Tools and Algorithms for
Querying and Mining Large Graphs
Thesis Proposal

Hanghang Tong
Carnegie Mellon University
htong@cs.cmu.edu

Thesis Committee
Christos Faloutsos, CMU
William Cohen, CMU
Jeff Schneider, CMU
Philip S. Yu, UIC

1 Abstract

Graphs appear in a wide range of settings, such as computer networks, the world wide web, biological networks, social networks (MSN/FaceBook/LinkedIn) and many more. How can we find user-specific patterns (e.g., master mind, money laundry ring) from such graphs? How can we spot anomaly in a dynamic and intuitive way? How can we find the communities with optional constraints? How can we mine time/space in the complex context? In this thesis, we focus on two types of tasks according to the interaction with users: (1) querying and (2) mining.

For the task of querying, we have mainly studied the following three sub tasks. First, we focus on how to find complex user-specific patterns from large graphs, where we have addressed three applications: (1) Center-piece subgraph discovery for plain graphs; (2) Best effort pattern match for attributed graphs; and (3) Querying with feedback. Then, we focus on (1) how to predict the direction of the link; and (2) how to address the temporal issue in querying. Finally, since the main tool for querying large graphs is the proximity measurement, we have designed a family of fast solutions (FastProx) in several different settings, which often achieves up to 2 orders of magnitude speedup without or with very little quality loss.

For the task of mining, we have studied the following two sub tasks. First, we have proposed a family of example-based low-rank approximation methods (Colibri) for anomaly detection. It can work for both static graphs and dynamic graphs. It achieves significant speedups and space saving over the existing methods without quality loss. Then, we have designed (T3) to mine temporal information in the context of graphs, which can find similar time stamps as well as abnormal time stamps; and also provide the interpretations for our findings. Furthermore, we have proposed (MT3) to speedup the multiple resolution analysis.

Future work includes two aspects. First, we would like to design tools to detect communities from graphs with optional constraints. Then, we will study how to mine the spatial information in the context of graphs. Besides, we plan to investigate diffusion wavelet as an alternative way for querying and mining large graphs.
Contents

1 Abstract .............................................. 1

2 Introduction ........................................ 3
   2.1 Motivating Questions .................................................. 3
   2.2 Impacts and Applications .............................................. 3
   2.3 Thesis Organization .................................................... 4

3 Survey ................................................. 4
   3.1 Querying on Graphs .................................................... 4
   3.2 Mining on Graphs ...................................................... 5

4 Completed Work on Querying .............................. 6
   4.1 Q1: Find Complex User-Specific Patterns ......................... 6
       4.1.1 (Q1.1) Center-Piece Subgraph discovery ....................... 6
       4.1.2 (Q1.2) Best Effort Pattern Match ............................ 7
       4.1.3 (Q1.3) Querying with Feedback .................................. 8
   4.2 Q2: Link Prediction and Proximity Tracking ...................... 10
       4.2.1 (Q2.1) Link Prediction ........................................ 10
       4.2.2 (Q2.2) Proximity and Centrality Tracking ................... 11
   4.3 Q3: Scalability Issues in Querying ................................ 12

5 Completed Work on Mining ............................... 13
   5.1 M1: Anomaly Detection ............................................. 13
   5.2 M2: Mining Time ..................................................... 15

6 Proposed Work ......................................... 15
   6.1 P1: Community Detection with Optional Constraints ............ 15
   6.2 P2: Mining Space .................................................. 17
   6.3 P3: Diffusion Wavelet ............................................... 17
   6.4 Time Line .......................................................... 17

7 Conclusion ............................................ 17
2 Introduction

Graphs appear in a wide range of settings, such as computer networks, the world wide web, biological networks, social networks (MSN/FaceBook/LinkedIn) and many more. How can we find user-specific patterns (e.g., master mind, money laundry ring) from such graphs? How can we spot anomaly in a dynamic and intuitive way? How can we find the communities with optional constraints? How can we mine time/space in the complex context?

2.1 Motivating Questions

Depending on the interaction with users, we focus on two types of tasks in this thesis: (1) querying; and (2) mining.

For the task of querying, we mainly focus on the following sub tasks:

Q1: How to find complex user-specific patterns? e.g.,
   - (Q1.1) How find master-mind criminal given some suspects X, Y and Z?
   - (Q1.2) How to find a money-laundering ring?
   - (Q1.3) How to deal with user feedback (such as negation) from users?

Q2: Link prediction and proximity tracking? e.g.,
   - (Q2.1) How to predict the direction of citation (phone/email)?
   - (Q2.2) How to track the most influential authors over time?

Q3: How to answer all the above questions quickly in large, disk resident graphs?

For the task of mining, we mainly focus on the following sub tasks:

M1: How to spot the anomaly in an efficient, intuitive and dynamic way?

M2: How to mine time/space in the context of graphs?

M3: How to detect community with optional constraints?

2.2 Impacts and Applications

There are a lot important applications behind the above tasks. To name a few, Q1.1 allows us to (1) find common advisor or somebody who has initiated research field that querying authors belong to in an co-authorship network; (2) spot master-mind criminal in law enforcement; and (3) identify the protein that participates in pathways with all or most of the given queries in gene regulatory networks. Similarly, we can spot money laundry ring by Q1.2. By M1, we can spot abnormal network behavior (e.g., port scanning) in an intuitive and efficient way. By M2, we can find similar time stamps, abnormal time stamp and also the interpretations of our findings. Tab. 1 summarizes the impact and applications for the tasks we address in this thesis.
Table 1: Impact and Applications of Thesis Work

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Impacts and Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1.1</td>
<td>Find common advisor, master-mind criminal and similar genes etc</td>
</tr>
<tr>
<td>Q1.2</td>
<td>Best effort pattern match (e.g., money laundry ring)</td>
</tr>
<tr>
<td>Q1.3</td>
<td>Interactive neighbor search, summarization, etc.</td>
</tr>
<tr>
<td>Q2.1</td>
<td>Predict who calls whom, or who trusts whom, etc.</td>
</tr>
<tr>
<td>Q2.2</td>
<td>Scale sophisticated trend analysis (e.g., GoogleTrend) to time-evolving graphs</td>
</tr>
<tr>
<td>M1</td>
<td>Anomaly detection in an intuitive, efficient and dynamic way</td>
</tr>
<tr>
<td>M2</td>
<td>Mine time/space in complex settings</td>
</tr>
<tr>
<td>M3</td>
<td>Detect community with optional constraints</td>
</tr>
</tbody>
</table>

2.3 Thesis Organization

Tab. 2 gives an overview of the thesis work, mapping our completed work as well as the proposed work to the tasks listed above, together with the types of graphs we address with.

In the following sections, we will first survey the related work, and then briefly describe the completed work, followed up the plan for the future work.

Table 2: Thesis Overview

<table>
<thead>
<tr>
<th>Task</th>
<th>Our Completed Work</th>
<th>Proposed Work</th>
<th>Graph Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>[TF06], [TFGER07], [TQJ08]</td>
<td>NA</td>
<td>Static</td>
</tr>
<tr>
<td>Q2</td>
<td>[TFK07], [TPYF08b], [TPYF08a]</td>
<td>NA</td>
<td>Static, Dynamic</td>
</tr>
<tr>
<td>Q3</td>
<td>[TF06], [TFP06], [TFGER07], [TFK07], [TQJ08], [TPYF08b], [TPYF08a], [TFP08]</td>
<td>P3</td>
<td>Static, Dynamic</td>
</tr>
<tr>
<td>M1</td>
<td>[TPS+08]</td>
<td>P3</td>
<td>Static, Dynamic</td>
</tr>
<tr>
<td>M2</td>
<td>[TSERF08]</td>
<td>P2</td>
<td>Dynamic</td>
</tr>
<tr>
<td>M3</td>
<td>NA</td>
<td>P1,P3</td>
<td>Static, Dynamic</td>
</tr>
</tbody>
</table>

3 Survey

In this Section, we briefly review the related work, which can be categorized into two parts: (1) querying and (2) mining.

3.1 Querying on Graphs

Graph proximity. The main tool for querying is proximity measurement. One of the most popular proximity measurements is random walk with restart [HLZ+04, PYFD04, TFP08], which is the baseline of our work. Other representative proximity measurements include the sink-augmented delivered current [FMT04],
cycle free effective conductance \[KNV06\], survivable network \[GMS93\], and direction-aware proximity \[TFK07\]. All these methods aim to summarize the multiple weighted relationship between nodes by a single number. In term of dealing with the side information on ranking, our work is also related to \[ACA06\], where the goal is to use partial order information to learn the weights of different types of edges. Other remotely related work includes \[GKRT04\], where the goal is to propagate the trust/distrust to predict the trust between any two persons. In terms of dealing with the temporal aspect of querying, our work is also related to incremental proximity search \[SMP08\], Asynchronous PageRank \[APC03, McS05\], and dynamic personalized PageRank \[Cha07, PC08\].

**Applications of graph proximity.** Graph proximity is an important building block in many graph mining settings. In terms of link prediction (Q2.1), our work relates to \[LNK03\], where the goal is to predict the existence of the link. In our FastDAP we take one step further and aim to predict the direction of the link given the existence of the link. For Center-Piece subgraph discovery (Q1.1), Related work includes connection subgraphs \[FMT04, KNV06\], the BANKS system \[ABC+02\], and ObjectRank \[BHP04\]. For best effort pattern match (Q1.2), there are a lot of related work in the literature. Many graph matching techniques focus strictly on matching graph structure and do not utilize attributes. We refer the reader to a recent survey \[Gal06\] on this topic. Best effort pattern match also relates to inexact graph matching \[SH81, TF79, GMV99, WBH+03, CGM04, AMHWS+05\], matching attributed graphs \[TF79, SH81, GMV99, BIJ02, WBH+03, CGM04\], and matching in the single-graph setting \[BIJ02, WBH+03, CGM04, AMHWS+05\]. However, there are relatively few algorithms that combine the three to tackle inexact matching in large, attributed graphs \[CH94, WBH+03, CGM04, AMHWS+05\]. Furthermore, while these algorithms employ various optimizations to mitigate the computational complexity of the problem, they all exhibit super-linear complexity in the worst case. On the other hand, our \(G-Ray\) is linear wrt the number of edges in the graph.

Other representative work includes neighborhood search in bipartite graphs \[SQCF05\], content-based image retrieval \[HLZ+04\], cross-modal correlation discovery \[PYFD04\], RelationalRank \[GMT04\], email management \[MCN06\] and recommendation system \[CTSP07\].

### 3.2 Mining on Graphs

**Low Rank Approximation.** Low rank approximation \[GVL89, DKM05, AM07\] plays a very important role in graph mining. It is related to both M1 and M3. For example, the low rank approximation structure is often a good indicator to identify the community in the graph. A significant deviation from such structure often implies anomalies in the graph.

For static graphs, the most popular choices include SVD/PCA \[GVL89, KAS98\] and random projection \[Ind00\]. However, these methods often ignore the sparseness of many real graphs and therefore often need huge amount of space and processing time (See \[SXZF07\] for a detailed evaluation). More recently, Drineas et al \[DKM05\] proposed the CUR decomposition, which partially deals with the sparsity of the graphs. CUR is proved to achieve an optimal approximation while maintain the sparsity of the matrix. Sun et al \[SXZF07\] further improve CUR by removing the duplicate columns/row in the sampling stage. Their method, named as CMD, is shown to produce the same approximation accuracy, but it often requires much less time and space. Our method (\textit{Colibri-S}) further improves the efficiency in speed and space by leveraging the linear correlation among different sampled columns. As a result, our method saves the computational time and space cost, while it outputs exactly the same low rank approximation as CUR/CMD.

For dynamic graphs, a lot of SVD based techniques have been proposed, such as multiple time series mining \[GGK03, PSF05\], dynamic tensor analysis \[STF06\], incremental spectral clustering \[NXC+07\] etc. As for the static graphs, these methods might suffer from the loss-of-sparsity issue for large sparse graphs.
despite their success in the general cases. Sun et al [SXZF07] deal with this issue by applying their CMD method independently for each time step. However, how to make use of the smoothness between two consecutive time steps to do even more efficient computation is not exploited in [SXZF07]. This is exactly the unique feature of our Colibri-D - it leverages such smoothness to do fast update while maintaining the sparseness of the resulting low rank approximation.

Information-theoretic clustering. Besides low-rank approximation, the most related work for community detection (M3) is the family of compression based methods, which can be traced back to the information-theoretic co-clustering algorithm [DMM03], where the normalized non-negative contingency table is treated as a joint probability distribution between two discrete random variables that take values over the rows and columns. The optimal co-clustering is the one that minimizes the difference in mutual information between the original random variables and the mutual information between the clustered random variables. Followed up work includes the Consistent Bipartite Graph Co-partitioning (CBGC) algorithm [GLZ+07], the spectral relational clustering algorithm [LZWY06], the collective matrix factorization [SG08a, SG08b], the cross association algorithm [CPMF04], autopart [Cha04], GraphScope [SFPY07], etc. M3 also relates to cluster with constraints, such as semi-supervised clustering [WLZ08].

Besides community and anomaly detection, other related work in terms of mining large graph includes pattern and law mining [AJB99, DM02, FFF99, BKM+00, New03], frequent substructure discovery [XHYC05], graph generator [LF07], influence propagation [KKT03], densification laws and shrinking diameters [LKF05], dynamic tensor analysis [STF06], Blog analysis [LMF+07], etc.

4 Completed Work on Querying

In this Section, we summarize our completed work for querying (Q1-Q3). For each sub task, we briefly describe the problem definitions and the main idea of our solutions. We skip the details of our methods for brevity. (Please refer to our published papers listed in Tab. 2 for more details)

4.1 Q1: Find Complex User-Specific Patterns

The goal of Q1 is to find some (complex) user-specific patterns. There are three related sub tasks: (1) Center-Piece subgraph discovery; (2) Best effort pattern matching; and (3) Querying with feedback.

4.1.1 (Q1.1) Center-Piece Subgraph discovery

Here, we want to address Q1.1, where given $Q$ nodes in a social network (say, authorship network), we aim to find the node/author that is the center-piece, and has direct or indirect connections to all, or most of them. For example, this node could be the common advisor, or someone who started the research area that the $Q$ nodes belong to. Isomorphic scenarios appear in law enforcement (find the master-mind criminal, connected to all current suspects), gene regulatory networks (find the protein that participates in pathways with all or most of the given $Q$ proteins), viral marketing and many more.

Formally, the center-piece subgraph problem is defined as follows:

Problem 1 Center-Piece Subgraph Discovery(CePS)

Given: an edge-weighted undirected graph $W$, $Q$ nodes as source queries $Q = \{q_i\} (i = 1, ..., Q)$, the softAND coefficient $k$ and an integer budget $b$
Find: a suitably connected subgraph $\mathcal{H}$ that (a) contains all query nodes $q_i$ (b) at most $b$ other vertices and (c) it maximizes a “closeness” function $g(\mathcal{H})$.

There are two key points in CePS: (1) how to measure the “closeness” for a given node wrt the query set; and (2) how to find a suitably connected subgraph $\mathcal{H}$ quickly. In addition to the problem definition, the main contributions of the proposed method (CePS [TF06]) are (1) we introduce $K_{softAND}$ queries, which includes AND and OR queries as special cases; (2) we propose $EXTRACT$ to quickly extract a subgraph with the appropriate connectivity and maximum “closeness” score.

Fig. 1 gives our solution on a DBLP graph, with $Q=4$ query nodes. All 4 researchers are in data mining, but the first two (Rakesh Agrawal and Jiawei Han) are more on the database side, while Michael Jordan and Vladimir Vapnik are more on the machine learning and statistical side. Figure 1(b) gives our CePS, when we request nodes with strong ties to all four query nodes (i.e., AND query). The results make sense: researchers like Daryl Pregibon, Padhraic Smythe and Heikki Mannila are vital links, because of their cross-disciplinarity and their strong connections with both the above sub-areas. On the other hand, figure 1(a) gives the result for $2_{softAND}$ query, where we want to find nodes with connections to at least 2 of the query nodes. Here the resulting subgraph is quite different from the AND query, which contains two disconnected subgraphs, reflecting two sub-communities in data mining (statistics vs. databases).

Figure 1: Center-piece subgraph among Rakesh Agrawal, Jiawei Han, Michael I. Jordan and Vladimir Vapnik.

4.1.2 (Q1.2) Best Effort Pattern Match

Here, we want to address Q1.2 for large graphs where nodes have attributes, such as a social network where the nodes are labeled with each person’s job title. In such a setting, we want to find subgraphs that match
a user query pattern. For example, a ‘star’ query would be, “find a CEO who has strong interactions with a Manager, a Lawyer, and an Accountant, or another structure as close to that as possible”. Similarly, a ‘loop’ query could help spot a money laundering ring.

Formally, Best-Effort Pattern Pattern can be defined as follows

Problem 2 Best-Effort Pattern Pattern

Given: (i) A (large) graph $G$ whose nodes have one categorical attribute (like ‘job-title’), (ii) a query (small) graph $H_q$ showing the desirable configuration of professionals, and (iii) the number of desired matching subgraphs $n'$.

Find: $n'$ matching subgraphs $H_t$ ($t = 1, \ldots, n'$), that match the query $H_q$ as well as possible, according to a goodness score $g()$.

We propose G-Ray [TFGER07], a fast method that finds subgraphs that either match the desirable query pattern exactly, or as well as possible. We propose an intuitive goodness score $g()$ to measure how well a subgraph matches the query pattern, and we give a fast algorithm to find and rank qualifying subgraphs. The main advantages of G-Ray lie in two folds: (1) Effectiveness: G-Ray returns the best-effort results. That is, the matching subgraphs will include all the nodes in the pattern query and will conform to the pattern query’s graph structure – even when the exact pattern does not exist in the data graph. The method carefully tolerates longer, indirect paths, as guided by our proposed goodness score $g()$; (2) Scalability: G-Ray scales up linearly with respect to the size of the data graph.

Fig. 2(a-f) show three queries (star, line, loop) and the resulting retrieved graphs, using the DBLP data set. In all the cases, the results make sense. Let us analyze the ‘star’ query first, which requests a star-shape group of co-authors, with one author from each of PODS, IAT (‘Intelligent Agent Technology’) and ISBMS (‘Int. Symposium on Biomedical Simulation’). We see that Philip Yu is in the center, with the rest of the matching nodes being well known domain experts (H. Wang of IBM, Mark Zhang for Agents); the connection to biomedical simulation is strained, requiring an interception (by Bing Liu). For the line query (‘find a chain of co-authors, from STOC to SIGMOD to ICML to ISBMS’), again G-Ray retrieves well established researchers from theory (Charikar), databases (Garcia-Molina), machine learning (Fayyad); and, again, the connection to biomedical simulation is strained, requiring 3 intermediate nodes (in white, or unshaded). The loop query (KDD, RECOMB, INFOCOMM, and ICML) is also very interesting. There is a gap between KDD96 and RECOMB00 (biology). In addition, there is a surprising, direct link between biomedical and computer networks (Karp-Shenker). Finally, there is a long path from INFOCOMM00 to ICML93 (probably due to both chronological difference, as well as the lack of interaction between the research communities).

4.1.3 (Q1.3) Querying with Feedback

Here, we want address users’ feedback in the querying process. There are a wide range of scenarios where users’ feedback, both implicit or explicit, exists in the querying process. For instance, in recommendation systems, side information could be users’ ratings on items (e.g., I like Kung-Fu Panda). In Blog analysis, it could be opinions and sentiments. Additionally, for many real applications, users’ preferences can be estimated from click-through data. That said, it is thus important to incorporate such side information in the querying process so that the results are well-tailored to reflect a user’s individual preferences.

The basic of the proposed ProSIN [TQJ08] is to incorporate users’ feedback to measure the node proximity based on which the querying result is built. Intuitively, for a given source node $s$, if positive nodes
Figure 2: Some typical queries on DBLP dataset and some of their results by G-Ray.

(a) The star-query

(b) One resulting subgraph

(c) The line-query

(d) One resulting subgraph

(e) The loop-query

(f) One resulting subgraph
exist, the proximity score from the source node to such positive nodes as well as their neighboring nodes should increase, compared to the case where such side information is unavailable. Analogously, if negative nodes exist, the proximity scores from the source node to such negative nodes as well as their neighboring nodes should decrease, compared to the case where such side information is unavailable.

By incorporating the users’ feedback, we can allow interactive neighborhood search on the graph. Fig. 3 gives one such example, where we want to find the top 10 neighbors of ‘KDD’ conferences (i.e., the 10 most similar conferences as ‘KDD’) from the DBLP dataset. In Fig. 3(a), we plot the initial results when there is no side information (i.e., $P = \phi$ and $N = \phi$). Subjectively, the result makes sense, which reflects two major sub-communities in ‘KDD’: the AI/statistic community (e.g., ‘ICML’, ‘NIPS’, and ‘IJCAI’) and the databases community (e.g., ‘SIGMOD’, ‘VLDB’, ‘ICDE’ etc). Then, if the user gives negative feedback on ‘ICML’ (i.e., $P = \phi$ and $N = \{‘ICML’\}$), all the AI/statistic related conferences (‘NIPS’ and ‘IJCAI’) disappear (See Fig. 3(b)). In Fig. 3(c), we present the updated result if the user further gives some positive feedback on ‘SIGIR’, which is one of the major conferences on information retrieval. Again, the result confirms the effectiveness of ProSIN: positive feedback on ‘SIGIR’ brings more information retrieval related conferences (e.g., ‘TREC’, ‘CIKM’, ‘ECIR’, ‘CLEF’, ‘ACL’, ‘JCDL’, etc).

Figure 3: Interactive neighborhood search for ‘KDD’ conference.

<table>
<thead>
<tr>
<th>(a) No feedback</th>
<th>(b) $N = {‘ICML’}$</th>
<th>(c) $N = {‘ICML’}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘ICDM’</td>
<td>‘ICDM’</td>
<td>‘SIGIR’</td>
</tr>
<tr>
<td>‘ICML’</td>
<td>‘SDM’</td>
<td>‘TREC’</td>
</tr>
<tr>
<td>‘SDM’</td>
<td>‘PKDD’</td>
<td>‘CIKM’</td>
</tr>
<tr>
<td>‘VLDB’</td>
<td>‘ICDE’</td>
<td>‘ECIR’</td>
</tr>
<tr>
<td>‘ICDE’</td>
<td>‘VLDB’</td>
<td>‘CLEF’</td>
</tr>
<tr>
<td>‘SIGMOD’</td>
<td>‘SIGMOD’</td>
<td>‘ICDM’</td>
</tr>
<tr>
<td>‘NIPS’</td>
<td>‘PAKDD’</td>
<td>‘JCDL’</td>
</tr>
<tr>
<td>‘PKDD’</td>
<td>‘CIKM’</td>
<td>‘VLDB’</td>
</tr>
<tr>
<td>‘IJCAI’</td>
<td>‘SIGIR’</td>
<td>‘ACL’</td>
</tr>
<tr>
<td>‘PAKDD’</td>
<td>‘WWW’</td>
<td>‘ICDE’</td>
</tr>
</tbody>
</table>

4.2 Q2: Link Prediction and Proximity Tracking

There are two sub tasks for Q2: (1) link prediction, where we focus on how to prediction the direction (as opposed to existence) of the link; and (3) proximity and centrality tracking, where we deal with the temporal issue in querying process.

4.2.1 (Q2.1) Link Prediction

Here, we focus on how to predict the direction of the link: given a pair of nodes $i$ and $j$ and we know that there is a single link between them; and we want to know the direction of the link (e.g., in phone network, to predict who calls whom, etc).
Figure 4: Density histogram of $\text{Prox}(i_k, j_k) - \text{Prox}(j_k, i_k)$ on the Web Link data set. Successful predictions correspond to the positive values of the $x$ axis, and they clearly outnumber the wrong predictions (the negative $x$-values).

Our solution (FastDAP [TFK07]) is to compare the proximity scores in two directions: if the proximity score from node $i$ to node $j$ is bigger than the score in the opposite direction, we predict a link from $i$ to $j$; otherwise predict the opposite direction. To verify this idea, we construct a test set $(i_1, j_1), \ldots, (i_{n'}, j_{n'})$, such that for each pair $(i_k, j_k)$ in the test set, there exists a link from $i_k$ to $j_k$ and there is no link in the opposite direction. Fig. 4 plots the histogram of $\text{Prox}(i_k, j_k) - \text{Prox}(j_k, i_k)$ for a WebLink graph. It can be seen that as desired, the histogram is biased w.r.t. the origin: there are many more pairs in the positive zone.

4.2.2 (Q2.2) Proximity and Centrality Tracking

Here, we focus on the temporal aspect in querying large graphs. In many real settings, the graphs are evolving and growing over time, e.g., new links arrive or link weights change. Consider an author-conference evolving graph, which effectively contains information about the number of papers (edge weights) published by each author (type 1 node) in each conference (type 2 node) for each year (time stamp). Trend analysis tools are becoming very popular. For example, Google Trends\(^1\) provides useful insights, despite the simplicity of its approach. For instance, in the setting of our example, a tool similar to Google Trends might answer questions such as “How does the number of papers published by an author vary over time?” or “How does the number of papers published in a particular conference or research area (i.e., set of conferences) vary over time?” This kind of analysis takes into account paper counts for either an author or a conference alone or, at best, a single, specific author-conference pair. Instead, we want to employ powerful analysis tools to analyze the entire graph and provide further insight, taking into account all author-conference information so far, i.e., including indirect relationships among them. However, if we need to essentially incorporate all pairwise relationships in the analysis, scalability quickly becomes a major issue. This is precisely the problem we address in pTrack and cTrack [TPYF08b]: how can we efficiently keep track of proximity and avoid global re-computation as new information arrives.

Fig. 5(a) gives an example of pTrack applied to DBLP author-conference data set, where we want to track the top-5 conferences for Dr. ‘Philip S. Yu’. The major research interest (top-5 conferences) for ‘Philip S. Yu’ is changing over time. For example, in the years 1988-1992, his major interest is in databases (‘ICDE’

\(^1\)http://www.google.com/trends/
Figure 5: Some examples of pTrack and cTrack.

<table>
<thead>
<tr>
<th>ICDE</th>
<th>CIKM</th>
<th>KDD</th>
<th>ICDM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICDCS</td>
<td>ICDCS</td>
<td>SIGMOD</td>
<td>KDD</td>
</tr>
<tr>
<td>SIGMETRICS</td>
<td>ICDE</td>
<td>ICDCS</td>
<td>ICDM</td>
</tr>
<tr>
<td>PDIS</td>
<td>SIGMETRICS</td>
<td>CIKM</td>
<td>SDM</td>
</tr>
<tr>
<td>VLDB</td>
<td>ICMCS</td>
<td>ICDCS</td>
<td>VLDB</td>
</tr>
</tbody>
</table>

(a) Philip S. Yu’s top 5 conferences at four time steps, using a window of 5 years.

(b) The ranking of centrality for some authors in NIPS.

and ‘VLDB’), performance (‘SIGMETRICS’) and distributed systems (‘ICDCS’ and ‘PDIS’). However, during 2003-2007, while databases (‘ICDE’ and ‘VLDB’) are still one of his major research interests, data mining became a strong research focus (‘KDD’, ‘SDM’ and ‘ICDM’). Fig. 5(b) gives an example of cTrack applied to NIPS co-authorship data set, where we want to track how influential for a given author in the whole NIPS community. The results are consistent with intuition. For example, Michael I. Jordan starts to have significant influence (within top-30) in the NIPS community from 1991 on; his influence rapidly increases in the following up years (1992-1995); and stays within the top-3 from 1996 on. Prof. Christof Koch (‘Koch, C’) from Caltech remains one of the most influential (within top-3) authors in the whole NIPS community over the years (1990-1999).

4.3 Q3: Scalability Issues in Querying

Here, we address the scalability issues for querying. The main tool for Q1-Q3 is proximity measurement, that is, given two nodes A and B on the graph, how close is the target node B related to the source node A? Straight-forward methods either need to pre-compute and store a big matrix inversion or need a lot of on-line sparse matrix-vector multiplications, which is prohibitive for large, disk resident graphs and/or undesirable.
for real/near-real time response requirements in many applications.

To deal with this issue, we have designed a family of fast solutions to compute node proximity in several different settings, which are summarized in Tab. 3. An example of our B\_LIN applied to DBLP data set is plotted in Fig. 6.

**Table 3: FastProx: The family of fast solutions for proximity measurement.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Settings</th>
<th>Basic Ideas</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>B_LIN [TFP06, TFP08]</td>
<td>One proximity score on general static graphs</td>
<td>Efficiently approximate one big linear system</td>
<td>90%+ quality preserving; up to 150x speedup</td>
</tr>
<tr>
<td>BB_LIN [TFP06]</td>
<td>One proximity score on skewed static bipartite graphs</td>
<td>Leverage the intrinsic complexity of the given linear system</td>
<td>No quality loss; up to 1,800x speedup</td>
</tr>
<tr>
<td>FastAllDAP [TFK07]</td>
<td>all or many proximity on median size graphs</td>
<td>Simultaneously solve multiple correlated linear systems</td>
<td>No quality loss; up to 1,000x speedup</td>
</tr>
<tr>
<td>FastUpdate [TPYF08b, TPYF08a]</td>
<td>One proximity score on time-evolving skewed bipartite graphs</td>
<td>Efficiently track smooth linear system</td>
<td>No quality loss; up to 176x speedup</td>
</tr>
<tr>
<td>FastProSIN [TQJ08]</td>
<td>One proximity score on general static graphs with on-line feedback</td>
<td>Leverage the smoothness of graphs with and without feedback</td>
<td>90%+ quality preserving; 49x speedup</td>
</tr>
</tbody>
</table>

5 Completed Work on Mining

In this Section, we summarize our completed work for mining (M1 and M2). For each sub task, we briefly describe the problem definitions and the main ideas of our solutions. We skip the details of our methods for brevity. (Please refer to our published papers listed in Tab. 2 for more details)

5.1 M1: Anomaly Detection

The goal of M1 is to spot anomaly in graphs. Naturally, low-rank approximations on matrices provide powerful tools to answer this question. Formally, a rank-\(c\) approximation of matrix \(A\) is a matrix \(\tilde{A}\) where \(\tilde{A}\) is of rank \(c\) and \(\|A - \tilde{A}\|\) is small. The low-rank approximation is usually presented in a factorized form e.g., \(\tilde{A} = LMR\) where \(L\), \(M\), and \(R\) are of rank-\(c\).

Depending on the properties of those matrices, many different approximations have been proposed in the literature. For example, in SVD [GVL89], \(L\) and \(R\) are orthogonal matrices whose columns/rows are singular vectors and \(M\) is a diagonal matrix whose diagonal entries are singular values. Among all the possible rank-\(c\) approximations, SVD gives the best approximation in terms of squared error. However, the SVD is usually dense, even if the original matrix is sparse. Furthermore, the singular vectors are abstract notions of best orthonormal basis, which is not intuitive for the interpretation.
Figure 6: Illustration of our B\textsubscript{LIN}, on the DBLP graph, with 2 million edges. The X-axis is the pre-compute time and Y-axis is the on-line response time (both in logarithm scales). Our method and its variations (circle, diamond, triangle) carefully balance the pre-compute and on-line query time, and achieves 2 orders of magnitude savings over the other two methods (‘OnTheFly’ and ‘PreCompute’). See details in [TFP06].

Recently, alternatives have started to appear, such as CUR [DKM05] and CMD [SXZF07], which use the actual columns and rows of the matrix to form L and R. We call these example-based low-rank approximations. The benefit is that they provide an intuitive as well as sparse representation, since L and R are directly sampled from the original matrix. However, the approximation is often sub-optimal compared to SVD and the matrix M is no longer diagonal, which means a more complicated interaction.

Despite of the vast amount of literature on these topics, one of the major research challenges lies in the efficiency: (1) for a static graph, given the desired approximation accuracy, we want to compute the example-based low-rank approximation with the least computational and space cost; and (2) for a dynamic graph, we want to monitor/track this approximation efficiently over time.

To deal with the above challenges, we propose the family of Colibri methods. Adjacency matrices for large graphs may contain near-duplicate columns. For example, all nodes that belong to the same closed and tightly-connected community would have the same sets of neighbors (namely, the community’s members). CMD addresses the problem of duplicate elimination. However, even without duplicates, it is still possible that the columns of L are linearly dependent, leading to a redundant representation of the approximating subspace, which wastes both time and space. The main idea of our method for static graphs (Colibri-S) is to eliminate linearly dependent columns while iterating over sampled columns to construct the subspace used for low rank approximation. Formally, the approximation $\tilde{A} = LMR$ where L consists of judiciously selected columns, M is an incrementally maintained core matrix, and R is another small matrix. Colibri-S is provably better or equal compared to the best competitors in the literature, in terms of both speed and space.
cost, while it achieves the same approximation accuracy. In addition, we provide an analysis of the gains in terms of the redundancy present in the data. Furthermore, our experiments on real data show significant gains in practice. With the same approximation accuracy, Colibri-S is up to $52 \times$ faster than the best known competitor, while it only requires about $1/3$ of the space.

For dynamic graphs, we propose Colibri-D. Again, for the same accuracy, Colibri-D is provably better or equal compared to the best known methods (including our own Colibri-S) in terms of speed. The main idea of Colibri-D is to leverage the “smoothness”, or similarity between two consecutive time steps, to quickly update the approximating subspace. Our experiments show that, with the same accuracy, Colibri-D achieves up to $112 \times$ speedup over the best published competitor, and is 5 times faster than Colibri-S applied from scratch for each time step.

5.2 M2: Mining Time

The goal of M2 is to mine time in some complex context. Given a collection of complex, time-stamped events, how do we find patterns and anomalies? Events could be meetings with one or more persons with one or more agenda items at zero or more locations (e.g., teleconferences), or they could be publications with authors, keywords, publishers, etc. In such settings, we want to solve the following problems: (1) find time stamps that look similar to each other and group them; (2) find anomalies; (3) provide interpretations of the clusters and anomalies by annotating them; (4) automatically find the right time-granularity in which to do analysis. Moreover, we want fast, scalable algorithms for all these problems.

We address the above challenges through two main ideas. The first (T3) is to turn the problem into a graph analysis problem, by carefully treating each time stamp as a node in a graph. This viewpoint brings to bear the vast machinery of graph analysis methods (PageRank, graph partitioning, proximity analysis, and CenterPiece Subgraphs, to name a few). Thus, T3 can automatically group the time stamps into meaningful clusters and spot anomalies. Moreover, it can select representative events/persons/locations for each cluster and each anomaly, as their interpretations. Fig. 7(a) gives the mining result on CIKM data set using T3 which reveal a line shape of time over publication years. Also, we find two time clusters (green circles vs. red dots in Fig. 7(a)) as well as their interpretations in Fig. 7(b). (For simplicity, we do not show the representative papers in the table.) From Fig. 7, we can see that while there are quite a lot of research interest in deductive databases and rule systems in the CIKM community in 1990s, attention has been shifted to XML, statistical learning, language, etc since 2000.

The second idea (MT3) is to use temporal multi-resolution analysis (e.g., minutes, hours, days). We show that MT3 can quickly derive results from finer-to-coarser resolutions, achieving up to 2 orders of magnitude speedups.

6 Proposed Work

6.1 P1: Community Detection with Optional Constraints

There are two seemingly opposite efforts in terms of community detection on large graphs. On one hand, researchers focus on how to develop parameter-free methods which require no user invention. On the other hand, there are a lot of research interest in constraints based clustering (i.e., how to incorporate side information such as ‘must link’, ‘cannot link’) in the recent years.

We propose to develop community detection methods to fill the above gap, which will naturally encode the constraints from users but require no extra parameters.
Figure 7: Mining result on CIKM data set by $T3$.

(a) Clustering for publication years

<table>
<thead>
<tr>
<th>Time Cluster</th>
<th>Selected Sessions</th>
<th>Selected Authors</th>
<th>Selected Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>1993</td>
<td>deductive &amp; rule systems</td>
<td>elke_rundensteiner</td>
<td>knowledge</td>
</tr>
<tr>
<td></td>
<td>query_processing</td>
<td>daniel_miranker</td>
<td>system</td>
</tr>
<tr>
<td>1994</td>
<td>knowledge_representation &amp; expert systems</td>
<td>andreas_henrich</td>
<td>unstructured</td>
</tr>
<tr>
<td>1995</td>
<td>object-oriented databases</td>
<td>il-yeol_song</td>
<td>rule</td>
</tr>
<tr>
<td>1996</td>
<td>access_to_unstructured_information</td>
<td>scott_k_huffman</td>
<td>transaction</td>
</tr>
<tr>
<td>1997</td>
<td>deductive databases</td>
<td>ling_liu</td>
<td>object-oriented</td>
</tr>
<tr>
<td>1998</td>
<td>document_processing</td>
<td>robert_jh_hall</td>
<td>document</td>
</tr>
<tr>
<td>1999</td>
<td>logical &amp; deductive databases</td>
<td>jian_tang</td>
<td>deductive</td>
</tr>
<tr>
<td></td>
<td>tools_for_realizing_intelligent_systems</td>
<td>ibrahim_kamel</td>
<td>ai</td>
</tr>
<tr>
<td>2000</td>
<td>xml_schemas; integration_and_translation</td>
<td>james_p_callan</td>
<td>web</td>
</tr>
<tr>
<td>2001</td>
<td>classification</td>
<td>javed_a_ashkan</td>
<td>cluster</td>
</tr>
<tr>
<td>2002</td>
<td>language_models</td>
<td>w_bruc_croft</td>
<td>classification</td>
</tr>
<tr>
<td>2003</td>
<td>corpus_linguistics</td>
<td>marius_pascu</td>
<td>language</td>
</tr>
<tr>
<td>2004</td>
<td>high-dimensional_indexing</td>
<td>james_allan</td>
<td>xml</td>
</tr>
<tr>
<td>2005</td>
<td>semistructured_data</td>
<td>philip_s_yu</td>
<td>similarity</td>
</tr>
<tr>
<td>2006</td>
<td>data warehousing and olap</td>
<td>anton_leuski</td>
<td>stream</td>
</tr>
<tr>
<td>2007</td>
<td>text_classification_and_categorization</td>
<td>george_karypis</td>
<td>learn</td>
</tr>
<tr>
<td></td>
<td>summarization_and_corpus_analysis</td>
<td>charles_clarke</td>
<td>mobile</td>
</tr>
</tbody>
</table>

(b) Interpretations for time clusters
6.2 P2: Mining Space

Given the data sets collected at different locations, how can we find the locations that look similar with each other? how to spot the abnormal locations? And finally we would like to provide the interpretations for our findings.

We propose to extend our $T3$ and $MT3$ to address this problem.

6.3 P3: Diffusion Wavelet

The normalized graph Laplacian $T$ is the base for a lot of graph querying and mining techniques. For instance, the spectrum of $T$ (often by eigen-decomposition) is often a good indicator for community detection as well as anomaly detection. The inverse of $(I - T)$ is closed related to proximity measurement, which is the main tool for querying on large graphs. Diffusion wavelet [CM] is a recently developed tool, which suggests looking at the local spectrum of the graph Laplacian $T$ in multi-scale way. We will try to investigate this new tool to see if it can be used as an alternative/better way for querying and mining large graphs.

6.4 Time Line

- **December, 2008**: Thesis proposal.
- **January-February, 2009**: P1: Community Detection.
- **March-April, 2009**: P2: Mining Space.
- **May-July, 2009**: P3: Diffusion Wavelet.
- **August, 2009**: Thesis write-up.
- **September, 2009**: Thesis defense.

7 Conclusion

Graphs pose a wealth of fascinating problems. The research focus of this thesis work lies in two parts: (1) querying and (2) mining. Our contributions so far are the following. For querying, we have proposed (a) CePS for center-piece subgraph discovery, (b) G-Ray for best effort pattern match, and (c) ProSIN for interactive querying; all of which aim to find some complex user-specific patterns. Then, we have proposed (a) FastDAP to predict the direction of the link, and (b) $pTrack$ and $cTrack$ to deal with the temporal aspect in querying. Finally, we have designed a family of fast solutions ($FastProx$) to address the scalability issues in querying. For mining, we have proposed a family of example-based low-rank approximation methods ($Colibri$) for anomaly detection. To mine the temporal information in some complex context, we have proposed $T3$ and $MT3$.

Future work includes two aspects. First, we would like to design tools to detect communities from graphs with optional constraints. Then, we will study how to mine the spatial information in the context of graphs. Besides, we plan to investigate diffusion wavelet as an alternative way for querying and mining large graphs.
References


