Using Entropy Metrics for Pruning Very Large Graph Cubes

Dritan Bleco [1], Yannis Kotidis[2]

Department of Computer Science, Athens University of Economics and Business, 76 Patission Street, Athens GR 10434, Greece

Abstract

Emerging applications face the need to store and analyze interconnected data. Graph cubes permit multi-dimensional analysis of graph datasets based on attribute values available at the nodes and edges of these graphs. Like the data cube that contains an exponential number of aggregations, the graph cube results in an exponential number of aggregate graph cuboids. As a result, they are very hard to analyze. In this work, we first propose intuitive measures based on the information entropy in order to evaluate the rich information contained in the graph cube. We then introduce an efficient algorithm that suggests portions of a precomputed graph cube based on these measures. The proposed algorithm exploits novel entropy bounds that we derive between different levels of aggregation in the graph cube. Per these bounds we are able to prune large parts of the graph cube, saving costly entropy calculations that would be otherwise required. We experimentally validate our techniques on real and synthetic datasets and demonstrate the pruning power and efficiency of our proposed techniques.

Keywords: Graph Cube, Entropy, Big Data

1. Introduction

The data management community has long been interested in problems related to modeling, storing and querying graph databases [1, 2]. Recently, this

[1]dritanbleco@aueb.gr
[2]kotidis@aueb.gr

interest has been renewed with the emergence of applications in social networking, location based services, biology and the semantic web, where data graphs of massive scale need to be analyzed. As a consequence, business intelligence techniques such as the data cube [3], which have been developed for flat, relational data, need to be revised in order to accommodate the needs of complex graph datasets.

Of particular interest in graph data are the relationships between nodes depicted via the edges of the graph. These relationships should be analyzed with respect to attribute values available at the nodes and edges. For example, a data scientist may want to investigate how users of a social network, depending on their gender, relate to other users based on their nationality. As we will see this inquiry can be accommodated by aggregating existing relationships (edges) in the data graph based on gender and nationality attribute values of their constituent nodes. This process forms a *graph cuboid*, as is depicted in Figure 1.

Graph cubes have been recently proposed [4, 5, 6, 7, 8] in order to describe all possible such cuboids. They provide a solid foundation that an analyst may build upon, in a manner similar to what the data cube is for OLAP analysis [9, 10, 11, 12]. Nevertheless, graph cubes contain an exponential collection of cuboids. Moreover, a decision maker, familiar with the simpler framework of data cubes, may be overwhelmed when she tries to navigate not flat records, but rather complex graph cuboids containing aggregated views of graph nodes and relationships.

In this work, we first revisit the graph cube framework highlighting the relationships among the cuboids contained in it. These relationships are modeled as a graph cube lattice produced by taking the Cartesian product of simpler data cubes on the attributes of the nodes and edges of the data graph. We utilize the graph cube lattice as a foundation, where interesting relationships can be revealed using entropy-based calculations. A benefit of the lattice representation and of the metrics we propose is that certain entropy bounds can be established between the graph cuboids. This realization permits us to design an efficient algorithm that prunes significant parts of the graph cube from consideration.

Compared to a straightforward evaluation of the entropy metrics on the whole lattice, the proposed algorithm saves up to 90% of computation time.

Our major contributions are summarized as follows:

- We utilize two novel entropy measures, in particular external and internal entropy, as the means to evaluate the content of the graph cube. External entropy weighs a drill-down edge in the graph cube lattice in order to determine whether the addition of a new attribute that is used to form the more detailed child cuboid results in non-uniform interactions. In such cases, the internal entropy is used to examine each cuboid that is constituent to such a drill-down edge and select subgraphs that exhibit significant skew.

- A straightforward approach that would evaluate the proposed entropy metrics over the whole graph cube would be very inefficient. In this work, we propose a bisection algorithm which, given a precomputed graph cube, prunes whole cuboids from consideration by exploiting certain bounds between their entropies. Our results in real and synthetic datasets demonstrate the effectiveness of our techniques in processing multi-terabyte graph cubes with tens of billions of records. These results further reveal that often, only small parts of the graph cube contain interesting aggregations, with respect to other aggregations available in the graph cube lattice.

- We compare our techniques against an alternative method that prunes parts of the graph cube based on a minimum support threshold. We observe that our framework maintains the most varied parts of the data distribution independently of their frequencies.

1.1. Comparison to Prior Work

The work presented in this manuscript is a consolidation of prior work by the authors that has appeared in [13] and [14], extended further with new algorithmic results that significantly increase the efficiency of the proposed techniques. More specifically, the work of [13] first introduced the idea of using information entropy in order to prune graph cubes. However, this work offered a limited definition of entropy suitable only for the *SUM* aggregation function. This article models the graph data as a probability distribution and extends the definitions of the entropy metrics so as to be applicable to different functions and not only *SUM*. The work of [14] was based on [13] and proposed a graph cube analysis workflow for identifying and visualizing prominent trends in large graph cubes. However, neither [13] nor [14] provide an efficient algorithm for computing the proposed entropy metrics. As a result, their methods are inefficient when used in very large graph datasets.

In this work, as highlighted above, we first introduce novel theoretical bounds on the proposed entropy rates. Per these bounds, we are able to present a novel bisection algorithm that reduces the entropy computation times by up to 90%, compared to the straightforward evaluation discussed in our prior work. In our experimental results we evaluate the benefits of this new algorithm compared to the techniques discussed in [13, 14]. This experimental evaluation is performed using a new implementation over Spark that permits parallel execution of the entropy calculations in a distributed system. Moreover, we provide a more thorough evaluation of our methods that includes additional real and synthetic datasets. Finally, in this work we further discuss extensions to the entropy metrics and provide experimental results for datasets that contain attributes on both their nodes and edges.

1.2. Manuscript Organization

The rest of this paper is organized as follows: Section 2 uses a motivational example to present our data model and discusses the construction of the graph cube lattice from the constituent data cubes on the graph nodes. We then formally introduce in Section 3 the concepts of external and internal entropy and explain their calculations. Section 4 presents our proposed algorithm for entropy-aware selection of graph cuboids and discusses extensions to the proposed graph cube framework. In Section 5 we provide qualitative and quantitative indicators on the effectiveness of our techniques when used on real

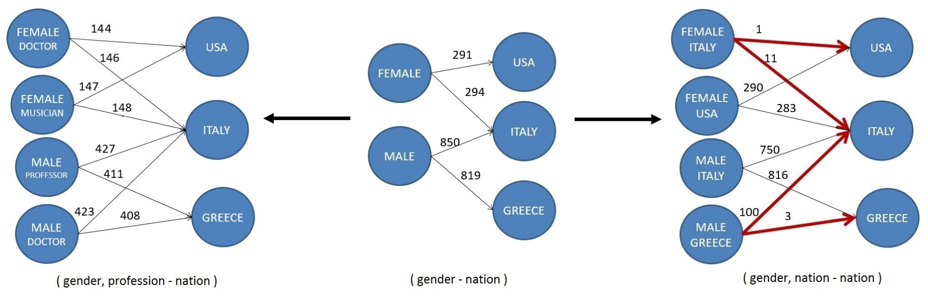


Figure 1: Three possible cuboids: (gender, profession - nation), (gender - nation) and (gender, nation - nation). Notice that the drill-down to the more fine-grained cuboid on the right reveals irregular associations, conditioned to what has been revealed by the cuboid in the middle. In contrast, the relationships contained on the (gender, profession-nation) cuboid seem to follow the same patterns as the original top-level cuboid.

and synthetic datasets. Finally, in Section 6 we discuss related work, while in Section 7 we provide concluding remarks.

2. Data Model

2.1. Motivational Example

We consider a social network which depicts relationships between different users. Each user profile can be represented as a graph node with three attributes: gender (male, female), nation (Greece, Italy, USA) and profession (doctor, professor, musician). Every edge in the data graph is associated with a numeric value that indicates the number of interactions between the respective users.

A possible inquiry on this network is to examine how users depending on their gender, relate to other users based on their nationality. To accommodate this query we need to perform three different aggregations. First, starting nodes (i.e. nodes with outgoing edges) are grouped into two aggregate nodes corresponding to gender values male and female, respectively. Similarly, three aggregate nodes corresponding to nations Greece, Italy and USA are formed. Finally, each edge of the network, depending on the gender attribute value of its starting node and the nation attribute value of its ending node, is aggregated into an edge between the corresponding aggregate nodes created at the previous steps. At this time, a desired aggregate function can be computed. In this example, we assume that this function is SUM(). The resulting aggregate graph is depicted in the middle of Figure 1. Based on its construction we refer to it as the (gender - nation) cuboid.

Continuing with the running example, the cuboid on the left part of the figure depicts the outcome of drilling-down from (gender - nation) to the (gender, profession - nation) cuboid. The intuition is that we would like to explore whether the profession of the source node, in addition to its gender, affects the number of observed relationships. In this contrived example, the aggregated edges from cuboid (gender - nation) are split almost evenly when drilling down to the (gender, profession - nation) cuboid. Thus, this particular navigation step does not seem to reveal interesting correlations for this data, conditioned on what is already observed in the (gender - nation) cuboid.

On the right part of Figure 1, we depict another possible drill-down, this time to the (gender, nation - nation) cuboid. In this new context, some interesting irregularities are revealed. First, while female users are linked evenly to users from USA and Italy, when these links are conditioned based on the nationality we can see that females from Italy are mainly linked to users from the same country. Similarly, Greek males are mostly linked to users from Italy. Thus, while cuboid (gender - nation) suggest a uniform relationship based on the nationality of the target node, cuboid (gender, nation - nation) reveals that this is not true for certain members of the user community. It is worth noting that the majority of the links in the (gender, nation - nation) cuboid still follow the same uniform pattern suggested by the (gender - nation) cuboid, since most links emanate from female users in USA and male users in Italy. Thus, the examples discussed above are exceptions to what is suggested by the (gender - nation) cuboid. These are depicted in bold red color inside the (gender, nation - nation) cuboid.

2.2. The Graph Cube

We assume that our dataset is depicted by a directed graph. The nodes and edges of the graph may have several attributes associated with them. In the data warehouse literature the grouping attributes used in the analysis are called dimensions and the result-set of a particular grouping on the selected dimensions is called cuboid. For n attributes there are 2^n different groupings or cuboids. All these cuboids form the data cube [3]. In our running example, if we only concentrate for the moment on the graph nodes, the resulting data cube has three dimensions: gender (G), nation (N) and profession (P) and 2^3=8 possible cuboids.

The data cube framework is extended to work on graph data by considering also, the relationships between aggregated graph nodes [8]. In particular, the graph cube is the Cartesian product of two cubes: of the starting- and the ending-cube, as is depicted in Figure 2. In this example, a graph cuboid can be ((gender, *,*) - (*,nation,*)) or, for brevity, (gender - nation). The starting nodes on this cuboid are aggregated graph nodes based on the gender attribute. Similarly, the ending nodes are aggregations of raw graph nodes based on the nation attribute. Starting and ending nodes in this cuboid are interconnected according to the raw graph edges. These raw data edges are consolidated producing a graph cube edge along with a measure. The user can choose any combination of functions based on measure attributes on the constituent nodes and edges. For instance, in a different example, we may use the age attribute of users that are connected in a social network in order to compute in a data edge the absolute age difference of their relationship and aggregate these values using the AVG() or MAX() functions in the graph cube. In our running example, for simplicity, we assume that the computed function is the SUM() function over a single numerical measure on the edges.

Since the graph cube is a Cartesian product of two data cubes that form lattices containing their corresponding cuboids, it may also represented as a lattice constructed by the associated graph cuboids. As a result, the graph cube builds on the solid foundation of multidimensional modeling guaranteeing

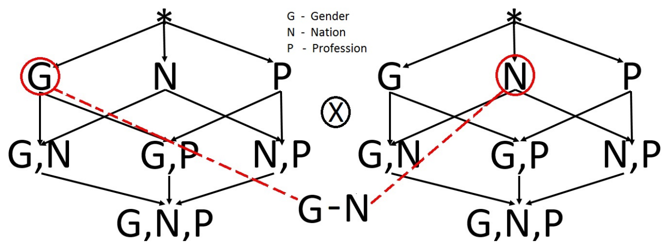


Figure 2: The Graph Cube

correct summarisability in decision support applications [15, 16]. The full graph cube lattice for one of the real datasets used in our experiments is depicted in Figure 3.

Each cuboid in the graph cube is a graph but it may also be represented (virtually) as a relation the schema of which contains the attributes of the starting and ending nodes in the cuboid as well as the computed aggregate. We refer to this relation as the cuboid dual relation.

Definition 1. *Cuboid dual relation*

Let's assume a cuboid C from the graph cube that consists of s_1, s_2,..., s_t starting attributes, $e_1, e_2, \ldots, e_w$ ending attributes and a numerical attribute a. The cuboid dual representation is a relation DC with schema

$$DC(s_1, s_2, \ldots, s_t, e_1, e_2, \ldots, e_w, a)$$

Each edge from the cuboid graph is mapped to a single record whose attribute values are determined by the attributes of the constituent nodes and the value of attribute a is the aggregate value associated with the edge, depending on the selected aggregation function.

In Table 1, we depict the content of the cuboid (gender - nation) dual relation. Notice that the number of records in the relation is equal to the number of edges in the cuboid.

3. Using Entropy to Navigate the Graph Cube

$gender_s$	$nation_e$	a	p(a)
female	USA	291	12.9%
female	Italy	294	13.0%
male	Italy	850	37.7%
make	Greece	819	36.4%

Table 1: Dual relation of cuboid (gender - nation) from Figure 1. The last column is not part of the relation but denotes the probability value associated with each record, computed by normalizing the value of aggregate a (i.e. by dividing with $SUM(a)$ computed over all records), as in [17].This process is independent of the aggregate function used to compute a.

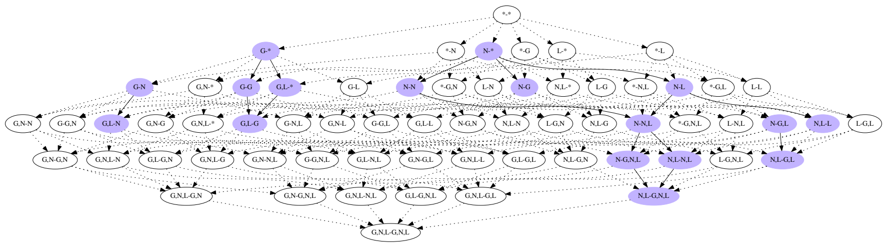


Figure 3: Selected drill-down edges and corresponding cuboids using external entropy rates, Twitter dataset

3.1. Main concepts

Almost always, the analysts are attracted to data that are far away from uniformity; data from which they can discover patterns and rules; data that are hidden in peaks and valleys. In order to explore such cases of data skew, we revisit the idea of the information entropy (or Shannon entropy [18]), which is the expected value of the information contained in the data, and transform it in a manner that is suitable for processing graph cuboids. The entropy captures the amount of uncertainty; it increases when the data is closer to random or there is noticeable uniformity and decreases when the data are less random or there are high peaks. In our model, we look for skewed data distributions within and across cuboids, in order to steer the user towards interesting parts of the

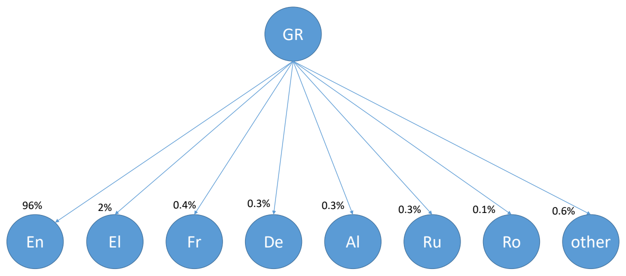


Figure 4: Graph cube slice for the (nation - language) cuboid when the starting node nation value is "Greece". We observe that users from Greece communicate mostly with English-speaking users in the Twitter dataset.

graph cube lattice. As a consequence, data in the graph cube that depict nearly uniform behavior can be overlooked during exploratory analysis. Our intuition is that the remaining parts are characterized by a high degree of disorder, and it is exactly this disorder that an analyst seeks to explore and exploit.

In what follows we introduce two entropy metrics, which we use in our work. The *external entropy* and the *internal entropy*. The first one that we also refer to as *cuboid entropy* captures the entropy of a cuboid as a unit. We use this metric in order to decide whether a cuboid provides interesting information with respect to other cuboids in the lattice (as in a drill-down or roll-up operation). The external entropy helps the user navigates the graph cube lattice and may be used to prune a significant portion of the lattice from consideration.

Figure 3 highlights the pruning power of external entropy on a real dataset crawled from Twitter. This dataset is discussed and analyzed in more detail in the experimental section. There are three attributes on the nodes (user profiles) of this data graph, namely gender (G), nation (N) and language (L), resulting in 64 cuboids depicted in the figure. The solid edges in the lattice denote drill-down operations that are suggested by our techniques based on the external entropy of the constituent cuboids, depicted in dark color. A drill-down operation allows the user to navigate the graph-cube lattice by adding a new attribute in her selection. In the figure we observe that the drill-down from cuboid (nation - *) to (nation - language) is selected by our framework. This implies that by grouping the ending nodes of the former cuboid by their language attribute values we observe non-uniform interactions. This is because, in most nations, users mainly use the English language to interact. Thus, a drill-down step at this part of the lattice will help the analyst focus on this behavior.

Each cuboid that is selected based on these external entropy calculations is a complex graph that can be quite large. The *internal entropy* helps the decision worker navigate inside a large cuboid and identify non-uniform interactions. In our running example, the internal entropy may be utilized in order to pick portions of the (nation - language) cuboid that are characterized by very skewed interactions. For example, if we slice the (nation - language) cuboid using Greece as a starting node, then Figure 4 depicts the distribution of ending nodes based on their language attribute values. Is it clear that the vast majority of the Greek users (96%) communicate with English speaking users in this social network.

3.2. External Entropy

Let us consider a cuboid C_i from the graph cube and its dual relation DC_i, as defined in the previous section. Each record in the dual relation is associated with an aggregate value a that denotes the result of the aggregate function applied over the measure of the corresponding data edges. As suggested by [17] each record $(s_1, \dots s_t, e_1, \dots e_w, a)$ of DC_i can be viewed as a discrete probability distribution $P(s_1, \dots s_t, e_1, \dots e_w)$ by normalizing the aggregate a value on each record by the sum of all aggregate values in the instance of the relation. In Table 1 we demonstrate this process on the (gender - nation) cuboid of our running example. Thus, each record r_j of DC_i is associated with a probability value $p(a_j) = \frac{a_j}{\sum_{i=1}^{m}(a_i)}$, as shown in the table (m refers to the number of records in DC_i).

We define the external entropy (eH) of a cuboid as the negative of the logarithm of the probability distribution of the cuboid records in its dual relation.

Thus, the external entropy of C_i is calculated as

$$eH(C_i) = -\sum_{j=1}^{m} p(a_j) * \log_2 p(a_j) \tag{1}$$

Based on this formula, one can easily find the maximum and minimum values that the external entropy can reach. The minimum value is when we have a cuboid consisting of a single edge resulting in a dual relation that contains only one record. In this case $eH_{min}(C_i) = -p(a_1) * log_2 p(a_1) = 0$, where $p(a_1) = 1$. The maximum entropy value is obtained when all m records in DC_i have the same probability $p(a_i)=\frac{1}{m}$. In that case $eH_{max}(C_i) = -\log_2(\frac{1}{m})$.

Recall that each cuboid has a certain selection of starting and ending attributes. If we add another attribute (starting or ending) in the cuboid C_i then we get another cuboid C_k of the lattice. We refer to cuboid C_k as the "child" of C_i, while C_i is the "parent" of C_k.

Let us consider the relationship between the external entropies of these cuboids. Drilling down from the parent C_i to the child C_k we can calculate the delta-entropy, i.e. the difference between the two external entropies as:

$$\delta_{(C_k,C_i)} = eH(C_k) - eH(C_i) \tag{2}$$

This difference equals to the conditional entropy of the child given the parent. Each record r_j from the dual relation DC_i of C_i by the drill down process results in one or more detailed records in the dual relation of DC_k. For example, record (female, USA, 291) from the dual relation of the (gender - nation) cuboid of Figure 1 results in two more detailed records, namely (female, Italy, USA, 1) and (female, USA, USA, 290) in the dual relation of cuboid (gender, nation - nation). Thus, the conditional entropy of the child given the parent can be computed as

$$\delta_{(C_k,C_i)} = \sum_{j=1}^{m} p(a_j^i) * eH(C_k|C_i = a_j^i)$$

$$= -\sum_{j=1}^{m} \{p(a_j^i) * \sum_{o=1}^{d_j} p(a_o^k|a_j^i) * \log_2 p(a_o^k|a_j^i)\} \quad (3)$$

where $p(a_o^k|a_j^i) = \frac{a_o^k}{a_j^i}$, r_o^k from DC_k is a more detailed record of r_j^i from DC_i and there are d_j such records in DC_k.

The delta entropy is a non-negative number. This is because the external entropy of the child cuboid C_k is greater or equal to the external entropy of its parent C_i. For the minimum and the maximum entropy of C_k it holds that

$$0 \leq eH_{min}(C_i) = eH_{min}(C_k) \leq eH_{max}(C_i) \leq eH_{max}(C_k) \quad (4)$$

The minimum entropy value of the child equals to the entropy of its parent, i.e. when the delta entropy $\delta_{(C_k,C_i)} = 0$. In this case, each record of the dual relation DC_i results in a single more detailed record in DC_k.

The maximum external entropy of the child is obtained when the aggregate a of each record of DC_i is distributed evenly among the more detailed records of DC_k and their number is maximized. Let d_{max} denote the number of possible values of the attribute on which the drill down process was performed. In order to maximize the entropy of a child cuboid, a record r_j^i with aggregate value a_j^i in DC_i is replaced during the drill-down with d_{max} more detailed records r_o^k ($o \in [1..d_{max}]$) in DC_k with aggregate values $a_o^k{=}\frac{a_j^i}{d_{max}}$. Thus, the maximum possible external entropy value of the child cuboid given its parent is

$$eH_{max}^i(C_k) = -\sum_{j=1}^{m} p(a_j^i) * \log_2 \frac{p(a_j^i)}{d_{max}} \quad (5)$$

Based on these observations, we introduce the *external entropy rate* in order to quantify how informative, the process of drilling down from parent C_i to its child C_k is. We define the external entropy rate as

$$eH_{rate}(C_k, C_i) = \frac{eH(C_k) - eH(C_i)}{eH_{max}^i(C_k) - eH(C_i)} \quad (6)$$

Where $0 \leq eH_{rate}(C_k, C_i) \leq 1$. When this value is close to 1, the drill-down process doesn't change significantly the distribution of the records and, thus, no new insights are given to the analyst. We can therefore exclude less interesting navigations in the lattice by defining an external entropy rate threshold value between zero and one. When the eH_{rate} of a drill down surpasses the threshold, then this drill down is omitted from consideration, as in Figure 3.

3.3. Internal Entropy

In order to gain insight into the distribution of records within a cuboid, we introduce an additional type of entropy termed internal entropy. Due to the fact that we consider directed data graphs, we distinguish between two kinds of internal entropies namely starting internal entropy and ending internal entropy.

Consider cuboid C_i with s starting attributes and t ending attributes in its dual relation DC_i. Assume there are l distinct combinations of starting attribute values of the form $(s_1^y, s_2^y, \ldots, s_s^y)$ in DC_i and m_y is the sum of the aggregate values of all such records, where $y \in [1, l]$. For each such combination (indicated by parameter y) there are f_y records in DC_i with different combinations of ending attribute values. Let z_{q_y} be sum of their aggregate values as well. We calculate the starting internal entropy as the conditional entropy of the ending attributes' values conditioned from each starting attribute combination of values. Thus, for the combination of starting attribute values indicated by y, we define the starting internal entropy as

$$siH(C_i^y) = -\sum_{j=1}^{f_y} p(q_j^y) * \log_2 p(q_j^y) \quad where\, p(q_j^y) = \frac{z_{q_y}}{m_y} \tag{7}$$

The ending internal entropy eiH is defined in an analogous manner. As in the case of external entropy, we introduce the internal entropy rate (for the starting or ending internal entropy, respectively) as the fraction between the (starting/ending) internal entropy and the maximum possible value of internal entropy. For example, the starting internal entropy rate is defined as

$$siH_{rate}(C_i^y) = \frac{siH(C_i^y)}{siH_{max}(C_i^y)} \tag{8}$$

The value of the internal entropy rate is between 0 and 1 and can be used to select the most prominent trends within a cuboid, as will be explained in the next Section.

4. Entropy-guided Selection on Graph Cubes

4.1. Problem Statement

In our framework, we seek to utilize the proposed entropy metrics in order select parts of the graph cube that satisfy the following objectives:

External-entropy Objective: Given a graph cube lattice GCL and an external entropy rate threshold eH_r return a set of drill-down navigations $navGCL = \{e = (C_k, C_i) | eH_{rate}(C_k, C_i) \leq eH_r\}$.

Internal Entropy Objective: Given a cuboid C_i and an internal entropy rate threshold iH_r, return all edges in C_i whose starting or ending internal entropy rates are less or equal to iH_r.

As we will show in this section, there are certain bounds that we can derive in the entropy calculations that enable us to skip whole cuboids from consideration, avoiding this ways computation of their entropy. This is the topic of the next subsection.

4.2. Pruning Entropy Calculations

We will omit computing certain cuboids by utilizing the following observations. Consider three cuboids C_i, C_k and C_g where C_i is the parent of C_k and C_k the parent of C_g. Assume that the external entropies for C_i and C_g have already been computed. We also note that while calculating the external entropy of a cuboid we can also compute, at the same time the maximum entropy of its children, as this computation does not require accessing a child cuboid's data

(Equation 5). Based on the monotonicity of the entropy values we know that $eH(C_i) \leq eH(C_k) \leq eH(C_g)$ and $eH^i_{max}(C_k) \leq eH^k_{max}(C_g)$. When computing the external entropy rate for edge (C_k, C_i) the only missing value is that of $eH(C_k)$ that takes values in the range $[eH(C_i), eH(C_g)]$. Thus, $eH_{rate}(C_k, C_i)$ ranges in $[0, \frac{eH(C_g)-eH(C_i)}{eH^i_{max}(C_k)-eH(C_i)}]$. If the upper bound $\frac{eH(C_g)-eH(C_i)}{eH^i_{max}(C_k)-eH(C_i)}$ is less or equal to eH_r, the edge would be added to the result of the computation without calculating C_k and its external entropy.

For the rate of edge (C_g, C_k) it holds that

$$eH_{rate}(C_g, C_k) = \frac{eH(C_g) - eH(C_k)}{eH^k_{max}(C_g) - eH(C_k)} \leq \frac{eH(C_g) - eH(C_i)}{eH^i_{max}(C_k) - eH(C_i)} \tag{9}$$

since $eH^i_{max}(C_k) \leq eH^k_{max}(C_g)$ and $eH(C_i) \leq eH(C_k)$.

If $\frac{eH(C_g)-eH(C_i)}{eH^i_{max}(C_k)-eH(C_i)} \leq eH_r$ both edges (C_k, C_i) and (C_g, C_k) can be added to the result without further calculations. With similar arguments, this bound holds when C_i is an ancestor of C_k and C_g is a descendant of C_k in the lattice. In this case, Equation 9 helps us bound the external entropy rates of all edges in a path from cuboid C_i to cuboid C_g.

Based on this property we introduce in Figure 5 a recursive algorithm that performs a binary search type of traversal across the levels of the lattice and seeks paths whose constituent edges can be added immediately to the result set $navGCL$, omitting this way the computation of the external entropy of cuboids that are internal nodes in these paths. The algorithm takes as input the graph cube lattice, an external rate threshold, and the upper and lower level s and m of the lattice. In the initial invocation of the algorithm $s = 0$, indicating the top level of the lattice that consists of the $(* - *)$ cuboid and m is the lowest level that contains the cuboid that includes all attributes for the starting and ending nodes in the graph. The algorithm considers all pairs of cuboids $(C_g(m), C_i(s))$, where cuboid $C_g(m)$ is from level m and $C_i(s)$ is from level s. If this is the first time one of these cuboids is considered we compute its external entropy and the maximum entropy of its children (lines 5-10). We then check whether the upper bound from Equation 9 holds and if this the case, all edges from paths that

Input: $grapCubeLattice : GCL, threshold : eH_r, Ascedant_{Level} : s, Descendant_{Level} : m$

```
1:  procedure pruneLattice($GCL, eH_r, s, m$)
2:      $navGCL \leftarrow \emptyset$
3:      for each $Cuboids(C_g(m), C_i(s))$ do
4:          if $Path(C_g(m), C_i(s)).exists$
              $AND\ Path(C_g(m), C_i(s)).notVisited$ then
5:              if $C_i(s).notVisited$ then
6:                  $(eH(C_i(s)), eH^i_{max}[]) \leftarrow$ findEh($C_i(s)$)
7:              end if
8:              if $C_g(m).notVisited$ then
9:                  $(eH(C_g(m)), eH^g_{max}[]) \leftarrow$ findEh($C_g(m)$)
10:             end if
11:             $eH_{rate}(C_g(m), C_i(s)) \leftarrow min(\frac{eH(C_g(m)) - eH(C_i(s))}{eH^i_{max}(s)[k] - eH(C_i(s))} | k : C_k \in C_i(s).children\ AND\ C_k \in Path(C_g(m), C_i(s)))$
12:             if $eH_{rate}(C_g(m), C_i(s)) \leq eH_r$ then
13:                 for each $path \leftarrow Paths(C_g(m), C_i(s))$ do
14:                     $navGCL.add(edges_{path})$
15:                 end for
16:                 else
17:                 if $m > s + 1$ then
18:                     $navGCL$.add(
19:                     pruneLattice($GCL$,$eH_r$,$s$,$\lfloor \frac{m+s}{2} \rfloor$)
20:                     )
21:                     $navGL$.add(
22:                     pruneLattice($GCL$,$eH_r$,$\lfloor \frac{m+s}{2} \rfloor$, m)
23:                     )
24:                 end if
25:             end if
26:         end if
27:     end for
28:     return $navGCL$
29: end procedure
```

Figure 5: A bisection algorithm for selecting edges from the graph cube lattice.

connect these cuboids are added to the result set, without further calculations (lines 11-15). In order to obtain the tightest bound, we iterate over all children of $C_i(s)$ that are in the path to cuboid $C_g(m)$. If the test fails, a recursive calculation is triggered between levels s, $\lfloor\frac{(m+s)}{2}\rfloor$ and $\lfloor\frac{m+s}{2}\rfloor$, m (lines 17-24). We note that the selection process is over the edges of the lattice. This means that it is quite possible that a particular cuboid is not suggested (i.e. none of its adjacent edges is in the result set), while some of its descendant cuboids may as well be. This is also evident in Figure 3.

We note that the algorithm of Figure 5 operates over the lattice space requiring, in all of our experiments, less than 1 sec for selecting the appropriate edges (excluding the entropy calculations that would be computed anyway by a brute-force approach).

4.3. Applying our techniques on Data Cubes

The presented techniques also work for the case of regular (non-graph) data cubes by modeling its cuboids (aggregate relations) as probability distributions as well. Thus, the proposed external entropy rate can be used to reduce the number of aggregations (cuboids) that the analyst should consider in a OLAP data cube. Of course, in such cases, the internal entropy calculations are not applicable, unless there is an application-specific way to distinguish between "starting" and "ending" attributes in the flat records.

4.4. Aggregation on Edge Attributes

In many applications, edges of the data graph may have attributes that can be used in the analysis. Attributes on the edges of the data graph can be aggregated creating a set of cuboids or an edge-cube lattice. For example, in a social network a connection can have some attributes like the type T of the relationship (family, friend, sibling etc.) and the date D that this connection was established. The edge-cube lattice in this example contains four cuboids, namely (*), (T), (D), and (D,T). These cuboids can participate in the Cartesian product of our graph cube adding another dimension in the final cube. A cuboid in this

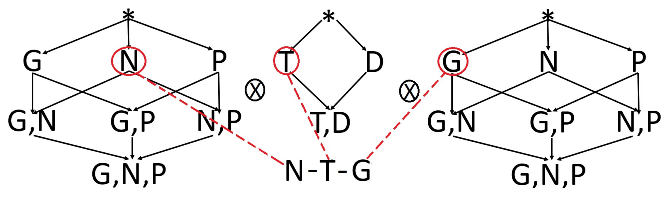


Figure 6: The graph cube with edge attributes

extended cube is described as (starting node-aggregation - edge-aggregation - ending-node-aggregation), for example cuboid (N - T - G) as shown in Figure 6.

Using attributes available at the edges of the data graph does not significantly alter the way the external entropy is calculated. Those attributes are considered during the calculation of external entropy as another dimension of the cuboid. Consequently, when looking within a cuboid, we can define the internal edge entropy in an analogous manner as the starting/ending internal entropies. We calculate the internal edge entropy as the conditional entropy of the node (starting and ending) attributes' values conditioned from each edge attribute combination of values. Similarly, we can use the internal edge entropy rate that takes values between 0 and 1 in order to focus on very skewed subgraphs, with respect to the edge attribute values selected in the analysis.

5. Experiments

5.1. Experimental Set Up

In this section, we provide an experimental evaluation of the proposed framework.[3] We first present results using four real datasets. We then discuss additional experiments using synthetic datasets in Section 5.4. The first real dataset consists of data sampled from Twitter. The second dataset is from VKontakte (VK), the largest European on-line social networking service. The third dataset

[3]The code and some sample data from the presented experiments are available at https://github.com/dritanbleco/GraphCubeFilteringUsingInformationEntropy.

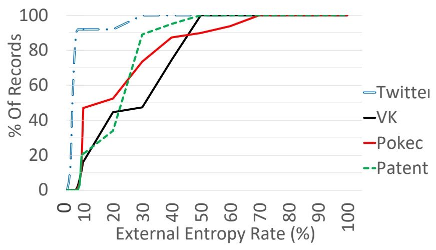
is from Pokec, the most popular on-line social network in Slovakia. The first two datasets were crawled by our team while the Pokec dataset is available at [19]. The last dataset contains citation information from U.S. patents [20].

	Twitter	VK	Pokec	Patent
Profiles (nodes)	34M	3.9M	1.6M	27.5M
Relations (edges)	910M	493M	31M	16.5M
Number of Attributes	3	5	6	3
Number of Cuboids	64	1024	4096	64
Graph Cube Records	4M	362M	66.3B	4.4M
Graph Cube Size	143MB	235GB	1.58TB	132MB
Graph Cube Computation Time	8 mins	87 mins	232 mins	7.5 mins
Cluster CPUs	30×4 Cores			
Cluster RAM	30×8 GB			
Cluster Storage	3 TB			

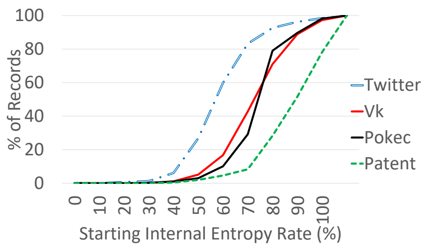
Table 2: Description of real datasets and hardware (VMs) used

Table 2 provides details on these datasets. The Twitter dataset contains 3 attributes on the nodes (profiles): the gender, location and language used from the profile. The VK dataset contains 5 attributes: birthyear, country, city, gender and education level of the user. The Pokec dataset uses 6 node attributes: age, region, gender, registration year, public profile and completion percentage of the profile. Finally, the Patent dataset uses 3 node attributes: grant year, country of first inventor and technological category.

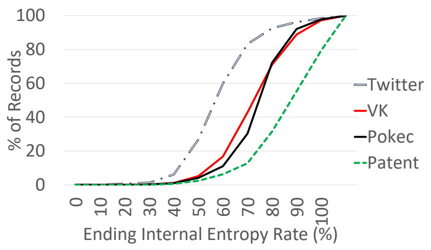
All presented experiments were run in a cluster of 30 VMs with 4 low spec cores each on the Cyclades cloud service for the Greek Research and Academic Community. All calculations were performed using the popular Apache Spark [21] framework. In order to compute the graph cube for each dataset we used a adaptation of the BUC algorithm for graph cubes, described in [13].



(a) Scaling external entropy rate



(b) Scaling starting internal entropy rate

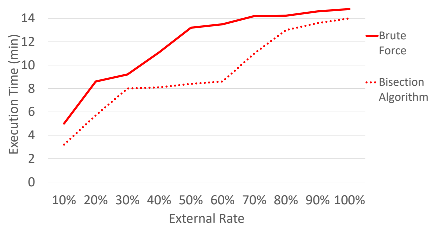


(c) Scaling ending internal entropy rate

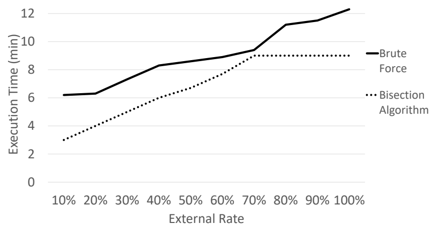
Figure 7: Records remaining in the graph cube using proposed entropy rates

5.2. *External and Internal Entropy Statistics*

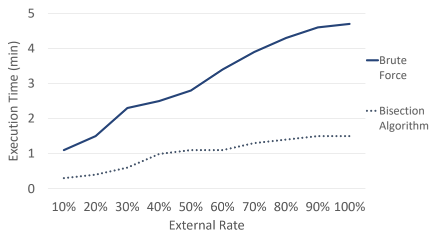
In these experiments we provide evidence on the pruning power of the proposed entropy metrics. In Figure 7a we plot the percentage of records that are retained in the graph cube (y-axis) for all the datasets when we vary the threshold for the external entropy rate (x-axis). The absolute sizes of the corresponding graph cubes are presented in Table 2. The figure reveals a steep reduction in the graph cube sizes, when we decrease this threshold below a certain value. For the Twitter dataset only 14% of the cube remains for a threshold rate of 3.5%. Moving up this threshold to 4%, the percentage jumps to 50% of the Twitter graph cube. This suggests that there is skew in the distribution of values across cuboids that we can investigate further using the internal entropy rates (discussed next). On the other hand, an increase of the external entropy rate threshold beyond 4% overwhelms the user with a significant increase in the result set, as many near-uniform relationships are retained complicating further analysis. Similar observations hold for the other three datasets.



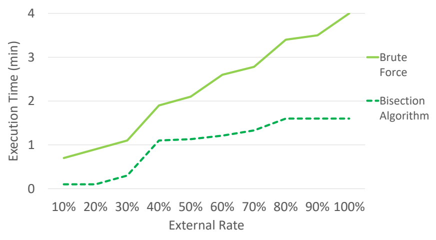
(a) Pokec Dataset



(b) VK Dataset



(c) Twitter Dataset



(d) Patent Dataset

Figure 8: Running times varying the external entropy threshold

We next evaluate the effects of using an internal entropy rate thresholding scheme. Since we wanted to concentrate on the effects of this step, we did not apply any thresholding on the external entropy (e.g. we used a threshold value of 100% that retains all cuboids). Figures 7b and 7c illustrate the percentage of records of the graph cubes retained for the four datasets, scaling the starting and ending internal entropy rates, respectively. For a starting internal entropy rate threshold of 10% we are left with just 0.7% of the Twitter graph cube, 0.003% of the VK graph cube, 0.002% of the Pokec graph cube and 0.00007% of the Patent graph cube records. The same observations hold for the ending internal entropy rate (Figure 7c). In conclusion, only a small percentage of the billions of records in these graph cubes reveal interconnections that are far from uniformity.

5.3. Performance Evaluation and Comparison to Alternative Techniques

In our prior work [13, 14], we presented a processing framework that first computes the external entropy rates for all possible drill-down operations in the graph cube lattice in order to select those that do not exceed the desired threshold. In a latter step, the internal entropy rates for all cuboids that participated in those drill-downs were examined. The new algorithm of Section 4 utilizes the bounds proposed in this work in order to prune many drill-downs that can not possible reach the desired threshold. In Figure 8, we depict the running times of our suggested framework compared to the brute-force method of [13, 14]. The internal entropy rate threshold used was 20% and we varied the external entropy rate threshold as shown in the x-axis of each plot. For all datasets the bisection algorithm proposed in this work provides significant benefits compared to the prior technique. Depending on the external entropy rate and the dataset it reduces the running times from 8% up to 90%.

These plots also suggest that using a smaller external entropy rate threshold results in faster execution because of the increased pruning achieved. One may wonder whether by utilizing a small external entropy rate in order to prune whole cuboids from consideration, there is a danger that certain skewed parts of low entropy within those cuboids may be missed by our technique. In order to assess this, we first computed the top-10 subgraphs ranked by their internal entropy rates using the full graph cube of each dataset. This is equivalent to using a 100% external entropy rate threshold. We then scaled the external entropy rate and calculated the percentage of those subgraphs retained. Since the number of target subgraphs is fixed to ten, this percentage can be interpreted as both a ”recall” and a ”precision” indicator. In Figure 9 we present the results for the four datasets. In the Twitter dataset, a 40% threshold retains 9 out of the top-10 subgraphs, while a 20% threshold retains 6/10 of them reducing the execution time from 1.5 minutes to 0.4 minutes (Figure 8c). For the VK dataset, a threshold of 20% retains 90% of the subgraphs and reduces the execution time from 9 minutes to 4 minutes. For the much larger Pokec dataset, a threshold of only 10% retains 90% of the top subgraphs and reduces the execution time by a

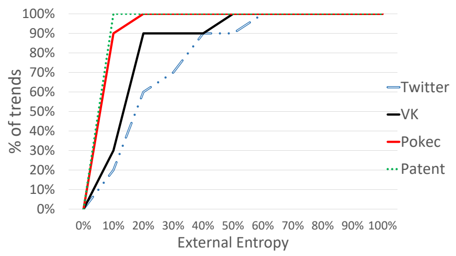


Figure 9: Percentage of top-10 subgraphs retained by scaling the external entropy threshold. Execution times are depicted in Figure 8

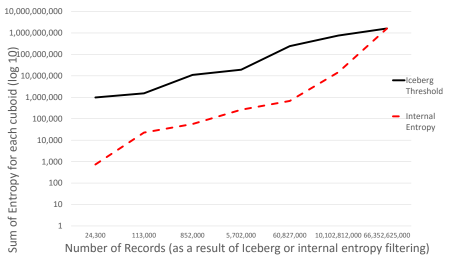


Figure 10: The sum of Entropy (log 10), scaling the number of records given as output from the internal and iceberg threshold respectively - Pokec dataset

factor of four, from 14 minutes to 3.2 minutes. Finally, for the Patent dataset, a threshold of 10% retains all the subgraphs reducing the execution time from 1.6 minutes to 0.1 minutes.

The internal entropy rates we introduce in our framework enable us to select subgraphs from the graph cube at a fine-grain. To our knowledge, there is no alternative technique in the literature that can achieve the same goal while using entropy to select portion of the graph cube. Existing techniques like the iceberg cube [22] can be used to select portions of the graph cube by utilizing a minimum support threshold. The intuition is that we would like to materialize cube records that are the result from at least a minimum number of raw data observations. Of course, this process does not take into consideration data skew, as we do in our proposal. For comparison, in Figure 10 we compute the iceberg graph cube in the case of the Pokec dataset, for different values of minimum support. We then adjust the internal entropy rate threshold so as to retain the same number of graph cube records (x-axis). In the figure we compare the resulting subsets of the graph cube in terms of the entropy retained in them. As expected our method maintains portions of the graph cube we significantly lower entropy (more skew) than the iceberg method that tends to keep more uniform associations. Nevertheless, we see of both techniques as complementary. It is rather straightforward for our method to also utilize a minimum support threshold, while performing the internal entropy calculations.

5.4. Experiments with Synthetic Datasets

In this section, we provide an experimental evaluation of the proposed framework using two synthetically generated datasets. The first one, denoted as S_{edges}, was derived from the Twitter dataset by adding two attributes on the edges of the data graph. Each of these attributes was following the Zipf distribution with parameters s=1.2 and n=1000.

The second dataset, S_{skewed}, is actually a family of datasets containing 5 attributes on the nodes of the data graph. Each attribute could either follow the uniform distribution with n=1000 distinct values or the Zipf distribution with parameters s=1.2 and n=1000. We created five instances of this dataset. In the first instance, a single attribute was following the Zipf distribution and the remaining four were uniformly distributed. We then progressively increased the number of skewed attributes up to the point were all five of them were following the Zipf distribution. In all instances the data edges between the graph nodes were randomly constructed. In Figure 11 we show how different levels of skew affect the total number of records remaining for analysis. The external and internal entropy rate thresholds were 20%, in all cases. As our techniques seek skewed interactions, the number of records retained increases linearly with the number of skewed attributes.

Figure 12 illustrates the percentage of records selected when scaling the internal edge entropy rate threshold. In order to concentrate on the effects of this parameter only, we used all cuboids for analysis setting the external entropy rate threshold to 100%. Similar to the internal entropy computations on the node attributes, the entropy calculations on the edge attributes provide significant pruning by selecting the most skewed parts of the data.

Finally, Figures 13 and 14 depict the running times when varying the external entropy threshold for the brute force algorithm of [13, 14] versus our bisection algorithm. The comparison is shown for dataset S_{skewed} (with 3 skewed

	Dataset S_{edges}	Dataset S_{skewed}
Profiles (nodes)	34M	4M
Relations (edges)	910M	493M
Number of Attributes	3 (nodes) + 2 (edges)	5
Number of Cuboids	256 ($2^3 * 2^2 * 2^3$)	1024 ($2^5 * 2^5$)
Graph Cube Records	124M	250M (all skewed) - 392M (all uniform)
Graph Cube Size	143MB	199GB (all skewed) - 290GB (all uniform)
Graph Cube Comp. Time	31 mins	40 - 87 mins

Table 3: Description of synthetic datasets

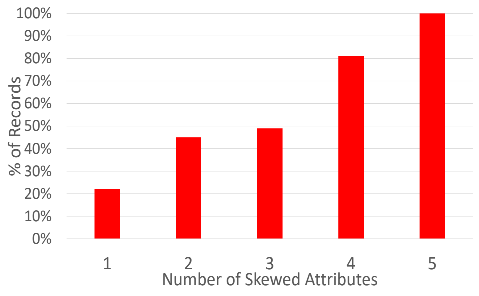


Figure 11: The number of records remain using 20% of both threshold internal and external entropy rate using skewed data from 1 to 5 attributes % (S_{edge} dataset)

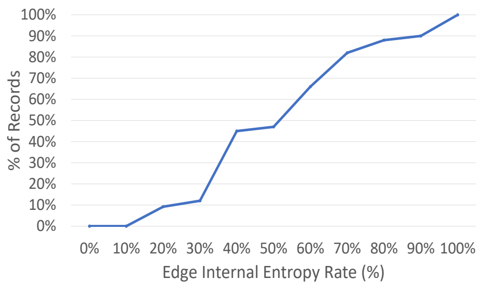


Figure 12: Scaling the edge internal entropy rate using 100% external threshold rate (S_{edge} dataset)

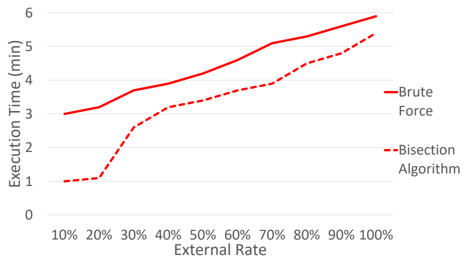


Figure 13: Varying the external entropy threshold, S_{skewed} dataset

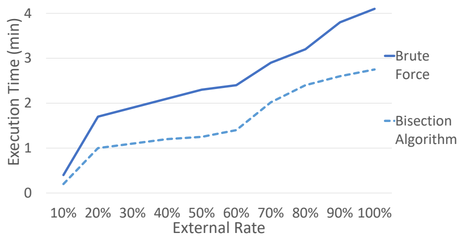


Figure 14: Varying the external entropy rate threshold, S_{edge} dataset

attributes) and S_{edge}. The starting internal entropy threshold was 20%. In all runs, the bisection algorithm was faster, providing gains up to 67% for S_{skewed} and up to 33% for S_{edge}.

6. Related Work

The data cube operator, introduced in [3] defines a foundational framework for declaring all possible aggregations along a list of selected domains, often referred to as "dimensions". The cube was proposed for flat, basket-type datasets. However, it has been recently extended for the case of interconnected datasets. The work in [8] introduced the graph cube that takes into account both attribute aggregation and structure summarization of the underlying graphs. This work mainly focuses on cuboids that aggregate the starting and ending nodes on the same dimensions, e.g. (nation - nation). More general aggregations that differentiate between the starting and ending nodes of the graph are not specifically mentioned but can be addressed under a cross-cuboid computation that is mentioned as an extension. In our work, we elevate such cuboids as first-class-citizens in the graph cube framework. As our experiments with real datasets indicate, such cuboids often hold significant insights for the underlying interconnections. Another distinction is that the work of [8] considers all records in the proposed graph cube. As we show in our work, only a small part of a complex graph cube carries interesting information when analyzed under the lens of our entropy-based navigation framework.

A recent work [23] considers aggregate attributed graphs. The authors name their model as a hyper graph cube and show how to compute it using MapReduce batches. The hyper graph cubes aggregate separately attributes at vertices and edges and then calculate the Cartesian product between them. Thus, they do not exploit and analyze the existing relationships under different levels of aggregation on the starting and ending nodes of the graph. OLAP-style summarization in the context of RDF graphs has been recently studied in [24]. The most significant difference from the previous works in graph cubes, is that our techniques address the vast size and complexity of the produced cuboids. To the best of our knowledge we are the first that utilize the entropy in order to filter the information of a graph cube.

The authors of [17] propose a novel framework for reconstructing multidimensional data from stored aggregates using the maximum entropy principle. In a nutshell, the proposed technique finds the model with the least information (maximum entropy) given a set of constraints that can be the $2^n - 2$ different aggregations in the cube (excluding the raw data and the grand total aggregate). The method uses a multi-pass algorithm called Iterative Proportional Filtering (IPF) that converges to the maximum entropy solution. IPF assumes that C, which is the part of the dataset we want to reconstruct, fits in memory. For the case of graph cubes C can be a whole cuboid that is often much larger than main memory. In that case, each iteration makes $\binom{n}{k}$ passes over C. Each pass updates the marginals of order k (the dimensionality of the known aggregates we use for reconstruction). In our context, these are all cuboids that are ancestors of the given cuboid C. In a latter step, reconstructed values based on the maximum entropy model can be compared to the original data. IPF necessitates access to all ancestor cuboids, while this is not required in our technique where all calculations are internal in C. For these reasons, we have been able to execute our entropy calculations on datasets containing billions of records, while the largest dataset used in [17] has 50K tuples. Thus, the technique of [17] is suitable for small datasets or when only a small part of a cuboid needs to be reconstructed or evaluated for possible deviations. Nevertheless, both techniques are a testimony to the benefits of using an information theoretic approach based on entropy that is not subjective or application dependent.

The information entropy was first introduced in [18] as a measure of unpredictability of information content. It measures how much information there is in an event. Entropy is frequently used for splitting decisions when computing Decision Trees [25] The information gain measures the change in information entropy from a prior state to new state after a split. Our external entropy rate measure utilizes the information gain metric in the nominator of its respective formula but differs in that it also takes into consideration the maximum possible increase in the entropy of a child cuboid in a drill down step. By conditioning the information gain over this quantity we are able to obtain the bounds that our selection algorithm utilizes.

Recently, an entropy-based model has been proposed [26] in order to estimate the strength of social connections by analyzing users' occurrences in space and time. This work considers triplets of (user, location, time) data and utilizes entropy to measure the diversity of user co-occurrences. In our work, we utilize entropy to measure the diversity within and across graph cuboids. The works of [27, 28] consider the case of analyzing very large collections of smaller data graphs, while in this work we consider a singe massive graph that is under investigation.

Application of graph mining techniques [29, 30, 31, 32, 33, 34, 35, 36] is orthogonal to our framework and can be used in conjunction. For instance, the work of [33] looks for structural patterns (or motifs) in the k-hop neighborhood of a node. The work of [32] suggests aggregation of graph nodes scores on vertices that contain some attribute of interest. Unlike conventional iceberg queries, the authors propose an aggregation method that is based on random walks and demonstrate their effectiveness and scalability. The authors of [37] explore data mining techniques to analyze tagging behavior on social graphs. The authors of [38] introduce graph-pattern association rules (GPAR). These rules extend traditional association rules with graph patterns that specify association between entities in a social graph.

Our techniques can also be used in conjunction with existing graph summarization tools like Perseus [39] that summarizes an input graph using statistics such as PageRank, radius, degree and flags outlier nodes [40], graph visualization tools, exploration tools [41], or with systems that recommend promising visualizations on aggregated datasets like SEEDB [42]. For example, the work of [43] takes a collection of input graphs and computes a graph summary where visual enhancements (colors, edge thicknesses, etc) are used to display relative frequencies of common features or to highlight differences between the input set. In this setting, our techniques can be used to compute interesting subgraphs (using our internal entropy calculations) that will then feed the interactive visualization engine. The work of [41] proposes a data cube exploration framework that seeks to provide sub-second levels of interactive cube exploration. This is achieved via speculative execution of queries based on the observed user workflow. The authors propose four basic traversal patterns that enable full exploration of the data cube lattice. Our work can be combined with this technique for prioritizing certain navigation steps (for instance a roll-up or drill down operation) based on the entropy bounds we introduce. SEEDB [42] suggests interesting visualizations if they depict large deviations from some reference (that can be a historical dataset or the rest of the data). Eventhough their techniques also investigate grouping operations, they are primarily concerned with selecting sub-sets from a list of candidate grouping dimensions. In contrast our techniques seek to suggest drill-down operations that increase the scope of the analysis by introducing new dimensions to the selected group-bys. Thus, they may complement that work for the case of OLAP analytics. Our techniques may also be combined with the work of [44] that seeks intuitive drill-down operations from aggregated views of data by utilizing the proposed entropy metrics in order to suggest interesting drill-downs.

7. Conclusions

In this work we first revisited the framework of graph cubes and proposed an intuitive representation of it as the Cartesian product of independent data cubes on the starting and ending nodes of the graph and, as an extension, of available attributes on the edges. We then addressed the enormous size and complexity of the resulting graph cubes by proposing an efficient algorithm that selects interesting parts of the aggregate graphs using information entropy calculations. Key to our algorithm is its ability to skip entropy calculations over large chunks of the graph cube by utilizing relationships between the cuboids in the graph cube lattice and some interested entropy bounds we proposed. Our experimental results validate the effectiveness of our techniques in managing graph cubes containing tens of billions of records.

8. Acknowledgements

This research is financed by the Research Centre of Athens University of Economics and Business, in the framework of the project entitled "Original Scientific Publications".

9. Appendix

Lemma 1. *Let us assume that there are two cuboids C_i and C_k and that C_i is the parent of C_k. Then, the delta entropy $\delta_{(C_k,C_i)} = eH(C_k) - eH(C_i)$ can be calculated as:*

$$\delta_{(C_k,C_i)} = \sum_{j=1}^{m} p(a_j^i) * eH(C_k|C_i = a_j^i)$$
$$= -\sum_{j=1}^{m}\{p(a_j^i) * \sum_{o=1}^{d_j} p(a_o^k|a_j^i) * \log_2 p(a_o^k|a_j^i)\} \quad (10)$$

where $p(a_o^k|a_j^i) = \frac{a_o^k}{a_j^i}$, r_o^k from DC_k is a more detailed record of r_j^i from DC_i and there are d_j such records in DC_k.

Proof. By definition, the two cuboids differ on a single attribute y added in child cuboid C_k. If there are m records in the dual representation DC_i of the parent cuboid (Section 2.2), these can be described as

$DC_i(s_1^1, s_2^1, \ldots, s_t^1, e_1^1, e_2^1, \ldots, e_w^1, a^1)$
$DC_i(s_1^2, s_2^2, \ldots, s_t^2, e_1^2, e_2^2, \ldots, e_w^2, a^2)$
$\ldots$
$DC_i(s_1^m, s_2^m, \ldots, s_t^m, e_1^m, e_2^m, \ldots, e_w^m, a^m)$

Also, let us assume that N= a^1 + a^2 + a^3 + ... + a^m. We can calculate the external entropy of cuboid C_i as

$$eH(C_i) = -\sum_{j=1}^{m} p(a_j) * \log_2 p(a_j) \quad , where\, p(a_j) = \frac{a^j}{N} \quad (11)$$

or $eH(C_i) = -(\frac{a^1}{N} * \log_2 \frac{a^1}{N} + \frac{a^2}{N} * \log_2 \frac{a^2}{N} + ... + \frac{a^m}{N} * \log_2 \frac{a^m}{N})$

Given the child C_k of cuboid C_i, then for each record in the dual representation $DC_i(s_1^j, s_2^j, \ldots, s_t^j, e_1^j, e_2^j, \ldots, e_w^j, a^j)$ of C_i, $j \in [1 \ldots m]$ there are d_j more detailed records in the dual representation DC_k, where $d_j \in [1 \ldots d]$ and d denotes the number of distinct values of the newly added attribute y in the drill-down process. Thus, the dual representation of cuboid C_k contains records of the form (y is the additional attribute in the child)

$DC_k(s_1^1, s_2^1, \ldots, s_t^1, y_1^1, e_1^1, e_2^1, \ldots, e_w^1, a_1^1)$
$DC_k(s_1^1, s_2^1, \ldots, s_t^1, y_2^1, e_1^1, e_2^1, \ldots, e_w^1, a_2^1)$
$\ldots$
$DC_k(s_1^1, s_2^1, \ldots, s_t^1, y_{d_j}^1, e_1^1, e_2^1, \ldots, e_w^1, a_{d_1}^1)$
$\ldots$
$DC_k(s_1^m, s_2^m, \ldots, s_t^m, y_1^m, e_1^m, e_2^m, \ldots, e_w^m, a_1^m)$
$DC_k(s_1^m, s_2^m, \ldots, s_t^m, y_2^m, e_1^m, e_2^m, \ldots, e_w^m, a_2^m)$
$\ldots$
$DC_k(s_1^m, s_2^m, \ldots, s_t^m, y_{d_j}^m, e_1^m, e_2^m, \ldots, e_w^m, a_{d_m}^m)$

Since the same number of data edges are aggregated in both cuboids, it holds that

$$\begin{aligned} &a_1 = a_1^1 + a_2^1 + \cdots + a_{d_1}^1, \\ &a_2 = a_1^2 + a_2^2 + \cdots + a_{d_2}^2, \\ &\ldots \\ &a_m = a_1^m + a_2^m + \cdots + a_{d_m}^m \end{aligned} \tag{12}$$

The external entropy of C_k is calculated as

$$\begin{aligned} eH(C_k) = &-(\frac{a_1^1}{N} * \log_2 \frac{a_1^1}{N} + \frac{a_2^1}{N} * \log_2 \frac{a_2^1}{N} + ... + \frac{a_{d_1}^1}{N} * \log_2 \frac{a_{d_1}^1}{N} \\ &+ \frac{a_1^2}{N} * \log_2 \frac{a_1^2}{N} + \frac{a_2^2}{N} * \log_2 \frac{a_2^2}{N} + ... + \frac{a_{d_2}^2}{N} * \log_2 \frac{a_{d_2}^2}{N} + \\ &.... \\ &+ \frac{a_1^m}{N} * \log_2 \frac{a_1^m}{N} + \frac{a_2^m}{N} * \log_2 \frac{a_2^m}{N} + ... + \frac{a_{d_m}^m}{N} * \log_2 \frac{a_{d_m}^m}{N}) \end{aligned}$$

The delta entropy $\delta_{(C_k,C_i)}$ will be computed as

$$\begin{aligned}
\delta_{(C_k,C_i)} =& eH(C_k) - eH(C_i) \\
\delta_{(C_k,C_i)} =& \frac{a_1}{N} * \log_2 \frac{a_1}{N} - (\frac{a_1^1}{N} * \log_2 \frac{a_1^1}{N} + \frac{a_2^1}{N} * \log_2 \frac{a_2^1}{N} + ... + \frac{a_{d_1}^1}{N} * \log_2 \frac{a_{d_1}^1}{N}) + \\
& \frac{a_2}{N} * \log_2 \frac{a_2}{N} - (\frac{a_1^2}{N} * \log_2 \frac{a_1^2}{N} + \frac{a_2^2}{N} * \log_2 \frac{a_2^2}{N} + ... + \frac{a_{d_2}^2}{N} * \log_2 \frac{a_{d_2}^2}{N}) + \\
& \dots \\
& \frac{a_m}{N} * \log_2 \frac{a_m}{N} - (\frac{a_1^m}{N} * \log_2 \frac{a_1^m}{N} + \frac{a_2^m}{N} * \log_2 \frac{a_2^m}{N} + ... + \frac{a_{d_m}^m}{N} * \log_2 \frac{a_{d_m}^m}{N})
\end{aligned}$$

We observe that (Equation 12)

$$\begin{aligned}
\frac{a_1}{N} * \log_2 \frac{a_1}{N} =& \frac{a_1^1}{N} * \log_2 \frac{(a_1^1 + a_2^1 + ... + a_{d_1}^1)}{N} + \frac{a_2^1}{N} * \log_2 \frac{(a_1^1 + a_2^1 + ... + a_{d_1}^1)}{N} + ... + \\
& \frac{a_{d_1}^1}{N} * \log_2 \frac{(a_1^1 + a_2^1 + ... + a_{d_1}^1)}{N}
\end{aligned}$$

$$\begin{aligned}
\frac{a_2}{N} * \log_2 \frac{a_2}{N} =& \frac{a_1^2}{N} * \log_2 \frac{(a_1^2 + a_2^2 + ... + a_{d_2}^2)}{N} + \frac{a_2^2}{N} * \log_2 \frac{(a_1^2 + a_2^2 + ... + a_{d_2}^2)}{N} + ... + \\
& \frac{a_{d_2}^2}{N} * \log_2 \frac{(a_1^2 + a_2^2 + ... + a_{d_2}^2)}{N}
\end{aligned}$$

$\dots$

$$\begin{aligned}
\frac{a_m}{N} * \log_2 \frac{a_m}{N} =& \frac{a_1^m}{N} * \log_2 \frac{(a_1^m + a_2^m + ... + a_{d_m}^m)}{N} + \frac{a_2^m}{N} * \log_2 \frac{(a_1^m + a_2^m + ... + a_{d_m}^m)}{N} + ... + \\
& \frac{a_{d_m}^m}{N} * \log_2 \frac{(a_1^m + a_2^m + ... + a_{d_m}^m)}{N}
\end{aligned}$$

Thus,

$$\delta_{(C_k,C_i)} = eH(C_k) - eH(C_i)$$

$$\tfrac{a_1^1}{N} * (\log_2 \tfrac{(a_1^1+a_2^1+...+a_{d_1}^1)}{N} - \log_2 \tfrac{a_1^1}{N}) + \tfrac{a_2^1}{N} * (\log_2 \tfrac{(a_1^1+a_2^1+...+a_{d_1}^1)}{N} - \log_2 \tfrac{a_2^1}{N})$$
$$+ \cdots + \tfrac{a_{d_1}^1}{N} * (\log_2 \tfrac{(a_1^1+a_2^1+...+a_{d_1}^1)}{N} - \log_2 \tfrac{a_{d_1}^1}{N}) +$$

$$\frac{a_1^2}{N} * (\log_2 \frac{(a_1^2+a_2^2+...+a_{d_2}^2)}{N} - \log_2 \frac{a_1^2}{N}) + \frac{a_2^2}{N} * (\log_2 \frac{(a_1^2+a_2^2+...+a_{d_2}^2)}{N} - \log_2 \frac{a_2^2}{N})$$
$$+ \cdots + \frac{a_{d_2}^2}{N} * (\log_2 \frac{(a_1^2+a_2^2+...+a_{d_2}^2)}{N} - \log_2 \frac{a_{d_2}^2}{N}) +$$
$$\ldots$$
$$+ \frac{a_1^m}{N} * (\log_2 \frac{(a_1^m+a_2^m+...+a_{d_m}^m)}{N} - \log_2 \frac{a_1^m}{N}) + \frac{a_2^m}{N} * (\log_2 \frac{(a_1^m+a_2^m+...+a_{d_m}^m)}{N} - \log_2 \frac{a_2^m}{N})$$
$$+ \cdots + \frac{a_{d_m}^m}{N} * (\log_2 \frac{(a_1^m+a_2^m+\cdots+a_{d_m}^m)}{N} - \log_2 \frac{a_{d_m}^m}{N})$$

and since $logX - logY = log(\frac{X}{Y})$

$$\delta_{(C_k,C_i)} =$$
$$\frac{a_1^1}{N} * \log_2 \frac{(a_1^1+a_2^1+...+a_{d_1}^1)}{a_1^1} + \frac{a_2^1}{N} * \log_2 \frac{(a_1^1+a_2^1+...+a_{d_1}^1)}{a_2^1} + ... + \frac{a_{d_1}^1}{N} * \log_2 \frac{(a_1^1+a_2^1+...+a_{d_1}^1)}{a_{d_1}^1} +$$
$$\frac{a_1^2}{N} * \log_2 \frac{(a_1^2+a_2^2+...+a_{d_2}^2)}{a_1^2} + \frac{a_2^2}{N} * \log_2 \frac{(a_1^2+a_2^2+...+a_{d_2}^2)}{a_2^2} + ... + \frac{a_{d_2}^2}{N} * \log_2 \frac{(a_1^2+a_2^2+...+a_{d_2}^2)}{a_{d_2}^2} +$$
$$\ldots$$
$$+ \frac{a_1^m}{N} \log_2 \frac{(a_1^m+a_2^m+...+a_{d_m}^m)}{a_1^m} + \frac{a_2^m}{N} * \log_2 \frac{(a_1^m+a_2^m+...+a_{d_m}^m)}{a_2^m} + ... + \frac{a_{d_m}^m}{N} * \log_2 \frac{(a_1^m+a_2^m+...+a_{d_m}^m)}{a_a^m}$$

This can be written as (Equation 12)

$$\delta_{(C_k,C_i)} =$$
$$\frac{a_1^1}{N} * \log_2 \frac{a_1}{a_1^1} + \frac{a_2^1}{N} * \log_2 \frac{a_1}{a_2^1} + ... + \frac{a_{d_1}^1}{N} * \log_2 \frac{a_1}{a_{d_1}^1} +$$
$$\frac{a_1^2}{N} * \log_2 \frac{a_2}{a_1^2} + \frac{a_2^2}{N} * \log_2 \frac{a_2}{a_2^2} + ... + \frac{a_{d_2}^2}{N} * \log_2 \frac{a_2}{a_{d_2}^2} +$$
$$\ldots$$
$$+ \frac{a_1^m}{N} \log_2 \frac{a_m}{a_1^m} + \frac{a_2^m}{N} * \log_2 \frac{a_m}{a_2^m} + ... + \frac{a_{d_m}^m}{N} * \log_2 \frac{a_m}{a_{d_m}^m}$$

Due to the fact that $\frac{a_j}{a_w^j} \geq 1$ for all $j \in [1 \ldots m]$ and $w \in [1 \ldots d_j]$ then $\log_2 \frac{a_j}{a_j^w} \geq 0$. This implies that the delta entropy cannot be negative. Moreover we can write the above equation as

$$\delta_{(C_k,C_i)} =$$
$$\frac{a_1^1}{N} * \frac{a_1}{a_1^1} * \frac{a_1^1}{a_1} * (-\log_2 \frac{a_1^1}{a_1}) + \frac{a_2^1}{N} * \frac{a_1}{a_2^1} * \frac{a_2^1}{a_1} * (-\log_2 \frac{a_2^1}{a_1}) + \cdots +$$
$$\frac{a_{d_j}^1}{N} * \frac{a_1}{a_{d_1}^1} * \frac{a_{d_1}^1}{a_1} * (-\log_2 \frac{a_{d_1}^1}{a_1}) +$$
$$\frac{a_1^2}{N} * \frac{a_2}{a_1^2} * \frac{a_1^2}{a_2} * (-\log_2 \frac{a_1^2}{a_2}) + \frac{a_2^2}{N} * \frac{a_2}{a_2^2} * \frac{a_2^2}{a_2} * (-\log_2 \frac{a_2^2}{a_2}) + \cdots +$$
$$\frac{a_{d_2}^2}{N} * \frac{a_2}{a_{d_2}^2} * \frac{a_{d_2}^2}{a_2} * (-\log_2 \frac{a_{d_2}^2}{a_2}) +$$
$$\ldots$$
$$\frac{a_1^m}{N} * \frac{a_m}{a_1^m} * \frac{a_1^m}{a_m} * (-\log_2 \frac{a_1^m}{a_m}) + \frac{a_2^m}{N} * \frac{a_m}{a_2^m} * \frac{a_2^m}{a_m} * (-\log_2 \frac{a_2^m}{a_m}) + \cdots +$$
$$\frac{a_{d_m}^m}{N} * \frac{a_m}{a_{d_m}^m} * \frac{a_{d_m}^m}{a_m} * (-\log_2 \frac{a_{d_m}^m}{a_m})$$

the above formula can be simplified further resulting in

$\delta_{(C_k,C_i)} =$

$$-(\tfrac{a_1}{N} * \tfrac{a_1^1}{a_1} * \log_2 \tfrac{a_1^1}{a_1} + \tfrac{a_1}{N} * \tfrac{a_2^1}{a_1} * \log_2 \tfrac{a_2^1}{a_1} + ... + \tfrac{a_1}{N} * \tfrac{a_{d_1}^1}{a_1} * \log_2 \tfrac{a_{d_1}^1}{a_1} +$$
$$\tfrac{a_2}{N} * \tfrac{a_1^2}{a_2} * \log_2 \tfrac{a_1^2}{a_2} + \tfrac{a_2}{N} * \tfrac{a_2^2}{a_2} * \log_2 \tfrac{a_2^2}{a_2} + ... + \tfrac{a_2}{N} * \tfrac{a_{d_2}^2}{a_2} * \log_2 \tfrac{a_{d_2}^2}{a_2} +$$
$$\dots$$
$$+ \tfrac{a_m}{N} * \tfrac{a_1^m}{a_m} * \log_2 \tfrac{a_1^m}{a_m} + \tfrac{a_m}{N} * \tfrac{a_2^m}{a_m} * \log_2 \tfrac{a_2^m}{a_m} + ... + \tfrac{a_m}{N} * \tfrac{a_{d_m}^m}{a_m} * \log_2 \tfrac{a_{d_m}^m}{a_m})$$

Moreover, it holds that

$$p(a_j^i) = \tfrac{a_j^i}{N}$$

and

$$p(a_o^k | a_j^i) = \tfrac{a_o^k}{a_j^i}$$

The above equation can be rewritten using these probabilities as

$$\delta_{(C_k,C_i)} =$$
$$-(p(a_1) * p(a_1^1|a_1) * \log_2 p(a_1^1|a_1) + p(a_1) * p(a_2^1|a_1) * \log_2 p(a_2^1|a_1) + \cdots +$$
$$p(a_1) * p(a_{d_1}^1|a_1) * \log_2 p(a_{d_1}^1|a_1) +$$
$$p(a_2) * p(a_1^2|a_2) * \log_2 p(a_1^2|a_2) + p(a_2) * p(a_2^2|a_2) * \log_2 p(a_2^2|a_2) + \cdots +$$
$$p(a_2) * p(a_{d_2}^2|a_2) * \log_2 p(a_{d_2}^2|a_2) +$$
$$\dots$$
$$p(a_m) * p(a_1^m|a_m) * \log_2 p(a_1^m|a_m) + p(a_m) * p(a_2^m|a_m) * \log_2 p(a_2^m|a_m) + \cdots +$$
$$p(a_m) * p(a_{d_m}^m|a_m) * \log_2 p(a_{d_m}^m|a_m))$$

and consequently

$$\delta_{(C_k,C_i)} =$$
$$-(p(a_1) * [p(a_1^1|a_1) * \log_2 p(a_1^1|a_1) + p(a_2^1|a_1) * \log_2 p(a_2^1|a_1) + \cdots +$$
$$p(a_{d_1}^1|a_1) * \log_2 p(a_{d_1}^1|a_1)] +$$
$$p(a_2) * [p(a_1^2|a_2) * \log_2 p(a_1^2|a_2) + p(a_2^2|a_2) * \log_2 p(a_2^2|a_2) + \cdots +$$
$$p(a_{d_2}^2|a_2) * \log_2 p(a_{d_2}^2|a_2)] +$$
$$\dots$$
$$p(a_m) * [p(a_1^m|a_m) * \log_2 p(a_1^m|a_m) + p(a_2^m|a_m) * \log_2 p(a_2^m|a_m) + \cdots +$$
$$p(a_{d_m}^m|a_m) * \log_2 p(a_{d_m}^m|a_m)])$$
$$= -(p(a_1) * \textstyle\sum_{o=1}^{d_1} p(a_o^1|a_1) * \log_2 p(a_o^1|a_1) +$$
$$p(a_2) * \textstyle\sum_{o=1}^{d_2} p(a_o^2|a_2) * \log_2 p(a_o^2|a_2) +$$
$$\dots$$
$$+ p(a_m) * \textstyle\sum_{o=1}^{d_m} p(a_o^m|a_m) * \log_2 p(a_o^m|a_m))$$

Which is Equation 3.

□

References

[1] B. Amann, M. Scholl, Gram: A Graph Data Model and Query Languages, in: Proceedings of the ACM conference on Hypertext, New York, NY, USA, 1992, pp. 201–211.

[2] R. H. Güting, Graphdb: Modeling and Querying Graphs in Databases, in: VLDB, 1994, pp. 297–308.

[3] J. Gray, A. Bosworth, A. Layman, H. Pirahesh, Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total, in: ICDE, 1996, pp. 152–159.

[4] C. Chen, X. Yan, F. Zhu, J. Han, P. S. Yu, Graph OLAP: Towards Online Analytical Processing on Graphs, in: ICDM, 2008, pp. 103–112.

[5] A. Ghrab, O. Romero, S. Skhiri, A. A. Vaisman, E. Zimányi, A Framework for Building OLAP Cubes on Graphs, in: Proceedings of ADBIS, 2015.

[6] K. Khan, K. Najeebullah, W. Nawaz, Y. Lee, OLAP on structurally significant data in graphs, CoRR abs/1401.6887.

[7] X. Li, J. Han, H. Gonzalez, High-dimensional OLAP: A minimal cubing approach, in: (e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004, 2004, pp. 528–539.

[8] P. Zhao, X. Li, D. Xin, J. Han, Graph cube: On warehousing and olap multidimensional networks, in: Proceedings of ACM SIGMOD, 2011.

[9] W. H. Inmon, Building the Data Warehouse, QED Information Sciences, Inc., Wellesley, MA, USA, 1992.

[10] R. Kimball, M. Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd Edition, John Wiley & Sons, Inc., New York, NY, USA, 2002.

[11] N. Roussopoulos, Y. Kotidis, M. Roussopoulos, Cubetree: Organization of and Bulk Incremental Updates on the Data Cube, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, USA., 1997, pp. 89–99.

[12] Y. Sismanis, A. Deligiannakis, N. Roussopoulos, Y. Kotidis, Dwarf: Shrinking the PetaCube, in: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 3-6, 2002, 2002, pp. 464–475.

[13] D. Bleco, Y. Kotidis, Entropy-based Selection of Graph Cuboids, in: Proceedings of the 5th International Workshop on Graph Data Management Experiences and Systems (GRADES), 2017.

[14] D. Bleco, Y. Kotidis, Finding the Needle in a Haystack: Entropy Guided Exploration of Very Large Graph Cubes, in: Proceedings of the International Workshop on Big Data Visual Exploration and Analytics (BigVis), Vienna, Austria, March 2018, 2017.

[15] H. Lenz, A. Shoshani, Summarizability in OLAP and Statistical Data Bases, in: Ninth International Conference on Scientific and Statistical Database Management, Proceedings, August 11-13, 1997, Olympia, Washington, USA, 1997, pp. 132–143.

[16] J. Mazón, J. Lechtenbörger, J. Trujillo, A survey on summarizability issues in multidimensional modeling, Data Knowl. Eng. 68 (12) (2009) 1452–1469.

[17] T. Palpanas, N. Koudas, Entropy Based Approximate Querying and Exploration of Datacubes, in: Proceedings of SSDM, 2001, pp. 81–90.

[18] C. E. Shannon, A mathematical theory of communication, SIGMOBILE Mob. Comput. Commun. Rev. 5 (1) (2001) 3–55.

[19] J. Leskovec, A. Krevl, SNAP Datasets: Stanford large network dataset collection, http://snap.stanford.edu/data (Jun. 2014).

[20] B. Hall, A. Jaffe, M. Trajtenberg, The NBER Patent Citations Data File: Lessons, Insights and Methodological Tools, Working papers (2001).

[21] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, Spark: Cluster Computing with Working Sets, in: Proceedings of HotCloud, 2010.

[22] K. Beyer, R. Ramakrishnan, Bottom-up computation of sparse and iceberg cube, in: Proceedings of SIGMOD, 1999, pp. 359–370.

[23] Z. Wang, Q. Fan, H. Wang, K. Tan, D. Agrawal, A. El Abbadi, Pagrol: Parallel graph olap over large-scale attributed graphs, in: ICDE, 2014.

[24] E. A. Azirani, F. Goasdoué, I. Manolescu, A. Roatis, Efficient OLAP operations for RDF analytics, in: ICDE Workshops, 2015, pp. 71–76.

[25] J. R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.

[26] H. Pham, C. Shahabi, Y. Liu, EBM: An Entropy-Based Model to Infer Social Strength from Spatiotemporal Data, in: Proc. of SIGMOD, 2013.

[27] D. Bleco, Y. Kotidis, Business Intelligence on Complex Graph Data, in: Proceedings of the 2012 Joint EDBT/ICDT Workshops, Berlin, Germany, 2012, pp. 13–20.

[28] D. Bleco, Y. Kotidis, Graph Analytics on Massive Collections of Small Graphs, in: Proceedings of the EDBT, Athens, Greece, 2014, pp. 523–534.

[29] A. Arora, M. Sachan, A. Bhattacharya, Mining Statistically Significant Connected Subgraphs in Vertex Labeled Graphs, in: International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014, 2014, pp. 1003–1014.

[30] M. Elseidy, E. Abdelhamid, S. Skiadopoulos, P. Kalnis, GRAMI: frequent subgraph and pattern mining in a single large graph, PVLDB 7 (7) (2014) 517–528.

[31] B. Kimelfeld, P. G. Kolaitis, The complexity of mining maximal frequent subgraphs, ACM Trans. Database Syst. 39 (4) (2014) 32:1–32:33.

[32] N. Li, Z. Guan, L. Ren, J. Wu, J. Han, X. Yan, gIceberg: Towards Iceberg Analysis in Large Graphs, in: 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013, 2013, pp. 1021–1032.

[33] W. E. Moustafa, A. Deshpande, L. Getoor, Ego-centric Graph Pattern Census, in: Proceedings of ICDE, 2012, pp. 234–245.

[34] G. Qi, C. C. Aggarwal, T. S. Huang, Community Detection with Edge Content in Social Media Networks, in: IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012, 2012, pp. 534–545.

[35] A. Silva, W. M. Jr., M. J. Zaki, Mining Attribute-structure Correlated Patterns in Large Attributed Graphs, PVLDB 5 (5) (2012) 466–477.

[36] Y. Tao, C. Sheng, J. Li, Finding Maximum Degrees in Hidden Bipartite Graphs, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010, 2010, pp. 891–902.

[37] M. Das, S. Thirumuruganathan, S. Amer-Yahia, G. Das, C. Yu, An Expressive Framework and Efficient Algorithms for the Analysis of Collaborative Tagging, VLDB J. 23 (2) (2014) 201–226.

[38] W. Fan, X. Wang, Y. Wu, J. Xu, Association Rules with Graph Patterns, PVLDB 8 (12) (2015) 1502–1513.

[39] D. Koutra, D. Jin, Y. Ning, C. Faloutsos, Perseus: An Interactive Large-Scale Graph Mining and Visualization Tool, PVLDB 8 (12).

[40] J. Sun, H. Qu, D. Chakrabarti, C. Faloutsos, Neighborhood Formation and Anomaly Detection in Bipartite Graphs, in: Proceedings of ICDM, 2005.

[41] N. Kamat, P. Jayachandran, K. Tunga, A. Nandi, Distributed and Interactive Cube Exploration, in: IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014, 2014, pp. 472–483.

[42] M. Vartak, S. Rahman, S. Madden, A. G. Parameswaran, N. Polyzotis, SEEDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics, PVLDB 8 (13) (2015) 2182–2193.

[43] D. Koop, J. Freire, C. T. Silva, Visual Summaries for Graph Collections, in: IEEE Pacific Visualization Symposium, PacificVis 2013, February 27 2013-March 1, 2013, Sydney, NSW, Australia, 2013, pp. 57–64.

[44] M. Joglekar, H. Garcia-Molina, A. G. Parameswaran, Interactive Data Exploration with Smart Drill-down, in: Proceedings of ICDE, 2016.