OLAP over Probabilistic Data Cubes II

Parallel Materialization and Extended Aggregates

Xie, X.; Zou, K.; Hao, X.; Pedersen, T. B.; Jin, Peiquan; Yang, W.

Published in:
IEEE Transactions on Knowledge and Data Engineering

DOI (link to publication from Publisher):
10.1109/TKDE.2019.2913420

Publication date:
2020

Document Version
Accepted author manuscript, peer reviewed version

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2913420, IEEE Transactions on Knowledge and Data Engineering

OLAP over Probabilistic Data Cubes II: Parallel Materialization and Extended Aggregates

Xike Xie†, Kai Zou†, Xingjun Hao†, Torben Bach Pedersen#, Peiquan Jin†, and Wei Yang†

Abstract—On-Line Analytical Processing (*OLAP*) enables powerful analytics by quickly computing aggregate values of numerical measures over multiple hierarchical dimensions for massive datasets. However, many types of source data, e.g., from GPS, sensors, and other measurement devices, are intrinsically inaccurate (imprecise and/or uncertain) and thus OLAP cannot be readily applied. In this paper, we address the resulting *data veracity* problem in OLAP by proposing the concept of probabilistic data cubes. Such a cube is comprised of a set of probabilistic cuboids which summarize the aggregated values in the form of probability mass functions (pmfs *in short*) and thus offer insights into the underlying data quality and enable confidence-aware query evaluation and analysis. However, the probabilistic nature of data poses computational challenges, since a probabilistic database can have exponential number of possible worlds under the possible world semantics. Even worse, it is hard to share computations among different cuboids, as aggregation functions that are distributive for traditional data cubes, e.g., `SUM`, become holistic in probabilistic settings. In this paper, we propose a complete set of techniques for probabilistic data cubes, from cuboid aggregation, over cube materialization, to query evaluation. We study two types of aggregation: convolution and sketch-based, which take polynomial time complexities for aggregation and jointly enable efficient query processing. Also, our proposal is versatile in terms of: 1) its capability of supporting common aggregation functions, i.e., `SUM`, `COUNT`, `MAX`, and `AVG`; 2) its adaptivity to different materialization strategies, e.g., full versus partial materialization, with support of our devised cost models and parallelization framework; 3) its coverage of common OLAP operations, i.e., probabilistic slicing and dicing queries. Extensive experiments over real and synthetic datasets show that our techniques are effective and scalable.

Index Terms—Probabilistic Databases, OLAP, Data Warehousing

1 Introduction

On-Line Analytical Processing (*OLAP*) enables fast interactive aggregate queries at multi-levels on large data volumes and thus facilitates decision support and business intelligence. Usually, the multidimensional data is organized in a fact table with both categorical *dimension* attributes and numerical *measure* attributes. Measure attributes are associated with aggregate functions, e.g., `SUM`, `COUNT`, `MAX`, or `AVG`, for aggregating the low-level values.

However, source data collected in real applications is often inherently inaccurate, the so-called *veracity* challenge. For example, consider a case where we want to collect and analyze temperatures at different locations at different times, which yields data as seen in Table 1 (a). Conceptually, we have two sensing *objects*, o_1 and o_2, and each object has several data *instances* (from multiple readings from the same sensor or multiple sensors at the same location): t_1 and t_2 for o_1, and t_3-t_5 for o_2. Sensor data inaccuracies can be temperature measurement errors, GPS errors, or sensor transmission delays resulting in different time values. In probabilistic databases, such imprecision is captured by modeling a random variable following a probability mass function (*pmf*), indicating the possibility of its existence. The probability can be collected by parameters of the sensing devices [1], or analysis of historical records [2].

TABLE 1: Example of Multi-dimensional Probabilistic Facts

$D_1[1]$ is spatial region {(33:50-34:50N], (121:50-122:50W]} and $D_1[2]$ is {(32:50-33:50N], (121:50-122:50W]}; $D_2[1]$ is a time interval (01/21/14:1400 -01/21/14:1500] and $D_2[2]$ is (01/21/14:1500-01/21/14:1600]

Object ID	Instance ID	Location	Time	Temperature
o_1	t_1	33:78N, 121:74W	01/21/14:1500	2
	t_2	34:39N, 122:45W	01/21/14:1500	1
o_2	t_3	34:16N, 121:65W	01/21/14:1500	1
	t_4	33:58N, 122:25W	01/21/14:1458	2
	t_5	33:43N, 122:40W	01/21/14:1515	1

(a) Multi-dimensional Facts

ObjID	Instance ID	Dimensions		Existential probability	Measure ($\mathcal{M}$)
		D_1	D_2		
o_1	t_1	$D_1[1]$	$D_2[1]$	0.6	2
	t_2	$D_1[1]$	$D_2[1]$	0.4	1
o_2	t_3	$D_1[1]$	$D_2[1]$	0.2	1
	t_4	$D_1[1]$	$D_2[1]$	0.5	2
	t_5	$D_1[2]$	$D_2[2]$	0.3	1

(b) Multi-dimensional Probabilistic Facts

In this paper, we consider *block-independent* model, where a "block" corresponds to an object and objects are independent. Within an object, instances are mutually independent and their existential probabilities sum up to 1 meaning that the object must appear in the database[1]. Then, the multi-dimensional probabilistic fact data area of schema $\langle OID, IID, D_1, D_2, ..., D_{|D|}, \mathcal{M}, prob\rangle$, where, 1) IID uniquely identifies an instance and OID identifies which object the instance belongs to; 2) D_i is a dimensional attribute; 3) $\mathcal{M}$ is a measure attribute; 4) $prob$ represents the existential probability of an instance.

An example of probabilistic facts is shown in Table 1

1. If the sum s is less than 1, we can also create a virtual instance with probability $1-s$ like [3] does. The virtual instance will not affect the completeness of the theorems proposed in this work.

- †*School of Computer Science and Technology, University of Science and Technology of China, Email:* $\{xkxie, jpq, qubit\}$*@ustc.edu.cn,* $\{slnt, hxjall\}$*@mail.ustc.edu.cn*
- #*Department of Computer Science, Aalborg University, Denmark, Email: tbp@cs.aau.dk*
- *Xike Xie is supported by NSFC (No. 61772492), Fundamental Research Funds for the Central Universities, Jiangsu NSF (No. BK20171240), and the CAS Pioneer Hundred Talents Program.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2913420, IEEE Transactions on Knowledge and Data Engineering

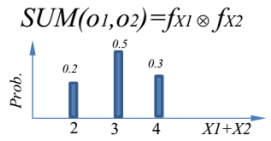
(b). There are two objects o_1 and o_2. Each object has several instances where each instance corresponds to an existential probability. In the example, each object is captured by two dimension attributes D_1 and D_2 and a measure attribute $\mathcal{M}$.

The table of probabilistic facts can be interpreted by the Possible World Semantic (*PWS*) [4], which transforms a probabilistic table into a set of non-probabilistic tables by selecting an instance from each object at a time. Each such table is called a possible world (*PWD*) W associated with a probability $p(W)$. For example, there exists a possible world $\{t_2, t_4\}$ with probability $0.4 \times 0.5 = 0.2$. By doing such, aggregate functions, e.g., `SUM`, can be run on each possible world, and then integrated to get the summary. Since each possible world has a probability, their aggregate is in the form of a pmf. PWS guarantees the correctness of query evaluation in probabilistic databases, such that an algorithm is correct if it derives the same result as if done under PWS.

Fig. 1 (b) shows the probabilistic aggregate (`SUM`) over Table 1 (b). Using PWS, one can represent Table 1 (b) by Fig. 1 (a) where each tuple is a possible world. If we consider the probability of the aggregate being 3, W_1, W_3, and W_5 are selected and the total probability is $0.12 + 0.18 + 0.2 = 0.5$. With this process repeated for all possible sums, the probabilistic aggregate pmf is shown in Fig. 1 (b). Nevertheless, the computational overhead is high. Suppose there are n objects, each of which has m instances. There can be m^n possible worlds.

Possible world W_i	$p(W_i)$	SUM
$W_1 = \{t_1, t_3\}$	0.12	3
$W_2 = \{t_1, t_4\}$	0.3	4
$W_3 = \{t_1, t_5\}$	0.18	3
$W_4 = \{t_2, t_3\}$	0.08	2
$W_5 = \{t_2, t_4\}$	0.2	3
$W_6 = \{t_2, t_5\}$	0.12	2

(a) PWS Table(`SUM`)



(b) pmf for $X_1 + X_2$

Fig. 1: Possible World Interpretation

As seen above, the aggregation of a single cell has a high computational overhead. Even worse, there are many cells in a cuboid, and many cuboids in a data cube. In a data cube, each cuboid corresponds to a `GROUP-BY` of a particular combination of dimensions. In general, there are $2^{|D|}$ cuboids for $|D|$-dimensional data. In the presence of hierarchies, there are $\prod(L_i + 1)$ cuboids, where L_i is the number of hierarchy levels of the i-th dimension [5].

The core part of OLAP is to evaluate aggregate results of different granularities, i.e., cells and cuboids. For efficient construction, previous works, e.g., [5] and [6], consider the partial order relationship between cuboids. A cuboid $\mathcal{C}^A$ is an ancestor of cuboid $\mathcal{C}$ if $\mathcal{C}$'s attributes is a subset of $\mathcal{C}^A$'s attributes. A cuboid can be materialized by grouping particular attributes over its ancestor cuboids. Similarly, the partial order relationship exists for cells. For example, cell C_3 is a descendant cell of cells C_1 an C_2 in Fig. 2. In traditional data cubes, C_2's summation can be obtained by summarizing those of C_1 and C_2. Therefore, `SUM` is a distributive operation [5]. However, basic aggregate functions, e.g., `SUM`, which are distributive for traditional data cubes become holistic in probabilistic settings. It is because the pmfs of a cell might have correlations with other parts, making aggregates of a cell conditionally dependent on other cells.

Probabilistic Data Cubes. In this work, we study the probabilistic data cube which offers confidence to the aggregation of multidimensional probabilistic data. An example is demonstrated in Table 2. The cube consists of four cuboids. Each cuboid (except the apex cuboid) has multiple cells. Each cell is associated with a set of possible values instead of a precise value, and each possible value is with a probability.

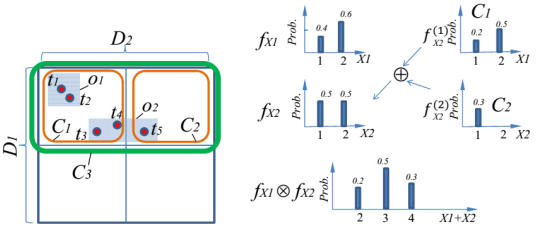


Fig. 2: Example. C_3:$\langle *, D_2[1]\rangle$; C_1:$\langle D_1[1], D_2[1]\rangle$; C_2:$\langle D_1[2], D_2[1]\rangle$

TABLE 2: Data Cube for Table 1 ($\mathcal{C}$: cuboid, $\mathcal{M}$: measure)

$\mathcal{C}$	D_1	D_2	$\mathcal{M}$
(D_1, D_2)	$D_1[1]$	$D_2[1]$	$\{(1, 0.1), (2, 0.26), (3, 0.32), (4, 0.3)\}$
	$D_1[2]$	$D_2[1]$	$\{(1, 0.3)\}$
(D_1)	$D_1[1]$	$*$	$\{(1, 0.1), (2, 0.26), (3, 0.32), (4, 0.3)\}$
	$D_1[2]$	$*$	$\{(1, 0.3)\}$
(D_2)	$*$	$D_2[1]$	$\{(2, 0.2), (3, 0.5), (4, 0.3)\}$
	$*$	$D_2[2]$	0
(all)	$*$	$*$	$\{(2, 0.2), (3, 0.5), (4, 0.3)\}$

Contributions. We propose a framework in our previous work [7] which offers a comprehensive coverage of the concepts, properties, and algorithms for OLAP on probabilistic data including the follows.

- We provide two efficient aggregation techniques satisfying PWS, namely convolution and sketch-based methods, which are with polynomial and linear time complexity, respectively.
- We investigate both full and partial cube materialization based on a proposed cost model.
- We study probabilistic OLAP queries, such as slicing and dicing.

This paper substantially extends work [7] as follows.

- We utilize parallel computation for speeding up the aggregation, and design cost models for supporting the parallel materialization (Section 4).
- We support more aggregation functions, i.e., `MAX` and `AVG`, following the cube materialization framework previously developed (Section 6).
- We present comprehensive experimental performance studies of our proposal using both synthetic and real data (Section 7).

Organization. Section 2 studies related works. Section 3 presents a convolution and sketch-based methods for aggregating a probabilistic cube. Section 4 covers parallel materialization framework as well as the corresponding cost model. Section 5 investigates two representative queries, probabilistic slicing and dicing queries. Section 6 extends the aggregating functions to `COUNT`, `MAX`, and `AVG`. Section 7 gives the experimental results. Section 8 concludes the paper. Table 3 lists all notations.

2 Related Works

Data aggregation is an important building block for OLAP systems. In particular, [8] studies a parallel algorithm for temporal aggregation which supports full bitemporal data model and share computation. [9] presents a suit of bit-parallel algorithms for accelerating aggregation operations by leveraging intra-cycle parallelism in CPU cores. [10]

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2913420, IEEE Transactions on Knowledge and Data Engineering

TABLE 3: Notations

Notation	Meaning
D, M	dimension attributes, the measure attribute
$dom(D)$	domain for dimension attributes
$dom(M)$	domain for the measure attribute
$D_i \in D$	dimension i of D, s.t. $D = D_1 \times D_2 ... \times D_{\|D\|}$
$dom(D_i)$	domain for dimension D_i
$\mathcal{C}$, $\mathcal{C}^{\mathcal{A}}$, $\mathbb{C}$	a cuboid, a ancestor cuboid of $\mathcal{C}$, cuboids in the lattice
$(D_1, ..., D_c)$	a cuboid specified by dimension attributes $D_1, ..., D_c$
I_i	an interval on dimension D_i. $I_i = [D_i^l, D_i^u]$
C	a multidimensional cell, i.e., $C = \langle I_1, I_2, ..., I_{\|D\|} \rangle$
e_i	side length of C on dimension i, or $e_i = \|I_i\|$
t, $t.prob$	an instance, and its existential probability
o_i, $o_i[j]$	an object, a truncated object, i.e., $o_i[j] = o_i \cap C_j$
X_i	a random variable for object o_i's measure value
$X_i^{(j)}$	a random variable for object $o_i[j]$'s measure value
f_{X_i}	probability mass function of object o_i's measure values
$f_{X_i}^{(j)}$	truncated pmf for object o_i w.r.t. cell C_j
f_{X+Y}	pmf for the convolution of f_X and f_Y
F_C	pmf for aggregated values of cell C
$\oplus$	plus operator for two truncated pmfs of the same object
$\otimes$	convolution operator for two pmfs
κ_k	k-th order cumulant
μ_k	k-th order moment
τ_p, τ_v	probability threshold, value threshold
W, $\mathcal{W}$	a possible world, a set of possible words
PWD, PWS	possible worlds, possible world semantic

proposes a dynamic structure for efficient bundled range aggregation, namely aBB-tree, which takes linear space and logarithm time for querying and updating. In [11], authors study efficient evaluation of iceberg cells of s-cuboids.

Many research interests are drawn on optimizing OLAP systems. For example, [12] and [13] study replication techniques to support mixed OLTP and OLAP workloads. [14] describes new concurrency control protocols providing snapshot isolation on distributed in-memory OLAP systems. [15] studies the exploration of biased queries together with techniques of rewriting biased queries into unbiased ones. However, none of these works are on probabilistic data.

There have also been research works on handling probabilistic data aggregation. In particular, [16] studies aggregation with the algebraic structures of semirings and semimodules. [17] studies dichotomies of tractable and intractable queries so that query evaluation can efficiently switch between exact and approximate versions. [18] investigates probabilistic keys in order to facilitate data processing for probabilistic data. [19] considers the aggregation representation with histograms and wavelets. [20] and [22] investigate ranking semantics for probabilistic data. [23] and [24] use frequency moments for efficient probabilistic data aggregation. Other query variants include trajectory queries [25], [26], reverse nearest neighbor queries [21], [27], and skyline queries [28], etc. However, these techniques cannot be directly used in the OLAP settings, since they do not consider measures, hierarchies, etc.

Meanwhile, the data veracity problem in OLAP, i.e., correlation between uncertain data and cells, has been recognized in [29]–[33]. However, [29] and [30] bypass the problem by resorting to higher level cuboids that fully accommodate an object. Thus, they do not address how aggregation is done on base-level cells. [31] and [32] estimate the expected belongingness of an object to a cell by allocation policies, e.g., with EM algorithms. They thus do not offer probabilistic confidence. Moreover, results with expected values or sampling might lead to nonintuitive query answers, when data values and their probabilities follow skewed and non-aligned distributions [34]. Actually, the expected value solution is a special case of our sketch method with the 1st order cumulant. Last but not the least, none of these works address cube materialization and query processing as considered in our work.

3 AGGREGATION

We present the concept of measure pmf in Section 3.1. We study convolution and sketch based aggregation in Sections 3.2 and 3.3, respectively[2].

3.1 Measure Pmf

Since the existence of an object is uncertain, its measure value is also uncertain. We denote the distribution of o_x's measure value by $f_X(x)$, where X is a random variable for the measure value. If o_x's existential pmf represent the join distribution of both dimension and measure values, $f_X(x)$ is the marginal distribution on the measure domain $dom(M)$, satisfying $\sum f_X(x) = 1$. Also, the length of a measure pmf is no larger than m, if an object has m instances at most.

An object's measure pmf can be obtained from its existential probabilities with simple transformations. For example, in Fig. 2, we can enumerate o_2's instances $\{t_3, t_4, t_5\}$ and merge similar items, i.e., combine instances with equal measure values. Formally, $f_X(x) = \sum_{t_i \in o_x \wedge t_i.x = x} t_i.prob$.

However, measure pmfs are insufficient for deriving a cell's aggregation, because unlike a precise multidimensional point that deterministically resides in a cell, an uncertain object possibly exists in multiple cells. As we will show, the aggregation is conditionally dependent on the topological relationship between cells and objects.

3.2 Convolution-based Aggregation

Definition 1. Each cell is defined as a multidimensional vector, $C = \{I_1, I_2, ..., I_m\}$, where $I_i = [D_i^l, D_i^u]$ is an interval on Dimension D_i, and the I_is are mutually exclusive. For a cuboid, the union of all its cells is equal to the corresponding domain space.

Definition 2. **Complete Object and Pmf.** If an object is entirely contained by a cell, the object is a complete object w.r.t. the cell and its pmf is a complete pmf w.r.t. the cell.

For example, o_1 is a complete object w.r.t. C_1 and both o_1 and o_2 are complete objects w.r.t. C_3. All objects are complete objects w.r.t. the apex cell, i.e., the domain space. The aggregation over complete pmfs is their convolution.

The convolution of complete pmfs w.r.t. a cell satisfies PWS. We show the possible worlds of the above example in Table 4(a). There exist 6 possible worlds W_1 to W_6. We can get the pmf of SUM by scanning the PWD table and merging items with the same values. For example, the probability of $X_1 + X_2$ being 2 is $p(W_4) + p(W_6) = 0.2$. By repeating this, we get the pmf for the aggregation and it is the same as the convolution result.

TABLE 4: Possible World Interpretation

(a) $X_1 + X_2$ for cell $D[1:2, 1]$

ID	PWD	$p(W_i)$	SUM
W_1	t_1, t_3	0.12	3
W_2	t_1, t_4	0.3	4
W_3	t_1, t_5	0.18	3
W_4	t_2, t_3	0.08	2
W_5	t_2, t_4	0.2	4
W_6	t_2, t_5	0.12	2

(b) $X_1 + X_2^{(1)}$ for $D[1:1]$

ID	PWD	$p(W_i)$	SUM
W_1	t_1, **null**	0.18	2
W_2	t_1, t_3	0.12	3
W_3	t_1, t_4	0.3	4
W_4	t_2, **null**	0.12	1
W_5	t_2, t_3	0.08	2
W_6	t_2, t_4	0.2	3

2. For some proofs of lemmas of this section, we refer interesting readers to the conference version [7].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2913420, IEEE Transactions on Knowledge and Data Engineering

Definition 3. **Truncated Object and Pmf.** If object o_x is partially contained by cell C_j, we denote their overlapping part as a *truncated object* $o_x[j]$. For instances of $o_x[j]$, the distribution of their measure values follows a *truncated pmf* $f_X^{(j)}$, which can be obtained in the same way as for a non-truncated object. Formally, $f_X^{(j)}(x) = \sum_{t_i \in o_x[j] \wedge t_i.x=x} t_i.prob$.

In Fig. 2, o_2 overlaps with cells C_1 and C_2. So, o_2's measure pmf is "truncated" into pmfs $f_2^{(1)}$ and $f_2^{(2)}$, respectively. The integration of all truncated pmfs of an object sums up to 1, as shown in Lemma 1.

Lemma 1.
$$\sum_{o[j] \subseteq o} \sum_{x \in o[j].M} f_X^{(j)}(x) = 1 \quad (1)$$

However, convolution cannot be directly applied for truncated pmfs, because truncated pmfs corresponds to "*null tuples*" in possible words. In Table 4(b), we show SUM over cell C_1. There are 6 possible worlds W_1 to W_6, whereas W_1 and W_4 are with bolded "null" instances of o_2, because it is necessary to consider the case that o_2 is not in cell. We denote o_2's disappearance in C_1 as "null", whose probability equals to $1 - \sum_{t_i \in o_2 \wedge t_i \notin C_1} t_i.prob = 0.3$. Its product with t_1's existential probability is $p(W_1) = 0.18$. Reversely, ignoring null tuples leads to wrong calculations of aggregated pmfs[3].

Definition 4. **Complemented pmfs.** For a truncated pmf f_X in cell C, its complemented pmf is found by adding a *null item* $(0, 1 - \sum_{x \in C} f_X(x))$ to f_X.

For example, in Fig. 2, $f_{X_2}^{(1)} = \{(1, 0.2), (2, 0.5)\}$. Then, the complemented pmf $f_{X_2}^{(1)}$ is $\{(0, 0.3), (1, 0.2), (2, 0.5)\}$. For a complete pmf, the complemented pmf is itself.

Definition 5. **Operation $\oplus$.** Suppose object o_i is truncated into two parts with truncated pmfs$f_{X_i}^{(s)}$ and $f_{X_i}^{(t)}$. We have: $f_{X_i}(x) = f_{X_i}^{(s)}(x) \oplus f_{X_i}^{(t)}(x)$.

For brevity, we denote it as $f_{X_i} = f_{X_i}^{(s)} \oplus f_{X_i}^{(t)}$. Operation $\oplus$ is similar to vector addition. In Fig. 2, for example, o_2 has truncated pmfs $f_{X_2}^1(x)$ and $f_{X_2}^2(x)$. It shows that $f_{X_2}^1(x) \oplus f_{X_2}^2(x)$ equals $\{(1, 0.2+0.3), (2, 0.5+0)\}$ which equals f_{X_2}. Here, the null item is $(0, 1 - 0.2 - 0.3 - 0.5 = 0)$ and is omitted. More, if an object is truncated into three parts, $\oplus$ is applied for the sequence of the three, and so on.

Definition 6. **Operation $\otimes$.** Suppose f_X and f_Y are two complemented pmfs of variables X and Y. The distribution of the two variables' summation is their convolution.
$$f_{X+Y}(z) = (f_X \otimes f_Y)(z) = \sum f_X(x) \cdot f_Y(z - x)$$

For brevity, we denote it as $f_{X+Y} {}^{x}{=} f_X \otimes f_Y$. It is easy to verify that $\oplus$ and $\otimes$ satisfy the *distribute* and *communicative laws*. Without causing any ambiguity, we refer to a truncated pmf as its complemented format for the rest of the paper. Next, we show that the convolution-based aggregation satisfies PWS and thus ensures correctness.

Lemma 2. Suppose two truncated pmfs whose complemented formats are f_X and f_Y, respectively. Their convolution $f_{X+Y} = f_X \otimes f_Y$ satisfies PWS.

3. A counter example is that there would be 4 possible worlds $\{(t_1, t_3), (t_1, t_4), (t_2, t_3), (t_2, t_4)\}$ if without null cases. Its corresponding SUM would be $\{(2, 0.08), (3, 0.32), (4, 0.3)\}$. But the actual pmf is $\{(1, 0.12), (2, 0.26), (3, 0.32), (4, 0.3)\}$.

Recall that a complete pmf is also a complemented pmf. Thus, Lemma 2 can be generalized to the case that either or both of the two are complete pmfs. Then, using mathematical induction, it is easy to generalize Lemma 2 from supporting two pmfs to multiple pmfs. So, we can conclude that for the set of pmfs of a cell their convolution satisfies PWS. The time complexity for aggregation is depending on two factors: 1) the number of pmfs; 2) the length of a pmf, i.e., the number of instances in a pmf. Suppose each pmf is of length m. A convolution operation, i.e., $\otimes$, can be evaluated in $O(m^2)$ time. The time efficiency can be improved to $O(mlog_2m)$ if Fast Fourier Transformation (FFT) is used. We prove that the convolution of n pmfs takes the cost of $O(mn^2log_2(mn))$ as proved in [7]. It can be simplified as $O(n^2log_2(n))$, if m is a constant.

Lemma 3. Suppose each pmf is of length m, the time complexity for convoluting n pmfs is $O(mn^2 \cdot log_2(mn))$.

Proof. The most commonly used FFT is the Cooley-Tukey algorithm that depends on the factorization of the pmf length. Therefore, the pmf length is limited to power-of-two sizes [35]. It means if a pmf is of length m, we needs to scale it up to length 2^i, where $i = \lceil log_2 m \rceil$. So, a more accurate convolution cost for two pmfs of length m is $O(2^i \cdot i)$ instead of $O(mlog_2m)$.

Then, given n pmfs, how many of them will be scaled up to 2^{j+1}? The answer is $\frac{2^{j+1}-2^j}{m}$. Let $i = \lceil log_2 m \rceil$ and $i^* = \lceil log_2(m \cdot n) \rceil$. The time cost of convoluting n pmfs is:

$$\underbrace{\alpha \cdot 2^i log_2(2^i) + ... + \alpha \cdot 2^i log_2(2^i)}_{(2^i - m)/m} + \underbrace{\alpha \cdot 2^{i+1} log_2(2^{i+1}) + ... + \alpha \cdot 2^{i+1} log_2(2^{i+1})}_{(2^{i+1} - 2^i)/m}$$
$$+ ... + \underbrace{\alpha \cdot 2^{i^*} log_2(2^{i^*}) + ... + \alpha \cdot 2^{i^*} log_2(2^{i^*})}_{(2^{i^*} - 2^{i^*})/m}$$
$$\approx \sum_{j=i}^{i^*} \alpha \frac{2^j - 2^{j-1}}{m} \cdot 2^j log_2(2^j) = \frac{\alpha}{m} \sum_{j=i}^{i^*} 2^{2j-1} \cdot j = O(\frac{(2^{i^*})^2 \cdot i^*}{m})$$
$$= O(mn^2 \cdot log_2(mn))$$

3.3 Sketch-based Aggregation

Now we consider another way of probability distribution representations to alleviate the aggregation overheads. The cumulants (κ_k) and moments (μ_k) are quantitative measures for defining the shape of a probability distribution. The first and second order cumulants coincide with *mean* and *variance*. The third and fourth order of the central moments are called *skewness* and *kurtosis*. Theoretically, the collection of all cumulants (or moments) of all orders, i.e., $k = 1, ...\infty$, can uniquely determine a probability distribution[4].

Cumulants. Cumulants have a property called the *linear summation property*. The cumulant of a sum of independent random variables is the sum of the corresponding cumulants of the addends [37]. For example, the expectation (1st order cumulant) of $X_1 + X_2$ equals the summation of $E(X_1)$ and $E(X_2)$. Also, the variance (2nd order cumulant) of $X_1 + X_2$ equals the summation of $Var(X_1)$ and $Var(X_2)$[5].

4. It is also possible to approximately recover the probability distribution with first few cumulants or moments [36]. But there is no theoretical guarantees on the approximation quality to our best knowledge.

5. The linear cumulant summation is true according to *Central Limit Theorem*. In most database applications, it can be seen from the context that it is reasonable to assume that random variables are independently and identically distributed [38]. If it is not clear that the distributions are identical, Liapounov's condition can be checked to ensure the convergence of the cumulant sum.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2913420, IEEE Transactions on Knowledge and Data Engineering

Then, we can have that the k-th order cumulant of cell C is the linear summation of all C's pmfs' cumulants. Equivalently, $\kappa_k(F_C) = \sum_{o_i \in C} \kappa_k(o_i)$.

Cumulants can be derived from moments, and vice versa. The k-th order cumulant κ_k is a k-th degree polynomial in the first k non-central *moments*, $\{\mu_i\}_{i \leq k}$.

Moments. The concept of moments is proposed by Chebyshev for the proof of central limit theorem. A general form of k-th order moment is: $\mu_k(X) = E(X^k) = \sum_{x \in X} x^k f_X(x)$.

For aggregating a cell, we first calculate each pmf's moments and therefore its cumulants. Then, we use the linear summation of pmfs' cumulants to get the cumulants of the cell's aggregate for complete pmfs. We proceed to show how to use Lemma 4 to calculate the moments of an object's pmf from its truncated parts.

Lemma 4. The k-th order moment of an object's pmf is the linear summation of k-order moments of its truncated pmfs. Formally, if object o is truncated into $o[1], ..., o[\tau]$ with truncated pmfs $f_X^{(1)}, ..., f_X^{(\tau)}$, we have: $\mu_k(X) = \sum_{i=1}^{\tau} \mu_k(X^{(i)})$.

Complexity. The time complexity of sketch-based aggregation depends on two factors: 1) n, the number of pmfs; 2) k, how many cumulants are used for representing a pmf. According to the linear summation property, the sketch-based aggregation takes $O(k \cdot n)$ time.

4 Parallel Cube Materialization

We introduce the parallel framework in Section 4.1. We devise a semi-distributive strategy which covers full and partial cube materialization in Sections 4.2 and 4.3, respectively. We study how the parallel materialization can be optimized with a cost model in Section 4.4.

4.1 Parallel Framework

Data cube materialization is known as a computationally intensive task. We thus consider leveraging the power of parallel computing for efficient task processing. The work is to match the available computational resources with the computational needs, i.e., the materialization task. On the other hand, it is suitable for the materialization to be parallelized according to the multi-dimensional and multi-level nature of data cubes.

For parallelizing the materialization, the computational task is split into smaller sub-tasks and assigned to computational cores. In our implementation, we make the parallelism implemented on the cell levels. In particular, we start with the base cuboid, and dynamically assign tasks of cell aggregation to available cores in a round-robin manner. Such task assignment continues until all aggregation tasks of the cuboid are done. The process is then repeated for subsequent cuboids until the materialization finishes.

The parallelization plan is made according to three properties: 1) the aggregation tasks of different cells in a cuboid are independent; 2) the aggregation between ancestor and descendant cells has correlations and the computational efforts have potentials to be shared; 3) the number of cells is much larger than the number of computation cores.

The first property, i.e., cell-level independence, enables the feasibility of cell-level parallelism. The second property implies that we cannot naively employ cuboid-level parallelism, i.e., assigning different cuboids to different cores, to maximally share the computation. The third fact makes it feasible to distribute workloads in a round-robin fashion in order to avoid cases of over- and under-provisioning and therefore a finer level of parallelism is not necessary.

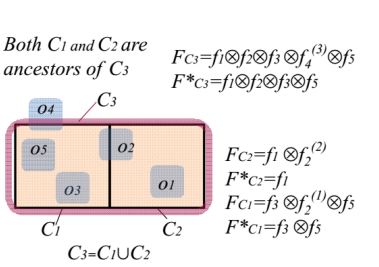


Fig. 3: F_c and F_c^*

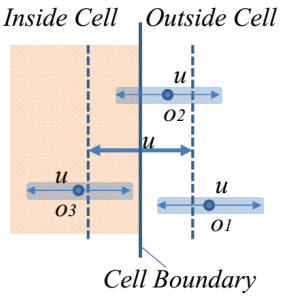


Fig. 4: T-pmf Model

In the sequel, we introduce algorithms of full and partial materialization for the parallel implementation, as well as corresponding cost models.

4.2 Full Materialization

The efficient materialization is achieved by two techniques. The first is the acceleration with parallel computing. The second is by maximizing the sharing of aggregation computation, especially the sharing between ancestor and descendant cuboids. For the first part, we will show detailed algorithms based on the proposed parallel framework. The second part is handled by answering the following questions: 1) which part of the convolution-based aggregation between ancestor and descendant cuboids can be reused and how? 2) given multiple materialized ancestor cuboids, which one derives a descendant cuboid most efficiently?

We store two parts of aggregates for each cell C, the convolution of F_C and F_C^*, and their sketches, $\{\kappa_j(F_C)\}$ and $\{\kappa_j(F_C^*)\}$[6]. F_C is the convolution of all pmfs, including complete and truncated pmfs. The other aggregation F_C^* is for convolution of complete pmfs, which can be reused for aggregating its descendant cells. The aggregation of complete pmfs, i.e., F_C^*, can be reused, because if a pmf is complete for a cell it is complete for all the cell's descendant cells. We call such this property *Pmf Containment Monotonicity*. An example is shown in Fig. 3. Suppose C_3 is a descendant cell of C_1 and C_2. Pmfs f_3 and f_5 are complete for C_1 and f_1 is complete for C_2. According to the monotonicity property, f_1, f_3, and f_5 are complete for descendant cell C_3. The aggregation of complete pmfs can be reused, meaning $F_{C3}^* = F_{C1}^* \otimes F_{C2}^*$. It means the aggregation of complete pmfs of a cell, i.e., F_C^*, is distributive.

Another observation is that truncated pmfs are necessary for the aggregation of a cell but they cannot be reused for descendant cells. For example, F_{C2} equals $F_{C2}^* \otimes f_2^{(2)}$. But $f_2^{(2)}$ cannot be used for getting F_{C3}. So, for truncated pmfs, we store their identifiers in a quarantine zone maintained together with the cuboid. Notice that there is no quarantine zone for the apex-cuboid, since all objects are complete for the domain.

6. The aggregation between ancestor and descendent cells is similar for convolution and sketch-based methods. By viewing the pmf in the sketch representation, we can replace $\otimes$ and $\oplus$ with linear summations over culumants and moments, respectively. For ease of presentation, we focus on the convolution-based aggregation in this section.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2913420, IEEE Transactions on Knowledge and Data Engineering

In summary, the convolution operation is distributive for complete pmfs but not distributive for truncated pmfs isolated in quarantine zones. Thus, we come up with a semi-distributive framework that is covered by Algorithms 1 and 2. It is possible that a pmf is truncated for an ancestor cell but complete for its descendant cell. For example, f_2 is truncated for C_1 but complete for C_3. In this case, we can remove f_2 from cuboid $\mathcal{C}_3$'s quarantine zone, supposing C_3 belongs to cuboid $\mathcal{C}_3$. It is covered by lines 18-20 of Algorithm 2. A cuboid's quarantine zone can be cleaned if all its descendant cuboids are materialized.

Algorithm 1: Base Cuboid Parallel Materialization

Output: A materialized base cuboid
1 **for** *each cell* C_k *in the base cuboid* **in parallel do**
2 $F_{C_k} \leftarrow 0$
3 $F^*_{C_k} \leftarrow 0$ ▷ convolution for complete pmfs in C_k
4 Let all moments and cumulants of F_{C_k} and $F^*_{C_k}$ be 0
5 **for** *each object* $o_i \in C_k$ **do**
6 **if** o_i *is a truncated object w.r.t.* C_k **then**
7 Store o_i in the quarantine zone
8 Let $f^{(k)}_{X_i}$ be o_i complemented pmf w.r.t. C_k
9 $F_{C_k} \leftarrow F_{C_k} \otimes f^{(k)}_{X_i}$ ▷ Convolution
10 Calculate $\{\mu_j[X_i^{(k)}]\}_{\forall j}$ and $\{\kappa_j[X_i^{(k)}]\}_{\forall j}$
11 Store o_i and $\{\mu_j[X_i^{(k)}]\}$ in the quarantine zone
12 $\kappa_j(F_{C_k}) \leftarrow \kappa_j(F_{C_k}) + \kappa_j(X_i^{(k)})$ ▷ $(\forall j)$ ▷ Sketch
13 **else**
14 $F_{C_k} \leftarrow F_{C_k} \otimes f_{X_i}$ ▷ f_{X_i} is a complete pmf
15 $F^*_{C_k} \leftarrow F^*_{C_k} \otimes f_{X_i}$ ▷ Convolution
16 Calculate $\{\kappa_j[X_i]\}_{\forall j}$
17 $\kappa_j(F_{C_k}) \leftarrow \kappa_j(F_{C_k}) + \kappa_j(X_i)$ ▷ $(\forall j)$
18 $\kappa_j(F^*_{C_k}) \leftarrow \kappa_j(F^*_{C_k}) + \kappa_j(X_i)$ ▷ $(\forall j)$ ▷ Sketch

Algorithm 2: non-Base Cuboid Parallel Materialization

Input : an ancestor cuboid $\mathcal{C}^A$
Output: a materialized cuboid $\mathcal{C}$
1 Initialize $\mathcal{C}$'s quarantine zone as $\mathcal{C}^A$'s quarantine zone
2 **for** *each cell* C_k *in the current cuboid* $\mathcal{C}$ **in parallel do**
3 $F_{C_k} \leftarrow 0$
4 $F^*_{C_k} \leftarrow 0$ ▷ convolution for complete pmfs in C_k
5 Let all cumulants of F_{C_k} and $F^*_{C_k}$ be 0
6 **for** *each* C_k*'s ancestor cell* $C^A_k \in \mathcal{C}^A$ **do**
7 $F^*_{C_k} \leftarrow F^*_{C_k} \otimes F^*_{C^A_k}$ ▷ Merge ancestor cells' complete convolution
8 $\kappa_j(F^*_{C_k}) \leftarrow \kappa_j(F^*_{C_k}) + \kappa_j(F^*_{C^A_k})$ ▷ $(\forall j)$
9 $F_{C_k} \leftarrow F^*_{C_k}$
10 **for** *each object* o_i *in cuboid* $\mathcal{C}$*'s quarantine zone* **do**
11 **if** o_i *is a truncated object w.r.t.* C_k **then**
12 Let $f^{(k)}_{X_i}$ be o_i complemented pmf w.r.t. C_k
13 $F_{C_k} \leftarrow F_{C_k} \otimes f^{(j)}_{X_i}$ ▷ Convolution
14 Calculate $\{\sum_k \mu_j[X_i^{(k)}]\}_{\forall j}$ and $\{\sum_k \kappa_j[X_i^{(k)}]\}_{\forall j}$
15 Store o_i and the moments in $\mathcal{C}$'s quarantine zone
16 $\kappa_j(F_{C_k}) \leftarrow \kappa_j(F_{C_k}) + \kappa_j(X_i^{(k)})$ ▷ $(\forall j)$ ▷ Sketch
17 **else**
18 $F^*_{C_k} \leftarrow F^*_{C_k} \otimes f_{X_i}$ ▷ f_{X_i} is a complete pmf
19 $F_{C_k} \leftarrow F_{C_k} \otimes f_{X_i}$
20 Remove o_i from $\mathcal{C}$'s quarantine zone ▷ Convolution
21 Calculate $\{\kappa_j(X_i)\}_{\forall j}$
22 $\kappa_j(F_{C_k}) \leftarrow \kappa_j(F_{C_k}) + \kappa_j(X_i)$ ▷ $(\forall j)$
23 $\kappa_j(F^*_{C_k}) \leftarrow \kappa_j(F^*_{C_k}) + \kappa_j(X_i)$ ▷ $(\forall j)$ ▷ Sketch

If there are multiple ancestor cuboids, we choose the one incurring minimum aggregation cost. This problem is known as *multiway aggregation* [39]. Suppose a cost model $Cost(\mathcal{C}, \mathcal{C}^A)$ that estimates the materialization cost of $\mathcal{C}$ from $\mathcal{C}^A$. The best efficiency is expected to be achieved by choosing the ancestor cuboid $\mathcal{C}^{A*}$ with the smallest cost. Formally, $\mathcal{C}^{A*} = argmin Cost(\mathcal{C}, \mathcal{C}^A)$.

4.3 Partial Materialization

To get the best query performance, the *full materialization* approach precomputes all cuboids. However, full materialization is not space efficient and the construction time is very high. In commercial OLAP products, a commonly adapted method, called *partial materialization*, chooses a subset of cuboids to materialize [6]. The selection of a cuboid depends on: 1) whether the cuboid is beneficial, i.e., whether it helps in efficiently answering queries; 2) whether the cuboid helps in deriving other cuboids that are beneficial. A simple way of measuring such benefits is to use a linear cost model [6]. Here, only a set of selected (not all) cuboids are materialized. Then, during query evaluation, if the requested cuboid has been computed, it is directly used. Otherwise, the cuboid is materialized on-the-fly from one of its materialized ancestor cuboids according to the linear cost model. The base cuboid is assumed to be materialized.

However, the existing solution cannot be directly applied for the probabilistic data cube for two reasons. First, such a model assumes that the cost is solely dependent on the number of tuples scanned for constructing a cuboid and thus is linear to I/Os. The cost in our case depends on many other factors instead of just I/Os. Second, the on-the-fly computation of a cuboid is expected to be efficient to meet the demands of OLAP. Actually, this cuboid selection problem refers to the space constrained query optimization problem based on the cuboid dependencies [6] that enables partial materialization. In the sequel, we study a cost model for probabilistic cubes.

4.4 Cost Estimation

Now we study a unified cost model for both full and partial materialization. As have been illustrated, the cost estimation can be used in two ways, *multiway aggregation* for full materialization and *cuboid selection* for partial materialization. In either case, we need to estimate the cost for building cuboid $\mathcal{C}$ from its ancestor $\mathcal{C}^A$ in order to optimize the materialization. In particular, we have to consider the cost on both I/Os and parallelized computation, denoted as $Cost_{IO}$ and $Cost_{comp}$, respectively[7]. The cost model is formalized below.

$$Cost(\mathcal{C}, \mathcal{C}^A) = Cost_{IO}(\mathcal{C}, \mathcal{C}^A) + \frac{Cost_{comp}(\mathcal{C}, \mathcal{C}^A)}{d_{parallelism}}$$

Here, $d_{parallelism}$ represents the degree of parallelism, i.e., the number of computation cores. Both parts of the cost are measured in elapsed time so that the heterogeneous costs can be unified in the equation. We argue that the computation part cannot be ignored. As will be shown, the I/O part is almost proportional to the data size, while the growth of the computation part is much faster. Extensive experiments are also conducted to measure the estimation accuracy, as well as the necessity of considering computational cost.

Estimation of $Cost_{IO}$. Suppose cuboid $\mathcal{C}^A$ has n^A tuples and its quarantine zone has q^A tuples, and cuboid

7. In the cost model, we did not consider I/O parallelization, which depends on the de facto computer configuration and data layout. However, with specific configuration, our cost model can be easily extended to support the I/O parallelization scenario, e.g., as it is done for the computation parallelization.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2913420, IEEE Transactions on Knowledge and Data Engineering

$\mathcal{C}$ is of size n with a quarantine zone sized q. We assume the main memory is large enough for accommodating a cuboid. The I/O cost of generating $\mathcal{C}$ from $\mathcal{C}^A$ can be modeled as:

$$Cost_{IO}(\mathcal{C}, \mathcal{C}^A) = \alpha_i \cdot \mathcal{PA}(n^A + q^A) + \alpha_o \cdot \mathcal{PA}(n + q)$$

where α_i and α_o denote read and write time of a disk page, and function $\mathcal{PA}(n)$ denotes the number of disk pages taken for n tuples.

Given the categories or domain space defined for each of cuboid $\mathcal{C}$'s dimensions, the number of cells in $\mathcal{C}$ can be obtained. In real applications, some cells can be empty due to skewed data distributions. A common way to derive n and n^A is to use the *FM-sketches* [40], which estimates the number of distinct items, i.e., how many cells a cuboid contains, with limited space and one single pass through the database.

Estimation of Quarantine Zone Size q. The size of a quarantine zone is estimated by Equation 2. Within each cell e, we assume the objects associated are uniformly distributed. The volume of cell e can be represented by $V(e) = \prod_{1 \le i \le d} e_i$, where e_i is the side length on dimension i. We suppose objects are of the same size $V(o) = \prod_{1 \le i \le d} u_i$, where u_i is the side length of the uncertainty region on dimension i. The density ρ_e is the number of objects per unit volume of the cell. It can be replaced by a global density ρ with some math transformations as shown in Equation 2.

$$\begin{aligned} q &= N - \sum_{e \in \mathcal{C}} \rho_e \cdot \max\{0, \prod_{1 \le i \le |D|} (e_i - u_i)\} \\ &= N - \rho \cdot n \cdot \max\{0, \prod_{1 \le i \le |D|} (e_i - u_i)\} \end{aligned} \quad (2)$$

The idea of the equation is to: 1) count the number of complete pmfs inside a cell; 2) its difference with the total number of pmfs (N) is the number of truncated pmfs; 3) return the difference as the size of the quarantine zone.

Estimation of # of complete pmfs in cell e. We use an example of a one-dimensional cell to explain the first step, as shown in Fig. 4. There exist two dashed lines enclosing an "infecting zone" such that if an object is truncated w.r.t. the cell, its center must be within the infecting zone. In the example, o_1 and o_3, whose centers are out of the zone, are not truncated objects. If offsetting the cell boundary inside by $u/2$, we get a region such that if a object's center is in the region, the object must be completely contained by the cell. So, the number of complete pmfs can be estimated by the product of the local density times the cell's volume, i.e., $\rho_e \prod_{1 \le i \le |D|} (e_i - u_i)$. There is no complete pmf in e, if $e_i < u_i$.

The parameters N, $\{u_i\}$, and $\{e_i\}$ can be obtained while using the FM-sketch method to scan the database. ρ can be calculated by dividing N with the domain volume.

Estimation of $Cost_{comp}$. The computational cost of deriving cuboid $\mathcal{C}$ is the number of convolutions taken from its materialized ancestor $\mathcal{C}^A$. The convolution computation has three parts, including the aggregation for complete pmfs, truncated pmfs, and for the pmfs which are truncated in ancestor cells but complete in descendant cells. The three parts are connected by a weighted summation function, since the coefficients, i.e., α_c, α_c, and $\alpha_{t \to c}$, are different at different data scale.

$$\begin{aligned} Cost_{comp} &= \alpha_c \cdot Cost^c_{pmf} + \alpha_t \cdot Cost^t_{pmf} + \alpha_{t \to c} \cdot Cost^{t \to c}_{pmf} \\ &= \alpha_c \cdot log_2 N^c_{pmf} (N^c_{pmf})^2 + \alpha_t \cdot log_2 N^t_{pmf} (N^t_{pmf})^2 + \\ &\quad \alpha_{t \to c} \cdot log_2 N^{t \to c}_{pmf} (N^{t \to c}_{pmf})^2 \end{aligned} \quad (3)$$

N^c_{pmf} is the number of convolutions for aggregating complete pmfs. There are n cells in $\mathcal{C}$ and each cell $C \in \mathcal{C}$ stores an aggregated pmf, F^*_C, for its complete pmfs. Thus, there are n such aggregated pmfs in $\mathcal{C}$ and n^A ones in $\mathcal{C}^A$. Note that n is smaller than n^A, because aggregates in $\mathcal{C}^A$ are combined and thus reused while deriving $\mathcal{C}$. In total, this process takes $(n^A - n)$ convolution operations, i.e., $N^c_{pmf} = n^A - n$. N^t_{pmf} is the number of convolutions for aggregating truncated pmfs. The truncated pmfs' aggregation cannot be reused, so the aggregation is redone for $\mathcal{C}$. Thus, N^t_{pmf} equals to the total number of truncated pmfs for all cells in $\mathcal{C}$. The idea of estimating the number of truncated pmfs per cell is similar to that of Equation 2. $N^{t \to c}_{pmf}$ is the number of pmfs which are truncated for cuboid $\mathcal{C}^A$ but complete for $\mathcal{C}$. It can thus be estimated by the total number of truncated pmfs of $\mathcal{C}^A$ minus that of $\mathcal{C}$.

5 Probabilistic Queries over Cuboids

Our data cube techniques support classic OLAP queries, such as roll-up, drill-down, slicing, and dicing. The drill-down or roll-up path conforms with the cuboid lattice. The implementation of roll-up and drill-down has been implicated by the cuboid materialization. For example, drilling down a cuboid refers to navigating into a more detailed level. It can be implemented by looking up the ancestor cuboid with the user-specified expanding dimension.

In this section, we study how to evaluate two representative queries, namely the *probabilistic slicing query* and the *probabilistic dicing query*, on the probabilistic data cube. We present the query semantics and corresponding evaluation frameworks in Section 5.1. In particular, we cover the sketch-based pruning techniques for probabilistic dicing queries in Section 5.2. For ease of presentation, we assume the data cube is fully materialized. The solutions can be easily extended to the partial materialization case with the techniques presented in Section 4.3.

5.1 Queries

Definition 7. A probabilistic slicing query (PSQ *in short*) performs aggregation over a set of dimensions (D_1 to D_k) and retrieves tuples whose corresponding dimension values are within given range, e.g., $[l_1, u_1]$, ..., $[l_k, u_k]$.

The purpose of PSQ is to offer confidence indicators to traditional slicing queries. A PSQ returns a set of qualified cells, i.e., within the query range, together with the aggregates. The evaluation of PSQ has two phases. The first phase, *cuboid selection*, locates the cuboid that matches the dimensions specified in the query. The second phase, *tuple selection*, elaborates each tuple of the cuboid to see if they are within the query range.

Definition 8. A probabilistic dicing query (PDQ *in short*) performs aggregation over one or more dimensions and return the tuples with measure values higher than the value threshold τ_v and qualification probabilities higher than the probability threshold τ_p.

With a pre-aggregated cuboid on dimensions D_1 to D_k, the query can be evaluated by scanning each cell and check if its corresponding qualification probability is above the given threshold. Suppose $x \sim F_c(x)$ is a random variable representing the aggregated value of cell C. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2913420, IEEE Transactions on Knowledge and Data Engineering

qualification probability can be calculated by integrating the aggregation pmf F_c as $Prob(X \geq \tau_v) = \sum_{x \geq \tau_v} F_c(x)$.

Then, if a tuple's qualification probability is at least τ_p, it is accepted as a part of the query answer. Otherwise, it is rejected. In summary, the evaluation of PDQ contains 3 phases. The first *cuboid selection* phase is similar to that of PSQ. The purpose is to retrieve the appropriate cuboid. The second *pruning* phase utilizes probabilistic pruning bounds (in Section 5.2) to quickly qualify or disqualify candidate tuples/cells in the cuboid. The third *refinement* phase is to compute qualification probabilities for suspected cells, i.e., those are neither pruned nor qualified in the second phase, and compare them with τ_p. For the partial materialization case, it is only necessary to materialize the requested cells instead of the entire cuboid.

5.2 Sketch-based Pruning

Sketch values, i.e., cumulants and moments, can be used to derive upper and lower bounds for cells' qualification probabilities. If a tuple's lower bound is above τ_p, it is included in the query answer. Or if its upper bound is below τ_p, it is excluded. The bounds help in reducing the query effort in the refinement phase.The upper and lower bounds are formalized by Lemmas 5 and 6, respectively.

Lemma 5. (**Upper Bound.**) If the expectation of the aggregated value X is at most τ_v, the probability that $X \geq \tau_v$ is at most $\frac{\kappa_2}{\kappa_2+a^2}$, where $a = |\tau_v - \mu_1(X)|$. Formally,

$$\mu_1(X) \leq \tau_v \Rightarrow Prob(X \geq \tau_v) \leq \frac{\kappa_2}{\kappa_2 + a^2}$$

Lemma 6. (**Lower Bound.**) If the expectation of the aggregated value X is at least τ_v, the probability that $X \geq \tau_v$ is at least $\frac{a^2}{\kappa_2+a^2}$, where $a = |\tau_v - \mu_1(X)|$. Formally,

$$\mu_1(X) \geq \tau_v \Rightarrow Prob(X \geq \tau_v) \geq \frac{a^2}{\kappa_2 + a^2}$$

Notice that we do not derive both bounds for a tuple's qualification probability. Whether to derive the upper or lower bound is case-dependent. If a tuple's expectation is greater than τ_v, we upper bound its probability by Lemma 5. Otherwise, we lower bound it by Lemma 6. Since the bounds can be directly derived from the one-sided Chebyshev's inequality (or Chebyshev-Cantelli inequality), their proofs are omitted due to page limits[8].

6 Extensions to New Aggregates

In this section, we investigate several extensions regarding the aggregation function supported by probabilistic data cubes. In particular, we consider the aggregation functions `COUNT`, `AVG`, and `MAX`. `MIN` can be derived in a similar way as `MAX` and is omitted due to page limits.

6.1 COUNT

`COUNT` can be considered as a special case of `SUM`. Essentially, it is to measure multi-dimensional objects by their existence. In other words, existence becomes the measure. For example, if an object is completely inside a cell, the existence is 100%. If an object partially belongs to a cell, e.g. 80%, it means the existence can be represented by a complemented pmf $\{(0, 20\%), (1, 80\%)\}$. Here, the pmf of an object w.r.t. a cell is defined over set $\{0, 1\}$ where 0 refers to non-existence and 1 refers to existence of the the cell. The aggregated `COUNT` of a cell is still a pmf, representing the possible counts together with their possibilities. This count can also be calculated through convolution. Thus, the whole materialization framework of `SUM` can be applied.

8. It's possible to use Zelen's inequality for the first four cumulants resulting in a tighter pruning bound. We stick to Chebyshev's inequality that is efficient yet simple for deployment. Although the tightest bounds can be derived by higher order cumulants, it might cost more than calculating the exact probability integration according to our experiments, which contradicts our intention.

TABLE 5: PWS Table (`AVG`) for cell $D[1:1]$

$F_C^{avg} = \{(1, 0.2), (1.5, 0.32), (2, 0.48)\}$

ID	PWD	$p(W_i)$	SUM	COUNT	AVG
W_1	t_1, **null**	0.18	2	1	2
W_2	t_1, t_3	0.12	3	2	1.5
W_3	t_1, t_4	0.3	4	2	2
W_4	t_2, **null**	0.12	1	1	1
W_5	t_2, t_3	0.08	2	2	1
W_6	t_2, t_4	0.2	3	2	1.5

TABLE 6: An Example of $F_C^{sum,count}$

Probability: SUM / COUNT	1	2	3	4
1	0.12	0.18	0	0
2	0	0.08	0.32	0.3

6.2 AVG

In traditional data cubes, `AVG` is an algebraic operation, meaning it can be evaluated by algebraic function on other aggregates. For example, in case of precise values, `AVG` can be calculated by `SUM / COUNT`. However, it is challenging to calculate such aggregates in the setting of probabilistic data cubes, because 1) data values are represented by pmfs; 2) `SUM` and `COUNT` are often not independent.

Overview. We first show how the probabilistic distribution of `AVG` can be calculated. Then, we demonstrate that such calculation is computationally expensive and the simple approximation would incur considerable errors. To tackle that, we study the properties of truncated pmfs which would scale down the problem. We also show that the computation can further be accelerated with parallel processing. At last, we investigate the sketch method for `AVG` in order to speed up the pruning efficiency for dicing queries.

6.2.1 Aggregation of $\mathbf{F_C^{avg}}$

We denote the probabilistic distribution of `AVG` for cell C as F_C^{avg}. First, we use PWS semantic to illustrate the derivation of F_C^{avg} for cell $D[1:1]$, as shown in Table 5. For every possible world of the cell, $p(W_i)$, we can get the `SUM` and `COUNT` and therefore the value of `AVG`. Then, the possible worlds with same `AVG` values are combined and the corresponding probabilities are summed so that $F_C^{AVG} = \{(1, 0.2), (1.5, 0.32), (2, 0.48)\}$. Similar to `SUM`, such a process is #-P hard and incurs extensive computational overheads.

The `AVG` can be efficiently obtained, if the joint distribution of `SUM` and `COUNT` is available. With such a joint distribution, the average aggregate can be evaluated by scanning the joint distribution and by combining the items with the same ratio of the value of `SUM` to `COUNT`. We can use quotient convolution to represent such a process as follows.

$$F_C^{avg}(z) = \sum_y F_C^{sum,count}(zy, y)$$

For example, in Table6, the average could be 1 when both `SUM` and `COUNT` are equal to 1, or when both `SUM` and `COUNT` are equal to 2. So, the probability for `AVG` $= 1$ is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2913420, IEEE Transactions on Knowledge and Data Engineering

$0.12+0.08 = 0.2$, which is consistent with the result derived with the PWS semantic.

Lemma 7. Given a cell C and the joint distribution of `SUM` and `COUNT`, the distribution of `AVG` can be derived by the quotient convolution and it satisfies PWS.

The proof is similar to Lemma 2 and is omitted due to page limits. It guarantees the correctness of the deriving joint distribution with quotient convolution. However, the joint distribution for pmfs of a cell are computationally expensive. For example, if a cell has 3 objects, o_1, o_2, and o_3, there can be $\binom{3}{2}$ cases for calculating $F^{sum,count}(*, 2)$. They are $\{o_1, o_2\}$, $\{o_2, o_3\}$, and $\{o_1, o_3\}$. It means that one has to check all the combination object sets with size equal to the given count. In general, if there are n objects/pmfs belonging to a cell, there can be $\binom{n}{k}$ cases for calculating $F^{sum,count}(*, k)$. In total, to get the joint distribution $F^{sum,count}()$ requires $\sum_{k=1}^{n} k \cdot \binom{n}{k} = n2^{n-1}$ convolutions. The time complexity is $O(n \cdot 2^n)$. So, an efficient way of calculating the joint distribution is needed.

Notice that the count and sum are independent for complete pmfs. When adding a complete object, the cell's object count will definitely increase by one, since a complete object definitely belongs to a cell. It means the effect of the sum-count correlation can be ignored for complete pmfs. So, if a cell is with n_t truncated pmfs, the complexity can be reduced to $O(n_t \cdot 2^{n_t})$. However, the complexity is still high, especially for objects with more uncertainties, i.e., less complete pmfs. A polynomial-time solution is thus needed for the efficient distribution evaluation.

To do that, one may mistakenly derive the aggregation by avoiding the computation of correlations between `SUM` and `COUNT`. For example, suppose pmfs of `SUM` and `COUNT` of cell C are represented by F_C^{sum} and F_C^{count}, which can be derived independently.

$$F_C^{avg}(z) \approx \sum_y F_C^{sum}(zy) \cdot F_C^{count}(y)$$

According to Table 5, $F_C^{avg}(1.5) = 0.32$. It can be observed as follows, which shows that the error caused by this incorrect assumption is over 50%:

$$F_C^{avg}(1.5) = 0.32 \neq F_C^{sum}(3) \cdot F_C^{count}(2) = 0.32 * 0.7 = 0.224$$

Thus, unlike other aggregation results, the correlation between the two, `COUNT` and `SUM`, has to be calculated. Next, we show how to tackle these challenges. The joint distribution can also be evaluated in a semi-distributive manner. Then, for complete pmfs, their `SUM` and `COUNT` are independent, because `COUNT` is now a certain value.

6.2.2 Evaluation of Joint Distribution

We tackle the problem by evaluating the joint distribution incrementally. Given a set of objects of a cell, we insert objects and update the states of the joint distribution iteratively. The joint distribution can be viewed as a two-dimensional table, as shown in Table 6. Therefore, we can denote the k-th row of the joint distribution table as $F_C^{sum,count}[k]$. Upon inserting a new object, we can elaborate the joint distribution table and update the corresponding information by rows.

Let o_i be the object inserted at iteration i. The effect of inserting o_i has to be checked. If o_i is a truncated pmf, there can be two possible cases, i.e., the existence and non-existence of o_i to the cell, as summarized by Equation 4.

In the existence case, the effect of non-null items of o_i is addressed. The non-null items of o_i represent the fact that o_i exists in the cell and its corresponding probabilities. Let $pmf[i]$ be the probability distribution for non-null items of o_i. The renewed joint probability can be done by making each row of $F_C^{sum,count}$ convoluted with $pmf[i]$.

In the non-existence case, the effect of null item of o_i is addressed. The null item of o_i denotes the fact that the object does not exist in the cell and its probability. The count will not be affected by the null item. The joint probability of `SUM` and `COUNT` will be recalculated, since it conditionally depends on the existence of o_i. Let $!pmf[i]$ be probability of the null item of o_i. The joint probability can be renewed by making the product of $!pmf[i]$ with each row of $F_C^{sum,count}$.

$$F_C^{sum,count}[k] = F_C^{sum,count}[k-1] \cdot pmf[i] + F_C^{sum,count}[k] \cdot !pmf[i] \quad (4)$$

In particular, if o_i is a complete pmf, Equation 4 degenerates into the convolution of $F_C^{sum,count}[k-1]$ and o_i's pmf, i.e., f_{X_i}. Equation 4 covers truncated and complete pmfs.

Analysis. The process of the joint distribution evaluation is depicted by Algorithm 3. Line 3 utilizes Equation 4 for the recalculation of the joint distribution with object o_i. Algorithm 3 has a nested-loop in which the convolution can be run n^2 times, if the cell has n objects $|C| = n$. Hence, the time complexity is $O(n^3 log(n))$ if FFT is used.

Algorithm 3: Joint Distribution Evaluation

```
Output: The joint distribution of cell C, F_C^{sum,count}
1 for each object o_i of the cell C do
2     for k = 1 to |C| do
3         ▷ |C| is the total number of objects of cell C Update
          F_C^{sum,count}[k] by Equation 4;
4 return F_C^{sum,count};
```

Example. We use a brief example (Table 7) to verify the correctness. First, o_1 is inserted. At this stage, there is only one row for the joint distribution table, which is the same as o_1's pmf. Next, o_2 is inserted. The table is expanded to two rows. The first row, denoting the probability distribution of `COUNT` $= 1$, is updated by multiplying each item with the null item of o_2, i.e., $\{(1, 0.4 * 0.3), (2, 0.6 * 0.3)\}$. The second row, denoting the probability distribution of `COUNT` $= 2$, is obtained by the convolution of current first row $\{(1, 0.4), (2, 0.3)\}$ and null items of o_2, i.e., $\{(1, 0.2), (2, 0.5)\}$. The result is $\{(2, 0.08), (3, 0.32), (4, 0.3)\}$ and is consistent with the result derived by PWS semantics, i.e., Table 6. The process is repeated until all objects, o_1 to o_3, are inserted, as shown in Table 7.

TABLE 7: An Example of Evaluating $F_C^{sum,count}$

Probability SUM / COUNT	1	2	3	4	5
After Inserting $o_1 = \{(1, 0.4), (2, 0.6)\}$					
1	0.4	0.6			
After Inserting $o_2 = \{(0, 0.3), (1, 0.2), (2, 0.5)\}$					
1	0.12	0.18			
2		0.08	0.32	0.3	
After Inserting $o_3 = \{(0, 0.3), (1, 0.7)\}$					
1	0.036	0.054			
2		0.108	0.222	0.09	
3			0.056	0.224	0.21

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2913420, IEEE Transactions on Knowledge and Data Engineering

Materialization. The materialization for AVG follows the semi-distributive framework introduced in Section 4. Recall that there is no correlation between SUM and COUNT for complete pmfs, as we have shown in this section. The aggregation, or equivalently the evaluation of the joint distribution, of a cell can thus be done by combining two parts. The first part is the integration of all truncated pmfs with Equation 4, denoted by $F_C^{(sum,count)t}$. The second part is the convolution of all complete pmfs, denoted by $F_C^{(sum,count)*}$. The convolution result, or the joint distribution for complete pmfs, is one-dimensional, since the count of complete objects is a fixed value. To merge the two, we replace each row of $F_C^{(sum,count)t}$ by its convolution with $F_C^{(sum,count)*}$. The updated distribution is the sum-count joint distribution to be derived.

According to the Pmf Containment Monotonicity, a complete pmf for a cell is also complete for all its descendant cells. The second part can thus be reused between the ancestor and descendant cuboids. So, we can use the quarantine zone for storing the truncated pmfs of a cuboid in order to maximally share the computation efforts on complete pmfs and apply the parallel materialization framework for the distribution evaluation.

Next, we introduce an efficient method for sketching out the joint distribution and hence speeding up the query process.

6.2.3 Sketch of AVG

Let X denote SUM and Y denote COUNT. To get the sketch of AVG is equivalently to getting the sketch of $\frac{X}{Y}$. To derive the sketch of $\texttt{AVG} = \frac{\texttt{X}}{\texttt{Y}}$ requires calculating the first two orders of cumulants and moments, which is equivalent to evaluate the expectation and variance, i.e., $E(\frac{X}{Y})$, and $Var(\frac{X}{Y})$.

$$E(\frac{X}{Y}) = \sum_{(x,y)\in(X,Y)} \frac{x}{y} f(x,y) = \sum_{(x,y)\in(X,Y)} \frac{x}{y} f_X(x|Y=y) f_Y(y)$$
$$Var(\frac{X}{Y}) = E[(\frac{X}{Y})^2] - E^2[\frac{X}{Y}]$$

From the equation group, to get the expectation and the variance requires checking combinations of SUM and COUNT, which is expensive. More, the process has to be repeated for deriving $\texttt{E}[(\frac{\texttt{X}}{\texttt{Y}})^2]$. A more efficient calculation is thus needed.

We can expand $E(\frac{X}{Y})$ around (μ_x, μ_y) by a 2nd-order Taylor series, following the expanding process of $E(XY)$ shown in [37]. The result is shown by Equation 5, where R is the remainder and can be omitted for approximation purpose[9]. Terms $f'_X()$ and $f''_{XX}()$ are for the first and second order derivatives, respectively.

$$\begin{aligned} E(\frac{X}{Y}) &= E\{\frac{\mu_x}{\mu_y} + f'_X(\frac{\mu_x}{\mu_y})(x-\mu_x) + f'_Y(\frac{\mu_x}{\mu_y})(y-\mu_y) \\ &\quad + \frac{1}{2}[f''_{XX}(\frac{\mu_x}{\mu_y})(x-\mu_x)^2 + 2f''_{XY}(x-\mu_x)(y-\mu_y) \\ &\quad + f''_{YY}(\frac{\mu_y}{\mu_y})(y-\mu_y)^2] + R\} \quad (5) \\ &\approx \frac{\mu_x}{\mu_y} + \frac{1}{2}[f''_{XX}(\frac{\mu_x}{\mu_y})Var(X) + 2f''_{XY}(\frac{\mu_x}{\mu_y})Cov(X,Y) + f''_{YY}(\frac{\mu_x}{\mu_y})] \\ &= \frac{E(X)}{E(Y)} - \frac{Cov(X,Y)}{E^2(Y)} + \frac{Var(Y)E(X)}{E^3(Y)} \end{aligned}$$

With Equation 5, we can get the variance of AVG. After some transformation, we can get the following.

$$\begin{aligned} Var(\frac{X}{Y}) &= E(\frac{X^2}{Y^2}) - E^2(\frac{X}{Y}) \\ &\approx \frac{E(X^2)}{E(Y^2)} - \frac{Cov(X^2,Y^2)}{E^2(Y^2)} + \frac{Var(Y^2)E(X^2)}{E^3(Y^2)} - E^2(\frac{X}{Y}) \\ &= \frac{E^2(X)}{E^2(Y)} \cdot [\frac{Var(X)}{E^2(X)} - 2 \cdot \frac{Cov(X,Y)}{E(X)\cdot E(Y)} + \frac{Var(Y)}{E^2(Y)}] \end{aligned}$$

From the equations of $E(\frac{X}{Y})$ and $VAR(\frac{X}{Y})$, we can see most items can obtained from the expectations and variance of SUM and COUNT, except their covariance $\texttt{COV(X, Y)}$. It means that the computation for sketching SUM and COUNT can be reused and the calculation is thus transferred to deriving the covariance.

According to the definition of covariance, $\texttt{COV(X, Y)}$ measures the the joint variability of SUM and COUNT:

$$Cov(X,Y) = E(XY) - E(X)\cdot E(Y) \quad (6)$$

The expectations of SUM and COUNT are known after the corresponding sketching process is done. Only the expectation of the production of SUM and COUNT needs to be evaluated. Such an expectation can be efficiently derived from the joint distribution $F_C^{sum,count}$.

6.3 MAX

Overview. The aggregation of MAX is a bit different from the other operators. We show that the aggregation function MAX is not distributive in a probabilistic setting. Then, we apply the idea of quarantine zone which helps to make the operation semi-distributive.

6.3.1 Aggregation

The pmf of cell C can be derived by calculating each possible value that can be the maximum of the cell. Let $F_C^{max}(x)$ denote the probability that value x is higher than all other values for objects in the cell C. The probability equals the sum of probabilities that an object has value x and all other objects have valuesj less than x. Formally,

$$F_C^{max}(x) = \sum_{o_i\in C} \int_{-\infty}^{+\infty} f_{X_i}(x) \cdot \prod_{o_j\in C \wedge o_i \neq o_j} Pr(X_j < X_i) \quad (7)$$

Considering the computational cost, to calculate such a probability distribution, one needs to: 1) go through all possible values of objects belonging to C; 2) for each value, calculate the product of probabilities that all other objects are below such a value. We can reduce the computational overheads by: 1) checking only a smaller subset instead of all possible values of those objects; 2) avoiding unnecessary calculation for the probability product by pruning some irrelevant objects.

The optimization can be done with a *cutting object*. The cutting object o_* of a cell is the object which has the maximum minimum value. Formally, $o_* = argmax_{o_i\in C}(X_i.l)$. There can be multiple cutting objects, if multiple minimum values of objects are equal. But the cutting value of a cell is unique, denoted as l^*. Taking Table 1 as an example, o_1's minimum value in C_1 is 1 and o_2's minimum value in C_1 is 1. Both o_1 and o_2 can be cutting objects, with cutting value 1 for cell C_1.

Within a cell, if another object's maximum value is smaller than o_*'s minimum value, then the object has no chance to contribute to the maximum value of this cell. It implies

9. The approximated sketch is used for rendering the pruning bound in the form of probabilistic inequalities, which is found to be accurate in terms of query evaluation.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2913420, IEEE Transactions on Knowledge and Data Engineering

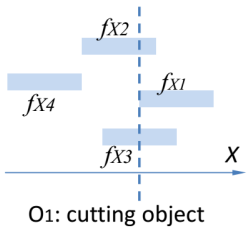


Fig. 5: Calculating F_C^{max} for Cell C

that there is no need to consider objects whose instances' values are below l^* for calculating the distribution, F_C^{max}.

For example, in Figure 5, o_1 will be chosen as the cutting object, since it is the object with the maximum minimum value within Cell C. o_4 can be pruned, because its maximum value is lower than the cutting bound. This way, only unpruned objects, i.e., o_1, o_2, and o_3, are substituted into Equation 7 for deriving F_C^{max}.

This way, we can thus shrink the integration range from $[-\infty, +\infty]$ to $[l*, +\infty]$. The upper bound can further be shortened from $+\infty$ to u^*, if u^* denotes the maximum value of all objects in the cell. Also, we can get a smaller object set for the integration of Equation 7. The candidate object set, denoted as S_C, includes the objects whose maximum values are no smaller than the cutting value, l^*. Equation 7 can thus be rewritten as follows.

$$F_C^{max} = \sum_{o_i \in S_C} \int_{l^*}^{u^*} f_{X_i}(x) \cdot \prod_{o_j \in C \wedge o_* \neq o_j} Pr(X_j < X_*)\ dx \quad (8)$$

From the equation of F_C^{max}, it can be inferred that the computation part is not distributive, due to the integration part and the probability production part. More, the outer summation is based on the candidate object set which is cell-dependant. We observe that the I/O cost dominates the cost of the materialization and therefore the reduction of object traversing overheads would significantly accelerate the overall materialization. In the sequel, we study how the I/O part can be handled in a semi-distributive manner.

6.3.2 Materialization

We have shown the process for calculating the `MAX` aggregation on the level of cells. Now, we study how such aggregation can be deployed for cuboid levels, i.e., how the computation of a cell affects the aggregation of its descendant cells during the materialization.

Suppose a cell C consisting of a set of ancestor cells, i.e., $C = \cup C^A$. The distribution F_C^{max} can be obtained by checking the objects in cell C, pruning irrelevant ones, and applying Equation 7. Equivalently, all objects in ancestor cells $\{o \in \cup C^A\}$ should be checked. We find that the pruning effect, i.e., the candidate object set, can be inherited for descendant cells. Let candidate object sets S_{C^A} and S_C be for cells C^A and C, respectively. We can prove that SC must be a subset of $\cup S_{C^A}$. The correctness is guaranteed by Lemma 8.

Lemma 8. If C consists of a set of ancestor cells ($C = \cup C^A$), then S_C must be a subset of $\cup S_{C^A}$ ($S_C \subseteq \cup S_{C^A}$).

Proof. Let $l^*(C)$ and $l^*(C^A)$ be the cutting value of cells C and C^A, respectively. The candidate set can thus be represented by the objects with maximum values no less than the cutting value, i.e., $S_C = \{o | o.u \geq l^*(C) \wedge o \in C\}$ and $S_{C^A} = \{o | o.u \geq l^*(C) \wedge o \in C^A\}$. Also, we have: 1) $l^*(C) \geq l^*(C^A)$; 2) $C = \cup C^A$. To sum up, S_{C^A} must be a subset of S_{C^A}.

We can thus store the candidate set in the quarantine zone and save the pruning power for descendant cuboids. In particular, during the materialization, the union of candidate sets for a cuboid is stored in the quarantine zone, and is reused while materializing descendant cuboids. This way, the cost for object traversing for cuboids of higher levels is significantly reduced.

7 Experimental Evaluation

7.1 Settings

Datasets[10]**.** We use both synthetic data (an adapted version of the wellknown TPC-H benchmark[11]) and real data (US Climate 2014[12]). Each object in the raw data can be viewed as a multi-dimensional point. Then, for each object, we model its uncertainty of by creating a set of 10, **20**, 30, and 40 possible instances. Of the same object, the instances reside within a multi-dimensional orthogonal region. The region is centered at the multi-dimensional point representing the original object. For each dimension, the region's side length is set to 0.5%, **1%**, and 2% of its domain. Within such a region, the instances follow a multi-dimensional Gaussian distribution whose mean is the region center and the variance of each dimension is the square of 1/6 of the corresponding side length. Then, we get two probabilistic datasets called *TPC* and *Climate*. The statistics of the two datasets are shown as below.

- **TPC.** We identify attributes Time, Item, Supplier, Customer as dimension attributes, and Price as the measure.
- **Climate.** We identify attributes Time, Wind Direction, Wind Speed, and Temperature as dimension attributes, and Rainfall Amount as the measure.

By default, both datasets store 100MB data for $20,141$ objects and 0.4M tuples. We make the two datasets with similar sizes in order to purely examine the effects of different data distributions. For scalability testing, we evaluate the performance with up to 1GB data. The hierarchies of TPC are of two levels on average. For Climate, we construct a two-level hierarchy for each dimension. Then, there are 108 cuboids in the cuboid lattice for both datasets.

Parameters. For partial materialization, we examine the tradeoff between the space cost and performance for cube materialization and query evaluation by varying the space budgets. We use the term *space budget* to represent the ratio of the space cost of the partial materialization to the space cost of the full materialization. By default, it equals 30%.

Implementation. The convolution-based method is implemented using the wavelet package[13]. The sketch-based method uses the first two orders of cumulants and moments by default. All our programs were implemented in Java and run on a PC with a 24 cores / 48 threads processor @ 2.4GHz per core and 32GB RAM. We use 4 core to run all the experiments by default.

10. Default parameters are given in bold.
11. http://www.tpc.org/tpch/
12. http://www.ncdc.noaa.gov/cdo-web/datasets
13. http://www.nayuki.io/page/free-small-fft-in-multiple-languages

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2913420, IEEE Transactions on Knowledge and Data Engineering

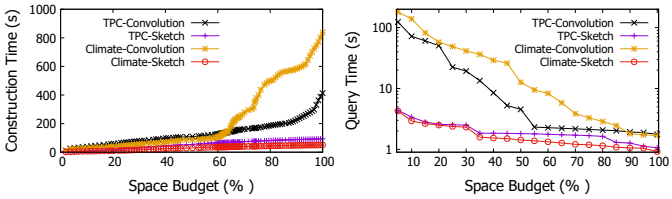
7.2 Results

Aggregation. Results for the aggregation is shown in Fig. 6. We first consider the time cost of the aggregation in Fig. 6(a). The amount of time for all the three methods increases with the data volume. The convolution and sketch methods need orders of magnitude less time than PWD. Note that PWD can only finish on a smaller number of objects. It follows from the fact that the two methods run in polynomial time, but PWD has an exponential time complexity. We then consider the space cost in Fig. 6 (b). The PWD method requires enumerating an exponential number of possible worlds and thus has very high space complexity. For convolution and sketch aggregations, the space cost increases polynomially and linearly, respectively. The effect of parallelization is shown in Fig. 6 (c) and (d). The acceleration on sketch-based method is not as significant as convolution-based method, because of it is of light-weighted computation. However, the overall acceleration for the two aggregation is still impressive, since the cost for convolution method is orders of magnitude higher.

Full Materialization. We examine the performance of full cube materialization in 7. First, we compare the construction efficiency of the proposed semi-distributive approach (Algorithms 1 and 2) with the holistic approach. For a fair comparison, both approaches are equipped with the same cost model meaning they are able to follow the optimal aggregation path. In Fig. 7 (a), the semi-distributive approach is about 3 orders of magnitude more efficient than the holistic approach. Note that the y-axis is in log scale. The efficiency is achieved by maximally reusing ancestor cuboids' computation. Second, we show the efficiency provided by the cost model. We consider two other competitors: one is the random model which arbitrarily selects an ancestor cuboid; another one is to simply use the I/O model proposed in [6] which does not address the computational cost considered in our problem settings. Fig. 7 (b) reports the effectiveness of the parallelization. In particular, when data size is $1000MB$, the overall materialization cost with 8 cores is about 28% of that with single core.

Partial Materialization. We report the results on partial materialization by varying the space budgets in Fig. 8. For both TPC and Climate data, the results with our proposed model dominate the others. For example, in Fig. 8 (a), when the space budget is equal to 50%, our model results in an efficient cube construction costing only 2/3 of the I/O model and the random model. We compare the construction efficiency between the convolution and sketch-based methods, as shown in Fig. 8 (b). First, the construction time of convolution-based aggregation increases with the space budget. Because in higher level cuboid aggregation, it corresponds to a larger pmf length which incurs higher computational cost. The construction time for the sketch-based method increases linearly and is much less.

SUM. We continue to examine the query performance. Each reported value is the average of 200 runs. We evaluate the query performance of `SUM` with different levels of materialization in Fig. 9 (a-b). In all cases, our model outperforms the random model by two times with the default space budget. The curve "optimum" represent the query time on the fully materialized cube, i.e., the space budget equals



(a) Materialization Time (b) Query Time (PSQ)

Fig. 10: Extensions to COUNT

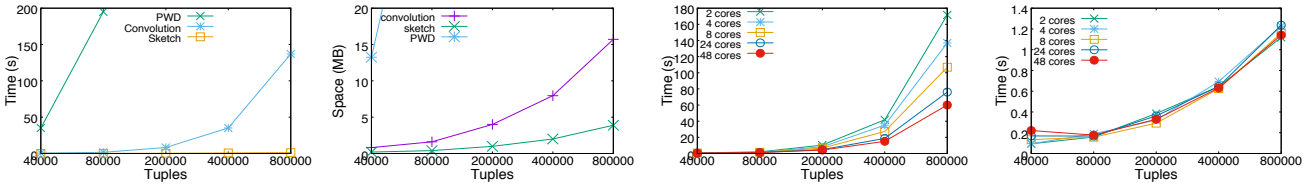
100%. In Fig. 9 (a), it achieves a good trade-off between construction and query evaluation when the space budget is greater than 40%. The query performance becomes very close to that of a fully materialized cube. Similar results on Climate dataset are observed from Fig. 9 (b). The efficiency of our method is achieved by accurate cost estimation, as shown in Fig. 9 (c). When the space budget is below 70%, the accuracy is above 90%. Notice that the error becomes big when the budget is above 90%. We argue that the accuracy is adequate because the cost model is mostly used with small space budgets and such a range of accuracy achieves best trade-offs between the on-the-fly construction and querying. The reasonability of studying cost models is explained by Fig. 9 (d), which shows the on-the-fly cuboid materialization dominates other parts. The results of PDQ is shown in Fig. 9 (e). The query time decreases with the increase of space budget as expected. More, the sketch-based pruning significantly improves the query efficiency. When the space budget equals 50%, the method with sketch pruning spends only 50% time of the one without pruning.

COUNT. As we have discussed, `COUNT` can be viewed as another form of `SUM`. So, we just briefly report its results. Fig. 10 (a) shows the materialization time w.r.t. the space budget. For both datasets, the trends are similar to the results on `SUM`, and the construction time for the sketch-based aggregation method increases linearly. Also, similar trends are observed for PSQ in Fig. 10 (b). Compared with results on `SUM`, the query time is much less, because the pmf length is much smaller. Results on PDQ is omitted due to page limits.

MAX. We show the query performance of `MAX` in Fig. 11. For PSQ queries, we show the performance by varying cost models adopted, as shown in Fig. 11 (a). In all cases, our cost model is better than the other two. The reason is explained by Fig. 11 (b). The estimation accuracy is steadily above 92% when the budget is below 90%. We further analyze the cost of materialization by varying the amount of objects in Fig. 11 (c). The query time increases moderately w.r.t. the data size and the `MAX` computation time, i.e., time cost for evaluating the probability distribution, forms only a very small part of the total. Due to the specialty of the aggregation function, `MAX` does not have closed form equation for sketching. Also, from the figure, there seems no need for doing so, since the computation efficiency is adequate, i.e., within 200 milliseconds. Similar facts are observed by varying the number of cores in Fig. 11 (d), where the max computation time takes only about 2 percents of the total. The performance of PDQ is reported in Fig. 11 (e). When the space budget is above 40%, the query can be finished within 2 seconds.

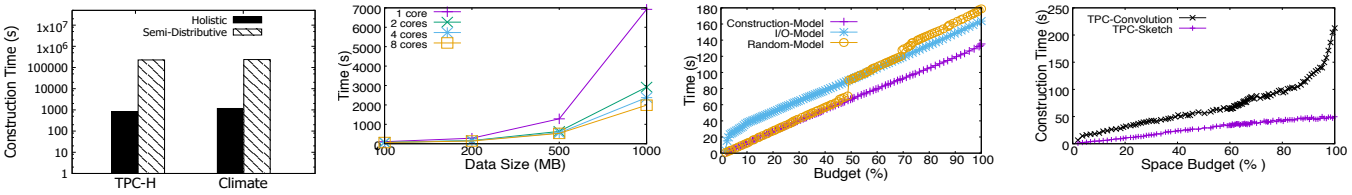
AVG. We report the PSQ performance of `AVG` in Fig. 12 (a), where the result with our proposed mode is better than

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2913420, IEEE Transactions on Knowledge and Data Engineering



(a) Time Cost vs. # of Obj (b) Space Cost vs. # of Obj (c) Parallelization (Convolution) (d) Parallelization (Sketch)

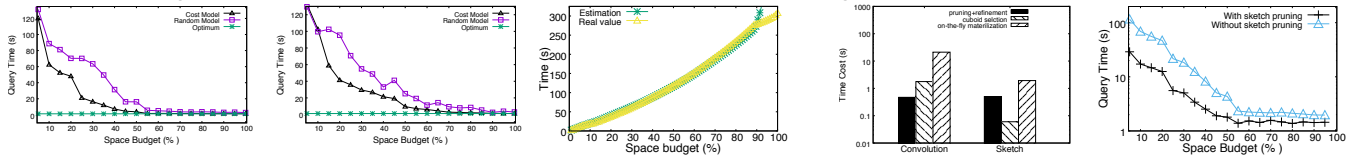
Fig. 6: Aggregation (TPC)



(a) Semi-distributive (b) Parallelization (a) Cost Models (b) Convolution vs. Sketch (TPC)

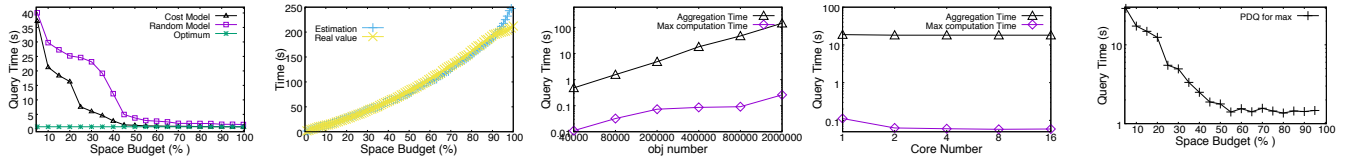
Fig. 7: Full Cube Materialization

Fig. 8: Partial Cube Materialization



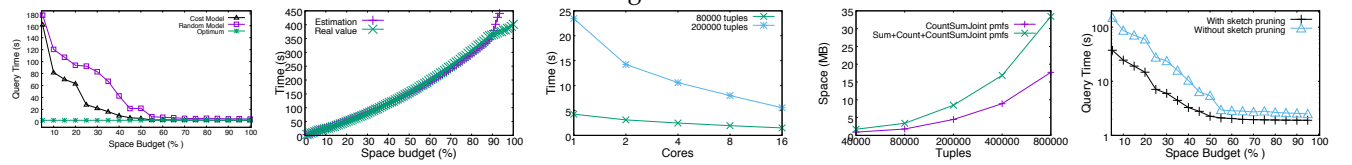
(a) PSQ (TPC) (b) PSQ (Climate) (c) Estimation Accuracy (d) Time Breakdown (TPC) (e) PDQ (TPC)

Fig. 9: SUM



(a) PSQ (TPC) (b) Estimation Accuracy (c) Scalability (d) Parallelization (e) PDQ (TPC)

Fig. 11: MAX



(a) PSQ (TPC) (b) Estimation Accuracy (d) Parallelization (d) Space Cost (e) PDQ (TPC)

Fig. 12: AVG

its competitors. The result of estimation is shown in Fig. 12 (b), where the accuracy is as high as over 90% when the budget is less than 90%. The parallelization result shows that the time cost decreases w.r.t. the number of cores. Also, the improvement is more significant if the object set is larger. From Fig. 12 (c), we can see the query cost scales down w.r.t. the number of computation cores, and the downward trend is more obvious with larger datasets. In Fig. 12 (d), we report the space cost for running `AVG`, because it takes extra space for storing the joint probability distribution. The storage cost for the joint distribution takes only about 50-60% of the total cost. The result of PDQ is shown in Fig. 12 (e). The result with sketch pruning is about 5 times faster, when space budget is 50%.

8 Conclusion

In this paper, we study multi-dimensional probabilistic data cubes by providing a complete set of techniques, including cuboid aggregation, parallel cube materialization, and query evaluation. We study convolution and sketch-based aggregation methods. We consider full and partial materialization with support of the proposed parallel framework. We support common OLAP aggregation functions, e.g., `SUM`, `COUNT`, `MAX` and `AVG`. Also, our proposal supports probabilistic versions of common OLAP queries, such as roll-up, drill-down, slicing, and dicing. Extensive experimental results show that our proposals are effective and efficient for querying multi-dimensional probabilistic data.

References

[1] D. Pfoser and C. S. Jensen, "Capturing the uncertainty of moving-object representations," in *SSD*, 1999.
[2] Z. Istvan, L. Woods, and G. Alonso, "Histograms as a side effect of data movement for big data," in *SIGMOD*, 2014.
[3] R. Cheng, J. Chen, and X. Xie, "Cleaning uncertain data with quality guarantees," in *VLDB*, 2008.
[4] N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," in *VLDB*, 2004.
[5] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh, "Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total," in *ICDE*, 1996.

[6] V. Harinarayan, A. Rajaraman, and J. D. Ullman, "Implementing data cubes efficiently," in *SIGMOD*, 1996.

[7] X. Xie, X. Hao, T. B. Pedersen, P. Jin, and J. Chen, "Olap over probabilistic data cubes i: Aggregating, materializing, and querying," in *ICDE*, 2016.

[8] M. Pilman, M. Kaufmann, F. Köhl, D. Kossmann, and D. Profeta, "Partime: Parallel temporal aggregation," in *SIGMOD*, 2016, pp. 999–1010.

[9] Z. Feng and E. Lo, "Accelerating aggregation using intra-cycle parallelism," in *ICDE*, 2015, pp. 291–302.

[10] Y. Tao and C. Sheng, "I/o-efficient bundled range aggregation," *TKDE*, vol. 26, no. 6, pp. 1521–1531, 2014.

[11] Z. He, P. Wong, B. Kao, E. Lo, R. Cheng, and Z. Feng, "Efficient pattern-based aggregation on sequence data," *TKDE*, vol. 29, no. 2, pp. 286–299, 2017.

[12] J. Lee, W. Han, H. J. Na, C. G. Park, K. H. Kim, D. H. Kim, J. Lee, S. K. Cha, and S. Moon, "Parallel replication across formats for scaling out mixed OLTP/OLAP workloads in main-memory databases," *VLDB J.*, vol. 27, no. 3, pp. 421–444, 2018.

[13] D. Makreshanski, J. Giceva, C. Barthels, and G. Alonso, "Batchdb: Efficient isolated execution of hybrid OLTP+OLAP workloads for interactive applications," in *SIGMOD*, 2017.

[14] P. Pedreira, Y. Lu, S. Pershin, A. Dutta, and C. Croswhite, "Rethinking concurrency control for in-memory OLAP dbmss," in *ICDE*, 2018.

[15] B. Salimi, J. Gehrke, and D. Suciu, "Bias in OLAP queries: Detection, explanation, and removal," in *SIGMOD*, 2018.

[16] R. Fink, L. Han, and D. Olteanu, "Aggregation in probabilistic databases via knowledge compilation," in *VLDB*, 2012.

[17] R. Fink and D. Olteanu, "Dichotomies for queries with negation in probabilistic databases," *TODS*, vol. 41, no. 1, pp. 4:1–4:47, 2016.

[18] P. Brown and S. Link, "Probabilistic keys," *TKDE*, vol. 29, no. 3, pp. 670–682, 2017.

[19] G. Cormode and M. N. Garofalakis, "Histograms and wavelets on probabilistic data," in *ICDE*, 2009.

[20] G. Cormode, F. Li, and K. Yi, "Semantics of ranking queries for probabilistic data and expected ranks," in *ICDE*, 2009.

[21] X. Lian and L. Chen, "Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data," *VLDB J.*, vol. 18, no. 3, pp. 787–808, 2009.

[22] J. Li, B. Saha, and A. Deshpande, "A unified approach to ranking in probabilistic databases," in *VLDB*, 2009.

[23] G. Cormode and M. Garofalakis, "Sketching probabilistic data streams," in *SIGMOD*, 2007.

[24] T. S. Jayram, S. Kale, and E. Vee, "Efficient aggregation algorithms for probabilistic data," in *SODA*, 2007.

[25] X. Xie, M. L. Yiu, R. Cheng, and H. Lu, "Scalable evaluation of trajectory queries over imprecise location data," *TKDE*, 2013.

[26] X. Xie, B. Mei, J. Chen, X. Du, and C. S. Jensen, "Elite: an elastic infrastructure for big spatiotemporal trajectories," *VLDB J.*, vol. 25, no. 4, pp. 473–493, 2016.

[27] X. Xie, R. Cheng, M. L. Yiu, L. Sun, and J. Chen, "UV-diagram: A voronoi diagram for uncertain spatial databases," *VLDBJ*, 2013.

[28] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *VLDB*, 2007.

[29] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson, "Supporting imprecision in multidimensional databases using granularities," in *SSDBM*, 1999.

[30] I. Timko, C. E. Dyreson, and T. B. Pedersen, "Pre-aggregation with probability distributions," in *DOLAP*, 2006.

[31] D. Burdick, P. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan, "OLAP over uncertain and imprecise data," in *VLDB*, 2005.

[32] D. Burdick, P. M. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan, "Efficient allocation algorithms for OLAP over imprecise data," in *VLDB*, 2006.

[33] A. Cuzzocrea and D. Gunopulos, "Efficiently computing and querying multidimensional olap data cubes over probabilistic relational data." in *ADBIS*, 2010.

[34] C. Ré and D. Suciu, "The trichotomy of HAVING queries on a probabilistic database," *VLDB J.*, vol. 18, no. 5, pp. 1091–1116, 2009.

[35] M. Heideman, D. Johnson, and C. Burrus, "Gauss and the history of the fast fourier transform," *ASSP Magazine, IEEE*, vol. 1, no. 4, pp. 14–21, 1984.

[36] V. John, I. Angelov, A. onc ul, and D. Thévenin, "Techniques for the reconstruction of a distribution from a finite number of its moments." in *Chemical Engineering Science*, 2007.

[37] M. G. Kendall, A. Stuart, and J. K. Ord, Eds., *Kendall's Advanced Theory of Statistics*. Oxford University Press, Inc., 1987.

[38] E. T. Jaynes, *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[39] J. Han, *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[40] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *J. Comput. Syst. Sci.*, vol. 31, no. 2, pp. 182–209, 1985.

Xike Xie is currently a professor in the School of Computer Science and Technology, University of Science and Technology of China. His research interests include data uncertainty, spatiotemporal databases, and mobile computing.

Kai Zou is a master student of School of Computer Science and Technology, University of Science and Technology of China. His research interests include data warehousing and parallel computation.

Xingjun Hao is a PhD student of School of Computer Science and Technology, University of Science and Technology of China. His research interests include data aggregation and big data analysis.

Torben Bach Pedersen is a Professor of Computer Science at Aalborg University, Denmark. His research concerns business intelligence and big data, especially "Big Multidimensional Data" -the integration and analysis of large amounts of complex and highly dynamic multidimensional data. He is an ACM Distinguished Scientist, an IEEE Senior Member, and a member of the Danish Academy of Technical Sciences.

Peiquan Jin is now an associate professor in the School of Computer Science and Technology, USTC. His research interests focus on spatiotemporal databases, flash-based databases, and Web information retrieval.

Wei Yang is an associate professor in the University of Science and Technology of China. His research interests include quantum data processing and data mining.