IJCSMS International Journal of Computer Science & Management Studies, Vol. 12, Issue 03, September 2012
ISSN (Online): 2231 –5268
www.ijcsms.com

On-Line Analytical Processing (OLAP) on Networks

Sandeep Kharb[1], Dr. J.S Sohal[2]

[1]**Research Scholar, NIMS, Shobha Nagar, Jaipur**
kharbs@rediffmail.com

[2]**Ludhiana College of Technology, Ludhiana**
lcet02ludhiana@rediffmail.com

Abstract

We propose a framework for efficient OLAP on networks with a focus on the most interesting kind, the topological OLAP (called "T- OLAP"), which incurs topological changes in the underlying networks. Topological OLAP operations generate new networks from the original ones by rolling up a subset of nodes chosen by certain constraint criteria. The key challenge is to efficiently compute measures for the newly generated networks and handle user queries with varied constraints. The effective computational techniques, Topological-Distributiveness is proposed to achieve efficient query processing and cube materialization. We also provide a Topological OLAP query processing framework into which this technique is weaved.

Keywords: *OLAP, DBLP Query, Topological Theorems.*

1. Introduction

OLAP is On-Line Analytical Processing has been a critical and powerful component lying at the core of the data warehouse systems. OLAP analysis helps companies improve their performance by:

• Providing quick response times. Conducting fast, concise analysis lets companies quickly get to the "why" behind business issues so they can address them in a timely manner

• Delivering powerful, built-in time trending analysis that let users spot trends quickly

• Aligning complex data with the business so it is easy to understand enterprise-wide

• Reducing the burden on IT by providing fast and easy self-service access to information

• Delivering a scalable, efficient technology that is quickly refreshed with current data, and economically scales to satisfy the informational needs of many users.

With the increasing popularity of network data, a compelling question is the following: "*Can we perform efficient OLAP analysis on* networks?" A positive answer to this question would offer us the capability of interactive, *multi-dimensional* and *multi-level* analysis over tremendous amount of data with complicated network structure.

2. Techniques and Framework

We propose one constraint-pushing technique based on the unique characteristics of InfoNet OLAP, T-Distributiveness and T-Monotonicity. The framework taps the pow-erful techniques in traditional OLAP on data cube and extends them further into the information network setting. We use a simple motivating example to introduce the two techniques.

DBLP Query Example:-

Given the DBLP author network, suppose the measure θ of interest is the "total number of publications", i.e., for a given node v, denoted as θ (v) its total number of publications. Depending on the level of network to which v belongs, v could represent an individual researcher, a research group, or an institution. A user could then submit queries asking to return "all researchers v such that θ (v) ≥ δ". The measure in the above example is in fact the "Degree Centrality". We use CD (v) to denote this measure, Degree Centrality, for a node v. To formally represent the concept of

networks at different levels, we need a definition of OLAP network hierarchy

Definition 1 [OLAP Network Hierarchy] Given a network G (V, E) and a partition Π of V (G) such that Π G = {V1, V2,…. Vm}, m ≤ |V (G)|. A network G′ is called a higher-level network of G if G′ is obtained by merging each Vi ∈ Π G, 1 ≤ i ≤ m into a higher-level node v′i and the edges accordingly. G is then called a lower-level network of G′ and denoted by G ≼ G′. For each v ∈ V (G), v′ĭ is called the higher-level node of v if v ∈ V ĭ, which is denoted as v ≼V v ĭ

The topological OLAP operations could be asynchronous. A higher-level network can be obtained by merging portions of a lower-level one, leaving the rest unchanged.

2.1 Topological Distributiveness

Suppose we have three levels of networks where nodes represent individuals, research groups and institutions in each network respectively. Instead of individuals, users could query about the institutions with the total number of publications beyond a certain threshold δ. The straightforward way is to construct the network *G″* at the institution level by merging the constituent author nodes for each institution from the original network *G*, and compute the measure by summing up over each. For large institutions, the computation could be costly. Now suppose we have already computed the measure for the network *G′* at the research group level, can we exploit this partial result to improve efficiency? It turns out we can do that in this case due to the distributiveness of this measure function. Basically, the measure value of an institution can be correctly obtained by summing up over the measure values already computed for the research groups. Consider any set of vertices *S* = *{v*1, *v*2,……, *vk}* and a partition Π_S of *S* such that *_S = {S*1,*S*2,….., *Sm}, m ≤ k*. Each *Si* $\in \Pi_S$ is merged to a new vertex v'_i and the whole set *S* is merged to a new vertex *v″* by a topological OLAP roll-up operation. We also overload the notation to denote $\Pi_S = \{v'_1, v'_2, \ldots v'_m\}$. It is easy to verify that where *ES* is the set of edges with both end vertices in *S*.

$$CD(v'') = \left(\sum_{v_i \varepsilon s} C_D\ (v_i)\right) - 2\,|E_S|$$

$$= \sum_{1 \le i \le m}\left(\sum_{v_i \varepsilon s} C_D\ (v_i) - 2|E_{S_i}|\right) - 2\,|E_{\prod_s}|$$

$$= \left(\sum_{v'_i \varepsilon \prod_S} C_D\ (v'_i)\right) - 2\,|E_{\prod_s}|$$

It is clear that, since addition and subtraction are commutative, distributive and associative, the result of computing by definition from the bottom-level network is the same as the result of computing from the intermediate-level one. Figure 4 is an illustration of the computation. *CD* (*v″*) is a total of 4+2+5+3 = 14 from *G″*. We can get this measure directly from the original network *G* by the given formula $\left(\sum_{v_i \varepsilon s} C_D\ (v_i)\right) - 2\,|E_S|$ = (3 + 8 + 3 + 7 + 10 + 11 + 7 + 5 + 6) − 2(2 + 3 + 3 + 1 + 2 + 4 + 1 + 2 + 4 + 1) = 14. We can also use partial measure results computed for the intermediate network *G′* and compute by $\left(\sum_{v'_i \varepsilon \prod_S} C_D\ (v'_i)\right) - 2\,|E_{\prod_s}||$ = (8+12+14) −2 (3+1+6) = 14. The computational cost is reduced to *O* (*m* + $|E_{\Pi s}|$). This example shows that the computation cost is greatly reduced by taking advantage of partial measure results already computed.

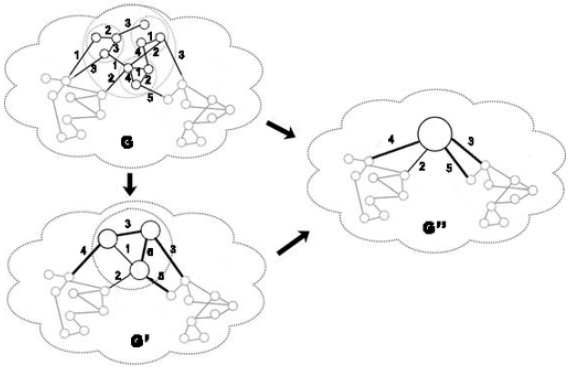

Figure 1 Topological Distributiveness for Degree Centrality

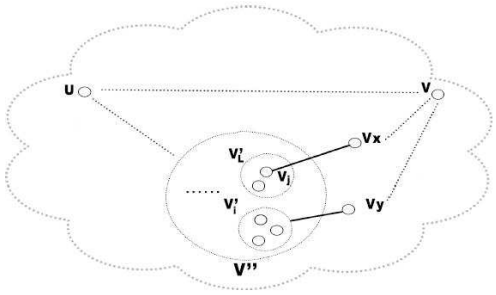

Figure 2 T-Distributiveness for Shortest Path

This kind of distributiveness of a measure function is termed *T-Distributiveness* in this topological OLAP setting.

Definition 2 ***[Topological-Distributiveness]*** *Given a measure θ and three attributed networks G, G′ and G″ obtained by T-OLAP operations such that G □ G′ □ G″, suppose we have available θ (G) and θ (G′), then θ is* Topological Distributive *if there exists a function g such that $\theta(G'') = g(\theta(G')) = g(\theta(G))$.*

Although this example of "Degree Centrality" may seem simple, it is interesting to note that other more complicated measures, even those involving topological structures, are also Topological distributive. For instance, it can be shown that the measure of "Shortest Path" is also T-distributive. Shortest path computation is a key problem underlying many centrality measures, such as *Closeness Centrality* and *Between's Centrality*, as well as important network measures like *Diameter*.

Topological distributiveness for Shortest Paths It is well-known that the shortest path problem has the property of optimal substructures. In fact, shortest-path algorithms typically rely on the property that a shortest path between two vertices contains other shortest paths within it. Formally, we have the following lemma, the proof of which is omitted and readers are referred to.

Lemma 1. *Given an attributed network G with a weight attribute on edges given by function $w : E(G) \rightarrow R$, let $p = \langle v_1, v_2, \ldots, v_k \rangle$ be a shortest path from vertex v_1 to vertex v_k and, for any i and j such that $1 \leq i \leq j \leq k$, let $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ be the sub-path of p from vertex v_i to vertex v_j . Then, p_{ij} is a shortest path from v_i to v_j.*

Rationale: The significance of the optimal substructure property of the shortest path problem is that it means the measure is Topological distributive, thus providing an efficient way to compute the measure for Topological OLAP roll-up operations. We show our algorithm in Algorithm 2. The main algorithm is Algorithm 1 in which we show that, instead of computing from scratch from the lowest network *G*, we are actually able to compute the measure network θ (G″) for G″ from the measure network θ (G′) already computed for an intermediate network G′.

In Algorithm 1, in Line 3, we first compute all shortest paths from the single source v″ to all other vertices. From Lines 4 to 7, we update the shortest path between each pair of vertices (*u*, *v*) by picking the smaller-weight one between the existing shortest path between them and the one which passes through the new vertex v″. In Algorithm 2, in Lines 1 and 2, we first set the shortest path weight between v″ and other vertices to be a maximum weight value. From lines 3 to 6, we calculate the shortest paths between v″ and every other vertex *u* by picking the one with the minimum weight among all the shortest paths between vertices in S′ and *u*. It is easy to verify that the time complexity of computational cost of *Shortest Path Local* is $O(|S'| \cdot |V(G) \setminus S|)$. The time complexity of the entire algorithm is therefore $O(|V(G)|2)$. The correctness of the entire algorithm can be seen from the observation that for any pair of vertices *u* and *v*, if the final shortest path $p_{u,v}$ in G″ does not pass through the new vertex v″, then it should also be the shortest path between *u* and *v* in the lower-level network G′. Therefore, the final shortest path $p_{u,v}$ in G″ must be the smaller-weight one between the existing shortest path between them in G′ and the new shortest path passing through v″. By the optimal substructure property in Lemma 1, the new shortest path passing through v″ must be the union of the two shortest paths, one between *u* and v″, and the other between v″ and *v*. When computing the shortest paths between v″ and other vertices, we do not use standard single source shortest path algorithms. Instead, Algorithm *Shortest Path Local* harnesses the Topological distributiveness of the shortest path measure.

Theorem 1 *Given an attributed networkG with edge weights, G″ is obtained by mering a set of vertices S ={v_1,v_2,…….v_k},S ⊆ V (G) in a Topological OLAP roll-up operation to a new vertex v″, and G′ is obtained by partitioning S by Π = {S_1, S_2,…. S_k} and merging the vertices in each Si into $v'_i \in S'$; $1 \leq i \leq k$, then given the shortest path measure network θ(G′), Shortest Path Local computes the shortest paths between v″ and all vertices in V (G) \ S.*

Algorithm 1 Shortest Path Main

Input: S′, G and θ (G′)
Output: θ (G″)

1: $\theta(G'') \leftarrow \theta(G')$
2: Merge S′ into v″ and add v″ to G″;
3: $\theta(G'') \leftarrow$ *ShortestP ath Local*(S′; G; θ (G″));
4: for each $u \in V(G'')$; $u \neq v''$
5: for each $v \in V(G'')$; $v \in \neq v''$
6: if $w(p_{uv}) > w(p_{uv'}) + w(p_{v'v})$
7: $w(p_{uv}) \leftarrow w(p_{uv'}) + w(p_{v'v})$
8: return θ (G″);

Algorithm 2 Shortest Path Local

Input: S', G and $\theta(G'')$
Output: $\theta(G'')$

1: for each $u \varepsilon V(G) \setminus S'$
2: $w(p_{v''u}) \leftarrow +\infty$;
3: for each $u \varepsilon V(G) \setminus S'$
4: for each $v \varepsilon S'$
5: if $w(p_{vu}) < w(p_{v''u})$
6: $w(p_{v''u}) \leftarrow w(p_{vu})$;
7:return $\theta(G'')$;

3 Experimental Results

3.1 Synthetic Data

All the experiments are conducted on a Pentium(R) 3GHz with 1G RAM running Windows XP professional Service Pack2.

Topological Distributiveness We perform experiments for two measures, *Degree Centrality* and *Closeness Centrality* on synthetic data to demonstrate the power of Topological distributiveness. Since our aim is to provide studies on measures for InfoNet OLAP in general, our synthetic data networks are not confined to specific types and statistical properties. Our synthetic data networks are generated in a random fashion such that (1) the entire network is connected, (2) the vertices have an average degree of d and (3) the edges have an average weight of w.

Given a network G, users can choose a subset S of vertices to roll-up into a single vertex v' and compute the measure network for the new network G'. Such an OLAP operation is called a user OLAP request. We give a model for incoming user OLAP requests as follows. For a network G, we recursively partition G into π connected non-overlapping components of equal number of vertices, until each resulting component is of a predefined minimum number of vertices, i.e., suppose $|V(G)| = 1024$ and $\pi = 4$, we first partition G into 4 connected sub-graphs each with 256 vertices, and recursively partition the 4 sub-graphs. The partition process identifies a sequence T of connected sub-graphs of the original network G. Now we reverse the sequence T and let the resulting sequence be T'. Consequently, observe that, for any sub-graph Q in sequence T', all the sub-graphs of Q appear before Q. We model the sequence of incoming user OLAP requests as the sub-graph sequence T', i.e., the i-th user OLAP request would take the original network G and choose to merge the i-th subgraph in T' into a single vertex and thus obtain a higher-level network G'. The task then is to compute the measure network $\theta(G')$ for G'.

Our baseline algorithm for comparison is denoted as NaiveOLAP. For each user OLAP request, the naive algorithm would first merge the corresponding subgraph into a single vertex and then compute the measure network for the new graph directly from the original network$G$. Our approach, called Topological distributive OLAP, would take advantage of the Topological distributiveness of the measure and take the measures already computed for π lower level networks as input to compute the new measure network. In other words, if put in traditional OLAP terminology, we are considering the best scenario here in which, when computing the measure for a cuboids, all the cuboids immediately below have already been materialized.

Degree Centrality The measure of *Degree Centrality* has the nice property of Topological distributiveness. TD-OLAP could therefore make use of the measures computed for the lower-level networks and gain significant efficiency boost than the NaiveOLAP.

The average vertex degree is set to $d = 5$. The partition size π is set as 4 such that each high level vertex has 4 lower-level children vertices.

Figure 3 shows the running time comparison for the two approaches as the number of vertices for the original network increases. In this case, the original network G is recursively partitioned for a recursion depth of two with a partition size of 4. The running time is the result of summing up the computation cost for all the user OLAP requests in T'. It can be observed that with Topological distributiveness the measure network computation cost increases much slower than the NaiveOLAP approach.

Figure 4 shows that, when the total number of vertices of the network G is fixed to 4096 and the average vertex degree is set to 5, how the granularity of Topological OLAP operations can affect the running time of both approaches. As the number of partitions increases, the size of the set of vertices to be merged in the T-OLAP roll-up get smaller, which means the user, is examining the network with a finer granularity. Since the measure of degree centrality has a small computational cost, both approaches have in this case rather slow increase in the running time. However, notice that the TD-OLAP still features a flatter growth curve compared with the NaiveOLAP approach.

Closeness Centrality The measure of *Closeness Centrality* has the nice property of Topological distributiveness. As such, TD-OLAP would use the

algorithms as shown in Algorithm 1 to assemble the measures computed for the lower-level networks and save tremendous computational cost than the NaiveOLAP which simply merge subsets of vertices and run costly shortest path algorithm to compute the new measure network from scratch. In this example, the average degree is set to $d = 5$ and the average weight on edges is set as $w = 10$. The partition size π is set as 4 such that each high level vertex has 4 lower-level children vertices.

Figure 5 shows the running time comparison for the two approaches as the number of vertices for the original network increases. In this case, the original network G is recursively partitioned for a recursion depth of two with a partition size of 4. The running time is the result of summing up the computation cost for all the 20 user OLAP requests in T'. It is clear that, by harnessing Topological distributiveness, the measure networks can be computed much more efficiently, almost in time linear to the size of the original data network, than the naive OLAP approach.

Figure 6 shows how the granularity of the Topological OLAP roll-up can impact the running time for both approaches. As the number of partitions increases, the original network is partitioned into components of increasingly smaller sizes. The figure shows the average cost for computing the new measure network for one OLAP request as users choose to merge smaller set of vertices in the T-OLAP operations. The network in this case contains 1024 vertices. As shown in the figure, for TD-OLAP, the granularity hardly affects the computational cost since the

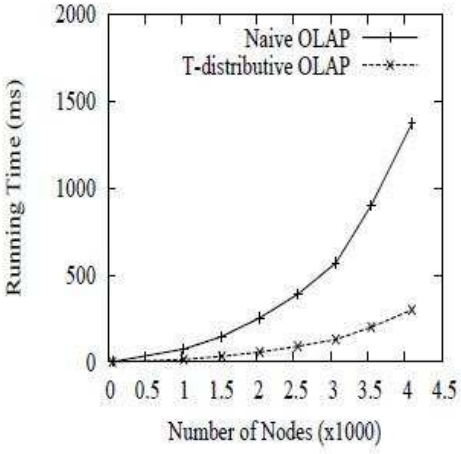

Figure 3 Run Time Comparison

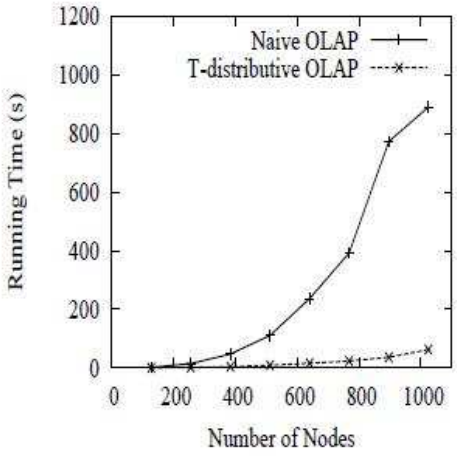

Figure 4 Topological-OLAP Granularities

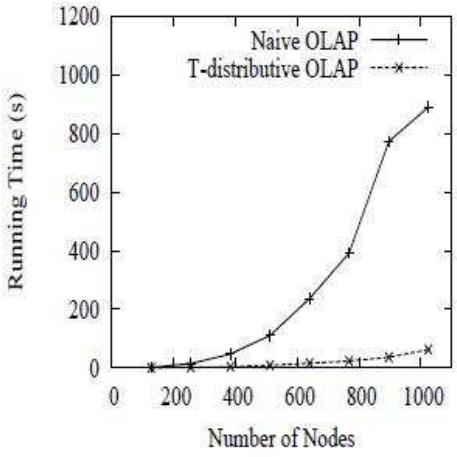

Figure 5 Run Time Comparison

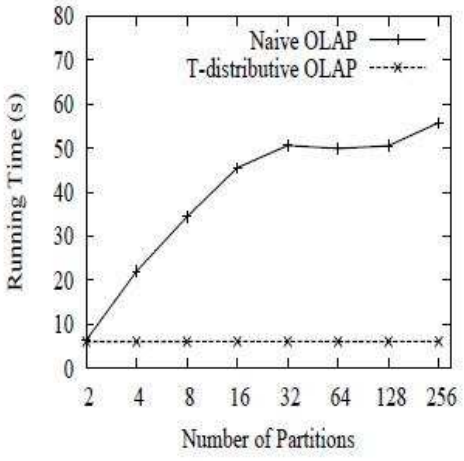

Figure 6 Topological-OLAP Granularity

Complexity of the function to combine the measures of lower-level networks to obtain the new one is in general very low compared with the function to compute the measure itself..As the partition size only affect the number of lower-level vertices to taken into consideration, the running time therefore remains steady. On the other hand, as fewer vertices are merged with increasing number of partitions, the NaiveOLAP has to compute the measure network with an input network of greater size. Hence the increasing running time for the NaiveOLAP

6 Conclusions

In this paper we have performed a framework study for topological network OLAP. In particular, we propose a technique in a constraint-pushing framework, Topological Distributiveness, to achieve efficient query processing and cube materialization. We put forward a query processing framework incorporating in this technique. Our experiments on both real and synthetic data networks have shown the effectiveness and efficiency of the application of our techniques and framework to the measures.

References

[1] G. Flake, S. Lawrence, C. L. Giles, and F. Coetzee. Self-organization and identification of web communities. IEEE Computer, 35:66–71, 2002.

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

[3] 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. Disc., 1(1):29–53, 1997.

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

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

[6] D. Jensen and J. Neville. Data mining in networks. In Papers of the Symp. Dynamic Social Network Modeling and Analysis, National Academy Press, 2002.

[7] R. Jin, Y. Xiang, N. Ruan, and H. Wang. Efficiently answering reachability queries on very large directed graphs. In SIGMOD ’08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pages 595–608, New York, NY, USA, 2008.ACM.

[8] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web as a graph: Measurements, models, and methods. In Proc. Int. Conf. Computing and Combinatorics (COCOON’99), pages 1–17, Tokyo, Japan, July 1999.

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

[10] J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is more: Sparse graph mining with compact matrix decomposition. Stat. Anal. Data Min., 1(1):6–22, 2008.

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

[12] N. Wang, S. Parthasarathy, K.-L. Tan, and A. K. H. Tung. CSV: visualizing and mining cohesive subgraphs. In SIGMOD Conference, pages 445–458, 2008.

[13] D. Archambault, T. Munzner, and D. Auber. TopoLayout: Multilevel graph layout by

topological features. IEEE Trans. Vis. Comput. Graph., 13(2):305–317, 2007.

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

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

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

[17] C. Chen, X. Yan, F. Zhu, J. Han, and P. S. Yu. Graph OLAP: Towards online analytical processing on graphs. In Proc. 2008 Int. Conf. Data Mining(ICDM), 2008.

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

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

[20] Efficient Topological OLAP on Information Networks 15

[21] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In Proc. 2005 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'05), pages 177–187, Chicago, IL, Aug. 2005.

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

[23] M. E. J. Newman. The structure and function of complex networks. SIAM Review, 45:167–256, 2003.

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

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