Title

Graph OLAP: a multi-dimensional framework for graph data analysis

Journal

Knowledge and Information Systems: An International Journal, 21(1)

ISSN

0219-3116

Authors

Chen, Chen

Yan, Xifeng

Zhu, Feida

et al.

Publication Date

2009-10-01

DOI

10.1007/s10115-009-0228-9

Peer reviewed

Knowl Inf Syst (2009) 21:41–63
DOI 10.1007/s10115-009-0228-9

REGULAR PAPER

Graph OLAP: a multi-dimensional framework for graph data analysis

Chen Chen · Xifeng Yan · Feida Zhu · Jiawei Han · Philip S. Yu

Received: 29 December 2008 / Revised: 5 April 2009 / Accepted: 9 May 2009 /
Published online: 1 August 2009

Abstract Databases and data warehouse systems have been evolving from handling normalized spreadsheets stored in relational databases, to managing and analyzing diverse application-oriented data with complex interconnecting structures. Responding to this emerging trend, graphs have been growing rapidly and showing their critical importance in many applications, such as the analysis of XML, social networks, Web, biological data, multimedia data and spatiotemporal data. Can we extend useful functions of databases and data warehouse systems to handle graph structured data? In particular, OLAP (On-Line Analytical Processing) has been a popular tool for fast and user-friendly *multi-dimensional* analysis of data warehouses. *Can we OLAP graphs?* Unfortunately, to our best knowledge, there are no OLAP tools available that can interactively view and analyze graph data from different perspectives and with multiple granularities. In this paper, we argue that it is critically important to OLAP graph structured data and propose a novel *Graph OLAP* framework. According to this framework, given a graph dataset with its nodes and edges associated with respective attributes, a multi-dimensional model can be built to enable efficient online analytical processing so that *any portions* of the graphs can be *generalized/specialized* dynamically, offering multiple, versatile views of the data. The contributions of this work are three-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*. Second, 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

C. Chen (✉) · F. Zhu · J. Han
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
e-mail: cchen37@uiuc.edu

X. Yan
Department of Computer Science, University of California at Santa Barbara, Santa Barbara, CA, USA

P. S. Yu
Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA

materialization where constraints are enforced to obtain an *iceberg cube*. As we can see, due to the increased structural complexity of data, aggregated graphs that depend on the underlying "network" properties of the graph dataset are much harder to compute than their traditional OLAP counterparts. Third, to provide more flexible, interesting and informative OLAP of graphs, we further propose a *discovery-driven* multi-dimensional analysis model to ensure that OLAP is performed in an intelligent manner, guided by expert rules and knowledge discovery processes. We outline such a framework and discuss some challenging research issues for discovery-driven Graph OLAP.

Keywords Multi-dimensional model · Graph OLAP · Efficient computation · Discovery-driven analysis

1 Introduction

OLAP (On-Line Analytical Processing) [2,7,11,12,31] 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), XML (data with loose schemas) and Web (human/computer networks), 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 multiple 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 a 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 the year it was introduced. 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 bus, by train, by air, by water., *etc.*, and each 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 of cargo from A to B using the planes it owns. Finally, we get a snapshot of capacity network for every transportation means 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 of cargo through 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 how the maximum flow can be configured, which has considered all transportation means and operating companies associated with c. Unlike Example 1, computing the aggregated graph of maximum flow 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 by definition must be smaller.

Example 3 (Collaboration patterns, revisited) Usually, the whole coauthor network could be too big for the users 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.*, merging 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 incurred 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 original network. Compared to this, the graph in Examples 1 and 2 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 the traditional OLAP: in addition to individual entities, the mutual interactions among them are also considered in the analytical process. Our major contributions are summarized in below.

1. **On 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 graph structured 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.
2. **On 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, centrality, *etc.*, 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 might help further optimize processings. Both full materialization and partial materialization (where constraints are enforced to obtain an *iceberg cube*) are discussed.
3. **On utilizing the framework for knowledge discovery**, we propose *discovery-driven Graph OLAP*, so that interesting patterns and knowledge can be effectively discovered. After presenting the general ideas, we outline a list of guiding principles and discuss some associated research problems. Being part of an ongoing project, it opens a promising path that could lead to more flexible and insightful OLAP of graph data, which we shall explore further in future works.

The remaining of this paper is organized as follows. In Sect. 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 computational optimizations. Constraints and partial materialization are studied in Sect. 5. Discovery-driven Graph OLAP is covered in Sect. 6. We report experiment results and related work in Sects. 7 and 8, respectively. Section 9 concludes this study.

2 A Graph OLAP framework

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

Definition 2.1 (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 concerning 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* $\rightarrow$ *area* $\rightarrow$ *all*,
- *time*: *year* $\rightarrow$ *decade* $\rightarrow$ *all*;

the role of these two dimensions is to organize snapshots into groups based on different perspectives, *e.g.*, *(db-conf, 2004)* and *(sigmod, all-years)*, where each of these groups corresponds to a "cell" in the OLAP terminology. They control what snapshots are to be looked at, without touching the inside of any single snapshot.

Definition 2.2 (Informational dimensions) With regard to the graph model presented in Definition 2.1, 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 formed will depict interactions among these groups as a whole, which summarizes the original network and hides specific details.

Definition 2.3 (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 existence, which plays a dual role: as a data source and as an 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 2.4 (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 Fig. 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 with regard to 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 2.5 (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 in G_i that have identical values on their Topo-Dims are grouped, whereas each group corresponds to a node of M^T, (2) the attributes attached to M^T are calculated as *aggregate functions* of the attributes attached to G_i.

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

- Roll-up: Shrink the network topology and obtain a T-aggregated graph that displays compressed views. Topological elements (*i.e.*, nodes and/or edges) are merged and replaced by corresponding higher-level ones during the process.
- 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.

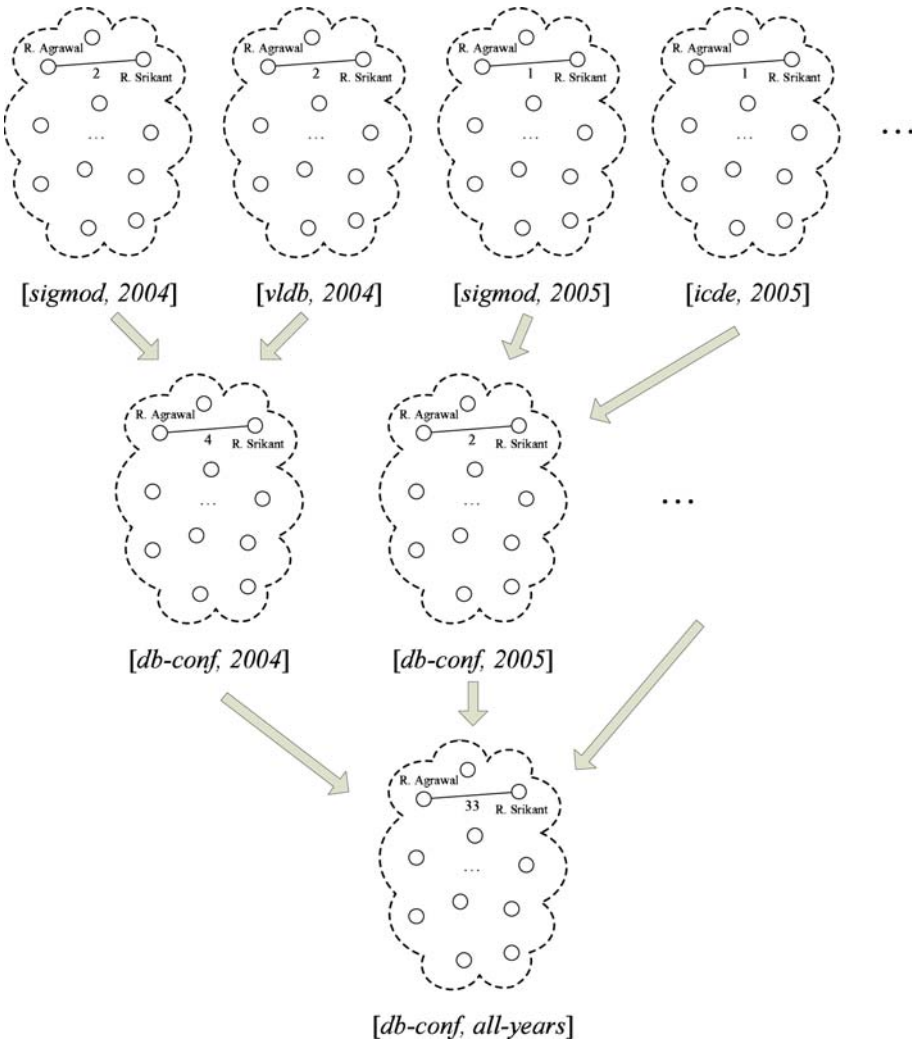

Fig. 1 The OLAP scenario for Example 1

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 has been used to illustrate algebraic measures, 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

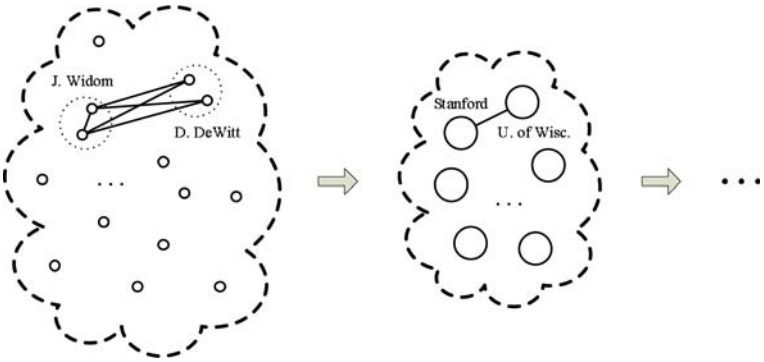

Fig. 2 The OLAP scenario for Example 3

without this property is put into the holistic category, which is intuitively much harder to calculate. Now, concerning Graph OLAP, based on similar criteria with regard to the aggregated graphs, we can also classify them into three categories.

Definition 3.1 (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 distinguish between I-OLAP and T-OLAP, there are actually six classes: I-distributive, I-algebraic, I-holistic and T-distributive, T-algebraic, T-holistic. Because of their wide differences with regard to semantics as well as implementation, in the remaining of this paper, we shall put more emphasis on the computational issues of I-OLAP; discussions for the other type will be covered in a separate future study that solely focuses on topological OLAP.

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 configuration in Example 2 is an I-algebraic measure based on the following reasoning. Suppose *transportation-type* and *company* comprise the two dimensions of a 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 each company's 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 decided 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 separated links from A to B, which are operated by different companies and have capacities c^1_{AB} and c^2_{AB}, respectively, are no different from a single link with capacity $c^1_{AB} + c^2_{AB}$, considering their contributions to the flow value. 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 3.1 is another question. Think about the maximum flow example we just mentioned, F_a takes the distributive capacity graph as input to compute the flow configuration, 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" situation: 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 the other authors to fulfill the computation. This is an ideal scenario, because the calculations can be 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^i_l, we only had a partial view about the c^i_l-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 difficulties in above, 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 4.1 (Localization) For an I-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 two researchers must surpass a δ_c threshold for the specified 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 unaggregated 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 frequency summed from the weights of these adjacent edges is 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 allow the path length to be at most 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 union of pairwise collaboration activities, whereas Example 1 can indeed be thought as a traditional OLAP scenario disguised under its graph appearances, because 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. Nevertheless, the case is special, anyway: Absolute localization would not hold for most information network, which is also the reason why traditional OLAP proves to be extremely restricted 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. [8]. 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 4.1 *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 $c_f = c - f$; now, 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 4.1 $f_1 + f_2 + f'$ 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, the overall capacity $c = c^1 + c^2$ must accommodate $f_1 + f_2$, even if link sharing is not enabled. As a result of subtracting $f_1 + f_2$, the residual graph becomes:

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

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.

However, we should admit that attenuation schemes usually take widely different forms, which might need to be developed with regard to specific aggregate measure graphs; furthermore, as we shall see next, there do exist cases where such properties are hard, if not impossible, to think of.

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 (as there could be 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 over all possible 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 Ref. [25], 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; now 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 $C_I(i)$ equivalent to its information centrality.

Claim 4.2 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:

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

From calculations performed on lower levels, 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 use intermediate results that are readily available to bound the answer within some range instead; as we shall elaborate in the following section, this will become very 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. Taking Example 2 for instance, it is possible that people may only care about those sub-networks that can transmit at least $\delta_{|f|}$ units of cargo, while the other cells are discarded from consideration, due to their limited usefulness for the overall transportation business.

Optimizations exist as to how such an iceberg cube can be calculated, *i.e.*, how to efficiently process constraints like $I(c) \geq \delta$ during materialization, without generating a full cube at first. In below, we will first classify different constraints into categories (Sect. 5.1), and then combine with some examples to see how each category should be dealt with (Sect. 5.2).

5.1 Constraint classification

Two most important categories of constraints are *anti-monotone* and *monotone*. They relate cells on different levels of the graph cube together, and are defined as follows.

Definition 5.1 A constraint C is *anti-monotone*, if for any high-level cell c_h and a low-level cell c_l covered by c_h, the following must hold: c_h violates $C \Rightarrow c_l$ violates C.

Definition 5.2 A constraint C is *monotone*, if for any high-level cell c_h and a low-level cell c_l covered by c_h, the following must hold: c_h satisfies $C \Rightarrow c_l$ satisfies C.

Note that, in Definition 5.1, "c_h violates $C \Rightarrow c_l$ violates C" is equal to "c_l satisfies $C \Rightarrow c_h$ satisfies C", and in Definition 5.2, "c_h satisfies $C \Rightarrow c_l$ satisfies C" is equal to "c_l violates $C \Rightarrow c_h$ violates C"; so, depending on whether c_h or c_l is computed (and thus verified against C) first, there are different ways to make use of these constraints, which we will demonstrate in below.

5.2 Constraint pushing

The anti-monotone and monotone constraints can be "pushed" deep into the computation process using the a priori principle [30]. In general, there are two approaches to compute a graph cube, *bottom-up* and *top-down*. In bottom-up computation, which can be contrasted with BUC [2] in traditional OLAP, high-level cells are calculated first, before drilling-down to low-level cells they cover. In top-down computation, which can be contrasted with Multi-Way [31] in traditional OLAP, we calculate low-level cells first, and then aggregate to high-level cells. Finally, which approach to adopt will depend on various parameters, including the size of the network, data sparsity, the measures to be computed, and the available constraints.

Now consider the bottom-up approach, 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 calculations can be immediately terminated, pruning c_l and its descendent from the cuboid lattice; 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. As for top-down computations, the roles of anti-monotonicity and monotonicity are reversed accordingly.

It is easy to see that 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. Let us examine a few examples.

Claim 5.1 Suppose maximum flow is the I-OLAP measure to be calculated, regarding its 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, constraint $|f| \geq \delta_{|f|}$ is anti-monotone.

Proof Intuitively, the transporting capability of one company must be smaller than that of all companies together, since there are now more available links for the flow to pass. In fact, as we showed in Sect. 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 a high-level cell and a single low-level cell it covers.

Claim 5.2 The diameter of a graph G is designated as the maximum shortest path length for all pairs of nodes in G. Now, denote diameter as d and let it be the I-OLAP measure we want to calculate, constraint $d \geq \delta_d$ is monotone.

Proof With more edges among individual vertices, the distance between any two nodes cannot go up, because the original shortest path still exists, which is not affected.

Because of space limit, we are not going to list more examples here. Certainly, having the conditions on interestingness classified into anti-monotone and monotone will greatly help us in constructing an iceberg cube. Here, combined with the top-down/bottom-up computation framework, a priori is a simple and effective principle to align search space pruning with particular computation orders. But unfortunately, we cannot use it to push every constraint into the mining process, as there do exist situations that are neither anti-monotone nor monotone. To this extent, we shall further investigate how the other types of more complex constraints can be successfully handled in the future.

6 Discovery-driven Graph OLAP

Apart from the type of data being dealt with, another important distinction of Graph OLAP from the traditional one is the need for flexibility in manipulating information networks. So, in addition to the uniform drilling of traditional OLAP, where all nodes at a given level of abstraction are simultaneously rolled-up/drilled-down, *e.g.*, from month to quarter, and then to year, Graph OLAP may require *selective drilling* for a given node or neighborhood. For instance, in the coauthor network example, a user could be interested in the collaborative relationship between Yahoo! Labs and related individual researchers. Such an analysis may show strong collaborations between AnHai Doan at Wisconsin and people at Yahoo! Labs. On the other hand, if all Wisconsin researchers are merged into a single institution in the same way as Yahoo! Labs, it would be hard to discover such a relationship since, collectively, there would be even stronger collaborations between Wisconsin and Berkeley (instead of Yahoo! Labs), which may overshadow AnHai's link to Yahoo! Labs.

Selective drilling, though promising, may generate an exponential number of combinations, which is too costly to compute and explore. To ensure that one can pinpoint to the "real gold" during exploration, *discovery-driven Graph OLAP* should be adopted, *i.e.*, rather than searching the complete cube space, one has to be sure that the planned drilling should help the discovery of interesting knowledge.

6.1 Discovery-driven: guiding principles

In the following, we first outline a few guiding principles for discovery-driven Graph OLAP.

1. Discovery driven by rule-based heuristics. When performing Graph OLAP, we usually have some general intuitions about where and when the selective drilling should be focused and at what levels/nodes one is most likely to discover something interesting.

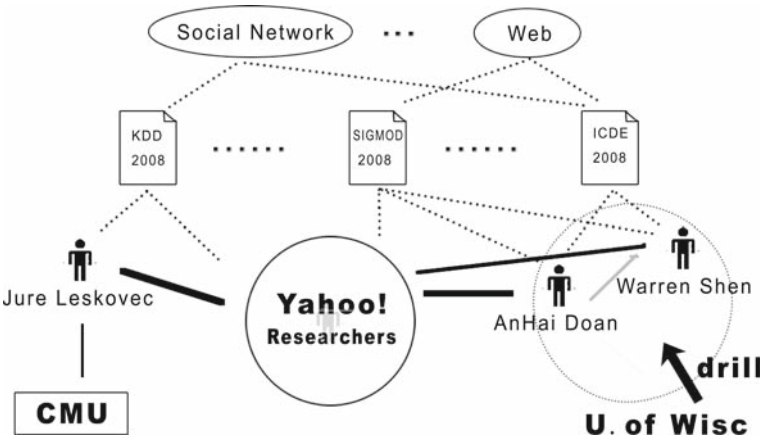

Fig. 3 Discovery-driven Graph OLAP

For instance, when studying research collaborations, it may not be wise to quickly merge prominent researchers into groups, before exploring some interesting patterns for them at first. As an example, if Raghu Ramakrishnan were merged into an even "bigger" entity like Yahoo! Labs, it may blur the individual relationships that can be discovered; rather, we could keep his own identity in the network, on which clearer patterns with finer granularity can be observed. Alternatively, for ordinary graduate students, it seems to be more interesting to group them together or have them absorbed by nearby "hub" nodes, because for such individuals, it is not likely that something significant can stand out from the rest. In this sense, a rule like "*delay the merge of 'big' or 'distinct' nodes*" could be quite simple to work out and follow, as the system only needs to consult attributes of an entity itself (*e.g.*, how many papers a researcher has published) or explore some very local information (*e.g.*, how many persons he/she has collaborated with) for decision making.

2. Discovery driven by statistical analysis. In many cases, it is beneficial to conduct some global (rather than local) preliminary analysis before selecting any drilling operation. Consider a top-down exploratory scenario where the granularity is currently set at the institution level, *e.g.*, both Yahoo! Labs and University of Wisconsin are treated as a whole; now, if an automatic background computation shows that the collaboration activities between Yahoo! Labs and Anhai Doan are significantly higher than normal, then it is rewarding to drill-down, because the status of Anhai is like an outlier in Wisconsin; otherwise, such drilling may not disclose anything truly interesting. This is similar to discovery-driven OLAP proposed by Sarawagi et al. [23], but in the environment of graphs. As a simple implementation of this idea, we may take the collaboration frequencies between Yahoo! Labs and every person from Wisconsin, calculate the variance: if the variance is high, one may set up an indicator which may suggest people to click on "U. of Wisc." and expand it to get a refined view, which is just the situation described in Fig. 3. Compared to the first guiding principle, there are no longer simple rules that stipulate whether a drill-down or roll-up should be performed, everything depends on a global statistical analysis about the entities in the network, which aims to find out those interesting (or outlying) phenomena that are worthwhile to explore.

3. Discovery driven by pattern mining. Another way for discovery-driven OLAP is fueled by the potential to discover interesting patterns and knowledge on graphs using data mining techniques: if splitting or merging of certain sets of nodes may lead to the discovery of interesting clusters, frequent patterns, classification schemes, and evolution regularities/outliers, then the drilling should be performed as selected, with the results/patterns demonstrated to users. Otherwise, the drilling will not be performed. Notice that, although such pattern discovery can be executed on the fly at the time of user interaction, the discovery process could be too time-consuming with regard to the user's mouse clicking. Therefore, we suggest the pre-computation of some promising drilling paths as intermediate results to speed-up interactive knowledge discovery. It is an interesting research issue to determine the things to be computed in order to facilitate such on-line analysis.

6.2 Network discovery for effective OLAP

As we discussed in above, for effective discovery-driven OLAP, it is important to perform some essential data mining on the underlying graphs to reveal interesting network patterns that may help us discover the hidden knowledge. Here, we discuss some interesting network discovery procedures by taking the collaboration scenario of researchers as an example.

For studying coauthor networks, it is important to distinguish different roles the authors may play in the network. For example, based on the coauthor relationships over time, it is possible to dig out advisor–advisee relationships. Usually, advisee is a mediocre node in the network without many publications, who then starts to coauthor substantially with his prominent advisor; after graduation, he/she joins industry or moves on to another institution, and the publishing behaviors are changed again. Advisors can also be identified, based on his/her long-term publication history as well as a center role of working with many junior coauthors. It is interesting to organize researchers under such a phylogeny tree and examine the interactions between different clusters of academic "families".

Besides some not-so-sophisticated knowledge discovery procedures, a mining process may involve induction on the entire information networks, as well. For example, in order to partition an interconnected, heterogeneous information network into a set of clusters and rank the nodes in each cluster, one could develop a RankClus framework [26], which integrates clustering and ranking together to effectively cluster information networks into multiple groups and rank nodes in each group based on certain nice properties (such as authority). By examining authors, research papers, and conferences, one can group conferences in the same fields together to form conference clusters, group authors based on their publication records into author clusters and in the meantime rank authors and conferences based on their corresponding authorities. Such clustering results enhanced by ranking information would be an ideal feed into Graph OLAP. Interestingly, such clustering-ranking can be performed based on the links only, without checking the citation information nor the keywords or text information contained in the conferences and/or publication titles. The details of such techniques is beyond the discussions of this paper, but it sheds light on automated processes to effectively identify concept hierarchies and important nodes for discovery-driven Graph OLAP.

7 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++.

7.1 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],…. In this way, we obtain two informational dimensions: *venue* and *time*, on which I-OLAP operations can be performed.

Figure 4 shows a classic OLAP scenario. Based on the definition in Sect. 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 Fig. 4 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.

7.2 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 un-generalized level, we generate a snapshot of capacity network $L5H1K\alpha 0.01$, *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 [31].

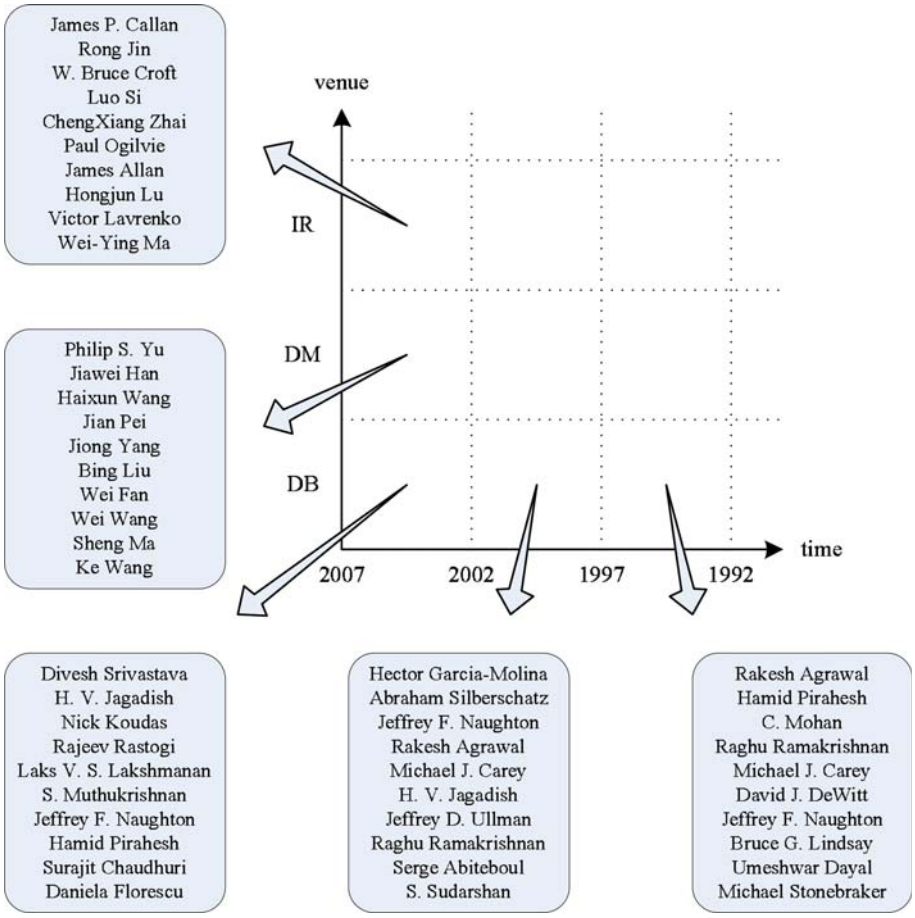

Fig. 4 A multi-dimensional view of top-10 "Central" Authors

The results are depicted in Figs. 5 and 6, with Fig. 5 fixing the cardinality as 2 and varying d from 2, 3, . . . up to 6, and Fig. 6 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, because in effect we do not need to compute a full-scale aggregated graph from scratch, and part of the burden has been transferred to previous rounds of calculations. Especially, when the dimensionality goes high in Fig. 5, so that more levels of cube cells are present, the superiority of attenuation-based methods becomes more significant, and one may reap orders of magnitude savings.

8 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 Ref. [7,11], and a set of papers on materialized views and data warehouse implementations are collected in [12]. There have been a lot of works that deal with the efficient computation of a

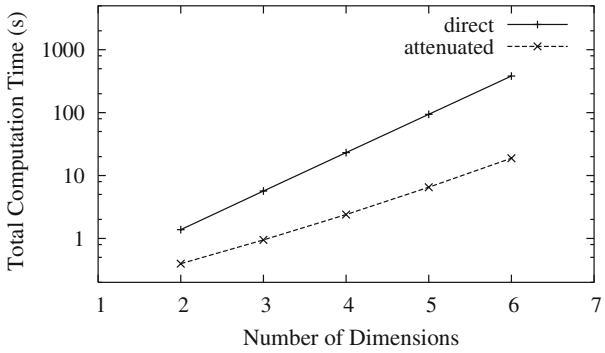

Fig. 5 The effect of optimization w.r.t. number of dimensions

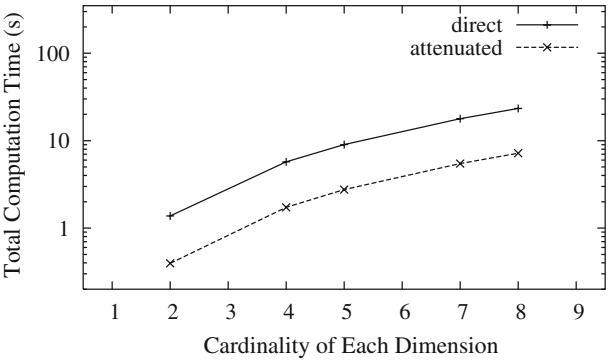

Fig. 6 The effect of optimization w.r.t. dimension cardinality

data cube, such as [2,31], 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 [9], partial materialization [17] and constraint pushing [21]. It is important to see how these basic aspects are dealt with in the Graph OLAP scenario; and as we have seen from the discussions, things become much more complicated due to 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, which helps users get informative insights into the data [6]. 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, Boldi and Vigna and Raghavan and Garcia-Molina [3,22] 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 provide any pointers into the graph structures.

Similarly, Chakrabarti and Faloutsos [4] develops statistical summaries that analyze simple graph characteristics like degree distributions and hop-plots; Leskovec et al. [16] goes one step further by looking into the evolutionary behavior of these statistics, and proposes a generative model that helps explain the latent mechanism. These compressed views 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 [1,15,19,29] are often referred as graph simplification, *e.g.*, Archambault et al. [1] aims to condense a large network by preserving its skeleton in terms of topological features, and Kossinets et al. [15] tries to extract the "backbone" of a social network, *i.e.*, the subgraph that consists of edges on which information has the potential to flow the quickest. 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.*), graph visualization and evolutionary pattern extraction/contrast include Ref. [20], Ref. [10], Ref. [13,28], and Ref. [5], 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, Tian et al. [27] 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 (actually, it might not be necessary to require exactly the same label for real applications, *e.g.*, Lu et al. [18] introduces a way to find similar group of entities in a network, and this can be taken as the basis to guide node merges), 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.

Sarawagi et al. [23] introduces the discovery-driven concept for traditional OLAP, which aims at pointing out a set of direct handles or indirect paths that might lead to the interesting/outlying cells of a data cube. Their method is statistics oriented. As we proposed in this paper, Graph OLAP can also adopt the discovery-driven concept, but in the context of graphs and information networks, which suggests new classes of discovery-driven methods using topological measures and graph patterns. Different from traditional OLAP, where dimensions are often globally drilled-down and rolled-up, Graph OLAP takes selective drilling into consideration, which leverages graph mining (*e.g.*, [14]), link analysis (*e.g.*, [24]), *etc.*, and might only involve some local portion of a big network.

9 Conclusions

We examine the possibility to apply *multi-dimensional* 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, we focus more 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. Towards more intelligent Graph OLAP, we further propose a *discovery-driven* multi-dimensional analysis model and discuss many challenging research issues associated with it. 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, hyper-graphs, *etc.*, and our immediate target would be providing a thorough study on topological Graph OLAP.

Acknowledgments We thank Dr. Raghu Ramakrishnan for useful comments and discussions on this article and related researches. 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.

References

1. Archambault D, Munzner T, Auber D (2007) Topolayout: Multilevel graph layout by topological features. IEEE Trans Vis Comput Graph 13(2):305–317
2. Beyer KS, Ramakrishnan R (1999) Bottom-up computation of sparse and iceberg cubes. In: SIGMOD Conference, pp 359–370
3. Boldi P, Vigna S (2004) The WebGraph framework I: Compression techniques. In: WWW, pp 595–602
4. Chakrabarti D, Faloutsos C (2006) Graph mining: Laws, generators, and algorithms. ACM Comput Surv 38(1)
5. Chan J, Bailey J, Leckie C (2008) Discovering correlated spatio-temporal changes in evolving graphs. Knowl Inf Syst 16(1):53–96
6. Chandola V, Kumar V (2007) Summarization—compressing data into an informative representation. Knowl Inf Syst 12(3):355–378
7. Chaudhuri S, Dayal U (1997) An overview of data warehousing and olap technology. SIGMOD Record 26(1):65–74
8. Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to algorithms. MIT Press, Cambridge
9. Fang M, Shivakumar N, Garcia-Molina H, Motwani R, Ullman JD (1998) Computing iceberg queries efficiently. In: VLDB, pp 299–310
10. Gibson D, Kumar R, Tomkins A (2005) Discovering large dense subgraphs in massive graphs. In: VLDB, pp 721–732
11. Gray J, Chaudhuri S, Bosworth A, Layman A, Reichart D, Venkatrao M, Pellow F, Pirahesh H (1997) Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub totals. Data Min Knowl Discov 1(1):29–53
12. Gupta A, Mumick IS (1999) Materialized Views: Techniques, Implementations, and Applications. MIT Press
13. Herman I, Melançon G, Marshall MS (2000) Graph visualization and navigation in information visualization: a survey. IEEE Trans Vis Comput Graph 6(1):24–43
14. Jeh G, Widom J (2004) Mining the space of graph properties. In: KDD, pp 187–196
15. Kossinets G, Kleinberg JM, Watts DJ (2008) The structure of information pathways in a social communication network. In: KDD, pp 435–443
16. Leskovec J, Kleinberg JM, Faloutsos C (2005) Graphs over time: densification laws, shrinking diameters and possible explanations. In: KDD, pp 177–187
17. Li X, Han J, Gonzalez H (2004) High-dimensional olap: a minimal cubing approach. In: VLDB, pp 528–539
18. Lu W, Janssen JCM, Milios EE, Japkowicz N, Zhang Y (2007) Node similarity in the citation graph. Knowl Inf Syst 11(1):105–129
19. Navlakha S, Rastogi R, Shrivastava N (2008) Graph summarization with bounded error. In: SIGMOD conference, pp 419–432
20. Ng AY, Jordan MI, Weiss Y (2001) On spectral clustering: analysis and an algorithm. In: NIPS, pp 849–856

21. Ng RT, Lakshmanan LVS, Han J, Pang A (1998) Exploratory mining and pruning optimizations of constrained association rules. In: SIGMOD Conference, pp 13–24
22. Raghavan S, Garcia-Molina H (2003) Representing web graphs. In: ICDE, pp 405–416
23. Sarawagi S, Agrawal R, Megiddo N (1998) Discovery-driven exploration of olap data cubes. In: EDBT, pp 168–182
24. Sen P, Namata GM, Bilgic M, Getoor L, Gallagher B, Eliassi-Rad T (2008) Collective classification in network data. CS-TR-4905, University of Maryland, College Park
25. Stephenson K, Zelen M (1989) Rethinking centrality: Methods and examples. Soc Netw 11(1):1–37
26. Sun Y, Han J, Zhao P, Yin Z, Cheng H, Wu T (2009) Rankclus: Integrating clustering with ranking for heterogeneous information network analysis. In: EDBT, pp 565–576
27. Tian Y, Hankins RA, Patel JM (2008) Efficient aggregation for graph summarization. In: SIGMOD conference, pp 567–580
28. Wang N, Parthasarathy S, Tan K-L, Tung AKH (2008) Csv: visualizing and mining cohesive subgraphs. In: SIGMOD conference, pp 445–458
29. Wu AY, Garland M, Han J (2004) Mining scale-free networks using geodesic clustering. In: KDD, pp 719–724
30. Wu X, Kumar V, Quinlan JR, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng AFM, Liu B, Yu PS, Zhou Z-H, Steinbach M, Hand DJ, Steinberg D (2008) Top 10 algorithms in data mining. Knowl Inf Syst 14(1):1–37
31. Zhao Y, Deshpande P, Naughton JF (1997) An array-based algorithm for simultaneous multidimensional aggregates. In: SIGMOD conference, pp 159–170

Author Biographies

Chen Chen is a Ph.D. student in the Department of Computer Science, University of Illinois at Urbana-Champaign. He received B.E. in Computer Science and Technology from University of Science and Technology of China in 2003, and M.S. in Computer Science from University of Illinois at Urbana-Champaign in 2006, respectively. Chen has been working in the area of data mining in general, and his current research is focused on modeling, managing and analyzing large-scale graph and information network data, with applications from chem/bio-informatics, social networks, the Web and computer systems.

Xifeng Yan is an assistant professor at the University of California at Santa Barbara. He received a Ph.D. degree in Computer Science from the University of Illinois at Urbana-Champaign in 2006. He was a research staff member at the IBM T. J. Watson Research Center between 2006 and 2008. Dr. Yan's research interests include data mining, databases, and bioinformatics. He investigates models and algorithms for managing and mining complex graphs and networks in the biological, information, and social worlds. He has filed 6 patents and published more than 40 papers in refereed journals and conferences, including TODS, Bioinformatics, TSE, SIGMOD, SIGKDD, VLDB, FSE, and ISMB. He gave tutorials in major data mining and database conferences, including ICDM'05, ICDE'06, SIGKDD'06, and SIGKDD'08. Dr. Yan received the 2007 ACM SIGMOD Doctoral Dissertation Runner-Up Award for his work in graph mining and graph data management. He also received the Best Student Paper Award in ICDE'07 and PAKDD'07.

Feida Zhu got his B.S. in Computer Science from Fudan University. He is currently working towards his Ph.D. in Computer Science at University of Illinois at Urbana-Champaign. His research interests are data mining and algorithm analysis in general. In particular, he has been working on graph mining, graph search and information network analysis.

Jiawei Han is a Professor in the Department of Computer Science at the University of Illinois. He has been working on research into data mining, data warehousing, stream data mining, spatiotemporal and multimedia data mining, biological data mining, social network analysis, text and Web mining, and software bug mining, with over 350 conference and journal publications. He has chaired or served in over 100 program committees of international conferences and workshops and also served or is serving on the editorial boards for Data Mining and Knowledge Discovery, IEEE Transactions on Knowledge and Data Engineering, Journal of Computer Science and Technology, and Journal of Intelligent Information Systems. He is currently the founding Editor-in-Chief of ACM Transactions on Knowledge Discovery from Data (TKDD). Jiawei has received three IBM Faculty Awards, the Outstanding Contribution Award at the 2002 International Conference on Data Mining, ACM Service Award (1999) and ACM SIGKDD Innovation Award (2004), and IEEE Computer Society Technical Achievement Award (2005). He is an ACM and IEEE Fellow. His book "Data Mining: Concepts and Techniques" (Morgan Kaufmann) has been used popularly as a textbook.

Philip S. Yu received the M.S. and Ph.D. degrees in E.E. from Stanford University, and the M.B.A. degree from New York University. He is a Professor in the Department of Computer Science at the University of Illinois at Chicago and also holds the Wexler Chair in Information and Technology. Previously he was manager of the Software Tools and Techniques group at the IBM Thomas J. Watson Research Center. His research interests include data mining, Internet applications and technologies, database systems, multimedia systems, parallel and distributed processing, and performance modeling. Dr. Yu has published more than 530 papers in refereed journals and conferences. He holds or has applied for more than 350 US patents. Dr. Yu is a Fellow of the ACM and the IEEE. He is associate editors of ACM Transactions on the Internet Technology and ACM Transactions on Knowledge Discovery from Data.