Compressing Big OLAP Data Cubes over Mobile Clouds: A Hierarchy-Based Data Partitioning Approach

Alfredo Cuzzocrea [1,2,*,†], Mojtaba Hajian [1], Abderraouf Hafsaoui [1]

[1] iDEA LAB, University of Calabria, Rende, Italy
[2] Department of Computer Science, University of Paris City, Paris, France

Abstract
This paper introduces the Indexed Quad-Tree Summary (IQTS) algorithm, along with its concepts, models, and "philosophy". IQTS allows us to compress multidimensional *OLAP* views derived from big *OLAP* data cubes that populate Mobile Clouds. In addition to this, IQTS effectively and efficiently supports approximate query answering over compressed multidimensional *OLAP* views. These functionalities turn to be enabling functionalities for big data applications in a wide collection of modern scenarios, such as healthcare analytics. In light of these considerations, this paper provides motivations, anatomy and procedures of IQTS, enriched by several case studies that contribute to pinpoint the benefits coming from our proposed algorithm.

Keywords
Big Data, Big OLAP Cubes, Mobile Clouds, Approximate Query Answering, Intelligent Big Data Applications

1. Introduction

Mobile Cloud Environments (e.g., [1,2,3]) are conquering the scene, even because coupled with big data research initiatives (e.g., [4,5,6]). Mobile Clouds are now very popular, as they characterize a plethora of modern intelligent applications, ranging from environmental monitoring, healthcare support, online auction, hazardous/extreme environments, pharmacovigilance, and so forth.

Indeed, ubiquitous computing (e.g., [7,8]) is now everywhere, especially with the tremendous rising of ICT technologies, based on which we now use an impressive number of mobile devices including smartphones, tables, personal digital assistants, and so forth. At the same, the volume, velocity and variety of big data is exploding day by day, therefore accessing these datasets to retrieve information and knowledge via mobile devices is a critical open issue in modern big data research (e.g., [9]).

On a wider perspective, these systems can be intended as mobile big data systems, since they really rely on big datasets, with the goal of finally supporting actionable knowledge discovery and intelligent decision making (e.g., [10]). Due to this evident property, managing, querying and analyzing big datasets in these systems has become a real research challenge, especially due to nature of big data, complexity of big data, topology of the network/delivery model, data availability, and so forth (e.g., [11]).

Fig. 1 shows a typical mobile big data Cloud environment, where massive Big Data Sources are made available via a proper Cloud Infrastructure. In turn, the Cloud is accessed via a Wireless Network by several Mobile Devices, which finally access, manage, query and analyze target datasets.

Among the various family of big data sources, big *OLAP* data cubes play an astonishing role (e.g., [12]). Indeed, these multidimensional structures store (big) data at a high number of dimensions, as

SEDB 2024: 32nd Symposium on Advanced Database Systems, June 23-26, 2024, Villasimius, Sardinia, Italy
* Corresponding author.
† This research has been made in the context of the Excellence Chair in Big Data Management and Analytics at University of Paris City, Paris, France.
alfredo.cuzzocrea@unical.it (A. Cuzzocrea); mojtaba.hajian@unical.it (M. Hajian); ahafsaoui.idealab.unical@gmail.com (A. Hafsaoui)
0000-0002-7104-6415 (A. Cuzzocrea); 0009-0007-3740-776X (M. Hajian); 0000-0002-2364-172X (A. Hafsaoui)

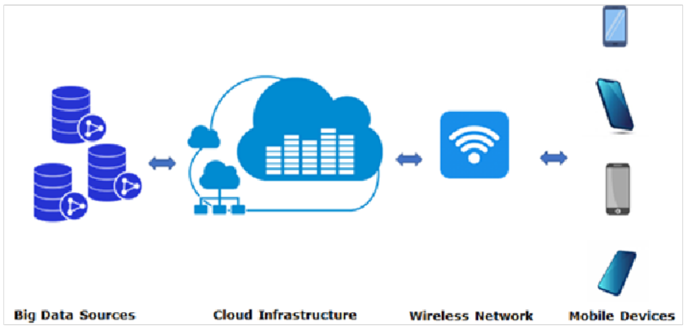


Figure 1: Mobile big data Cloud environment

required by modern big data application scenarios, and provide support to a plethora of powerful analytics that can be developed on top of them (e.g., [13,14,15]). On the other hand, their size and complexity make prohibitive the cost for accessing and querying them, so that data compression paradigms are necessary in order to mitigate this relevant drawback. Also, the presence of mobile devices makes the problem worse, due to their well-known limitations (e.g., [16]).

Inspired by these considerations, this paper introduces the Indexed Quad-Tree Summary (IQTS) algorithm, along with its concepts, models, and "philosophy". IQTS allows us to compress multidimensional *OLAP* views derived from big *OLAP* data cubes that populate Mobile Clouds. In addition to this, IQTS effectively and efficiently supports approximate query answering over compressed multidimensional *OLAP* views. These functionalities turn to be enabling functionalities for big data applications in a wide collection of modern scenarios, such as healthcare analytics, including approaches inspired from visualization metaphors (e.g., [17]). We provide motivations, anatomy and procedures of IQTS, enriched by several case studies that contribute to pinpoint the benefits coming from our proposed algorithm. The full version of this contribution appears in [31].

2. The IQTS Algorithm

In this Section, we provide a detailed description of the IQTS algorithm. As noticed in Section 1, computational cost is a drawback in multidimensional OLAP data cubes, however, there is an approach to use handheld devices as a solution. Today, mobile devices have been very popular and every people possess a mobile device. IQTS arises to enable mobile users (m-users) to access, query and receive approximate answers from remote OLAP servers, using intelligent compression techniques. IQTS applies a partitioned representation of the OLAP view to create a compressed two-dimensional OLAP view which can be downloaded into the handheld device, then the m-user is able to make a query off-line.

Fig. 2 illustrates the procedure in which a handheld device uses the data cubes in an OLAP server. For this goal, the steps are as follows: (i) considering a multidimensional data cube C, the m-user selects two dimensions of the data cube Visualization Dimensions (VD). Each VD is made from merging of other dimensions of the data cube C; (ii) since we have a new data cube, we need to obtain new OLAP aggregations, then a two-dimensional OLAP view of C is created which is designated V; (iii) a compressed representation of V is created in an application server, then it is transmitted to the m-user.

OLAP compression process has a number of steps including: (i) dimension flattening; (ii) compression of two-dimensional OLAP view (iii) in-memory representation.

The first step of the compression technique is dimension flattening. Consider two dimensions $d_{j+1}$and d_j, with the hierarchies $X(d_{j+1})$ and $X(d_j)$, to be merged together and create a new dimension hierarchy called $X^*(d_j)$, as it is shown in Fig. 3. In this case, a sub-tree of $X(d_{j+1})$ rooted at the root node of $X(d_{j+1})$ with a depth of $D_{j+1} = 1$ is selected, then a clone of it is inserted to each member of the level L_{X_j} of $X(d_j)$ at the depth $L_{X_j} = 1$. Therefore, the $X^*(d_j)$ is resulted from merging of the two dimensions d_j and d_{j+1}. We can continue this procedure to merge other dimension of the multidimensional OLAP, C, with $X^*(d_j)$, so that flatten more than two dimensions to one new dimension.

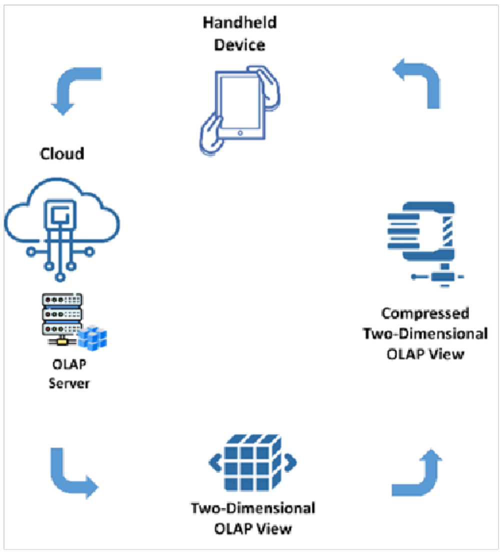

Handheld Device
Cloud
OLAP Server
Compressed Two-Dimensional OLAP View
Two-Dimensional OLAP View

Figure 2: Data flow procedure in the IQTS algorithm

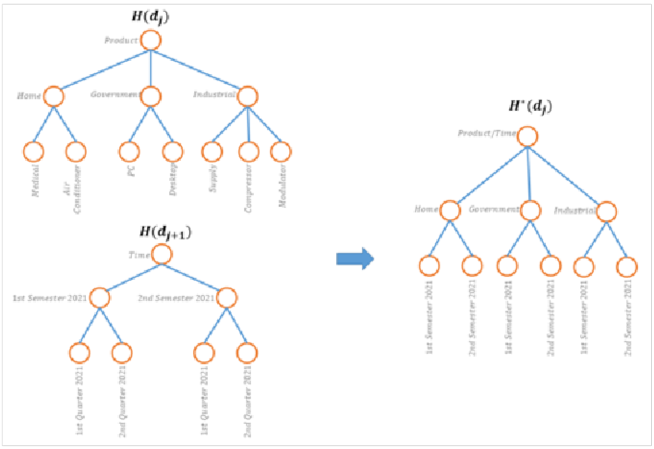


Figure 3: An example of dimension flattening (merging two dimensions to one dimension)

After the flattening procedure, it should be noted that the data must be re-aggregated according to the new VDs.

The main compression technique applies on the resulted two-dimensional OLAP view V after dimension flattening. The compression procedure applies iterative methods to develop a quad-tree partition of V, where each bucket contains the sum of the items exist inside it. The algorithm targets to select the bucket p_j where it includes the least homogeneity calculated by the following criterion:

$$SSE(p) = \sum_{i \in p} (V[i] - AVG(p))^2 \tag{1}$$

where i denotes a position inside p, $V[i]$ is the value of V at position i, and $AVG(p)$ is the average of values inside p.

To split the buckets into four new buckets with less homogeneity, the algorithm of splitting finds a splitting position ($N_{j,k}$). Consider dimensions d_0 and $d_1$of the OLAP view V, the selected bucket p_j has a range $[l_{j,0}{:}\, r_{j,0}]$ on dimension d_0 and $[l_{j,1}{:}\, r_{j,1}]$ on dimension d_1, then $N_{j,k}$ is selected from $[l_{j,k}{:}\, r_{j,k}]$. Thus, the interval is divided to two parts: $[l_{j,k}{:}\, N_{j,k}]$ and $[N_{j,k} + 1{:}\, r_{j,k}]$, where $N_{j,k}$ is such a splitting point at a level of $X(d_k)$.

Firstly, we select the half-way point of the interval $[l_{j,k}{:}\, r_{j,k}]$ as follows:

$$H_{j,k} = \lfloor (r_{j,k} - l_{j,k} + 1)/2 \rfloor \tag{2}$$

and designate $\alpha_{j,k}$. Now, considering the $L_{j,k}$ as a specific level of the tree $T_{j,k}$, we have the $\beta_{j,k}$, the ancestor of $\alpha_{j,k}$ at the level $L_{j,k}$. Moreover, $\alpha^L_{j,k}$ and $\alpha^R_{j,k}$ are the Leftmost Descendant Leaf (LDL) and

Rightmost Descendant Leaf (RDL) of $\beta_{j,k}$, respectively. We define $\alpha^{*}_{j,k}$ of $\{\alpha^{L}_{j,k}, \alpha^{R}_{j,k}\}$, where there are two states as following if it is inside or outside the interval:

$$\emptyset_{j,k} = [H_{j,k} - \left\lfloor \frac{1}{3}(r_{j,k} - l_{j,k} + 1) \right\rfloor : H_{j,k} + \left\lfloor \frac{1}{3}(r_{j,k} - l_{j,k} + 1) \right\rfloor] \quad (3)$$

noting that we intend to select a splitting position close to the half-way point of the $[l_{j,k} : r_{j,k}]$:

- if $\alpha^{*}_{j,k}$ is inside the $\emptyset_{j,k}$: then the $\alpha^{*}_{j,k}$ is the splitting position of $[l_{j,k} : r_{j,k}]$;
- if $\alpha^{*}_{j,k}$ is outside the $\emptyset_{j,k}$: then select another $\beta_{j,k}$ where it is an ancestor node of $\alpha_{j,k}$ at a deeper level of the hierarchy. This procedure is repeated until an appropriate $\alpha^{*}_{j,k}$ inside the $\emptyset_{j,k}$ is found. Then the $\alpha^{*}_{j,k}$ is the splitting position of $[l_{j,k} : r_{j,k}]$.

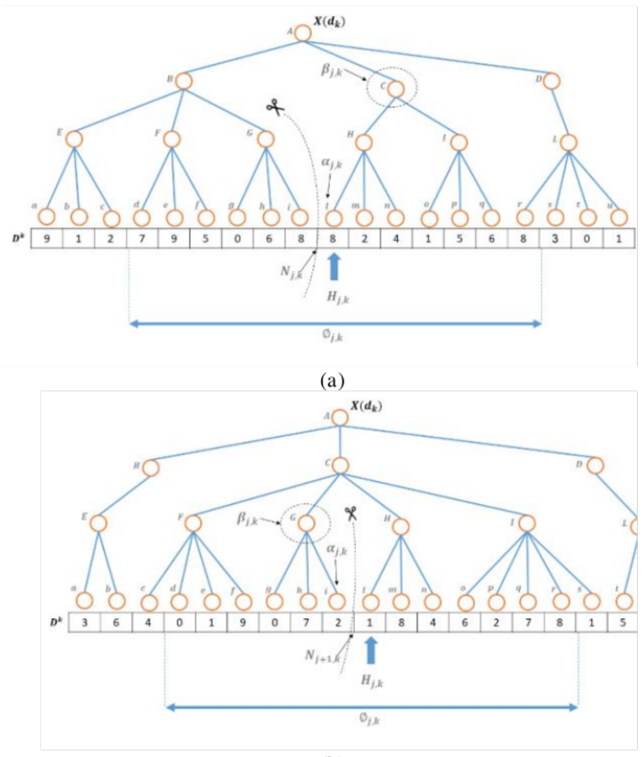


Figure 4: Two examples of hierarchy splitting

The above procedure is shown in Fig. 4 as two examples of splitting position $N_{j,k}$. From Fig. 5 to Fig. 6. we show the compression procedure of the last example two-dimensional OLAP view V Zone/Time-Product dimensions. In Fig. 5 left, the algorithm considers one bucket where represented as the whole OLAP view V. At Fig. 5 right, each dimension is split to two new partitions, thus there are four new buckets. Note that the target is to select the buckets with least homogeneity (i.e. containing the least uniform distribution of data), and split them to make new homogenous buckets. In the next step, we have new buckets that should be investigated for their homogeneity, and the least homogenous buckets must be divided to four new buckets; as it is shown in Fig. 6 left, one bucket from last partitioning step needs to be split. In the next steps this procedure is continued with the above criteria (see Fig. 6 right). The procedure will be continued until the storage assigned in the handheld device is full.

The representation of compressed OLAP view is an important issue, because of the memory limitations in mobile systems. Thus, an approach is necessary to make use of the available memory in an efficient way. In this approach, we assume that only non-negative integers are included in the OLAP view V, and each bucket contains the sum of the measure values inside it, where it is assigned 32 bits for these values. After splitting procedure, each bucket is split to four sub-buckets, then we store the sum of the values existing inside each three of the new sub-buckets (if the sum value is non-zero; if the sum value is zero then we leave 32 bits as null in order to use that for other additional buckets) and

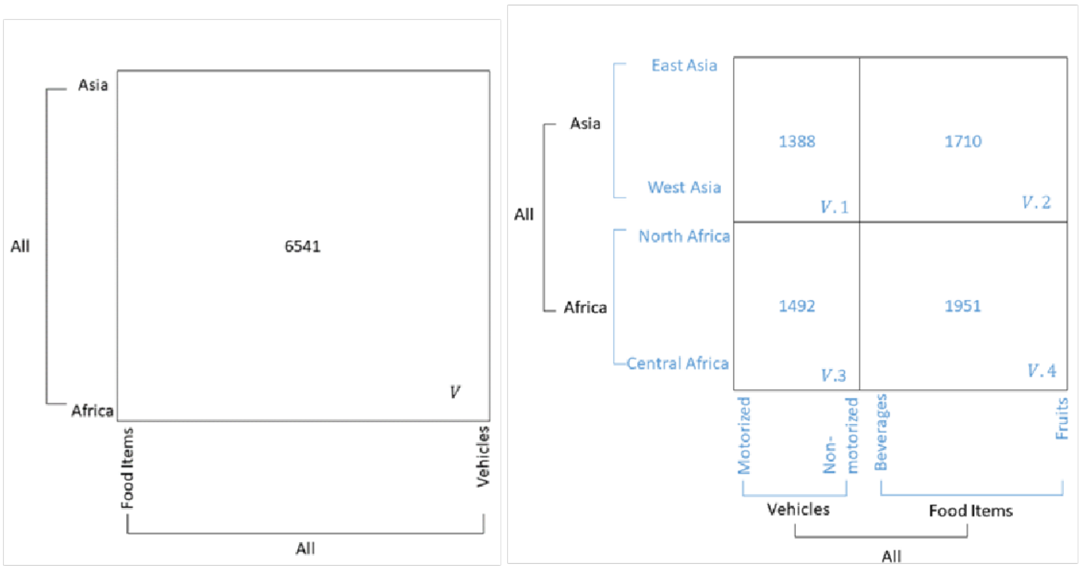


Figure 5: Partitioning procedure of an example two-dimensional data cube: step 1/4 (left) and step 2/4 (right)

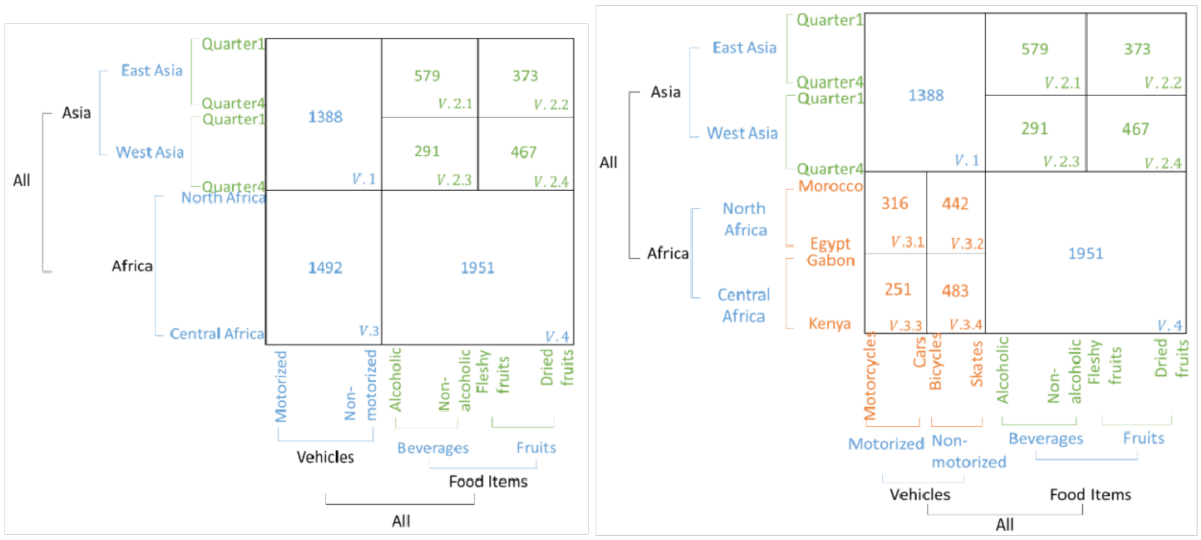


Figure 6: Partitioning procedure of an example two-dimensional data cube: step 3/4 (left) and step 4/4 (right)

leave the fourth one. Using this approach, the fourth bucket's value can be calculated by subtraction of sum of the three sub-buckets from the parent bucket's value. In the next steps, the splitting procedure is continued as selecting the least uniform distributed data to be split to four buckets. Therefore, the elements inside each new bucket will have small variance and the values will be close to the average of the elements' values inside the bucket.

Our technique, at this step (i.e., compression after the first-stage splitting described above), is inspired by [18] and produces a quad-tree partition of V, finally represented by IQTS, which singles-out a number of buckets and stores the sums of measure values only for them. More into details, we are given a two-dimensional data cube view V of size N, already partitioned in four buckets We want to construct a compressed representation of V using a size M≪N. Such a partitioning scheme is based on choosing, at each step, as that to further partitioning the bucket having the maximum Sum of Square Errors (SSE) between the original data cube view V and its current partitioned representation (i.e., IQTS(V)), defined as follows:

$$SSE\big(IQTS(V)\big) = \sum_{q_i \in Leaves(IQTS(V))} SSE(q_i) \tag{4}$$

such that Leaves($IQTS(V)$) is the set composed by the leave nodes of $IQTS(V)$, also called terminal buckets, and:

$$SSE(q_i) = \sum_{j \in q_i} \left(V[j] - AVG(q_i)\right)^2 \tag{5}$$

In addition to the above issue, there is another challenge that should be considered. In the query estimation, the reconstruction of the data after compression process, can be performed by a linear interpolation if the distribution of data in each bucket is uniform. But, the problem is that practically this will not happen, it means that the buckets don't contain uniformly distributed enough to enable us to use linear interpolation for query estimates. To solve this challenge, we need to store some information that enhances the approximation accuracy in data distribution reconstruction of each buckets. The Indexing method is used to address the problem of data distribution. For the sake of efficiency, Indexing method only is applied on the non-zero sum buckets and/or buckets containing data with not very uniform distribution, because this type of data distribution does not need linear interpolation and does not enhance the accuracy.

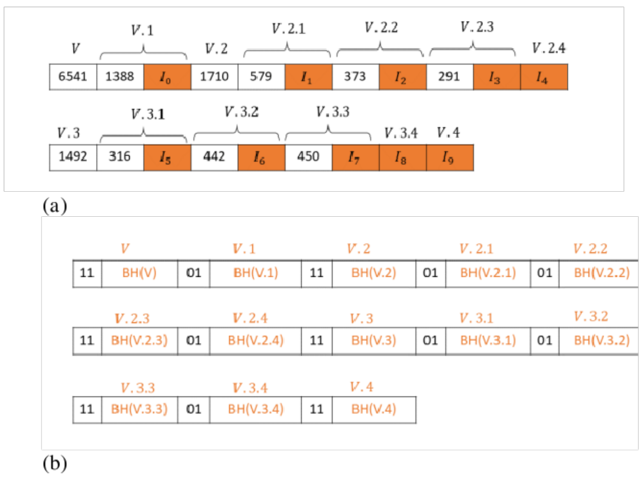

V
V.1
V.2
V.2.1
V.2.2
V.2.3
V.2.4
6541
1388
I0
1710
579
I1
373
I2
291
I3
I4
V.3
V.3.1
V.3.2
V.3.3
V.3.4
V.4
1492
316
I5
442
I6
450
I7
I8
I9
(a)
11 BH(V) 01 BH(V.1) 11 BH(V.2) 01 BH(V.2.1) 01 BH(V.2.2)
11 BH(V.2.3) 01 BH(V.2.4) 11 BH(V.3) 01 BH(V.3.1) 01 BH(V.3.2)
11 BH(V.3.3) 01 BH(V.3.4) 11 BH(V.4)
(b)

Figure 7: *Sum Array* (*a*) and *Structure Array* (*b*) of the IQTS

There are three Indexing methods proposed to select the best approach to approximate the bucket's data distribution: (i) the 2/3LT-Index, is appropriate if the distributions don't contain strong asymmetry; (ii) the 2/4LT-Index, is appropriate if the distributions are biased; (iii) the 2/p(eak)LT-Index, is appropriate if the distributions contain a few high-density peaks. The purpose of the Index is approximating the aggregate (sum) of the data existing in the sub-buckets and sub-sub-buckets. Finally, after performing above steps the Sum Array and Structure Array are prepared to be stored in memory as shown in Fig. 7 for the aforementioned OLAP view example.

Finally, the information about the values of the nodes and the structure of partitioned quad-trees is stored in the memory as shown in Fig. 7. The Sum Array contains the sum value of each node if it is non-zero besides the Index that is an approximation of the data inside sub-buckets.

The Structure Array represents the partitioning process of the two-dimensional OLAP view V, for the node n_j corresponding to bucket p_j, using (i) two bits assigned for Topology Header, $TH(p_j)$, that the first bit denotes whether nod e n_j is a leaf or not; and the second bit represents whether the sum value is zero or not; (ii) Bound Header, is a number of bits designated by $BH(p_j)$ to represent the partitioning procedure of each buckets. The bounds of the child buckets will be represented by the splitting position $N_{j,k}$ on the interval $[l_{j,k}: r_{j,k}]$ of p_j, using a signed offset $O_{j,k}$ which the sign is respect to the $H_{j,k}$.

As mentioned before, IQTS also provides support for approximate query answering over the compressed OLAP views. Range queries are evaluated on compressed OLAP views by performing linear interpolation, i.e. by assuming that data distribution inside each bucket is uniform (i.e., the Continuous Value Assumption (CVA) [19]); as a consequence, the contribution of the buckets that partially overlap the ranges of the input query is generally approximate, unless the original distribution of frequencies inside these buckets is actually uniform.

Our query algorithm navigates the quad-tree searching for: (i) all maximal buckets which are completely involved in the query, and (ii) all buckets which are partially included in the range of the query, and which are not split (such buckets correspond to leaves of the quad-tree (i.e., the terminal buckets). The former buckets do not introduce any approximation, since for each of them the exact value of the sum inside the corresponding ranges is stored. On the contrary, the latter ones introduce approximation, since we cannot re-construct the exact distribution of the original data contained in the corresponding ranges. It should be noted here that, dealing with approximation in query answering, is a significant research challenge (e.g., [20]).

Linear interpolation is used to answer overlapping queries against a bucket b in both the cases that b is equipped with an Index or not. In more detail, given a bucket b and a range query Q overlapping b (i.e., $|Q \cap b| \neq \emptyset$), if b is not equipped with an Index, we compute the approximate answer to Q on b $\tilde{A}(Q)$ by applying the CVA:

$$\tilde{A}(Q) = \frac{|Q \cap b|}{|b|} \cdot S(b) \tag{6}$$

such that S(b) is the sum stored in b, $|Q \cap b|$ is the area on the intersection region between Q and b, and $|b|$ is the area of b.

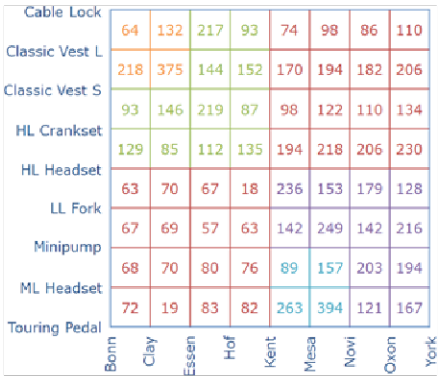
3. IQTS At Work!

In this Section, we provide an example of IQTS on a two-dimensional view obtained from the data cube SaleAnalyzer built on top the synthetic multidimensional database AdventureWorks, provided by Microsoft. This multidimensional database stores data produced by a fictitious large multi-national manufacturing company called, not by chance, Adventure Works Cycles. Over this multidimensional database, the data cube SaleAnalyzer is implemented. SaleAnalyzer stores the following information: (i) sales to retailers; (ii) data and related details of Internet sale orders to customers; (iii) shipped orders; (iv) data in USD, but keeping track about original currency. SaleAnalyzer is characterized by the fact table FactResellerSales that contains 60,855 facts related to sales performed by retailers. Other tables with interesting data that are related to FactResellerSales are the following: (i) FactInternetSales, which contains 60,398 facts related to InternetSales; (ii) FactInternetSalesReason that stores 64,515 facts related to the buying motivations. We obtained the two-dimensional view on SaleAnalyzer starting from the query reported in Fig. 8. by selecting as dimensions Geography and Product, and drilling down to the lower level in the hierarchies. On Geography, we chosen the cities, alphabetically, in the range from Bonn to York and, on Product, we selected the English product names, alphabetically, in the range from Cable Lock to Touring Pedal. We chose Total Product Cost as measure.

```
SELECT
{[DimGeography].[Hierarchy].[City].&[Bonn]:
[DimGeography].[Hierarchy].[City].&[York]}
       ON AXIS(0)
   {[DimProduct].[Hierarchy].[English Product
                Name].&[Cable Lock]:
   [DimProduct].[Hierarchy].[English Product
                Name].&[Touring Pedal]}
       ON AXIS(1),
FROM    SaleAnalyzer
WHERE   [Measures].[Total Product Cost]
```

Figure 8: Query on SaleAnalyzer from which the two-dimensional view is obtained

The query result shown in Fig. 9 represents the two-dimensional view. As shown in Fig. 9, the different colors denote the buckets splitting performed by our algorithm. It can be seen that red parts contain more homogeneous values than the remaining parts. Thus, green and purple buckets are further split. Between the two red buckets, the four green sub-buckets and the four purple ones, the highest variance is in the orange sub- bucket, which is split even more. Finally, the last splitting is performed in the blue sub- bucket.


Cable Lock
Classic Vest L
Classic Vest S
HL Crankset
HL Headset
LL Fork
Minipump
ML Headset
Touring Pedal
Bonn
Clay
Essen
Hof
Kent
Mesa
Novi
Oxon
York

64	132	217	93	74	98	86	110
218	375	144	152	170	194	182	206
93	146	219	87	98	122	110	134
129	85	112	135	194	218	206	230
63	70	67	18	236	153	179	128
67	69	57	63	142	249	142	216
68	70	80	76	89	157	203	194
72	19	83	82	263	394	121	167

Figure 9: Two-dimensional view on SaleAnalyzer originated by query shown in Fig. 8

The last procedure is better shown in Fig. 10. In order to make it clear, we use the same colors of the two-dimensional view (shown in Fig. 9). Firstly, the view is a single bucket holding the sum 8,890. Then, the partitioning starts by splitting it into four buckets. Thus, the procedure continues on those buckets that hold the highest variance and it stops when the maximum memory threshold set for compression is reached. According to the last statement, the larger available memory storage, the higher precision in the results.

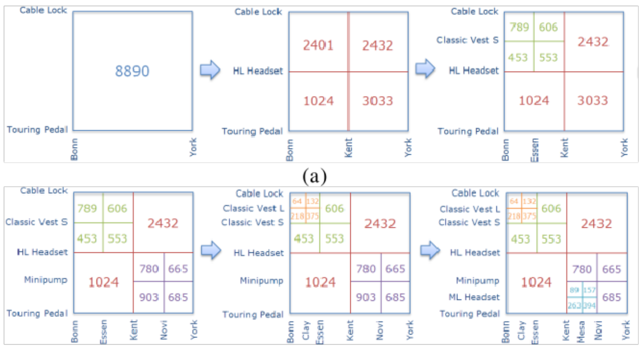


Figure 10: Two-dimensional IQTS compression process over the two-dimensional view on SaleAnalyzer shown in Fig. 9

Finally, the two secondary-memory data structures for obtaining the compressed representation of the partitioned representation of Fig. 10, described in Section 2, are generated.

4. Conclusions and Future Work

This paper has proposed the IQTS algorithm, which allows us to compress multidimensional OLAP views derived from big OLAP data cubes that populate Mobile Clouds. In addition to this, IQTS effectively and efficiently supports approximate query answering over compressed multidimensional OLAP views. These functionalities turn to be enabling functionalities for big data applications in a wide collection of modern scenarios, such as healthcare analytics, social network analysis, advanced recommendation systems, and so forth. We provided motivations, anatomy and procedures of IQTS, enriched by several case studies that contributed to pinpoint the benefits coming from our proposed algorithm.

Future work is mostly oriented towards extending our framework with innovative characteristics of big data, since big data research is conquering the scene in both industrial and academic settings. While there is a plethora of topics to touch, here we provide a short list of those that we retain as the most challenges during next future-years research efforts.

Among these, we pinpoint: (i) advanced machine learning (e.g., [21,22]); (ii) semantics-based techniques (e.g., [23,24]); (iii) explainability approaches (e.g., [25,26]); (iv) privacy-preservation methods over compressed data (e.g., [27,28,29]).

Acknowledgments

This work was funded by the Next Generation EU - Italian NRRP, Mission 4, Component 2, Investment 1.5 (Directorial Decree n. 2021/3277) - project Tech4You n. ECS0000009.

References

[1] Zichuan Xu, Wanli Gong, Qiufen Xia, Weifa Liang, Omer F. Rana, Guowei Wu, "NFV-Enabled IoT Service Provisioning in Mobile Edge Clouds". IEEE Trans. Mob. Comput. 20(5), pp. 1892-1906, 2021

[2] Hua Deng, Zheng Qin, Qianhong Wu, Zhenyu Guan, Hui Yin, "Revocable Attribute-Based Data Storage in Mobile Clouds". IEEE Trans. Serv. Comput. 15(2), pp. 1130-1142, 2022

[3] Zichuan Xu, Lizhen Zhou, Sid Chi-Kin Chau, Weifa Liang, Haipeng Dai, Lixing Chen, Wenzheng Xu, Qiufen Xia, Pan Zhou, "Near-Optimal and Collaborative Service Caching in Mobile Edge Clouds". IEEE Trans. Mob. Comput. 22(7), pp. 4070-4085, 2023

[4] Vijayakumar Varadarajan, Neelanarayanan Venkataraman, Ron Doyle, Imad Fakhri Alshaikhli, Sven Groppe, "Emerging Solutions in Big Data and Cloud Technologies for Mobile Networks". Mob. Networks Appl. 24(3), pp. 1015-1017, 2019

[5] Mohammad Shorfuzzaman, M. Shamim Hossain, Amril Nazir, Ghulam Muhammad, Atif Alamri, "Harnessing the Power of Big Data Analytics in the Cloud to Support Learning Analytics in Mobile Learning Environment". Comput. Hum. Behav. 92, pp. 578-588, 2019

[6] Tian Wang, Haoxiong Ke, Xi Zheng, Kun Wang, Arun Kumar Sangaiah, Anfeng Liu, "Big Data Cleaning Based on Mobile Edge Computing in Industrial Sensor-Cloud". IEEE Trans. Ind. Informatics 16(2), pp. 1321-1329, 2020

[7] Xiaoxu Ren, Chao Qiu, Xiaofei Wang, Zhu Han, Ke Xu, Haipeng Yao, Song Guo, "AI-Bazaar: A Cloud-Edge Computing Power Trading Framework for Ubiquitous AI Services". IEEE Trans. Cloud Comput. 11(3), pp. 2337-2348, 2023

[8] Dongfen Li, Jinshan Lai, Ruijin Wang, Xiong Li, Pandi Vijayakumar, Brij B. Gupta, Wadee Alhalabi, "Ubiquitous Intelligent Federated Learning Privacy-Preserving Scheme under Edge Computing". Future Gener. Comput. Syst. 144, pp. 205-218, 2023

[9] Hao Jiang, Lixia Li, Haoran Xian, Yulin Hu, Hehe Huang, Juzhen Wang, "Crowd Flow Prediction for Social Internet-of-Things Systems Based on the Mobile Network Big Data". IEEE Trans. Comput. Soc. Syst. 9(1), pp. 267-278, 2022

[10] Shinwoo Hyun, Seung Mo Yang, Junhwan Kim, Kwang Soo Kim, Jae Hoon Shin, Sang Min Lee, Byun-Woo Lee, Robert M. Beresford, David H. Fleisher, "Development of a Mobile Computing Framework to Aid Decision-Making on Organic Fertilizer Management Using a Crop Growth Model". Comput. Electron. Agric. 181, art. 105936, 2021

[11] Alfredo Cuzzocrea, "CAMS: OLAPing Multidimensional Data Streams Efficiently". In: 2009 DaWaK Int. Conf., pp. 48-62, 2009

[12] Alfredo Cuzzocrea, "Compressing Big OLAP Data Cubes in Big Data Analytics Systems: New Paradigms and Future Research Perspectives". In: 2022 ICSBT Int. Conf., p.7, 2022

[13] Elena N. Stankova, Andrey V. Balakshiy, Dmitry A. Petrov, Vladimir V. Korkhov, Andrey V. Shorov, "OLAP Technology and Machine Learning as the Tools for Validation of the Numerical Models of Convective Clouds". Int. J. Bus. Intell. Data Min. 14(1/2), pp. 254-266, 2019

[14] Carson K. Leung, Yubo Chen, Calvin S. H. Hoi, Siyuan Shang, Alfredo Cuzzocrea, "Machine Learning and OLAP on Big COVID-19 Data". In: IEEE BigData 2020 Int. Conf., pp. 5118-5127, 2020

[15] Vinayakumar Ravi, Tuan D. Pham, Mamoun Alazab, "Attention-Based Multidimensional Deep Learning Approach for Cross-Architecture IoMT Malware Detection and Classification in Healthcare Cyber-Physical Systems". IEEE Trans. Comput. Soc. Syst. 10(4), pp. 1597-1606, 2023

[16] Alfredo Cuzzocrea, Filippo Furfaro, Domenico Saccà, "Enabling OLAP in Mobile Environments via Intelligent Data Cube Compression Techniques". J. Intell. Inf. Syst. 33(2), pp. 95-143, 2009

[17] Katrina E. Barkwell, Alfredo Cuzzocrea, Carson K. Leung, Ashley A. Ocran, Jennifer M. Sanderson, James Ayrton Stewart, Bryan H. Wodi, "Big Data Visualisation and Visual Analytics for Music Data Mining". In: IEEE IV 2018 Int. Conf., pp. 235-240, 2018

[18] Francesco Buccafurri, Filippo Furfaro, Giuseppe Massimo Mazzeo, Domenico Saccà, "A Quad-Tree Based Multiresolution Approach for Two-Dimensional Summary Data," Information Systems 36(7), pp. 1082-1103, 2011

[19] G. Colliat, "OLAP, Relational, and Multidimensional Database Systems," ACM Sigmod Record 25(3), pp.64-69, 1996

[20] Alfredo Cuzzocrea, "Retrieving Accurate Estimates to OLAP Queries over Uncertain and Imprecise Multidimensional Data Streams". In: ACM SSDBM 2011 Int. Conf., pp. 575-576, 2021

[21] Rocco Langone, Alfredo Cuzzocrea, Nikolaos Skantzos, "Interpretable Anomaly Prediction: Predicting Anomalous Behavior in Industry 4.0 Settings via Regularized Logistic Regression Tools". Data Knowl. Eng. 130, art. 101850, 2020

[22] Xiaoming Li, Dan Zhang, Ye Zheng, Wuyang Hong, Weixi Wang, Jizhe Xia, Zhihan Lv, "Evolutionary Computation-Based Machine Learning for Smart City High-Dimensional Big Data Analytics". Appl. Soft Comput. 133, art. 109955, 2023

[23] Alfredo Cuzzocrea, Domenico Saccà, Paolo Serafino, "Semantics-Aware Advanced OLAP Visualization of Multidimensional Data Cubes". Int. J. Data Warehous. Min. 3(4), pp. 1-30, 2007

[24] Dongxiao He, Yanli Wu, Youyou Wang, Zhizhi Yu, Zhiyong Feng, Xiaobao Wang, Yuxiao Huang, "Identification of Communities with Multi-Semantics via Bayesian Generative Model". IEEE Trans. Big Data 8(4), pp. 869-881, 2022

[25] Mohanad Sarhan, Siamak Layeghy, Marius Portmann, “Evaluating Standard Feature Sets Towards Increased Generalisability and Explainability of ML-Based Network Intrusion Detection”. Big Data Res. 30, art. 100359, 2022

[26] Gaurav Duggal, Tejas Gaikwad, Bhupendra Sinha, “Dependable Modulation Classifier Explainer with Measurable Explainability”. Frontiers Big Data 5, 2022

[27] Ayman Ibaida, Alsharif Abuadbba, Naveen K. Chilamkurti, “Privacy-Preserving Compression Model for Efficient IoMT ECG Sharing”. Comput. Commun. 166, pp. 1-8, 2021

[28] Kamalika Chaudhuri, Chuan Guo, Mike Rabbat, “Privacy-Aware Compression for Federated Data Analysis”. In: UAI 2022 Int. Conf., pp. 296-306, 2022

[29] Francesco Taurone, Daniel E. Lucani, Qi Zhang, “Zip to Zip-It: Compression to Achieve Local Differential Privacy”. CoRR abs/2308.14627, 2023

[30] Kun Xie, Xueping Ning, Xin Wang, Shiming He, Zuoting Ning, Xiaoxiao Liu, Jigang Wen, Zheng Qin, “An Efficient Privacy-Preserving Compressive Data Gathering Scheme in WSNs”. Inf. Sci. 390, pp. 82-94, 2017

[31] Alfredo Cuzzocrea, Mojtaba Hajian, “Effective and Efficient Big OLAP Data Cube Compression in Mobile Cloud Environments: The IQTS Algorithm”. In: IEEE Big Data 2023 Int. Conf., pp. 5187-5196, 2023