



(a) Avg Interval Length v.s. β



(b) Interval v.s. corr ($\beta = 10$)



(c) Interval Length v.s. corr ($\beta = 10$)

Figure 5-2: β Elicitation

in the dataset;

2. Generate 5 user feedback training datasets. Each contains trn non-trivial and consistent pairwise comparisons, i.e., Case 3 and Case 4 in Section 5.1.4. Each pairwise comparison is associated with a probability randomly chosen from $(0, 1]$;

3. Run the elicitation algorithm (Algorithm 8) with a training set, and compute the length of the interval returned by the algorithm;

4. Take 5 runs of the above step, each time with a different training set, and return the average interval length.

We use the elicited *interval length* to measure the quality of elicitation. A small interval can identify the true β value more accurately than a large interval. Figure 5-2(a) illustrates the results where we experiment with a true β value ranging from $1/10$ to 10, and a training set size trn ranging from 100 to 500. Each curve in Figure 5-2(a) represents a set of experiments using training sets of the same size. It follows our intuition that the larger the training data size, the more accurate the elicitation result, i.e., a smaller the interval length. In addition, the elicitation result of a large training dataset is also more stable than that of a small one in terms of the fluctuation in between consecutive true β values. In general, the elicitation result is more accurate when the true β tends to the two extremes, which follows the intuition that it is easier to elicit when the user's preference is biased towards probabilities or scores.

Figure 5-2(b) and Figure 5-2(c) shows the influence of the data correlation on the elicitation quality. We fix the true $\beta = 10$ and $k = 300$. Figure 5-2(b) shows the average elicited start and end values of the intervals for datasets of different correlations, while Figure 5-2(c) show the corresponding interval length. In Figure 5-2(c), we observe double peaks in the curve. The elicitation is more accurate when the correlation between probabilities and scores are either low ($corr = 0$), or high (negatively or positively). The experiments suggest that user feedbacks are needed when the correlation in the dataset is moderate.

5.1.6 Related Work

The concept of *preference strength* has been discussed under the name of *preference intensity* or *preference difference* in the literature of decision making, utility theory, operations research and psychology. The literature related to our work can be roughly divided into two areas. One studies the decision making under risk/uncertainty and the other studies the preference strength in decision making.

For the former, *risk management* in utility theory [39] discusses how to use *extensive measurement*, i.e. an axiomatic method, to measure *risk*. The results are built on a different uncertain data model. Essentially, given a probabilistic database $R^p = \langle R, p, \mathcal{C} \rangle$, instead of measuring each tuple $t \in R$, this model measures each part $C \in \mathcal{C}$. Choices are made among parts other than tuples. Another theory uses the same model is the *expected utility theory* [39]. For a more recent survey and comparison between those two theories, please refer to Fishburn's survey [26]. Due to the difference in the underlying data model, it is not clear how we can utilize those results in our context.

For the latter, [38] gives a historical review of ordinal v.s. cardinal utility and also simplifies earlier derivations of cardinal utility. It summarizes the context where cardinal utility is usually accepted, and decision making under risk/uncertainty is one of them. It points out that if "preference difference are not rejected per se, then it follows that relatively simple consistency conditions already imply cardinal utility". [18, 2] discusses the problem of eliciting a consensus of preferences given the information of degree of preferences. [54] discusses the application of preference strength in multiple criteria decision making. It also gives a result concerning strength of preference in decision making under risk.

Chapter 6

Set Preferences

In this chapter, we propose a framework to handle *set preferences* [1]. We start by introducing several notions that will be used in this framework, including those inherited from the *tuple preference* framework proposed by Chomicki [16].

6.1 Basic Notions

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 6.1.1 (Tuple Preference [16]). *Given a relation schema $R = \langle A_1, \ldots, A_m \rangle$, a* tuple preference relation $\succ$ *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 then denote the preference relation by $\succ_C$. The* indifference relation $\sim_C$ *generated by $\succ_C$ is*

$$t_1 \sim_C t_2 \Leftrightarrow t_1 \not\succ t_2 \wedge t_2 \not\succ t_1$$

As a binary relation, a preference relation $\succ$ can have typical properties such as *ir-*

[1] An earlier version of some of the results was presented in [59].

reflexivity, *asymmetry*, *transitivity*, *negative transitivity* and *connectivity*. Like other binary relations, a preference relation $\succ$ can be a *strict partial order*, a *weak order*, or a *total order*. Interested readers can consult Section 2.1 for the definitions of various properties and orders of binary relations.

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

Definition 6.1.2 (Winnow Operator [16]). *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: $\omega_C(r) = \{t \in r | \neg\exists t' \in r.t' \succ_C t\}$.*

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

Definition 6.1.3 (Subset Feature). *Given a relation r, a* subset feature $\mathcal{F}(\cdot)$ *is a function: subsets$(r) \mapsto Dom(\mathcal{F})$, where $Dom(\mathcal{F})$ (either $\mathcal{Q}$ or $\mathcal{D}$) is the domain of feature $\mathcal{F}$.*

Definition 6.1.4 (Subset Profile Schema). *Given a relation r, a* subset profile schema Γ *is a schema $\langle \mathcal{F}_1, \dots, \mathcal{F}_m \rangle$, where $\mathcal{F}_i$ is a subset feature, $i = 1, \dots, m$.*

Definition 6.1.5 (Subset Profile Relation). *Given a relation r and its subset profile schema $\Gamma = \langle \mathcal{F}_1, \dots, \mathcal{F}_m \rangle$, the* subset profile relation γ *is defined as*

$$\gamma = \{t | \exists s \in \textit{subsets}(r), t = \langle \mathcal{F}_1(s), \dots, \mathcal{F}_m(s) \rangle\}.$$

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

6.2 Aggregate Feature Definition

In this work, we consider *single-valued* features whose value is a real number, as it is often the case in real applications [21]. This is achieved by defining features as aggregate values in Definition 6.2.1. Other possible features includes boolean features, which we do not consider here.

Definition 6.2.1 (Aggregate Subset Features)**.** *Given a relation r with 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. $Dom(\texttt{\$S}) \subseteq$ 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) the `FROM` *list contains a single item which is* `$S` *or an alias for* `$S`*;*

Example 15. *In Example 3, 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 3-subset of $Book$, since 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 in the feature definition with s. 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 evaluation 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`*.*

The following example illustrates a subset profile schema and relation based on aggregate subset features in Example 15.

Example 16. *Continuing Example 15. Let the subset profile schema* $\Gamma = \langle \mathcal{F}_1, \mathcal{F}_2 \rangle$*. The profile relation* γ *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* $\{a_3, a_7, a_8\}$*.*

6.3 Profile-based Set Preferences

Now we can define set preferences over subsets as tuple preferences over the corresponding profiles. Typically, a tuple preference relation is defined using a first-order formula [16], as is the case for the tuple preference for (C1) in Example 3.

Definition 6.3.1 (Set Preference)**.** *Given a relation schema* $R = \langle A_1, \ldots, A_m \rangle$*, a* set preference relation $\ggcurly$ *is a finite subset of the product* $[\mathit{subsets}(\mathit{Dom}(R))]^2$.

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 6.3.2 (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* $[\mathit{Dom}(\mathcal{F}_1) \times \ldots \times \mathit{Dom}(\mathcal{F}_m)]^2$*. A set preference* $\ggcurly$ *is* based on Γ and $\succ_C$ *if for every set* s_1 *and* s_2*,*

$$s_1 \ggcurly s_2 \Leftrightarrow \mathit{profile}_\Gamma(s_1) \succ_C \mathit{profile}_\Gamma(s_2).$$

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

Proposition 6.3.1. *If a tuple preference relation* $\succ_C$ *is a strict partial order, then for any profile schema* Γ*, the set preference relation* $\ggcurly_{(\Gamma, C)}$ *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. 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.

Definition 6.3.2 provides a general framework for set preferences. As we will see shortly in Section 6.4, we restrict the computation problem to the evaluation of set preferences among subsets of fixed cardinality in this work. In our book purchase running example (Example 3), Alice is buying three books. Example 17 and Example 18 of set preferences build on Example 3, where we essentially work with subsets of fixed cardinality 3.

Example 17. *Assume $\Gamma = \langle \mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3 \rangle$ as in Example 15. We can define the proper preference formula $Ci, i = 1, \ldots, 4$ over Γ, such that individual set preference (C1-C3) is based on Γ and $\succ_{Ci}$. For example, we define $\ggcurly_{(\Gamma, C1)}$ as*

$$\begin{aligned} & s_1 \ggcurly_{(\Gamma, C1)} s_2 \\ \Leftrightarrow\ & \langle \mathcal{F}_1(s_1), \mathcal{F}_2(s_1), \mathcal{F}_3(s_1) \rangle \succ_{C1} \langle \mathcal{F}_1(s_2), \mathcal{F}_2(s_2), \mathcal{F}_3(s_2) \rangle \\ \Leftrightarrow\ & \mathcal{F}_1(s_1) < \mathcal{F}_1(s_2). \end{aligned}$$

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 [16].

Example 18. *Consider the prioritized composition of (C2) and (C1) in Example 3. Let the profile schema $\Gamma = \langle \mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3 \rangle$, and the preference formula $C4$ over Γ be such that*

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

Then $C4$ is the prioritized composition of $C2$ and $C1$.

6.4 Computing the Best k-subsets

We make a decision to work with subsets of fixed cardinality. There are two reasons for this choice. First, fixed cardinality allows us to focus on the composition of a subset and the interaction among the tuples in the subset. The cardinality of a subset might have influence on certain set properties. For example, if we do not limit the cardinality in preference (C1), then we certainly prefer small subsets, since buying fewer books costs less. In this extreme case, the *best* subset would be the empty set which costs $\$0$. However, in reality, this is rarely what we intend. 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 6.4.1 (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 6.4.2 (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*

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

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

6.4.1 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., Definition 6.1.2). In addition to the universal *Nested Loops (NL)* algorithm, many other efficient evaluation algorithms have been proposed when the preference relation is a strict partial order. Among other, there are *Block Nested Loops (BNL)* [7] and *Sort-Filter-Skyline (SFS)* [17].

In our framework, a set preference is formulated as a tuple preference relation $\succ_C$ over a profile schema Γ, which in turn defines a *winnow* operator, i.e. $\omega_C(\Gamma)$. The *best*

k-subsets are computed by *winnowing* over the profile relation γ containing the profiles of all k-subsets of a given relation r.

Algorithm 9 applies *winnow* on a stream of profiles of all k-subsets.

Algorithm 9 Basic Algorithm

```
Require: a profile schema $\Gamma$, a 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 preference $\ggcurly_{(\Gamma,C)}$
 1: Generate all $k$-subsets of relation $r$ and for each compute its profile based on the
    schema $\Gamma$, obtaining the profile relation $\gamma$.
 2: Compute $\gamma' = \omega_C(\gamma)$ using any winnow evaluation algorithm, e.g. BNL [7].
 3: Retrieve the subsets corresponding to the profiles in $\gamma'$.
```

For the generation of candidates k-subsets in Line 1 of Algorithm 9, any sound and complete k-subset generator suffices. We arbitrarily choose a lexicographical k-subset generator[40] which produces k-subsets in the lexicographical order of the tuple indices. An *index vector representation* of a k-subset is the vector of its tuple indices in ascending order. For example, the index vector representation of a 3-subset $\{t_2, t_5, t_1\}$ is $\langle 1, 2, 5\rangle$. The representation is unique for each distinct k-subset. Consequently, the enumeration of k-subsets is equivalent to the enumeration of their index vector representations. Algorithm 10 illustrates the core algorithm used in the lexicographical k-subset generator. It takes the index vector representation of a k-subset and returns the index vector representation of the next k-subset in the lexicographical order (if any).

Algorithm 9 is only practical for a small k; for a large k, the number of k-subsets $\binom{n}{k}$ can be very large, 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 in the basic algorithm, i.e., k-subsets whose profiles will be dominated by other profiles or whose profiles are repeated. We propose two optimization techniques: *superpreference* and *M-relation*. Roughly speaking, *superpreference* can filter out tuples that do not contribute to any best k-subset, and *M-relation* groups together tuples that are *exchangeable* with regard to the given set preference. Both techniques tends to reduce the number of candidate k-subsets

Algorithm 10 k**-subset Lexicographical Successor**

```
Require: a relation $r$, a positive integer $k, k < |r|$ and the index vector representation $\vec{T}$
    of a $k$-subset
Ensure: the index vector representation $\vec{T}_{next}$ of the lexicographical successor of $\vec{T}$
 1: $\vec{T}_{next} = \vec{T}$
 2: $n = |r|$
 3: $i = k$
 4: while $i \geqslant 1$ and $\vec{T}.i == n - k + i$ do
 5:     $i = i - 1$
 6: end while
 7: if i==0 then
 8:     return "no more $k$-subsets"
 9: else
10:     for $j = i$ to $k$ do
11:         $\vec{T}_{next}.j = \vec{T}.i + 1 + j - i$
12:     end for
13:     return $\vec{T}_{next}$
14: end if
```

and therefore speed up the preference query evaluation.

6.4.2 Superpreference

The idea is that, given the 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, and vice versa.

Definition 6.4.3 (*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*

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

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

Proposition 6.4.1. *Given a relation r, a positive integer $k < |r|$ and a set preference relation $\succ\!\!\succ_{(\Gamma,C)}$, for every $s \in$ k-subsets(r),*

$$[\nexists s' \in \textit{k-subsets}(r), s' \succ\!\!\succ_{(\Gamma,C)} s] \Rightarrow [\forall t \in s, cover(t) \subseteq s]. \tag{6.1}$$

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

Equation 6.1 is a necessary condition for a best k-subset. In Algorithm 11, we apply Equation 6.1 in two places:

1. Every tuple t whose $|cover(t)| \geqslant k$ is discarded, as it cannot be in any best k-subsets (Line 2);

2. During the candidate k-subset generation in Line 3, we skip those candidate k-subsets *not* leading to a best k-subset by Proposition 6.4.1. To be more specific, in Line 3 of Algorithm 11, we use a modified version of Algorithm 10 as the lexicographical k-subset generator. This modified version applies a filter after Line 12 in Algorithm 10 to check whether property (6.1) holds for all the new elements generated from Line 10 to Line 12. If not, it repeats Line 1 to Line 12 until it finds the first successor k-subset satisfying property (6.1), or all k-subsets are exhausted.

Algorithm 11 Superpreference Algorithm

```
Require: a profile schema $\Gamma$, a profile preference relation $\succ_C$, a relation $r$ and a positive
    integer $k, k < |r|, \succ^+$ locally defined using $C^+$
Ensure: the best $k$-subsets of $r$ under the set preference $\succ\!\!\succ_{(\Gamma,C)}$
1: Do pairwise comparison between tuples in $r$, and determine $cover(t)$ for each $t \in r$.
2: Let $r' = \{t \in r || cover(t)| < k\}$.
3: Using a modified version of Algorithm 10 to generate 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 $\gamma''$.
```

If the superpreference $\succ^+$ is a weak order, Algorithm 12 can further reduce the input (r') to the lexicographical k-subset generator, which leads to fewer candidate k-subsets.

Algorithm 12 Superpreference Algorithm under Weak Order Superpreference

```
Require: a profile schema $\Gamma$, a profile preference relation $\succ_C$, a relation $r$ and a positive
    integer $k, k < |r|$, $\succ^+$ locally defined using $C^+$
Ensure: the best $k$-subsets of $r$ under the set preference $\ggcurly_{(\Gamma,C)}$
 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 $\gamma''$.
```

In order to illustrate the importance of the weak order requirement in Algorithm 12, let $r_1 = \omega_{C^+}(r), r_2 = \omega_{C^+}(r - r_1), r_3 = \omega_{C^+}(r - r_1 - r_2) \ldots$ so on and so forth until all tuples in r are exhausted. If the superpreference $\succ^+$ is a weak order, then by the definition of weak orders every tuple in $r_1 \cup \ldots \cup r_i$ is superpreferred to every tuple in $r - r_1 - \ldots - r_i$. In other words, every tuple in $r_1 \cup \ldots \cup r_i$ belongs to the cover of every tuple in $r - r_1 - \ldots - r_i$. If $r_1 \cup \ldots \cup r_i$ contains no less than k tuples, we already know that the cover of every tuple in $r - r_1 - \ldots - r_i$ has a cardinality no less than k, and therefore can be discarded. A general strict partial order does not guarantee such a relationship and thus we have to keep track of the covers of individual tuples (Algorithm 11).

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

Definition 6.4.4 introduces an important class of features, i.e., *additive* features. As we will see shortly in this section as well as in Section 6.4.4, the additivity of features enables various efficient optimization techniques.

Definition 6.4.4 (Additive Subset Features). *Given a relation r and a subset feature $\mathcal{F}$, $\mathcal{F}$ is* additive *if for every subset $s \in$ subsets(r), and every $t \in r - s$,*

$$\begin{aligned} \mathcal{F}(s \cup \{t\}) &= \mathcal{F}(s) + f(t) \\ \mathcal{F}(\{t\}) &= f(t) \end{aligned}$$

where f is a function of t only, called the base *of $\mathcal{F}$.*

Proposition 6.4.2. *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}`*,* `A` *is an attribute of r, or a linear combination of constants and the above aggregates*

(2) `simple-condition` *does not contain subqueries*

then $\mathcal{F}$ is additive.

Proof. Under the above conditions, we can show by case study that function $f(t)$ in Definition 6.4.4 always exists. For example, if the aggregate is `sum`, $\mathcal{F}(s \cup \{t\}) = \mathcal{F}(s) + c(t) \cdot t.A$, where A is the quantity of interest in $\mathcal{F}$, i.e. the attribute in `SELECT` clause, and $c(\cdot)$ is an indicator function of the `condition` in the definition of $\mathcal{F}$. An *indicator function* returns 1 when the condition is satisfied, 0 otherwise. $\square$

Theorem 6.4.1. *If a profile-based set preference is defined as a constant-free DNF formula*

$$s_1 \ggcurly_{(\Gamma,C)} s_2 \Leftrightarrow \bigvee_{i=1}^{n} (\bigwedge_{j=1}^{m_i} (\mathcal{F}_{ij}(s_1)\ \theta \mathcal{F}_{ij}(s_2))) \tag{6.2}$$

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

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

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

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

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

Therefore,

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

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'. It can be shown that we can eliminate s' in the above formula, in which case,

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

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

The additive subset features identified in Proposition 6.4.2 are eligible for the rewriting technique in Theorem 6.4.1. However, this rewriting does not work for features defined by min, or max, or avg with non-TRUE WHERE condition, since they are non-additive. In those cases, if we rewrite $\mathcal{F}(s)$ as an expression of s' and t, the term(s) containing variable s' cannot be cancelled 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 Theorem 6.4.1 generates the following inequality:

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

where $b_{ij}(s') = |\{t|t \in s' \wedge c_{ij}(t)\}|$. 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 6.4.1. For the rewriting example in the proof of Theorem 6.4.1, if A_{ij} is $price$, which is always positive, then the rewriting is simplified to

$$c_{ij}(t_1) \wedge (t_1.A_{ij} > t_2.A_{ij} \vee \neg c_{ij}(t_2))$$

Example 19. *In Example 3, 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.

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

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

$\mathcal{F}_6 \equiv$ `SELECT sum(rating) FROM $S`

and $s_1 \succ_{(\Gamma,C)} s_2$ *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*

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

Note that two important classes of preferences skyline [7] and p-skyline [47] can be expressed in the form of the formula in (6.2).

6.4.3 Properties of Superpreference

A Special Case

Proposition 6.4.3. *For a constant-free DNF profile preference formula C (c.f., Equation 6.2), if all features in C are additive , then the superpreference $\succ^+$ constructed preserves the order properties of $\succ_C$ (strict partial order, weak order and total order).*

Proof. Similar to the proof of Theorem 6.4.1, for every $t_1 \succ^+ t_2$ and every $s' \in$ *(k-1)-subsets*$(r \backslash \{t_1, t_2\})$,

$$\bigvee_{i=1}^{n}(\bigwedge_{j=1}^{m_i}(\mathcal{F}_{ij}(s' \cup \{t_1\})\, \theta \mathcal{F}_{ij}(s' \cup \{t_2\})))$$

Since $\mathcal{F}_{ij}$ is additive, we have

$$\begin{aligned} & \mathcal{F}_{ij}(s' \cup \{t_1\})\, \theta \mathcal{F}_{ij}(s' \cup \{t_2\}) \\ \Leftrightarrow\ & (\mathcal{F}_{ij}(s') + f_{ij}(t_1))\, \theta (\mathcal{F}_{ij}(s') + f_{ij}(t_2)) \\ \Leftrightarrow\ & f_{ij}(t_1)\, \theta f_{ij}(t_2) \\ \Leftrightarrow\ & \mathcal{F}_{ij}(\{t_1\})\, \theta \mathcal{F}_{ij}(\{t_2\}) \end{aligned}$$

Therefore, the superpreference formula C^+ is

$$\bigvee_{i=1}^{n}(\bigwedge_{j=1}^{m_i}(\mathcal{F}_{ij}(\{t_1\})\, \theta \mathcal{F}_{ij}(\{t_2\})))$$

which is the exact formula C over singleton subsets.

By induction on k, it is easy to verify that if the formula C represents a strict partial order (weak order, total order, resp.) in the domain of *k-subsets*(r), then it is a strict partial order (weak order, total order, resp.) in the domain of 1-subsets(r), i.e., singleton subsets of r. The conclusion follows. □

In Proposition 6.4.3, the superpreference $\succ^+$ is by itself order-preserving. The integration of any domain knowledge might change the order property of $\succ^+$. For example, in Example 19, the set preference $\succ_C$ is a weak order while the superpreference $\succ^+$ is not. It is due to the fact that we integrate the domain knowledge $price > 0$ in $\succ^+$.

General Superpreference

Proposition 6.4.3 states the property of superpreference in a special case. In general, strict partial order is preserved by the superpreference rewriting, while weak order and total order are not.

Proposition 6.4.4. *For any set preference relation $\succ\!\!\succ_{(\Gamma,C)}$, if the profile preference relation $\succ_C$ is a strict partial order (i.e. irreflexive and transitive), then the corresponding superpreference relation $\succ^+$ is a strict partial order as well.*

Proof. If $\succ^+$ is empty, then it is trivially true. Otherwise, we need to show that $\succ^+$ is irreflexive and transitive. Given the irreflexitivity of $\succ_C$, the irreflexitivity of $\succ^+$ can be easily shown by contradiction. Here, we only prove the transitivity of $\succ^+$.
We need to show that $t_1 \succ^+ t_2 \wedge t_2 \succ^+ t_3 \Rightarrow t_1 \succ^+ t_3$.

$$\begin{aligned} t_1 \succ^+ t_2 \quad &\Leftrightarrow \quad t_1 \in r \wedge t_2 \in r \wedge [\forall s \in \textit{(k-1)}\text{-subsets}(r\backslash\{t_1,t_2\}), \\ &\qquad s \cup \{t_1\} \succ\!\!\succ_{(\Gamma,C)} s \cup \{t_2\}] \qquad (6.3) \\ t_2 \succ^+ t_3 \quad &\Leftrightarrow \quad t_2 \in r \wedge t_3 \in r \wedge [\forall s \in \textit{(k-1)}\text{-subsets}(r\backslash\{t_2,t_3\}), \\ &\qquad s \cup \{t_2\} \succ\!\!\succ_{(\Gamma,C)} s \cup \{t_3\}] \qquad (6.4) \end{aligned}$$

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

Case 1: $t_2 \notin s$

By (6.3), (6.4) and the transitivity of $\succ_C$, we have $s \cup \{t_1\} \succ\!\!\succ_{(\Gamma,C)} s \cup \{t_3\}$.

Case 2: $t_2 \in s$

Let $s' = s\backslash\{t_2\}$, by (6.3),

$$s' \cup \{t_3\} \cup \{t_1\} \succ\!\!\succ_{(\Gamma,C)} s' \cup \{t_3\} \cup \{t_2\} \qquad (6.5)$$

By (6.4),

$$s' \cup \{t_1\} \cup \{t_2\} \succ\!\!\succ_{(\Gamma,C)} s' \cup \{t_1\} \cup \{t_3\} \qquad (6.6)$$

By (6.5), (6.6) and the transitivity of $\succ_C$, we have $s \cup \{t_1\} \ggcurly_{(\Gamma,C)} s \cup \{t_3\}$.

□

Proposition 6.4.5. *For some weak order profile preference relation $\succ_C$ (i.e., irreflexive, transitive and negatively transitive) over a relation r, the corresponding superpreference relation $\succ^+$ is not a weak order.*

Proof. We show a counterexample here. S is a relation with the schema $\langle B_1, B_2 \rangle$. The profile relation $\Gamma = \langle \mathcal{F}_7, \mathcal{F}_8 \rangle$. We are interested in k-subsets, where $k = 2$.

$\mathcal{F}_7 \equiv$ `SELECT count(`B_1`) % 2 FROM $S WHERE `B_1`=TRUE`

$\mathcal{F}_8 \equiv$ `SELECT sum(`B_2`) FROM $S`

and the set preference is

$s_1 \ggcurly_{(\Gamma,C)} s_2$ iff $\mathcal{F}_7(s_1) > \mathcal{F}_7(s_2) \vee (\mathcal{F}_7(s_1) = \mathcal{F}_7(s_2) \wedge \mathcal{F}_8(s_1) > \mathcal{F}_8(s_1))$

In other words, the set preference is a prioritized composition of the two (weak order) set preferences preferring larger values of $\mathcal{F}_7$ and $\mathcal{F}_8$, respectively. Chomicki [16] shows that prioritized composition preserves weak order. Therefore, it is easy to see that the set preference C is a weak order. For the superpreference computation, Theorem 6.4.1 is not applicable because $\mathcal{F}_7$ is not additive. On the other hand, the feature $\mathcal{F}_8$ is additive.

In this particular example, we are able to compute the superpreference $\succ^+$ from Definition 6.4.3. In the following equation, $c_7(\cdot)$ is the indicator function corresponding to the feature $\mathcal{F}_7$. That is, for every tuple t, $c_7(t) = 1$ if $t.B_1 =$`TRUE`, otherwise $c_7(t) = 0$.

$$\begin{aligned}
t_1 \succ^+ t_2 \quad \Leftrightarrow \quad & t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)}\text{-subsets}(r \backslash \{t_1, t_2\}), \\
& s' \cup \{t_1\} \ggcurly_{(\Gamma,C)} s' \cup \{t_2\}] \\
\Leftrightarrow \quad & t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)}\text{-subsets}(r \backslash \{t_1, t_2\}), \\
& \mathcal{F}_7(s') \text{ xor } c_7(t_1) > \mathcal{F}_7(s') \text{ xor } c_7(t_2) \\
& \vee (\mathcal{F}_7(s') \text{ xor } c_7(t_1) = \mathcal{F}_7(s') \text{ xor } c_7(t_2) \\
& \quad \wedge \mathcal{F}_8(s') + f_8(t_1) > \mathcal{F}_8(s') + f_8(t_2))]
\end{aligned}$$

where f_8 is the function corresponding to the additive feature $\mathcal{F}_8$ in Definition 6.4.4.

- Case 1: $c_7(t_1) = c_7(t_2)$

  For every s', $\mathcal{F}_7(s')$ xor $c_7(t_1) > \mathcal{F}_7(s')$ xor $c_7(t_2)$ is always evaluated to `FALSE`. Therefore,

  $$\begin{aligned} t_1 \succ^+ t_2 \ &\Leftrightarrow\ t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)}\text{-subsets}(r \backslash \{t_1, t_2\}), \\ &\qquad \mathcal{F}_7(s') \text{ xor } c_7(t_1) = \mathcal{F}_7(s') \text{ xor } c_7(t_2) \\ &\qquad \wedge \mathcal{F}_8(s') + f_8(t_1) > \mathcal{F}_8(s') + f_8(t_2)] \\ &\Leftrightarrow\ t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)}\text{-subsets}(r \backslash \{t_1, t_2\}), \\ &\qquad c_7(t_1) = c_7(t_2) \wedge f_8(t_1) > f_8(t_2)] \end{aligned}$$

  By expanding the definition of function c_7 and function f_8, we have

  $$t_1 \succ^+ t_2 \Leftrightarrow t_1.B_1 = t_2.B_1 \wedge t_1.B_2 > t_2.B_2 \tag{6.7}$$

- Case 2: $c_7(t_1) \neq c_7(t_2)$

  For every s', $\mathcal{F}_7(s')$ xor $c_7(t_1) = \mathcal{F}_7(s')$ xor $c_7(t_2)$ is always evaluated to `FALSE`. Therefore,

  $$\begin{aligned} t_1 \succ^+ t_2 \ &\Leftrightarrow\ t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)}\text{-subsets}(r \backslash \{t_1, t_2\}), \\ &\qquad \mathcal{F}_7(s') \text{ xor } c_7(t_1) > \mathcal{F}_7(s') \text{ xor } c_7(t_2)] \end{aligned}$$

  As long as the relation r is not extremely limited, i.e., there are at least two (k-1)-subsets with different $\mathcal{F}_7$ values, then for some s', $\mathcal{F}_7(s')$ xor $c_7(t_1) > \mathcal{F}_7(s')$ xor $c_7(t_2)$ is evaluated to `FALSE`. Hence, there is no superpreference relationship between such t_1 and t_2.

It is easy to verify that the superpreference $\succ^+$ defined by Equation 6.7 is not a weak order. □

Proposition 6.4.6. *For some total order profile preference relation $\succ_C$ (i.e., irreflexive, transitive and connected) over a relation r, the corresponding superpreference relation $\succ^+$*

is not a total order.

Proof. Given a total order set preference, its superpreference can be empty. For example, in the counterexample used in the proof of Proposition 6.4.5, assume the set preference C is instead

$s_1 \ggcurly_{(\Gamma,C)} s_2$ iff $\mathcal{F}_7(s_1) > \mathcal{F}_7(s_2) \vee (\mathcal{F}_7(s_1) = \mathcal{F}_7(s_2) = 0 \wedge \mathcal{F}_8(s_1) > \mathcal{F}_8(s_2)) \vee (\mathcal{F}_7(s_1) = \mathcal{F}_7(s_2) = 1 \wedge \mathcal{F}_8(s_1) < \mathcal{F}_8(s_2))$

It is easy to verify that the set preference is a total order. The superpreference $\succ^+$ is empty. This is because for any two tuples $t_1, t_2 \in r$ and any (k-1)-subset $s \in r \backslash \{t_1, t_2\}$, the comparison between $s \cup \{t_1\}$ and $s \cup \{t_2\}$ depends on $t_1.B_1$, $t_2.B_2$ and the number of tuples with TRUE B_1 value in s. For any relation with sufficiently many tuples and a non-extreme B_1 value distribution, i.e., $|r| \geqslant \max(4, k+2)$ and at least two tuples of TRUE B_1 values and at least two tuples of FALSE B_1 values, there are choices of s having an even number of tuples with TRUE B_1 value, as well as choices of s having an odd number of tuples with TRUE B_1 values. Consequently, there is not an unanimous order between $s \cup \{t_1\}$ and $s \cup \{t_2\}$ for every (k-1)-subset $s \in r \backslash \{t_1, t_2\}$. □

6.4.4 M-relation

Superpreference is a pruning technique in order to filter out tuples which do not contribute 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 during generation. Besides the *inferior* k-subsets, we often observe *redundant* candidate k-subsets during generation, as in Example 20.

Example 20. *Assume we add two more tuples to the* Book *relation in Example 3*

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

and we have the same set preference $\ggcurly_{(\Gamma,C)}$ *as that in Example 19. That is,* $\Gamma = \langle \mathcal{F}_5, \mathcal{F}_6 \rangle$

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

$\mathcal{F}_6 \equiv$ `SELECT sum(rating) FROM $S`

The tuple a_7 *and* a_9 *are exchangeable with regard to the set preference, because for every* 2*-subset* s *of* `Book`$\backslash\{a_7, a_9\}$*, extending* s *with* a_7 *or* a_9 *leads to the same profile in the profile relation, i.e.,* $profile_\Gamma(s \cup \{a_7\}) = profile_\Gamma(s \cup \{a_9\})$*. By the same argument,* a_7*,* a_9 *and* a_{10} *are mutually exchangeable.*

Example 20 illustrates possible redundancy in the k-subset generation. 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. 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.

Based on the above idea, we define an *exchangeability relation* among tuples in Definition 6.4.5.

Definition 6.4.5 (Exchangeability Relation)**.** *Given a relation* r*, a positive integer* $k < |r|$ *and a set preference relation* $\ggcurly_{(\Gamma,C)}$*, an equivalence relation* $\sim_{(\Gamma,C)}$ *is an* exchangeability relation *over* r *if*

$$t_1 \sim_{(\Gamma,C)} t_2 \;\Rightarrow\; t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)-subsets}(r\backslash\{t_1,t_2\}), \quad (6.8)$$
$$profile_{\Gamma_{[m_b]}}(s' \cup \{t_1\}) = profile_{\Gamma_{[m_b]}}(s' \cup \{t_2\})]$$

Tuple t_1 *and* t_2 *are* exchangeable *iff* $t_1 \sim_{(\Gamma,C)} t_2$.

Notice the superficial similarity between Definition 6.4.5 and Definition 6.4.3. It is easy to show that the exchangeability relation in Definition 6.4.5 is different from the indifference relation induced by the superpreference relation in Definition 6.4.3.

Also notice that, given a set preference, there can be more than one exchangeability relation according to Definition 6.4.5. 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.

An exchangeability relation $\sim_{(\Gamma,C)}$ over r is *optimal* if and only if it contains every other exchangeability relation. That is, its corresponding partition of r is not a *refinement* of the partition of any other exchangeability relation. Formally,

Definition 6.4.6 (Optimal Exchangeability Relation)**.** *An exchangeability relation* $\sim_{(\Gamma,C)}$ *over the relation* r *is* optimal *if and only if*

$$t_1 \sim_{(\Gamma,C)} t_2 \;\Leftrightarrow\; t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)-subsets}(r\backslash\{t_1,t_2\}), \quad (6.9)$$
$$profile_{\Gamma_{[m_b]}}(s' \cup \{t_1\}) = profile_{\Gamma_{[m_b]}}(s' \cup \{t_2\})]$$

The optimal exchangeability relation is an equivalence relation.

Equation 6.9 defines the maximal exchangeability relation in terms of partition containment as the optimal one. It differs from Equation 6.8 in that it requires a sufficient condition as well.

Example 21. *For the relation in Example 20, assume the same set preference* $\succ_{(\Gamma,C)}$ *in Example 19, i.e., prefer spending less money on sci-fi books and higher book ratings. By*

Definition 6.4.5, one exchangeability relation and its partition are

$$\begin{aligned}\sim_0 &= \{(a_i, a_i) | i = 1, \ldots, 10\} \\ P_0 &= \{\{a_i\} | i = 1, \ldots, 10\}\end{aligned}$$

Another exchangeability relation and its partition are

$$\begin{aligned}\sim_1 &= \{(a_i, a_i) | i = 1, \ldots, 10\} \cup \\ &\quad \{(a_7, a_9), (a_9, a_7), (a_7, a_{10}), (a_{10}, a_7), (a_9, a_{10}), (a_{10}, a_9)\} \\ P_1 &= \{\{a_i\} | i = 1, \ldots, 6, 8\} \cup \{\{a_7, a_9, a_{10}\}\}\end{aligned}$$

The exchangeability relation $\sim_0$ is the equality relation. Since P_0 is a refinement of P_1, i.e., $P_0 \subset P_1$, the exchangeability relation $\sim_0$ is not optimal. It is easy to verify that $\sim_1$ is optimal since the merge of any equivalence classes in partition P_1 does not lead to an exchangeability relation over the Book *relation.*

We introduce a profile consolidation optimization using *M-relations*. Given an exchangeability relation $\sim_{(\Gamma,C)}$, an M-relation contains *M-tuples*, and is such that there is a one-one mapping between its *M-tuples* and *parts* in the partition of $\sim_{(\Gamma,C)}$.

There are various ways to define an M-relation corresponding to an exchangeability relation. In Definition 6.4.7, we give one way of defining an M-relation using SQL. As we will in Theorem 6.4.2, such M-relations corresponding to the *optimal* exchangeability relation under certain conditions.

Before we introduce how to define an M-relation using SQL, we first classify the attributes in a profile schema into three categories, based on their additivity and their involvement in the set preference. Without loss of generality, the profile schema in a set preference $\gg_{(\Gamma,C)}$ 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 involved in the profile preference

formula C, denote by $\Gamma_{[m_b]}$. The features $\mathcal{F}_{m_{b+1}}, \ldots, \mathcal{F}_m$ are not involved in C.

(2) The features $\mathcal{F}_1, \ldots, \mathcal{F}_{m_a}$ are additive (c.f., Definition 6.4.4).

In Definition 6.4.7, 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 calculation of additive features over sets: in Definition 6.4.7, g_i is either f_i, the base of the additive feature $\mathcal{F}_i$, or f_i with a default value, where $i = 1, \ldots, m_a$. This procedure is expressed using a subquery and a group-by clause in Definition 6.4.7.

The M-relation also keeps track of the number of tuples consolidated into an M-tuple, which is the special attribute A_{cnt} in Definition 6.4.7.

Definition 6.4.7 (*M-relation* using SQL). *Given a relation r with schema R, a non-negative integer k and a set preference $\ggcurly_{(\Gamma,C)}$, 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 $g_1$(R) AS $A_1$, ..., $g_{m_a}$(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. g_i(R), $i = 1, \ldots, m_a$ *is either f_i, or f_i with a default value defined by the following* `CASE` *statement:*

```
CASE
  WHEN condition THEN $f_i$(R)
  ELSE default_value
END
```

3. `attrs`$(\mathcal{F}_{m_a+1}, \ldots, \mathcal{F}_{m_b}) = \{A_{m_a+1}, \ldots, A_{m_c}\}$, *i.e., the set of attributes mentioned in the definitions of the features $\mathcal{F}_{m_a+1}, \ldots, \mathcal{F}_{m_b}$;*

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 22. *Continuing Example 20, the M-relation schema is determined by the set preference. For the set preference in Example 20, whose definition can be found in Example 19, its M-relation schema is $\langle A_5, A_6, A_{cnt}\rangle$. Just for illustration purpose, 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 A5, r.rating AS A6, r.vendor AS A3, count(*) AS Acnt
FROM r
GROUP BY A5,A6,A3
```

In the following M-relation o', the subscripts of each M-tuple are the indices of 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

The actual M-relation o for Example 20 happens to be the above o' relation with the attributes $\langle A_5, A_6, A_{cnt}\rangle$ only.

In Example 22, we use a `CASE` statement for a feature definition in the M-relation. In the `CASE` statement, the `ELSE` statement gives a *default* value for the new feature if the

`WHEN` condition is not satisfied. The default value is 0 for A_5 in Example 22. In general, it can be any constant.

The purpose of M-relations is to identify *exchangeable* tuples as illustrated by Example 20. In Definition 6.4.5, we formally define the *exchangeability* of tuples under a set preference.

Theorem 6.4.2. *For a constant-free DNF profile preference formula C (c.f., Equation 6.2) and a relation r, if all features in C are additive, then the exchangeability relation corresponding to the M-relation in Definition 6.4.7 is optimal over r.*

Proof. The optimal exchangeability relation is

$$\begin{aligned} t_1 \sim_{(\Gamma,C)} t_2 \quad \Leftrightarrow \quad & t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)}\text{-subsets}(r \backslash \{t_1, t_2\}), \\ & profile_{\Gamma_{[m_b]}}(s' \cup \{t_1\}) = profile_{\Gamma_{[m_b]}}(s' \cup \{t_2\})] \\ \Leftrightarrow \quad & t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)}\text{-subsets}(r \backslash \{t_1, t_2\}), \forall i, i = 1, \ldots, m_b, \\ & \mathcal{F}_i(s' \cup \{t_1\}) = \mathcal{F}_i(s' \cup \{t_2\})] \end{aligned}$$

Since all the features in formula C are additive, $m_b = m_a$, and for each additive feature $\mathcal{F}_i, i = 1, \ldots, m_a$,

$$\begin{aligned} & \mathcal{F}_i(s' \cup \{t_1\}) = \mathcal{F}_i(s' \cup \{t_2\}) \\ \Leftrightarrow \quad & \mathcal{F}_i(s') + f_i(t_1) = \mathcal{F}_i(s') + f_i(t_2) \\ \Leftrightarrow \quad & f_i(t_1) = f_i(t_2) \end{aligned}$$

Therefore,

$$\begin{aligned} t_1 \sim_{(\Gamma,C)} t_2 \quad \Leftrightarrow \quad & t_1 \in r \wedge t_2 \in r \wedge [\forall s' \in \textit{(k-1)}\text{-subsets}(r \backslash \{t_1, t_2\}), \forall i, i = 1, \ldots, m_a, \\ & f_i(t_1) = f_i(t_2)] \\ \Leftrightarrow \quad & t_1 \in r \wedge t_2 \in r \wedge [\forall i = 1, \ldots, m_a, f_i(t_1) = f_i(t_2)] \end{aligned}$$

This is exactly the partition we get from the SQL query in Definition 6.4.7. It is crucial that all the features in formula C are additive. Otherwise, we do not have a base of a non-additive feature. □

Example 23. *In Example 22, the partition generated by the M-relation 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 Equation 6.9.*

6.4.5 Computing Profiles via M-relations

Our goal is to compute the profiles from an M-relation directly. 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 corresponds to tuples t_1 and t_2 in the original relation r. We therefore 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, since a k-subset of the M-relation contains at most one M-tuple m_1 and $\{m_1, m_1\}$ 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* (Definition 6.4.8) of an M-relation.

Definition 6.4.8 (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 s of o and $|s| = k$, where 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.*

Theorem 6.4.3 states we can compute the projection of the profile relation γ to the relevant features by evaluating those features of k-multisubsets.

It is crucial to the k-multisubset generation that the M-relation records the number of tuples consolidated into each M-tuple. For example, in Example 20, assume a_1 is the only tuple consolidated to the M-tuple m_1, then the 3-multisubset $\{m_1, m_{7,9,10}, m_{7,9,10}\}$ represents 3 3-subsets of the original relation: a_1 and any two tuples from $\{a_7, a_9, a_{10}\}$. The subtlety is that in the k-subset generation of the original tuples, each tuple can appear at most once in a candidate k-subset, while in the k-multisubset generation, each M-tuple can appear multiple times in a candidate k-multisubset. 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.

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

Theorem 6.4.3. *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 $profile_{\Gamma_{[m_b]}}(s) = profile_{\Gamma_{[m_b]}}(s')$, and vice versa.*

Proof. ($\Rightarrow$) The SQL query defining the M-relation o creates a surjection $g : r \to o$. For each $s \in$ *k-subsets*(r), without loss of generality, assume $s = \{t_1, \dots, t_k\}$. Let $g(s) = \{g(t_1), \dots, g(t_k)\} = s'$, and $s' \in$ *k-multisubsets*(o).

By Definition 6.4.4, for each additive feature $\mathcal{F}_i$, $i = 1, \dots, m_a$, $\mathcal{F}_i(s) = \mathcal{F}_i(s \backslash \{t_k\}) + f_i(t_k)$, $\mathcal{F}_i(s) = \sum_{j=1}^{k} f_i(t_j)$. On the other hand, following the definition of the M-relation (Definition 6.4.7), for the k-multisubset s', we have $\mathcal{F}_i(s') = \sum_{j=1}^{k} g(t_j).A_i = \sum_{j=1}^{k} f_i(t_j)$. Therefore, $\mathcal{F}_i(s') = \mathcal{F}_i(s)$.

For each non-additive relevant feature $\mathcal{F}_i$, $i = m_a+1, \dots, m_b$, $\mathcal{F}_i(s') = \mathcal{F}_i(\{g(t_1), \dots, g(t_k)\})$. By Definition 6.4.7, each tuple in s and its image in s' have the same value on attributes in $\mathcal{F}_i$. That is, $t_j.A_l = g(t_j).A_l$ where $j = 1, \dots, k$ and $l = m_a + 1, \dots, m_c$. Therefore, $\mathcal{F}_i(s') = \mathcal{F}_i(\{t_1, \dots, t_k\}) = \mathcal{F}_i(s)$.

($\Leftarrow$) It is sufficient to show that for each $s \in$ *k-multisubsets*(o), there is a $s' \in$ *k-subsets*(r) such that $g(s') = s$. Rewrite s as $\{m_1 : c_1, \dots, m_{k'} : c_{k'}\}$, $k' \leqslant k$, where $m_j : c_j$,

$j = 1, \ldots, k'$, denotes that $m_j \in o$ repeats c_j times in s. Construct s' as the union of any c_j tuples from $\{t \in r | g(t) = m_j\}$ for $j = 1, \ldots, k'$. It is easy to verify that $g(s') = s$. $\square$

Theorem 6.4.3 guarantees that the projection of the original profile relation computed from the M-relation is *sound* and *complete*. Algorithm 13 uses M-relations to compute the best k-subsets. Algorithm 13 is similar to Algorithm 9 except that it works on the M-relation and uses a k-multisubset generator.

Algorithm 13 M-relation Algorithm

```
Require: a profile schema $\Gamma$, a 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 preference $\ggcurly_{(\Gamma,C)}$
1: Compute the M-relation $o$ by Definition 6.4.7.
2: Generate all $k$-multisubsets of $o$ and for each compute the profile features in $\Gamma$ relevant
    to $C$, i.e.,$\Gamma_{[m_b]}$, obtaining $\gamma'$, the projection of profile relation $\gamma$ onto $\Gamma_{[m_b]}$.
3: Compute $\gamma'' = \omega_C(\gamma')$ using any winnow evaluation algorithm.
4: Retrieve the subsets whose profile projections correspond to the elements of $\gamma''$.
```

We present in Algorithm 14 the core algorithm used in a k-multisubset generator. An index vector representation of a k-multisubset is the vector of its tuple indices in the non-decreasing order. The lexicographical order of k-multisubsets is defined as the lexicographical order of their index vector representations. Algorithm 14 is a generalization of Algorithm 10.

Example 24. *Continuing Example 22, the following table gives a few examples of candidate k-subsets in* `Book` *and their corresponding k-multisubsets in the M-relation o, together with the corresponding profiles in the profile relation γ.*

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

In fact, a k-subset generator of `Book` *will enumerate all $\binom{10}{3} = 120$ candidate k-subsets ($k = 3$), 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.*

Algorithm 14 k**-multisubset Lexicographical Successor**

```
Require: an M-relation $o$, a positive integer $k$, $k < \sum_{m_i \in o} m_i.A_{cnt}$ and the index vector
    representation $\vec{T}$ of a $k$-multisubset
Ensure: the index vector representation $\vec{T}_{next}$ of the lexicographical successor of $\vec{T}$
 1: {compute the last $k$-multisubset in the lex order}
 2: $id = |o|$
 3: for $j = k$ to $1$ do
 4:    {$m_{id}$ is the M-tuple with index $id$ from M-relation $o$}
 5:    $l = 1$
 6:    while $j > 0$ and $l \leqslant m_{id}.A_{cnt}$ do
 7:       $\vec{T}_{last}.j = id$
 8:       $j = j - 1$
 9:       $l = l + 1$
10:    end while
11:    $id = id - 1$
12: end for
13: {compute the next $k$-multisubset in the lex order}
14: $\vec{T}_{next} = \vec{T}$
15: $i = k$
16: while $i \geqslant 1$ and $\vec{T}.i == \vec{T}_{last}.i$ do
17:    $i = i - 1$
18: end while
19: if i==0 then
20:    return "no more $k$-subsets"
21: else
22:    $id = \vec{T}.i + 1$
23:    for $j = i$ to $k$ do
24:       $l = 1$
25:       while $j \leqslant k$ and $l \leqslant m_{id}.A_{cnt}$ do
26:          $\vec{T}_{next}.j = id$
27:          $j = j + 1$
28:          $l = l + 1$
29:       end while
30:       $id = id + 1$
31:    end for
32:    return $\vec{T}_{next}$
33: end if
```

By Definition 6.4.7, for k-multisubsets of the M-relation o, the definition of feature $\mathcal{F}_5$ is

```
SELECT sum(A_5) FROM $S
```

and the definition of feature $\mathcal{F}_6$ is

```
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 $\mathcal{F}_6(s) = m_1.A_6 + m_2.A_6 + m_{7,9,10}.A_6 = 5.0 + 4.8 + 4.0 = 13.8$. Therefore, the profile of the multiset s is $(\$15.00, 13.8)$.

The profile relation Γ contains 64 distinct profiles. In this particular example, the k-multisubset generator eliminates all redundancy in the set 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 saves significant number of sets generated in most cases.

Example 24 illustrates the savings achieved by the M-relation. In general, 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 Definition 6.4.7, 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 tuple not satisfying the `WHERE` condition of the feature definition will contribute a *default* value in the feature evaluation. Recall that the M-relation definition query in Example 22 has default value 0 for attribute A_5. Such default values fully exploit the selectivity of the `WHERE` condition. A highly selective `WHERE` condition will lead to more tuples with the default value, and therefore result in a smaller M-relation.

(3) Non-relevant attributes.

The projection to attributes relevant to the set preference reduces the redundancy in k-subset generation.

6.4.6 Hybrid

Superpreference and *M-relation* target two different aspects of pruning: *superpreference* filters out tuples which cannot be part of any best k-subsets, while *M-relation* consolidates exchangeable tuples and process k-subsets in groups. It is natural to combine them to achieve further 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 $\ggcurly_{(\Gamma,C)}$.

- (SM) *Superpreference* followed by *M-relation*. (Algorithm 15)

Algorithm 15 Hybrid_SM Algorithm

```
Require: a profile schema $\Gamma$, a profile preference relation $\succ_C$, a relation $r$ and a positive
    integer $k, k < |r|$, $\succ^+$ locally defined using $C^+$
Ensure: the best $k$-subsets of $r$ under the set preference $\ggcurly_{(\Gamma,C)}$
1: Do pairwise comparison between tuples in $r$, and determine $cover(t)$ for each $t \in r$.
2: Let $r' = \{t \in r | |cover(t)| < k\}$.
3: Return the result of the M-relation Algorithm (Algorithm 13) with the input relation $r'$
    and the profile preference relation $\succ_C$ restricted to $r'$ instead.
```

- (MS) *M-relation* followed by *superpreference*. (Algorithm 16)

In this algorithm, we apply the superpreference technique to the M-relation. The difference between computing profiles from the original relation r and from its M-relation o is that we use k-multisubsets instead of k-subsets in the latter case. The profiles computed from the M-relation are the projection of the original profiles to the relevant features. The profile computation of a k-multisubset is straightforward. For an additive feature, simply sum up the values of the corresponding attribute of every M-tuples in the k-multisubset. For a non-additive feature, the M-relation schema has every attribute involved in this feature and we compute its value for the k-multisubset 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 Definition 6.4.3, a tuple t_1 is superpreferred to a tuple t_2 if and only if 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 similar: an M-tuple m_1 is superpreferred to an M-tuple m_2 if and only if 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 m' dominating m under $\succ_o^+$ with each such m' occurring $m'.A_{cnt}$ times, i.e., $cover(m) = \{m' \in o, m' \text{ repeats } m'.A_{cnt} \text{ times} | m' \succ_o^+ m\}$.

Algorithm 16 Hybrid_MS Algorithm

```
Require: a profile schema $\Gamma$, a profile preference relation $\succ_C$, a relation $r$ and a positive
    integer $k, k < |r|$, $\succ^+$ locally defined using $C^+$
Ensure: the best $k$-subsets of $r$ under the set preference $\ggcurly_{(\Gamma,C)}$
1: Compute the M-relation $o$ of the relation $r$.
2: Compute the superpreference relation $\succ_o^+$ in the M-relation $o$.
3: Compute $o' = \{m \in o || cover(m)| < k\}$, where $cover(m) = \{m' \in o \text{ repeats } m'.A_{cnt} \text{ times} | m' \succ_o^+ m\}$.
4: Retrieve relation $r' \subseteq r$ contributing to the M-relation $o'$.
5: Return the result of the M-relation Algorithm (Algorithm 13) with the input relation $r'$
    and the profile preference relation $\succ_C$ restricted to $r'$ instead.
```

6.5 Experiments

We reported here an experimental study of the performance of various query evaluation algorithms proposed. We implemented all the 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 Algorithm 9. SUPER and MREL are the implementation of Algorithm 11 and Algorithm 13, respectively. SM and MS are the hybrid algorithms in Section 6.4.6. Superpreference and M-relation are both used but applied in different order in SM and MS.

For the performance measure, we focus on the number of candidate k-subsets (or k-multisubsets) generated, as it is the dominant factor in each algorithm. There are usually three major types of operations in each algorithm:

- *Preprocessing*: the superpreference filtering and/or the M-relation generation;
- *Generation*: candidate k-subset (or k-multisubset) generation;
- *Winnow*.

In *Preprocessing*, the superpreference filtering is implemented by computing $cover$ for each tuple, which takes quadratic time. The M-relation generation involves a `GROUP BY` clause, and is therefore linear if implemented by hashing or $O(n \log n)$ if implemented by sorting. As a result, *Preprocessing* and *Winnow* together have worst-case quadratic complexity. On the other hand, *Generation* takes $\binom{n}{k}$ time. When $k << n$, this is $O(n^k)$. When $k = \lfloor n/2 \rfloor$, $\binom{n}{k}$ is approximately $\frac{2^{n+1}}{\sqrt{2\pi n}}$. Consequently, the complexity of *Generation* dominates the complexity of all algorithms when $k > 2$.

In our experiments, denote by g the number of sets generated in *Generation*. Recall the definitions of feature $\mathcal{F}_5$ and $\mathcal{F}_6$ in Example 19.

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

$\mathcal{F}_6 \equiv$ `SELECT sum(rating) FROM $S`

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 we tested are listed in Table 6.1. They are all conjunctions since disjunctions easily lead to cyclic preferences which are not strict partial orders. The corresponding superpreferences and M-relation generation SQL queries are listed in Table 6.2

and Table 6.3, respectively. Notice that the only difference between SP1 and SP2 is that we apply the `WHERE` condition to the attribute `price` in $\mathcal{F}_5$ of SP1 while the same `WHERE` condition is applied to the attribute `rating` in $\mathcal{F}_9$ of SP2. We intend to see the influence of selectivity of attributes. In SP1, since `price` has a `WHERE` condition while `rating` does not, the selectivity of `rating` is dominating. In contrast, in SP2, for a similar reason, the selectivity of `price` is dominating.

Set Pref. Name	Profile 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 6.1: Set Preferences

Set Pref. Name	Superpreference
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'*$)$
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'*$)$
SP3	$t_1.genre =$ *'sci-fi'* $\wedge\, t_1.price < 20.00$ $\wedge(t_2.genre \neq$ *'sci-fi'* $\vee\, t_2.price \geqslant 20.00) \wedge t_1.rating \geqslant 4.0$ $\wedge(t_2.rating \leqslant 4.0 \vee t_1.rating > t_2.rating)$

Table 6.2: Superpreferences

Notice that SP1 is used extensively in our running examples. Its superpreference and M-relation generating SQL already appears in Example 19 and Example 22 before. We list them here for easy reference.

Summary of experiments

We drew the following conclusions from the forthcoming experimental results:

- 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`

Set Pref. Name	M-relation Generating SQL

SP1	`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	`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	`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 6.3: M-relation Generation SQLs

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.

6.5.1 Performance

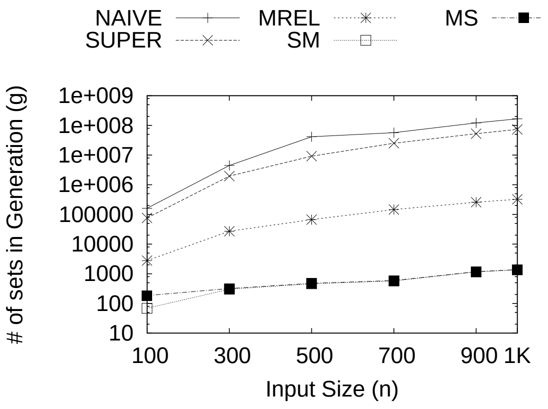
In our first experiment, we wanted to study how much computation we could save by applying superpreference and/or M-relation optimization. We fixed k to be 3. We tested the dataset up to 1000 tuples as some algorithms do not scale up well with n.

Figures 6-1(a), 6-1(c) and 6-1(e) illustrate the increase of the number g of sets *Generation* with the increase of input size n for SP1, SP2 and SP3, respectively. For the six datasets of different sizes used, Figures 6-1(b), 6-1(d) and 6-1(f) show the average ratio of the number of sets generated in each algorithm to that in NAIVE. Notice that the y-scale is logarithmic in Figure 6-1.

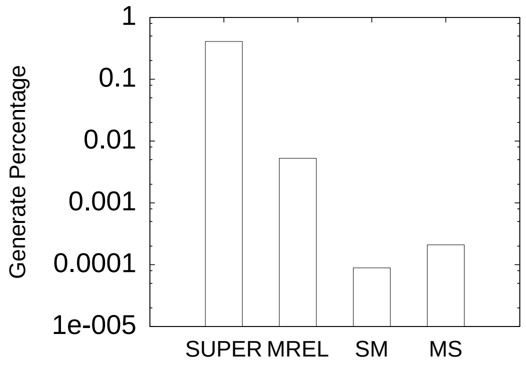
As we can see from Figure 6-1, for standalone optimization techniques, i.e., SUPER and MREL, the relative efficiency depends on the set preference. MREL is more efficient in SP1 and SP3 (Figure 6-1(b), 6-1(f)), while SUPER is more efficient in SP2 (Figure 6-1(d)). If we compare SP1 and SP2 in juxtaposition, it is clear the different reaction to optimizations is caused by the different selectivity of the attribute `rating` and `price`: in our largest dataset, there are 9 distinct `rating` values and 406 distinct `price` values. For SUPER, the difference between the superpreferences of SP1 and SP2 are underlined.

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

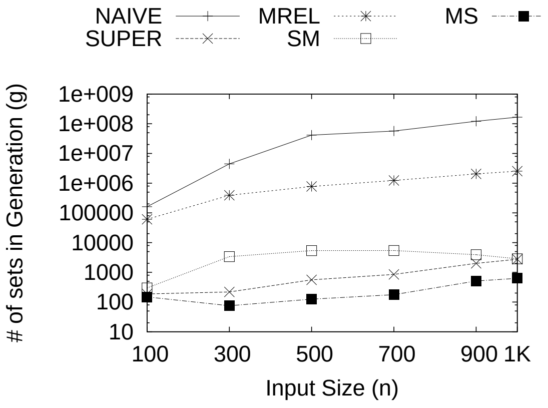
$$\succ^{+} \text{ in SP2} \;\equiv\; \begin{array}{l}(t_1.price < t_2.price \wedge t_1.genre = \textit{'sci-fi'} \wedge t_1.rating > t_2.rating)\\ \vee(\underline{t_1.price < t_2.price} \wedge t_1.genre = \textit{'sci-fi'} \wedge t_2.genre \neq \textit{'sci-fi'})\end{array}$$



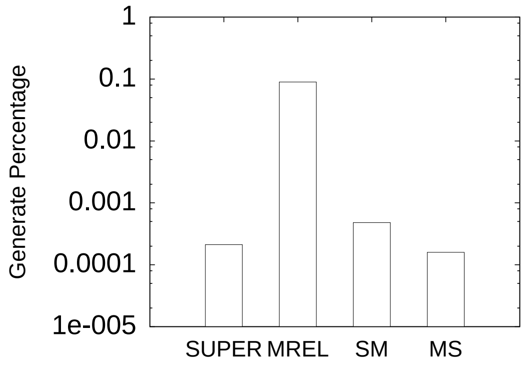
(a) # of sets v.s. n (SP1)



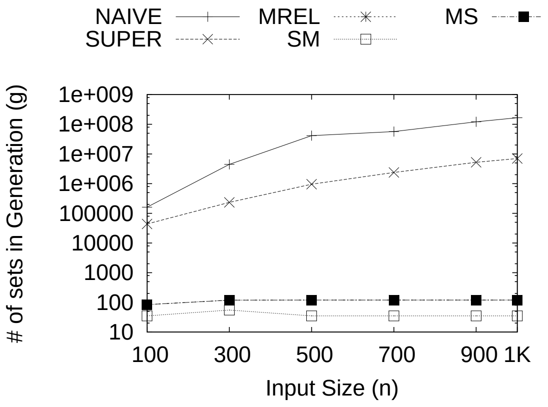
(b) Avg Generate Percentage (SP1)



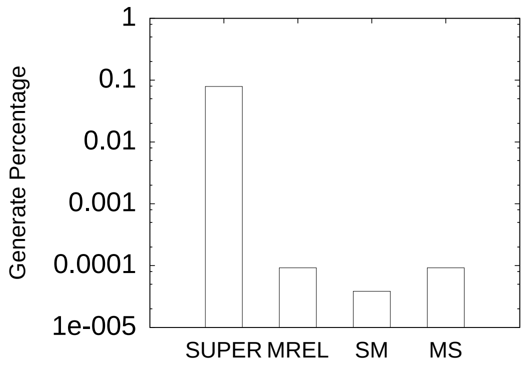
(c) # of sets v.s. n (SP2)



(d) Avg Generate Percentage (SP2)

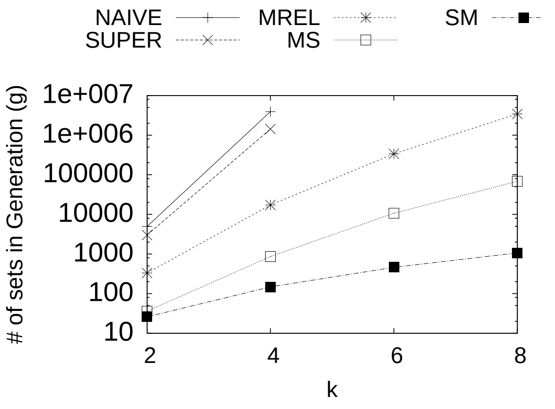


(e) # of sets v.s. n (SP3)

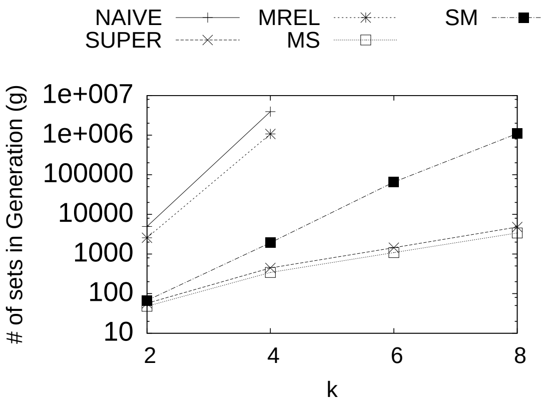


(f) Avg Generate Percentage (SP3)

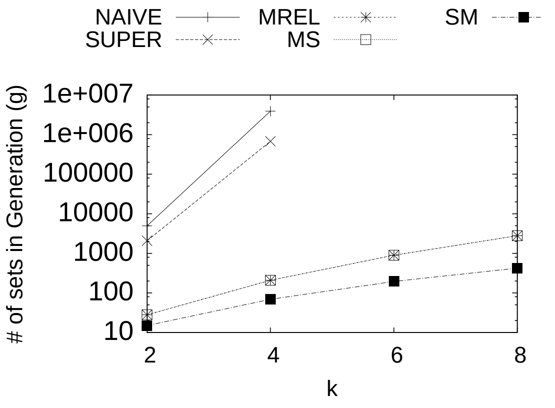
Figure 6-1: Performance of Different Algorithms under Varying Input Size n



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



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



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

Figure 6-2: Performance of Different Algorithms under Varying k

SUPER is more effective in SP2 as 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`. Therefore, in SP1, we are able to consolidate more tuples into one M-tuple on average. Therefore, MREL is more effective in SP1.

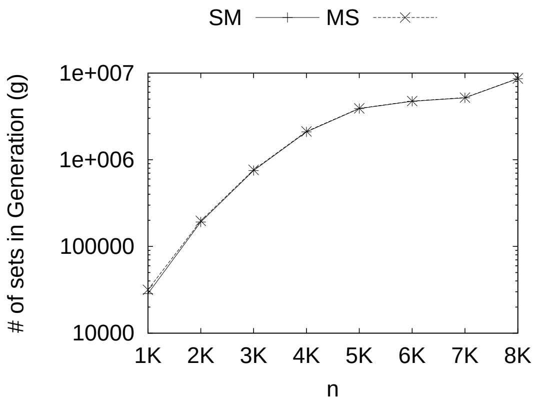
For SP3, since the `WHERE` conditions in both features of SP3 are more restricted than that in SP1, e.g., the condition in $\mathcal{F}_{11}$ requires `price<20.00` in addition to the condition `genre='sci-fi'` used in $\mathcal{F}_5$, MREL is able to benefit more from the low selectivity and therefore achieves higher efficiency.

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 (cf., Figure 6-1(b), 6-1(f)), and MS outperforms SM when SUPER outperforms MREL (cf., Figure 6-1(d)). 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.

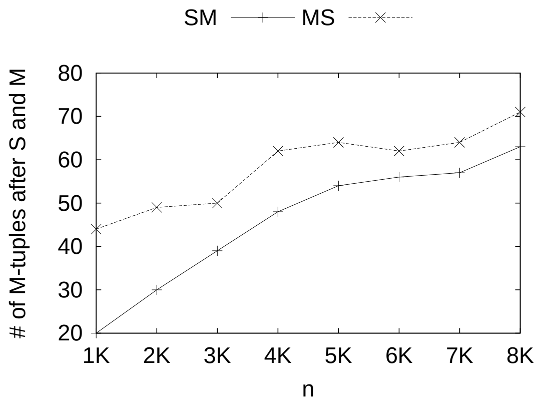
Not only do the optimizations help reducing the number of sets generated in *Generation*, they also improve the scalability with the input size n. For example, for SP1 (Figure 6-1(a)), NAIVE displays a cubic trend with the increase of n as predicted by the theoretic analysis, i.e. $\binom{n}{k} \approx O(n^k)$. SUPER displays a similar trend, while MREL grows slower than that of SUPER, and MS and SM grow slower than that of MREL. In all three cases, i.e., SP1, SP2 and SP3, (Figure 6-1(a), Figure 6-1(c) and 6-1(e)), the best algorithm displays a much slower trend compared to that of NAIVE.

Figure 6-2 illustrates the increase of the number g of sets in *Generation* 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 for each set preference scales somewhere in between linear and quadratic with the increase of k. Compared to the increase in n (Figure 6-1), the increase in k has a bigger influence on the performance for each algorithm (Figure 6-2).

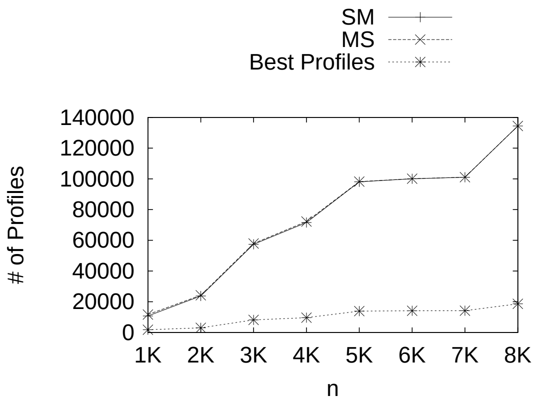
It is worth noticing that regardless of which set preference is used, the performance of different algorithms is stable with the increase of n or k, as the curves of each algorithms do not cross each other in Figure 6-1 or Figure 6-2. We have also seen in Figure 6-1 and Figure 6-2 that it is crucial to choose the right algorithm given a set preference. A practical strategy would be to take a small sample of the data, and try out the set preference with a small k in order to pick the best algorithm. 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.



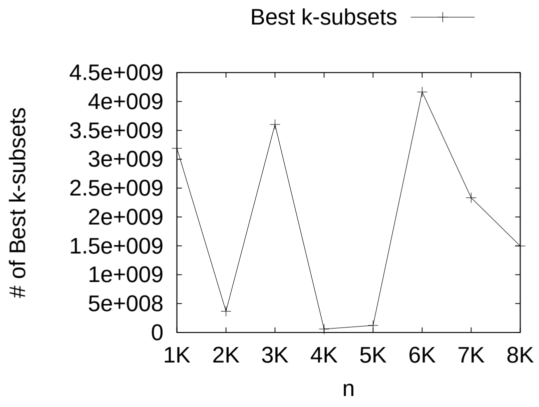
(a) Generate Size (g)



(b) # of M-tuples after S and M



(c) # of Profiles



(d) # of Best k-subsets

Figure 6-3: SM v.s. MS in SP1

6.5.2 SM v.s. MS

In the second set of experiments, we took a close look at the two best algorithms: SM and MS. In Figure 6-1, 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 Figure 6-3, we compare the analytics of the hybrid algorithms SM and MS when $k = 5$ and n varies from 1000 to 8000 in SP1. Figure 6-3(a) illustrates the number of sets generated in *Generation* (g) for different parameter settings. Figure 6-3(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. Their number has a positive correlation with the number of k-multisubsets generated (Figure 6-3(a)). In Figure 6-3(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 Figure 6-3(a), SM generates g sets while MS generates slightly more than g sets. The difference is relatively small compared to g. As a result, the two lines in Figure 6-3(a) almost overlap with each other. Figure 6-3(c) displays the number of profiles computed in SM and MS. The two lines also overlap with each other, since SM generates slightly fewer profiles than MS does. As a reference, we also illustrate the number of best profiles in the profile relation in Figure 6-3(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 number of best k-subsets in each parameter setting in Figure 6-3(d), i.e., the result size. Both SM and MS correctly identify all the best k-subsets. There is no obvious relationship between the input size n and the number of best k-subsets. On the other hand, the number of best k-subsets is huge in general, i.e., in the order of $10^8 \sim 10^9$. In fact, the result size in Figure 6-3(d) is $2 \sim 5$ orders of magnitude larger than the number of candidate sets generated (Figure 6-3(a)), which illustrates the compactness of the M-relation representation.

6.6 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 [29, 21, 6].

[6] 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 [9] or a scoring function. [6] gives heuristic search algorithms for finding an optimal subset. [6] considers subsets of any cardinality. For fixed-cardinality subsets, the language in [6] can easily be expressed in our framework with a simple extension of Definition 6.2.1 to boolean features. Each Class 1 *set property* in [6] 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 Definition 6.2.1. In fact, the extension to boolean features can easily be accomplished by introducing a relational operator in the SQL definition:

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

where $\theta \in \{=, \neq, <, >, \leqslant, \geqslant\}$. Each Class 2 *set property* in [6] can be translated to such a boolean feature. Such boolean features are additive when $\theta \in \{<, >, \leqslant, \geqslant\}$. Otherwise, they are non-additive.

The preference model in [6], i.e., either a scoring function or a TCP-net set preference, can be captures by our *set preference relation* as well. General aggregate features are not supported in [6].

[21] 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[2] 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. In short, in order to express the diversity of an attribute, we can define one feature for the attribute's *mean*, one feature for its *mode*, and one feature for its *second moment*. Features derived from *depth*, 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 [21] 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 showed that we can express the set preferences in both [6] and [21] under our current framework with the aforementioned techniques. In order to efficiently evaluate the set preferences derived from [6, 21], the M-relation optimization is always applicable. For the superpreference optimization, if the set preference formula describing the resulting set preference relation can be written a constant-free DNF, then we can apply the superpreference optimization in a systematic manner based to Theorem 6.4.1. Moreover, the features and the preference model in [21] are of special forms that a specialized optimization schema can be designed to achieve better optimization results.

[29] 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. [29] considers subsets of any cardinality. The atomic parametric aggregation constraint is of the form $aggr(A) < parameter$ and the

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

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.

Appendices

Appendix A

Semantic Postulates

Semantics	Exact k	Faithfulness	Stability
[†] Global-Top$k$	✓(1)	✓/× (5)	✓(9)
PT-k	× (2)	✓/× (6)	✓(10)
U-Top$k$	× (3)	✓/× (7)	✓(11)
U-kRanks	× (4)	× (8)	× (12)

[†] Postulates of Global-Top$k$ semantics are proved under general scoring functions with *Equal* allocation policy.

Table A.1: Postulate Satisfaction for Different Semantics in Table 3.1

The following proofs correspond to the numbers next to each entry in the above table. Assume that we are given a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, a non-negative integer k and an injective scoring function s.

Exact k

(1) Global-Top$k$ satisfies *Exact* k.

We compute the Global-Top$k$ probability for each tuple in R. If there are at least k tuples in R, we are always able to pick the k tuples with the highest Global-Top$k$ probability. In case when there are more than $k - r + 1$ tuple(s) with the rth highest

Global-Top$k$ probability, where $r = 1, 2 \ldots, k$, only $k - r + 1$ of them will be picked nondeterministically.

(2) PT-k violates *Exact* k.

Example 6 illustrates a counterexample in a simple probabilistic relation.

(3) U-Top$k$ violates *Exact* k.

Example 6 illustrates a counterexample in a simple probabilistic relation.

(4) U-kRanks violates *Exact* k.

Example 6 illustrates a counterexample in a simple probabilistic relation.

Faithfulness

(5) Global-Top$k$ satisfies *Faithfulness* in simple probabilistic relations while it violates *Faithfulness* in general probabilistic relations.

(5a) Simple Probabilistic Relations

By the assumption, $t_1 \succ_s t_2$ and $p(t_1) > p(t_2)$, so we need to show that $P_{k,s}(t_1) > P_{k,s}(t_2)$.

For every $W \in pwd(R^p)$ such that $t_2 \in all_{k,s}(W)$ and $t_1 \notin all_{k,s}(W)$, obviously $t_1 \notin W$. Otherwise, since $t_1 \succ_s t_2$, t_1 would be in $all_{k,s}(W)$. Since all tuples are independent, there is always a world $W' \in pwd(R^p)$, $W' = (W \backslash \{t_2\}) \cup \{t_1\}$ and $Pr(W') = Pr(W)\frac{p(t_1)\bar{p}(t_2)}{\bar{p}(t_1)p(t_2)}$. Since $p(t_1) > p(t_2)$, $Pr(W') > Pr(W)$. Moreover, t_1 will substitute for t_2 in the top-k answer set to W'. It is easy to see that $\alpha(t_1, W') = 1$ in W' and also in any world W such that both t_1 and t_2 are in $all_{k,s}(W)$, $\alpha(t_1, W) = 1$.

Therefore, for the Global-Top$k$ probability of t_1 and t_2, we have

$$\begin{aligned}
P_{k,s}(t_2) &= \sum_{\substack{W \in pwd(R^p)\\ t_1 \in all_{k,s}(W)\\ t_2 \in all_{k,s}(W)}} \alpha(t_2, W) Pr(W) + \sum_{\substack{W \in pwd(R^p)\\ t_1 \notin all_{k,s}(W)\\ t_2 \in all_{k,s}(W)}} \alpha(t_2, W) Pr(W)\\
&< \sum_{\substack{W \in pwd(R^p)\\ t_1 \in all_{k,s}(W)\\ t_2 \in all_{k,s}(W)}} Pr(W) + \sum_{\substack{W' \in pwd(R^p)\\ t_1 \in all_{k,s}(W')\\ t_2 \notin W'}} Pr(W')\\
&= \sum_{\substack{W \in pwd(R^p)\\ t_1 \in all_{k,s}(W)\\ t_2 \in all_{k,s}(W)}} \alpha(t_1, W) Pr(W) + \sum_{\substack{W' \in pwd(R^p)\\ t_1 \in all_{k,s}(W')\\ t_2 \notin W'}} \alpha(t_1, W') Pr(W')\\
&\leqslant \sum_{\substack{W \in pwd(R^p)\\ t_1 \in all_{k,s}(W)\\ t_2 \in all_{k,s}(W)}} \alpha(t_1, W) Pr(W) + \sum_{\substack{W' \in pwd(R^p)\\ t_1 \in all_{k,s}(W')\\ t_2 \notin W'}} \alpha(t_1, W') Pr(W')\\
&\quad + \sum_{\substack{W'' \in pwd(R^p)\\ t_1 \in all_{k,s}(W'')\\ t_2 \in W''\\ t_2 \notin all_{k,s}(W'')}} \alpha(t_1, W'') Pr(W'')\\
&= P_{k,s}(t_1).
\end{aligned}$$

The equality in $\leqslant$ holds when $s(t_2)$ is among the k highest scores and there are at most k tuples (including t_2) with higher or equal scores. Since there is at least one inequality in the above equation, we have

$$P_{k,s}(t_1) > P_{k,s}(t_2).$$

(5b) General Probabilistic Relations

The following is a counterexample.

Say $k = 1$, $R = \{t_1, \ldots, t_9\}$, $t_1 \succ_s \ldots \succ_s t_9$, $\{t_1, \ldots, t_7, t_9\}$ are exclusive. $p(t_i) = 0.1, i = 1 \ldots 7$, $p(t_8) = 0.4$, $p(t_9) = 0.3$.

By Global-Top$k$, the top-1 answer is $\{t_9\}$, while $t_8 \succ_s t_9$ and $p(t_8) > p(t_9)$, which violates *Faithfulness*.

(6) PT-k satisfies *Faithfulness* in simple probabilistic relations while it violates *Faithful-*

ness in general probabilistic relations.

For simple probabilistic relations, we can use the same proof in (5) to show that PT-k satisfies *Faithfulness*. The only change would be that we need to show $P_{k,s}(t_1) > p_\tau$ as well. Since $P_{k,s}(t_2) > p_\tau$ and $P_{k,s}(t_1) > P_{k,s}(t_2)$, this is obviously true. For general probabilistic relations, we can use the same counterexample in (5) and set threshold $p_\tau = 0.15$.

(7) U-Top$k$ satisfies *Faithfulness* in simple probabilistic relations while it violates *Faithfulness* in general probabilistic relations.

(7a) Simple Probabilistic Relations

By contradiction. If U-Top$k$ violates *Faithfulness* in a simple probabilistic relation, there exists $R^p = \langle R, p, \mathcal{C} \rangle$ and exists $t_i, t_j \in R, t_i \succ_s t_j, p(t_i) > p(t_j)$, and by U-Top$k$, t_j is in the top-k answer set to R^p under the scoring function s while t_i is not.

S is a top-k answer set to R^p under the function s by the U-Top$k$ semantics, $t_j \in S$ and $t_i \notin S$. Denote by $Q_{k,s}(S)$ the probability of S under the U-Top$k$ semantics. That is,

$$Q_{k,s}(S) = \sum_{\substack{W \in pwd(R^p) \\ S = top_{k,s}(W)}} Pr(W).$$

For any world W contributing to $Q_{k,s}(S)$, $t_i \notin W$. Otherwise, since $t_i \succ_s t_j$, t_i would be in $top_{k,s}(W)$, which is S. Define a world $W' = (W \backslash \{t_j\}) \cup \{t_i\}$. Since t_i is independent of any other tuple in R, $W' \in pwd(R^p)$ and $Pr(W') = Pr(W)\frac{p(t_i)\bar{p}(t_j)}{\bar{p}(t_i)p(t_j)}$. Moreover, $top_{k,s}(W') = (S \backslash \{t_j\}) \cup \{t_i\}$. Let $S' = (S \backslash \{t_j\}) \cup \{t_i\}$, then W' contributes to $Q_{k,s}(S')$.

$$\begin{aligned}
Q_{k,s}(S') &= \sum_{\substack{W\in pwd(R^p)\\ S'=top_{k,s}(W)}} Pr(W)\\
&\geqslant \sum_{\substack{W\in pwd(R^p)\\ S=top_{k,s}(W)}} Pr((W\backslash\{t_j\})\cup\{t_i\})\\
&= \sum_{\substack{W\in pwd(R^p)\\ S=top_{k,s}(W)}} Pr(W)\frac{p(t_i)\bar{p}(t_j)}{\bar{p}(t_i)p(t_j)}\\
&= \frac{p(t_i)\bar{p}(t_j)}{\bar{p}(t_i)p(t_j)} \sum_{\substack{W\in pwd(R^p)\\ S=top_{k,s}(W)}} Pr(W)\\
&= \frac{p(t_i)\bar{p}(t_j)}{\bar{p}(t_i)p(t_j)} Q_{k,s}(S)\\
&> Q_{k,s}(S),
\end{aligned}$$

which is a contradiction. The last inequality holds because $p(t_i) > p(t_j)$.

(7b) General Probabilistic Relations

The following is a counterexample.

Say $k = 2$, $R = \{t_1, t_2, t_3, t_4\}$, $t_1 >_s t_2 >_s t_3 >_s t_4$, t_1 and t_2 are exclusive, t_3 and t_4 are exclusive. $p(t_1) = 0.5, p(t_2) = 0.45, p(t_3) = 0.4, p(t_4) = 0.3$.

By U-Top$k$, the top-2 answer is $\{t_1, t_3\}$, while $t_2 >_s t_3$ and $p(t_2) > p(t_3)$, which violates *Faithfulness*.

(8) U-kRanks violates *Faithfulness*.

The following is a counterexample.

Say $k = 2$, R^p is simple. $R = \{t_1, t_2, t_3\}$, $t_1 >_s t_2 >_s t_3$, $p(t_1) = 0.48, p(t_2) = 0.8, p(t_3) = 0.78$.

The probabilities of each tuple at each rank are as follows:

	t_1	t_2	t_3
rank 1	0.48	0.416	0.08112
rank 2	0	0.384	0.39936
rank 3	0	0	0.29952

By U-kRanks, the top-2 answer set is $\{t_1, t_3\}$ while $t_2 \succ t_3$ and $p(t_2) > p(t_3)$, which contradicts *Faithfulness*.

Stability

(9) Global-Top$k$ satisfies *Stability*.

In the rest of this proof, let A be the set of all winners under the Global-Top$k$ semantics.

Part I: Probability.

Case 1: Winners.

For any winner $t \in A$, if we only raise the probability of t, we have a new probabilistic relation $(R^p)' = \langle R, p', \mathcal{C} \rangle$, where the new probability function p' is such that $p'(t) > p(t)$ and for any $t' \in R, t' \neq t, p'(t') = p(t')$. Note that $pwd(R^p) = pwd((R^p)')$. In addition, assume $t \in C_t$, where $C_t \in \mathcal{C}$. By Global-Top$k$,

$$P_{k,s}^{R^p}(t) = \sum_{\substack{W \in pwd(R^p) \\ t \in all_{k,s}(W)}} \alpha(t, W) Pr(W)$$

and

$$\begin{aligned} P_{k,s}^{(R^p)'}(t) &= \sum_{\substack{W \in pwd(R^p) \\ t \in all_{k,s}(W)}} \alpha(t, W) Pr(W) \frac{p'(t)}{p(t)} \\ &= \frac{p'(t)}{p(t)} P_{k,s}^{R^p}(t). \end{aligned}$$

For any other tuple $t' \in R, t' \neq t$, we have the following equation:

$$\begin{aligned}
P_{k,s}^{(R^p)'}(t') &= \sum_{\substack{W \in pwd(R^p) \\ t' \in all_{k,s}(W),\ t \in W}} \alpha(t', W) Pr(W) \frac{p'(t)}{p(t)} \\
&\quad + \sum_{\substack{W \in pwd(R^p) \\ t' \in all_{k,s}(W),\ t \notin W \\ (C_t \backslash \{t\}) \cap W = \varnothing}} \alpha(t', W) Pr(W) \frac{c - p'(t)}{c - p(t)} \\
&\quad + \sum_{\substack{W \in pwd(R^p) \\ t' \in all_{k,s}(W),\ t \notin W \\ (C_t \backslash \{t\}) \cap W \neq \varnothing}} \alpha(t', W) Pr(W) \\
&\leqslant \frac{p'(t)}{p(t)} \Big(\sum_{\substack{W \in pwd(R^p) \\ t' \in all_{k,s}(W),\ t \in W}} \alpha(t', W) Pr(W) \\
&\quad + \sum_{\substack{W \in pwd(R^p) \\ t' \in all_{k,s}(W),\ t \notin W \\ (C_t \backslash \{t\}) \cap W = \varnothing}} \alpha(t', W) Pr(W) \\
&\quad + \sum_{\substack{W \in pwd(R^p) \\ t' \in all_{k,s}(W),\ t \notin W \\ (C_t \backslash \{t\}) \cap W \neq \varnothing}} \alpha(t', W) Pr(W) \Big) \\
&= \frac{p'(t)}{p(t)} P_{k,s}^{R^p}(t'),
\end{aligned}$$

where $c = 1 - \sum_{t'' \in C_t \backslash \{t\}} p(t'')$.

Now we can see that, t's Global-Top$k$ probability in $(R^p)'$ will be raised to *exactly* $\frac{p'(t)}{p(t)}$ times of that in R^p under the same weak order scoring function s, and for any tuple other than t, its Global-Top$k$ probability in $(R^p)'$ can be raised to *as much as* $\frac{p'(t)}{p(t)}$ times of that in R^p under the same scoring function s. As a result, $P_{k,s}^{(R^p)'}(t)$ is still among the highest k Global-Top$k$ probabilities in $(R^p)'$ under the function s, and therefore still a winner.

Case 2: Losers.

This case is similar to *Case 1*.

Part II: Score.

Case 1: Winners.

For any winner $t \in A$, we evaluate R^p under a new general scoring function s'. Comparing to s, s' only raises the score of t. That is, $s'(t) > s(t)$ and for any $t' \in R, t' \neq t, s'(t') = s(t')$. Then, in addition to all the worlds already *totally* (i.e., $\alpha(t, W) = 1$) or *partially* (i.e., $\alpha(t, W) < 1$) contributing to t's Global-Top$k$ probability when evaluating R^p under the function s, some other worlds may now totally or partially contribute to t's Global-Top$k$ probability. Because, under the function s', t might climb high enough to be in the top-k answer set of those worlds. Moreover, if a possible world W contributes partially under scoring function s, it is easy to see that it contributes totally under scoring function s'.

For any tuple t'' other than t in R,

(i) If $s(t'') \neq s(t)$, then its Global-Top$k$ probability under the function s' either stays the same (if the "climbing" of t does not knock that tuple out of the top-k answer set in some possible world) or decreases (otherwise);

(ii) If $s(t'') = s(t)$, then for any possible world W contributing to t'''s Global-Top$k$ under scoring function s, $\alpha(t'', W) = \frac{k-a}{b}$, and now under scoring function s', $\alpha'(t'', W) = \frac{k-a-1}{b-1} < \frac{k-a}{b} = \alpha(t'', W)$. Therefore the Global-Top$k$ of t'' under scoring function s' is less than that under scoring function s.

Consequently, t is still a winner when evaluating R^p under the function s'.

Case 2: Losers.

This case is similar to *Case 1.*

(10) PT-k satisfies *Stability*.

In the rest of this proof, let A be the set of all winners under the PT-k semantics.

Part I: Probability.

Case 1: Winners.

For any winner $t \in A$, if we only raise the probability of t, we have a new probabilistic relation $(R^p)' = \langle R, p', \mathcal{C}\rangle$, where the new probability function p' is such that $p'(t) > p(t)$ and for any $t' \in R, t' \neq t, p'(t') = p(t')$. Note that $pwd(R^p) = pwd((R^p)')$. In addition, assume $t \in C_t$, where $C_t \in \mathcal{C}$. The Global-Top$k$ probability of t is such that

$$P_{k,s}^{R^p}(t) = \sum_{\substack{W \in pwd(R^p) \\ t \in top_{k,s}(W)}} Pr(W) \geqslant p_\tau$$

and

$$\begin{aligned} P_{k,s}^{(R^p)'}(t) &= \sum_{\substack{W \in pwd(R^p) \\ t \in top_{k,s}(W)}} Pr(W)\frac{p'(t)}{p(t)} \\ &= \frac{p'(t)}{p(t)} P_{k,s}^{R^p}(t) > P_{k,s}^{R^p}(t) \geqslant p_\tau. \end{aligned}$$

Therefore, $P_{k,s}^{(R^p)'}(t)$ is still above the threshold p_τ, and t still belongs to the top-k answer set of $(R^p)'$ under the function s.

Case 2: Losers.

This case is similar to *Case 1.*

Part II: Score.

Case 1: Winners.

For any winner $t \in A$, we evaluate R^p under a new scoring function s'. Comparing to s, s' only raises the score of t. Use a similar argument as that in (9) Part II Case 1 but under injective scoring functions, we can show that the Global-Top$k$ probability of t is non-decreasing and is still above the threshold p_τ. Therefore, tuple t still belongs to the top-k answer set under the function s'.

Case 2: Losers.

This case is similar to *Case 1.*

(11) U-Top$k$ satisfies *Stability*.

In the rest of this proof, let A be the set of all winners under U-Top$k$ semantics.

Part I: Probability.

Case 1: Winners.

For any winner $t \in A$, if we only raise the probability of t, we have a new probabilistic relation $(R^p)' = \langle R, p', \mathcal{C} \rangle$, where the new probabilistic function p' is such that $p'(t) > p(t)$ and for any $t' \in R, t' \neq t, p'(t') = p(t')$. In the following discussion, we use superscript to indicate the probability in the context of $(R^p)'$. Note that $pwd(R^p) = pwd((R^p)')$.

Recall that $Q_{k,s}(A_t)$ is the probability of a top-k answer set $A_t \subseteq A$ under U-Top$k$ semantics, where $t \in A_t$. Since $t \in A_t$, $Q'_{k,s}(A_t) = Q_{k,s}(A_t)\frac{p'(t)}{p(t)}$.

For any candidate top-k answer set B other than A_t, i.e., $\exists W \in pwd(R^p), top_{k,s}(W) = B$ and $B \neq A_t$. By definition,

$$Q_{k,s}(B) \leqslant Q_{k,s}(A_t).$$

For any world W contributing to $Q_{k,s}(B)$, its probability either increase $\frac{p'(t)}{p(t)}$ times (if $t \in W$), or stays the same (if $t \notin W$ and $\exists t' \in W, t'$ and t are exclusive), or decreases (otherwise). Therefore,

$$Q'_{k,s}(B) \leqslant Q_{k,s}(B)\frac{p'(t)}{p(t)}.$$

Altogether,

$$Q'_{k,s}(B) \leqslant Q_{k,s}(B)\frac{p'(t)}{p(t)} \leqslant Q_{k,s}(A_t)\frac{p'(t)}{p(t)} = Q'_{k,s}(A_t).$$

Therefore, A_t is still a top-k answer set to $(R^p)'$ under the function s and $t \in A_t$ is still a winner.

Case 2: Losers.

It is more complicated in the case of losers. We need to show that for any loser t, if we decrease its probability, no top-k candidate answer set B_t containing t will be a new top-k answer set under the U-Top$k$ semantics. The procedure is similar to that in *Case 1*, except that when we analyze the new probability of any original top-k answer set A_i, we need to differentiate between two cases:

(a) t is exclusive with some tuple in A_i, or t has a score higher than that of some tuple in A_i;

(b) t is independent of all the tuples in A_i and t has a score lower than that of every tuple in A_i.

It is easier with (a), where all the worlds contributing to the probability of A_i do not contain t. In (b), some worlds contributing to the probability of A_i contain t, while others do not. And we calculate the new probability for those two kinds of worlds differently. As we will see shortly, the probability of A_i is non-decreasing in either (a) or (b).

For any loser $t \in R, t \notin A$, by applying the technique used in *Case 1*, we have a new probabilistic relation $(R^p)' = \langle R, p', \mathcal{C} \rangle$, where the new probabilistic function p' is such that $p'(t) < p(t)$ and for any $t' \in R, t' \neq t, p'(t') = p(t')$. Again, $pwd(R^p) = pwd((R^p)')$.

For any top-k answer set A_i to R^p under the function s, $A_i \subseteq A$, and therefore the loser $t \notin A_i$ by definition. Denote by S_{A_i} all the possible worlds contributing to $Q_{k,s}(A_i)$. Based on the membership of t, S_{A_i} can be partitioned into two subsets $S^t_{A_i}$ and $S^{\bar{t}}_{A_i}$.

$$\begin{aligned}
&S_{A_i} = \{W | W \in pwd(R^p), top_{k,s}(W) = A_i\};\\
&S_{A_i} = S^t_{A_i} \cup S^{\bar{t}}_{A_i}, S^t_{A_i} \cap S^{\bar{t}}_{A_i} = \emptyset,\\
&\forall W \in S^t_{A_i}, t \in W \text{ and } \forall W \in S^{\bar{t}}_{A_i}, t \notin W.
\end{aligned}$$

$$Q_{k,s}(A_i) = \sum_{\substack{W \in pwd(R^p)\\ W \in S^t_{A_i}}} Pr(W) + \sum_{\substack{W \in pwd(R^p)\\ W \in S^{\bar{t}}_{A_i}}} Pr(W)$$

In case (a), $S^t_{A_i} = \varnothing$. For any world $W \in S^{\bar{t}}_{A_i}$, its probability is unchanged if it contains one of t's exclusive tuples, and is increasing if it does not contain any tuple from the part C_t where t belongs.

$$\begin{aligned}
Q'_{k,s}(A_i) &= \sum_{\substack{W \in pwd(R^p) \\ W \in S^{\bar{t}}_{A_i}, C_t \cap W \neq \varnothing}} Pr(W) + \sum_{\substack{W \in pwd(R^p) \\ W \in S^{\bar{t}}_{A_i}, C_t \cap W = \varnothing}} Pr(W) \frac{c - p'(t)}{c - p(t)} \\
&\geqslant \sum_{\substack{W \in pwd(R^p) \\ W \in S^{\bar{t}}_{A_i}, C_t \cap W \neq \varnothing}} Pr(W) + \sum_{\substack{W \in pwd(R^p) \\ W \in S^{\bar{t}}_{A_i}, C_t \cap W = \varnothing}} Pr(W) \\
&= Q_{k,s}(A_i).
\end{aligned}$$

where $c = 1 - \sum_{t' \in C_t \setminus \{t\}} p(t')$.

In case (b), t is independent of all the tuples in A_i and has a score lower than that of every tuple in A_i. In this case,

$$\frac{\sum_{\substack{W \in pwd(R^p) \\ W \in S^t_{A_i}}} Pr(W)}{\sum_{\substack{W \in pwd(R^p) \\ W \in S^{\bar{t}}_{A_i}}} Pr(W)} = \frac{p(t)}{1 - p(t)}$$

and

$$\begin{aligned}
Q'_{k,s}(A_i) &= \sum_{\substack{W \in pwd(R^p) \\ W \in S^t_{A_i}}} Pr(W) \frac{p'(t)}{p(t)} \\
&\quad + \sum_{\substack{W \in pwd(R^p) \\ W \in S^{\bar{t}}_{A_i}}} Pr(W) \frac{1 - p'(t)}{1 - p(t)} \\
&= \sum_{\substack{W \in pwd(R^p) \\ W \in S_{A_i}}} Pr(W) \\
&= Q_{k,s}(A_i).
\end{aligned}$$

We can see that in both cases, $Q'_{k,s}(A_i) \geqslant Q_{k,s}(A_i)$.

Now for any top-k candidate answer set containing t, say B_t such that $B_t \nsubseteq A$, by definition, $Q_{k,s}(B_t) < Q_{k,s}(A_i)$. Moreover,

$$Q'_{k,s}(B_t) = Q_{k,s}(B_t)\frac{p'(t)}{p(t)} < Q_{k,s}(B_t).$$

Therefore,

$$Q'_{k,s}(B_t) < Q_{k,s}(B_t) < Q_{k,s}(A_i) \leqslant Q'_{k,s}(A_i).$$

Consequently, B_t is still not a top-k answer set to $(R^p)'$ under the function s. Since no top-k candidate answer set containing t can be a top-k answer set to $(R^p)'$ under the function s, t is still a loser.

Part II: Score.

Again, $A_i \subseteq A$ is a top-k answer set to R^p under the function s by U-Top$k$ semantics.

Case 1: Winners.

For any winner $t \in A_i$, we evaluate R^p under a new scoring function s'. Comparing to s, s' only raises the score of t. That is, $s'(t) > s(t)$ and for any $t' \in R, t' \neq t, s'(t') = s(t')$. In some possible world such that $W \in pwd(R^p)$ and $top_{k,s}(W) \neq A_i$, t might climb high enough to be in $top_{k,s'}(W)$. Define T to the set of such top-k candidate answer sets.

$$T = \{top_{k,s'}(W) | W \in pwd(R^p), t \notin top_{k,s}(W) \wedge t \in top_{k,s'}(W)\}.$$

Only a top-k candidate set $B_j \in T$ can possibly end up with a probability higher than that of A_i across all possible worlds, and thus substitute for A_i as a new top-k answer set to R^p under the function s'. In that case, $t \in B_j$, so t is still a winner.

Case 2: Losers.

For any loser $t \in R, t \notin A$. Using a similar technique to *Case 1*, the new scoring function s' is such that $s'(t) < s(t)$ and for any $t' \in R, t' \neq t, s'(t') = s(t')$. When evaluating R^p under the function s', for any world $W \in pwd(R^p)$ such that $t \notin top_{k,s}(W)$, the score decrease of t will not effect its top-k answer set, i.e., $top_{k,s'}(W) = top_{k,s}(W)$. For any world $W \in pwd(R^p)$ such that $t \in top_{k,s}(W)$, t might go down enough to drop out of $top_{k,s'}(W)$. In this case, W will contribute its probability to a top-k candidate answer set without t, instead of the original one with t. In other words, under the function s', comparing to the evaluation under the function s, the probability of a top-k candidate answer set with t is non-increasing, while the probability of a top-k candidate answer set without t is non-decreasing[1].

Since any top-k answer set to R^p under the function s does not contain t, it follows from the above analysis that any top-k candidate answer set containing t will not be a top-k answer set to R^p under the new function s', and thus t is still a loser.

(12) U-kRanks violates *Stability*.

The following is a counterexample.

Say $k = 2$, R^p is simple. $R = \{t_1, t_2, t_3\}$, $t_1 >_s t_2 >_s t_3$. $p(t_1) = 0.3, p(t_2) = 0.4, p(t_3) = 0.3$.

	t_1	t_2	t_3
rank 1	0.3	0.28	0.126
rank 2	0	0.12	0.138
rank 3	0	0	0.036

By U-kRanks, the top-2 answer set is $\{t_1, t_3\}$.

Now raise the score of t_3 such that $t_1 >_{s'} t_3 >_{s'} t_2$.

	t_1	t_3	t_2
rank 1	0.3	0.21	0.196
rank 2	0	0.09	0.168
rank 3	0	0	0.036

[1]Here, any subset of R with cardinality at most k that is not a top-k candidate answer set under the function s is conceptually regarded as a top-k candidate answer set with probability zero under the function s.

By U-kRanks, the top-2 answer set is $\{t_1, t_2\}$. By raising the score of t_3, we actually turn the winner t_3 to a loser, which contradicts *Stability*.

Appendix B

Proofs

B.1 Proof for Proposition 3.3.1

Proposition 3.3.1. *Given a simple probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$ and an injective scoring function s over R^p, if $R = \{t_1, t_2, \ldots, t_n\}$ and $t_1 \succ_s t_2 \succ_s \ldots \succ_s t_n$, the following recursion on Global-Top$k$ queries holds.*

$$q(k,i) = \begin{cases} 0 & k = 0 \\ p(t_i) & 1 \leqslant i \leqslant k \\ (q(k,i-1)\dfrac{\bar{p}(t_{i-1})}{p(t_{i-1})} + q(k-1,i-1))p(t_i) & \textit{otherwise} \end{cases}$$

where $q(k,i) = P_{k,s}(t_i)$ and $\bar{p}(t_{i-1}) = 1 - p(t_{i-1})$.

Proof. By induction on k and i.

- Base case.
  - $k = 0$

    For any $W \in pwd(R^p)$, $top_{0,s}(W) = \varnothing$. Therefore, for any $t_i \in R$, the Global-Top$k$ probability of t_i is 0.
  - $k > 0$ and $i = 1$

    t_1 has the highest score among all tuples in R. As long as tuple t_1 appears in a possible world W, it will be in the $top_{k,s}(W)$. So the Global-Top$k$ probability of t_i is the probability that t_1 appears in possible worlds, i.e., $q(k,1) = p(t_1)$.

- Inductive step.

  Assume the theorem holds for $0 \leqslant k \leqslant k_0$ and $1 \leqslant i \leqslant i_0$. For any $W \in pwd(R^p)$, $t_{i_0} \in top_{k_0,s}(W)$ iff $t_{i_0} \in W$ and there are at most $k_0 - 1$ tuples with a higher score in W. Note that any tuple with score lower than the score of t_{i_0} does not have any influence on $q(k_0, i_0)$, because its presence/absence in a possible world will not affect the presence of t_{i_0} in the top-k answer set of that world.

  Since all the tuples are independent,

$$q(k_0,i_0) = p(t_{i_0}) \sum_{\substack{W \in pwd(R^p) \\ |\{t | t \in W \wedge t \succ_s t_{i_0}\}| < k_0}} Pr(W).$$

  (1) $q(k_0, i_0+1)$ is the Global-Topk_0 probability of tuple t_{i_0+1}.

$$\begin{aligned} q(k_0, i_0+1) &= \sum_{\substack{W \in pwd(R^p) \\ t_{i_0+1} \in top_{k_0,s}(W) \\ t_{i_0} \in top_{k_0,s}(W)}} Pr(W) \\ &+ \sum_{\substack{W \in pwd(R^p) \\ t_{i_0+1} \in top_{k_0,s}(W) \\ t_{i_0} \in W,\ t_{i_0} \notin top_{k_0,s}(W)}} Pr(W) \\ &+ \sum_{\substack{W \in pwd(R^p) \\ t_{i_0+1} \in top_{k_0,s}(W) \\ t_{i_0} \notin W}} Pr(W). \end{aligned}$$

  For the first part of the left hand side,

$$\sum_{\substack{W \in pwd(R^p) \\ t_{i_0+1} \in top_{k_0,s}(W) \\ t_{i_0} \in top_{k_0-1,s}(W)}} Pr(W) = p(t_{i_0+1}) q(k_0 - 1, i_0).$$

  The second part is zero. Since $t_{i_0} \succ_s t_{i_0+1}$, if $t_{i_0+1} \in top_{k_0,s}(W)$ and $t_{i_0} \in W$, then $t_{i_0} \in top_{k_0,s}(W)$.

  The third part is the sum of the probabilities of all possible worlds such that

$t_{i_0+1} \in W, t_{i_0} \notin W$ and there are at most $k_0 - 1$ tuples with score higher than the score of t_{i_0} in W. So it is equivalent to

$$\begin{aligned} & p(t_{i_0+1})\overline{p}(t_{i_0}) \sum_{|\{t|t\in W \wedge t>_s t_{i_0}\}|<k_0} Pr(W) \\ = & \; p(t_{i_0+1})\overline{p}(t_{i_0})\frac{q(k_0, i_0)}{p(t_{i_0})}. \end{aligned}$$

Altogether, we have

$$\begin{aligned} & q(k_0, i_0 + 1) \\ = & \; p(t_{i_0+1})q(k_0 - 1, i_0) + p(t_{i_0+1})\overline{p}(t_{i_0})\frac{q(k_0, i_0)}{p(t_{i_0})} \\ = & \; (q(k_0 - 1, i_0) + q(k_0, i_0)\frac{\overline{p}(t_{i_0})}{p(t_{i_0})})p(t_{i_0+1}). \end{aligned}$$

(2) $q(k_0 + 1, i_0)$ is the Global-Top$(k_0 + 1)$ probability of tuple t_{i_0}. Use a similar argument as above, it can be shown that this case is correctly computed by Equation (3.3) as well.

□

B.2 Proof for Theorem 3.3.2

Theorem 3.3.2 (Correctness of Algorithm 1^{TA}). *Given a simple probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, a non-negative integer k and an injective scoring function s over R^p, Algorithm 1^{TA} correctly finds a Global-Top$k$ top-k answer set.*

Proof. In every iteration of Step (2), say $\underline{t} = t_i$, for any unseen tuple t, s' is an injective scoring function over R^p, which only differs from s in the score of t. Under the function s', $t_i >_{s'} t >_{s'} t_{i+1}$. If we evaluate the top-k query in R^p under the function s' instead of s, $P_{k,s'}(t) = \frac{p(t)}{\underline{p}}UP$. On the other hand, for any $W \in pwd(R^p)$, the fact that W contributes to $P_{k,s}(t)$ implies that W contributes to $P_{k,s'}(t)$, while the reverse is not necessarily true. So, we have $P_{k,s'}(t) \geqslant P_{k,s}(t)$. Recall that $\underline{p} \geqslant p(t)$, therefore $UP \geqslant \frac{p(t)}{\underline{p}} UP = P_{k,s'}(t) \geqslant P_{k,s}(t)$. The conclusion follows from the correctness of the original TA algorithm and Algorithm 1. $\square$

B.3 Proof for Proposition 3.3.2

Lemma B.3.1. *Let $R^p = \langle R, p, \mathcal{C} \rangle$ be a probabilistic relation, s an injective scoring function, $t \in R$, and $E^p = \langle E, p^E, \mathcal{C}^E \rangle$ the event relation induced by t. Define $Q^p = \langle E - \{t_{e_t}\}, p^E, \mathcal{C}^E - \{\{t_{e_t}\}\} \rangle$. Then, the Global-Top$k$ probability of t satisfies the following:*

$$P_{k,s}^{R^p}(t) = p(t) \sum_{\substack{W_e \in pwd(Q^p) \\ |W_e| < k}} Pr(W_e).$$

Proof. Given $t \in R$, k and s, let A be a subset of $pwd(R^p)$ such that $W \in A \Leftrightarrow t \in top_{k,s}(W)$. If we group all the possible worlds in A by the set of parts whose tuple in W has a score higher than that of t, then we will have the following partition:

$$A = A_1 \cup A_2 \cup \ldots \cup A_q, A_i \cap A_j = \emptyset, i \neq j$$

and

$$\begin{aligned}
&\forall A_i, \forall W_1, W_2 \in A_i, i = 1, 2, \ldots, q, \\
&\{C_j | \exists t' \in W_1 \cap C_j, t' >_s t\} = \{C_j | \exists t' \in W_2 \cap C_j, t' >_s t\}.
\end{aligned}$$

Moreover, denote $CharParts(A_i)$ to A_i's characteristic set of parts.

Now, let B be a subset of $pwd(Q^p)$, such that $W_e \in B \Leftrightarrow |W_e| < k$. There is a bijection $g : \{A_i | A_i \in A\} \to B$, mapping each part A_i in A to a possible world in B which contains only tuples corresponding to the parts in A_i 's characteristic set.

$$g(A_i) = \{t_{e_{C_j}} | C_j \in CharParts(A_i)\}.$$

The following equation holds from the definition of an induced event relation (Definition 3.3.1).

$$\begin{aligned}\sum_{W \in A_i} Pr(W) &= p(t) \prod_{C_i \in CharParts(A_i)} p(t_{e_{C_i}}) \prod_{\substack{C_i \in \mathcal{C} - \{C_{id(t)}\} \\ C_i \notin CharParts(A_i)}} (1 - p(t_{e_{C_i}})) \\ &= p(t) Pr(g(A_i)).\end{aligned}$$

Therefore,

$$\begin{aligned}P^{R^p}_{k,s}(t) &= \sum_{W \in A} Pr(W) = \sum_{i=1}^{q} (\sum_{W \in A_i} Pr(W)) \\ &= \sum_{i=1}^{q} p(t) Pr(g(A_i)) = p(t) \sum_{i=1}^{q} Pr(g(A_i)) \\ &= p(t) \sum_{W_e \in B} Pr(W_e) \quad (g \text{ is a bijection}) \\ &= p(t) (\sum_{\substack{W_e \in pwd(Q^p) \\ |W_e| < k}} Pr(W_e)).\end{aligned}$$

□

Proposition 3.3.2 (Correctness of Algorithm 4). *Given a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$ and an injective scoring function s, for any $t \in R^p$, the Global-Top$k$ probability of t equals the Global-Top$k$ probability of t_{e_t} when evaluating top-k in the induced event relation $E^p = \langle E, p^E, \mathcal{C}^E \rangle$ under the injective scoring function $s^E : E \to \mathbb{R}, s^E(t_{e_t}) = \frac{1}{2}$ and $s^E(t_{e_{C_i}}) = i$:*

$$P^{R^p}_{k,s}(t) = P^{E^p}_{k,s^E}(t_{e_t}).$$

Proof. Since t_{e_t} has the lowest score under the function s^E, for any $W_e \in pwd(E^p)$, the only chance $t_{e_t} \in top_{k,s^E}(W_e)$ is when there are at most k tuples in W_e, including t_{e_t}.

$$\forall W_e \in pwd(E^p), t_{e_t} \in top_{k,s}(W_e) \Leftrightarrow (t_{e_t} \in W_e \wedge |W_e| \leqslant k).$$

Therefore,

$$P^{E^p}_{k,s^E}(t_{e_t}) = \sum_{t_{e_t} \in W_e \wedge |W_e| \leqslant k} Pr(W_e).$$

In the proof of Lemma B.3.1, B contains all the possible worlds having at most $k-1$ tuples from $E - \{t_{e_t}\}$. By Definition 3.3.1,

$$\sum_{t_{e_t} \in W_e \wedge |W_e| \leqslant k} Pr(W_e) = p(t) \sum_{W'_e \in B} Pr(W'_e).$$

By Lemma B.3.1,

$$p(t) \sum_{W'_e \in B} Pr(W'_e) = P_{k,s}^{R^p}(t).$$

Consequently,

$$P_{k,s}^{R^p}(t) = P_{k,s^E}^{E^p}(t_{e_t}).$$

□

B.4 Proof for Proposition 4.2.1

Proposition 4.2.1 (Correctness of Algorithm 5). *Let $R^p = \langle R, p, \mathcal{C} \rangle$ be a simple probabilistic relation where $R = \{t_1, \ldots, t_n\}$, $t_1 \succeq_s t_2 \succeq_s \ldots \succeq_s t_n$, k a non-negative integer and s a scoring function. For every $t_l \in R$, the Global-Top$k$ probability of t_l can be computed by the following equation:*

$$P_{k,s}^{R^p}(t_l) = \sum_{k'=0}^{k-1} T_{k',[i_l]} \cdot P_{k-k',s}^{R_s^p(t_l)}(t_l)$$

where $R_s^p(t_l)$ is R^p restricted to $\{t \in R | t \sim_s t_l\}$.

Proof. Given a tuple $t_l \in R$, let R_θ be the support relation R restricted to $\{t \in R | t \; \theta \; t_l\}$, and R_θ^p be R^p restricted to R_θ, where $\theta \in \{\succ, \sim, \prec, \preceq\}$ (subscript s omitted). Similarly, for each possible world $W \in pwd(R^p)$, $W_\theta = W \cap R_\theta$.

Each possible world $W \in pwd(R^p)$ such that $t_l \in all_{k,s}(W)$ contributes $\min(1, \frac{k-a}{b})Pr(W)$ probability to $P_{k,s}^{R^p}(t_l)$, where $a = |W_\succ|$ and $b = |W_\sim|$.

$$
\begin{aligned}
P_{k,s}^{R^p}(t_l) &= \sum_{\substack{W \in pwd(R^p), t_l \in W \\ |W_>|=a, 0 \leqslant a \leqslant k-1 \\ |W_\sim|=b, 1 \leqslant b \leqslant m}} \min(1, \frac{k-a}{b}) Pr(W) \\
&= \sum_{a=0}^{k-1} \sum_{b=1}^{m} \min(1, \frac{k-a}{b}) (\sum_{\substack{W \in pwd(R^p), t_l \in W \\ |W_>|=a, |W_\sim|=b}} Pr(W)) \\
&= \sum_{a=0}^{k-1} \sum_{b=1}^{m} \min(1, \frac{k-a}{b}) (\sum_{\substack{W_> \in pwd(R^p_>) \\ |W_>|=a}} Pr(W_>) \sum_{\substack{W_\preceq \in pwd(R^p_\preceq), t_l \in W_\preceq \\ |W_\sim|=b}} Pr(W_\preceq)) \\
&= \sum_{a=0}^{k-1} (\sum_{\substack{W_> \in pwd(R^p_>) \\ |W_>|=a}} Pr(W_>) \sum_{b=1}^{m} \min(1, \frac{k-a}{b}) (\sum_{\substack{W_\preceq \in pwd(R^p_\preceq), t_l \in W_\preceq \\ |W_\sim|=b}} Pr(W_\preceq))) \\
&= \sum_{a=0}^{k-1} (T_{a,[i_l]} \sum_{b=1}^{m} \min(1, \frac{k-a}{b}) (\sum_{\substack{W_\sim \in pwd(R^p_\sim), t_l \in W_\sim \\ |W_\sim|=b}} Pr(W_\sim) \sum_{W_\prec \in pwd(R^p_\prec)} Pr(W_\prec))) \\
&= \sum_{a=0}^{k-1} (T_{a,[i_l]} \sum_{b=1}^{m} \min(1, \frac{k-a}{b}) (\sum_{\substack{W_\sim \in pwd(R^p_\sim), t_l \in W_\sim \\ |W_\sim|=b}} Pr(W_\sim))) \\
&= \sum_{a=0}^{k-1} T_{a,[i_l]} \cdot P_{k-a,s}^{R^p_s(t_l)}(t_l)
\end{aligned}
$$

where m is the number of tuples tying with t_l (inclusive), i.e., $m = |R^p_s(t_l)|$. □

B.5 Proof for Proposition 4.3.1

Proposition 4.3.1. *Given a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$ and a scoring function s, for any $t \in R^p$, the Global-Top$k$ probability of t equals the Global-Top$k$ probability of $t_{e_t,\sim}$ when evaluating top-k in the induced event relation $E^p = \langle E, p^E, \mathcal{C}^E \rangle$ under the scoring function $s^E : E \to \mathbb{R}$, $s^E(t_{e_t,>}) = \frac{1}{2}$, $s^E(t_{e_t,\sim}) = \frac{1}{2}$, $s^E(t_{e_{C_i},\sim}) = \frac{1}{2}$ and $s^E(t_{e_{C_i},>}) = i$:*

$$P_{k,s}^{R^p}(t) = P_{k,s^E}^{E^p}(t_{e_t,\sim}).$$

Proof. Similar to what we did in the proof for Lemma B.3.1, we are trying to create a bijection.

Given $t \in R$, k and s, let A be a subset of $pwd(R^p)$ such that $W \in A \Leftrightarrow t \in all_{k,s}(W)$. If we group all the possible worlds in A by the set of parts whose tuple in W has a score higher than or equal to that of t, then we will have the following partition:

$$A = A_1 \cup A_2 \cup \ldots \cup A_q, A_i \cap A_j = \varnothing, i \neq j$$

and

$$\begin{aligned}
&\forall A_i, \forall W_1, W_2 \in A_i, i = 1, 2, \ldots, q, \\
&\{C_{j,>} | \exists t' \in W_1 \cap C_j, t' >_s t\} = \{C_{j,>} | \exists t' \in W_2 \cap C_j, t' >_s t\} \\
&\text{and} \\
&\{C_{j,\sim} | \exists t' \in W_1 \cap C_j, t' \sim_s t\} = \{C_{j,\sim} | \exists t' \in W_2 \cap C_j, t' \sim_s t\}.
\end{aligned}$$

Moreover, denote $CharParts(A_i)$ to A_i's characteristic set of parts. Note that every $W \in A_i$ has the same allocation coefficient $\alpha(t, W)$, denoted by α_i.

Now, let B be a subset of $pwd(E^p)$, such that $W_e \in B \Leftrightarrow t_{e_t,\sim} \in all_{k,s}(W_e)$. There is a bijection $g : \{A_i | A_i \in A\} \rightarrow B$, mapping each part A_i in A to the a possible world in B which contains only tuples corresponding to parts in A_i 's characteristic set.

$$g(A_i) \quad = \quad \{t_{e_{C_j},>} | C_{j,>} \in CharParts(A_i)\} \cup \{t_{e_{C_j},\sim} | C_{j,\sim} \in CharParts(A_i)\}$$

Furthermore, the allocation coefficient α_i of A_i equals to the allocation coefficient $\alpha(t_{e_t,\sim}, g(A_i))$ under the function s^E.

The following equation holds from the definition of an induced event relation under general scoring functions (Definition 4.3.1).

$$\begin{aligned}
\sum_{W \in A_i} Pr(W) \quad &= \quad \prod_{C_{i,>} \in CharParts(A_i)} p(t_{e_{C_i},>}) \prod_{C_{i,\sim} \in CharParts(A_i)} p(t_{e_{C_i},\sim}) \\
&\quad \prod_{\substack{C_i \in \mathcal{C} \\ C_{i,\sim} \notin CharParts(A_i) \\ C_{i,>} \notin CharParts(A_i)}} (1 - p(t_{e_{C_i},>}) - p(t_{e_{C_i},\sim})) \\
&= \quad Pr(g(A_i)).
\end{aligned}$$

Therefore,

$$\begin{aligned}
P_{k,s}^{R^p}(t) &= \sum_{W \in A} \alpha(t, W) Pr(W) = \sum_{i=1}^{q} (\alpha_i \sum_{W \in A_i} Pr(W)) \\
&= \sum_{i=1}^{q} \alpha_i Pr(g(A_i)) = \sum_{i=1}^{q} \alpha(t_{e_t,\sim}, g(A_i)) Pr(g(A_i)) \\
&= \sum_{W_e \in B} \alpha(t_{e_t,\sim}, W_e) Pr(W_e) \quad (g \text{ is a bijection}) \\
&= P_{k,s^E}^{E^p}(t_{e_t,\sim}).
\end{aligned}$$

□

B.6 Proof for Theorem 4.3.1

Theorem 4.3.1. *Given a probabilistic relation* $R^p = \langle R, p, \mathcal{C} \rangle$*, a scoring function* s*,* $t \in R^p$*, and its induced event relation* $E^p = \langle E, p^E, \mathcal{C}^E \rangle$*, where* $|E| = 2m$*, the recursion in Table 4.1 on* $u_>(k', i, b)$ *and* $u_\sim(k', i, b)$ *holds, where* $b_{\max}$ *is the number of tuples with a positive probability in* $E^p_\sim$*. The Global-Topk probability of* $t_{e_t,\sim}$ *in* E^p *under the scoring function* s^E *can be computed by the following equation:*

$$\begin{aligned}
P_{k,s^E}^{E^p}(t_{e_t,\sim}) &= P_{k,s^E}^{E^p}(t_{m,\sim}) \\
&= \sum_{b=1}^{b_{\max}} (\sum_{k'=1}^{k} u_\sim(k', m, b) + \sum_{k'=k+1}^{k+b-1} \frac{k-(k'-b)}{b} u_\sim(k', m, b)) (4.8)
\end{aligned}$$

Proof. Equation (4.8) follows from Equation (4.6) and Equation (4.7) as it is a simple enumeration based on Definition 4.1.1. We are going to prove Equation (4.6) and Equation (4.7) by an induction on i.

- Base case: $i = 1, 0 \leqslant k' \leqslant m$ and $0 \leqslant b \leqslant b_{\max}$

  When $i = 1$, based on the definition of u, the only non-zero entries are $u_>(1, 1, 0)$ and $u_\sim(1, 1, 1)$. By definition, $u_>(1, 1, 0)$ is the probability sum of all possible worlds which contain $t_{1,>}$ and do not contain $t_{1,\sim}$. The second requirement is redundant since those two tuples are exclusive. Therefore, it is simply the probability of $t_{1,>}$. Similarly, by definition, $u_{\sim}(1,1,1)$ is the probability sum of all possible worlds which contain $t_{1,\sim}$ and do not contain $t_{1,>}$. Again, it is simply the probability of $t_{1,\sim}$. It is easy to check that no possible worlds satisfy other combinations of k' and b when $i = 1$, therefore their probabilities are 0.

- Inductive step.

  Assume the theorem holds for $i \leqslant i_0$, $0 \leqslant k' \leqslant m$ and $0 \leqslant b \leqslant b_{\max}$, where $1 \leqslant i_0 \leqslant m-1$.

  Denote by $E_{>,[i]}$ and $E_{\sim,[i]}$ the set of the first i tuples in $E_{>}$ and $E_{\sim}$, respectively.

  For any $W \in pwd(E^p)$, by definition, W contributes to $u_{>/\sim}(k', i_0, b)$ iff $t_{i_0,>/\sim} \in W \wedge |W \cap (E_{>,[i_0]} \cup E_{\sim,[i_0]})| = k' \wedge |W \cap E_{\sim,[i_0]}| = b$. Since $E_{>,[i_0]} \cap E_{\sim,[i_0]} = \varnothing$, we have:

  W contributes to $u_{>/\sim}(k', i_0, b) \Leftrightarrow t_{i_0,>/\sim} \in W \wedge |W \cap E_{>,[i_0]}| = k' - b \wedge |W \cap E_{\sim,[i_0]}| = b$.

  (1) $u_{>}(k', i_0+1, b)$ is the probability sum of all possible worlds W such that

$t_{i_0+1,>} \in W \wedge |W \cap E_{>,[i_0+1]}| = k' - b \wedge |W \cap E_{\sim,[i_0+1]}| = b.$

$$\begin{aligned}
u_>(k', i_0+1, b) &= \sum_{\substack{W \in pwd(E^p), t_{i_0+1,>} \in W \\ |W \cap E_{>,[i_0+1]}| = k'-b \\ |W \cap E_{\sim,[i_0+1]}| = b}} Pr(W) \\
&= \sum_{\substack{W \in pwd(E^p), t_{i_0+1,>} \in W \\ |W \cap E_{>,[i_0]}| = k'-1-b \\ |W \cap E_{\sim,[i_0]}| = b}} Pr(W) \quad \text{(Since } t_{i_0+1,>} \in W, \ t_{i_0+1,\sim} \notin W\text{)} \\
&= \sum_{\substack{W \in pwd(E^p) \\ t_{i_0+1,>} \in W, t_{i_0,>} \in W \\ |W \cap E_{>,[i_0]}| = k'-1-b \\ |W \cap E_{\sim,[i_0]}| = b}} Pr(W) \\
&+ \sum_{\substack{W \in pwd(E^p) \\ t_{i_0+1,>} \in W, t_{i_0,\sim} \in W \\ |W \cap E_{>,[i_0]}| = k'-1-b \\ |W \cap E_{\sim,[i_0]}| = b}} Pr(W) \\
&+ \sum_{\substack{W \in pwd(E^p) \\ t_{i_0+1,>} \in W, t_{i_0,>} \notin W, t_{i_0,\sim} \notin W \\ |W \cap E_{>,[i_0]}| = k'-1-b \\ |W \cap E_{\sim,[i_0]}| = b}} Pr(W)
\end{aligned}$$

For the first part of the left hand side,

$$\begin{aligned}
\sum_{\substack{W \in pwd(E^p) \\ t_{i_0+1,>} \in W, t_{i_0,>} \in W \\ |W \cap E_{>,[i_0]}| = k'-1-b \\ |W \cap E_{\sim,[i_0]}| = b}} Pr(W) &= p(t_{i_0+1}) \sum_{\substack{W \in pwd(E^p), t_{i_0,>} \in W \\ |W \cap E_{>,[i_0]}| = k'-1-b \\ |W \cap E_{\sim,[i_0]}| = b}} Pr(W) \\
&= p(t_{i_0+1}) u_>(k'-1, i_0, b).
\end{aligned}$$

For the second part of the left hand side,

$$\begin{aligned}
\sum_{\substack{W \in pwd(E^p) \\ t_{i_0+1,>} \in W, t_{i_0,\sim} \in W \\ |W \cap E_{>,[i_0]}| = k'-1-b \\ |W \cap E_{\sim,[i_0]}| = b}} Pr(W) &= p(t_{i_0+1}) \sum_{\substack{W \in pwd(E^p), t_{i_0,\sim} \in W \\ |W \cap E_{>,[i_0]}| = k'-1-b \\ |W \cap E_{\sim,[i_0]}| = b}} Pr(W) \\
&= p(t_{i_0+1}) u_\sim(k'-1, i_0, b).
\end{aligned}$$

For the third part of the left hand side, if $p(t_{i_0,\succ}) + p(t_{i_0,\sim}) = 1$, then there is no possible world satisfying this condition, therefore it is zero. Otherwise,

$$\sum_{\substack{W \in pwd(E^p) \\ t_{i_0+1,\succ} \in W \\ t_{i_0,\succ} \notin W, t_{i_0,\sim} \notin W \\ |W \cap E_{\succ,[i_0]}| = k'-1-b \\ |W \cap E_{\sim,[i_0]}| = b}} Pr(W) = p(t_{i_0+1}) \sum_{\substack{W \in pwd(E^p) \\ t_{i_0,\succ} \notin W, t_{i_0,\sim} \notin W \\ |W \cap E_{\succ,[i_0]}| = k'-1-b \\ |W \cap E_{\sim,[i_0]}| = b}} Pr(W) \quad \text{(B.1)}$$

Equation (B.1) can be computed either by Equation (B.2) when $p(t_{i_0}, \succ) > 0$ or by Equation (B.3) when $p(t_{i_0}, \sim) > 0$ and $b < b_{\max}$. Notice that at least one of $p(t_{i_0}, \succ)$ and $p(t_{i_0}, \sim)$ is positive, otherwise neither tuple is in the induced event relation E^p according to Definition 4.3.1.

$$\begin{aligned}
&\sum_{\substack{W \in pwd(E^p) \\ t_{i_0,\succ} \notin W, t_{i_0,\sim} \notin W \\ |W \cap E_{\succ,[i_0]}| = k'-1-b \\ |W \cap E_{\sim,[i_0]}| = b}} Pr(W) \\
= \; & \frac{1 - p(t_{i_0,\succ}) - p(t_{i_0,\sim})}{p(t_{i_0,\succ})} \sum_{\substack{W \in pwd(E^p), t_{i_0,\succ} \in W \\ |W \cap E_{\succ,[i_0]}| = k'-b \\ |W \cap E_{\sim,[i_0]}| = b}} Pr(W) \\
= \; & \frac{1 - p(t_{i_0,\succ}) - p(t_{i_0,\sim})}{p(t_{i_0,\succ})} u_{\succ}(k', i_0, b). \qquad \text{(B.2)}
\end{aligned}$$

$$\begin{aligned}
&\sum_{\substack{W \in pwd(E^p) \\ t_{i_0,\succ} \notin W, t_{i_0,\sim} \notin W \\ |W \cap E_{\succ,[i_0]}| = k'-1-b \\ |W \cap E_{\sim,[i_0]}| = b}} Pr(W) \\
= \; & \frac{1 - p(t_{i_0,\succ}) - p(t_{i_0,\sim})}{p(t_{i_0,\sim})} \sum_{\substack{W \in pwd(E^p), t_{i_0,\sim} \in W \\ |W \cap E_{\succ,[i_0]}| = k'-1-b \\ |W \cap E_{\sim,[i_0]}| = b+1}} Pr(W) \\
= \; & \frac{1 - p(t_{i_0,\succ}) - p(t_{i_0,\sim})}{p(t_{i_0,\sim})} u_{\sim}(k', i_0, b+1). \qquad \text{(B.3)}
\end{aligned}$$

A subtlety is that when $p(t_{i_0}, >) = 0$ and $b = b_{\max}$, neither Equation (B.2) nor Equation (B.3) applies. However, in this case, one of the conditions in Equation (B.1) is that $|W \cap E_{\sim,[i_0]}| = b = b_{\max}$, which implies $i_0 = m$. Otherwise, the world W does not have enough tuples from $E_{\sim}$. On the other hand, we know that $i_0 \leqslant m - 1$. Therefore, there are simply no possible worlds satisfying the condition in Equation (B.1), and Equation (B.1) equals 0.

Altogether, we show that this case can be correctly computed by Equation (4.6).

(2) $u_{\sim}(k', i_0 + 1, b)$ is the probability sum of all possible worlds W such that $t_{i_0+1,\sim} \in W \wedge |W \cap E_{>,[i_0+1]}| = k' - b \wedge |W \cap E_{\sim,[i_0+1]}| = b$. Using a similar argument as above, it can be shown that this case is correctly computed by Equation (4.7) as well.

$\square$

Appendix C

Postulates Satisfaction of Global-Topk^{β}

Semantics	Exact k	Faithfulness	Stability	Sens. to Prob.	Sens. to Score
Global-Topk^{β}	✓(1)	✓/× (2)	✓(3)	✓(4)	✓(5)

Table C.1: Postulate Satisfaction for Global-Topk^{β} ($\beta > 0$) in Table 5.1

Proof. The following proofs correspond to the numbers next to each entry in the above table.

Assume that we are given a probabilistic relation $R^p = \langle R, p, \mathcal{C} \rangle$, a non-negative integer k, an injective scoring function s and $\beta > 0$.

(1) Global-Topk^{β} satisfies *Exact* k

This is obvious. The proof is similar to that of Global-Top$k$. The only difference here is that in Global-Topk^{β}, the measure for each tuple is *Global-Topk^{β} value* instead of *Global-Top$k$ probability*.

(2) Global-Topk^{β} satisfies *Faithfulness* in simple probabilistic relations while it violates *Faithfulness* in general probabilistic relations.

Simple Probabilistic Relations

For every $\beta \in (0, 1]$, by the assumption, $t_1 \succ_s t_2$ and $p(t_1) > p(t_2)$, so we need to show that $v^{\beta}_{k,s}(t_1) > v^{\beta}_{k,s}(t_2)$, i.e. $P_{k,s}(t_1)s(t_1)^{\beta} > P_{k,s}(t_2)s(t_2)^{\beta}$. By the *Faithfulness* of Global-Top$k$ in simple probability relations, we know that $P_{k,s}(t_1) > P_{k,s}(t_2)$

in this case. Since $1 \geqslant s(t_1) > s(t_2) \geqslant 0$, and the function $f(x) = x^\beta$ is increasing on $(0, 1]$ when $\beta > 0$, the conclusion follows.

General Probabilistic Relations

We show that for every $\beta > 0$, there is always an instance, i.e. a probabilistic relation and a non-negative integer k, where the *Faithfulness* postulate is violated. The construction is as follows.

For a given β, choose three positive number $s_{\max}$, s_a and s_b such that $s_{\max} > s_a > s_b$. Then, choose p_a and p_b such that $0 < p_b < p_a < \frac{1}{1+(\frac{s_a}{s_b})^\beta}$. It is easy to verify that such p_a and p_b always exist. Let a positive integer $m = 2 + \lceil \frac{1-p_a}{p_a p_b (\frac{s_a}{s_{\max}+1})^\beta} \rceil$.

Say $k = 1$, $R = \{t_1, \ldots, t_{m-2}, t_{m-1}, t_m\}$, $t_1 \succ_s \ldots \succ_s t_m$, $s(t_1) = s_{\max}$, $s(t_{m-1}) = s_a$, $s(t_m) = s_b$, and $\{t_1, \ldots, t_{m-2}, t_m\}$ are exclusive. $p(t_i) = \frac{1-p_a}{m-2}, i = 1 \ldots m-2$, $p(t_{m-1}) = p_a$, $p(t_m) = p_b$.

By Global-Topk^β, the top-1 answer is $\{t_m\}$ because

(i) for t_{m-1}, since $0 < p_b < p_a < \frac{1}{1+(\frac{s_a}{s_b})^\beta}$, we have $v^\beta_{k,s}(t_{m-1}) = P_{k,s}(t_{m-1}) s(t_{m-1})^\beta = p_a p_b s_a^\beta < (1-p_a) p_b s_b^\beta = v^\beta_{k,s}(t_m)$;

(ii) for every $1 \leqslant i \leqslant m-2$, $v^\beta_{k,s}(t_i) = P_{k,s}(t_i) s(t_i)^\beta = \frac{1}{m-2}(1-p_a) s(t_i)^\beta$. Since $m \geqslant 2 + \frac{1-p_a}{p_a p_b (\frac{s_a}{s_{\max}+1})^\beta}$, $p(t_i) = \frac{1-p_a}{m-2} < p_a p_b (\frac{s_a}{s_{\max}+1})^\beta$. Therefore, $v^\beta_{k,s}(t_i) = p(t_i) s(t_i)^\beta < p(t_i)(s_{\max}+1)^\beta \leqslant p_a p_b (s_a)^\beta = v^\beta_{k,s}(t_{m-1})$.

However, $t_{m-1} \succ_s t_m$ and $p(t_{m-1}) > p(t_m)$, which violates *Faithfulness*.

(3) Global-Topk^β satisfies *Stability*.

Part I: Probability.

Case 1: Winners.

For any winner $t \in A$, if we only raise the probability of t, we have a new probabilistic relation $(R^p)' = \langle R, p', \mathcal{C} \rangle$, where the new probability function p' is such that $p'(t) > p(t)$ and for any $t' \in R, t' \neq t, p'(t') = p(t')$. Similar to the proof of *Faithfulness* of Global-Top$k$, we have $P^{(R^p)'}_{k,s}(t) = \frac{p'(t)}{p(t)} P^{R^p}_{k,s}(t)$. For any tuple $t' \in R$ other than

t, we have $P_{k,s}^{(R^p)'}(t') \leqslant \frac{p'(t')}{p(t')} P_{k,s}^{R^p}(t')$. Since scoring function s remains the same, if $v_{k,s}^{\beta,R^p}(t)$ is among the k highest Global-Topk^β values under R^p, so is $v_{k,s}^{\beta,R^p}(t)$ under $(R^p)'$.

Case 2: Losers. s This case is similar to *Case 1.*

Part II: Score.

Case 1: Winners.

For any winner $t \in A$, we evaluate R^p under a new general scoring function s'. Comparing to s, s' only raises the score of t. That is, $s'(t) > s(t)$ and for any $t' \in R, t' \neq t, s'(t') = s(t')$. The proof of the *Faithfulness* for Global-Top$k$ shows that the Global-Top$k$ probability of t is non-decreasing while that of $t' \in R, t' \neq t$, is non-increasing. As the Global-Topk^β value is positively correlated to both the Global-Top$k$ probability and the score, $v_{k,s}^{\beta,R^p}(t)$ is still among the k highest under function s'.

Case 2: Losers.

This case is similar to *Case 1.*

(4) Global-Topk^β satisfies *Sensitivity to Probability.*

We are going to show that if R^p is sufficiently large, for every $t \in R - \cup\, Ans_{k,s}(R^p)$, there is a p', such that $t \in \cup\, Ans_{k,s}((R^p)')$, where $(R^p)' = \langle R, p', \mathcal{C}\rangle$. Pick any $t \in R - \cup\, Ans_{k,s}(R^p)$. Since R^p is sufficiently large, such t always exists. Let the world $\{t\}$ have probability p_0 such that $\frac{1}{1+(s(t))^\beta} < p_0 < 1$ under $(R^p)'$. We can always achieve this by setting $p'(t)$ close to 1, and for each part $C_i \in \mathcal{C}$ not containing t, setting $\sum_{t' \in C_i}(p'(t'))$ close to 0. Since t is the top-1 tuple in the world $\{t\}$ and $s(t'') \in (0,1]$, for every $t'' \in R, t'' \neq t$, we have $v_{k,s}^{\beta,(R^p)'}(t) \geqslant p_0(s(t))^\beta > 1 - p_0 \geqslant (1-p_0)(s(t''))^\beta \geqslant v_{k,s}^{\beta,(R^p)'}(t'')$. Therefore, t has the highest Global-Topk^β value under $(R^p)'$ and is in $\cup\, Ans_{k,s}((R^p)')$.

(5) Global-Topk^β satisfies *Sensitivity to Score.*

We are going to show that if R^p is sufficiently large, for every $t \in R - \cup\, Ans_{k,s}(R^p)$ under function s, there is a function s', such that $t \in \cup\, Ans_{k,s}(R^p)$ under function

s'. Pick any $t \in R - \cup Ans_{k,s}(R^p)$. Since R^p is sufficiently large, such t always exists. Let $s'(t) = \frac{2}{3}$, and for every $t' \in R, t' \neq t$, $s'(t') = \frac{1}{3}p(t)^{\frac{1}{\beta}}$. Notice that $s'(t')$ is well-defined as $\beta > 0$. Since $s'(t) = \frac{2}{3} > \frac{1}{3} \geqslant \frac{1}{3}p(t)^{\frac{1}{\beta}} = s'(t')$, t is in the top$k$ set of every possible world containing it. Therefore, $v^{\beta}_{k,s'}(t) = p(t)s'(t)^{\beta}$. Furthermore, $v^{\beta}_{k,s'}(t) = p(t)s'(t)^{\beta} > p(t)(\frac{1}{3})^{\beta} = s'(t')^{\beta} \geqslant p(t')s'(t')^{\beta} = v^{\beta}_{k,s'}(t')$. Therefore, t has the highest Global-Topk^{β} value under function s' and is in $\cup Ans_{k,s'}(R^p)$.

□

Bibliography

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases : The Logical Level*. Addison Wesley, 1994.

[2] Iqbal Ali, D Wade Cook, and Moshe Kress. Ordinal ranking and intensity of preference: A linear programming approach. *Manage. Sci.*, 32(12):1642–1647, 1986.

[3] Mikhail J. Atallah and Yinian Qi. Computing all skyline probabilities for uncertain data. In *PODS*, pages 279–287, 2009.

[4] Omar Benjelloun, Anish Das Sarma, Alon Y. Halevy, and Jennifer Widom. Uldbs: Databases with uncertainty and lineage. In *VLDB*, 2006.

[5] George Beskales, Mohamed A. Soliman, and Ihab F. Ilyas. Efficient search for the top-k probable nearest neighbors in uncertain databases. *PVLDB*, 1(1):326–339, 2008.

[6] Maxim Binshtok, Ronen I. Brafman, Solomon Eyal Shimony, Ajay Mani, and Craig Boutilier. Computing optimal subsets. In *AAAI*, pages 1231–1236, 2007.

[7] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[8] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res. (JAIR)*, 21:135–191, 2004.

[9] Ronen I. Brafman, Carmel Domshlak, and Solomon Eyal Shimony. On graphical modeling of preference and importance. *J. Artif. Intell. Res. (JAIR)*, 25:389–424, 2006.

[10] Nicolas Bruno and Hui Wang. The threshold algorithm: From middleware systems to the relational engine. *IEEE Trans. Knowl. Data Eng.*, 19(4):523–537, 2007.

[11] Douglas Burdick, Prasad M. Deshpande, T. S. Jayram, Raghu Ramakrishnan, and Shivakumar Vaithyanathan. OLAP over uncertain and imprecise data. *VLDB J.*, 16(1):123–144, 2007.

[12] Roger Cavallo and Michael Pittarelli. The theory of probabilistic databases. In *VLDB*, 1987.

[13] Reynold Cheng, Lei Chen 0002, Jinchuan Chen, and Xike Xie. Evaluating probability threshold k-nearest-neighbor queries over uncertain data. In *EDBT*, pages 672–683, 2009.

[14] Reynold Cheng, Jinchuan Chen, Mohamed F. Mokbel, and Chi-Yin Chow. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *ICDE*, pages 973–982, 2008.

[15] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Querying imprecise data in moving object environments. *IEEE Trans. Knowl. Data Eng.*, 16(9):1112–1127, 2004.

[16] Jan Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, 2003.

[17] Jan Chomicki, Parke Godfrey, Jarek Gryz, and Dongming Liang. Skyline with presorting: Theory and optimizations. In *Intelligent Information Systems*, pages 595–604, 2005.

[18] Wade D. Cook and Moshe Kress. Ordinal ranking and intensity of preference. *Manage. Sci.*, 31(1):26–32, 1985.

[19] Graham Cormode, Feifei Li, and Ke Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE*, pages 305–316, 2009.

[20] Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.

[21] Marie desJardins and Kiri Wagstaff. DD-pref: A language for expressing preferences over sets. In *AAAI*, pages 620–626, 2005.

[22] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, pages 613–622, 2001.

[23] Ronald Fagin. Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci.*, 58(1):83–99, 1999.

[24] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. In *SODA*, pages 28–36, 2003.

[25] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.

[26] P.C. Fishburn. Preference structures and their numerical representations. *Theoretical Computer Science*, 217:359–383, 1999.

[27] Norbert Fuhr and Thomas Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.

[28] Tingjian Ge, Stanley B. Zdonik, and Samuel Madden. Top- queries on uncertain data: on score distribution and typical answers. In *SIGMOD Conference*, pages 375–388, 2009.

[29] Sudipto Guha, Dimitrios Gunopulos, Nick Koudas, Divesh Srivastava, and Michail Vlachos. Efficient approximation of optimization queries under parametric aggregation constraints. In *VLDB*, pages 778–789, 2003.

[30] Sudipto Guha, Nick Koudas, Amit Marathe, and Divesh Srivastava. Merging the results of approximate match operations. In *VLDB*, pages 636–647, 2004.

[31] Joseph Y. Halpern. An analysis of first-order logics of probability. *Artif. Intell.*, 46(3):311–350, 1990.

[32] Ming Hua, Jian Pei, Wenjie Zhang, and Xuemin Lin. Ranking queries on uncertain data: a probabilistic threshold approach. In *SIGMOD Conference*, pages 673–686, 2008.

[33] Ihab F. Ilyas, Walid G. Aref, and Ahmed K. Elmagarmid. Joining ranked inputs in practice. In *VLDB*, 2002.

[34] Ihab F. Ilyas, Walid G. Aref, and Ahmed K. Elmagarmid. Supporting top-k join queries in relational databases. In *VLDB*, 2003.

[35] Tomasz Imielinski and Witold Lipski. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[36] Werner Kießling. Foundations of preferences in database systems. In *VLDB*, pages 311–322, 2002.

[37] Werner Kießling and Gerhard Köstler. Preference SQL - design, implementation, experiences. In *VLDB*, pages 990–1001, 2002.

[38] Veronika Köbberling. Strength of preference and cardinal utility. *Economic Theory*, 27(2):375–391, January 2006.

[39] D.H. Krantz, R.D. Luce, P. Suppes, and A. Tversky. *Foundations of measurement*, volume 1: Additive and polynomial representations. Academic Press, New York, 1971.

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

[41] Laks V. S. Lakshmanan, Nicola Leone, Robert B. Ross, and V. S. Subrahmanian. Probview: A flexible probabilistic database system. *ACM Trans. Database Syst.*, 22(3):419–469, 1997.

[42] Jian Li, Barna Saha, and Amol Deshpande. A unified approach to ranking in probabilistic databases. *PVLDB*, 2(1):502–513, 2009.

[43] Xiang Lian and Lei Chen 0002. Probabilistic ranked queries in uncertain databases. In *EDBT*, pages 511–522, 2008.

[44] Xiang Lian and Lei Chen 0002. Top-k dominating queries in uncertain databases. In *EDBT*, pages 660–671, 2009.

[45] Amélie Marian, Nicolas Bruno, and Luis Gravano. Evaluating top- queries over web-accessible databases. *ACM Trans. Database Syst.*, 29(2):319–362, 2004.

[46] http://www.infosys.uni-sb.de/projects/maybms/.

[47] Denis Mindolin and Jan Chomicki. Discovering relative importance of skyline attributes. In *VLDB*, 2009.

[48] Apostol Natsev, Yuan-Chi Chang, John R. Smith, Chung-Sheng Li, and Jeffrey Scott Vitter. Supporting incremental join queries on ranked inputs. In *VLDB*, 2001.

[49] Dan Olteanu, Christoph Koch, and Lyublena Antova. World-set decompositions: Expressiveness and efficient algorithms. *Theor. Comput. Sci.*, 403(2-3):265–284, 2008.

[50] Jian Pei, Bin Jiang, Xuemin Lin, and Yidong Yuan. Probabilistic skylines on uncertain data. In *VLDB*, pages 15–26, 2007.

[51] Christopher Ré, Nilesh N. Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, 2007.

[52] Mohamed A. Soliman, Ihab F. Ilyas, and Kevin Chen-Chuan Chang. Top-k query processing in uncertain databases. In *ICDE*, 2007.

[53] Mohamed A. Soliman, Ihab F. Ilyas, and Kevin Chen-Chuan Chang. Probabilistic top- and ranking-aggregate queries. *ACM Trans. Database Syst.*, 33(3), 2008.

[54] Jean-Claude Vansnick. Strength of preference theorectial and practical aspects. *Operational Research*, pages 449–463, 1984.

[55] Jennifer Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, 2005.

[56] Ke Yi, Feifei Li, George Kollios, and Divesh Srivastava. Efficient processing of top-k queries in uncertain databases. In *ICDE*, pages 1406–1408, 2008.

[57] Wenjie Zhang, Xuemin Lin, Ying Zhang, Wei Wang 0011, and Jeffrey Xu Yu. Probabilistic skyline operator over sliding windows. In *ICDE*, pages 1060–1071, 2009.

[58] Xi Zhang and Jan Chomicki. On the semantics and evaluation of top-k queries in probabilistic databases. In *ICDE Workshops*, pages 556–563, 2008.

[59] Xi Zhang and Jan Chomicki. Profiling sets for preference querying. In *SEBD*, pages 34–44, 2008.

[60] Xi Zhang and Jan Chomicki. Semantics and evaluation of top- queries in probabilistic databases. *Distributed and Parallel Databases*, 26(1):67–126, 2009.

[61] Esteban Zimányi. Query evaluation in probabilistic relational databases. *Theor. Comput. Sci.*, 171(1-2):179–219, 1997.