Center-Piece Subgraphs: Problem Definition and Fast Solutions

Hanghang Tong and Christos Faloutsos

Machine Learning Department, School of Computer Science,
Carnegie Mellon Univeristy,
5000 Forbes Avenue, Pittsburgh, PA, 15213, USA
{htong,christos}@cmu.edu
www.cs.cmu.edu/~htong

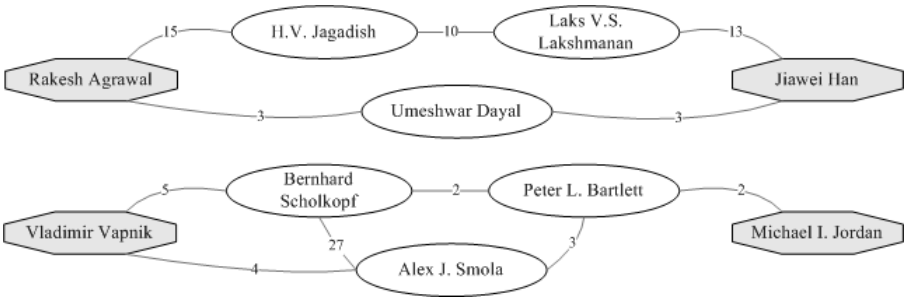
Abstract. Given Q nodes in a social network (say, authorship network), how can we find the node/author that is the center-piece, and has direct or indirect connections to all, or most of them? For example, this node could be the common advisor, or someone who started the research area that the Q nodes belong to. Isomorphic scenarios appear in law enforcement (find the master-mind criminal, connected to all or most of the 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.
Connection subgraphs is an important first step, handling the case of Q=2 query nodes. Then, the connection subgraph algorithm finds the b (say b=20) intermediate nodes, that provide a good connection between the two original query nodes.
Here we generalize the challenge in multiple dimensions: First, we allow more than two query nodes. Second, we allow a whole family of queries, ranging from 'OR' to 'AND', with 'softAND' in-between. Finally, we design and compare a fast approximation. We also present experiments on the DBLP dataset. The experiments confirm that our proposed method naturally deals with multi-source queries and that the resulting subgraphs agree with our intuition. Wall-clock timing results on the DBLP dataset show that our proposed approximation achieve good accuracy for about 6 : 1 speedup.

1 Introduction

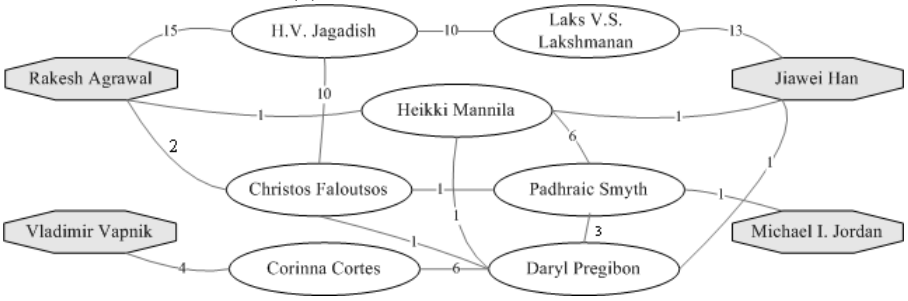
Graph mining has been attracting increasing interest recently, for community detection, partitioning, frequent subgraph discovery and many more. Here we introduce and solve a novel problem, the "*center-piece subgraph*" (*CePS*) problem: Given Q query nodes in a social network (e.g., co-authorship network), find the node(s) and the resulting subgraph, that have strong connections to all or most of the Q query nodes. The discovered nodes could contain a common advisor, or other members of the research group, or an influential author in the research area that the Q nodes belong to. As mentioned in the abstract, there are multiple alternative applications (law enforcement, gene regulatory networks).

Earlier work [8] focused on the so-called "connection subgraphs". Although the inspiration for the current work, the connection subgraph algorithm can only handle the case of Q=2. This is exactly the major contribution of our work: we allow not only pairs of query nodes, but any arbitrary number Q of them.

Figure 1 gives screenshots of our system, showing 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* subgraph, when we request nodes with strong ties to all four query nodes. 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. Figure 1(a) illustrates an important aspect of our work, the $K\_softAND$ feature, which we will discuss very soon. In a nutshell, in a $K\_softAND$ query, our method finds nodes with connections to at least k of the query nodes ($k = 2$ in Figure 1(a)).



(a) "K_softAND query": $k = 2$



(b) "AND query"

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

Thus, we define the center-piece subgraph problem, as follows:

Problem 1. Center-Piece Subgraph Discovery(*CePS*)

Given: an edge-weighted undirected graph $\mathbf{W}$, Q nodes as source queries $\mathcal{Q} = \{q_i\}$ $(i = 1, ..., Q)$, 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 $g(\mathcal{H})$.

By problem 1, there are three requirements in *CePS*: (a) the resulting subgraph is small (with less or equal than b nodes); (b) the subgraph is reasonably connected ("connection") and (c) the nodes in the resulting subgraph are close to the query set (the "closeness"). We will give the detailed definitions of "connection" and "closeness" later in the paper.

Allowing Q query nodes creates a subtle problem: do we want the qualifying nodes to have strong ties to all the query nodes? to at least one? to at least a few? We handle all of the above cases with our proposed $K\_softAND$ queries. Figure 1(a) illustrates the case where we want intermediate nodes with good connections to at least $k = 2$ of the query nodes. Notice that the resulting subgraph is much different now: there are two disconnected components, reflecting the two sub-communities (databases/statistics).

The contributions of this work are the following

- The problem definition, for arbitrary number Q of query nodes, with careful handling of a lot of the subtleties.
- The introduction and handling of $K\_softAND$ queries.
- *EXTRACT*, a novel subgraph extraction algorithm.
- The design of a fast, approximate method, which provides a 6 : 1 speedup with little loss of accuracy.

The system is operational, with careful design and numerous optimizations, like alternative normalizations of the adjacency matrix, a fast algorithm to compute the scores for $K\_softAND$ queries.

Our experiments on a large real dataset (DBLP) show that our method returns results that agree with our intuition, and that it can be made fast (a few seconds response time), while retaining most of the accuracy (about 90%).

The rest of the paper is organized as follows: in Section 2, we review some related work; Section 3 provides an overview of the proposed method: *CePS*. The closeness score calculation is proposed Section 4 and its variants are presented in the Appendix. The "*EXTRACT*" algorithm and the speeding up strategy are provided in Section 5 and Section 6, respectively. We present experimental results in Section 7; and conclude the paper in Section 8.

2 Related Work

In recent years, there is increasing research interest in large graph mining, such as pattern and law mining [3][7][9][26], frequent substructure discovery [33], influence propagation [22], community mining [11][14][15] and so on. Here, we make a brief review of the related work, which can be categorized into five groups: (1) measuring the goodness of closeness; (2) measuring the goodness of connection;

(3) community mining; (4) random walk and electricity related methods; (5) graph partition.

Measuring the goodness of closeness. Defining a good closeness score is the core for center-piece subgraph discovery. Here, the goal is to define a score to measure the closeness of a given node wrt the query set. To this end, we need to define a score to measure the closeness of a given node wrt a single query node. The two most natural measures for such purpose (i.e., the closeness between two nodes) are shortest distance and maximum flow. However, as pointed out in [8], both measurements might fail to capture some preferred characteristics for social network. To be specific, shortest path will suffer from high degree nodes, and also it cannot capture the multiple faceted relationship between two nodes on the graph; while maximum netflow does not punish the longer connections. The closeness function for survivable network [16], which is the count of edge-disjoint or vertex-disjoint paths from source to destination, also fails to adequately model social relationship. A more related distance function is proposed in [24] [29]. However, It cannot describe the multi-faceted relationship in social network since center-piece subgraph aims to discover collection of paths rather than a single path.

Measuring the goodness of connection. Another requirement in *CePS* is "connection". In [8], the authors propose an delivered current based method. By interpreting the graph as an electric network, applying +1 voltage to one query node and setting the other query node 0 voltage, their method proposes to choose the subgraph which delivers maximum current between the query nodes. In [31], the authors further apply the delivered current based method to multi-relational graph. However, the delivered current criterion can only deal with pairwise source queries. Moreover, the resulting subgraph might be sensitive to the order of the query nodes (See Figure 2 for an example). On the other hand, as we will show very soon, connection subgraph can actually be viewed as a special case of the proposed center-piece subgraph ("AND query" with pair source nodes).

The "connection" requirement is also related to Steiner tree [5, 25], where the goal is to find a tree of minimal weight which includes all query nodes. However, the Steiner tree cannot directly apply in our settings for the following reasons: (1) the Steiner tree might suffer from those high degree nodes exactly as the way the shortest path will suffer; (2) to find an exact Steiner tree is NP-complete; and (3) Steiner tree requires to find a tree which connects to all the source nodes. On the other hand, *CePS* tries to find a set of inter-correlated trees to connect the query nodes in an approximate way. By using the proposed closeness function, *CePS* will avoid the high-degree node effect. Also, in the proposed "EXTRACT" algorithm (which will be introduced Section 5), we try to search for a set of paths, instead of searching for a tree directly (as in Steiner tree). Finally, by introducing $K\_softAND$, we can further relax the requirement on connecting to all the source nodes in *CePS*.

Random walk related methods. The proposed importance score calculation is based on random walk with restart. There are many applications using random walk and related methods, including PageRank [28], personalized

PageRank [17], SimRank [19], neighborhood formulation in bipartite graph [32], content-based image retrieval [18], cross modal correlation discovery [30], BANKS system [1], ObjectRank [4], RelationalRank [13] and so on. *CePS* also relates Personized PageRank (PPR) [12] in the sense that PPR defines the combined score as an approximate "OR " query[1]. On the other hand, the proposed *CePS* can naturally deal with different kinds of queries, from "AND " to "OR ", with "$K\_softAND$ query" in-between.

Community detection. Center-piece subgraph discovery is also related with community detection, such as [11][14][15]. However, we cannot directly apply community detection to subgraph discovery especially when the source queries are remotely related or they lie in different communities.

Graph partition and clustering. There are a bunch of graph partition and clustering algorithms proposed in the literature, e.g. METIS [20], spectral clustering [27], flow simulation [10], co-clusterfing [6], betweenness based method [15]. It is worth pointing out that the proposed method is orthogonal to the specific graph partition algorithms.

3 Proposed Method: Overview

Given the budget b, we want to find a subgraph which (a) is reasonably connected ("connection") and (b) the nodes in this subgraph are close wrt the query set ("closeness").

For the "closeness" requirement, we want to find a subgraph $\mathcal{H}$ which is close wrt the query set. To this end, let us first define the closeness score for a single node in this subgraph $\mathcal{H}$. More specifically, for a given node j in $\mathcal{H}$, we have two types of closeness scores:

- Let $r(i, j)$ be the closeness score of a given node j wrt the query q_i;
- Let $r(\mathcal{Q}, j)$ be the closeness score of a given node j wrt the query set $\mathcal{Q}$.

A natural way to measure the closeness of the subgraph $\mathcal{H}$ wrt the query set is to measure the closeness of the nodes it contains: the more close nodes (wrt the source queries) it contains, the better (in terms of closeness) $\mathcal{H}$ is. Thus, the goodness criterion in terms of closeness of $\mathcal{H}$ can be defined as:

$$g(\mathcal{H}) = \sum_{j \in \mathcal{H}} r(\mathcal{Q}, j) \tag{1}$$

By eq. 1, a subgraph is good in terms of closeness if $g(\mathcal{H})$ is high. With the above criterion, a straightforward way to choose the "best" (in terms of closeness) subgraph should be the one which maximizes $g(\mathcal{H})$:

$$\mathcal{H}^* = argmax_{\mathcal{H}} g(\mathcal{H}) \tag{2}$$

[1] To see this, notice that the combined score is defined as $r(\mathcal{Q}, j) = \sum_{i=1}^{Q} r(i, j)$ in PPR.

However, no connection is guaranteed in this way and the resulting subgraph $\mathcal{H}$ might be a collection of isolated nodes. Thus, there are two basic problems in center-piece subgraph discovery: 1) how to define a reasonable closeness score $r(\mathcal{Q}, j)$ for a given node j; 2): how to quickly find a connection subgraph maximizing $g(\mathcal{H})$. Moreover, since it might be very difficult to directly calculate the closeness score $r(\mathcal{Q}, j)$, we further decompose it into two steps. The pseudo code for the proposed method (*CePS*) is listed as follows:

Table 1. *CePS*

Input: the weighted graph $\mathbf{W}$, the query set $\mathcal{Q}$, $K\_softAND$ coefficient k and the budget b
Output: the resulting subgraph $\mathcal{H}$
Step 1: Individual Score Calculation. Calculate the closeness score $r(i, j)$ for a single node j wrt a single query node q_i
Step 2: Combining Individual Scores. Combine the individual score $r(i, j)$ to get the closeness score $r(\mathcal{Q}, j)$ for a single node j wrt the query set $\mathcal{Q}$
Step 3: "*EXTRACT*". Extract quickly a connection subgraph $\mathcal{H}$ with budget b maximizing the closeness criteria $g(\mathcal{H})$

4 Closeness Score Calculation for a Single Node

In this Section, we deal with the closeness score calculation for a single node. That is, how to define the closeness score of a given node wrt the query set. For clarification, whenever we say that a node is 'good' in this Section, we mean that this node is 'good' in term of closeness. Also, we use the terms "goodness" and "closeness" interchangeably in this Section.

There are two basic concepts in closeness score calculation:

- Let $r_{i,j}$ be the *steady-state probability* that a particle will find itself at node j, when it does random walk with restarts (RWR) from query node q_i.
- Let $r(\mathcal{Q}, j, k)$ be the *meeting probability*, that is, the steady-state probability that at least k-out-of-Q particles, doing RWR from the query nodes of $\mathcal{Q}$, will all find themselves at node j in the steady state; k is the K_softAND coefficient.

These two kinds of steady probability ($r_{i,j}$ and $r(\mathcal{Q}, j, k)$) are the base of our closeness score calculation (for both $r(i, j)$ and $r(\mathcal{Q}, j)$). It's basic idea is that: suppose there are Q random particles doing RWR from each query node independently; then after convergency, each particle has some *steady-state probability* staying at the node j; and different particles have some *meeting probability* at the node j. The *steady-state probability* and the *meeting probability* provide some hints on how the node j is related with the source queries, and are used to compute the closeness score of node j. Moreover, by designing different *meeting*

Table 2. Symbols

Symbol	Description
N	total number of nodes in the weighted graph
m	iteration step
c	fly-out probability for random walk with restart
$\boldsymbol{e}_i$	$N \times 1$ unit query vector, with all zeros except one at row q_i
$\mathbf{W} = \{w_{i,j}\}$	the edge weighted matrix $(i, j = 1, ..., N)$
$\mathbf{D} = \{d_{i,j}\}$	$N \times N$ matrix, $d_{i,i} = d_i$, and $d_{i,j} = 0$ for $i \neq j$
d_i	the sum of the i^{th} row of W
$\mathcal{H}$	the chosen center-piece subgraph
Q	number of source query nodes
$\mathcal{Q} = \{q_i\}$	set of query nodes $(i = 1, ..., Q)$
$\acute{\mathcal{Q}}$	the first $(Q-1)$ query nodes of query set $\mathcal{Q}$, $\acute{\mathcal{Q}} = \{q_i\}, (i = 1, .., (Q-1))$
$\varnothing$	null query set, which contains no query node
$r(i, j)$	goodness score for a single node j wrt query node q_i
$r(\mathcal{Q}, j)$	goodness score for a single node j wrt query set $\mathcal{Q}$
$r(\mathcal{Q}, (j, l))$	goodness score for a single edge (j, l) wrt query set $\mathcal{Q}$
$r_{i,j}$	*steady-state probability* of a single node j wrt query node q_i
$\mathbf{R}$	$Q \times N$ matrix of $[r_{i,j}]$
$r(\mathcal{Q}, j, k)$	*meeting probability* of a single node j, wrt $k(k = 1, .., Q)$ or more of the query nodes of $\mathcal{Q}$
$r(i, (j, l))$	*meeting probability* of a single edge (j, l), wrt query node q_i
$r(\mathcal{Q}, (j, l), k)$	*meeting probability* of a single edge (j, l), wrt $k(k = 1, .., Q)$ or more of the query nodes of $\mathcal{Q}$

probability, we can get the specific type of closeness score tailored for the specific query scenario. Table 2 lists all the symbols and definitions used throughout this paper.

4.1 Individual score calculation

Here we want to compute the closeness score $r(i, j)$ of a single node j, for a single query node q_i. We propose to use random walks with restart, from the query node q_i.

Suppose a random particle starts from query q_i, the particle iteratively transmits to its neighborhood with the probability that is proportional to the edge weight between them, and also at each step, it has some probability c to return to node q_i. $r(i, j)$ is defined as the *steady-state probability* $r_{i,j}$ that the particle will finally state at node i:

$$r(i, j) \triangleq r_{i,j} \tag{3}$$

More formally, if we put all the $r_{i,j}$ probabilities into matrix form $\mathbf{R} = [r_{i,j}]$, then

$$\mathbf{R}^T = c\mathbf{R}^T \times \tilde{\mathbf{W}} + (1 - c)\mathbf{E} \tag{4}$$

where $\mathbf{E} = [\boldsymbol{e}_i](i = 1, ..., Q)$ is the $N \times Q$ matrix, $(1 - c)$ is the fly-out probability, and $\tilde{\mathbf{W}}$ is the adjacency matrix $\mathbf{W}$ appropriately normalized, say, column-normalized:

$$\tilde{\mathbf{W}} = \mathbf{W} \times \mathbf{D}^{-1} \tag{5}$$

The problem can be solved in many ways - we choose the iteration method, iterating Eq. 4 until convergence. For simplicity, in this paper, we iterate Eq. 4 m times, where m is a pre-fixed iteration number.

4.2 Combining individual scores

Here we want to combine the individual score $r(i, j)(i = 1, ..., Q)$ to get $r(\mathcal{Q}, j)$, the closeness score for a single node j wrt the query set $\mathcal{Q}$. We propose to use the *meeting probability* $r(\mathcal{Q}, j, k)$ of random walk with restart. Furthermore, by using different softAND coefficient k, we can deal with different types of query scenario.

The most common query scenario might be that "given Q query nodes, find the subgraph $\mathcal{H}$ the nodes of which are important/good wrt ALL queries". In this case, $r(\mathcal{Q}, j)$ should be high if and only if there is a high probability that ALL particles will finally meet at node j:

$$r(\mathcal{Q}, j) \triangleq r(\mathcal{Q}, j, Q) = \prod_{i=1}^{Q} r(i, j) \tag{6}$$

Eq. 6 actually defines a logic AND operation in terms of individual closeness scores: the node j is important wrt the query set $\mathcal{Q}$ if and only if it is important wrt every query node. Thus, we refer such query type as "AND query".

A complemental query scenario is "OR query": "given Q queries, find the subgraph $\mathcal{H}$ the nodes of which are important wrt at least ONE query". In this case, $r(\mathcal{Q}, j)$ should be high if and only if there is a high probability that at least one particle will finally stay at node j:

$$r(\mathcal{Q}, j) \triangleq r(\mathcal{Q}, j, 1) = 1 - \prod_{i=1}^{Q} (1 - r(i, j)) \tag{7}$$

Eq. 7 defines a logic OR operation in terms of individual importance scores: the node j is important wrt the source queries if and only if it is important wrt at least one source query.

Besides the above two typical scenarios, the user might also ask "given Q queries, find the subgraph $\mathcal{H}$ the nodes of which are important wrt at least $k(1 \leq k \leq Q)$ queries". We refer such query type as "$K\_softAND$ query". In this case, $r(\mathcal{Q}, j)$ should be high if and only if there is a high probability that at least k-out-of-Q particles will finally meet at node j.

$$r(\mathcal{Q}, j) \triangleq r(\mathcal{Q}, j, k) \tag{8}$$

To avoid exponential enumeration (which is $O(2^k)$), Eq. 8 can be computed in a recursive manner:

$$r(\mathcal{Q}, j, k) = r(\acute{\mathcal{Q}}, j, k-1) \cdot r(Q, j) + r(\acute{\mathcal{Q}}, j, k) \cdot (1 - r(Q, j)) \tag{9}$$

where $r(\mathcal{Q}, j, 1) = 1 - \prod_{i=1}^{Q} (1 - r(i, j))$.

Intuitively, Eq. 8 defines a logic operation in terms of individual importance scores that is between logic AND and logic OR. In this paper, we refer it as logic K_softAND: the node j is important wrt the source queries if and only if it is important wrt at least k-out-of-Q source queries.

It is worth pointing out that both "AND query" and "OR query" can be viewed as special cases of "$K\_softAND$ query": "AND query" is actually "Q_softAND query"; while "OR query" is actually "1_softAND query"

4.3 Variation: normalization on W

To compute the closeness score $r(i, j)$ and $r(\mathcal{Q}, j)$, we need to construct the transition matrix $\tilde{\mathbf{W}}$ for random walk with restart. A direct way is to normalize $\mathbf{W}$ by column as Eq. 5. However, as pointed out in [8], there might be the so called "pizza delivery person" problem, that is, the node with high degree is prone to receive too much attention (receiving too high individual closeness score in our case). To deal with this problem, we propose to normalize $\mathbf{W}$ as Eq. 10. The normalized weighted graph $\mathbf{W}$ will be further used to formulate the transition matrix $\tilde{\mathbf{W}}$ by Eq. 5.

$$w_{j,l} \leftarrow w_{j,l} / (d_j)^{\alpha} \tag{10}$$

for all $j, l = 1, ..., N$.

The motivation of normalization is as follows: for the high degree node j, every edge $(j, l)(l = 1, ..., N)$ is penalized by $(d_i)^{\alpha}$ and vice versa. The coefficient α control the penalization strength: bigger α indicates stronger penalization. Note that the idea of penalizing the node with high degree is similar with that of setting a universal sink node in [8].

5 The "Extract" Algorithm

In this Section, we propose "EXTRACT" algorithm to deal with the "connection" requirement of *CePS*: what do we mean by "connection" and how to find the resulting subgraph which satisfies the connection requirement while maximizing the goodness/closeness with the limited budget b.

The "EXTRACT" algorithm takes as input the weighted graph $\mathbf{W}$, the importance scores on all nodes, the budget b and the softAND coefficient k; and produces as output a small, unweighted, undirected graph $\mathcal{H}$. The basic idea is similar with the display generation algorithm in [8]: 1) instead of trying to find an optimal subgraph maximizing $g(\mathcal{H})$ directly, we decompose it into finding key

paths incrementally; 2) by sorting the nodes in order, we can quickly find the key paths by dynamic programming in the acyclic graph.

However, we cannot directly apply the original display generation algorithm since it can only deal with pair source queries (and also the resulting subgraph is sensitive to the order of the source queries). To deal with this issue, we extend the original algorithm in the following aspects:

(1) Instead of finding a source-source path, at each step, the algorithm will pick up a most promising destination node *pd*; and try to find a source-destination path for each source query node.
(2) The order (which will be used in the dynamic programming) is specified with each source query node.
(3) Key path discovery differs with the different query types: for "AND query" the algorithm will discover Q paths for all source nodes at each step; for "K_softAND query", it only discovers k paths for the first k source nodes; while for "OR query", the algorithm will only find 1 path at each step.

Before presenting the algorithm, we require the following definitions:

- *SPECIFIED DOWNHILL NODE.* Node u is downhill from node v wrt source q_i ($v \rightarrow d_i, u$) if $r(i,v) > r(i,u)$;
- *SPECIFIED PREFIX PATH.* A specified prefix path $P(i,u)$ is any downhill path that starts from source q_i and ends at node u; that is, $P(i,u) = (u_0, u_1, ..., u_n)$ where $u_0 = q_i, u_n = u$, and $u_j \rightarrow d_i, u_{j+1}$;
- *EXTRACTED GOODNESS.* The extracted goodness is the total goodness score of the nodes within the subgraph $\mathcal{H}$: $CF(\mathcal{H}) = \sum_{j \in \mathcal{H}} r(\mathcal{Q}, j)$.
- *EXTRACTED MATRIX.* $C_s(i,u)$ is the extracted goodness score from source node q_i to node u along the prefix path $P(i,u)$ so that:
  1. $P(i,u)$ has exactly s nodes not in the present output graph $\mathcal{H}$
  2. $P(i,u)$ extracts the highest goodness score among all such paths that start from q_i and end at u.
- *ACTIVE SOURCE.* For $K\_softAND$, the source node q_i is active wrt destination node *pd* if $r(i,pd) \geq r^{(k)}(i,pd)$, where $r^{(k)}(i,pd)$ is the k^{th} largest value among $r(i,pd), (i = 1, ..., Q)$. Note that the number of active source differs with the query type[2]: for "OR query", there is only one active source while for "AND query", all sources are active. For a specific query type, an active source q_i might turn into inactive when the destination node *pd* changes and vice versa.

The destination node *pd* can be decided by Eq. 11:

$$pd = argmax_{j \notin \mathcal{H}} r(\mathcal{Q}, j) \tag{11}$$

where $\mathcal{H}$ is the partially built output subgraph.

[2] Since both "AND query" and "OR query" can be viewed as special cases of "K_softAND query", the number of active sources is actually k for all query types.

In order to make the resulting subgraph to be "reasonably connected", we want to make sure that (1) there is at least one path that connects the destination node *pd* and each query node for AND query; and (2) there is at least one path that connects the destination node *pd* and k-out-of-Q query nodes. In this way, not only does the algorithm select good/close nodes wrt the query set (i.e., a destination node *pd* with high $r(\mathcal{Q}, j)$), but also it provides some interpretations on why such nodes are good/close wrt the query set.

However, we do not want to find an arbitrary path to connect the destination node *pd* and the one query node since (1) we also want to make sure that the remaining nodes (besides the destination node *pd*) in the resulting subgraph are good/close wrt the query set; and (2) the number of total nodes in the resulting subgraph is limited by the budget b. Therefore, we aim to find a path from one query node and the destination node *pd* which maximizes the total captured combined scores along the path over the length of the path. Also, since we try to find the resulting subgraph gradually, a new path might include some existing nodes in the current subgraph. In order to encourage different paths to share with the same nodes since the budget b is limited, we define the length of the path is defined as the number of new nodes in this path.

In order to discover a new path between the source q_i and the promising node *pd*, we arrange the nodes in descending order of $r(i, j)(j = 1, ..., n)$: $\{u_1 = q_i, u_2, u_3, ..., pd = u_n\}$. (note that all nodes with smaller $r(i, j)$ than $r(i, pd)$ are ignored). Then we fill the extracted matrix C in topological order so that when we compute $C_s(t, u)$, we have already computed $C_s(t, v)$ for all $v \rightarrow d_i, u$. On the other hand, as the subgraph is growing, a new path may include nodes that are already present in the output subgraph, our algorithm will favor such paths as in [8]. The complete algorithm to discover a single path from source node q_i and the destination node *pd* is given in table 3.

Table 3. Single Key Path Discovery

1. Let len be the maximum allowable path length
2. For $j \leftarrow [1, ..., n]$
   2.1. Let $v = u_j$
   2.2. For $s \leftarrow [2, ..., len]$
      If v is already in the output subgraph
         $s' = s$
      Else
         $s' = s - 1$
      Let $C_s(i, v) = max_{u|u \rightarrow d_i, v}(C_{s'}(i, u) + r(\mathcal{Q}, v))$
3. Output the path maximizing $C_s(i, pd)/s$, where $s \neq 0$

Based on the previous preparations, the *EXTRACT* algorithm can be given in table 4.

Table 4. Our *EXTRACT* Algorithm

1. Initialize output graph $\mathcal{H}$ null
2. Let *len* be the maximum allowable path length
3. While $\mathcal{H}$ is not big enough
3.1. Pick up destination node *pd* by Eq. 11
3.2. For each active source node q_i wrt node *pd*
3.2.1. use table 3 to discover a key path $P(q_i, pd)$
3.2.2. add $P(q_i, pd)$ to $\mathcal{H}$
4. Output the final $\mathcal{H}$

6 Speeding up CEPS

To compute $r(i, j)$, we have to solve a linear system. When the data set is large (or more precisely, when the total number of the edges in the graph is large), the processing time could be long.

Note that Eq. 4 can be solved in closed form:

$$\mathbf{R}^T = (1 - c)(\mathbf{I} - c\tilde{\mathbf{W}})^{-1}\mathbf{E} \tag{12}$$

Thus, an obvious way to speed up *CePS* is to pre-compute and store the matrix $\mathbf{A} = (\mathbf{I} - c\tilde{\mathbf{W}})^{-1}$, then $\mathbf{R}^T = (1 - c)\mathbf{AE}$ can be computed on-line nearly real-time. However, in this way, we have to store the whole $N \times N$ matrix A, which is a heavy burden when N is big.

As suggested by [32], the goodness score $r(i, j)(j = 1, ..., N)$ is very skewed, that is, most values of $r(i, j)$ are near zero and only a few nodes have high value. Based on this observation, we propose to pre-partition the original weighted graph $\mathbf{W}$ into several partitions and only use the partitions containing the source queries to run *CePS*. In this paper, we use METIS [20] as the partition algorithm.

The pseudo code for the accelerated *CePS* is summarized as follows:

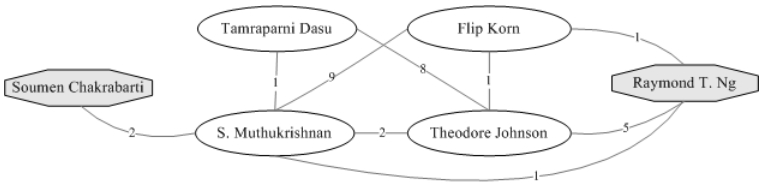
Table 5. Fast *CePS*

Input: the weighted graph $\mathbf{W}$, the query set $\mathcal{Q}$, $K\_softAND$ coefficient k, the budget b, and the number of partitions p;
Output: the resulting subgraph $\mathcal{H}$.
Step 0: pre-partition $\mathbf{W}$ into p pieces (one-time cost)
Step 1: pick up partitions of $\mathbf{W}$ that contain all the query nodes to construct the new weighted graph $\mathbf{nW}$
Step 2:. run *CePS* as in table 1 on $\mathbf{nW}$

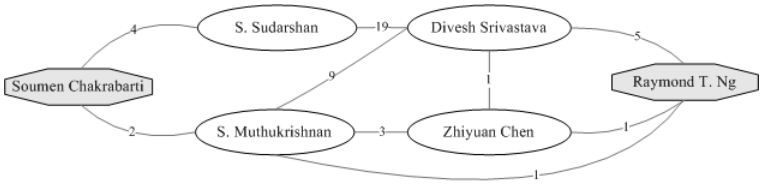
7 Experimental Evaluation

In this section, we demonstrate some experimental results. The experiments are designed to answer the following questions.

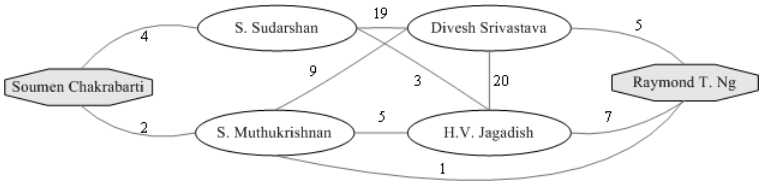
– Does the proposed goodness criterion make sense?
– Does the *EXTRACT* algorithm capture the most goodness score?
– Does the extra normalization step really help?
– how does the pre-partition balance the quality and response time?



(a) by delivered current method (+1 voltage for Raymond and 0 voltage for Soumen)



(b) by delivered current method (+1 voltage for Soumen and 0 voltage for Raymond sink)



(c) by the proposed method

Fig. 2. Connection subgraph between Soumen Chakrabarti and Raymond T. Ng.

Data Set We use the DBLP data set to evaluate the proposed method. To be specific, the author-paper information is used to construct the weighted graph **W**: every author is denoted as a node in **W**; and the edge weight is the number of co-authored papers between the corresponding two authors. On the whole, there is $\approx 315K$ nodes and $\approx 1,834K$ non-zero edges in **W**.

Source Queries To test the proposed algorithm, we select several people from different communities to compose the source-query repository: 13 people from database and mining; 13 people from statistical and machine learning; 11 people from information retrieval; and 11 people from computer vision. Then the source queries are generated by randomly selecting a small number of queries from the repository.

Parameter Setting The re-starting coefficient c in Eq. 4 is set 0.5 and the iteration number m is set 50 since we do not observe performance improvement

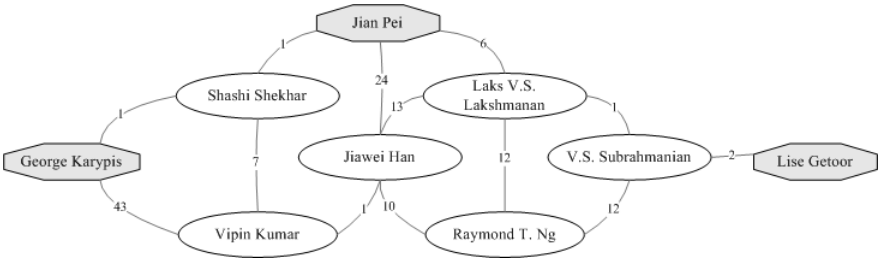


Fig. 3. Center-piece subgraph among Lise Getoor, George Karypis, and Jian Pei.

with more iteration steps. The maximum allowable path length len is decided by the budget b and the number of active sources k as $[b/k]$. For normalization coefficient α, a parametric study is provided in Section 7.3. For other experiments, $\alpha = 0.5$.

Evaluation Criterion Firstly, the resulting $g(\mathcal{H})$ can be evaluated by "Important Node Ratio ($NRatio$)". That is, "how many important/good nodes are captured by $g(\mathcal{H})$?":

$$NRatio = \frac{\sum_{j \in \mathcal{H}} r(\mathcal{Q}, j)}{\sum_{j \in \mathbf{W}} r(\mathcal{Q}, j)} \tag{13}$$

Complementally, we can also evaluate by "Important Edge Ratio ($ERatio$)". That is, "how many important/good edges are captured by $g(\mathcal{H})$?":

$$ERatio = \frac{\sum_{(j,l) \in \mathcal{H}} r(\mathcal{Q}, (j,l))}{\sum_{(j,l) \in \mathbf{W}} r(\mathcal{Q}, (j,l))} \tag{14}$$

The goodness score $r(\mathcal{Q}, (j,l))$ of an edge (j,l) is defined similarly as the goodness score for a node: what is the probability that the specific edge (j,l) will be traversed simultaneously by all (or at least k) of the particles. Firstly, we calculate the goodness score $r(i, (j,l))$ for an edge (j,l) wrt a single query node q_i:

$$r(i, (j,l)) = \frac{1}{2} \cdot (r(i,j) \cdot \tilde{\mathbf{W}}_{l,j} + r(i,l) \cdot \tilde{\mathbf{W}}_{j,l}) \tag{15}$$

Based on Eq. 15, we can easily define $r(\mathcal{Q}, (j,l))$ according to the specific query type. For example, for "AND query", $r(\mathcal{Q}, (j,l))$ can be computed as Eq. 16; while for "OR query" and "K_softAND query", $r(\mathcal{Q}, (j,l))$ can be computed as Eq. 17 and Eq. 18, respectively.

$$r(\mathcal{Q}, (j,l)) \triangleq r(\mathcal{Q}, (j,l), Q) = \prod_{q_i=1}^{Q} r(i, (j,l)) \tag{16}$$

$$r(\mathcal{Q},(j,l)) \triangleq r(\mathcal{Q},(j,l),1) = 1 - \prod_{q_i=1}^{Q} (1 - r(i,(j,l))) \tag{17}$$

$$\begin{aligned} r(\mathcal{Q},(j,l)) &\triangleq r(\mathcal{Q},(j,l),k) \\ &= r(\acute{\mathcal{Q}},(j,l),k-1) \cdot r(Q,(j,l)) + r(\acute{\mathcal{Q}},(j,l),k) \end{aligned} \tag{18}$$

where $r(\varnothing,(j,l),0) = 1$.

For all experiments except subsection 7.1, we run the proposed algorithm multiple times and report the mean $NRatio$ as well as mean $ERatio$.

7.1 Evaluation on the goodness $g(\mathcal{H})$: case study

As we mentioned before, connection subgraph is a special case of center-piece subgraph ("AND query" with pair source nodes). Figure 2 shows the connection subgraph with budget 4 for "Soumen Chakrabarti" and "Raymond T. Ng". It can be seen that both our method and the delivered current method output somewhat reasonable results. It is worth pointing out that the subgraph by the delivered current method is very sensitive to the order of the source queries: comparing figure 2(a) and (b), there is only one common node ("S. Muthukrishnan"). On the other hand, if we compare figure 2(b) and (c), while most nodes are the same for the two methods, It is clear that our method captures more strong connection: compared with figure 2(b), the different node ("H.V. Jagadish") in figure 2(c), 1) has more connections (4 vs. 3) with the remaining nodes and 2) has more co-authored papers with those connected neighbors than the corresponding node in figure 2(b) ("Zhiyuan Chen").

Figure 1 shows an example for multi-source queries. When the user asks for $2 - SoftAND$, the algorithm outputs two clear cliques (figure 1(a)), which makes some sense since "Vladimir Vapnik" and "Michael I. Jordan" belong to statistical machine learning community; while "Rakesh Agrawal" and "Jiawei Han" are database and mining people. On the other hand, if the user asks for "AND", the resulting subgraph shows a strong connection with all four queries.

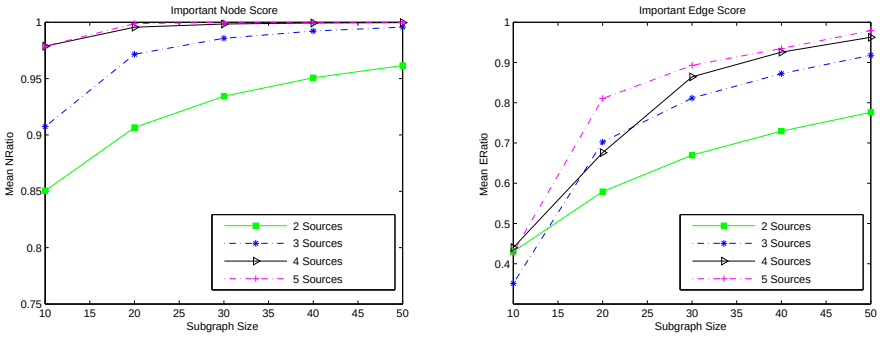
Figure 3 shows an example for "AND query", with "George Karypis", "Lise Getoor" and "Jian Pei" as source nodes. All three researchers are working on graphs. The nodes of the retrieved "center-piece subgraph" are all database, data mining and graph mining people, forming three groups: the nodes close to "Lise Getoor" are related to the University of Maryland ("V.S. Subrahmanian" is a faculty member there and he was the advisor of "Raymond Ng"). The nodes close to "George Karypis" are faculty members at Minnesota ("Vipin Kumar", "Shashi Shekar"). The nodes close to "Jian Pei" are professors at Simon Fraser (SFU) or University of British Columbia (UBC), which are geographically nearby, both in Vancouver: "Jiawei Han" was a faculty member at SFU and thesis advisor of

"Jian Pei" ; "Laks Lakshmanan" and "Raymond Ng" are faculty members at UBC. Not surprisingly, the "center-pieces" of the subgraph consist of "Raymond Ng", "Jiawei Han", "Laks Lakshmanan", which all have direct, or strong indirect connections with the three chosen query sources.

7.2 Evaluation on "*EXTRACT*" algorithm

By the "*EXTRACT*" algorithm, we might miss some good/close nodes (which have high goodness scores) in order to meet the requirement of "connection". To evaluate this potential risk, we use both *NRatio* and *ERatio* as functions of the budget b (Higher *NRatio* and *ERatio* indicate lower risk). Here, we fix the query type as "AND query".

Figure 4(a) shows the mean *NRatio* vs. the budget b for different numbers of source queries; while figure 4(b) shows the mean *ERatio* vs. the budget b for different numbers of source queries. Note that in both cases, our method captures most of important nodes as well as edges by a small number of budget b. For example, for 2 source queries, the resulting subgraph with budget 50 captures 95% important nodes and 70% important edges on average; for 4 source queries, the resulting subgraph with budget 20 captures 100% important nodes and 70% important edges on average. An interesting observation is that for the same budget, the subgraph with more source queries captures higher *NRatio* as well as *ERatio* than those with less source queries. This is consistent with the intuition: generally speaking, finding people that are important wrt all source queries becomes more difficult when the number of source queries increases. In other words, $r(\mathcal{Q}, j)$ becomes more skewed by increasing the number of source queries.

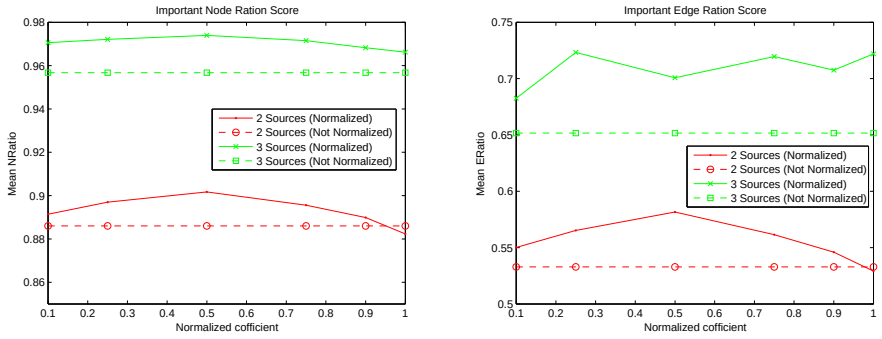


(a) Important node ratio vs. budget (b) Important edge ratio vs. budget

Fig. 4. Evaluation on "*EXTRACT*"

7.3 Evaluation on normalization step

Here we conduct the parametric study for normalization coefficient α. The mean $NRatio$ vs. α is plotted in figure 5(a); and the mean $iERatio$ vs. α is plotted in figure 5(b).



(a) Important node ratio vs. α (b) Important edge ratio vs. α

Fig. 5. Evaluation on normalization step

It can be seen that in most cases, the normalization step does help to improve the performance of the resulting subgraph $g(\mathcal{H})$. For example, the normalization with $\alpha = 0.5$ helps to capture 17.7% more important nodes and 9.1% more important edges for 2 source queries on average; while for 3 source queries, it captures 18.1% more important nodes and 7.6% more important edges on average.

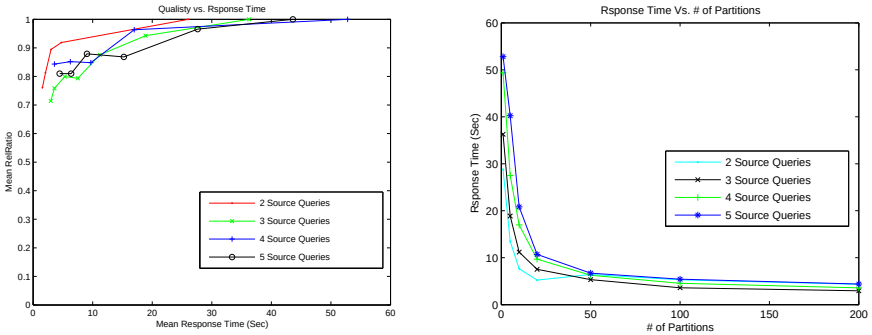
7.4 Evaluation on speedup strategy

For large graph, the response time for importance score calculation could be long. By pre-partition the original graph and performing subgraph discovery only on the partitions containing the source queries, we could dramatically reduce the response time. On the other hand, we might miss a few important nodes if they do not lie in these partitions. To measure such kind of quality loss, we use "Relative Important Node Ratio ($RelRatio$)":

$$RelRatio = \frac{\widehat{NRatio}}{NRatio} \tag{19}$$

where $\widehat{NRatio}$ and $NRatio$ are "Important Node Ratio" for the subgraph by pre-partition and by the original whole graph, respectively.

We fix the budget 20 and the query scenario as "AND query". The mean $RelRatio$ vs. response time is shown in figure 6(a); and the mean response time



(a) Quality vs Time (b) Time vs Number of partitions

Fig. 6. Evaluation on speeding up strategy

vs. the number of partitions is shown in figure 6(b). It can be seen that with a little quality loss, the response process is largely speeded up. For example, with $\approx 10\%$ loss, the subgraph for 2 source queries can be generated within 5 seconds on average; with $\approx 10\%$ quality loss, the subgraph for 5 source queries can be generated within 10 seconds on average. On the other hand, it might take $40s \sim 60s$ without pre-partition. Note that in figure 6 (b), even with a small number of partitions, we can greatly reduce the mean response time.

8 Conclusion and Future Work

Summary of Current Work. We have proposed the problem of "*center-piece subgraphs*", and provided fast and effective solutions. In addition to the problem definition, other contributions of the paper are the following:

- The introduction and handling of $K\_softAND$ queries, which include AND and OR queries as special cases.
- $EXTRACT$, a fast novel algorithm to quickly extract a subgraph with the appropriate connectivity and maximum "goodness" score
- The design and implementation of a fast, approximate algorithm that brings a 6:1 speedup
- Experiments on real data (DBLP), illustrating that our algorithm and "goodness score" indeed derive results that agree with intuition.

Future Work. In the future, we would like to investigate this problem in the following aspects:

1. To study *CePS* from the point view of generative model, such as Kronecker graph [23], mixed-membership model [2], infinite relational model [21] etc.

2. In terms of evaluation, we plan to continue to collect more anecdotal evidence to further verify whether or not the resulting subgraphs are consistent with the users' intuition. Besides, we can test *CePS* by the following two ways: (1) we inject the resulting center-piece which are well justified the users into the original graph and test if the proposed algorithm can find them; (2) use the proposed *CePS* as a retrieval/classification tool and evaluate it by standard precision/recall.
3. Automatic parameter tuning. For example, if the user does not provide the $K\_softAND$ coefficient, how can we infer the 'optimal' k. One possible way to attach this problem is through cross validation (by treating *CePS* as a retrieval/classification tool.
4. Steiner tree and *CePS*. For example, how to leverage the approximate algorithms for Steiner tree so that we can provide theoretic performance guarantee for *CePS*; how to generalize the Steiner tree by *CePS* (e.g., to find a set of inter-correlated, rather than one, Steiner tree; to find the "soft" Steiner tree which connects at least k-out-of-Q queries node etc).

References

1. B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, and S. S. Parag. Banks: Browsing and keyword searching in relational databases. In *VLDB*, pages 1083–1086, 2002.
2. E. A.Erosheva, S. E. Fienberg, and J. Lafferty. Mixed-membership models of scientific publications. In *Proceedings of the National Academy of Sciences*, 2004.
3. R. Albert, H. Jeong, and A.-L. Barabasi. Diameter of the world wide web. *Nature*, (401):130–131, 1999.
4. A. Balmin, V. Hristidis, and Y. Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *VLDB*, pages 564–575, 2004.
5. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
6. I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 03)*, Washington, DC, August 24-27 2003.
7. S. Dorogovtsev and J. Mendes. Evolution of networks. *Advances in Physics*, 51:1079–1187, 2002.
8. C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *KDD*, pages 118–127, 2004.
9. M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *SIGCOMM*, pages 251–262, Aug-Sept. 1999.
10. G. Flake, S. Lawrence, and C. Giles. Efficient identification of web communities. In *KDD*, pages 150–160, 2000.
11. G. Flake, S. Lawrence, C. L. Giles, and F. Coetzee. Self-organization and identification of web communities. *IEEE Computer*, 35(3), Mar. 2002.
12. D. Fogaras and B. Racz. Towards scaling fully personalized pagerank. In *Proc. WAW*, pages 105–117, 2004.
13. F. Geerts, H. Mannila, and E. Terzi. Relational link-based ranking. In *VLDB*, pages 552–563, 2004.

14. D. Gibson, J. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *Ninth ACM Conference on Hypertext and Hypermedia*, pages 225–234, New York, 1998.
15. M. Girvan and M. E. J. Newman. Community structure is social and biological networks.
16. M. Grötschel, C. L. Monma, and M. Stoer. Design of survivable networks. In *Handbooks in Operations Research and Management Science 7: Network Models*. North Holland, 1993.
17. T. H. Haveliwala. Topic-sensitive pagerank. *WWW*, pages 517–526, 2002.
18. J. He, M. Li, H. Zhang, H. Tong, and C. Zhang. Manifold-ranking based image retrieval. In *ACM Multimedia*, pages 9–16, 2004.
19. G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *KDD*, pages 538–543, 2002.
20. G. Karypis and V. Kumar. Parallel multilevel k-way partitioning for irregular graphs. *SIAM Review*, 41(2):278–300, 1999.
21. C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, 2006.
22. D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. *KDD*, 2003.
23. J. Leskovec, D. Chakrabarti, J. M. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In *PKDD*, pages 133–145, 2005.
24. D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proc. CIKM*, 2003.
25. C. L. Lu, C. Y. Tang, and R. C.-T. Lee. The steiner tree problem. *Theoretical Computer Science*, 306:55–67, 2003.
26. M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.
27. A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.
28. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. Paper SIDL-WP-1999-0120 (version of 11/11/1999).
29. C. R. Palmer and C. Faloutsos. Electricity based external similarity of categorical attributes. *PAKDD 2003*, April-May 2003.
30. J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. In *KDD*, pages 653–658, 2004.
31. C. Ramakrishnan, W. Milnor, M. Perry, and A. Sheth. Discovering informative connection subgraphs in multi-relational graphs. *SIGKDD Explorations Special Issue on Link Mining*, 2005.
32. J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *ICDM*, pages 418–425, 2005.
33. D. Xin, J. Han, X. Yan, and H. Cheng. Mining compressed frequent-pattern sets. In *VLDB*, pages 709–720, 2005.
34. D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Scholkopf. Learning with local and global consistency. In *NIPS*, 2003.

A Appendix

Here, we provide and discuss some variants on goodness score calculation.

– **Variant 1: calculate $r_{i,j}$ by manifold ranking**

One potential problem with Eq. 4 is that such goodness score might be asymmetric, that is $r_{i,j} \neq r_{j,i}$. For social network, this is OK since that person X is important/good for person Y does not necessarily mean that person Y is also important/good for person X. However, in some other applications, symmetry might be a desirable property for the goodness score. To deal with this problem, we can define $r_{i,j}$ as manifold ranking score [34].

Formally, $r_{i,j}$ in this case can be computed by replacing the transition matrix $\tilde{\mathbf{W}}$ in Eq. 4 by graph Laplacian S:

$$\mathbf{R}^T = c\mathbf{R}^T \times \mathbf{S} + (1-c)\mathbf{E} \tag{20}$$

where $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ is graph Laplacian.

Note that since S is symmetric, the individual goodness score $r_{i,j}$ by Eq. 20 is always symmetric. That is, $r_{i,j} = r_{j,i}$. However, in this case, the resulting goodness score $r_{i,j}$ is no longer the *steady-state probability*, that is $\sum_{j=1}^{N} r_{i,j} \neq 1$. In our experiments, we find that the resulting subgraphs by Eq. 4 and Eq. 20 are actually quite similar.

– **Variant 2: calculate $r(\mathcal{Q}, j)$ by order statsitic**

Let $r^{(k)}(i,j)$ be the order statistic of $r(i,j), (i = 1, ..., Q)$. That is, $r^{(k)}(i,j)$ is the k^{th} largest value among $r(i,j), (i = 1, ..., Q)$.

Then, we can also use $r^{(k)}(i,j)$ to get $r(\mathcal{Q}, j)$. For example, we can use minimum order statistic as goodness score for "AND query":

$$r(\mathcal{Q}, j) \triangleq r^{(Q)}(i,j) = min(r(1,j), r(2,j), ..., r(Q,j)) \tag{21}$$

The probabilistic interpretation of Eq. 21 is that the node j is important wrt the source queries if and only if there is at least some high probability for every particle to finally stay at node j.

Similarly, the order statistic variants for "OR query" and "K_softAND query" can be defined as $r^{(1)}(i,j)$ and $r^{(k)}(i,j)$, respectively.