www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 6 Issue 2 Feb. 2017, Page No. 20196-20201
Index Copernicus Value (2015): 58.10, DOI: 10.18535/ijecs/v6i2.05

A Unified Framework for Both Graph and Pattern Matching Using IPaM-WAG

Mrs. R.Hemalatha[1], *Ms.M.Aarthi*[2]
Associate professor & Head, Department of Computer Science,
Tiruppur Kumaran College for Women, Tiruppur, Tamil Nadu.
Research Scholar, Department of Computer Science,
Tiruppur Kumaran College for Women, Tiruppur, Tamil Nadu.

Abstract: In general the matter of finding sub graphs that best match a user's query on weighted Attributed Graphs (wags) is an open research area. There is a tendency to outline a WAG as a graph where every nodes exhibit multiple attributes with varied, non-negative weights. An example of a WAG could be a coauthor ship network, wherever every author has multiple attributes, every such as a specific topic (e.g., databases, data processing, and machine learning), and therefore the quantity of experience in a very specific topic is delineate by a non-negative weight on it attribute. A typical user query during this setting specifies each property patterns between query nodes and constraints on attribute weights of the query nodes. A ranking perform that unifies the matching on attribute weights over the nodes and on the graph structure is proposed. To prove that the matter of retrieving the best match for such queries is complete. Moreover, there is a tendency to propose a quick and effective top-k pattern matching algorithm and top-k graph search algorithm for weighted attributed graphs. In an intensive experimental study with multiple real-world datasets, the projected algorithm exhibits important speed-up over competitive approaches. On average, projected technique is quicker in query process than the strongest competitive technique.

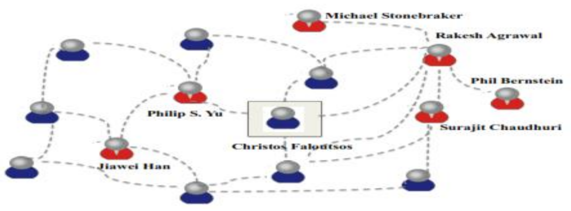
Keywords: Weighted Attribute Graph, Graph Search, Top-K Algorithms, Index based Pattern Matching (IPaM)

1. INTRODUCTION

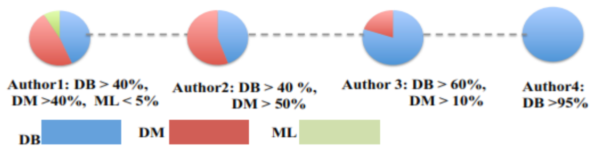
Graphs offer a natural method for representing entities that are "connected"—e.g., individuals connected by their co-authorship relationships. A graph is commonly denoted by G = (V, E), where every V is a set of nodes or vertices and E is a set of links or edges.

Given a graph G, a long-standing downside of interest has been to search out patterns that match user queries—e.g., notice all triangles in an exceedingly graph. An extension of the first pattern-matching downside is to seem at graphs wherever nodes have one attribute [1, 2]. For instance, given a co-authorship graph wherever every node contains the first experience of an author, notice matches to a triangle question wherever the three nodes (in the triangle) square measure specialists in databases, data processing, and machine learning, severally. To have an interest within the additional general downside of pattern-matching on graphs wherever (1) nodes have multiple attributes (for example. An author having multiple expertise) and (2) the node attributes have non-negative weights related to them (e.g., an author has varied degrees of experience across totally different topics).

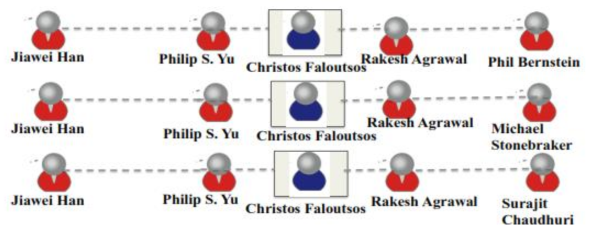
IPaM-WAG will simply be extended to alternative cases, e.g., per-attribute standardization wherever the total of the weights for every attribute is one across all nodes. Figure one depicts an example WAG and pattern-query. A good style of world applications are naturally modeled as wags, multiple experience with varied degrees specified from co-authorship networks with author-nodes, to IP communication networks with IP-nodes having multiple functionalities (such as DNS server, Web server, and P2P client) with varied degrees. Vary queries are versatile, where a weighted-attribute will be in an exceedingly vary of values (see top of Figure 1b).



(a) A Part Made DBLP Co-Authorship Network



WAG Query



(b) A Question Over The DBLP Co-Authorship Network And Top-3 Results Came Back By IPam

Figure 1: Example: The DBLP Co-Authorship Network May Be A WAG. Every Node Is An Author With Three Attributes: Expertise In Databases (DB), Data Mining (DM), And Machine Learning (ML).

A pattern question on the sound until co-authorship network might need to find a path of four authors such two of them have virtually equal experience in databases (DB) and data processing (DM) with a lot of less experience in machine learning (ML); and therefore the remaining authors square measure primarily DB specialists. Such a pattern question with path structure and vary constraints on attribute-weights is represented on the top-half of Figure 1(b). The bottom half Figure 1(b) describes the 3-best answers came back by IPaM. Note that for all three answers, author Christos Faloutsos acts as "bridge" connecting authors whose primary experience is in sound unit to authors whose experience is additional uniformly unfold between sound unit and DM.

To summarize, unified framework for both graph and pattern matching measures are as follows:

- To introduce the notion of wags, and investigate the matter of pattern matching on wags.
- To prove that finding the optimum match for a given pattern question on a WAG is NP-complete.
- To propose a versatile querying mechanism on wags and a completely unique ranking technique that unifies each structural pattern and attribute weights into one live.
- To propose IPaM-WAG for graph search and pattern matching on wags. IPaM-WAG encompasses a novel hybrid indexing scheme that unifies attribute weights and structural properties of a WAG. IPaM-WAG uses a completely unique rule to expeditiously come back the simplest solutions.
- To demonstrate IPaM's effectiveness through intensive experiments on real-world data and comparisons with the closest competitor.

2. RELATED WORK

Tong et al. [1] proposed a pattern matching downside over an oversized directed graph (based on reachability constraints) and a corresponding approximate matching algorithm to pick out the simplest set of patterns once an explicit match isn't potential, considering one attribute per node. Afterward, Zou et al. [3] extend the drawback by planning constraints on distance rather than reach ability and style algorithms for that problem.

The drawback is tangentially totally different owing to the existence of weights over vertex attributes. Additionally pattern matching, is made downside tangentially is totally different owing to the existence of weights over node attributes. A comprehensive survey describing variations among totally different graph matching issues, general and specific resolution approaches, and analysis techniques is found in [4]. The information literature has extensively studied the matter of keyword search over relative databases [5].

Graph Indexing: Given a graph database, graph indexing literature aims at indexing the graphs supported the frequent substructures gift in them [6]. To it finish, graphgrep [7] could be a renowned representative of path-based indexing approach. Graphgrep enumerates all existing ways up to an exact length l in an exceedingly graph database G and selects them as indexing options. Compared to the path-based indexing approach, in [8] uses graphs as basic indexing options that is commonly remarked as graph-based indexing approach.

Inference Queries in Graphical Models: Efficiently evaluating illation queries [10] has been a serious analysis space within the probabilistic reasoning community. Observe that, downside is totally different in nature. Not like the previous body of labor, the existence of a footing in weighted attributed graph isn't probabilistic. Moreover, not like this shaped body of labor, the nodes of a WAG contain multiple weighted attributes. Search in graphs, notably social networks, is a vigorous space of analysis (e.g., [11], [12]), since vertices with weighted attributes arise naturally. Aim to know graphs with vertex attributes [13], [14]; or work that leverages such graphs for a few explicit purposes [15]. Of explicit connation here square measure recommendations supported social network graphs [16].

3. IPaM-RANKING

Given a query (point or range), the task is to come its top-k best matches. To do so, IPaM has to rank the results considering divergence on the graph structure and on the weighted attributes. The selection of divergence measures is orthogonal. Cha provides a comprehensive survey on varied similarity and distance measures. Whereas divergence on graph weighted attributes is basically different from the divergence on the graph structure, projected ranking function unifies these into one single function. Suppose G_s Is a candidate subgraph in response to the pattern query H_q , then IPaM-WAG's ranking for G_s With respect to H_q Is defined as follows:

$$F(G_s, H_q) = \sum_{i=1}^{|V_s|} D(v_i, v_i^{'})$$

Where $v_i \in G_s$ And $v_i^{'} \in H_q$. Then, the divergence function $D(v_i, v_i^{'})$ Is defined as follows:

$$D(v_i, v_i^{'}) = \begin{cases} 1 & if\ v_i\ is\ an\ unmatched\ node \\ Kullback - Leibler(v_i, v_i^{'}) & otherwise \end{cases}$$

Kullback-Leibler difference is normalized so that a 0 value means a perfect match and 1 value means a perfect non-match. The Kullback-Leibler difference for divergence function gives a good information-theoretic formulation. As mentioned before, IPaM-WAG can use any divergence function. The definition of the Kullback-Leibler difference is as follows:

$$F_w(v_i, v_i^{'}) = \sum_{m=l}^{l} \left[\frac{w_{im} \times \mathrm{Ln}(w_{im}) + w_{i'm} \times ln(w_{i'm})}{2} - \frac{w_{im} + w_{i'm}}{2} \times \mathrm{Ln}\left(\frac{w_{im} + w_{i'm}}{2}\right) \right]$$

Where w_{im} Denotes the weight of i-th node for the m-th attribute; and l is the number of weighted attributes. An unmatched node as a extra node is needed to add to the solution for satisfying the pattern query's connectivity requirements. Such a node will have maximal divergence (i.e., 1).

3.2. Hybrid Indexing and Matching With IPaM-WAG

In this section, to enable pattern matching on wags, a novel hybrid index structure, called IPaM-WAG, is proposed and.how it lends itself to efficient pattern matching and for graph searching is demonstrated. Hybrid indexing and matching depicts how IPaM-WAG works overall, and also it highlights the key components and processes.

The indexing technique is presented next in more detail, while defining the matching algorithm (IPaM-Match). Given a WAG, IPaM-WAG builds and maintains an index structure offline, which is used to speed-up pattern matching and graph search during query time and facilitates the matching over both the weighted attributes and the structure of the WAG. First, a balanced tree index is constructed for the matrix T. Consider that T consists of the W weight matrix concatenated with the node-degree vector (i.e. [W, degrees]). IPaM-WAG has the following properties:

- The root of IPaM-WAG is between 1 and M entries (unless it is a leaf node). M is the degree of the tree. All intermediate nodes have between M/2 and M entries. Each leaf node is between M and 2M entries.
- A leaf node in an IPaM-WAG is a WAG vertex with attribute weights R.
- Each intermediate node is a minimum bounding rectangle (MBR) of dimensionality l +1(recall l is the number of weighted attributes). An intermediate node with j entries has j children, and the bounding rectangle indexes the space of its children.

Next, at each leaf node (where a leaf node could be a vertex in G), an inverted-list index of immediate neighbors of every graph vertex is maintained. This index allows economical structural match. The IPaM-WAG on T to be a balanced tree is devised. This tree is constructed by remodeling every row in T to a multi-dimensional purpose, where the amount of dimensions corresponds to the amount of attributes and a dimension for categorization the degree. Later, this house is indexed. Whereas the core plan is borrowed from the Rtree family, that facilitates multi-dimensional spatial looking out and therefore the integration into object-relational management systems.

This includes: (1) indexing over T that features structural and weighted attribute properties; (2) integrating this tree with an inverted-list index to change quick structural match; and (3) planning a getnext() interface over T , that returns consecutive best candidate vertex of G that has the smallest amount divergence with reference to a WAG query node. Every leaf node of the IPaM-WAG corresponds to a vertex within the input WAG. Every leaf of the IPaM-WAG contains AN inverted-list that represents the set of nodes that are its immediate neighbors.

Getnext(): Given any pattern query $H_q = (V', E')$, for each vertex $v_i' \in V'$,agetnext() call is issued to the IPaM-WAG to return the node in G that has the closest divergence with v (i.e., nearest neighbor to v_i'). The inverted-list of the IPaM-WAG is subsequently used for the purpose of structural match, after the candidate node for each query node is returned.

Range Query: Given a range query vertex v_i', its weight distributions over the specified attributes and degree are used to transform v_i' To a query rectangle in the l+1-dimensional attribute-space. As an example, a query node with range specification such as <.2 on DB and >.5 on DM, and degree 3 can be translated as ranges [0 -.2], [.5-1.0], and [3 -maxdegree] respectively, and a query rectangle can be formed. For the attributes whose weights are not specified in the query explicitly, their respective ranges are considered as [0 - 1]. To enable searching, IPaM-Match starts from the root of the IPaM-WAG tree and traverses down the tree. If a current node is non-leaf and overlaps with the query rectangle, it continues to search further down in that subtree. If the current node is a leaf, and the leaf is contained in the query rectangle that leaf is returned as an answer.

Point Query: Given a point query node v', it is first transformed to a point in l +1-dimensional space. Like range query, searching for the best matching node of v 0 begins at the root of the IPaM-WAG and traverses down. It tries to prune some of the intermediate branches, and keeps a list of active branches to be expanded further. The algorithm terminates once the active branch list is empty. Pruning: using the bounding rectangles of the IPaM-WAG to decide whether or not to search inside the subtree that it indexes. These rectangles can be searched efficiently using MINDIST and MINMAXDIST. MINDIST is the optimistic distance between the point and any object indexed by MBR. Specifically, if v is inside the MBR, MINDIST(MBR,v 0 i)=0. Otherwise, MINDIST(MBR, v_i') is the minimal possible divergence from the query point v 0 i to any node inside or on the perimeter of the rectangle.

4. GRAPH SEACHING AND PATTERN MATCHING USING IPaM-WAG

In this section, an efficient, optimal algorithm (referred to as IPaM in the experiments) for top-k pattern matching and graph search problem, building upon IPaM-WAG is discussed. "Optimality" here is with respect to answer quality, and not necessarily with respect to query processing time. A naive algorithm will enumerate over all possible candidate results before it determines the final top-k results. This naive algorithm is prohibitive for even moderately large graphs.

Given any pattern and graph query, the algorithm executes the following tasks:

(1) It uses the getnext() interface of IPaM-WAG and retrieves the next best candidate node and vertex for every query node and vertex.

(2) It tries to establish all query edges by joining vertices of the WAG that represent the endpoints of a query edge. The inverted-list of the IPaM-WAG is used for this purpose.

(3) When k answers are not fully computed, the algorithm also judiciously determines whether to expand structurally (i.e., introduce additional nodes to connect the candidate nodes that represent the endpoints of a query edge), or to issue getnext() calls to retrieve the next best candidate node from IPaM-WAG.

(4) It returns the top-k matches ranked by the increasing order of the overall divergence score.

In order to perform this last step efficiently, the algorithm maintains a threshold value that captures the minimum divergence that an unseen or partially computed candidate-answer may have. The algorithm achieves early-termination when the threshold is not smaller than the score of the k-th best result thus far. In order to accomplish tasks (3) and (4) described above, IPaM-WAG exploits the overall divergence score as a threshold and treats it as monotonic. Keen readers may observe that the high-level intuition of leveraging the threshold bears resemblance to the top-k family of algorithms in [6]. However, proposed technique requires non-trivial extensions to make such schemes applicable to graphs, and considers divergence measures in the place of scoring functions.

Range Query: Algorithm 1 describes the pseudocode of IPaM.

Algorithm 1: IPaM -WAG Optimal Algorithm for Top-k range query

```
Require: IPaM-WAG, Query $H_q = (V', E')$, k
1: Issue getnext() in round robin fashion to get the next best matching node and candidate vertex for each query node $v_i'$ Or query vertex $v_i'$
2: Form candidate edges and add to candidateedgeset C
3: Update resultset with top-k answers based on divergence
4: Compute threshold
5: while the C is not empty do
6: if (threshold<resultset.k-th Score) then
7: Issue getnext() in round robin fashion
8: Update resultset with top-k candidate outputs
9: else
10: Output resultset as the best k-results
11: break
12: end if
13: end while
14: return resultset
```

Given the query nodes, the task is to step by step retrieve the candidate vertices during a round-robin manner by provision getnext() calls to the index, and "stitch" the nodes along to recover the graph structure within the query. The concept of candidate edge to it end is defined as: a candidate edge could be a path (or merely an edge) that could be a representative of a query edge in the WAG. A candidate edge corresponds to a candidate vertex at a minimum of one end point. The candidate edge is complete if each endpoint corresponds to candidate nodes, as an alternative the candidate edge is partially complete. An entire candidate edge doesn't have to be compelled to be swollen any, whereas, the partially complete ones might required to be swollen. For each candidate edge (complete or partially complete), the algorithmic rule keeps track of its current structural divergence. Note that for the vary queries; all candidate nodes that satisfy the specified vary constraints of the corresponding query node are equally fascinating.

Therefore, the divergence score primarily counts the extra nodes (corresponding to writer distinction of 1) that are to connect the candidate vertices so as to match the query structure for the aim of ranking. At a given purpose throughout query process, once the k-results don't seem to be totally computed, IPaM (Algorithm 1) problems further calls (in a spherical robin manner) to the index to retrieve successive best candidate node which will be a kind of query pattern. If the IPaM-WAG now returns a replacement candidate node with the getnext() decision, the algorithm makes an attempt to expand the retrieved candidate nodes structurally, one by one, till k-results are computed.

This step is valid by considering the worth of the edge. The worth of the edge is that the minimum ranking (divergence) score that any unseen candidate answer might have. So as to cypher the edge efficiently, for each query edge e', the algorithm keeps track of further quantities considering its candidate edges: (1) the tiniest divergence score considering its candidate edges, and (2) the most recent divergence score of its last partly complete candidate edge. Given the query H_q With |E'| edges, the edge at that step is computed by aggregating the various divergences of the query edges that cause the tiniest total. The algorithm terminates once the divergence score of the k-th best result doesn't exceed the edge. Note that, as presently because the algorithmic rule retrieves a replacement node from the IPaM-WAG or expands the partly computed answers structurally by a footing, it updates the edge price.

Point Query: The optimum algorithm for purpose query matching is comparable in essence to it for varies queries. The most variations here are that (1) ranking of some extent question needs stricter matches on the weighted attributes; and (2) throughout query process, if topk answers don't seem to be totally computed, then the algorithm must judiciously decide whether or not to issue another getnext() decision, or to expand structurally. This call is taken by analyzing the various threshold values and selecting the one those leads to a smaller threshold. Recall that with each new getnext() decision, or structural enlargement, have a tendency to update the edge, and note that a smaller threshold is healthier, since it implies that the unseen results can have smaller divergence.

5. EXPERIMENTAL EVALUATION

The efficiency of IPaM-WAG against WAG-ray is reported. First, report average query interval (Avg QPT) by variable result size (i.e., k). Next, report Avg QPT by variable the quantity of nodes within the query. (Note that since the queries have specific patterns, the quantity of edges is additionally identified consequently.) For these runtime experiments, differing kinds of pattern queries is used.The query work consists of eighty queries (20 queries of a specific pattern), and that report the common query interval. For brevity, report the results on one little graph (DBLPDB), and 2 massive graphs (Amazon and Yahoo!). The omitted results on the tiny graphs are kind of like those delineate.

Query-processing time as a function of result-size: Compare query-processing time of IPaM-WAG and WAG-ray by variable k (result size). IPaM-WAG outperforms WAG-ray each for little and enormous graphs. The results are listed in Figure 2. Clearly, IPaM-WAG significantly outperforms WAG-ray altogether the cases; Among the three Recall low sparsity signifies little no of 0's within the W matrix. Three massive graphs, the distinction in query interval is additional significant in Amazon than in Yahoo!. The terribly low sparsity of the weighted attribute matrix (W) of Amazon graph makes the computation favorable to IPaM-WAG, whereas WAG-ray was primarily designed for graphs that contain one attribute per node (i.e., terribly high sparsity of the weighted attribute matrix). Naturally, the efficiency of IPaM-WAG is determined to be most for Amazon graph.

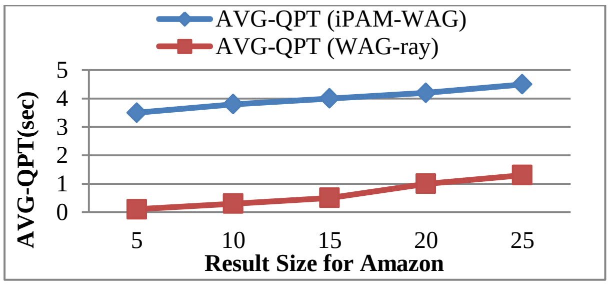


Figure 2: Comparison Of Query Processing Time Between Ipam-WAG And WAG-Ray When Varying Result-Size K

Query-processing time as a function of range of nodes: Compare query-processing time of IPaM-WAG and WAG-ray by variable range of nodes within the query. IPaM-WAG outperforms WAG-ray each for little and enormous graphs. The results are listed in Figure 3. Each IPaM-WAG and WAG-ray scale well with the increasing range of nodes. Amazon exhibits the simplest performance with increasing range of nodes. This observation is because of the terribly low poorness price of the weighted attribute matrix (W) of Amazon that helps it scale very well with the increasing range of nodes within the query.

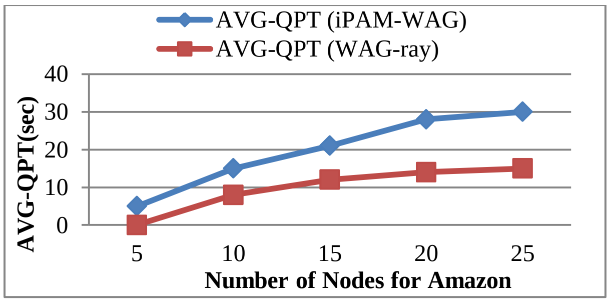


Figure 3: Comparison Of Query Processing Time Between Ipam-WAG And WAG-Ray When Varying Number Of Nodes

Query-processing time as a function of weighted-attribute matrix sparsity: In these experiments, the sparsity of the matrix W is varied, and report the common query interval of IPaM-WAG and WAG-ray. For every node, cypher the entropy of its weight distribution, and type supported their entropy. That is, the nodes at the highest are those whose distributions are nighest to uniform. To satisfy bound poorness share larger than its original poorness, scan this sorted list from very cheap, and rework the membership of a node to one attributes (attribute that has the most important weight gets one, rest gets 0) till the required poorness is satisfied. K is ready to five, and range of nodes within the question is ready of five.

Figure 4 depicts results for this experiment. Observe that the query interval will increase with increasing poorness for IPaM-WAG, however decreases for WAG-ray, and also the impact is additional in massive graph compared to little graph. Note that with increasing poorness, WAG-ray for WAG becomes the quality G-ray [1] that completely impacts its query interval. Conversely, IPaM- WAG takes larger time with increasing poorness (more zeros in W), since it needs generating additional candidate nodes and playacting structural match over them.

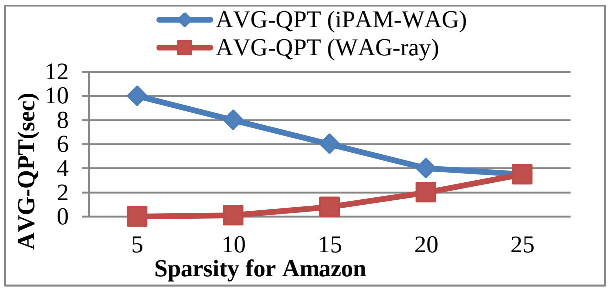


Figure 4: Comparison Of Query Processing Time Between Ipam-WAG And WAG-Ray When Varying Sparsity.

6. CONCLUSION

In this research described weighted Attribute Graphs (wags), which may model a large vary of information arising in various applications. Investigate the matter of pattern matching on wags. Although, prove, finding the optimum match for a given pattern query on a WAG is NP-complete, introduce IPaM-WAG that may perform efficient and effective pattern matching on wags. IPaM-WAG includes a novel hybrid assortment theme that comes with each the weighted attributes and also the graph structure. IPaM-WAG uses a unique algorithm to efficiently come back the simplest answers to a pattern query. Demonstrate the effectiveness and measurability of IPaM-WAG supported intensive experiments on real-world knowledge, exhibiting higher question response times. Future work includes the extension of IPaM-WAG for time-evolving graphs.

REFERENCES

1. H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad, "Fast besteffort pattern matching in large attributed graphs," in Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2007, pp. 737–746.

2. L. Zhu, W. K. Ng, and J. Cheng. Structure and attribute index for approximate graph matching in large graphs. Inf. Syst., 36(6):958–972, 2011

3. L. Zou, L. Chen, and M. T.Ozsu, "Distance-join: Pattern match query in a large graph database," Proc. VLDB Endowment, vol. 2, no. 1, pp. 886–897, 2009.

DOI: 10.18535/ijecs/v6i2.05

4. B. Gallagher. Matching structure and semantics: A survey on graph-based pattern matching. In AAAI Fall Symposia, pages 45–53, 2006.

5. V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in relational databases. In VLDB, pages 670–681, 2002.

6. Y. Xie and P. S. Yu. In CP-Index: On The Efficient Indexing of Large Graphs, 2011.

7. D. Shasha, J. T. L. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In Symposium on Principles of Database Systems, pages 39–52, 2002.

8. X. Yan, P. S. Yu, and J. Han. Graph indexing: A frequent structure-based approach. In SIGMOD, pages 335–346, 2004.

9. P. Zhao, J. X. Yu, and P. S. Yu. Graph indexing: Tree + delta >= graph. In VLDB, pages 938–949, 2007.

10. R. G. Cowell, A. P. David, S. L. Lauritzen, and D. J. Spiegelhalter. Probabilistic networks and expert systems. ACM Trans. On Speech and Language Processing, 1999.

11. J. Tang, S. Wu, B. Gao, and Y. Wan, “Topic-level social network search,” in Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2011, pp. 769–772.

12. J. Tang, J. Zhang, R. Jin, Z. Yang, K. Cai, L. Zhang, and Z. Su, “Topic level expertise search over heterogeneous networks,” Mach. Learn. J., vol. 82, no. 2, pp. 211–237, 2011.

13. D. Quercia, L. Capra, and J. Crowcroft, “The social world of Twitter: Topics, geography, and emotions,” in Proc. Int. Conf. Weblogs Social Media, 2012, pp. 298–305.

14. D. Quercia, R. Lambiotte, D. Stillwell, M. Kosinski, and J. Crowcroft, “The personality of popular Facebook users,” in Proc.ACM Conf. Comput. Supported Cooperative Work, 2012, pp. 955–964.

15. S. Ghosh, N. K. Sharma, F. Benevenuto, N. Ganguly, and P. K. Gummadi, “Cognos: Crowdsourcing search for topic experts in microblogs,” in Proc. 35th Int. ACM SIGIR Conf. Res. Develop. Inf.Retrieval, 2012, pp. 575–590.

16. S. Wu, J. Sun, and J. Tang, “Patent partner recommendation in enterprise social networks,” in Proc. 6th ACM Int. Conf. Web Search Data Mining, 2013, pp. 43–52.

17. Y. Sun and J. Han, Mining Heterogeneous Information Networks: Principles and Methodologies. San Rafael, CA, USA: Morgan & Claypool, 2012.