Graph OLAP: Towards Online Analytical Processing on Graphs*

Chen Chen[1] Xifeng Yan[2] Feida Zhu[1] Jiawei Han[1] Philip S. Yu[3]

[1]University of Illinois at Urbana-Champaign
{cchen37, feidazhu, hanj}@cs.uiuc.edu
[2]IBM T. J. Watson Research Center
xifengyan@us.ibm.com
[3]University of Illinois at Chicago
psyu@cs.uic.edu

Abstract

OLAP (On-Line Analytical Processing) is an important notion in data analysis. Recently, more and more graph or networked data sources come into being. There exists a similar need to deploy graph analysis from different perspectives and with multiple granularities. However, traditional OLAP technology cannot handle such demands because it does not consider the links among individual data tuples. In this paper, we develop a novel graph OLAP framework, *which presents a* multi-dimensional *and* multi-level *view over graphs.*

The contributions of this work are two-fold. First, starting from basic definitions, i.e., what are dimensions *and* measures *in the graph OLAP scenario, we develop a conceptual framework for data cubes on graphs. We also look into different semantics of OLAP operations, and classify the framework into two major subcases:* informational OLAP *and* topological OLAP. *Then, with more emphasis on informational OLAP (topological OLAP will be covered in a future study due to the lack of space), we show how a graph cube can be materialized by calculating a special kind of measure called* aggregated graph *and how to implement it efficiently. This includes both full materialization and partial materialization where constraints are enforced to obtain an iceberg cube. We can see that the aggregated graphs, which depend on the graph properties of underlying networks, are much harder to compute than their traditional OLAP counterparts, due to the increased structural complexity of data. Empirical studies show insightful results on real datasets and demonstrate the efficiency of our proposed optimizations.*

*The work was supported in part by the U.S. National Science Foundation grants IIS-08-42769 and BDI-05-15813, Office of Naval Research (ONR) grant N00014-08-1-0565, and NASA grant NNX08AC35A.

1 Introduction

OLAP (On-Line Analytical Processing) [9, 5, 20, 2, 10] is an important notion in data analysis. Given the underlying data, a cube can be constructed to provide a *multi-dimensional* and *multi-level* view, which allows for effective analysis of the data from different perspectives and with multiple granularities. The key operations in an OLAP framework are slice/dice and roll-up/drill-down, with slice/dice focusing on a particular aspect of the data, roll-up performing generalization if users only want to see a concise overview, and drill-down performing specialization if more details are needed.

In a traditional data cube, a data record is associated with a set of dimensional values, whereas different records are viewed as *mutually independent*. Multiple records can be summarized by the definition of corresponding aggregate measures such as COUNT, SUM, and AVERAGE. Moreover, if a concept hierarchy is associated with each attribute, multi-level summaries can also be achieved. Users can navigate through different dimensions and multiple hierarchies via roll-up, drill-down and slice/dice operations. However, in recent years, more and more data sources beyond conventional spreadsheets have come into being, such as chemical compounds or protein networks (chem/bio-informatics), 2D/3D objects (pattern recognition), circuits (computer-aided design), loosely-schemaed data (XML), and social or informational networks (Web), where not only individual entities but also the *interacting relationships* among them are important and interesting. This demands a new generation of tools that can manage and analyze such data.

Given their great expressive power, graphs have been widely used for modeling a lot of datasets that contain structure information. With the tremendous amount of graph data accumulated in all above applications, the same need to deploy analysis from different perspectives and with mul-

tiple granularities exists. To this extent, our main task in this paper is to develop a *graph OLAP framework*, which presents a multi-dimensional and multi-level view over graphs.

In order to illustrate what we mean by "graph OLAP" and how the OLAP glossary is interpreted with regard to this new scenario, let us start from a few examples.

Example 1 (Collaboration Patterns). There are a set of authors working in a given field: For any two persons, if they coauthor w papers in a conference, *e.g.*, SIGMOD 2004, then a link is added between them, which has an collaboration frequency attribute that is weighted as w. For every conference in every year, we may have a coauthor network describing the collaboration patterns among researchers, each of them can be viewed as a snapshot of the overall coauthor network in a bigger context.

It is interesting to analyze the aforementioned graph dataset in an OLAP manner. First, one may want to check the collaboration patterns for a group of conferences, say, all DB conferences in 2004 (including SIGMOD, VLDB, ICDE, *etc.*) or all SIGMOD conferences since its inauguration. In the language of data cube, with a *venue* dimension and a *time* dimension, one may choose to obtain the *(db-conf, 2004)* cell and the *(sigmod, all-years)* cell, where the *venue* and *time* dimensions have been generalized to *db-conf* and *all-years*, respectively. Second, for the subset of snapshots within each cell, one can summarize them by computing a measure as we did in traditional OLAP. In the graph context, this gives rise to an *aggregated graph*. For example, a summary network displaying total collaboration frequencies can be achieved by overlaying all snapshots together and summing up the respective edge weights, so that each link now indicates two persons' collaboration activities in the DB conferences of 2004 or during the whole history of SIGMOD. ■

The above example is simple because the measure is calculated by a simple sum over individual pieces of information. A more complex case is presented next.

Example 2 (Maximum Flow). Consider a set of cities connected by transportation networks. In general, there are many ways to go from one city A to another city B, *e.g.*, by car, by train, by air, by water., *etc.*, and each way is operated by multiple companies. For example, we can assume that the capacity of company x's air service from A to B is c^x_{AB}, *i.e.*, company x can transport at most c^x_{AB} units from A to B using the planes it owns. Finally, we get a snapshot of capacity network for every service of every company.

Now, consider the transporting capability from a source city S to a destination city T, it is interesting to see how this value can be achieved by sending flows via different paths if 1) we only want to go by air, or 2) we only want to choose services operated by company x. In the OLAP language, with a *transportation-type* dimension and a *company* dimension, the above situations correspond to the *(air, all-companies)* cell and the *(all-types, company x)* cell, while the *measure* computed for a cell c is a graph displaying the overall maximum flow, which has considered all types and companies that are allowed in c. Unlike Example 1, computing the aggregated graph is now a much harder task; also, the semantics associated with un-aggregated network snapshots and aggregated graphs are different: The former shows capacities on its edges, whereas the latter shows transmitted flows, which must be smaller by its definition. ■

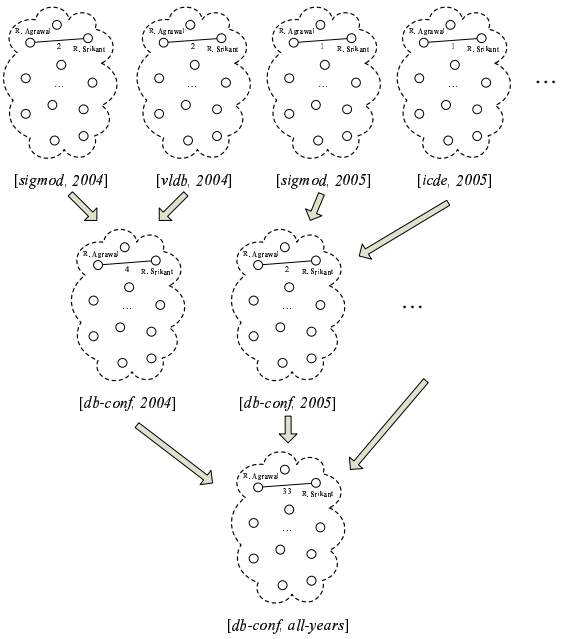

Figure 1: The OLAP Scenario for Example 1

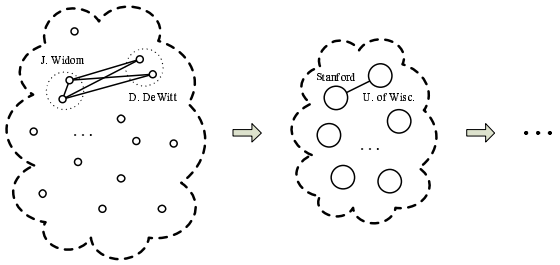

Figure 2: The OLAP Scenario for Example 3

Example 3 (Collaboration Patterns, Revisited). Usually, the whole coauthor network could be too big to comprehend, and thus it is desirable to look at a more compressed view. For example, one may like to see the collaboration activities organized by the authors' associated affiliations, which requires the network to be generalized one step up, *i.e.*, merg-

ing all persons in the same institution as one node and constructing a new summary graph at the institution level. In this "generalized network", for example, an edge between Stanford and University of Wisconsin will aggregate all collaboration frequencies occurred between Stanford authors and Wisconsin authors. Similar to Examples 1 and 2, an aggregated graph (*i.e.*, the generalized network defined above) is taken as the OLAP measure. However, the difference here is that a roll-up from the individual level to the institution level is achieved by consolidating multiple nodes into one, which shrinks the whole graph. Compared to this, in Examples 1 and 2, the graph is not collapsed because we are always examining the relationships among the same set of objects – it poses minimum changes with regard to network topology upon generalization. ■

The above examples demonstrate that OLAP provides a powerful primitive to analyze graph datasets. In this paper, we will give a systematic study on graph OLAP, which is more general than traditional OLAP: In addition to individual entities, the mutual interactions among them are also taken into consideration. Our major contributions are summarized as below.

- **For conceptual modeling**, a *graph OLAP framework* is developed, which defines *dimensions* and *measures* in the graph context, as well as the concept of *multi-dimensional* and *multi-level* analysis over the underlying networked data. We distinguish different semantics of OLAP operations and categorize them into two major subcases: *informational OLAP* (as shown in Examples 1 and 2) and *topological OLAP* (as shown in Example 3). It is necessary since these two kinds of OLAP demonstrate substantial differences with regard to the construction of a graph cube.
- **For efficient implementation**, the computation of aggregated graphs as graph OLAP measures is examined. Due to the increased structural complexity of data, calculating certain measures that are closely tied with the graph properties of a network, *e.g.*, maximum flow and centrality, poses greater challenges than their traditional OLAP counterparts, such as COUNT, SUM and AVERAGE. We investigate this issue, categorize measures based on the difficulty to compute them in the OLAP context and suggest a few measure properties that may help optimize the processing further. Both full materialization and partial materialization (where constraints are enforced to obtain an iceberg cube) are discussed.

The remaining of this paper is organized as follows. In Section 2, we formally introduce the graph OLAP framework. Section 3 discusses the general hardness to compute aggregated graphs as graph OLAP measures, which categorizes them into three classes. Section 4 looks into some properties of the measures and proposes a few further optimizations. Constraints and partial materialization are studied in Section 5. We report experiment results and related work in Sections 6 and 7, respectively. Section 8 concludes this study.

2 A Graph OLAP Framework

In this section, we present the general framework of graph OLAP.

Definition 1 *(Graph Model). We model the data examined by graph OLAP as* a collection of network snapshots $\mathcal{G} = \{\mathbb{G}_1, \mathbb{G}_2, \ldots, \mathbb{G}_N\}$*, where each snapshot* $\mathbb{G}_i = (I_{1,i}, I_{2,i}, \ldots, I_{k,i}; G_i)$ *in which* $I_{1,i}, I_{2,i}, \ldots, I_{k,i}$ *are* k informational attributes *describing the snapshot as a whole and* $G_i = (V_i, E_i)$ *is a* graph*. There are also* node attributes *attached with any* $v \in V_i$ *and* edge attributes *attached with any* $e \in E_i$*. Note that, since* $\mathbb{G}_1, \mathbb{G}_2, \ldots, \mathbb{G}_N$ *only represent different observations,* $V_1, V_2, \ldots, V_N$ *actually correspond to the same set of objects in real applications.*

For instance, with regard to the coauthor network described in the introduction, *venue* and *time* are two informational attributes that mark the status of individual snapshots, *e.g.*, SIGMOD 2004 and ICDE 2005, *authorID* is a node attribute indicating the identification of each node, and *collaboration frequency* is an edge attribute reflecting the connection strength of each edge.

Dimension and *measure* are two concepts that lay the foundation of OLAP and cubes. As their names imply, first, dimensions are used to construct a cuboid lattice and partition the data into different cells, which act as the basis for multi-dimensional and multi-level analysis; second, measures are calculated to aggregate the data covered, which deliver a summarized view of it. In below, we are going to formally re-define these two concepts for the graph OLAP scenario.

Let us examine dimensions at first. Actually, there are two types of dimensions in graph OLAP. The first one, as exemplified by Example 1, utilizes informational attributes attached at the whole snapshot level. Suppose the following concept hierarchies are associated with *venue* and *time*:

- *venue*: *conference* → *area* → *all*,
- *time*: *year* → *decade* → *all*;

the role of these two dimensions is to organize snapshots into groups based on different perspectives and granularities, *e.g.*, *(db-conf, 2004)* and *(sigmod, all-years)*, where each of these groups corresponds to a "cell" in OLAP terminology. They control what snapshots are to be looked at, but they do not touch the inside of any single snapshot.

Definition 2 *(Informational Dimensions). With regard to the graph model presented in Definition 1, the set of informational attributes* $\{I_1, I_2, \ldots, I_k\}$ *are called the* informational dimensions *of graph OLAP, or* Info-Dims *in short.*

The second type of dimensions are provided to operate on nodes and edges within individual networks. Take Example 3 for instance, suppose the following concept hierarchy

- *authorID*: *individual* $\rightarrow$ *institution* $\rightarrow$ *all*

is associated with the node attribute *authorID*, then it can be used to group authors from the same institution into a "generalized" node, and a new graph thus resulted will depict interactions among these groups as a whole, which summarizes the original network and hides specific details.

Definition 3 *(Topological Dimensions). The set of dimensions coming from the attributes of topological elements (i.e., nodes and edges of G_i), $\{T_1, T_2, \ldots, T_l\}$, are called the* topological dimensions *of graph OLAP, or* Topo-Dims *in short.*

The OLAP semantics accomplished through Info-Dims and Topo-Dims are rather different, and in the following we shall refer to them as *informational OLAP* (abbr. *I-OLAP*) and *topological OLAP* (abbr. *T-OLAP*), respectively.

For roll-up in **I-OLAP**, the characterizing feature is that, snapshots are just different observations of the same underlying network, and thus when they are all grouped into one cell in the cube, it is like *overlaying* multiple pieces of information, *without changing* the objects whose interactions are being looked at.

For roll-up in **T-OLAP**, we are no longer grouping snapshots, and the reorganization switches to happen inside individual networks. Here, *merging* is performed internally which "zooms out" the user's focus to a "generalized" set of objects, and a new graph formed by such *shrinking* might greatly alter the original network's topological structure.

Now we move on to measures. Remember that, in traditional OLAP, a measure is calculated by aggregating all the data tuples whose dimensions are of the same values (based on concept hierarchies, such values could range from the finest un-generalized ones to "all/*", which form a multi-level cuboid lattice); casting this to our scenario here:

First, in graph OLAP, the aggregation of graphs should also take the form of a graph, *i.e.*, an *aggregated graph*. In this sense, graph can be viewed as a special kind of measure, which plays a dual role: as a data source and as a special (aggregated) measure. Of course, other measures that are not graphs, such as node count, average degree, diameter, *etc.*, can also be calculated; however, we do not explicitly include such non-graph measures in our model, but instead treat them as derived from corresponding graph measures.

Second, due to the different semantics of I-OLAP and T-OLAP, aggregating data with identical Info-Dim values groups information among the snapshots, whereas aggregating data with identical Topo-Dim values groups topological elements inside individual networks. As a result, we will give a separate measure definition for each case in below.

Definition 4 *(I-Aggregated Graph). With regard to Info-Dims $\{I_1, I_2, \ldots, I_k\}$, the* I-aggregated graph M^I *is an* attributed graph *that can be computed based on a set of network snapshots $\mathcal{G}' = \{\mathbb{G}_{i_1}, \mathbb{G}_{i_2}, \ldots, \mathbb{G}_{i_{N'}}\}$ whose Info-Dims are of identical values; it satisfies: 1) the nodes of M^I are as same as any snapshot in $\mathcal{G}'$, and 2) the node/edge attributes attached to M^I are calculated as* aggregate functions *of the node/edge attributes attached to $G_{i_1}, G_{i_2}, \ldots, G_{i_{N'}}$.*

The graph in Figure 1 that describes collaboration frequencies among individual authors for a particular group of conferences during a particular period of time is an instance of I-aggregated graph, and the interpretation of classic OLAP operations in graph I-OLAP is summarized as follows.

- Roll-up: Overlay multiple snapshots to form a higher-level summary via I-aggregated graph.
- Drill-down: Return to the set of lower-level snapshots from the higher-level overlaid (aggregated) graph.
- Slice/dice: Select a subset of qualifying snapshots based on Info-Dims.

Definition 5 *(T-Aggregated Graph). With regard to Topo-Dims $\{T_1, T_2 \ldots, T_l\}$, the* T-aggregated graph M^T *is an* attributed graph *that can be computed based on an individual network G_i; it satisfies: 1) the nodes of G_i with identical values on their Topo-Dims are grouped, whereas each group corresponds to a node in M^T, 2) the attributes attached to M^T are calculated as* aggregate functions *of the attributes attached to G_i.*

The graph in Figure 2 that describes collaboration frequencies among institutions is an instance of T-aggregated graph, and the interpretation of classic OLAP operations in graph T-OLAP is summarized as follows.

- Roll-up: Shrink the topology and obtain a T-aggregated graph that displays a compressed view, whose topological elements (*i.e.*, nodes and/or edges) have been merged and replaced by corresponding higher-level ones.
- Drill-down: A reverse operation of roll-up.
- Slice/dice: Select a subgraph of the network based on Topo-Dims.

3 Measure Classification

Now, with a clear concept of dimension, measure and possible OLAP operations, we are ready to discuss implementation issues, *i.e.*, how to compute the aggregated graph in a multi-dimensional and multi-level way.

Recall that in traditional OLAP, measures can be classified into *distributive*, *algebraic* and *holistic*, depending on whether the measures of high level cells can be easily computed from their low level counterparts, without accessing base tuples residing at the finest level. For instance, in the classic *sale*(*time*, *location*) example, the total sale of [*2008*, *California*] can be calculated by adding up the total sales of [*January 2008*, *California*], [*February 2008*, *California*], . . ., [*December 2008*, *California*], without looking at base data points such as [*04/12/2008*, *Los Angeles*], which means that SUM is a distributive measure. Compared to this, AVG is often cited as an algebraic measure, which is actually a *semi-distributive* category in that AVG can be derived from two distributive measures: SUM and COUNT, *i.e.,* algebraic measures are functions of distributive measures.

(Semi-)distributiveness is a nice property for top-down cube computation, where the cuboid lattice can be gradually filled up by making level-by-level aggregations. Measures without this property is put into the holistic category, which is intuitively much harder to calculate. Now, considering graph OLAP, based on similar criteria with regard to the aggregated graphs, we can also classify them into three categories.

Definition 6 *(Distributive, Algebraic and Holistic). Consider a high level cell c_h and the corresponding low level cells it covers: $c_l^1, c_l^2, \ldots$. An aggregated graph M_d is* distributive *if $M_d(c_h)$ can be directly computed as a function of $M_d(c_l^1), M_d(c_l^2), \ldots$, i.e.,*

$$M_d(c_h) = F_d\big[M_d(c_l^1), M_d(c_l^2), \ldots\big].$$

For a non-distributive aggregated graph M_a, if it can be derived from some other distributive aggregated graphs $M_d^1, M_d^2, \ldots$, i.e., for $\forall c_h$,

$$M_d(c_h) = F_a\big[M_d^1(c_h), M_d^2(c_h), \ldots\big],$$

then we say that it is algebraic. *Aggregated graphs that are neither distributive nor algebraic belong to the* holistic *category.*

If we further consider the differences between I-OLAP and T-OLAP, there are actually six classes: I-distributive, I-algebraic, I-holistic and T-distributive, T-algebraic, T-holistic. Due to limited space, in the remaining of this paper, we shall put more emphasis on I-OLAP; discussions for T-OLAP will be covered in a future study.

I-distributive. The I-aggregated graph describing collaboration frequency in Example 1 is an I-distributive measure, because the frequency value in a high level I-aggregated graph can be calculated by simply adding those in corresponding low level I-aggregated graphs.

I-algebraic. The graph displaying maximum flow in Example 2 is an I-algebraic measure based on the following reasoning. Suppose *transportation-type* and *company* comprise the two dimensions of the cube, and we generalize from low level cells *(all-types, company 1)*, *(all-types, company 2)*, . . . to *(all-types, all-companies)*, *i.e.,* compute the maximum flow based on all types of transportation operated by all companies. Intuitively, this overall maximum flow would not be a simple sum (or other indirect manipulations) of these companies' individual maximum flows. For instance, company 1 may have excessive transporting capability between two cities, whereas the same link happens to be a bottleneck for company 2: Considering both companies simultaneously for determination of the maximum flow can enable capacity sharing and thus create a double-win situation. In this sense, maximum flow is not distributive by definition. However, as an obvious fact, maximum flow f is determined by the network c that shows link capacities on its edges, and this capacity graph is distributive because it can be directly added upon generalization: When link sharing is enabled, two companies having two separate links from A to B with capacity c_{AB}^1 and c_{AB}^2 is no different from a single link with capacity $c_{AB}^1 + c_{AB}^2$ for maximum flow calculation. Finally, being a function of distributive aggregated graphs, maximum flow is algebraic.

I-holistic. The I-holistic case involves a more complex aggregate graph, where base level details are required to compute it. In the coauthor network of Example 1, the median of researchers' collaboration frequency for all DB conferences from 1990 to 1999 is holistic, similar to what we saw in a traditional data cube.

4 Optimizations

Being (semi-)distributive or holistic tells us whether the aggregated graph computation needs to start from completely un-aggregated data or some intermediate results can be leveraged. However, even if the aggregated graph is distributive or algebraic, and thus we can calculate high level measures based on some intermediate level ones, it is far from enough, because the complexity to compute the two functions F_d and F_a in Definition 6 is another question. Think about the maximum flow example we just mentioned, F_a takes the distributive capacity graph as input to compute the flows, which is by no means an easy transformation.

Based on our analysis, there are mainly two reasons for such potential difficulties. First, due to the interconnecting nature of graphs, the computation of many graph properties is "global" as it requires us to take the whole network into consideration. In order to make this concept of globalness clear, let us first look at a "local" case: For I-OLAP, aggregated graphs are built for the same set of objects as the underlying network snapshots; now, in the aggregated graph of Example 1, "R. Agrawal" and "R. Srikant"'s collaboration frequency for cell *(db-conf, 2004)* is locally determined by "R. Agrawal" and "R. Srikant"'s collaboration frequency

for each of the DB conferences held in 2004; it does not need any information from other authors to fulfill the computation. This is an ideal scenario, because the calculations are localized and thus greatly simplified. Unfortunately, not all measures provide such local properties. For instance, in order to calculate a maximum flow from S to T for the cell *(air, all-companies)* in Example 2, only knowing the transporting capability of each company's direct flights between S and T is not enough, because we can always take an indirect route via some other cities to reach the destination.

Second, the purpose for us to compute high level aggregated graphs based on low level ones is to reuse the intermediate calculations that are already performed. However, when computing the aggregated graph of a low level cell c_l^i, we only had a partial view about the c_l^i-portion of network snapshots; now, as multiple pieces of information are overlaid into the high level cell c_h, some full-scale consolidation needs to be performed, which is very much like the merge sort procedure, where partial ranked lists are reused but somehow adjusted to form a full ranked list. Still, because of the structural complexity, it is not an easy task to develop such reuse schemes for graphs, and even it is possible, reasonably complicated operations might be involved.

Admitting the above difficulties, let us now investigate the possibility to alleviate them. As we have seen, for some aggregated graphs, the first aspect can be helped by their *localization* properties with regard to network topology. Concerning the second aspect, the key is how to effectively reuse partial results computed for intermediate cells so that the workload to obtain a full-scale measure is *attenuated* as much as possible. In the following, we are going to examine these two directions in sequel.

4.1 Localization

Definition 7 *(Localization). For an I-OLAP aggregated graph M_l that summarizes a group of network snapshots $\mathcal{G}' = \{\mathbb{G}_{i_1}, \mathbb{G}_{i_2}, \ldots, \mathbb{G}_{i_{N'}}\}$, if 1) we only need to check a* neighborhood *of v in $G_{i_1}, G_{i_2}, \ldots, G_{i_{N'}}$ to calculate v's node attributes in M_l, and 2) we only need to check a* neighborhood *of u, v in $G_{i_1}, G_{i_2}, \ldots, G_{i_{N'}}$ to calculate (u, v)'s edge attributes in M_l, then the computation of M_l is said to be* localizable.

Example 4 (Common Friends). With regard to the coauthor network depicted in Example 1, we can also compute the following aggregated graph: Given two authors a_1 and a_2, the edge between them records the number of their common "friends", whereas in order to build such "friendship", the total collaboration frequency between a_1 and a_2 must surpass a δ_c threshold for the selected conferences and time. ■

The above example provides another instance that leverages localization to promote efficient processing. Consider a cell, *e.g., (db-conf, 2004)*, the determination of the aggregated graph's a_1-a_2 edge can be restricted to a 1-neighborhood of these two authors in the un-aggregated snapshots of 2004's DB conferences, *i.e.,* we only need to check edges that are directly adjacent to either a_1 or a_2, and in this way a third person a_3 can be found, if he/she publishes with both a_1 and a_2, while the total collaboration frequencies summed from the weights of these adjacent edges are at least δ_c. Also, note that, the above aggregated graph definition is based on the number of length-2 paths like a_1-a_3-a_2 where each edge of it represents a "friendship" relation; now, if we further increase the path length to k, computations can still be localized in a $\lfloor \frac{k}{2} \rfloor$-neighborhood of both authors, *i.e.,* any relevant author on such paths of "friendship" must be reachable from either a_1 or a_2 within $\lfloor \frac{k}{2} \rfloor$ steps. This can be seen as a situation that sits in the middle of Example 1's "absolute locality" (0-neighborhood) and maximum flow's "absolute globality" (∞-neighborhood).

There is an interesting note we want to put for the *absolutely local distributive* aggregated graph of Example 1. Actually, such a 0-neighborhood localization property degenerates the scenario to a very special case, where it is no longer necessary to assume the underlying data as a graph: For each pair of coauthors, we can construct a traditional cube showing their collaboration frequency "OLAPed" with regard to *venue* and *time*, whose computation does not depend on anything else in the coauthor network. In this sense, we can treat the coauthor network as a virtual union of pairwise collaboration activities, whereas Example 1 can indeed be thought as a traditional OLAP scenario disguised under its graph appearances, since the graph cube we defined is nothing different from a collection of pair-wise traditional cubes. As a desirable side effect, this enables us to leverage specialized technologies that are developed for traditional OLAP, which in general could be more efficient. But after all, the case is special anyway, most graph measures would not hold such absolute localization, which is also the reason why traditional OLAP proves to be extremely restrictive when handling networked data.

4.2 Attenuation

In below, we are going to explain the idea of attenuation through examples, and the case we pick is maximum flow. In a word, *the more partial results from intermediate calculations are utilized, the more we can decrease the cost of obtaining a full-scale aggregated graph*.

To begin with, let us first review some basic concepts, cf. [6]. Given a directed graph $G = (V, E)$, $c : \binom{V}{2} \to R^{\geq 0}$ indicates a *capacity* for all pairs of vertices and E is precisely the set of vertex pairs for which $c > 0$. For a source node s and a destination node t, a *flow* in G is a function $f : \binom{V}{2} \to R$ assigning values to graph edges such that, (i) $f(u, v) = -f(v, u)$: skew symmetry, (ii) $f(u, v) \leq c(u, v)$: capacity constraint, and (iii) for each

$v \neq s/t, \sum_{u \in V} f(u,v) = 0$: flow conservation. Since most maximum flow algorithms work incrementally, there is an important lemma as follows.

Lemma 1 *Let f be a flow in G and let G_f be its residual graph, where* residual *means that the capacity function of G_f is $c_f = c - f$; f' is a maximum flow in G_f if and only if $f + f'$ is a maximum flow in G.*

Note that, the $+/-$ notation here means edge-by-edge addition/subtraction; and in summary, this lemma's core idea is to look for a flow f' in G_f and use f' to *augment* the current flow f in G.

For the graph OLAP context we consider, in order to compute the algebraic aggregated graph displaying maximum flow, the function F_a takes a distributive capacity graph c as its input; now, since capacity can be written as the sum of a flow and a residual graph: $c = f + c_f$, does this decomposition provide us some hints to pull out the useful part f, instead of blindly taking c and starting from scratch?

Suppose that the capacity graph of cell *(all-types, company 1)* is c^1, where f_1 is the maximum flow and $c^1_{f_1} = c^1 - f_1$ denotes the corresponding residual graph. Likewise, we have c^2, f_2 and $c^2_{f_2}$ for cell *(all-types, company 2)*. Without loss of generality, assume there are only these two companies whose transportation networks are overlaid into *(all-types, all-companies)*, which has a capacity of $c = c^1 + c^2$.

Claim 1 *$f_1 + f_2 + f'$ is a maximum flow for c if and only if f' is a maximum flow for $c^1_{f_1} + c^2_{f_2}$.*

PROOF. Since f_1 and f_2 are restricted to the transportation networks of company 1 and company 2, respectively, $f_1 + f_2$ must be accommodated by the overall capacity $c = c^1 + c^2$, even if link sharing is not enabled. After subtracting $f_1 + f_2$, the residual graph thus resulted is:

$$\begin{aligned} c_{f_1+f_2} &= (c^1 + c^2) - (f^1 + f^2) \\ &= (c^1 - f^1) + (c^2 - f^2) = c^1_{f_1} + c^2_{f_2}. \end{aligned}$$

A direct application of Lemma 1 finishes our proof. ■

As it is generally hard to localize maximum flow computations with regard to network topology, the above property is important because it takes another route, which reuses partial results f_1, f_2 and attenuates the overall workload from $c^1 + c^2$ to $c^1_{f_1} + c^2_{f_2}$. By doing this, we are much closer to the overall maximum flow $f_1 + f_2 + f'$ because a big portion of it, $f_1 + f_2$, has already been decided even before we start an augmenting algorithm. Unfortunately, there are also cases where such attenuation properties are hard, if not impossible, to develop.

Example 5 (Centrality). *Centrality* is an important concept in social network analysis, which reflects how "central" a particular node's position is in a given network. One definition called *betweenness centrality* C_B uses shortest path to model this: Let n_{jk} denote the number of shortest paths (*i.e.*, equally short ones) between two nodes j and k; for any node i, $\frac{n_{jk}(i)}{n_{jk}}$ is the fraction of shortest paths between j, k that go through i, with $C_B(i)$ summing it up for all node pairs: $C_B(i) = \sum_{j,k \neq i} \frac{n_{jk}(i)}{n_{jk}}$. Intuitively, for a "star"-shaped network, all shortest paths must pass the network center, which makes C_B achieve its maximum value $(|V| - 1)(|V| - 2)/2$.

Only considering shortest paths is inevitably restrictive in many situations; and thus, *information centrality* C_I goes one step further by taking all paths into account. It models any path from j to k as a signal transmission, which has a channel noise proportional to its path length. For more details, we refer the readers to [17], which has derived the following formula based on information theoretic analysis: Let A be a matrix, whose a_{ij} entry designates the interaction strength between node i and node j; define $B = D - A + J$, where D is a diagonal matrix with $D_{ii} = \sum_{j=1}^{|V|} a_{ij}$ and J is a matrix having all unit elements; perform an inverse operation to get the *centrality matrix* $C = B^{-1}$, write its diagonal sum as $T = \sum_{j=1}^{|V|} c_{jj}$ and its row sum as $R_i = \sum_{j=1}^{|V|} c_{ij}$, the information centrality of node i is then equivalent to

$$C_I(i) = \frac{1}{c_{ii} + (T - 2R_i)/|V|}.$$

Now, with regard to the coauthor network described in Example 1, if we define the interaction strength between two authors as their total collaboration frequency for a set of network snapshots, then an aggregated graph M_{cen} can be defined, whose node i is associated with a node attribute $C_I(i)$ equaling its information centrality. ■

Claim 2 *The computation of M_{cen} is hard to be attenuated in a level-by-level aggregation scheme.*

PROOF. As we can see, the core component of information centrality computation is a matrix inverse. Now, given two portions of network snapshots that are overlaid, the overall centrality matrix is:

$$\big[(D_1 + D_2) - (A_1 + A_2) + J\big]^{-1} = (B_1 + B_2 - J)^{-1}.$$

From intermediate results, we know the centrality matrices $C_1 = B_1^{-1}$ and $C_2 = B_2^{-1}$; however, it seems that they do not help much to decrease the computation cost of inverting $B_1 + B_2 - J$. ■

When things like this happen, an alternative is to abandon the exactness requirement and bound the answer within some range instead; as illustrated by the following section, this would become useful if the cube construction is subject to a set of constraints.

5 Constraints and Partial Materialization

In above, we have focused on the computation of a full cube, *i.e.*, each cell in each cuboid is calculated and stored. In many cases, this is too costly in terms of both space and time, which might even be unnecessary if the users are not interested in obtaining all the information. Usually, users may stick with an *interestingness* function I, indicating that only those cells above a particular threshold δ make sense to them. Considering this, all cells c with $I(c) \geq \delta$ comprise an *iceberg* cube, which represents a *partial materialization* of the cube's interesting part.

Optimizations exist as to how such an iceberg cube can be calculated, *i.e.*, how to efficiently process the constraint of $I(c) \geq \delta$ during materialization, without generating a full cube at first. Two most important categories of constraints are *anti-monotone* and *monotone*. They can be "pushed" into the computation process as follows: In bottom-up computation, *i.e.*, high level cells are calculated first, followed by low level cells they cover, on one hand, if a high level cell c_h does not satisfy an anti-monotone constraint, then we know that no low level cell c_l covered by c_h would satisfy it, and thus the computation can be immediately terminated with c_l pruned from the cube; on the other hand, if a high level cell c_h already satisfies a monotone constraint, then we no longer need to perform checkings for any low level cells covered by c_h because they would always satisfy it. Concerning top-down computation, the roles of anti-monotonicity and monotonicity are reversed accordingly.

It is easy to see that the anti-monotone and monotone properties depend on specific analysis of measures and interestingness functions. Here, since we are working with networked data, some graph theoretic studies need to be made. For example, regarding the maximum flow's value $|f| = \sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t)$, *i.e.*, the highest amount of transportation a network can carry from S to T, $|f| \geq \delta_{|f|}$ is anti-monotone, because the transporting capability of one company must be smaller than that of all companies together (actually, as we showed in Section 4, the flow value of c_h is no smaller than the flow sum of all c_l's that are covered by c_h, which is a condition stronger than the normal anti-monotonicity defined between two individual cells); also, regarding the diameter d of a graph, which is designated as the maximum shortest path length for all node pairs, $d \geq \delta_d$ is monotone, because more viable links can only cut down the distance between two vertices. Certainly, having such properties derived will greatly help us in constructing an iceberg cube.

6 Experiments

In this section, we present empirical studies evaluating the effectiveness and efficiency of the proposed graph OLAP framework. It includes two kinds of datasets, one real dataset and one synthetic dataset. All experiments are done on a Microsoft Windows XP machine with a 3GHz Pentium IV CPU and 1GB main memory. Programs are compiled by Visual C++.

6.1 Real Dataset

The first dataset we use is the DBLP Bibliography (http://www.informatik.uni-trier.de/~ley/db/) downloaded in April 2008. Upon parsing the author field of papers, a coauthor network with multiple snapshots can be constructed, where an edge of weight w is added between two persons if they publish w papers together. We pick a few representative conferences for the following three research areas:

- Database (DB): PODS/SIGMOD/VLDB/ICDE/EDBT,
- Data Mining (DM): ICDM/SDM/KDD/PKDD,
- Information Retrieval (IR): SIGIR/WWW/CIKM;

and also distribute the publications into five-year bins: $(2002, 2007], (1997, 2002], (1992, 1997], \ldots$. In this way, we obtain two informational dimensions: *venue* and *time*, on which I-OLAP operations can be performed.

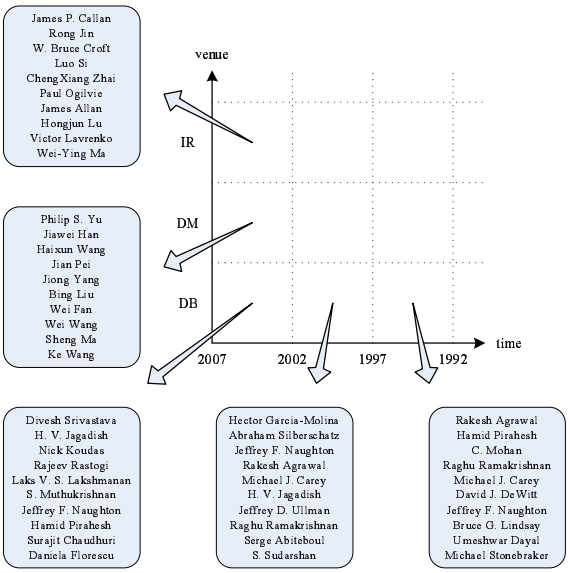

Figure 3: A Multi-Dimensional View of Top-10 "Central" Authors

Figure 3 shows a classic OLAP scenario. Based on the definition in Section 4, we compute the information centrality of each node in the coauthor network and rank them from high to low. in general, people who not only publish a lot but also publish frequently with a big group of collaborators will be ranked high. Along the *venue* dimension, we can see how the "central" authors change across different research areas, while along the *time* dimension, we can see how the "central" authors evolve over time. In fact, what Figure 3 gives is a multi-dimensional view of the graph cube's base cuboid; without any difficulty, we can also aggregate DB,

DM and IR into a broad Database field, or generalize the *time* dimension to *all-years*, and then compute respective cells. The results are omitted here due to the lack of space.

6.2 Synthetic Dataset

The second experiment we perform is used to demonstrate the effectiveness of the optimizations that are proposed to efficiently calculate a graph data cube. The test we pick is the computation of maximum flow as a graph OLAP measure, which has been used as an exemplifying application in above.

Generator Mechanism. Since it is generally hard to get real flow data, we develop a synthetic generator by ourselves. The data is generated as follows: The graph has a source node s and a destination node t, and in between of them, there are L intermediate layers, with each layer containing H nodes. There is a link with infinite capacity from s to every node in layer 1, and likewise from every node in layer L to t. Other links are added from layer i to layer $i+1$ on a random basis: For the total number of $H \cdot H$ choices between two layers, we pick αH^2 pair of nodes and add a link with capacity 1 between them.

For the graph cube we construct, there are d dimensions, each dimension has $card$ different values (*i.e.*, cardinality), which can be generalized to "all/*". For a base cell where all of its dimensions are set on the finest ungeneralized level, we generate a snapshot of capacity network $L5H1\text{K}\alpha 0.01$, *i.e.*, there are 5 layers of intermediate nodes, and $0.01 \cdot (1K)^2 = 10K$ links are randomly added between neighboring layers.

The algorithm we use to compute the maximum flow works in an incremental manner. It randomly picks an augmenting path from s to t until no such paths exist. To accommodate top-down computation, where high level cells are computed after low level cells so that intermediate results can be utilized, we integrate our attenuation scheme with the classic Multi-Way aggregation method for cube computation [20].

The results are depicted in Figure 4 and Figure 5, with Figure 4 fixing the cardinality as 2 and varying d from 2, 3, ... up to 6, and Figure 5 fixing the number of dimensions as 2 and varying $card$ from 2, 4 ..., up to 8. It can be seen that, the optimization achieved through attenuation is obvious; especially, when the dimensionality goes high, one may reap orders of magnitude savings.

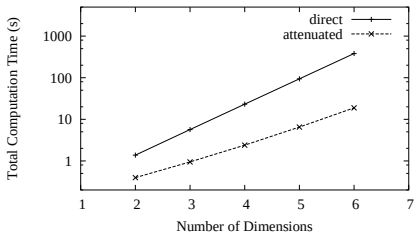

Figure 4: The Effect of Optimization w.r.t. Number of Dimensions

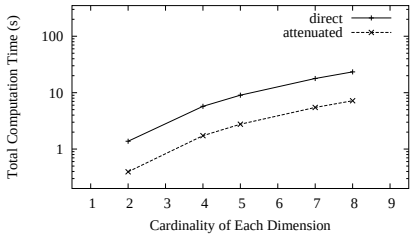

Figure 5: The Effect of Optimization w.r.t. Dimension Cardinality

7 Related Work

OLAP (On-Line Analytical Processing) is an important notion in data mining, which has drawn a lot of attention from the research communities. Representative studies include [9, 5], and a set of papers on materialized views and data warehouse implementations are collected in [10]. There have been a lot of works that deal with the efficient computation of a data cube, such as [20, 2], whereas the wealth of literature cannot be enumerated. However, all these researches target conventional spreadsheet data, *i.e.*, OLAP analysis is performed on independent data tuples that mathematically form a set. In contrast, as far as we know, ours is the first that puts graphs in a rigid multi-dimensional and multi-level framework, where due to the nature of the underlying data, an OLAP measure in general takes the form of an aggregated graph.

The classification of OLAP measures into distributive, algebraic and holistic was introduced in the traditional OLAP arena, where we can also find related works for iceberg cubing [7], partial materialization [12] and constraint "pushing" [15]. It is important to see how these basic aspects are dealt with in the graph OLAP scenario; and as we can see from the discussions, things become much more complicated due to the the increased structural complexity.

In graph OLAP, the aggregated graph can be thought as delivering a summarized view of the underlying networks based on some particular perspective and granularity. In this sense, concerning the generation of summaries for graphs, there have been quite a few researches that are associated with terminologies like compression, summarization, simplification, *etc.*. For example, [16, 3] study the problem of compressing large graphs, especially Web graphs; however, they only focus on how the Web link information can be efficiently stored and easily manipulated to facilitate computations such as PageRank and authority vectors, which do not give any insight into the graph structures. Similarly, [4] develops statistical summaries that analyze simple graph characteristics like degree distributions and hop-plots. They

are useful but hard to be navigated with regard to the underlying networks; also, the multi-dimensional functionality that can conduct analysis from different angles is missing. Another group of papers [19, 1, 13], which we refer as graph simplification, aim to condense a large network by preserving its skeleton in terms of topological features. In this case, attributes on nodes and edges are not important, and the network is indeed an unlabeled one in its abstract form. Works on graph clustering (to partition similar nodes together), dense subgraph detection (for community discovery, link spam identification, *etc.*) and graph visualization include [14], [8] and [11], respectively. They all provide some kind of summaries, but the objective and result achieved are substantially different from those of this paper.

With regard to summarizing attributed networks that incorporates OLAP-style functionalities, [18] is the closet to ours in spirit. It introduces an operation called SNAP (Summarization by grouping Nodes on Attributes and Pairwise relationships), which merges nodes with identical labels, combines corresponding edges, and aggregates a summary graph that displays relationships for such "generalized" node groups. Users can choose different resolutions by a k-SNAP operation just like rolling-up and drilling-down in an OLAP environment. Essentially, the above setting showcases an instance of topological graph OLAP that is defined here. Though we did not emphasize T-OLAP in this paper, based on our analysis of the aggregated graph describing collaboration frequency, it seems that SNAP can also be categorized as locally distributive, even if the context is switched from I-OLAP to T-OLAP.

8 Conclusions

We examine the possibility to apply *multi-dimensional* and *multi-level* analysis on networked data, and develop a *graph OLAP framework*, which is classified into two major subcases: *informational OLAP* and *topological OLAP*, based on the different OLAP semantics. Due to the nature of the underlying data, an OLAP measure now takes the form of an *aggregated graph*. With regard to efficient implementation, because of the space limit, we focus on I-OLAP in this paper. We categorize aggregated graphs based on the difficulty to compute them in an OLAP context, and suggest two properties: localization and attenuation, which may help speedup the processing. Both full materialization and constrained partial materialization are discussed. Experiments show insightful results on real datasets and demonstrate the efficiency of our proposed optimizations.

As for future works, there are a lot of directions we want to pursue on this topic, for example, extending the current framework to heterogenous-typed graphs, hypergraphs, *etc.*, and our immediate target would be providing a thorough study on topological OLAP.

References

[1] D. Archambault, T. Munzner, and D. Auber. Topolayout: Multilevel graph layout by topological features. *IEEE Trans. Vis. Comput. Graph.*, 13(2):305–317, 2007.

[2] K. S. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. In *SIGMOD Conference*, pages 359–370, 1999.

[3] P. Boldi and S. Vigna. The WebGraph framework I: Compression techniques. In *WWW*, pages 595–602, 2004.

[4] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1), 2006.

[5] S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. *SIGMOD Record*, 26(1):65–74, 1997.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, , and C. Stein, editors. *Introduction to Algorithms*. MIT Press, 2001.

[7] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing iceberg queries efficiently. In *VLDB*, pages 299–310, 1998.

[8] D. Gibson, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. In *VLDB*, pages 721–732, 2005.

[9] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals. *Data Min. Knowl. Discov.*, 1(1):29–53, 1997.

[10] A. Gupta and I. S. Mumick, editors. *Materialized Views: Techniques, Implementations, and Applications*. MIT Press, 1999.

[11] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Trans. Vis. Comput. Graph.*, 6(1):24–43, 2000.

[12] X. Li, J. Han, and H. Gonzalez. High-dimensional olap: A minimal cubing approach. In *VLDB*, pages 528–539, 2004.

[13] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *SIGMOD Conference*, pages 419–432, 2008.

[14] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[15] R. T. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained association rules. In *SIGMOD Conference*, pages 13–24, 1998.

[16] S. Raghavan and H. Garcia-Molina. Representing web graphs. In *ICDE*, pages 405–416, 2003.

[17] K. Stephenson and M. Zelen. Rethinking centrality: Methods and examples. *Social Networks*, 11(1):1–37, 1989.

[18] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *SIGMOD Conference*, pages 567–580, 2008.

[19] A. Y. Wu, M. Garland, and J. Han. Mining scale-free networks using geodesic clustering. In *KDD*, pages 719–724, 2004.

[20] Y. Zhao, P. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. In *SIGMOD Conference*, pages 159–170, 1997.