Context-based People Search in Labeled Social Networks

Cheng-Te Li[1], Man-Kwan Shan[2], Shou-De Lin[1]
Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei, Taiwan [1]
Department of Computer Science, National Chengchi University, Taipei, Taiwan [2]
d98944005@csie.ntu.edu.tw, mkshan@cs.nccu.edu.tw, sdlin@csie.ntu.edu.tw

ABSTRACT
In online social networking services, there are a range of scenarios in which users want to search a particular person given the targeted person one's name. The challenge of such people search is *namesake*, which means that there are many people possess the same names in the social network. In this paper, we propose to leverage the query contexts to tackle such problems. For example, given the information of one's graduation year and city, the last names of some individuals, one may wish to find classmates from his/her high school. We formulate such problem as the *context-based people search*. Given a social network in which each node is associated with a set of labels and given a query set of labels consisting of a targeted name label and other context labels, our goal is to return a ranking list of persons who possess the targeted name label and connects to other context labels with minimum communication costs through an effective subgraph in the social network. We consider the interactions among query labels to propose a grouping-based method to solve the context-based people search. Our method consists of three major parts. First, we model those nodes with query labels into a group graph which is able to reduce the search space to enhance the time efficiency. Second, we identify three different kinds of connectors which connecting different groups, and exploit connectors to find the corresponding detailed graph topology from the group graph. Third, we propose a Connector-Steiner Tree algorithm to retrieve a resulting ranked list of individuals who possess the targeted label. Experimental results on the DBLP bibliography data show that our grouping-based method can reach the good quality of returned persons as a greedy search algorithm at a considerable outperformance on the time efficiency.

Categories and Subject Descriptors
H.2.8 [**Database Management**]: Database Applications – *Data mining*. H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Information Filtering.*

General Terms
Algorithms, Performance, Design.

Keywords
People Search, Social Network, Context-based Search.

1. INTRODUCTION
In online social networking services such as Facebook, Twitter, and LinkedIn, it is essential to provide people search which searches for an individual by name. However, if the query name is a namesake, especially if there exist millions of individuals share the query name, it would be difficult to find the target person over social networking services.

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

One approach is the *context-based people search* which search for an individual not only by name of the target, by also by the social contexts of the target. The user specified social contexts may be the first names of target's friends, the last names of target's classmates, the hobbies of the target's colleagues, hometowns of target's relatives, and so on. These social contexts are *labels* associated with each individual in social networking services.

The idea of the proposed context-based social search is illustrated by Figure 1. Each person is associated with labels, where some have many labels and others provide few labels or only their user names (e.g., Angel). It can be observed that there are three people named Sam and two named Mary. The weights on the edges indicate the communication cost between two people. Higher weights indicate that the people tend to have a casual acquaintance, whereas lower values imply that they are more familiar with one another. If someone seeks to find a friend named "Mary" who has a friend named George and a colleague familiar with Java, the expected result by our context-based people search *individual search* is "Mary-1," rather than "Mary-2." Another example is to find an individual named "Sam" who had learned the "C++" programming language with his old friend "Robert" in his hometown "Boston", the expected target would be "Sam-1," who is connected with "James", "Jane" and "Robert." Not only they meet the query labels but also interact with each other in an effective manner (with minimal costs).

Given a social network, how to find the individual who is the most relevant to some user-interested social contexts? Specifically, given an arbitrary set of labels as queries in a social network (e.g., an acquaintance network in Facebook), in which each individual is characterized by some set of labels, how can we efficiently find the individual who has strong direct or indirect relationships to all of the query labels?

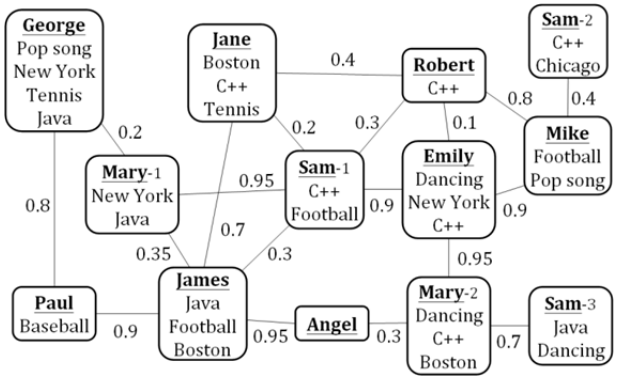


Figure 1. A social network with labels on each person. Labels include name, interest, skill, and hometown. Note that we append "-number" to individuals to distinguish between people with the same name (e.g. Mary-1, Mary-2).

To perform such search tasks in current social networking services containing considerable numbers of people, we have to address two critical and universal issues: *commonality* and *incompleteness*. Commonality indicates that there are a large number of individuals sharing common labels in the network. An obvious

illustration is the presence of popular given names, such as Mary, James, and John. Moreover, there are even more people possessing the same labels, such as interests, hometowns, schools, and graduation years. Incompleteness means that users in social networking sites usually do not (necessarily) provide complete information about themselves. Some enthusiastic members of online social networks fill in all their details, whereas others may only be willing to display their names or a few labels. To address the issues of commonality and incompleteness in real-life social networks, we propose to leverage the user-given social contexts with the social interactions to achieve a context-based social search: (1) for commonality, individuals with the same names can be distinguished by context attributes in the network; (2) for incompleteness, we can also make conjectures about individuals who provide limited information to their social circles.

In the literature, search tasks in the social contexts of Web 2.0, called *social search*, refer to many different aspects. Among these diverse perspectives on the term *social search*, the viewpoint of Kleinberg [5], who defines it as searching over a social graph to find the set of individuals closest to a given user, is the most relevant to our context-based people search. In addition, some commercial solutions to people search, such as Wink [13] and Intelius [14], have also been developed to provide search over online social networking sites. These people search services aim at exploiting social information such as tagging, voting, and profiles to find people related to any given individual. However, existing search services over social networks do not tackle the commonality issue, that is, that there are usually many people with the same names. In addition, they assume that the social information, including personal attributes and user tagging, is complete. Nevertheless, in real-life social networking, it is infeasible to collect complete personal profiles from all users.

In this paper, we propose, formally define, and efficiently and effectively solve the context-based people search problem for a labeled social network. Given a social network in which each node is associated with a set of labels and a user query consisting of a set of required labels, our goal is to find the individual that satisfies the query requirements while minimizing the communication costs. We define the communication cost as the sum of the weights on the edges in a minimum spanning tree connecting both the required labels and the found individual in a direct or indirect manner.

The central idea of the solution to solve the problem of context-based people search is to leverage the effective interactions of query labels. We propose a grouping-based method consists of three major parts. First, we group those nodes with the same query labels and transform the original social network into a group graph which can be regarded as high-level view for the interactions between groups. The group graph is able to reduce to graph space and provides an effective guidance for search tasks. Second, we identify three kinds of connectors which connecting different groups in the social network. The connectors can be regarded as key points controlling the bottleneck of cost in the final subgraph and allowing a more effective graph search. Finally, we propose a Connector-Steiner algorithm to compose the ranked list of effective connection subgraphs as the resulting context-based communities.

2. CONTEXT-BASED PEOPLE SEARCH

[**Definition 1**] (Labeled Social Network) Let $A=\{a_1,...,a_m\}$ be a universe of m labels. A *labeled social network* is defined as an undirected and weighted graph $G=(V, E)$. Each node i in $V=\{1,...,n\}$ is an individual that possesses a set of labels $X_i \subseteq A$. Each edge (i, j) in E captures the interaction between two individuals. The weight on each edge (i, j) represents the communication cost between individuals i and j. Note that edges with low (high) weights represent high (low) communication costs between two nodes. For example, in a social network, if two people share more interests, the weight of their edge is correspondingly lower.

[**Definition 2**] (Communication Cost) Given a labeled social network $G=(V, E)$ and $V' \subseteq V$, the *communication cost* of V' is defined as the sum of the weights of edges of the *minimum spanning tree* on the induced subgraph $G[V']$, denoted by $CC(V')$.

[**Definition 3**] (Context-based Query) A query for the context-based people search consists of a set of labels, which consists of a targeted label $t \in A$ (i.e., the first name of the targeted person) and more than one context labels (e.g., the names or labels of the targeted one's friends) provided by the users.

Given a labeled social network $G=(V, E)$ and a search task T consisting of a particular first name label to indicate the targeted person and a set of context labels $\{a_1,..., a_r\} \subseteq A$, the *context-based people search* problem is to return a ranking list of persons in which each retuned person connects to the query labels with minimum communication cost. Specifically, we aim to find a set of individuals $V' \subseteq V$ that is connected by a subgraph $G[V']$ such that (1) $T \subseteq \{X_i, i \in V'\}$ and (2) the communication cost $CC(V')$ is minimized. And for the ranking result, a ranked list is returned based on the communication cost.

3. THE APPROXIMATED METHOD: *ApxSteiner*

For the context-based people search in a labeled network, a simple approach is to exploit the Steiner tree algorithm. The minimum *Steiner tree problem* is to find a connected subgraph that connects some required nodes with minimum sum of edge weights. We can add the query labels as nodes in the labeled social network, and designate these nodes of query labels as the required ones in the Steiner tree problem. Using existing solutions to the minimum Steiner tree problem, a subgraph with minimum cost can be derived, and the individual with the targeted label in the resulting subgraph is reported as an answer person. Repeating such process up to N times and sort the returned answer persons based on their costs, a ranked list of N answer persons can be derived.

We first review the *Steiner tree problem*. Given an undirected graph with non-negative costs on edges, the nodes in this graph are separated to the *terminal/required vertex set T* and the *Steiner vertex set S*. The Steiner tree problem is to find the minimum-cost spanning tree in the given graph; the found tree must contain all nodes in the terminal set T and any subset of the Steiner nodes. Since the Steiner tree problem is a NP-hard problem, several approximation algorithms had been proposed. Here we modify the greedy algorithm in [6] to find the minimum Steiner tree. This algorithm starts by randomly selecting a node as a seed in the required (terminal) set T. Then at each step, a single terminal node v^* from T is incrementally added to the current solution V'. Each v^* is determined by the minimum distance to V', which contains nodes that have already been added to the solution set.

4. THE PROPOSED METHOD: *GrpSteiner*

We propose to perform the context-based people search by grouping nodes based on query context labels to provide effective and efficient search. Our method consists of seven steps. The first step is a *label-based node grouping* that collects individuals into groups according to each query label. The second step is to identify three kinds of *connectors* between groups. The connectors play key roles to construct an effective group graph. By exploiting the connectors, the third step is to construct the *group graph*, in which linkages capture the critical interactions between groups. To realize the group graph, the fourth step is to embed the topological information only relates to the query labels into the

relevance graph. The fifth step aims to construct the *label relevance graph* which integrates the social interactions with the query context labels to guide the search. To allow more effective and efficient search result, in the sixth step, we propose to find the *crucial label* as the seed node for the following Steiner tree algorithm. In the seventh step, by using the derived seed label and the identified connectors, we propose and apply the *Connector-Steiner tree algorithm* to find an *effective subgraph* of individual for the query labels. According to the targeted label and the communication cost on the effective subgraph, we perform a *weight enhancement* mechanism to return a *ranked list* of individuals who satisfy the targeted label.

4.2 Label-based Node Grouping

The first step is to group the individuals in the labeled social network according to the query labels. A *group*, with respect to one of the query labels, for example, a_i, is a connected subgraph in which each individual node contains at least a_i. We denote the group of a certain query label as *grp*. Note that for a query label a_i, it can have more than one group since there could exist multiple disconnected subgraphs belonging to a_i.

Figure 2 shows an example of grouping for the query labels a_1, a_2, and a_3, in which a_1 is the targeted label. Each group is surrounded by a dotted region. We can observe two groups correspond to label a_1. These two groups are separated components since they have no interactions. In addition, node *i* belongs to two groups because *i* contains both a_1 and a_2. Nodes *m* and *q* do not belong to any of groups because they contain none of the three query labels.

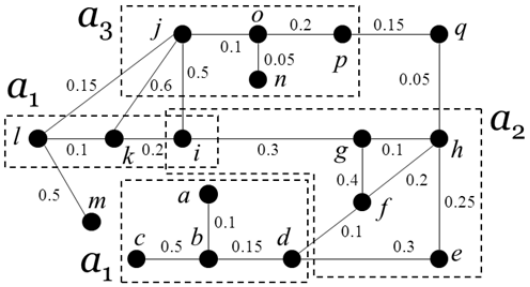


Figure 2. The label-based node grouping for the labeled social network, where there are three query labels a_1, a_2, and a_3 and a_1 is the targeted label specified by users.

4.3 Connector Identification

Based on the result of label-based node grouping, we can further distinguish different roles of individuals in a social network, especially for the *connectors* that take charge of connecting groups. According to the interaction behaviors between groups, there are three kinds of connectors: the overlap connector, the direct connector, and the indirect connector.

[**Definition 3**] (Overlap Connector) Given the label-based groups, if a node *v* belongs to more than one groups, it is called an *overlap connector*. The set of overlap connectors is denoted by *OC*.

In Figure 2, node *i* is the overlap connector since it contains multiple query labels, a_1 and a_2, and belong to multiple groups.

[**Definition 4**] (Direct Connector) Given the label-based groups, for each edge e=(u, v)$\in E$ in the social network G= (V, E), if $u \in grp_{ai}$, $v \in grp_{aj}$, and $a_i \neq a_j$, then u and v are the direct connectors of grp_{ai} and grp_{aj} respectively. The set of direct connectors in a network is denoted by *DC*.

Director connectors are those end nodes of a bridge edge which connects two different groups. If a node belongs to one group and directly link to another, it is direct connectors of that group. For example, in Figure 2, for the group of label a_3, its direct connector is node *j*. It directly links to groups of label a_1 and a_2 respectively. Note that although node *i* is an overlap connector, it is also a direct connector because it directly link to group of a_3.

[**Definition 5**] (Inter-Mediator) Given the label-based groups, if a node *v* belongs to no groups (contain no query labels), it is called an inter-mediator. The set of inter-mediators is denoted by *IM*.

For example, in Figure 2, node *q* is an inter-mediator because it does not contain any of query labels a_1, a_2, and a_3.

[**Definition 6**] (Indirect Connector) Given the label-based groups, for a path $p=(v_1,...,v_{n-1},v_n)$ (n>2) in the social network $G=(V,E)$, if $v_1 \in grp_{ai}$, $v_n \in grp_{aj}$, $a_i \neq a_j$ and $\forall j \in \{2,...,n\text{-}1\}$, $v_j \in IM$, then v_1 and v_n are the *indirect connectors* for grp_{ai} and grp_{aj} respectively. The set of indirect connectors in a network is denoted by *IC*.

Indirect connectors are those end nodes of a path (|path|>1) where nodes other than the end points do not contain any query labels. For example, in Figure 2, *p* and *h* are the indirect connectors for the groups of label a_3 and a_2 respectively.

4.4 Group Graph Construction

We have aggregated individuals into groups based on query labels. But the underlying interactions among groups have not yet been modeled. The group interactions are essential for finding effective communications of groups with respect to the query labels. If we can construct the *group graph* with lower costs to connect groups, it will guide the following search algorithm to have resulting subgraph with lower cost and cardinality. In this subsection, we exploit the benefits of the priority principle of connectors for constructing the group graph in an effective manner.

[**Definition 7**] (Group Graph) A *group graph* $H=(V_H, E_H)$ is a weighted graph constructed using query labels from the labeled social network $G=(V, E)$, where V_H is a finite set of *group nodes*, and $E_H \subseteq V_H \times V_H$ is a finite set of *group links*.

[**Definition 8**] (Group Node) *Group nodes* are defined according to query labels. For a query label $a_i \in T$, the group node $grp_{ai} \in V_H$ contains a set of nodes $V'(grp_{ai}) \subseteq V$ in G and must satisfy (1) $\forall u \in V'(grp_{ai})$, $a_i \subseteq X_u$, (2) the nodes in $V'(grp_{ai})$ form an induced connected subgraph $G[V'(grp_{ai})]$.

[**Definition 9**] (Group Link) A *group link* $e_H \in E_H$ is defined on two group nodes grp_{ai} and grp_{aj} in G_H, such that the corresponding induced subgraphs of these two group nodes, $G[V'(grp_{ai})]$ and $G[V'(grp_{aj})]$, are reachable from one another. Note that a group link between two induced subgraphs in G can be overlapped, direct, or indirect connection.

To have an effective group graph, we follow the priority of connectors, i.e., "overlap first, direct second, and indirect finally", to construct the group graph. The algorithm for group graph construction is shown in Algorithm 1. This algorithm starts by isolating all group nodes, and then constructs the group graph based on the priority of connectors in turn until the group graph is connected. For the example in Figure 2, the group graph is shown in Figure 3(a), where the circles represent the group nodes.

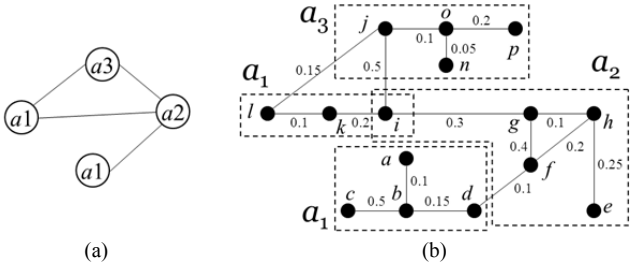


Figure 3. (a) The group graph constructed from Figure 2. (b) The relevance graph of Figure 3(a).

Algorithm 1. Group Graph Construction

Input: The labeled social network $G=(V,E)$; the set of all group nodes GRP; the set of groups that v_i involves R_{vi}; the sets of overlap, direct, and indirect connectors OC, DC, and IC.

Output: A connected group graph $H = (V_H, E_H)$.

$V_H \leftarrow V_H \cup \{v_H = grp_{ai} \in GRP\}$.
for each $v_i \in OC$ **do**
  $\{e_H\} \leftarrow (grp_x, grp_y)$, where $grp_x \in R_{vi}$, $grp_y \in R_{vi}$, and $x \neq y$.
  $E_H \leftarrow E_H \cup \{e_H\}$.
  if H is connected **do**: return H.
for each $e=(v_i,v_j) \in E$ and $v_i \in DC$ and $v_j \in DC$ **do**
  $\{e_H\} \leftarrow (grp_x, grp_y)$, where $grp_x \in R_{vi}$, $grp_y \in R_{vj}$, and $x \neq y$.
  $E_H \leftarrow E_H \cup \{e_H\}$.
  if H is connected **do**: return H.
$h \leftarrow 1$.
while (true**) do**
  for each h-hop path $(v_i,...,v_j)$ $v_i \in IC$ and $v_j \in IC$ **do**
    $\{e_H\} \leftarrow (grp_x, grp_y)$, where $grp_x \in R_{vi}$, $grp_y \in R_{vi}$, $x \neq y$.
    $E_H \leftarrow E_H \cup \{e_H\}$.
    if H is connected **do**: return H.
  $h \leftarrow h + 1$.

4.5 Relevance Graph Realization

We restore the corresponding graph topology in the original social network. We call this restored graph topology as the *relevance graph* G_R (i.e., relevant to query labels). In other words, we aim to obtain the induced graph of nodes contained in the group nodes. The restoration process from the group graph to the relevance graph consists of two steps. The first is to restore the group nodes. For each group node, we find the induced subgraph in the original social network corresponding to all nodes contained in it. The second is for the group links. For group links derived from overlap connectors, it is unnecessary to be processed since they have belonged to at least one group. For group links derived from direct connectors, there exists more than one bridge edge. For example, in Figure 2, there are three bridge edges between group nodes a_1 and a_3. We find the bridge edges with the lowest costs and add them into the relevance graph. For group links derived from indirect connectors, the h-hop paths found in Algorithm 1 are added into the relevance graph. For the example in Figure 3(a), the corresponding relevance graph is shown in Figure 3(b).

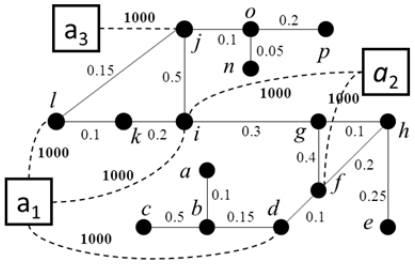


Figure 4. The label relevance graph for the previous example.

4.6 Label Relevance Graph Construction

To facilitate the people search considering query labels, we then embed these query labels into the relevance graph. By regarding each query label as a node, we add the query labels into the relevance graph and add edges to connect each query label node to nodes of connectors that possess that label. For these newly added edges between label nodes and connectors, we associate a positive weight higher than the sum of all weights in the social network. Eventually, a *label relevance graph* $G^{LR} = (V^{LR}, E^{LR})$ will be constructed. For example, the label relevance graph for Figure 3 is shown in Figure 4, where the squares represent the label nodes and the dotted lines represent the edges between label nodes and the corresponding connectors. In addition, we assign the new added edges a large weight. Note that the nodes corresponding to the query labels will serve as seeds to perform the search algorithm in the following subsection.

4.7 Seed Label Selection for Steiner Algorithm

To find the individuals with lower communication cost, our approach then finds the Steiner tree in the label relevance graph as the resulting context-based subgraph for people search. Given the label relevance graph $G^{LR} = (V^{LR}, E^{LR})$, the query labels as the terminal node set T, and the Steiner nodes are the nodes $V^{LR} \setminus T$, our goal is to find an effective subgraph $G^{LR}[U]$ with minimum communication cost $CC(U)$, where $U \subseteq V^{LR}$. The node with the targeted label in $G^{LR}[U]$ will be reported as a returned person.

We start the graph search by selecting a label node randomly from the label relevance graph. However, some experimental studies show that the seed node plays an important role for minimizing communication cost of the derived subgraph. If a good seed node is selected, the communication cost can be lowered. We propose the *connective degree* to measure the effectiveness of label nodes in the label relevance graph. The label node with the highest connective degree will be selected as the seed node to perform the greedy Steiner tree algorithm. We first define the ε-neighborhood of a label node.

[**Definition 10**] (ε-Neighborhood) The ε-*neighborhood* of a node $v \in V^{LR}$ is the set of nodes $N_\varepsilon(v)=\{u_i \mid 1 \le PathLength(v, u) \le \varepsilon, u \in V\}$.

[**Definition 11**] (Connective Degree) Given the label nodes in the label relevance graph and the three sets of connectors OC, DC, and IC, the *connective degree* of a label node v is defined as $CDegree(v)=|N_\varepsilon(v) \cap OC| \times \alpha_{OC} + |N_\varepsilon(v) \cap DC| \times \alpha_{DC} + |N_\varepsilon(v) \cap IC| \times \alpha_{IC}$, where the parameters α_{OC}, α_{DC}, and α_{IC} are used to control the effectiveness of different kinds of connectors. In the work, ε is set as 2, α_{OC} is set as 0.7, α_{DC} is set as 0.2, and α_{IC} is set as 0.1.

Algorithm 2. Connector-Steiner Tree Algorithm with Ranking.

Input: The graph $G^{LR} = (V^{LR}, E^{LR})$; a search task consists of a targeted label t and a set of context labels $T = \{a_1,...,a_r\}$; the number of people search results to be returned n.

Output: A ranked list of nodes $P=<(p_1,CC_1)...,(p_n,CC_n)>$.

for $k = 1$ to n **do**
  $U_k \leftarrow s$, where $s \in T$ and s is picked by *connective degree*.
  while $(T \setminus U_k) \neq \phi$ **do**
    $v^* \leftarrow \text{argmin}_{u \in T \setminus Uk}\ Dist(u, U_k)$ in G^{LR}.
    if $Path(v^*, U_k) \neq \phi$ **then**
      $U_k \leftarrow U_k \cup \{Path(v^*, U_k)\}$.
  $U_k \leftarrow U_k \setminus \{a_1, ..., a_r\}$.
  $p \leftarrow \{v_l \mid v_t \in U_k$ **and** $t \in X_l$ **and** $v_t \neq p_j, j=1,...,k\text{-}1\}$.
  $weight(e_t=(v_t, t)) \leftarrow weight(e_t=(v_t, t)) \times \text{LARGEVALUE}(=1000)$.
  $LIST \leftarrow (p, CC(U_k))$.
Sort $LIST$ according to $CC(U_k)$ and return as the ranked list of nodes $P=<(p_1,CC_1),...,(p_n,CC_n)>$, where $CC_1 < CC_2 <, ..., < CC_n$.

4.8 Connector-Steiner Algorithm with Ranking

Given the label relevance graph G^{LR}, and the set of query label nodes $\{a_1,...,a_r\}$ of the search task T, we propose the Connector-Steiner Tree algorithm to find a list of effective connection subgraph with respect to the query labels. We propose a *weight enhancement* mechanism, which increases the edge weights between each answer node and the targeted label nodes. Specifically, when each time an effective subgraph is constructed, we identify the node with the targeted label and regard it as one answer person. Then we replace the edge weight between the returned person and the targeted label node with a larger value. Such weight enhancement is capable of guiding the following Steiner tree searches to choose other persons with the targeted label as other answers. Each time when an answer person is returned, we also record the cost of the current effective subgraph

as the effectiveness of such answer. Eventually, we obtain a list of answer persons with their costs. By sorting such list based on the costs, a ranked list of effective subgraphs can be derived as the results. The detailed is shown in Algorithm 2.

Figure 5 shows the round-1 and round-2 effective subgraphs for the query search task $T=\{a_1,a_2,a_3\}$ and the targeted label is a_1, using the label relevance graph of Figure 4. For the round-1 result, the highlighted nodes $\{i, j, k, l\}$ and red edges $\{(i, k), (k, l), (l, j)\}$ compose the effective subgraph with the minimum cost 0.15+0.1+0.2=0.45. Node l that satisfies the targeted label a_1 is reported as the answer person with cost=0.45. Before the end of 1st round, the algorithm enhances the edge weight between the node l and the label node a_1 to be 1000×1000. For the round-2 result, the highlighted nodes $\{i, j\}$ with the green edge (i, j) compose the effective subgraph with cost=0.5. Node i satisfies the targeted label is reported as the answer person with cost=0.5. If we designate to have n=2 results, eventually, the ranking list will be <(l, 0.45), (i, 0.5)> (i.e., l is the rank-1, i is the rank-2).

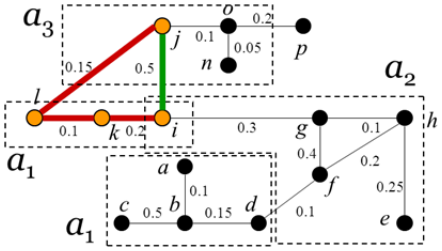


Figure 5. The round-1 and round-2 effective subgraphs derived by the proposed Connector-Steiner Tree algorithm is highlighted as the red and green ones respectively. Recall the targeted label is a_1. The round-1 answer person is node l with cost=0.45 while the round-2 answer is node i with cost=0.5.

5. EVALUATIONS

We conduct experiments to test the effectiveness and efficiency of the proposed method for tackling the context-based people search in a social network. By comparing our grouping-based method (**GrpSteiner**) to the original greedy Steiner tree algorithm (**ApxSteiner**), the experimental results demonstrate that our GrpSteiner performs as well as ApxSteiner in terms of ranking performance, and takes significantly less search time.

5.1 The DBLP Bibliography Data

We conduct the experiments on the DBLP bibliography database. The snapshot on December 31, 2010 of information retrieval and data mining related conferences (including KDD, ICDM, SDM, PAKDD, PKDD, ICML, CIKM, WWW, SIGIR, ACL, SIGMOD, VLDB, PODS, ICDE, EDBT, and ICDT) is used. We construct the labeled social network using co-authorship between authors. The set of labeled nodes consists of authors who have co-authored at least three papers. The label set X_i of each author i consists of his/her first names and keywords of textual terms (removed the stop words) occurring in the paper titles of at least three papers titles that he or she co-authored. Two authors are connected in the network if they co-authored at least three papers. There are a total of 10,443 authors and 25,354 edges. We compute the edge weights (i.e., communication costs) using the *Jaccard coefficient*. The formula of computing edge weights is given as: $w(i,j) = 1 - (|P_i \cap P_j|/|P_i \cup P_j|)$, where P_i is the set of papers of i.

5.2 Experiment Design

Evaluation Measures. The goals of the experiments are to evaluate the effectiveness and efficiency between the proposed grouping-based Steiner algorithm (GrpSteiner) and the original greedy Steiner tree algorithm (ApxSteiner). The effectiveness is measured by calculating the ranked position of the ground-truth targeted person in the returned ranking list. Specifically, for a query consisting of a set of context labels and the first name of the target person t_i, when a search method returns a ranking list of persons, we find the ranked position of the ground-truth targeted person, denoted by $rank(t_i)$. Given a set of N search queries, we compute the average value of ranked positions of targeted persons, $AvgRank=(\Sigma_{i=1...N} rank(t_i))/N$. On the other hand, the efficiency is measured by averaging the execution time (seconds) of search queries. Note that the ranked list of original Steiner algorithm (ApxSteiner) is generated by performing the algorithm n times and sorts the results by the communication costs, in which we apply the abovementioned *weight enhancement* mechanism when each resulting person is returned.

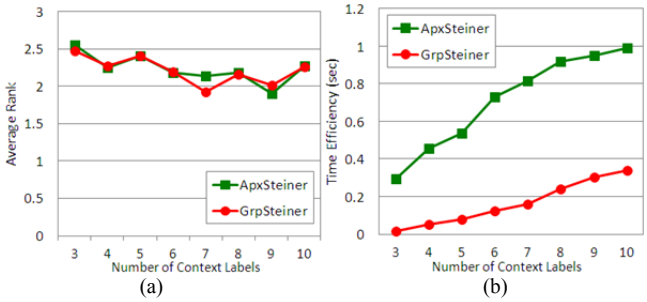
Generating Targeted Person with Context Labels. We simulate how users perform people search in a social network to generate a set of N the ground-truth targeted persons with the query sets of context labels. For each targeted person, we produce a set of $\boldsymbol{k}$ context labels for the search process. The generation procedure is given as the following. First, we sort the frequency of first names in the compiled DBLP co-authorship network, and derive a set of highly-frequent first names S_f by setting a first-name threshold τ_f. The same process is applied to derive a set of highly-frequent keywords S_w. using a keyword threshold τ_w. To impose the namesake effect which is what we mainly concern, we set (τ_F, τ_K) = (30, 250) in the experiments. Second, we randomly pick a first name from S_f, and then randomly pick a person i with this first name. We regard such individual i as a ground-truth targeted person and his/her first name as one of the query context labels. Third, using the social network, we perform Breath-first search up to r steps starting from t_i, and obtain a set of neighboring persons S_r. And then we randomly select $\boldsymbol{k}$-1 persons from S_r. For each selected person j, it has an equal probability to choose either his/her first name or one of his/her possessed keywords as a query context label. The neighboring step r is set to be 2 because in real-life cases it is less possible for users to employ far information about the targeted person for people search. Repeating the above second and third parts up to N times, we can attain N ground-truth targeted persons with the corresponding N $\boldsymbol{k}$-sized sets of context labels as the search queries. Here we set N to be 200.

Evaluation Plan. The experiments are divided into two parts to demonstrate the effectiveness and efficiency from diverse points of views. First, to investigate how the number of provided context information affects the performance, we vary the number of the query context labels $\boldsymbol{k}$ = 3, 4, ..., 10. Note that among $\boldsymbol{k}$ context labels, one is the first name of the ground-truth targeted person t_i and the other $\boldsymbol{k}$–1 ones are either the first names or the keywords in t_i's neighbors. Second, to examine how the ratio between the first-name and keyword context labels affects the performance, we fix $\boldsymbol{k}$=6 and vary the ratio of "*#NameLabel* versus *#KeywordLabel*" as 1:5, 2:4, ..., 5:1.

5.3 Experimental Results

Number of Context Labels. The results of varying the number of query context labels are shown in Figure 6(a) and 6(b) for the average rank and time efficiency respectively. For the average rank in Figure 6(a), we can find our GrpSteiner and the original ApxSteiner algorithms have very similar performance under different numbers of context labels. In details, which the number of query contexts are lower, both methods become slightly worse (i.e., average ranks are up to 2.5). As the numbers of query context labels increase, the average ranks generally decrease toward 2.0. This indicates both methods can averagely return the ground-truth targeted person within top-2 returned persons. On the

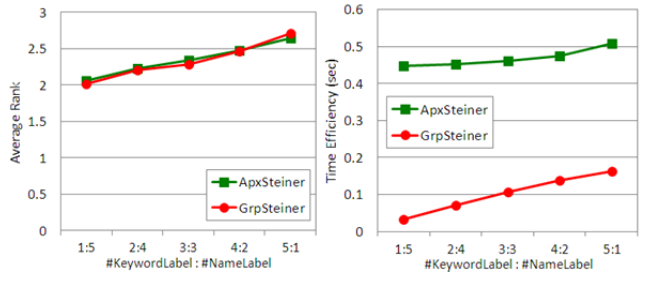
other hand, for the time efficiency of execution time (in seconds) in Figure 6(b), our GrpSteiner significantly outperforms the ApxSteiner method, especially when the number of query context labels gradually becomes higher. We believe such good efficiency is due to that the grouping mechanism successfully reduces the graph search space so that the execution of our Connector-Steiner Tree algorithm is more efficient to find the results.



(a) (b)

Figure 6. (a) Varying the number of query context labels (b) Varying the number of query context labels to show the average ranks of both our GrpSteiner and ApxSteiner.

#Keyword-#FirstName Ratio. We present the average rank and time efficiency by varying the ratio between #KeywordLabel and #FirstNameLabel. In Figure 7(a), we find the effectiveness of both GrpSteiner and ApxSteiner is well-performed and similar. Averagely both can find the ground-truth targeted person within around top-three returned persons. In more details, as the number of query keyword labels increases, the average ranks of both methods gradually raise from about 2.0 to 2.75, which indicates both needs more returns to catch the answers. We think such effect results from that the query with more keyword labels will produce higher potentials to allow those with the same keyword labels usually connect to one another in the relevance graph. Therefore the graph search, which aims to minimize the communication cost, will be guided toward different directions that are away from the targeted person. In contrast with confusing the search based on keyword labels, when the number of first name labels increases, the effectiveness becomes better. It is due to that those people with the same first names are usually hard connected to each other in the relevance graph, and thus are able to provide accurate guidance of graph search toward the targeted person. On the other hand, for the time efficiency in Figure 7(b), no matter what the ratio become, our GrpSteiner generally outperform the ApxSteiner. And, likewise, as the number of query keyword labels increases, more connections among those with the same keywords will make the search take more time.



(a) (b)

Figure 7. Varying the #KeywordLabel: #FirstNameLabel ratio to show (a) the average ranks. (b) the time efficiency.

6. RELATED WORKS

Existing work on social search originates from the IR field, and they focus on optimizing Web search using social content such as voting, tagging and bookmarking [2][12]. Since social annotations provided by the public are usually good summaries for web pages, some ranking methods, such as *SocialPageRank* [2] and *HubRank* [3], have been proposed to enhance the search quality. In addition, some works [1][4][9][10][11] categorize the annotations into diverse types of entities and exploit the relationships among these heterogeneous sources to perform multifaceted entity search. However, they neglect the underlying social network among the users. On the other hand, few works [5][7][8] describe social search methods to find individuals by considering the social relationships among people. Vieira et al. [8] modify the shortest path among individuals as a ranking function to recommend relevant friends for a single user. Their method cannot be used to perform people search based on querying users or other labels. Though Schendel et al. [7] retrieve relevant items satisfying a given set of user-given query tags in a social tagging network, the communication cost among these individuals is ignored, so their method cannot be applied to our context-based people search.

7. CONCLUSION

In this paper, we propose and formally define the problem of context-based people search in a labeled social network. Given a labeled social network and a set of query labels consists of a targeted label and other context labels, we aim to return a ranking list of persons who possess the targeted label and connects to other context labels with minimum communication costs through certain effective subgraph. Considering the namesake challenge, we propose to leverage context information with the potential interactions between the targeted person and such context labels to tackle the people search in a social network. Experiments on the DBLP bibliography data demonstrate the excellent quality of our found ranked results and take significantly less execution time comparing to a greedy approximation algorithm.

REFERENCES

[1] E. Amitay, D. Carmel, N. Har'El, S. Ofek-Koifman, A. Soffer, S. Yogev, and N. Golbandi. Social Search and Discovery Using a Unified Approach. *ACM HT 2009*, 199–208, 2009.

[2] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing Web Search Using Social Annotations. *WWW 2007*, 501–510.

[3] K. Chakrabarti, V. Ganti, J. Han, and D. Xin. Ranking Objects Based on Relationships. *SIGMOD 2006*, 371–382.

[4] J. Davitz, J. Yu, S. Basu, D. Gutelius, and A. Harris. iLink: Search and Routing in Social Networks. *KDD 2007*, 931–940.

[5] J. Kleinberg. Social Networks, Incentives, and Search. *SIGIR 2006*, 210–211.

[6] T. Lappas, K. Liu, and E. Terzi. Finding a Team of Experts in Social Networks. *KDD 2009*, 467–475.

[7] R. Schenkel, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J. X. Parreira, and G. Weikum. Efficient Top-k Querying over Social-Tagging Networks. *SIGIR 2008*, 523–530.

[8] M. V. Vieira, B. M. Fonseca, R. Damazio, P. B. Golgher, D. D. C. Reis, and B. Ribeiro-Neto. Efficient Search Ranking in Social Networks. *CIKM 2007*, 563–572.

[9] X. Wang, J. T. Sun, and Z. Chen. SHINE: Search Heterogeneous Interrelated Entities. *CIKM 2007*, 583–292.

[10] X. Wang, J. T. Sun, Z. Chen, and C. Zhai. Latent Semantic Analysis for Multiple-type Interrelated Data Objects. *SIGIR 2006*, 236–243.

[11] W. Xi, E. A. Fox, W. Fan, B. Zhang, Z. Chen, J. Yan, and D. Zhuang. SimFusion: Measuring Similarity Using Unified Relationship Matrix. *SIGIR 2005*, 130–137.

[12] Y. Yanbe, A. Jatowt, S. Nakamura, and K. Tanaka. Can Social Bookmarking Enhance Search in the Web? *JCDL 2007*, 107–116.

[13] Wink People Search. http://wink.com/

[14] Intelius People Search. http://search.intelius.com/