# Assessment Methods for the Interestingness of Cube Queries

Dimos Gkitsakis, Spyridon Kaloudis
Univ. Ioannina
Ioannina, Greece
dgkits@cs.uoi.gr,kaloudis.sp@gmail.com

Eirini Mouselli*
Natech S.A.
Ioannina, Greece
e.mouselli@natechsa.com

Veronika Peralta, Patrick Marcel
University of Tours
Blois, France
first.last@univ-tours.fr

Panos Vassiliadis
Univ. Ioannina
Ioannina, Greece
pvassil@cs.uoi.gr

## ABSTRACT

In this paper, we discuss methods to assess the interestingness of a query in an environment of data cubes. We assume a hierarchical multidimensional database, storing data cubes and level hierarchies. We focus our approach on a taxonomy of the dimensions of interestingness, and specifically, relevance, surprise, novelty, and peculiarity. We propose specific measures and algorithms for assessing the different dimensions of cube query interestingness in a quantitative fashion.

## 1 INTRODUCTION

*How interesting is a (data cube) query?* What are the fundamental characteristics that make a (data cube) query interesting for a user? Assessing query interestingness is important for at least two common scenarios: (a) *a-priori* interestingness prediction, and, (b) *a-posteriori* interestingness evaluation.

- A-priori prediction of query interestingness occurs in the case where a recommender system is in the process of automatically generating candidate queries, to provide the user with an overview of the information space, as well as with suggestions on how to explore it, or how to follow up on previous query in an on-going query session.
- A-posteriori evaluation of query interestingness is relevant in the case where a large number of queries have already been issued (possibly by other users too), they are cached and readily available, and we need to pick the ones that seem the most significant either in order to recommend them to a user, or, because they highlight best the user actions and goals in the query session.

The above are by no means an exhaustive enumeration of cases where the evaluation of query interestingness is important (e.g., the automatic generation of exploration sessions, can be considered as the middle ground between a priori and a posteriori cases [7]). The common thread in both cases, however, is that both for reasons of efficiency and computational overhead, and for reasons of cognitive load of the person who is involved in the process, it is imperative that a small subset of queries, out of a large number of candidates, are picked for further processing.

In our deliberations, we focus on data organized in cubes due to (a) their extreme relevance to the problem, due to the fact that analysts explore data in query sessions via Business Intelligence tools, (b) their simplicity – as the simplest possible database setting in terms of how data are presented to the end-users, (c) the most focused setup, also due to the simplicity of the underlying schema, but also because the queries follow a pattern of filtering and grouping with very specific joins between the dimension and fact tables, and, (d) the richness of information content, due to the presence of hierarchically structured dimensions that allow manipulating, examining and understanding the data from multiple layers of abstraction. In other words, cubes are relevant to the problem, simple, information rich, and allow focused query sessions to take place.

Therefore, in our work, we assume an OLAP environment, consisting of cubes, dimensions, levels, and aggregate cube queries posed in the context of user sessions. We will also assume the ability to register, extract, or simply approximate user goals, beliefs and profiles.

What is then the assessment of interestingness for cube queries? Well, to address the question, we will first frame the assessment aspect: we regard assessment as the process where an assessor (person or software) examines specific properties of an object that is evaluated (in our case: cube queries), within a certain context (in our case, as we will demonstrate, the multidimensional space, the query history, the goals, beliefs and interests of the user), for its degree of support/fulfillment of a property (in our case: interestingness aspects) via a method that objectively quantifies the above degree of support via a numerical score or label that is interpretable via a reference scale of assessment.

Intuitively speaking, we need to establish the different properties/aspects/dimensions of interestingness and introduce algorithms to numerically assess the objects of study (cube queries) for their "performance" with respect to these properties, in the context of a specific user (with his own characteristics) and a specific session.

Based on the study of the related literature, both in the area of psychology, and in the area of computer science, we have concluded that *interestingness is not a single entity, but rather, a vector of scores* along several *dimensions* [14]:

- Relevance: the extent to which a new piece of information (here: the results of the query) are related to the overall information goals, preferences, ... of the user.
- Surprise: the extent to which the result of the query contradicts, revises, updates the user's prior beliefs.
- Novelty: the extent to which the information presented to the users is new, and previously unseen to them.
- Peculiarity: the extent to which the query is different, and not in accordance with the previous queries of the session or history.

The fundamental premise upon which we build our taxonomy is the observation that the cornerstone of interest is curiosity (either in search to fill an information gap, or plainly the joy of

---

*Work done with Univ. Ioannina.

acquiring new information), and by following the need to "feed" the curiosity of the user, we end up with the above dimensions.

*In this paper, we link a taxonomy of the dimensions of interestingness with data cubes in hierarchically structured multidimensional spaces, and, we propose specific measures and algorithms for assessing the different dimensions of cube query interestingness in a quantitative fashion.*

**Roadmap**. The rest of this document is structured as follows: In Section 2 we review related work. In Section 3, we discuss the data and querying environment in which we operate. Then, we proceed to discuss formulae and algorithms for assessing interestingness for each of the four dimensions, with a varying degree of available information: novelty in Section 4, relevance in Section 5, peculiarity in Section 6, and, surprise in Section 7. We present experiments in Section 8 and conclude our deliberations in the final section.

## 2 RELATED WORK

In the literature, insights (also known as highlights, findings, discoveries, etc). demonstrate an interesting property or pattern for a subset of the data in a dataset, typically characterized by an interestingness score [7, 10, 15, 17, 20]. We start with the following fundamental dimensions of interestingness: (i) *relevance* (R): a cube query is contrasted to a user's exploration goal; (ii) *novelty* (N): a cube query is contrasted to a user's exploration history; (iii) *peculiarity* (P): the similarity of a cube query to a user's history is assessed (either at the level of the query expression or at the level of the query results); and (iv) *surprise* (S): the result of a cube query is contrasted to a user's belief.

*Peculiarity.* It appears that peculiarity has attracted most of the attention in the literature. The main measures defined in this dimension concern either (i) the significance, (ii) the coverage, or (iii) the coherency of the insights.

The *significance* of an insight [1, 5, 6, 8, 18, 21] allows to quantify its importance among its *peer data*. This importance is often related to the data distribution. [8] performs a preliminary ad-hoc attempt to measure significance via the difference in z-scores of the data obtained in two consecutive exploration steps. Recently, a trend is to turn insights into hypothesis testing [5, 6, 21], which has many advantages: (i) using the p-value for the insight significance, (ii) defining false discoveries (type-1 errors, e.g., visualizations supporting a non-significant insight) and false omissions (type-2 errors, e.g., visualizations not supporting a significant insight), (iii) defining credibility (e.g., percentage of visualizations supporting an insight). However, since the risk of type-1 error increases as more than one hypothesis are considered at once, a correction is needed in the statistical test to ensure that non-spurious insights are reported [21].

Measuring the *coverage* of the insight consists of quantifying how the subject of an insight represents the *entire dataset* [6, 13, 18]. In most cases, anti-monotonic conditions are checked to prune insights, like, for instance: if the subject of insight A is a superset of the subject of insight B, then the impact of A should be no less than the impact of B.

Characterizing the *coherency* of an insight compares the insight with others in the *exploration session*, to check whether a given exploratory operation is coherent at a certain point. For instance, in [7] heuristic classification rules are used to express general properties of the operations sequence (e.g., a group-by on a continuous, numerical attribute is incoherent) or on the input dataset's semantics (e.g., if the user focuses on flight delays,

aggregating on the "departure-delay time" columns is preferred). Other works use distances between exploration actions to measure how coherent a sequence of actions is; for instance, in [5] a weighted Hamming distance of relational query parts is used.

*Novelty.* Interestingness measures of the novelty dimension are used to characterize data in terms of either being new observations or operations in terms of *favoring going further* in the exploration. In its simplest expression, novelty can simply be measured as a Boolean indicating whether some data have already been seen [8]. However, more advanced definitions exist. For instance, in [15], a diversity measure is computed as the minimal Euclidean distance between the current observation and all the previous displays obtained. In [16], curiosity is inversely proportional to the number of times a result is encountered.

*Relevance.* Interestingness measures of the relevance dimension are used to characterize data in terms of the user being *familiar* with them. In [16], a familiarity measure is defined as the concentration ratio of target objects in a set. It is implemented as a variant of the Jaccard index between objects encountered during the exploration and a given target set of familiar objects. This measure is expected to increase as the EDA session goes on, to avoid over-exploiting a set of familiar objects.

*Surprise.* This dimension seems to be the one that attracted less attention for EDA. We can mention the work of Francia et al. [8] where surprise is measured as the proportion of values that have not been seen frequently, presented in models (e.g., clustering) extracted from the data under observation. A formal framework for defining measures of surprise had previously been introduced by De Bie for exploratory data mining [4]. Using an information-theoretic approach, the framework consists of quantifying the interactive exchange of information between data and user, accounting for the *user's prior belief state*. Approximating the belief that the user would attach to the result being expected is modeled as a background distribution, namely, a probability measure over the exploration results. This background distribution, which initially can e.g., be uniform over all the exploration results, is updated after each result is presented to the user.

Recently, we addressed the problem of *cell* interestingness [14]. However, a query is much more than a composition of its result cells, esp., if the interestingness of the query is to be assessed before deciding if we will execute it. *None of the previous works, however, exploits the hierarchies of a multidimensional space to compute cube query interestingness, along several dimensions. We address this shortcoming in the sequel of this paper.*

## 3 PRELIMINARIES & FORMAL BACKGROUND

In our deliberations, we assume the formal model of [19] for the definition of the multidimensional space, cubes and cube queries. We follow a simplified apodosis of the formalities here to allow a concise description. The reader is assumed to have knowledge of fundamental OLAP concepts.

**Multidimensional space**. Data are defined in the context of a multidimensional space. The multidimensional space includes a finite set of dimensions. Dimensions provide the context for factual measurements and will be structured in terms of dimension levels, which are abstraction levels that aid in observing the data at different levels of granularity. For example, the dimension *Time* is structured on the basis of the dimension levels *Day*, *Month*, *Year*, *All*.

A *dimension level L* includes a name and a finite set of values, $dom(L)$, as its domain. Following the traditional OLAP terminology, the values that belong to the domains of the levels are called *dimension members*, or simply *members* (e.g., the values Paris, Rome, Athens are members of the domain of level *City*, and, subsequently, of dimension *Geography*).

A *dimension* is a non-strict partial order of a finite set of *levels*, obligatorily including (a) a most detailed level at the lowest possible level of coarseness, and (b) an upper bound, which is called *ALL*, with a single value 'All'. We denote the partial order of dimensions with $\preceq$, i.e., $D.L_{low} \preceq D.L_{high}$ signifies that $D.L_{low}$ is at a lower level of coarseness than $D.L_{high}$ in the context of dimension $D$ – e.g., $Geo.City \preceq Geo.Country$.

We can map the members at a lower level of coarseness to values at a higher level of coarseness via an *ancestor function* $anc_{L^l}^{L^h}()$. Given a member of a level $L_l$ as a parameter, say $v_l$, the function $anc_{L^l}^{L^h}()$ returns the corresponding ancestor value, for $v_l$, say $v_h$, at the level $L_h$, i.e., $v_h = anc_{L^l}^{L^h}(v_l)$. The inverse of an ancestor function is not a function, but a mapping of a high level value to a set of *descendant values* at a lower level of coarseness (e.g., *Continent* Europe is mapped to the set of all European cities at the *City* level), and is denoted via the notation $desc_{L^h}^{L^l}()$. For example $Europe = anc_{City}^{Continent}(Athens)$. See [19] for more constraints and explanations.

**Cubes**. Facts are structured in cubes. The schema of a cube $schema(C)$, is a tuple, say $[D_1.L_1, ..., D_n.L_n, M_1, ..., M_m]$, *or simply* $[L_1, ..., L_n, M_1, ..., M_m]$, with the combination of the dimension levels (each coming from a different dimension) acting as primary key and context for the measurements and a set of measures as placeholders for the (aggregate) measurements. If all the dimension levels of a cube schema are the lowest possible levels of their dimension, the cube is a *detailed cube*, typically denoted via the notation $C^0$ with a schema $[D_1.L_1^0, ..., D_n.L_n^0, M_1^0, ..., M_m^0]$. The result of a query $q$ is a set of cells that we denote as $q.cells$.

Each record of a cube $C$ under a schema $[D_1.L_1, ..., D_n.L_n, M_1, ..., M_m]$, also known as a *cell*, is a tuple $c = [l_1, ..., l_n, m_1, ..., m_m]$, such that $l_i \in dom(D_i.L_i)$ and $m_j \in dom(M_j)$. The vector $[l_1, ..., l_n]$ signifies the *coordinates* of a cell.

**Queries**. A cube query is a cube too, specified by (a) the detailed cube over which it is imposed, (b) a selection condition that isolates the facts that qualify for further processing, (c) the grouping levels, which determine the coarseness of the result, and (d) an aggregation over some or all measures of the cube that accompanies the grouping levels in the final result.

$$q = < C^0, \phi, [L_1, ..., L_n, M_1, ..., M_m], [agg_1(M_1^0), ..., agg_m(M_m^0)] >$$

We assume:

- Selection conditions which are conjunctions of atomic filters of the form $L = value$, or in general $L \in \{v_1, ..., v_k\}$. Selection conditions of this form can eventually be translated to their *equivalent* selection conditions at the detailed level, via the conjunction of the detailed equivalents of the atoms of $\phi$. Specifically, assuming an atom $L \in \{v_1, ..., v_k\}$, then $L^0 \in \{desc_L^{L^0}(v_1) \cup ... \cup desc_L^{L^0}(v_k)\}$, eventually producing an expression $L^0 \in \{v_1', ..., v_{k'}'\}$ is its detailed equivalent, called *detailed proxy*. The reason for deriving $\phi^0$ is that $\phi^0$, as the conjunction of the respective atomic filters at the most detailed level, is directly applicable over $C^0$ and produces exactly the same subset of the multidimensional space as $\phi$, albeit at a most detailed level

of granularity. For example, assume $Year \in \{2018, 2019\}$, its detailed proxy is $Day \in \{2018/01/01, ..., 2019/12/31\}$. For a dimension $D$ that is not being explicitly filtered by any atom, one can equivalently assume a filter of the form $D.ALL = all$.

- Aggregation functions $agg_i$ include aggregate functions like $\{sum, max, min, count, ...\}$ with the respective well-known semantics.

**Signatures and detailed areas**. We will use the term **signature** to refer to sets of coordinates that specify an area of interest in the multidimensional space. Specifically:

- The signature of a cell c, denoted as $c^+$, is its coordinates.
- The signature of an atomic filter $\alpha : L \in \{v_1, ..., v_k\}$ is the value set $\{v_1, ..., v_k\}$ and it is denoted as $\alpha^+$.
- The signature of a selection condition of the form $\phi : \alpha_1 \wedge \cdots \wedge \alpha_n$ (assuming a single atom per dimension) is the expression $\phi^+ : \alpha_1^+ \times \cdots \times \alpha_n^+$. In other words, we compute the Cartesian product of the values of the involved atom signatures.
- The signature of a query $q$, $q^+$ is the *set* of coordinates computed as follows: (a) compute the signature, i.e., the set of coordinates pertaining to $\phi^0$, the detailed equivalent of its selection condition; (b) within each of these coordinates, replace the (detailed) value of each dimension by its ancestor value at the level of the schema of the query. This guarantees that the resulting coordinates will be the coordinates of the query result.

The *detailed signatures* of the above categories are produced by replacing the respective values of their regular signatures with the expression $desc_L^{L^0}(\cdot)$, computing the respective set of descendant values and taking their union. The *detailed signature of a query* is (simply) the set of coordinates that pertain to the signature of $\phi^0$.

The **detailed proxies** of expressions are the respective expressions transformed at the most detailed level for each of the involved dimensions. The *detailed proxy of a query*

$$q = < C^0, \phi, [L_1, ..., L_n, M_1, ..., M_m], [agg_1(M_1^0), ..., agg_m(M_m^0)] >$$

is the query (i.e., an expression again)

$$q^0 = < C^0, \phi^0, [L_1^0, ..., L_n^0, M_1^0, ..., M_m^0], [agg_1(M_1^0), ..., agg_m(M_m^0)] >$$

**Detailed areas** are sets of cells, pertaining to an aggregate cell, or set of cells, like, e.g., the result of a query. The *detailed area* of a cell $c :< v_1, ..., v_n >$ is the set of descendant cells that can be obtained by replacing each of its coordinates, say $v_i$, by $desc_L^{L^0}(v_i)$ and taking the Cartesian product of each such value set. The *detailed area* of the query $q$ is the set of cells of the result $q^0$, which we denote as $q^0.cells$.

# 4 NOVELTY OF A QUERY

Novelty assesses the amount of previously unknown information delivered to the user via a query. Due to this inherent characteristic, we need to either explicitly know, or at least estimate the prior knowledge of the data that the user has.

Naturally, a system is not in a position to actually have knowledge of the user's memory or knowledge. Knowledge can come to the user via external channels, not related to the query answering and thus, the system necessarily has "knowledge" of just a subset of the user's actual knowledge. At the same time, one should also take account of the effects of time that erases, hides or distorts the remembrance of facts encountered in the past. Although in our following deliberations we will not directly

address the above problems, we will occasionally offer insights on how to handle some of them. However, when we use the term "knowledge" we simplify and approximate the situation, by assuming that the system knows what the user has seen, or what the user has explicitly stated that she believes.

Explicit knowledge is primarily attained by knowing the history of user queries (and assuming that the user remembers it). A second way to approximate what the users remember is to exploit their registered beliefs with a low level of confidence, by making the rational assumption that since they have expressed practically uncertain beliefs about some cells, they do not know their values. Novelty is mostly goal-independent, i.e., it is not affected by neither the current (goal) or the typical (key interests) informational needs of the user.

Summarizing, novelty is mostly related (a) to history, and, (b) to registered values for beliefs with confidence below a certain threshold. We will examine the different alternatives in the respective subsections.

## 4.1 Novelty assessment in the presence of a query history

First, we will assess the novelty of a cube query $q$ assuming a query history $Q = \{q_1, \ldots, q_n\}$ exists. We use the following terminology:

(1) *Syntactic* vs *Extensional* Assessment: syntactic assessment is based only on query definitions, whereas extensional also assumes the presence of the cells of the query result(s).
(2) *Same-level* vs *Detailed* Assessment: same-level (equiv., immediate) assessment assumes that two cubes are at the same level of aggregation; detailed (equiv., indirect or derivable) assessment means that the comparison of two cubes will be done at levels lower than their definition – typically, we will use the most detailed level as the common ground upon which the constituting detailed cells for two cubes can be compared.
(3) *Full* vs *partial* Assessment: full assessment means that the checks made return a true/false answer on whether a new query is entirely novel or not; partial assessment means that the checks return a novelty score as a real number (typically in the interval $[0 \ldots 1]$). Naturally, a partial assessment that returns 1, also implies full novelty.

*4.1.1 Same-Level Assessment of Novelty.* Assume that we only check $q$ against members of $Q$ whose schema is at the same level with $Q$. We also require the same detailed measures and aggregate functions to be used, otherwise the comparison is referring to essentially different measures, and also different numbers, and, therefore, novelty is guaranteed.

**Full Syntactic Same-Level Assessment of Novelty**. In this case, the question to be answered is: Given $q$ and $Q = \{q_1, \ldots, q_n\}$, is there any $q_i \in Q$ such that $q = q_i$? In this case, the solution is a trivial *syntactic* check: we iterate through the syntactic definitions of the queries of $Q$ and check whether there is any query that is identical to $q$. The check is (a) full (practically a true/false decision), (b) syntactic (without using the cells of the query results), and, (c) same-level, i.e., only between same level cubes. The case of query containment (instead of equivalence, discussed here) is found in the long version of this paper [11].

*4.1.2 Detailed Assessment of Novelty.* Assume now that instead of checking cubes defined at the same level, we compare cubes with respect to their constituting cells at the most detailed level. We indicatively present an extension-based algorithm, that assumes cells are available. Syntactic checks are similar.

**Partial Extensional Detailed Assessment of Novelty**. In this case, the question to be answered is: Given $q$ and $Q = \{q_1, \ldots, q_n\}$, can we identify which part of the results of the detailed area of $q^0$ are already covered by the detailed areas of the queries of $Q$?

---

**Algorithm 1:** Cell-based extensional enumeration of covered detailed cells

---
**Input:** A query $q$; the query history $Q$ expressed as a set of queries $q_i$
**Output:** The subset of the cells of $q^0$, say $q^{cov}$ that are also part of the union of the results of the queries in $Q$, i.e., the union of $q_i^0$, and its complement $q^{nov}$

1 **begin**
2     produce $q^0.cells$
3     produce $q_i^0.cells$ for all $q_i$
4     populate the hashmap(cell signature) $Q^0 \leftarrow \bigcup_i q_i^0.cells$
5     $q^{cov^0} \leftarrow \emptyset$
6     $q^{nov^0} \leftarrow q^0.cells$
7     **forall** $c^0 \in q^0.cells$ **do**
8         **if** $c^0 \in Q^0$ **then**
9             remove $c^0$ from $q^{nov^0}$ and add it to $q^{cov^0}$
10         **end**
11     **end**
12     **return** $q^{cov^0}, q^{nov^0}$
13 **end**

---

Algorithm 1 computes the union of the detailed areas of the queries in the query list and intersects it with the detailed area of the query under question. We remark that only the queries in the history sharing the same measures and aggregation functions with $q$ are passed to the algorithm. The resulting novelty is the fraction of the detailed not covered (i.e., novel) cells over the entire detailed area of $q$.

$$PartialDetailedExtensionalNovelty = \frac{|q^{nov^0}|}{|q^{nov^0}| \cup |q^{cov^0}|}$$

The check is (a) partial (practically a normalized score), (b) extensional (via cells), and, (c) detailed, i.e., with respect to the detailed levels of the involved cubes.

The complexity of Algorithm 1 is mainly determined by the cost of answering the detailed queries in Lines 2 and 3 that produce the cells for $q^0$ and $q_i^0$ for both the input query and the query history. The complexity of these actions is: (a) linear with respect to the size of the query history, and, (b) linear with respect to the cube size, assuming that the cube query is linear with respect to the cube size. The rest of the algorithm, requires a linear in-memory pass of the result to populate $Q^0$ and a linear lookup for each cell of $q^0.cells$ to cross-check if it belongs to $Q^0$. Again, this cost is linear, yet, we consider it insignificant comparing it to the time needed for query answering. Therefore, the overall cost of the algorithm is linear with respect to the size of the query history and to the cube size.

## 4.2 Novelty assessment in the presence of belief statements

Assume now that we do not have explicit knowledge of the user history, or key interests, but we do have an estimation of probabilities for the likely values of some cells in the multidimensional space. Specifically, assume that for certain cells, it has been possible to either deduce or explicitly have the user register probabilities per expected value for the value $m = c.M$, of a cell $c$ and a certain measure $M$. So, some cells in the multidimensional space are annotated with a set of *cell expected-value statements*, which are statements of the form

$$p(M \in [l_i \dots u_i]|c) = p_i, p_i \in [0..1],$$
$$[l_i \dots u_i] \text{ is a range of values of } dom(M)$$

or of the form

$$p(M \in \mathbf{s_i}|c) = p_i, p_i \in [0..1],$$
$$\mathbf{s_i} \text{ is a discrete, finite set of values of } dom(M)$$

For uniformity of notation we will use the syntactic form $p(M \in m_i|c) = p_i$ to denote either a range or a finite set of values for the value-set of the expressed belief. The distinction makes no difference for the evaluation of novelty.

We refer to the set of statements of the above form for a cell $c$, the *probable active domain* of $c$, or $dom^{pa}(c)$. A *well-formed probable active domain* of a cell has the property that all its statements' probabilities sum up to 1. However, requiring well-formed probable active domains is too restrictive, in the sense that maybe some probabilities are unknown, or hard to evaluate; thus, we do not require it as a necessary property for the sequel.

We call a cell $c$ to be $\Pi - known$ if, within the statements of the probable active domain of $c$, there exists a probability $p_i$ which is equal or higher to a threshold $\Pi$. Otherwise, if all the probabilities of $c$ are below $\Pi$ the cell is called $\Pi - unknown$.

*The intuition behind this treatment lies on the observation that if a user has a set of beliefs about the behavior of a cell, with a high amount of certainty (i.e., the probability is above a certain threshold), then we cannot consider the cell to be "unknown" to the user.* The result of a query might be surprising, if it is far from the expected value, but the existence of this area of the multidimensional space is not novel to the user.

To give a practical example, assume the following user beliefs

$$p(sales \in [100..200] \mid city = Athens, year = 2020) = 30\%$$
$$p(sales \in [80..100] \mid city = Athens, year = 2020) = 70\%$$

assuming all other dimensions set to ALL. For a particular cell therefore, concerning the sales in Athens for 2020, we have a probability distribution for the range of its values. Let's also assume that we have agreed that if a user has a belief higher or equal to 50% for a cell's measure, then he "knows" the cell; this means setting a value of $\Pi = 50\%$. Given the above belief set, and the existence of a belief with probability 70% (i.e., higher than $\Pi$), we can say that this particular cell is indeed 50%-known, and thus consider it not novel.

Let $B$ a set of beliefs expressed as cell expected-value statements for a set of cells $C^B$. Assume now a query $q$, and its resulting cells $C = q.cells$. Assume also a threshold $\Pi$. Then, the $\Pi - direct \ novelty$ of $q$ is the percentage of cells of $q.cells$ that are $\Pi - unknown$. We can distinguish three cases for computing

belief-based novelty, depending on the level that the cells of $C^B$ have been defined: (a) at an arbitrary level of aggregation, (b) at the level of the query, or (c) at the most detailed level. We explore the last case in the sequel and refer the interested reader to the long version [11] for a thorough analysis. Here, the set of beliefs $B$ is expressed over a set of cells $C^B$ at the most detailed aggregation level. Then, we can compare the cells of $q$ with the cells of $C^B$ by converting them to their detailed equivalents. Algorithm 2 performs the computation of novelty.

---

**Algorithm 2:** Partial Extensional Detailed Belief-Based Enumeration Of Covered Cells

**Input:** A query $q$; a set of beliefs $B$ over a set of cells $C^B$ at the most detailed level; a threshold $\Pi$ for deciding if a cell is eligible for being novel

**Output:** The subset of the cells of $q^0$, say $q^{cov^0}$ that are also part of the space the beliefs cover, as well as its complement $q^{nov^0}$

1 **begin**
2     produce $q^0.cells$
3     $q^{cov^0} \leftarrow \emptyset$
4     $q^{nov^0} \leftarrow q^0.cells$
5     $C^\star \leftarrow$ the subset of $C^B$ for which there exists a known belief, i.e.,
    $\{c \mid c \in C^B, \exists \, p(M \in m|c) \in B, p(M \in m|c) \geq \Pi\}$
6     **forall** $c^0 \in q^0.cells$ **do**
7         **if** $c^{0^+} \in C^{\star^+}$ **then**
8             remove $c^0$ from $q^{nov^0}$ and add it to $q^{cov^0}$
9         **end**
10     **end**
11     **return** $q^{cov^0}, q^{nov^0}$
12 **end**

---

Then, we can compute the novelty of the query $q$ as usual:

$$PartialDetailedExtensionalBeliefNovelty = \frac{|q^{nov^0}|}{|q^{nov^0}| \, \cup \, |q^{cov^0}|}$$

The check is (a) partial (practically a normalized score), (b) extensional (via cells), and, (c) detailed, i.e., with respect to the most detailed cells of the data space. The *Syntactic* version of the algorithm (as contrasted to the *Extensional* one) is quite similar, albeit with the difference that no cells in the query result are needed and all sets and comparisons are performed with respect to the signatures of the queries. The complexity is linear to the result size (assuming the set $C^B$ is fixed) and linear to the size of the set $C^B$ assuming the query result size is fixed. In the case that only the query expression is given as input to the algorithm, and the query result has to be computed, the cost is dominated by the computation of $q.cells$, which is linear to the data cube size.

## 5 RELEVANCE OF A QUERY

Relevance is a dimension that pertains to retaining focus towards a specific information goal (or a set of them). The dimension of relevance ensures that the data exploration does not wander around areas of the multidimensional space that are not of interest to the current information acquisition goal.

This is particularly the case with business intelligence scenarios, where the need to satisfy an informational gap (either on an ad-hoc or a recurring basis) is the main driver for accessing

the database for data. This does not mean that the queries are pre-fixed, however: the quest for an information goal is very often "open" and an exploration of a certain sub-space of the data, possibly viewed from different angles and at different levels of granularity. In [14] we have named this exploration a "walk" in the multidimensional space.

As the above discussion demonstrates, a foundation for the assessment of the relevance of a query to an exploratory session or a recommendation to the use is the existence of an informational goal. The goal can be an ad-hoc goal for information, or a recurring one, based on a profile of data that have to be collected to answer recurring questions of the analyst. Specifically, we can discriminate between several cases: (a) the case where the goal is explicitly stated, or, (b) the case where the goal has to be inferred from collateral profile information. In the former case, we will assume that the analyst specifies an area of the information space via a selection predicate (again, the way this is extracted is orthogonal: it can be explicitly requested, it can be inferred from a natural-language expression, it can be part of a query or a KPI, etc). In the latter case, the user has not provided any such information, and the system has to infer the intended goal from other means – examples include the history of past analysis actions, or possibly a profile, or a set of registered KPIs. For lack of space, we refer the interested reader to the long version [11] for the former discussion, and here, present only the (more realistic) case where no explicit user goal has been given.

**Relevance assessment in the absence of a declared user goal**. Assume that an explicit goal to study a certain subset of the multidimensional space is not available, but instead, the system has access to a set of KPIs, expressed as a set of annotated queries $Q = \{q_1, \ldots, q_n\}$, which we call *beacon queries*, that approximate the user interest. KPIs are explicit expressions of time-invariant interests (rather than a current user goal), so, even if they do not explicate exactly what the user wants to achieve *now*, they act as reference points of relevance for the user's interest.

As a side-note, observe that, in extremis, one could even resort to the user's history for indications of relevance. Past queries are last-resort, coarse manifestations of relevance, as they are only in the past and not necessarily linked to what the user explores now, or, they could be erroneous, or playful, or eventually irrelevant, etc. However, despite all these valid reservations, it could be the case that this is the only thing that the system knows about the user's idea of what is relevant.

**Intuition**. What we want to assess is how much a new query $q$ overlaps with the set $Q$ of beacon queries. Observe that all the methods that we define assess the overlap of levels and coordinates between $q$ and the queries of $Q$; *measures and aggregate functions are not involved in the assessment of relevance, as the idea is to "highlight" the subset of the multidimensional space that seems relevant to the user.*

In the rest of this subsection, we simplify the discussion by avoiding aging factors and possible weights of the different queries and considering a single input for the interestingness assessment algorithm: a set of beacon queries which we (approximately) deem to be relevant. We will also use the notion of *coverage*, already discussed for novelty, aiming towards finding the overlap of the area covered by the beacon set and the area pertaining the current query.

**The special case where all queries are defined at the same level**. Assuming all cubes of $Q$ and $q$ are at the same level, we can assess relevance via (a) a full syntactic check returning

true/false and (b) a partial check returning a relevance score

$$PartialSameLevelSyntacticRelevance = \frac{|q^{cov^+}|}{|q^+|}$$

It is important to stress that the same-level relevance can only be applied in the case where all the cubes are at the same level of abstraction. Overall, the idea is that the beacon-set provides a homogeneous space for query evaluation at the same level, and thus, we can compute relevance without having to resort to the detailed space. The *Extensional* counterpart of relevance (e.g., *PartialSameLevelExtensionalRelevance*) is defined equivalently, with cells of the query result instead of signatures.

**Foundations of history-based relevance assessment**. The most fundamental assessment method of all is to compare the union of the detailed signatures of the queries of $Q$ with the signature of $q$. The amount of overlap signifies the relevance of the new query.

To characterize the cells of the result of $q$ (in fact: their coordinates) as previously-covered vs novel, we can refer to a signature-based variant of Algorithm 1, this time passing all the history as argument, i.e., without the requirement of same measures and aggregate functions. Equivalently, we can use (a) the detailed proxy of $q$, $q^0$ and (b) the detailed equivalents of the queries of $Q$, $q_i^0$, and pass them as input to the algorithm ComputePartial-ImmmediateCubeCoverage of [19]. Observe, that when working at the detailed level, coordinates and cells are equivalent, as measures are not taken into consideration. Then, the sets $q^{cov^{0^+}}$ and $q^{nov^{0^+}}$ (respectively, $q^{cov^0}$ and $q^{nov^0}$) are produced.

$$PartialDetailedExtensionalRelevance = \frac{|q^{cov^0}|}{|q^0|}$$

$$PartialDetailedSyntacticRelevance = \frac{|q^{cov^{0^+}}|}{|q^{0^+}|}$$

The complexity of computing all these formulas is practically the same with the one of Algorithm 1, and therefore, linear with respect to query history and fact table size.

# 6 PECULIARITY OF A QUERY

How peculiar is a query? To understand peculiarity we must understand that its essence lies in discriminating a particular object (in our case: a query) from its peers (in our case: a session, history, or just collection of other queries, to be used as the context for the assessment of peculiarity). Beliefs, Key Interests and Goals are not explicitly treated here; however to the extent that they can be expressed as queries, peculiarity based on these aspects can also be evaluated.

Therefore, in the rest of our deliberations, we assume that every query $q$ is going to be assessed against a collection of queries $Q = \{q_1, \ldots, q_n\}$. This generic setup can cover two alternative situations: (a) a set of KPIs, each expressed via a query, collectively describing a set of static key interests of the user, and, (b) a set of queries in the history (be it the current session, or the history of previous sessions).

## 6.1 Syntactic Outlierness

Assume the query $q$ and a collection of queries $Q = \{q_1, \ldots, q_n\}$. How different is $q$ from the collection $Q$?

Fundamentally, the question boils down to answering the assessment of the distance of two queries. To support our discussion

in the sequel we assume two queries *over the same data set* in a multidimensional space of $n$ dimensions.

$$q^a = \mathbf{DS}^0, \phi^a, [L_1^a, \dots, L_n^a, M_1^a, \dots, M_{m^a}^a], [agg_1^a(M_1^{a^0}), \dots, agg_m^a(M_{m^a}^{a^0})]$$

and

$$q^b = \mathbf{DS}^0, \phi^b, [L_1^b, \dots, L_n^b, M_1^b, \dots, M_{m^b}^b], [agg_1^b(M_1^{b^0}), \dots, agg_m^b(M_{m^b}^{b^0})]$$

To solve the problem of computing the distance of two queries, we use the syntactic formula from [19], which, in turn, is based on results from (see [2], [3], [12]).

The syntactic distance of the two queries is expressed by the weighted sum of structural distances between their selection conditions, their grouping levels, and their measures, as:

$$\delta(q^a, q^b) = w^\phi \delta^\phi(q^a, q^b) + w^L \delta^L(q^a, q^b) + w^M \delta^M(q^a, q^b),$$

such that the sum of the weights $w^i$ adds up to 1. We follow [2] and recommend the following weights: $w^\phi$: 0.5, $w^L$: 0.35, $w^M$: 0.15.

Given, then, the [19] method for computing distance of two queries $\delta(q^a, q^b)$, the computation of the distance of a new query $q$ to a pre-existing collection of queries $Q$ can be computed via several possible methods, out of which we highlight a couple of prominent ones:

(1) A simple statistic over the distances of the query to the set members, $\delta(q, Q) = \gamma(\delta(q, q_i)), q_i \in Q, \gamma \in \{min, max, average, median\}$.

(2) k-nn distance of the query to the set, $\delta(q, Q) = k$-th smallest $\delta(q, q_i), q_i \in Q$. Practically, this entails ranking all the distances of $q$ to the elements of $Q$ in ascending order and take the k-th one.

The check is (a) partial (practically a normalized score), (b) syntactical (without using the cells of the query results), (c) depending upon the statistic or function that determines the final value of the metric, and, (d) indifferent to the schema levels of the involved cubes. We can define a Partial Syntactic Cube Outlierness based on which method we pick for the determination of the final value, e.g., *Partial Syntactic Average Cube Outlierness* uses the average query distance to determine the outlierness of the measured query. To the extent that we refer to syntactic checks, data size is irrelevant for the complexity of the algorithm. However, the algorithm requires a linear pass from all the queries of the history and a pairwise computation of distance at its first phase, as well as the determination of the final peculiarity (again requiring at most a linear past of all distances): therefore, the complexity is linear with respect to the size of the collection $Q$.

## 6.2 Value-based Outlierness

When we address the issue of value-based outlierness assessment, we base the result of the assessment on the actual values of the cells of the result of the query. Then, we treat each query as a set of cells (each cell primarily identified by its coordinates). The question boils down to assessing how distant are the queries $q^a$ and $q^b$ with $q^a.cells = \{c_1^a, \dots, c_{n^a}^a\}$ vs. $q^b.cells = \{c_1^b, \dots, c_{n^b}^b\}$

Earlier works about comparing queries through their sets of cells, such as [9], have shown that it is not straightforward to assess their distance. The reasons can be identified as follows:

- It is not straightforward how to map the cells of the one query to another; this is especially true if the cardinality of the two queries is not the same;

- It is possible that the two queries are defined at different levels of aggregation, which means that they are not directly comparable;
- Even if the above problems are not present, deciding a mapping from the cells of $q^a$ to the cells of $q^b$ is not a straightforward task.

*6.2.1 Jaccard-based resolution via cell comparison at the detailed level.* A possible answer to the problem is to address the issue by referring to the detailed cells that pertain to the aggregate cells that constitute the results of the compared queries. Remember that we refer to the set of cells that produce an aggregate cell as the *detailed area* of the cell; the detailed area of a set of aggregate cells is defined respectively. Let $q_1^0.cells$ be the detailed area of $q_1$ over $C^0$ and $q_2^0.cells$ be the detailed area of $q_2$ over $C^0$. Then, we can compute the Jaccard similarity of the two detailed areas. The distance of the two queries is: $distance(q_1, q_2)$ = 1 - $JaccardSimilarity(q_1^0.cells, q_2^0.cells)$.

---

**Algorithm 3:** Partial Extensional Detailed Jaccard-Based (Value-based) Cube Peculiarity

**Input:** A new query $q$, the query history $Q$, and an integer $k$ for picking the k-th neighbour
**Output:** the PartialExtensionalDetailedJaccard-BasedCubePeculiarity $valueBasedPeculiarity(q|Q)$

1 **begin**
2     Let $L = \emptyset$ a list of Jaccard distances
3     Compute $q^0$, i.e., the detailed area of interest for the query $q$
4     **forall** $q_i \in Q$ **do**
5         Compute $q_i^0$, i.e., the detailed area of interest for the query $q_i$
6         Compute the Jaccard distance $JD_i = 1 - \frac{|q_i^0 \cap q^0|}{|q_i^0 \cup q^0|}$
7         add $JD_i$ to $L$
8     **end**
9     $L_s$ = Sort $L$ ascending into a sorted list
10     **return** $peculiarity(q|Q) = L_s[\,k\,]$
11 **end**

---

The intuition of the above is based on the idea that the outlierness of a query is based on how much overlap its detailed cells have with the detailed cells of the queries in the history. The check is (a) partial (practically a Jaccard distance), (b) extensional (with the use of the cells of the query results), and, (c) detailed, i.e., with respect to the detailed levels of the involved cubes. Thus, we define the Partial Extensional Detailed Jaccard-Based Cube Peculiarity (for short: Value-based Peculiarity) as the result of Algorithm 3.

The execution cost is dominated by the execution of the detailed queries for both the reference queries and the queries of the history $Q$. The complexity of the algorithm is obviously linear with respect to the history size, since there is a single detailed query $q_i^0$ to be executed per member of $Q$. Also, the in-memory check between the results of the queries is also linear with respect to the history size. At the same time, the complexity is also linear with respect to the cube size, assuming that the execution cost for all the queries linearly depends on the cube size (i.e., all the involved queries have their execution time scale linearly with the same scale factor over the cube size).

# 7 SURPRISE OF A QUERY

Surprise is an interestingess dimension that depends mainly (if not only) on prior beliefs. The main idea about assessing surprise is to evaluate how far from the prior beliefs of the analyst do the actual values lie. The two problems that one has to handle are: (a) *what kind of beliefs can we express, and how?*, and, (b) *assuming these beliefs have, somehow, been expressed, how can we compute surprise on their basis?*

## 7.1 Expressing beliefs

We can express beliefs in a variety of ways: specific values, expected intervals, probabilities; we can even label results and give probabilities for the labels, too [11]. Is it necessary, however, for the analysts to express beliefs manually? In the case of KPIs that label performance, this is explicitly done. In the general case, all analysts work with some form of predictions that are automatically derived via methods in the spectrum from a simple regression over past values to elaborate statistical models that economists use.

## 7.2 Computing surprise: the overall setup

Assume that for certain cells in the multidimensional space, we can register or compute their expected values for specific measures (several alternatives are discussed in the rest of this section). So, for such a cell, for each of these measures, we have (a) the actual value $m$, and, (b) the expected value $m^e$.

Then, the questions that we need to answer are (a) how do we assess the surprise for a specific cell over a specific measure, (b) how do we assess the surprise for a specific cell, with respect to all its measures (assuming multiple such measures exist), and, (c) how do we assess the overall surprise of a query result (which, of course, includes a set of cells)?

Let us start with a single measure for a single cell. Fundamentally, surprise is a function of how far the expected from the actual value lies. Therefore, $surprise(c.M) = (distance(m, m^e))$ – for example, $surprise(c.M) = |m - m^e|$. Assuming now a set of measures per cell, the total surprise of a cell is an aggregate measure computed over the set of surprise values for the various measures of a cell (e.g., the number of measures indicating a non-zero amount of surprise, or maybe the maximum, or the average surprise). Formally, $surprise(c) = f^{agg}_{cell}(surprise(c.M_i))$, with $f^{agg}_{cell} \in \{count, sum, mean, median, max, min, ...\}$.

Finally, now that we can compute the surprise for each individual cell, we can proceed in computing the surprise for a set of cells, e.g., a query result. The surprise of a set of cells, say $C = \{c_1, \ldots, c_n\}$ is $surprise(C) = f^{agg}(surprise(c_i))$, with $f^{agg} \in \{count, sum, mean, median, max, min, ...\}$

One possible concern here is what happens if there is no expected value registered for a measure of a cell. Then, there are two ways to handle the situation: (a) this particular measure value does not participate in the rest of the computation, or, (b) a mechanism for computing a derived expected value, against which we will perform the comparison (e.g., the average of the expected values, an interpolation over certain criteria, etc), is introduced. Unless explicitly mentioned otherwise, the former policy of excluding the respective measure value from any computation will be our reaction of choice.

## 7.3 Value-based average cell surprise

The most simple implementation of the assessment of surprise is to (a) compute a simple distance of the actual and the expected

---

**Algorithm 4:** Value-based surprise assessment for a single measured cube by absolute distance for expected values and averaging of cell surprise

**Input:** A cube $C$ including a set of cells $\{c_1, \ldots, c_n\}$ with a single measure $M$, a set of expected values for each cell $E = \{m^e_1, \ldots, m^e_n\}$

**Output:** The (average) surprise carried by the cube $C$

```
1  begin
2  │   countOfCellsWithSurprise = 0;
3  │   C.surprise = 0;
4  │   forall c ∈ C do
5  │   │   c.surprise = null;
6  │   │   if ∃ an expected value c.mᵉ for c.m then
7  │   │   │   c.surprise = |c.m − c.mᵉ|;
8  │   │   │   countOfCellsWithSurprise + +;
9  │   │   │   C.surprise += c.surprise;
10 │   │   end
11 │   end
12 │   if countOfCellsWithSurprise ≠ 0 then
13 │   │   C.surprise =
       │       C.surprise/countOfCellsWithSurprise;
14 │   else
15 │   │   C.surprise = null;
16 │   return C.surprise;
17 end
```

value per measure, and per cell, (b) aggregate the measures' surprise per cell, and (c) aggregate the different cell surprises to compute the surprise of the set of cells. As a concrete example, Algorithm 4 works on a single-measured cube, with absolute distance as the distance function to assess how far the actual and the expected measures are, and averaging over all cells with surprise to produce the aggregate cube surprise. The complexity of the algorithm is linear with respect to the result size for the query, assuming a fixed set of expected values $E$.

# 8 EXPERIMENTAL EVALUATION

In this Section, we present the experimental result for the evaluation of several algorithms working over different dimensions of interestingness. We measure efficiency in terms of time performance for the execution of the interestingness assessment algorithms, under different conditions of scale.

## 8.1 Experimentation methodology

The experiments were performed on the Loan cube of the pkdd99_star database, for which, we artificially generated data of different sizes. The contents of the cube were generated with a dedicated random generator. The server for the experiments came with an AMD Ryzen 9 5900HS 3.3GHz CPU processor, 16GB of RAM and a 1TB SSD NVMe M2 hard drive. For all experiments, 8GB of RAM was allocated to the MYSQL server, via Workbench 8.0 CE. *The experimental goal is to assess the efficiency of the algorithms, via their execution time, by tuning the scale of two parameters of the problem, fact table and history size.*

## 8.2 Novelty

**Partial Detailed Extensional Novelty**. In this experiment, we study the effect of the fact table size and the query history to the
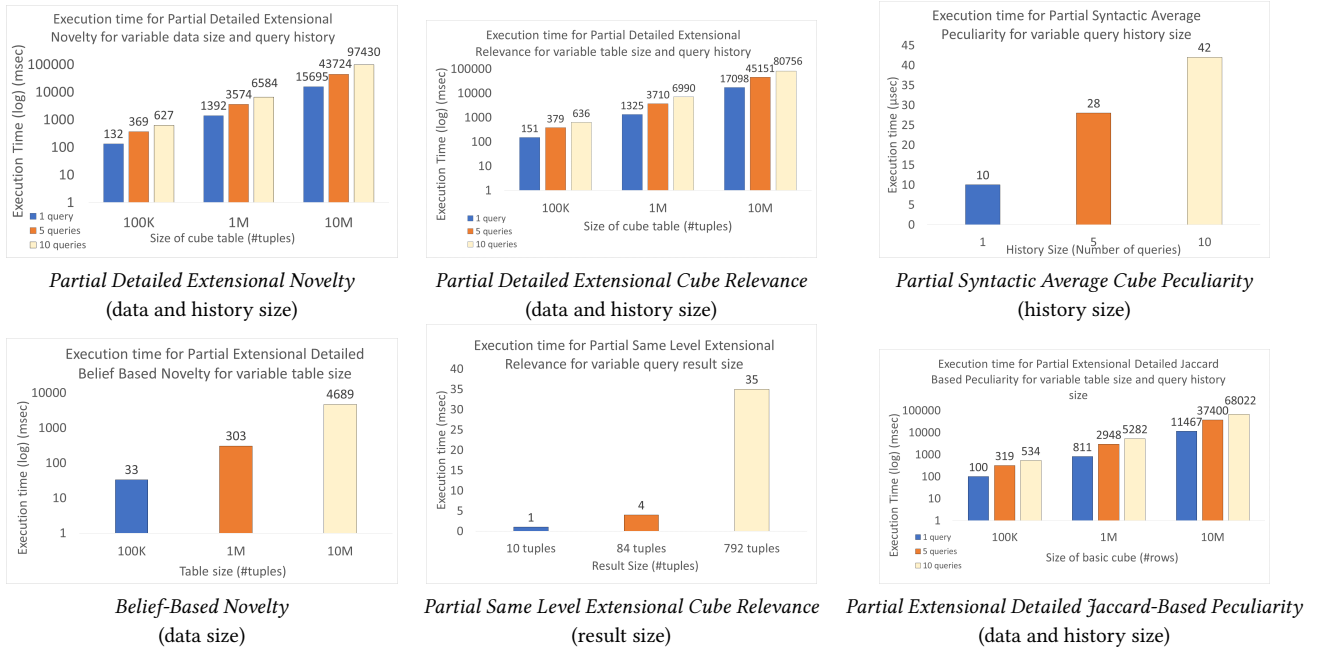
Figure 1: Execution time for (left) Novelty, (middle) Relevance, (right) Peculiarity algorithms

execution time of the Algorithm for the *Partial Detailed Extensional Novelty*. We have limited ourselves to table sizes of 100K, 1M and 10M tuples and query history of 1, 5 and 10 queries. As Figure 1 shows, both the increase of the table size and the size of query history, increase the total execution time of the algorithm. The vertical axis is logarithmic. Both the increase of the table size and query history size cause a linear increase in the total execution time.

**Belief Based Novelty**. In this experiment, we study the effect of the fact table size to the execution time of the Algorithm for the *Partial Extensional Detailed Belief-Based Novelty*. We have experimented with table sizes of 100K, 1M and 10M tuples. Fig. 1 demonstrates the results. Bear in mind that the vertical axis is logarithmic and observe that the execution time increases linearly with data size increase, a behavior that agrees with the complexity analysis of the algorithm.

**Comparison**. Evidently, *Partial Extensional Detailed Belief Based Novelty* is faster than *Partial Detailed Extensional Novelty*, due to the fact that the latter is based on the time-consuming procedure of calculating the detailed area of interest of all the queries participating in the query history and comparing them to the one of the given query. This requires additional queries to the database, while on the other hand, *Partial Extensional Detailed Belief-Based Novelty* simply decides if a detailed cell of the result is considered novel based on a set of user's beliefs.

## 8.3 Relevance

**Partial Detailed Extensional Cube Relevance**. In this experiment, we study the effect of the increase of the fact table size for a query history of 1, 5 and 10 queries to the Algorithm for the *Partial Detailed Extensional Cube Relevance*, which is practically assessing relevance with respect to a Detailed Area of Interest. We have experimented with 100K, 1M and 10M table sizes and query history of 1, 5 and 10 queries. As Figure 1 shows, increasing either the table size or the query history size results in an

increase of the execution time of the algorithm. Both experiments for the table and the history increase, agree with the complexity analysis of the algorithm, which presented that the algorithm is depended linearly on the query history size and the table size.

**Partial Same Level Extensional Cube Relevance**. In this experiment, we study the behavior of this goal-based algorithm's execution time when we increase the result size of a query, in terms of number of tuples. Specifically, we limit ourselves to result sizes of 10, 84 and 792 tuples respectively. Observe that even though the algorithm is relatively fast, the increase of the result size increases linearly the execution time of the algorithm.

**Comparison**. *Partial Same Level Extensional Cube Relevance* is a much faster algorithm than *Partial Detailed Extensional Cube Relevance*, due to the fact that the latter one is calculating the detailed area of interest for all the history queries, while the first algorithm simply calculates the coverage of the detailed cells based on the user's goal.

## 8.4 Peculiarity

**Partial Syntactic Average Cube Peculiarity**. In this experiment, we study the behavior of this algorithm's execution time when we increase the number of queries used as a query history. The results in Fig. 1, show that the increase of the query history size linearly increases the total execution time of the algorithm, as presented in the complexity analysis of the algorithm too.

**Partial Extensional Detailed Jaccard-Based Peculiarity**. In this experiment, we study the effect of the increase of (a) the fact table size, and (b) and the query history to the Partial Extensional Detailed Jaccard-Based Peculiarity algorithm (practically assessing peculiarity on the basis of a Jaccard similarity between the detailed areas of the query and the history of queries). Fig. 1, with its vertical axis in logarithmic scale, shows the results of the experiments. Both table size and query history size increase linearly the execution time of the algorithm, but the first one in a much larger scale.
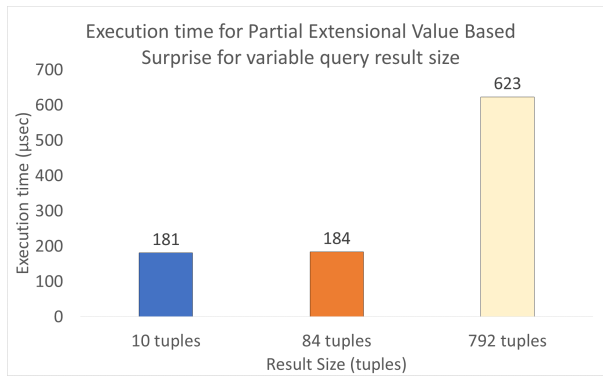
**Figure 2: The execution time for the *Partial Extensional Value-Based Surprise* with respect to query result size**

**Comparison**. *Partial Syntactic Average Cube Peculiarity* is a much faster algorithm than *Partial Extensional Detailed Jaccard-Based Peculiarity*. This is due to the fact that the first one simply does a syntactic analysis of the query and compares it to the already submitted ones, while the latter one needs to compute the detailed area of interest of all the queries in the history, which hides the execution of a series of new queries.

## 8.5 Surprise

**Partial Extensional Value-Based Surprise**. In this experiment, we study the behavior of this algorithm's execution time when we increase the result size of a query, in terms of number of tuples. The results, as presented in Fig. 2, show that the theoretical linear increase with respect to the result size is not exactly achieved. The algorithm is quite fast, of course, due to its simple nature that works on top of a query result (remember, surprise cannot work with signatures, and requires the query result and the cells measures to be computed). We attribute the variation of the execution time to the probability of hitting an expected value when the result size of the query is larger, which results in extra CPU time for computing the surprise.

## 9 CONCLUSIONS

In this paper, we have addressed the problem of assessing the interestingness of a cube query in the context of a hierarchical multidimensional database with cubes and level hierarchies. We have focused the discussion on 4 interestingness dimensions, specifically, relevance, surprise, novelty, and peculiarity. For these dimensions of interestingness, we have also proposed specific measures and algorithms for assessing them in a quantitative fashion. We take care to discriminate between result-based algorithms, after the query has been executed and signature-based algorithms, before the query is executed. We have also explored the runtime behavior of such algorithms, over different sizes and session histories.

Future work can continue in different roads. First, it is clear that the presented algorithms are only a first attack to the problem. Several parts of the domain of solutions are also possible (see [11] for a discussion). Moreover, while our four interestingness dimensions cover the majority of existing definitions of interestingness, a notable one concerns the *expression* aspect, in which data are assessed for their fitness to the medium used to express them – e.g., a cube can be described by the set of cells, or by a query, or by a visualization, etc.

## REFERENCES

[1] Firas Abuzaid, Peter Kraft, Sahaana Suri, Edward Gan, Eric Xu, Atul Shenoy, Asvin Ananthanarayan, John Sheu, Erik Meijer, Xi Wu, Jeffrey F. Naughton, Peter Bailis, and Matei Zaharia. 2021. DIFF: a relational interface for large-scale data explanation. *VLDB J.* 30, 1 (2021), 45–70.

[2] Julien Aligon, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Elisa Turricchia. 2014. Similarity measures for OLAP sessions. *Knowl. And Inf. Syst.* 39, 2 (2014), 463–489.

[3] Eftychia Baikousi, Georgios Rogkakos, and Panos Vassiliadis. 2011. Similarity measures for multidimensional data. In *Proceedings of ICDE.* 171–182.

[4] Tijl De Bie. 2013. Subjective Interestingness in Exploratory Data Mining. In *Proceedings of IDA.* 19–31.

[5] Alexandre Chanson, Nicolas Labroche, Patrick Marcel, Stefano Rizzi, and Vincent T'kindt. 2022. Automatic generation of comparison notebooks for interactive data exploration. In *EDBT.* OpenProceedings.org, 2:274–2:284.

[6] Rui Ding, Shi Han, Yong Xu, Haidong Zhang, and Dongmei Zhang. 2019. QuickInsights: Quick and Automatic Discovery of Insights from Multi-Dimensional Data. In *Proceedings of SIGMOD.* Amsterdam, The Netherlands, 317–332. https://doi.org/10.1145/3299869.3314037

[7] Ori Bar El, Tova Milo, and Amit Somech. 2020. Automatically Generating Data Exploration Sessions Using Deep Reinforcement Learning. In *Proceedings of SIGMOD.* Portland, OR, USA, 1527–1537.

[8] Matteo Francia, Patrick Marcel, Verónika Peralta, and Stefano Rizzi. 2022. Enhancing Cubes with Models to Describe Multidimensional Data. *Inf. Syst. Frontiers* 24, 1 (2022), 31–48.

[9] Arnaud Giacometti, Patrick Marcel, and Elsa Negre. 2009. Recommending Multidimensional Queries. In *DaWaK (Lecture Notes in Computer Science, Vol. 5691).* Springer, 453–466.

[10] Dimitrios Gkesoulis, Panos Vassiliadis, and Petros Manousis. 2015. CineCubes: Aiding data workers gain insights from OLAP queries. *Inf. Syst.* 53 (2015), 60–86.

[11] Dimos Gkitsakis, Spyridon Kaloudis, Eirini Mouselli, Veronika Peralta, Patrick Marcel, and Panos Vassiliadis. 2022. Cube Interestingness: Novelty, Relevance, Peculiarity and Surprise. *arXiv* (2022). https://doi.org/10.48550/ARXIV.2212.03294

[12] Matteo Golfarelli and Elisa Turricchia. 2014. A characterization of hierarchical computable distance functions for data warehouse systems. *Decis. Support Syst.* 62 (2014), 144–157. https://doi.org/10.1016/j.dss.2014.03.011

[13] Pingchuan Ma, Rui Ding, Shi Han, and Dongmei Zhang. 2021. MetaInsight: Automatic Discovery of Structured Knowledge for Exploratory Data Analysis. In *Proceedings of SIGMOD.* 1262–1274. https://doi.org/10.1145/3448016.3457267

[14] Patrick Marcel, Verónika Peralta, and Panos Vassiliadis. 2019. A Framework for Learning Cell Interestingness from Cube Explorations. In *Proc. ADBIS 2019.* 425–440.

[15] Tova Milo and Amit Somech. 2020. Automating Exploratory Data Analysis via Machine Learning: An Overview. In *SIGMOD.*

[16] Aurélien Personnaz, Sihem Amer-Yahia, Laure Berti-Équille, Maximilian Fabricius, and Srividya Subramanian. 2021. DORA THE EXPLORER: Exploring Very Large Data With Interactive Deep Reinforcement Learning. In *CIKM.*

[17] Sunita Sarawagi. 2000. User-Adaptive Exploration of Multidimensional Data. In *Proceedings of VLDB.* 307–316.

[18] Bo Tang, Shi Han, Man Lung Yiu, Rui Ding, and Dongmei Zhang. 2017. Extracting Top-K Insights from Multi-dimensional Data. In *SIGMOD Conference.* ACM, 1509–1524.

[19] Panos Vassiliadis. 2022. A Cube Algebra with Comparative Operations: Containment, Overlap, Distance and Usability. *arXiv* (2022). https://doi.org/10.48550/arXiv.2203.09390

[20] Yun Wang, Zhida Sun, Haidong Zhang, Weiwei Cui, Ke Xu, Xiaojuan Ma, and Dongmei Zhang. 2020. DataShot: Automatic Generation of Fact Sheets from Tabular Data. *IEEE Trans. Vis. Comput. Graph.* 26, 1 (2020), 895–905.

[21] Emanuel Zgraggen, Zheguang Zhao, Robert C. Zeleznik, and Tim Kraska. 2018. Investigating the Effect of the Multiple Comparisons Problem in Visual Analysis. In *Proceedings of CHI.* Montreal, QC, Canada, 479.