Graph Cube: On Warehousing and OLAP Multidimensional Networks

Peixiang Zhao[♮] Xiaolei Li[♭] Dong Xin[†] Jiawei Han[§]
[♮,§]University of Illinois, Urbana, IL, United States
[♭]Microsoft Corporation, Redmond, WA, United States
[†]Google Inc., Mountain View, CA, United States
[♮]pzhao4@uiuc.edu [♭]xiaoleil@microsoft.com [†]dongxin@google.com [§]hanj@uiuc.edu

ABSTRACT

We consider extending decision support facilities toward large sophisticated networks, upon which multidimensional attributes are associated with network entities, thereby forming the so-called *multidimensional networks*. Data warehouses and OLAP (Online Analytical Processing) technology have proven to be effective tools for decision support on relational data. However, they are not well-equipped to handle the new yet important multidimensional networks. In this paper, we introduce Graph Cube, a new data warehousing model that supports OLAP queries effectively on large multidimensional networks. By taking account of both attribute aggregation and structure summarization of the networks, Graph Cube goes beyond the traditional data cube model involved solely with numeric value based group-by's, thus resulting in a more insightful and structure-enriched aggregate network within every possible multidimensional space. Besides traditional cuboid queries, a new class of OLAP queries, *crossboid*, is introduced that is uniquely useful in multidimensional networks and has not been studied before. We implement Graph Cube by combining special characteristics of multidimensional networks with the existing well-studied data cube techniques. We perform extensive experimental studies on a series of real world data sets and Graph Cube is shown to be a powerful and efficient tool for decision support on large multidimensional networks.

Categories and Subject Descriptors

E.1 [**Data Structures**]: Graphs and networks; H.2.7 [**Database Administration**]: Data warehouse and repository

General Terms

Algorithms, Management, Performance

Keywords

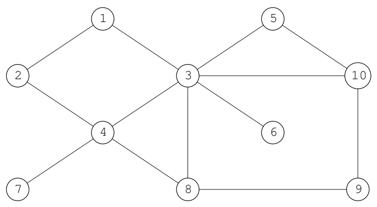
Data warehouse, OLAP, Data cube, Graph cube, Multidimensional network

1. INTRODUCTION

Recent years have seen an astounding growth of networks in a wide spectrum of application domains, ranging from sensor and communication networks to biological and social networks. And it becomes especially apparent as far as the great surge of popularity for Web 2.0 applications is concerned, such as Facebook, LinkedIn, Twitter and Foursquare. Typically, these networks can be modeled as large graphs with vertices representing entities and edges depicting relationship between entities [1]. Apart from the topological structures encoded in the underlying graph, multidimensional attributes are often specified and associated with vertices, forming the so-called *multidimensional networks*. While studies on contemporary networks have been around for decades [20], and a plethora of algorithms and systems have been devised for multidimensional analysis in relational databases [7, 4], none has taken both aspects into account in the multidimensional network scenario. As a result, there exist considerable technology gaps in managing, querying and summarizing such data effectively. And a growing need arises in order to shorten these technology gaps and develop specialized approaches for multidimensional networks.

EXAMPLE 1. *Figure 1 presents a sample social network consisting of several individuals interconnected with friend relationship. There are ten vertices (identified with user ID) and thirteen edges in the underlying graph, as shown in Figure 1(a). Each individual of the network contains a set of multidimensional attributes describing her/his properties, including user ID (as primary key), gender, location (in state), profession and yearly income, which is represented as a tuple in a vertex attribute table, as shown in Figure 1(b). The graph structure, together with the vertex-centric multidimensional attributes, forms a multidimensional network.*

One possible opportunity of special interest is to support data warehousing and online analytical processing (OLAP) on multidimensional networks. Data warehouses are critical in generating summarized views of a business for proper decision support and future planning [4]. This includes aggregations and group-by's of enterprise RDB data based on the multidimensional data cube model [7]. For example, in a sales data warehouse, time of sale, sales district, salesperson, and product might be the *dimensions* of interest, and numeric quantities such as sales, budget and revenue might be the *measures* to be examined. OLAP operations, such as roll-up, drill-down, slice-and-dice and pivot, are supported to explore different multidimensional views and al-

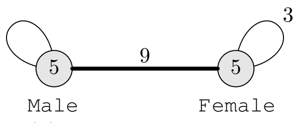


(a) Graph

ID	Gender	Location	Profession	Income
1	Male	CA	Teacher	$70,000
2	Female	WA	Teacher	$65,000
3	Female	CA	Engineer	$80,000
4	Female	NY	Teacher	$90,000
5	Male	IL	Lawyer	$80,000
6	Female	WA	Teacher	$90,000
7	Male	NY	Lawyer	$100,000
8	Male	IL	Engineer	$75,000
9	Female	CA	Lawyer	$120,000
10	Male	IL	Engineer	$95,000

(b) Vertex Attribute Table

Figure 1: A Sample Multidimensional Network with a Graph and a Multidimensional Vertex Attribute Table

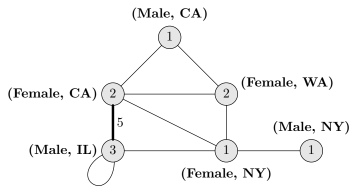


(a) Aggregate Network

Gender	COUNT(*)
Male	5
Female	5

(b) Aggregate Table

Figure 2: Multidimensional Network Aggregation vs. RDB Aggregation (Group by Gender)



(a) Aggregate Network

Gender	Location	COUNT(*)
Male	CA	1
Female	CA	2
Female	WA	2
Male	IL	3
Male	NY	1
Female	NY	1

(b) Aggregate Table

Figure 3: Multidimensional Network Aggregation vs. RDB Aggregation (Group by Gender and Location)

low interactive querying and analysis of the underlying data. As important means of decision support and business intelligence, data warehouses and OLAP are advantageous for multidimensional networks as well. For example, a company is investigating how to run a marketing campaign in order to maximize returns. They turn to a large national social network to study the business and preference patterns of interlinked people within different multidimensional spaces, such as genders, locations, professions, hobbies, income levels and possible combinations of these dimensions. This lets users analyze the underlying network in a summarized manner within multiple multidimensional spaces, which is typical and of great value in most data warehousing applications. In Facebook and Twitter, advertisers and marketers take advantage of their social networks within different multidimensional spaces to better promote their products via social targeting or viral marketing [2, 22]. In multidimensional networks, however, much of the valuation and interest lies in the network itself. Simple numeric value based group-by's in traditional data warehouses are no longer insightful and of limited usage, because the structural information of the networks is simply ignored. As a result, existing data warehousing and OLAP techniques need to be re-examined and revolutionized in order to improve the potential power and core competency of decision support facilities specifically tailored for multidimensional networks.

Example 2. *Figure 2(a) presents an aggregate network by summarizing the multidimensional network shown in Figure 1 on the dimension "Gender". The vertices with grey color represent condensed vertices "Male" and "Female", and the weight of each vertex means the number of individuals in the original network that comply with the same values for the dimension(s) represented by the condensed vertex. In this case, there are* 5 *males and* 5 *females in the multidimensional network. The edges represent aggregate relationships between condensed vertices while the edge weights present the number of edges in the original network connecting vertices belonging to two condensed vertices, respectively. Self-loops are allowed, as shown for the edges (Male, Male) and (Female, Female). The edge weight that equals* 1 *is not presented in the diagram by default.*

In contrast, Figure 2(b) presents a traditional group-by along the dimension "Gender" on the vertex attribute table, shown in Figure 1(b). In this case, we select COUNT(·) *as the default aggregate operator.*

Figure 3(a) presents another aggregate network by summarizing the original multidimensional network on the dimensions "Gender" and "Location". While Figure 3(b) illustrates a traditional group-by on the vertex attribute table along the dimensions "Gender" and "Location".

As shown in Example 2, a multidimensional network can be summarized to *aggregate networks* in coarser levels of granularity within different multidimensional spaces. During the network aggregation, we consider both vertex coalescence and structure summarization simultaneously, thus resulting in much meaningful and structure-enriched aggregate networks, as illustrated in Figure 2(a) and Figure 3(a). In contrast, the numeric value based aggregation for relational data can be regarded as a special case in our scenario, because the inter-tuple relationships are simply ignored during aggregation, as shown in Figure 2(b) and Figure 3(b). Therefore, the traditional concepts and techniques of data warehousing and OLAP have been enriched in a more structural way for multidimensional networks. Moreover, a set of new OLAP queries can be addressed on multidimensional networks, such as "*What is the network structure as grouped by users' gender?*" The answer is shown in Figure 2(a). We note that there are a lot of connections between males and females in the network (9 edges as the edge weight), while few connections exist between males (only 1 edge as the edge weight). A closer look at this interesting phenomenon could be expressed as a drill-down query: "*What is the network structure as grouped by both gender and location?*" The answer is shown in Figure 3(a). We notice in the aggregate network that between the 2 females in California and the 3 males in Illinois, there exist 5 connections, taking up 55.6% of the total 9 connections between males and females. While the only connection between males actually exists between two males in Illinois. These queries could reveal interesting structural behaviors and potentially insightful patterns, which are very hard, if not impossible, to detect from the original network, as shown in Figure 1.

In this paper, we consider extending decision support facilities on multidimensional networks by introducing a new data warehousing model, Graph Cube, for effective network exploration and summarization. Going beyond traditional data cubes which address simple value-based group-by's on relational data, Graph Cube considers both multidimensional attributes and network structures into one integrated framework for network aggregation. In every potential multidimensional space, the measure of interest now becomes an aggregate network in coarser resolution. In addition, we propose different query models and OLAP solutions for multidimensional networks. Besides traditional cuboid queries with refined structural semantics, a new class of OLAP queries, called *crossboid*, is introduced, which is uniquely useful in the multidimensional network scenario and has not been studied before. An example crossboid query could be "*what is the network structure between users grouped by profession and users grouped by income level?*" Despite definitely OLAP in flavor, this query breaks the boundaries established in the traditional OLAP model in that it straddles two different group-by's simultaneously. We implement Graph Cube by combining special characteristics of multidimensional networks with the existing well-studied data cube techniques. To the best of our knowledge, Graph Cube is the first to systematically address warehousing and OLAP issues on large multidimensional networks, and the solutions proposed in this paper will help improve decision support and business intelligence in large networks.

The contributions of our work can be summarized as follows:

1. We propose a new data warehousing model, Graph Cube, to extend decision support services on multidimensional networks. The multidimensional attributes of the vertices define the dimensions of a graph cube, while the measure turns out to be an aggregate network, which proves to be much more meaningful and comprehensive than numeric statistics examined in traditional data cubes.

2. We formulate different OLAP query models and provide new solutions in the multidimensional network scenario. Besides cuboid queries that explore all potential multidimensional spaces of a graph cube, we introduce a new class of OLAP queries, crossboid, which breaks the boundaries of the traditional OLAP model by straddling multiple different multidimensional spaces simultaneously. Crossboid has shown to be especially useful for network study and analysis.

3. We make use of well-studied partial materialization techniques to implement Graph Cube. Specific characteristics of multidimensional networks are leveraged as well for better implementation alternatives.

4. We evaluate our methods on a variety of real multidimensional networks and the experimental results demonstrate the power and effectiveness of Graph Cube in warehousing and OLAP large networks. In addition, our query processing and cube implementation algorithms have proven to be efficient even for very large networks.

The reminder of this paper is organized as follows. Section 2 gives preliminary concepts and examines the Graph Cube model on multidimensional networks. Section 3 formulates different OLAP queries defined upon Graph Cube. Section 4 focuses on the implementation details of Graph Cube. Experimental studies are shown in Section 5. After discussing the related work in Section 6, we conclude our study in Section 7.

2. THE GRAPH CUBE MODEL

Many networks in real applications can be abstracted as a *multidimensional network*, which is formally defined as follows,

DEFINITION 1. [**MULTIDIMENSIONAL NETWORK**] *A multidimensional network, $\mathcal{N}$, is a graph denoted as $\mathcal{N} = (V, E, A)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges and $A = \{A_1, A_2, \ldots, A_n\}$ is a set of n vertex-specific attributes, i.e., $\forall u \in V$, there is a multidimensional tuple $A(u)$ of u, denoted as $A(u) = (A_1(u), A_2(u), \ldots, A_n(u))$, where $A_i(u)$ is the value of u on i-th attribute, $1 \leq i \leq n$. A is called the dimensions of the network $\mathcal{N}$.*

As explained in Example 1, Figure 1 presents a sample multidimensional network drawn from a real social network. The dimensions of the network are ID, gender, location, profession, and income. For an individual with $ID = 1$ in the network, his corresponding multidimensional tuple is (1, Male, CA, Teacher, 70,000), the first tuple shown in Figure 1(b).

Data warehouses and OLAP for traditional RDB data have developed many mature technologies over the years. Here a brief primer of terminologies is listed. Given a relation R of n dimensions, an n-dimensional *data cube* is a set of 2^n aggregations from all possible group-by's on R. For any aggregation in a form of $(A_1, A_2, \ldots, A_n)$, some (or all) dimension A_i could be $*$ (ALL), representing a super-aggregation along A_i which is equivalent to the removal of A_i during aggregation. There are *cells* in an aggregation, represented as $c = (a_1, a_2, \ldots, a_n : m)$, where a_i is a value of c on i-th dimension, A_i $(1 \leq i \leq n)$, and m is a numeric value, called *measure*, computed by applying a specific aggregate function on c, e.g., COUNT($\cdot$) or AVERAGE($\cdot$). In Example 2, Figure 2(b) presents a group-by on (Gender, *, *), if three dimensions are chosen for aggregation: Gender, Location and Profession, and Figure 3(b) presents a group-by on (Gender, Location, *). Both group-by's adopt COUNT($\cdot$) as the underlying aggregate function.

By analogy, we can define possible aggregations upon multidimensional networks. Based on Definition 1, given a network with n dimensions, there exist 2^n multidimensional spaces (aggregations). However, the measure within each possible space is no longer simple numeric values, but an *aggregate network*, defined as follows,

DEFINITION 2. [**AGGREGATE NETWORK**] *Given a multidimensional network $\mathcal{N} = (V, E, A)$ and a possible aggregation $A' = (A'_1, A'_2, \ldots, A'_n)$ of A, where A'_i equals A_i or $*$, the aggregate network w.r.t. A' is a weighted graph $G' = (V', E', W_{V'}, W_{E'})$, where*

1. $\forall [v]$, a nonempty equivalence class of V, where $[v] = \{v | A'_i(u) = A'_i(v), u, v \in V, i = 1 \ldots n\}$, $\exists v' \in V'$ as a representative of $[v]$. The weight of v', $w(v') = \Gamma_V([v])$, where $\Gamma_V(\cdot)$ is an aggregate function defined upon vertices. v' is therefore called a condensed vertex;

2. *$\forall u', v' \in V'$, and a nonempty edge set $E_{(u',v')} = \{(u, v)| u \in [u]$ represented as u', $v \in [v]$ represented as v', $(u, v) \in E\}$, $\exists e' \in E'$ as a representative of $E_{(u',v')}$. The weight of e', $w(e') = \Gamma_E(E_{(u',v')})$, where $\Gamma_E(\cdot)$ is an aggregate function defined upon edges. e' is therefore called a condensed edge.*

As explained in Example 2, Figure 2(a) and Figure 3(a) present the aggregate networks for the aggregations (Gender, *, *) and (Gender, Location, *), respectively. We choose COUNT($\cdot$) in the example to derive weights for both vertices and edges, while more complicated aggregate functions can be chosen and the aggregate functions for vertices and edges can be different. For example, AVERAGE($\cdot$) can be used if the edges are weighted in the original network. For the sake of brevity, we will choose COUNT($\cdot$) as the default aggregate function to compute both vertex and edge weights, while our model and algorithms can be easily generalized to accommodate other aggregate functions.

Definition 3. **[GRAPH CUBE]** *Given a multidimensional network $\mathcal{N} = (V, E, A)$, the graph cube is obtained by restructuring $\mathcal{N}$ in all possible aggregations of A. For each aggregation A' of A, the measure is an aggregate network G' w.r.t. A', as defined in Definition 2.*

Given a multidimensional network $\mathcal{N} = (V, E, A)$, each aggregation A' of A is often called a *cuboid* [1]. The *size* of a cuboid A' is $(|V'| + |E'|)$, where V' and E' are the vertex and edge set of the aggregate network corresponding to A', respectively. For a cuboid A', $dim(A')$ denotes the set of non-$*$ dimensions of A'. For example, if $A' = (Gender, *, *)$, $dim(A') = \{Gender\}$. Consider two cuboids A' and A''. A' is an *ancestor* of A'' if $dim(A') \subseteq dim(A'')$, and therefore A'' is a *descendant* of A'. Specifically, if $|dim(A'')| = |dim(A')| + 1$, A' is a *parent* of A'', or A'' is a *child* of A'. If $|dim(A')| = |dim(A'')| = l$, then A' and A'' are *siblings*, both of which are at l-th *level* of the graph cube. A distinguished cuboid A_b with $|dim(A_b)| = n$ is called *base cuboid* and it is a descendant of all other cuboids in the graph cube. Another distinguished cuboid $A_{all} = (*, *, \ldots, *)$ where $|dim(A_{all})| = 0$, is called *apex cuboid*. The apex cuboid A_{all} is an ancestor of all other cuboids in the graph cube. If we denote the set of all cuboids of a graph cube as 2^A, i.e., the power set of A, a *graph cube lattice* $\mathcal{L} = \langle 2^A, \subseteq \rangle$ can be induced by the partial ordering $\subseteq$ upon 2^A.

Example 3. *Figure 4 presents a graph cube lattice, each node of which is a cuboid in the graph cube generated from the multidimensional network shown in Figure 1. The edges in the lattice depict the parent-child relationship between two cuboids. The size of each cuboid is shown within the lattice node.*

Given a multidimensional network G with n dimensions, there are 2^n cuboids in the graph cube. For each cuboid in a graph cube, there is a unique aggregate network corresponding to it. Specifically, the original multidimensional network is a special aggregate network corresponding to the base cuboid A_b. While the aggregate network for the apex cuboid A_{all} has a singleton vertex with a possible self-loop.

[1]Here after, we will use equivalently the terms *cuboid*, *aggregation*, *view* and *query*.

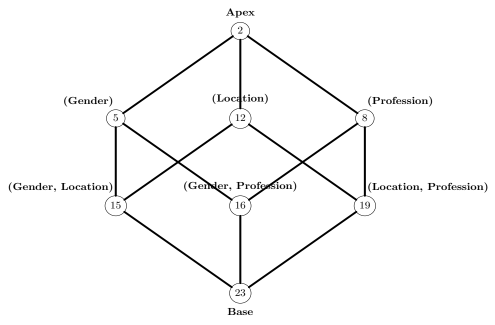


Figure 4: The Graph Cube Lattice

An aggregate network corresponding to an ancestor cuboid is more generalized than the aggregate network corresponding to one of its descendant cuboids, which is fine-grained and contains more attribute/structure details. In the graph cube framework, users can explore the original network in different multidimensional spaces by traversing the graph cube lattice. In this way, a set of aggregate networks with different summarized resolution can be examined and analyzed for decision support and business intelligence purposes.

3. OLAP ON GRAPH CUBE

In traditional OLAP on relational databases, numeric measures can be easily aggregated in the data cube. This naturally leads to queries such as "*What is the average income of females?*" or "*What is the maximum income of software engineers in Washington State?*" For multidimensional networks, however, aggregate networks become the measure of a graph cube. Consider some typical OLAP-style queries that might be asked on a multidimensional network:

1. "*What is the network structure between the various location and profession combinations*?"
2. "*What is the network structure between the user with ID = 3 and various locations?*"

These queries clearly involve some kind of aggregations upon the original network in different multidimensional spaces. What is atypical here is the answers returned. For the first query, the answer is the aggregate network corresponding to the cuboid ($*$, Location, Profession) in the graph cube. While the second query asks for an aggregate network across two different cuboids, (Gender, Location, Profession) and ($*$, Location, $*$). In the following sections, we will formulate and address two different queries posed on the graph cube: (1) cuboid queries in a single multidimensional space and (2) crossboid queries across multiple multidimensional spaces.

3.1 Cuboid Query

An important kind of query on graph cube is to return as output the aggregate network corresponding to a specific aggregation of the multidimensional network. This query is referred to as *cuboid query* because the answer is exactly the aggregate network of the desired cuboid in the graph cube. Algorithm 1 outlines a baseline algorithm to address cuboid queries in detail.

In Algorithm 1, we first create a hash structure, ζ, which maintains a mapping from each distinct tuple w.r.t. the aggregation A', to a condensed vertex in the desired aggregate network G' (Line 2). We then traverse the multidimensional network G. For each vertex u in G, we cre-

Algorithm 1: Cuboid Query Processing

```
Input: A Multidimensional Network $\mathcal{N} = (V, E, A)$, an
       aggregation $A'$
Output: The aggregate network
        $G' = (V', E', W_{V'}, W_{E'})$ w.r.t. $A'$
1  begin
2      Initialize a hash structure $\zeta : A' \rightarrow V'$
3      for $u \in V$ do
4          if $\zeta(A'(u))$ = NULL then
5              Create a condensed vertex $u' \in V'$, labeled
               with $A'(u) = (A'_1(u), A'_2(u), \ldots, A'_n(u))$
6              $u'.weight \leftarrow 0$
7              $\zeta(A'(u)) \leftarrow u'$
8          $\zeta(A'(u)).weight \leftarrow \zeta(A'(u)).weight + 1$
9      for $e(u, v) \in E$ do
10         $u' \leftarrow \zeta(A'(u))$
11         $v' \leftarrow \zeta(A'(v))$
12         if $e'(u', v') \notin E'$ then
13             Create a condensed edge $e'(u', v') \in E'$
14             $e'.weight \leftarrow 0$
15         $e'.weight \leftarrow e'.weight + 1$
16     return $G' = (V', E', W_{V'}, W_{E'})$
```

ate a new condensed vertex u' corresponding to the tuple $(A'_1(u), A'_1(u), \ldots, A'_n(u))$, if there is no such condensed vertex $u' \in V'$ before hand (Lines $4-7$). Otherwise, we simply update the weight for the condensed vertex (Line 8). For each edge $e(u, v)$ in G, we retrieve the mapped condensed vertices u' and v' for the adjacent vertices u and v, respectively (Lines $10-11$). If u' is not adjacent to v' in the aggregate network G', we create a new condensed edge $e'(u', v')$ (Lines $12-14$). Otherwise, we simply update the weight for the condensed edge e'(Line 15). The time complexity of Algorithm 1 is O($|V| + |E|$), the time used for traversing G. The space used to maintain the hash structure ζ is $O(|V'|)$ and we need O($|V'| + |E'|$) space to maintain the aggregate network G'. So the space complexity of Algorithm 1 is O($|V'| + |E'|$).

Based on Algorithm 1, it is straightforward to address all cuboid queries from the multidimensional network, which is exactly the aggregate network corresponding to the base cuboid A_b. However, as the original network could be very large and may not be held in memory, query processing can be extremely time-consuming. Consider two cuboids A' and A'' in the graph cube, if A'' is a descendant of A' ($A'' \neq A_b$) and A'' has been precomputed from A_b based on Algorithm 1, *can we make use of A'' to directly compute A', instead of computing it from A_b?* The following theorem guarantees a positive answer to this question.

THEOREM 1. *Given two cuboids A' and A'' in a graph cube, where $dim(A') \subseteq dim(A'')$ and $A'' \neq A_b$, the cuboid query A' can be answered directly from A''.*

PROOF. Refer to the Appendix. □

Based on Theorem 1, if the cuboid A'' has been precomputed, A' can be answered directly from A'', and not necessarily from the base cuboid A_b. The algorithm is the same as Algorithm 1 except that we change the input from the original multidimensional network G corresponding to A_b to the

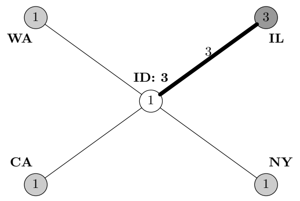


Figure 5: The Aggregate Network between a User (ID=3) and Various Locations

aggregate network G'' corresponding to A''. In this way, the time complexity of the algorithm becomes O($|V''| + |E''|$), way better than O($|V| + |E|$) for the baseline algorithm. Theoretically the aggregate network G'' is no greater than the original multidimensional network G, while in practice, G'' can be much smaller than G for some cuboids in the graph cube.

Now if we have a set of precomputed cuboids A'', which one should we choose to compute A'? We define the *descendant set* of cuboid A', $des(A')$, as $des(A') = \{A'' | dim(A') \subseteq dim(A''), A''$ is in the graph cube$\}$. Based on the aforementioned complexity analysis, the following cuboid A^* will be selected:

$$A^* = arg\min(size(A'')), A'' \in des(A') \qquad (1)$$

So, we always choose the precomputed cuboid, A^*, whose size, ($|V^*| + |E^*|$), is minimal among all cuboids in $des(A')$ to answer the cuboid query A'.

When all the cuboids have been computed, the support of OLAP operations, such as *roll-up*, *drill-down*, and *slice-and-dice*, becomes straightforward in the graph cube framework. Roll-up means going from a cuboid to one of its ancestors, such that we can summarize an aggregate network in finer resolution to another one in coarser resolution. Drill-down, on the contrary, goes from an ancestor cuboid to one of its descendants. As shown in Example 2, after examining the aggregate network for the cuboid (Gender, $*$, $*$), users may be more interested in how males interact with females across different locations. We can drill-down to the cuboid (Gender, Location, $*$) and more interesting interaction patterns in finer resolution can be discovered. As to slice-and-dice, selections are performed on a cuboid and an induced aggregate network will be generated as a result. For example, users may be interested in the network structure between NY and CA, aggregated by Location. A slice-and-dice operation can be performed upon the cuboid ($*$, Location, $*$) and only the interactions between the vertices NY and CA (including self-loops, if possible) are returned as output. Different OLAP operations can be further combined as advanced OLAP queries on the graph cube and they have formed a powerful tool set and new query mechanism on multidimensional networks.

3.2 Crossboid Query

Cuboid query discussed in Section 3.1 is the query within a single multidimensional space, which follows the traditional OLAP model proposed on relational data [7]. What is more interesting, however, is that multidimensional networks introduce a new kind of query, which crosses multiple multidimensional spaces of the network, i.e., more than one cuboid is involved in a query. We thus call such queries crossing different cuboids of the graph cube as *cross-cuboid* queries, or *crossboid* queries for short.

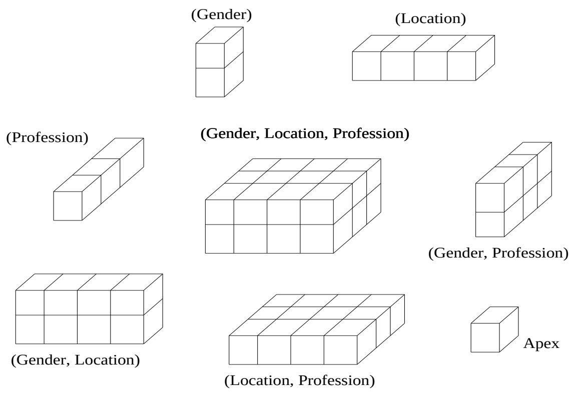


Figure 6: Traditional Cuboid Queries

EXAMPLE 4. *Consider the query proposed in Section 3:* "What is the network structure between the user 3 and various locations?" *The answer is shown in Figure 5. In the network, the vertex with white color represents user* 3 *in the cuboid (Gender, Location, Profession), while all the other vertices with grey color are different locations in the cuboid* $(*, \text{Location}, *)$. *Edges indicate relationships between user* 3 *and her friends grouped by different locations. For instance, this user has* 3 *friends at Illinois state, represented by the edge with weight* 3.

Example 4 shows a crossboid query in the multidimensional network. This query definitely has an OLAP flavor in that it involves aggregation upon the network. However, it deviates significantly from traditional OLAP semantics. In traditional OLAP on relational data, for example, it does not make sense to query the average income of user 3, a numeric value, with various locations. Although it is natural to compare user 3's income with the average income of users at Illinois, this comparison, however, is orthogonal to OLAP. In the multidimensional network scenario, aggregation involving multiple cuboids becomes possible within a single query, which is unique to the graph cube.

For a more graphical explanation of this difference, Figure 6 shows a 3-dimensional data cube on traditional relational data. In this model, queries exist wholly within a single cuboid. Note cuboid queries discussed in Section 3.1 follow this query model as well. For instance, the answer for "*What is the aggregate network between two genders*?" comes solely from the 1-dimensional (Gender) cuboid.

In contrast, the crossboid query in Example 4 breaks the traditional OLAP semantics and straddles two distinct cuboids in different levels of the graph cube. Figure 7 shows this query graphically: the dashed lines between cuboids present the regions in which crossboid queries are interested. For instance, the right region corresponds to the aggregate network between base cuboid and the (Location) cuboid. Imagine the black dot inside the 3-dimensional base cuboid is the user 3. The aggregate network shown in Figure 5 is the exact answer to this crossboid query if we slice-and-dice the right region only for the user 3. Similarly, the dashed region between the (Gender) cuboid and the (Location) cuboid on the left corresponds to a crossboid query: "*what is the network structure between users grouped by gender vs. users grouped by location?*". Here the crossboid query straddles two distinct cuboids (Gender) and (Location) in the same level of the graph cube, and the query answer is shown in Figure 8.

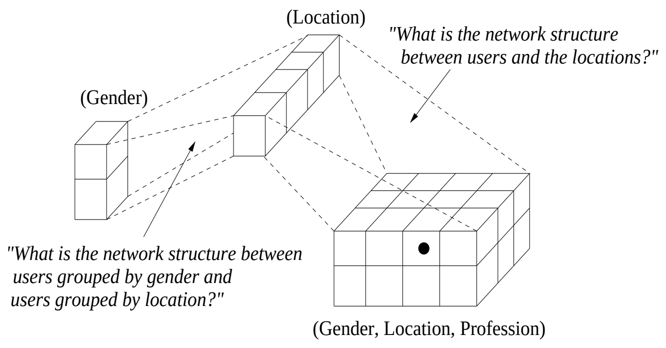

(Location)
"What is the network structure between users and the locations?"
(Gender)
"What is the network structure between users grouped by gender and users grouped by location?"
(Gender, Location, Profession)

Figure 7: Crossboid Queries Straddling Multiple Cuboids

In general, a crossboid query can include any number of cuboids from the graph cube. As shown in Figure 7, three different cuboids can be linked together to form a crossboid. In the rest of the paper, we focus on the crossboid queries straddling two cuboids, while our model can be generalized to address crossboid queries spanning multiple cuboids.

DEFINITION 4. **[CROSSBOID QUERY]** *Given two distinct cuboids S and T in the graph cube, the crossboid query, $S \bowtie T$, is a bipartite aggregate network, $G' = (V'_S \bigcup V'_T, E', W_{V'}, W_{E'})$, where for each vertex u in the multidimensional network G, it is aggregated into a condensed vertex $u' \in V'_S$ w.r.t. S and another condensed vertex $u'' \in V'_T$ w.r.t. T, respectively. For each edge $e(u,v) \in G$, it is aggregated into two condensed edges $e'(u', v'')$ and $e'(v', u'')$, respectively, where u' (v') and u'' (v'') are the condensed vertices for $u(v)$ to be aggregated to w.r.t. S and T, respectively. The weights for both condensed vertices and edges, $W_{V'}, W_{E'}$, are determined in the same way as dictated in Definition 2.*

In Definition 4, we abuse the *join* operator, $\bowtie$, to denote a crossboid query between two cuboids, S and T. Given an n-dimensional network, the graph cube contains $2^{n-1} \times (2^n - 1)$ crossboids if $S \neq T$ (note $S \bowtie T = T \bowtie S$). More specifically, if $S = T$, crossboid queries boil down to cuboid queries, as discussed in Section 3.1. That is, cuboid query is just a special case of crossboid query when $S = T$. Therefore, given a graph cube, there are 2^n cuboids and $(2^{2n-1} - 2^{n-1})$ crossboids, respectively, resulting in a total of $(2^{2n-1} + 2^{n-1})$ OLAP queries to be addressed.

Algorithm 2 presents a detailed procedure to address the crossboid query, $S \bowtie T$, from the multidimensional network G. It is similar to Algorithm 1, while for each vertex u in the network, we need to aggregate it to cuboid S and T, respectively (Lines $3 - 13$). And for each edge $e(u,v)$ in the network, we need to create or update two condensed edges $e'(u', v'')$ and $e'(v', u'')$ (Lines $14 - 24$). The reason is that

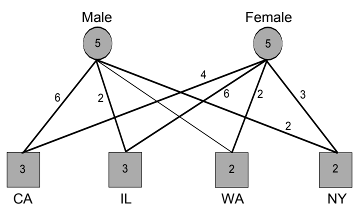


Figure 8: The Aggregate Network to the Crossboid Query Straddling (Gender) and (Location) Cuboids

Algorithm 2: Crossboid Query Processing

```
Input: A Multidimensional Network $\mathcal{N} = (V, E, A)$,
       cuboids $S$ and $T$
Output: The aggregate network $S \bowtie T = (V'_S \bigcup V'_T, E', W_{V'}, W_{E'})$
1  begin
2      Initialize two hash structures $\zeta_S : S \to V'_S$ and $\zeta_T : T \to V'_T$
3      for $u \in V$ do
4          if $\zeta_S(S(u)) =$ NULL then
5              Create a condensed vertex $u' \in V'_S$, labeled with $S(u) = (S_1(u), S_2(u), \dots, S_n(u))$
6              $u'.weight \leftarrow 0$
7              $\zeta_S(S(u)) \leftarrow u'$
8          $\zeta_S(S(u)).weight \leftarrow \zeta_S(S(u)).weight + 1$
9          if $\zeta_T(T(u)) =$ NULL then
10             Create a condensed vertex $u'' \in V'_T$, labeled with $T(u) = (T_1(u), T_2(u), \dots, T_n(u))$
11             $u''.weight \leftarrow 0$
12             $\zeta_T(T(u)) \leftarrow u''$
13         $\zeta_T(T(u)).weight \leftarrow \zeta_T(T(u)).weight + 1$
14     for $e(u, v) \in E$ do
15         $u' \leftarrow \zeta_S(S(u))$, $v'' \leftarrow \zeta_T(T(v))$
16         if $e'(u', v'') \notin E'$ then
17             Create a condensed edge $e'(u', v'') \in E'$
18             $e'(u', v'').weight \leftarrow 0$
19         $e'(u', v'').weight \leftarrow e'(u', v'').weight + 1$
20         $v' \leftarrow \zeta_S(S(v))$, $u'' \leftarrow \zeta_T(T(u))$
21         if $e'(v', u'') \notin E'$ then
22             Create a condensed edge $e'(v', u'') \in E'$
23             $e'(v', u'').weight \leftarrow 0$
24         $e'(v', u'').weight \leftarrow e'(v', u'').weight + 1$
25     return $G' = (V', E', W_{V'}, W_{E'})$
```

edge $e(u, v)$ in the original network is undirected and we need to consider the interaction between two condensed vertices from both directions. The time complexity of Algorithm 2 is $O(|V| + |E|)$. And if there are $|V_S|$ and $|V_T|$ condensed vertices in the aggregate networks w.r.t. the cuboid S and T, respectively, the space complexity of Algorithm 2 is $O(|V_S| \times |V_T|)$.

Given a multidimensional network G and two cuboids S, T in the graph cube ($S \neq T$), we can answer the crossboid query, $S \bowtie T$, based on Algorithm 2. However, it becomes extremely inefficient to compute every crossboid from the original network G. *Can we address crossboid queries by leveraging precomputed cuboids in the graph cube?* Before giving a positive answer to this question, we first define the *nearest common descendant*, $ncd(S, T)$, of two cuboids S, T in the graph cube, as follows,

Definition 5. [$\mathbf{NCD(S, T)}$] *The common descendant set of cuboids S and T in a graph cube, $cd(S, T)$, is defined as $cd(S, T) = des(S) \bigcap des(T)$. Then the nearest common descendant of S and T, $ncd(S, T)$, is a cuboid in the graph cube, such that $ncd(S, T) \in cd(S, T)$, and $\not\exists U \in cd(S, T)$, $dim(U) \subseteq dim(ncd(S, T))$.*

Example 5. *As shown in Figure 3, for cuboids (Gender) and (Profession), both the base cuboid and the (Gender, Profession) cuboid are their common descendants. However, only the (Gender, Profession) cuboid is the nearest common descendant.*

Theorem 2. *Given two cuboids S and T in the graph cube ($S \neq T$), the crossboid query $S \bowtie T$ can be answered directly from the cuboid $ncd(S, T)$.*

Proof. Refer to the Appendix. □

Based on Theorem 2, we can compute the crossboid query $S \bowtie T$ from $ncs(S, T)$ instead of the original network. Note $ncs(S, T)$ can be easily derived because $dim(ncs(S, T)) = dim(S) \bigcup dim(T)$. In this way, the time complexity of the algorithm becomes $O(|V_{ncs(S,T)}| + |E_{ncs(S,T)}|)$, which is way better than Algorithm 2 because the aggregate network w.r.t. $ncs(S, T)$ is always no greater than the original network.

4. IMPLEMENTING GRAPH CUBES

In order to implement a graph cube, we need to compute the aggregate networks of different cuboids grouping on all possible dimension combinations of a multidimensional network. (Note for a crossboid query, it can be indirectly answered by the nearest common descendant, which is a cuboid in the graph cube as well.) Such implementation of a graph cube is critical to improve the response time of OLAP queries and of operators such as roll-up, drill-down and slice-and-dice. The following implementation alternatives are possible:

1. **Full materialization:** We physically materialize the whole graph cube. This approach can definitely achieve the best query response time. However, precomputing and storing every aggregate network is not feasible for large multidimensional networks, in that we have 2^n aggregate networks to materialize and the space consumed could become excessively large. Sometimes it is even hard, if not impossible, to explicitly maintain the multidimensional network itself into main memory.
2. **No materialization:** We compute every cuboid query on request from the raw data. Although no extra space is required for materialization, the query response time can be slow because we have to traverse the multidimensional network once for each such query.
3. **Partial materialization:** We selectively materialize a subset of cuboids in the graph cube, such that queries can be addressed by the materialized cuboids, in light of Theorem 1 and 2. Empirically, the more cuboids we materialize, the better query performance we can achieve. In practice, due to the space limitation and other constraints, only a small portion of cuboids can be materialized in order to balance the tradeoff between query response time and cube resource requirement.

There have been many algorithms invented for cube implementation in the context of relational data [17], most of which chose to optimize the partial materialization approach that has proven to be NP-complete by a straightforward reduction from the *set cover* problem [10]. In [11], the authors further proved that partial materialization is inapproximable if P≠NP. Therefore, the ongoing research was mainly motivated to propose heuristics for sub-optimal solutions. Note the partial materialization problem in the graph cube scenario is still NP-complete because the problem in traditional data cubes can be regarded as a special case in our setting. Therefore the main focus here is to select a set S of k cuboids ($k < 2^n$) in the graph cube for partial materialization, such that the average time taken to evaluate OLAP queries can be minimized.

As it turns out, most of the existing algorithms proposed on data cubes can be used to implement the graph cube with minor modification. We adopt a greedy algorithm [10] for partial materialization on the graph cube. We define the *cost*, $C(v)$, of a cuboid v in the graph cube as the size of v, i.e., $C(v) = (|V'|+|E'|)$, where $G'(V', E')$ is the corresponding aggregate network of v. Advanced sampling methods [8, 13] can be used for estimating both $|V'|$ and $|E'|$ of the aggregate network. Assume the set S has already contained some materialized cuboids ($|S| < k$), the *benefit* to further including v into S, denoted by $B(S, v)$, is the total reduction cost of the cuboids in the graph cube if v is involved for cuboid computation. Formally,

$$B(S, v) = \sum_{dim(u) \subseteq dim(v)} (C(v) - C(w^*)) \qquad (2)$$

where

$$w^* = arg \min(C(w)), w \in des(u) \cap S$$

That is, we compute the benefit introduced by v, which indicates how much it can improve the cost for query evaluation in the presence of S. To this point, the greedy algorithm becomes straightforward: we initially include the base cuboid in S. Then we iterate for k times and for each iteration, we select the cuboid v with the highest benefit $B(S, v)$ into S. Note in practice the network corresponding to the base cuboid is usually too large to be materialized, so we actually compute $(k + 1)$ cuboids in S. In this way the base cuboid can be excluded while the other k cuboids are materialized. The time complexity of the greedy algorithm is $O((k + 1)N^2)$, where N is the total number of cuboids in the graph cube, or $O((k + 1)2^{2n})$, where n is the number of dimensions in the multidimensional network.

THEOREM 3. *Let B_{greedy} be the benefit of k cuboids chosen by the greedy algorithm and let B_{opt} be the benefit of any optimal set of k cuboids, then $B_{greedy} \leq (1-1/e) \times B_{opt}$ and this bound is tight.*

PROOF. Refer to [10]. $\square$

We make no assumption in the greedy algorithm about the query workload and distribution, i.e., all the cuboids in the graph cube will be queried with equal probability. However, it has been shown in the data cube scenario that most OLAP queries and operations are performed only on the cuboids with small number of dimensions, e.g., from 1 to 3 [14]. This evidence still holds for graph cubes. The main reason is that the aggregate networks corresponding to the cuboids with large set of dimensions can be massive and with comparable size to the original multidimensional network. So it is still hard to materialize these large aggregate networks explicitly. On the other hand, users will be easily overwhelmed by the large networks and the insights gained can be limited. Instead, they are more likely to query the cuboids with small sets of dimensions and crosscheck afterwards the corresponding aggregate networks with manageable size, for example, with tens of vertices and edges. Drill-downs are selectively performed only on few cuboids of interest toward the cuboids with medium size. To this end, we propose another heuristic algorithm, *MinLevel*, to materialize the cuboid c where $dim(c) = l_0$. l_0 is an empirical value specified by users, which indicates the level in the cube lattice at which we start materializing cuboids that contain l_0 non-$*$ dimensions. If all the cuboids with l_0 dimensions are included in S and $|S| < k$, we continue choosing cuboids with $(l_0 + 1)$ dimensions, until $|S| = k$. The time complexity of MinLevel is $O(k)$, which is irrelevant to the number of dimensions, n. In practice, MinLevel has proven to be a more efficient and practical approach for graph cube materialization, compared to the greedy algorithm.

Area	**Conferences**
DB	SIGMOD, VLDB, ICDE, PODS, EDBT
DM	KDD, ICDM, SDM, PKDD, PAKDD
IR	SIGIR, WWW, CIKM, ECIR, WSDM
AI	IJCAI, AAAI, ICML, CVPR, ECML

Table 1: Major Conferences Chosen For Each Research Area

Productivity	**Publication Number x**
Excellent	$50 < x$
Good	$21 \leq x \leq 50$
Fair	$6 \leq x \leq 20$
Poor	$x \leq 5$

Table 2: Four Different Buckets of the Publication Number for the Productivity Attribute

5. EXPERIMENTS

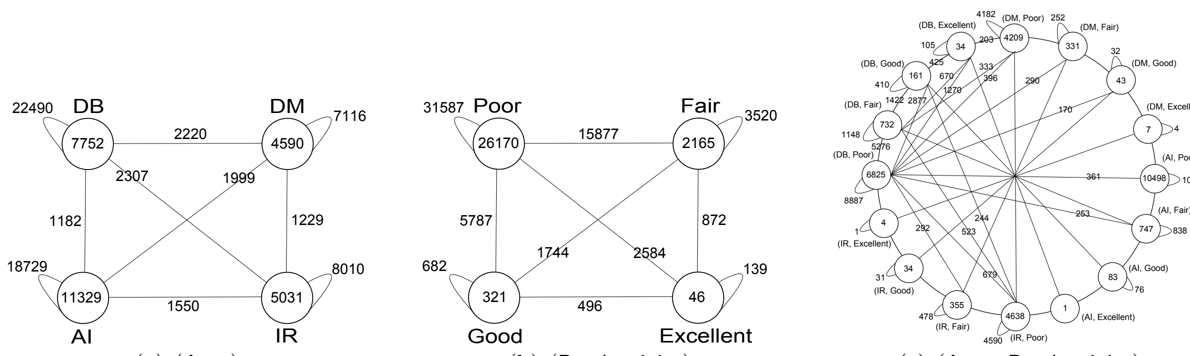
In this section, we present the major experimental results of our proposed method, Graph Cube. We examine two real data sets and our evaluation is conducted from both effectiveness and efficiency perspectives. All our algorithms and experimental methods are implemented in C++ and tested on a Windows PC with AMD triple-core processor 2.1GHz and 3G of RAM.

5.1 Experimental Data Sets

We perform our experimental studies on two real-world data sets: DBLP [2] and IMDB [3]. Specifically, we will focus the effectiveness study on the DBLP data set and the efficiency study on both data sets. The details of the two data sets are given as follows,

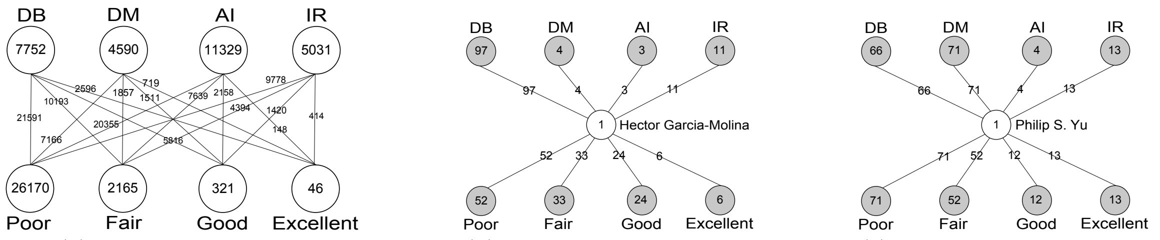
DBLP Data Set. This data set contains the DBLP Bibliography data downloaded in March, 2008. We further extract a subset of publication information from major conferences in four different research areas: database (DB), data mining (DM), artificial intelligence (AI) and information retrieval (IR). Table 1 shows the conferences we choose for each of the four research areas. We build a co-authorship graph with 28,702 authors as vertices and 66,832 coauthor relationships as edges. For each author, there are three dimensions of information: Name, Area, and Productivity. Area specifies a research area the author belongs to. Although an author may belong to multiple research areas, we select one only among the four in which she/he publishes most. For Productivity, we discretize the publication number of an author into four different buckets, as shown in Table 2.

[2]http://www.informatic.uni-trier.de/~ley/db/
[3]http://www.imdb.com/interfaces



(a) (Area) (b) (Productivity) (c) (Area, Productivity)

Figure 9: Cuboid Queries of the Graph Cube on DBLP Data Set



(a) Area ⋈ Productivity (b) Area ⋈ Base ⋈ Productivity for "Hector Garcia-Molina" (c) Area ⋈ Base ⋈ Productivity for "Philip S. Yu"

Figure 10: Crossboid Queries of the Graph Cube on DBLP Data Set

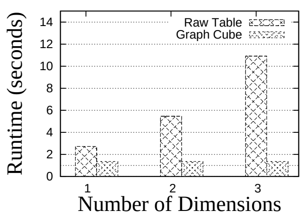
IMDB Data Set. This data set was extracted from the Internet Movies Data Base (IMDB) in September, 2010. It contains movie information including the following dimensions : Title, Year, Length, Budget, Rating, MPAA and Type. For the dimension Length, we further discretize it into short (within 30 minutes), medium (between 30 and 90 minutes) and long (beyond 90 minutes). For the dimension Budget, we bucketize it into 10 different histograms with the equal-width method. For the dimension Rating, we discretize the original absolute rating scores to the ten-star grading criterion, in which the more stars a movie gets, the better rating it has. For MPAA (The Motion Picture Association of America), it classifies a movie into one of the following categories: {G, PG, PG-13, R, NC-17, NR}. And for the dimension Type, it contains the following seven different genres for a movie: action, animation, comedy, drama, documentary, romance and short. Based on this data set, we build a movie network as follows. For each movie there is a corresponding vertex in the network. And there is an edge between two movies if they both share the same rating value. There are 116, 164 vertices and 5, 452, 350 edges in the network.

5.2 Effectiveness Evaluation

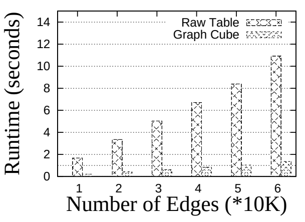
We first evaluate the effectiveness of Graph Cube as a powerful decision-support tool in the DBLP co-authorship network. We will present some interesting findings by addressing OLAP queries on the network. The summarized aggregate networks demonstrate a new angle to study and explore massive networks. In the experiments, we are interested in the co-authorship patterns between researchers from different perspectives. Upon the graph cube built on the DBLP co-authorship network, we first issue a cuboid query (Area) and the resulting aggregate network is shown in Figure 9(a). This aggregate network is a complete graph K_4 illustrating the co-authorship patterns between researchers grouped by different research areas. Note different research communities exhibit quite different co-authorship patterns. For example, among the four research areas we study, the DB community cooperates a lot with the IR community and the DM community, while the cooperations between DB and AI are not that frequent. More interestingly, the DB community cooperates most with itself (22, 490 coauthor relationships), compared with other communities. The AI community and the DM community cooperate a lot partially because some common algorithms and methods, e.g., SVM or k-means, are frequently shared by both communities.

Figure 9(b) presents another aggregate network corresponding to the cuboid query (Productivity) in the graph cube. This aggregate network illustrates the co-authorship patterns between researchers grouped by different productivity. As shown in the figure, the researchers with poor productivity (with publication number no greater than 5) take up around 91.2% of all the researchers we are examining. For this group of researchers, they cooperate a lot with researchers of fair productivity, while they cooperate much less with researchers of good or excellent productivity. If we define *density* of a condensed vertex v as $density(v) = w_E(e(v, v))/w_V(v)$, where the numerator denotes the weight of the self-loop edge of v and the denominator denotes the weight of v itself, then $density(Excellent) = 3.02$, which is much larger than the density values of Poor (1.207) and Fair (1.626) vertices. It means that excellent researchers have formed closer and more compact co-authorship patterns and the in-between cooperations are significantly more frequent than other groups.

If users are interested in zooming into a more fine-grained network for further inspection, a drill-down operation can be performed, or equivalently, a cuboid query (Area, Productivity) is addressed. The resulting aggregate network is a complete graph K_{16}, shown in Figure 9(c). For the sake of

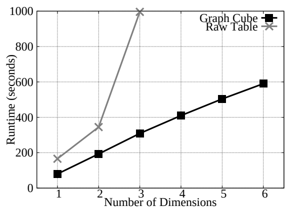


(a) Time vs. # Dimensions

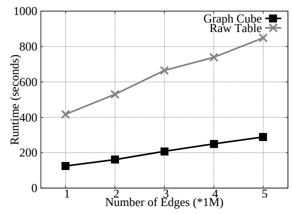


(b) Time vs. # Edges

Figure 11: Full Materialization of the Graph Cube for DBLP Data Set



(a) Time vs. # Dimensions



(b) Time vs. # Edges

Figure 12: Full Materialization of the Graph Cube for IMDB Data Set

clarity, we only illustrate part of the edges (with large edge weights) of the network. In this aggregate network, every vertex is in the (Area, Productivity) resolution, and therefore presents more detailed information for research cooperation. For example, for researcher in the DB community with good productivity (represented as the vertex (DB, Good)), they cooperate most with DB researchers of poor productivity, while they cooperate much less with DM or AI researchers. Interestingly, they cooperate frequently with researchers of poor productivity in IR community as well.

After examining the cuboid queries upon the graph cube, we further address different crossboid queries. Figure 10(a) present a crossboid query Area $\bowtie$ Productivity straddling two different cuboids (Area) and (Productivity) in the same level of the graph cube. The aggregate network presents a quite different view of co-authorship patterns by cross-checking interactions between research areas and the productivity of researchers. An interesting finding as shown in Figure 7(a) is that, although DB is not the largest research community (actually, AI is the largest one), it consistently attracts the highest number of researchers for cooperation across various levels of productivity, compared with the other three communities. From another direction, excellent researchers cooperate with the DB community most, and then the DM community, followed by the IR and AI community. And for the researchers with poor productivity, they cooperate frequently with the DB and AI community, while their cooperation with DM and IR is much less.

Figure 7(b) and Figure 7(c) present another crossboid query Area $\bowtie$ Base $\bowtie$ Productivity straddling three cuboids in different levels of the graph cube. We further slice-and-dice the result to show the cooperation patterns for specific researchers “Hector Garcia-Monlina” and “Philip S. Yu”, respectively. From Figure 7(b), it is pretty clear that Hector cooperated with researchers in the DB community most, and the number of cooperations is much larger than that in other areas. And he cooperated extensively with researchers in different productivity levels. In contrast, Philip cooperated almost equally frequently with both the DB community and the DM community. And he cooperated more with researchers in poor, fair and excellent productivity.

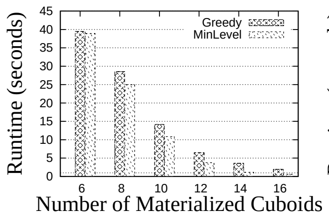
5.3 Efficiency Evaluation

In this section, we evaluate the efficiency of our Graph Cube method. We also test different Graph Cube implementation techniques on multidimensional networks.

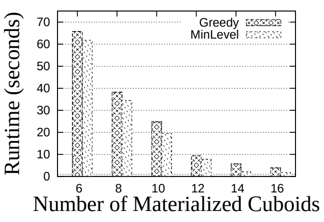
We first evaluate our algorithms on the DBLP data set. As this data set contains 3 dimensions only, it is fairly easy to hold all cuboids in main memory. We thus focus on the efficiency of full cube materialization on this data set. The raw network data is first stored on disk and we start building the graph cube based on Algorithm 1, which is a baseline method, denoted as Raw Table. Note for each cuboid in the graph cube, Raw Table has to access the disk for cuboid computation, which is inefficient. Graph Cube adopts a bottom-up method to compute cuboids and the intermediate results can be shared to facilitate the computation of ancestor cuboids, as described in Section 3.1. As shown in Figure 11(a), for different numbers of dimensions in the underlying network, the time consumed for two competing methods is significantly different. Graph Cube is consistently faster than Raw Table. More specifically, when the dimension value is 3, which means we need to materialize the full cube, Graph Cube is about 10 times faster than Raw Table. Figure 11(b) presents the time used for full cube materialization, while in this case, we start varying the size of the underlying network by changing the number of edges. As illustrated, both methods grow linearly w.r.t. the network size. However, Graph Cube outperforms Raw Table for $8-10$ times.

We then perform the same experiments on the IMDB data set. The raw network data is first stored on disk. In this experiment, we explicitly drop the Name dimension and keep the remaining 6 dimensions. And the pre-computed cuboids by Graph Cube are stored on disk as well because of limited space resource. As illustrated in Figure 12(a), Graph Cube can compute the full graph cube within 10 minutes. Although the cuboids on low levels still need to access the disk for the pre-computed cuboids, the intermediate aggregate networks are much smaller than the original network. In comparison, Raw Table spends around $1,000$ seconds when the network dimension is 3, and the time spent grows exponentially large w.r.t. the dimension. Raw Table therefore becomes extremely inefficient for the networks with high dimensionality. In Figure 12(b), we set the network dimension to be 3 and start varying the network size. As shown, Graph Cube still outperforms Raw Table for up to 4 times.

We then turn to another implementation alternative to partial-materialize the graph cube. In this experiment, we select a set of 20 cuboids in the graph cube with estimated size no greater than $1,000$ and use them as cuboid queries. We further choose arbitrary pairs of these cuboids to compose another set of 200 crossboid queries. The rationale to choose these queries is that, users seldom explore the aggregate networks whose sizes are larger than $1,000$. We compare two different partial materialization algorithms to address both cuboid queries and crossboid queries: the greedy algorithm, denoted as Greedy, and the heuristic algorithm, MinLevel, as described in Section 4. We set the materialization level $l_0 = 3$ for MinLevel to start materializing cuboids from the level 3 of the graph cube. The average response time of different queries are reported in Figure 13. By varying the number k of cuboids to be materialized into main



(a) Cuboid Queries



(b) Crossboid Queries

Figure 13: Average Query Respond Time w.r.t. Different Partial Materialization Algorithms

memory, we notice that MinLevel outperforms Greedy consistently, for both cuboid queries (Figure 13(a)) and crossboid queries (Figure 13(b)). The main reason is that, it is of little use to materialize a very large cuboid with great benefit, because this query is seldom issued on the graph cube. Instead, most of the commonly issued queries (with manageable size) can be successfully answered by the materialized cuboids chosen by MinLevel.

6. RELATED WORK

As key ingredients in decision support systems (DSS), data warehouses and OLAP have demonstrated competitive advantages for business, and kindled considerable research interest in the study of multidimensional data models and data cubes [7, 4]. In recent years, significant advances have been made to extend data warehousing and OLAP technology from the relational databases to new emerging data in different application domains, such as imprecise data [3], sequences [16], taxonomies [21], text [15] and streams [9]. A recent study by Chen et al. [5] aims to provide OLAP functionalities on graphs. However, the problem definition is different from Graph Cube. In [5], the input data is a *set* of graphs, each of which contains graph-related and vertex-related attributes. The algorithmic focus is to aggregate (overlay) multiple graphs into a summary static graph. In contrast, Graph Cube focuses on OLAP inside a single large graph. Furthermore, a set of aggregated networks with varying size and resolution is examined in the lens of multidimensional analysis.

Graph summarization is a field closely related to our work. Scientific applications such as DNA analysis and protein synthesis often involve large graphs, and effective summarization is crucial to help biologists solve their problems [19]. One path of approach for graph summarization is to compress the input graph [6, 18]. Such compressed graphs can be effectively used to summarize the original graph. Graph clustering [26] is another approach that partitions the graph into regions that can be further summarized. GraSS [12] summarizes graph structures based on a random world model and the target of summarization is to help improve the accuracy of common queries, such as adjacency, degree and eigenvector centrality. And finally, graph visualization [24] addresses the problem of summarization as well. In relation to our work, however, most of the aforementioned studies have not had multidimensional attributes assigned on the network vertices. As a result, the summarization techniques are free to choose the groupings and do not have to respect semantic differences between the vertices. In contrast, Graph Cube approaches the problem from a more data cube and OLAP angle, which has to honor the semantic boundaries to match decision support operations. The systematic aggregation in the multidimensional spaces and also the large network analysis aspects are topics not addressed in the above studies.

One interesting recent work by Tian et al. [23] and Zhang et al. [25] brings an OLAP flavor to graph summarization. It introduces the SNAP operation and a less restrictive k-SNAP operation that will aggregate the graph to a summarized version based on user input of attributes and edge relationships. As the authors mentioned, it is similar to OLAP-style aggregations. In contrast to Graph Cube, k-SNAP performs roll-up and drill-down by deceasing and increasing the number k of node-groupings in the summary graph, which is like specifying the number of clusters in clustering algorithms. While for Graph Cube, aggregation and OLAP operations are performed along the dimensions defined upon the network. Moreover, Graph Cube proposes a new class of OLAP queries, crossboid, which is new and has not been studied before.

7. CONCLUSIONS

In this paper, we have addressed the problem of supporting warehousing and OLAP technology for large multidimensional networks. Due to the recent boom in large-scale networks, businesses and researchers seek to build infrastructures and systems to help enhance decision-support facilities on networks in order to summarize and maximize the potential value around them. This work has studied this exact problem by first proposing a new data warehousing model, Graph Cube, which was designed specifically for efficient aggregation of large networks with multidimensional attributes. We formulated different OLAP query models for Graph Cube and proposed a new class of queries, crossboid, which broke the boundary of the traditional OLAP model by straddling multiple cuboids within one query. We studied the implementation details of Graph Cube and our experimental results have demonstrated the power and efficacy of Graph Cube as the first, to the best of our knowledge, tool for warehousing and OLAP large multidimensional networks. However, this is merely the tip of the iceberg. The marriage of network analysis and warehousing/OLAP technology brings many exciting opportunities for future study.

Acknowledgments

The work was supported in part by the NSF IIS-09-05215, and by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053 (NS-CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

8. REFERENCES

[1] C. C. Aggarwal and H. Wang. *Managing and Mining Graph Data*. Springer, 2010.
[2] T. Baird. *The Truth About Facebook*. Emereo Pty Limited, 2009.
[3] D. Burdick, A. Doan, R. Ramakrishnan, and S. Vaithyanathan. OLAP over imprecise data with domain constraints. In *VLDB*, pages 39–50, 2007.

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

[5] C. Chen, X. Yan, F. Zhu, J. Han, and P. S. Yu. Graph OLAP: Towards online analytical processing on graphs. In *ICDM*, pages 103–112, 2008.

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

[7] 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.

[8] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *VLDB*, pages 311–322, 1995.

[9] J. Han, Y. Chen, G. Dong, J. Pei, B. W. Wah, J. Wang, and Y. D. Cai. Stream cube: An architecture for multi-dimensional analysis of data streams. *Distrib. Parallel Databases*, 18(2):173–197, 2005.

[10] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *SIGMOD*, pages 205–216, 1996.

[11] H. Karloff and M. Mihail. On the complexity of the view-selection problem. In *PODS*, pages 167–173, 1999.

[12] K. LeFevre and E. Terzi. GraSS: Graph structure summarization. In *SDM*, pages 454–465, 2010.

[13] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *KDD*, pages 631–636, 2006.

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

[15] C. X. Lin, B. Ding, J. Han, F. Zhu, and B. Zhao. Text cube: Computing IR measures for multidimensional text database analysis. In *ICDM*, pages 905–910, 2008.

[16] E. Lo, B. Kao, W.-S. Ho, S. D. Lee, C. K. Chui, and D. W. Cheung. OLAP on sequence data. In *SIGMOD*, pages 649–660, 2008.

[17] K. Morfonios, S. Konakas, Y. Ioannidis, and N. Kotsis. ROLAP implementations of the data cube. *ACM Comput. Surv.*, 39(4):12, 2007.

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

[19] S. Navlakha, M. C. Schatz, and C. Kingsford. Revealing biological modules via graph summarization. *Journal of Computational Biology*, 16:253–264, 2009.

[20] M. Newman. *Networks: An Introduction.* Oxford University Press, 2010.

[21] Y. Qi, K. S. Candan, J. Tatemura, S. Chen, and F. Liao. Supporting OLAP operations over imperfectly integrated taxonomies. In *SIGMOD*, pages 875–888, 2008.

[22] H. Thomases. *Twitter Marketing: An Hour a Day.* Wiley, 2010.

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

[24] M. Wattenberg. Visual exploration of multivariate graphs. In *CHI*, pages 811–819, 2006.

[25] N. Zhang, Y. Tian, and J. M. Patel. Discovery-driven graph summarization. In *ICDE*, pages 880–891, 2010.

[26] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.

APPENDIX

Proof of Theorem 1

As $dim(A') \subseteq dim(A'')$, there exists at least one dimension A_t $(1 \leq t \leq n)$, s.t. $A_t = *$ for A' but $A_t \neq *$ for A''. For each condensed vertex v' in the aggregate network G' w.r.t. A', $v' = \{v | A'_i(u) = A'_i(v), u, v \in V, i = 1 \dots n\}$ (Note here after we use v' to represent its corresponding nonempty vertex equivalence class, $[v]$, for presentational brevity). We partition the vertices within v' into k disjoint sets, where $v'_j = \{v | A'_i(u) = A'_i(v), u, v \in V, i = 1 \dots n, i \neq t\}$, $1 \leq j \leq k$, i.e., the vertices are partitioned based on the value differences w.r.t. the dimension A_t. Note each v'_j is a condensed vertex for the aggregate network G'' w.r.t. A'', and $v'.weight = \sum_{1 \leq j \leq k} v'_j.weight$.

For each condense edge $e'(u', v')$ in the aggregate network G', $e' = \{(u, v) | u \in u', v \in v', u', v' \in V', (u, v) \in E\}$ (Note here after we use e' to represent its corresponding nonempty edge set, $E_{(u',v')}$, for presentational brevity). We partition the edges within e' into $m \times n$ disjoint sets e'_{ij} $(1 \leq i \leq m$, $1 \leq j \leq n)$, as follows. We first partition the condensed vertex u' into m disjoint sets u'_i, $(1 \leq i \leq m)$, and then partition the condensed vertex v' into n disjoint sets v'_j $(1 \leq j \leq n)$. Both partitions are performed in the same way such that vertices within one partition have the same values across all dimensions of A', except A_t. An edge $(u, v) \in E$ belongs to the partition e'_{ij} if $u \in u'_i$ and $v \in v'_j$. Note the partition e'_{ij} is exactly a condense edge in the aggregate network G'', w.r.t. A'', and $e'.weight = \sum_{1 \leq i \leq m} \sum_{1 \leq j \leq n} e'_{ij}.weight$.

Therefore, the aggregate network w.r.t. A' can be derived directly from the aggregate network w.r.t. A'', i.e., the cuboid query A' can be answered by A''. $\square$

Proof of Theorem 2

As $dim(S) \subseteq dim(ncd(S, T))$, there exists at least one dimension A_s $(1 \leq s \leq n)$, s.t., $A_s = *$ for S but $A_s \neq *$ for $ncd(S, T)$. Similarly, there exists at least one dimension A_t $(1 \leq t \leq n)$, s.t., $A_t = *$ for T but $A_t \neq *$ for $ncd(S, T)$. Note $s \neq t$, otherwise $S = T$.

Let $G' = (V'_S \bigcup V'_T, E', W_{V'}, W_{E'})$ be the aggregate network for the crossboid $S \bowtie T$. For each condensed vertex $u' \in V'_S$ w.r.t. S, $u' = \{v | S_i(u) = S_i(v), u, v \in V, i = 1 \dots n\}$. We partition the vertices within u' into k disjoint sets, $u'_j = \{v | S_i(u) = S_i(v), u, v \in V, i = 1 \dots n, i \neq s\}$, $1 \leq j \leq k$, i.e., the vertices are partitioned by differentiating their attribute values along A_s. Every u'_j is actually a condensed vertex in the aggregate network $G_{ncd(S,T)}$ w.r.t. $ncd(S, T)$, and $u'.weight = \sum_{1 \leq j \leq k} u'_j.weight$. Analogously, for each condensed vertex $v' \in V'_T$, we partition the vertices within v' along the dimension A_t into l disjoint sets v'_j $(1 \leq j \leq l)$, each of which corresponds to a condensed vertex in $G_{ncd(S,T)}$ and $v'.weight = \sum_{1 \leq j \leq l} v'_j.weight$.

For each condensed edge $e'(u', v') \in E'$, $e' = \{(u, v) | u \in u', v \in v', u' \in V'_S, v' \in V'_T, (u, v) \in E\}$. We partition the edges within e' into $m \times n$ sets e'_{ij} $(1 \leq i \leq m$, $1 \leq j \leq n)$, as follows. The condensed vertex u' is partitioned along the dimension A_s into m disjoint sets u'_i, $(1 \leq i \leq m)$, and the condensed vertex v' is partitioned along the dimension A_t into n disjoint sets v'_j $(1 \leq j \leq n)$. The edge $(u, v) \in E$ is put in the partition e'_{ij} if $u \in u'_i$ and $v \in v'_j$. The partition e'_{ij} is exactly a condense edge in $G_{ncd(S,T)}$ w.r.t. $ncd(S, T)$, and $e'.weight = \sum_{1 \leq i \leq m} \sum_{1 \leq j \leq n} e'_{ij}.weight$.

Therefore, the crossboid query $S \bowtie T$ can be directly answered by the cuboid $ncd(S, T)$.