

What distinguish one from its peers in social networks?

Yi-Chen Lo · Jhao-Yin Li · Mi-Yen Yeh ·
Shou-De Lin · Jian Pei

Received: 4 February 2013 / Accepted: 24 June 2013 / Published online: 18 July 2013
© The Author(s) 2013

Abstract Being able to discover the uniqueness of an individual is a meaningful task in social network analysis. This paper proposes two novel problems in social network analysis: how to identify the uniqueness of a given query vertex, and how to identify a group of vertices that can mutually identify each other. We further propose intuitive yet effective methods to identify the uniqueness identification sets and the mutual identification groups of different properties. We further conduct an extensive experiment on both real and synthetic datasets to demonstrate the effectiveness of our model.

Responsible editor: Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, Filip Zelezny.

Y.-C. Lo · S.-D. Lin
Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
e-mail: d00922006@csie.ntu.edu.tw

S.-D. Lin
e-mail: sdlin@csie.ntu.edu.tw

J.-Y. Li · M.-Y. Yeh (✉)
Institute of Information Science, Academia Sinica, Taipei, Taiwan
e-mail: miyen@iis.sinica.edu.tw

J.-Y. Li
e-mail: louisjyli@iis.sinica.edu.tw

M.-Y. Yeh
Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan

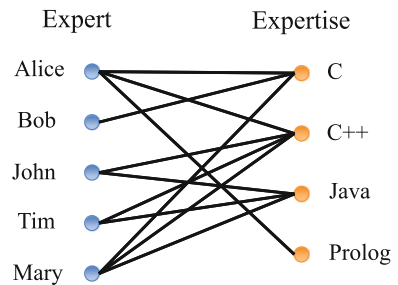
J. Pei
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
e-mail: jpei@cs.sfu.ca

Keywords Social query · Node identification · Social networks

1 Introduction

In a heterogeneous social network, each entity is assigned a type (or label) to describe its category. A node type can be a place, a person, an organization, etc. For example, consider an expertise bipartite network, where each node is an entity of either the type expert or expertise, as illustrated in Fig. 1. “Ego” is a node of focus while the ego-centric view of a node generally contains the neighborhood information of this node centralized as an ego. In this work, ego is considered as a query node of interests. It is often interesting and useful to distinguish an ego of some type from the rest of the entities of the same type. How can one expert be distinguished from another? In Fig. 1, Alice can be uniquely identified from all the other experts because she possesses the skill Prolog that nobody else does. Mary can be uniquely identified from the other experts using the set of expertise C, C++, and Java because she is the only person that owns all three skills. In other words, we call $\{C, C++, Java\}$ the *uniqueness identification set* of Mary.

Fig. 1 Example of a bipartite expertise network



Distinguishing a node from its peers in a social network is highly useful in many applications. In the above example, using the knowledge that Prolog uniquely identifies Alice from all the other experts, any consulting projects requiring the expertise on Prolog needs to involve Alice. To take the full advantage of Mary’s expertise, Mary should be assigned first to those projects requiring expertise on C, C++, and Java together. Similarly, we may also distinguish different expertise by finding a unique set of experts that have the interested expertise.

Distinguishing a node from its peers in a social network is far from trivial. Not every node can be uniquely identified *only* by its *one-degree* neighbors. For example, in Fig. 1, Tim and John cannot be distinguished from one another using only their expertise, since their expertise sets, $\{C++, Java\}$, are identical. To identify John, we have to use John’s 2-hop neighbors including C++, Java, and Tim. The identification set for John essentially indicates, “Besides Tim, John is the only person who knows C++ and Java”. Such information about John is useful—it provides a unique expertise set of John as well as his alternatives in the network.

Additionally, we may use the identification information to find interesting “communities” in a social network. For example, consider the sub-graph that contains the

5 nodes John, Tim, Mary, C++, and Java. Each of these nodes can be identified by the rest of the nodes (or a subset of them). For instance, the node C++ can be identified by the set Tim, Java, which can be read as “C++ is the only language other than Java that Tim knows”. Similarly, the node Java can be identified by the sets {Mary, C++} and {John, Mary, C++}. In other words, the induced graph on {John, Tim, Mary, C++, Java} is a closure in identification, which we call a *mutual identification group*. Intuitively, in a mutual identification group (MUID) S , each member can be identified by some other members of S . Such a closure group discloses some interesting information—John, Mary and Tim may replace each other as they are the only experts who master both C++ and Java.

Essentially, by designating any node as an ego, we can use its identification sets and MUID to conduct ego-centric analysis. Such analysis enables valuable applications including social entity search engines and substitution recommendation systems. For a social entity search engine, existing work such as (Tang et al. 2008; Zhu et al. 2009) did not emphasize on choosing neighboring nodes to display, instead they opted to show all the neighbors of the ego up to certain degree. In contrast, our goal is to leverage the social network itself and report the uniqueness identification set for a query entity/ego to highlight its unique information, which helps users quickly catch the main differences among this entity and others of the same type from a large amount of redundant information. Such a mechanism facilitates better visualization and summarization for the task of social relationship search of an entity. Moreover, with the MUID of each query entity/ego, we can build a substitution recommendation system that finds an alternative for a query entity that is out of stock or unavailable. That is, items that can be mutually identified by the same set of objects can be regarded as serving similar roles in a network, and therefore can be regarded as a surrogate for each other. Consider a movie network as an example, which contains relationship between movies, actors, directors, etc. For a movie, an entity search engine can report its unique identification set containing the minimal set of information of its actors, director, or places that distinguish this movie from others, rather than simply shows its neighborhood graph. Moreover, a mutual identification group may capture a set of movies, actors and places that are closely related to each other. If some movie is sold out, the substitution system may recommend other movies in the same uniqueness identification group. We will show several cases in Sect. 6.3.

As astute readers might point out, we may find more than one identification set for each ego we are interested in a network. For example, to identify Mary we may choose its neighbor Java. We then find that both Tim and John have the same expertise and thus are structure equivalent to Mary given their mutual neighbor Java. The final identification set of Mary includes Tim, John, and Java. On the other hand, we may also choose both C and C++ to distinguish Mary from other experts. This time, only Alice has both expertise and the resulting uniqueness identification set of Mary is {Alice, C, C++}. Compared to the former uniqueness identification set of Mary, the latter one is preferred since fewer experts are involved and the uniqueness of Mary possessing the combination of expertise C and C++ is highlighted. Similar concept of using as few entities as possible when choosing the MUID should be followed.

Motivated by the above identification needs, in this paper, we propose a novel ego-centric data mining task as follows. In a network where each node *is associated with*

a *unique type*, we want to achieve the following two goals: (1) Given an ego, find a uniqueness identification set (UID) that distinguishes the ego from its peers, *which are nodes of the same type*. As there can be more than one uniqueness identification set of an ego, we aim to find one containing minimum number of structurally equivalent nodes to distinguish the ego from others of the same type. When two UIDs have the same number of structurally equivalent nodes, we choose the one that can be identified with a smaller number of neighbors; (2) Given an ego, find a MUID where the uniqueness of each node in the group can be identified by the rest of the nodes, or a subset of them. Similarly, when more than one MUID is available, we aim to find the one with fewer structurally equivalent neighbor nodes to distinguish the nodes. We explore different evaluation criteria for the two tasks.

There are four main contributions in this paper. First, we define two novel data mining problems for social network analysis, namely mining uniqueness identification sets and mutual identification groups. Second, we introduce a series of simple yet effective methods, *1-Hop+*, *One-Neighbor*, and *Multiple-Neighbor*, to tackle the problems from different angles. We prove that our methods are guaranteed to identify a node set that can identify the uniqueness of the query node. Third, we evaluate the three methods systematically on three real data sets and three synthetic data sets. The results and follow up analysis show the advantages of these methods. Finally, we analyze some of the interesting MUID from a real-world movie dataset, and provide explanation about them. The analysis justifies the usefulness of our approach.

The rest of the paper is organized as follows. We review the related work briefly in Sect. 2. We define the problems of uniqueness identification set of a vertex in a network and the MUID in Sect. 3. We present three effective methods to solve the corresponding problems in Sect. 4. We report the experiment results in Sect. 5 and conclude the paper in Sect. 6.

2 Related works

Our work is related to the existing studies on social networks in three aspects: social network search/extraction, community queries, and social network anonymization.

The work on social entity search/extraction focuses on extracting social relationship among a specific set of people from available open resources such as the Web. For example, (Zhu et al. 2009) developed an entity relationship extraction framework for relation extraction from the Web data, and a search engine, Renlifang, was built to report the entity relationship graph of some query persons, locations, or organizations. Tang et al. (2008) developed Arnetminer, an academic search system that aims to automatically extract the researcher profile including the co-authorship relation graph from the Web. Those studies focus on extracting social networks. In contrast to the above works, we do not rely on parsing and extracting the information from external sources, but focus on extracting the MUID from a given social network.

The works on group or community queries, (Sozio and Gionis 2010; Lappas et al. 2009; Li and Shan 2010; Yang et al. 2011), focuses on selecting a set of nodes or searching a specific community according to the given query nodes or some constraints. Given a set of query nodes in a graph, Sozio and Gionis (2010) searched

a community containing the nodes by finding a densely connected sub-graph. They proposed a greedy algorithm with heuristics to find the optimum solution under the monotone constraints on the density measure. Lappas et al. (2009) tackled a team formation problem, which seeks a group of suitable people (each as a node in a graph) with different skills (as attributes of each node) to complete a task with certain skill requirement. The output results are determined by minimizing two types of communication cost between the selected people. Li and Shan (2010) further generalized the problem in Lappas et al. (2009) by associating each required skill with a specific number of experts. They proposed a density-based measure for selecting the seed node and a grouping-based approach to find the team for generalized tasks. Similar to the previous two works, given a query initiator, Yang et al. (2011) found a group of people that are available to attend activities while satisfying the acquaintanceship and social-temporal constraints. The problem was formulated as integer linear programming problem. With two efficient algorithms proposed to find the optimal solution. Based on a specified ego node, Li and Lin (2009) reported an egocentric abstraction graph to summarize the features of the given ego node using an unsupervised learning mechanism. Although all of the above works extract a query-based social graph, their goals are very different from ours, which is finding a group of nodes that can uniquely identify the query node.

Our work is also related to the problem of graph anonymization. Specifically, the *k*-anonymization of social networks Zhou and Pei (2008) alters a given social network such that the 1-neighborhood of each node is isomorphic to those of at least $k - 1$ other nodes. To achieve the goal, the critical step is to determine whether the 1-neighborhood of a node can identify the node with probability higher than $1/k$. However, in *k*-anonymization, the search is constrained to only 1-neighborhood, while our work does not have such constraint. Moreover, the goals in social network anonymization and ours are fundamentally different.

To our knowledge, we are the first to identify and tackle the problem of finding identification sets and mutual identification groups. Our work is clearly different from community detection (Fortunato 2010; Newman 2004) in a social network, which aims to divide an entire social network into a set of disjoint or overlapping partitions.

3 Preliminaries

In Sect. 3.1, we provide the formal definitions of the problems. We then prove that the problem of finding the optimal UID is NP-hard.

3.1 Problem definition

We are given an undirected, simple, and labeled graph $G(V, E)$ and a query node $v \in V$ whose 1st and 2nd order neighbor set is denoted as $N(v)$ and $N_2(v)$, respectively. Each vertex in G is labeled with a type denoted by $type(v)$.

Table 1 summarizes the symbols and terminology used in this paper.

Definition 1 In a graph $G(V, E)$, given a vertex $v \in V$ and a set of nodes $M \subseteq N(v)$, a node $u \in V$ is said to be *structure equivalent* (SE) to v given M , denoted by

Table 1 Symbols and terminology

Symbol	Definition
$G = (V, E)$	A graph consists of vertices V and edges E
$N(v)$	The first order neighbor set of vertex v
$N_2(v)$	The second order neighbor set of vertex v
$type(v)$	The vertex type of v
UI	Uniqueness identification. See Definition 2
UID	Uniqueness identification set. See Definition 1
$MUID$	Mutual identification group. See Definition 3
$SE(v, M)$	The structure equivalent set of node v given a set M . See Definition 1

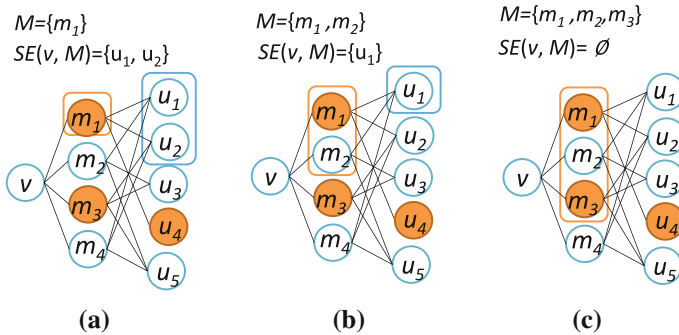


Fig. 2 Three examples to show $SE(v, M)$. **a** u_1 and u_2 are structure equivalent (SE) to v given $M = \{m_1\}$. **b** By adding m_2 into M , u_2 becomes not SE to v since u_2 has no link with m_2 . **c** Similarly by adding m_3 into M , $SE(v, M)$ becomes empty set. Note that here we assume two types of nodes: ones with filled color, one without

$u \in SE(v, M)$, if $type(u) = type(v)$ and $M \subseteq N(u)$. $SE(v, M)$ is the set of nodes structure equivalent to v given M . Please note that $SE(v, M) = \emptyset$ if there does not exist any node structure equivalent to v given M .

An example is shown in Fig. 2. In Fig. 2a, suppose $M = \{m_1\}$, then $SE(v, M) = \{u_1, u_2\}$. Note that u_4 is not included in $SE(v, M)$ because it is not the same type as v . Figure 2b, assuming an additional vertex m_2 is added into M (note that the nodes in M do not need to be of the same type), we can find that the set $SE(v, M)$ becomes smaller since $u_2 \notin N(m_2)$ and has to be removed from $SE(v, M)$. This example shows that adding nodes into M may decrease the number of nodes in $SE(v, M)$. In Fig. 2c, when $M = \{m_1, m_2, m_3\}$, no vertex is connected to every element in M . Therefore, $SE(v, M) = \emptyset$.

Definition 2 Given a vertex v and a non-empty set $M \subseteq N(v)$, we define that v 's uniqueness can be identified by the 2-tuples set $[M, SE(v, M)]$. This set is called a UID of v .

Note that $SE(v, M)$ can be empty as it is possible to interpret such situation as “ v is unique because there is no other vertex that connects to M as v does”. When

$SE(v, M)$ is non-empty, then the uniqueness of v can be interpreted as “ v is unique because, besides the vertices in $SE(v, M)$, the only vertex that connects to M is v ”. Next we introduce an interesting and useful property of UIDs.

Property 1 *Given a query vertex v , its UIDs can always be found within two-hops of v .*

Proof Since $M \subseteq N(v)$, according to Definition 1, $\forall u \in SE(v, M)$, $M \subseteq N(u)$. Therefore, $SE(v, M) \subseteq N_2(v)$. That is, any UID, $[M, SE(v, M)]$, is within two-hops of v . \square

By Definition 2, there are many possible UIDs given a query vertex v . Therefore, we define a function to compare the quality of UIDs.

Definition 3 Given two UIDs, $D_1 = [M_1, SE(v, M_1)]$ and $D_2 = [M_2, SE(v, M_2)]$, we define a comparison function $Q(v, M_1, M_2)$ as follows.

- (a) $Q(v, D_1, D_2) = 1$, i.e., the quality of D_1 is better than that of D_2 , if (i) $|SE(v, M_1)| < |SE(v, M_2)|$ or (ii) $|SE(v, M_1)| = |SE(v, M_2)|$ and $|M_1| < |M_2|$.
- (b) $Q(v, D_1, D_2) = 0$, i.e., the quality of D_1 equals to that of D_2 , if $|SE(v, M_1)| = |SE(v, M_2)|$ and $|M_1| = |M_2|$.
- (c) Otherwise, $Q(v, D_1, D_2) = -1$, i.e., the quality of D_1 is worse than that of D_2 .

Note that $Q(v, D_1, D_2) = -Q(v, D_2, D_1)$. Furthermore, Q is transitive as when $Q(v, D_1, D_2) = 1$ and $Q(v, D_2, D_3) = 1$, $Q(v, D_1, D_3) = 1$.

In Definition 3, the UIDs with a smaller $|SE(v, M)|$ is preferred. This reflects the intuition that having less structurally equivalent nodes in the UID indicates a more unique vertex v . Therefore our primary goal is to minimize $|SE(v, M)|$ of the identified UIDs. We treat $|M|$ as a secondary criterion because using smaller M to obtain the same $SE(v, M)$ implies that the query vertex can be identified with fewer critical neighbors.

In the three UIDs shown in Fig. 2, the quality of the UID in Fig. 2c is the best since there is no SE node for the query vertex. The UID of the query vertex in Fig. 2a is the least unique of the three.

We then further introduce a problem of finding MUID given a query vertex. An MUID is a set of vertices where each vertex can be uniquely identified by the subset of the remaining vertices in the set.

Definition 4 In a graph $G(V, E)$, given a vertex $v \in V$, a set of vertices $X \subseteq V$ is a MUID of v if the following two conditions hold.

- (a) $v \in X$.
- (b) $\forall u \in X, \exists D \subseteq X, D$ is a UID of v

The quality measure of MUIDs is conceptually similar to UIDs. For each MUID, the primary goal is to minimize $|SE(v, M)|$ and secondary goal is to minimize $|M|$ of the UIDs.

For the reason that each vertex does have a UID, it is less straightforward to define the quality of MUID. Again, we still prefer a smaller $SE(v, M)$ size than the M size. To compare the $|SE(v, M)|$ of MUIDs generated from different models, we propose two different metrics. The first metric uses the union of the SE set of each node in the MUID, while the second metric evaluates the summation of the size of the SE set of each node in the MUID. The same rule applies to M as the secondary criteria. The union measure evaluates how the vertices in the MUID exploit other members to identify themselves. If the nodes tend to include the same set of vertices to identify themselves, the union size would be smaller, creating better MUID. The sum of size of SE tells us whether we only need a small set of vertices to uniquely identify each individual member, regardless whether those vertices are overlapped or not.

Then, we can formally define the metric for MUID: Given an MUID X , for all $v \in X$, the UID $D_v = [M_v, SE(v, M_v)]$. For the metric of union size, we define size of union of SE set (USE) and size union of M set (UM):

$$USE = \left| \bigcup_{v \in X} SE(v, M_v) \right|, \text{ and}$$

$$UM = \left| \bigcup_{v \in X} M_v \right|.$$

We are now able to compare the quality of two MUIDs by replacing $|SE(v, M)|$ and $|M|$ in Definition 3 with USE and UM . Similarly, for the metric of sum of size, we define total size of SE set (TSE) and total size of M set (TM):

$$TSE = \sum_{v \in X} |SE(v, M_v)|, \text{ and}$$

$$TM = \sum_{v \in X} |M_v|.$$

We have considered to use other quality measure of UID such as comparing $|M| + |SE(v, M)|$, i.e. do not differentiate the importance of $|M|$ and $|SE(v, M)|$ but just compare the UID size. Under this measure, the three examples in Fig. 2 has the same quality which is against intuition because case (c) clearly identify v from others. Minimizing the above criteria while failed to minimize $|SE(v, M)|$ (i.e. due to a minimum $|M|$), and we will leave the ego with a set of structure equivalent nodes, which does not make it “unique”. For the same reason, another alternative measure to put more emphasis on minimizing size of $|M|$ than $|SE(v, M)|$ cannot highlight the uniqueness of vertices. The concept of priority of SE set and M set can be also applied to MUID. We have also considered other measures such as measuring the number of type included in UID or MUID found. For example, when forming a team, including more types means including more experts that know different skills. One of our future work is to modify the current proposed models to optimize such objective function.

3.2 Complexity analysis

In this section we prove that finding the optimal UID with a minimal M under the condition of a minimal $SE(v, M)$ is NP-hard. We first prove that increasing M can only decrease the size of $SE(v, M)$ in Theorem 1. Based on this theorem, we can reduce the *set covering optimization problem*, which is known as an NP-hard problem, to our problem of finding the optimal UID.

We first observe an interesting property that adding more nodes into an existing M set will cause some nodes in $SE(v, M)$ being removed from the set. Therefore, $|SE(v, M)|$ can only remain the same or become smaller, it can never become larger when more nodes are added into M .

Theorem 1 *Given a query vertex v and $M' \supseteq M$, then $SE(v, M') \subseteq SE(v, M)$.*

Proof By Definition 1, since for all $u' \in SE(v, M')$, we have:

$$\begin{aligned} u' &\in \bigcap_{m \in M'} N(m) \\ &= [(\bigcap_{m \in M} N(m)) \cap (\bigcap_{m \in M' - M} N(m))] \\ &\subseteq (\bigcap_{m \in M} N(m)) \\ &= SE(v, M). \end{aligned}$$

Therefore, for all $u' \in SE(v, M')$, $u' \in SE(v, M)$ is proved, i.e., $SE(v, M') \subseteq SE(v, M)$. □

This theorem implies that if $M = N(v)$, $SE(v, M)$ would be the minimal SE set since any other $|M'| < |M|$ can only produce $SE(v, M') \supseteq SE(v, M)$. Therefore, we can conclude that $SE(v, M_{max} = N(v))$ is the minimal SE set because $N(v)$ is the largest possible set for M . Next, we want to show that the *set covering optimization problem*, which is known as an NP-hard problem, can be reduced to the problem of finding optimal UID with the minimal size of M to obtain the optimal $SE(v, M)$.

Theorem 2 *Given a query vertex v , with the neighbor set $N(v) = m_1, m_2, \dots, m_y$. The optimization of UID first tries to find a minimum SE set, SE_{min} , then given the condition tries to find a minimum M set such that $M \subseteq N(v)$ and $SE(v, M) = SE_{min}$. This problem is NP-hard.*

Proof The set covering optimization Problem is given a universe set $U = u_1, u_2, \dots, u_N$ and a set of subsets of U , $S = S_1, S_2, \dots, S_k$ where $s_i \subseteq U, i \in [1, k]$, to find a subset $S_{opt} \subseteq S$ that $\bigcup_{s \in S_{opt}} s = U$.

For the problem of finding optimal UID, Theorem 1 tells us that each time when we add a node into M , some nodes in $T = N_2(v) - SE_{min}$ can be removed from the eventual SE set where $SE_{min} = SE(v, N(v))$ and the elements in it are not removable. Therefore, for each $m = m_i$, we can identify a subset $T_i = T - SE(v, m_i)$. We can say

Algorithm 1: Exhaustive search

Input: A Graph $G = (V, E)$, a query vertex v
Output: UID set of v

- 1 Get $N(v) = \{m_1, m_2, \dots, m_d\}$;
- 2 $d \leftarrow |N(v)|$
- 3 $M \leftarrow 2^d$ combination set of $N(v)$ // $M(i)$ is the i^{th} element M ;
- 4 $optM \leftarrow M(0)$
- 5 **for** $i = 1$ to $2^d - 1$ **do**
- 6 **if** $Q(v, M(i), optM) = 1$ **then**
- 7 $optM \leftarrow M(i)$
- 8 **end**
- 9 **end**
- 10 $optUID \leftarrow [optM, SE(optM, v)]$
- 11 Output $optUID$;

that for each m_i , there is a corresponding set T_i representing nodes that can be excluded from the potential SE set, T , when m_i is added into M . Note that in our problem the goal is to find a minimum set $M = M_{min}$ such that $SE(v, M_{min}) = SE_{min}$. In other words, we want to find a minimum M set which can remove all nodes in T to find the minimal SE set.

To reduce the set covering optimization to finding optimal UID, we first create a ego vertex v , creating a neighbor m_i to v for each $S_i \subseteq S$, and then create a vertex t_j for each $u_j \in U$. Finally for each m_i , we link m_i with t_j if $u_j \in U - S_i$. That is, the set U to be covered in the Set Covering Optimization Problem can be mapped to the removable SE set, T , in the finding optimal UID problem and the covering subsets, S_i , can be mapped to the identifying set of neighbors, T_i . Since by generating graph and finding the optimal UID in this way, the Set Covering Optimization Problem, which known as an NP-hard problem, can be also solved, the problem of finding optimal UID is proven to be NP-hard. □

Here we suspect the MUID problem is also NP-hard because in MUID all vertices need to be uniquely identified by the rest. However, we will leave the detailed theoretical analysis of it to the future work.

4 Finding optimal UIDs

In this section, we propose an exact and several greedy methods to find the uniquely identify set that satisfies a desired criterion. Since finding an optimal UID is an NP-hard problem, here we propose an exhaustive search method to find the exact outcome and some heuristic-based methods to identify the sub-optimal result for more efficiency. In Algorithm 1, an exhaustive search algorithm that requires $O(2^d)$ time complexity is provided which identifies the optimal UID, where d is the degree of the query node v . It basically tries all the combination of M . Note that this method is not very efficient for nodes with high degree. For example, in our real world datasets there are a few nodes with hundreds or even higher degree, which makes running the exhaustive algorithm impractical.

Fig. 3 The example graph to find UID given v

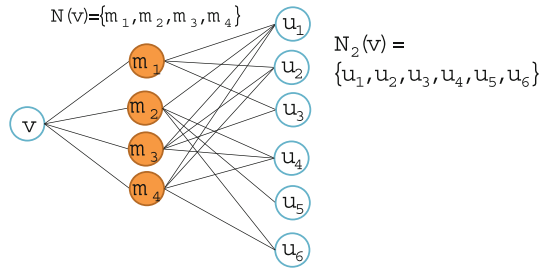


Fig. 4 The UID of v found by the 1-Hop+ method

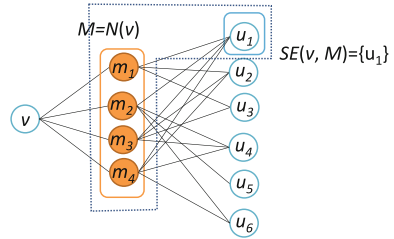
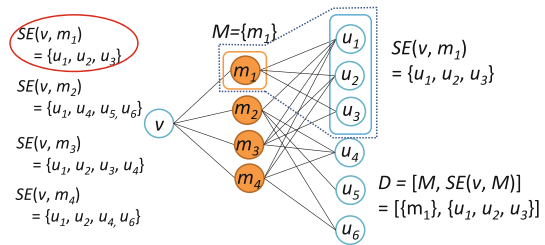


Fig. 5 The UID of v found by the $One\text{-Neighbor}$ method



To improve the efficiency, we propose the following three greedy methods, 1-Hop+ , $One\text{-Neighbor}$, and $Multiple\text{-Neighbor}$, to find sub-optimal UID sets faster. 1-Hop+ and $One\text{-Neighbor}$ are naïve baselines to be compared with $Multiple\text{-Neighbor}$. We will use the example graph of query v in Fig. 3 for demonstration.

In the 1-Hop+ method, the algorithm first includes all neighbors of v as M , and then adds $SE(v, M)$ into the UID set. This method extracts all the one-degree neighbors of v with their SE nodes. Figure 4 shows the UID m_1, m_2, m_3, m_4, u_1 with u_1 being the only node that connects to all the nodes in M . Note that this method guarantees to find the minimal SE set; however, it may produce the largest M set.

In the $One\text{-Neighbor}$ method, given a vertex v , we choose a neighbor $m \in N(v)$ and then obtain $SE(v, M = \{m\})$ to be included in the UID. Since the goal is to minimize the SE size, we observe the size of SE set produced by different m and choose the minimal one. In Fig. 5, m_1 is chosen and it produces $SE(v, m_1) = \{u_1, u_2, u_3\}$

The two algorithms have low time complexity of $O(d)$ where d is the degree of query node v . The 1-Hop+ method produces the minimal SE set but the maximal M . The $One\text{-Neighbor}$ method keeps the minimal size of M , but in general does not minimize the size of the SE set. To address the above concerns, Algorithm 2 introduces the $Multiple\text{-Neighbor}$ method whose goal is to identify a minimal M to minimize

SE set. In this method, we greedily add neighbor nodes of v into M , guaranteeing the optimal SE set and hoping to obtain a minimal M .

With the greedy heuristic to pick neighbors of v into M , SE_{gain} is defined to represent “the vertices to be removed from $SE(v, M)$ after adding a node into M ”.

Definition 5 Given UID of a query vertex v as $D = [M, SE(v, M)]$, and for $M' = M \cup \{n\}$, where $n \in N(v) - M$, then $SE_{gain}(v, n, M) = SE(v, M) - SE(v, M')$.

In Theorem 3 we prove that the set size of $SE_{gain}(v, n, M)$ is monotonic with the size of M . In this property, $\forall u \in SE(v, N(v))$, u cannot be in $SE_{gain}(v, n, M)$ for any v, n, M since there is no larger subset $M \subseteq N(v)$ than $N(v)$ itself.

Theorem 3 Given $SE(v, M), SE(v, M'), n \in N(v), n \notin M$ and $n \notin M'$, if $M' \supseteq M$ then $SE_{gain}(v, n, M') \subseteq SE_{gain}(v, n, M)$.

Proof By Theorem 1, we have the following derivation.

- (a) $\because M' \supseteq M, \therefore SE(v, M') \subseteq SE(v, M)$, and
- (b) $\because M' \cup \{n\} \supseteq M \cup \{n\}, \therefore SE(v, M' \cup \{n\}) \subseteq SE(v, M \cup \{n\})$.

From (a) and (b), we have

$$\begin{aligned}
 SE_{gain}(v, n, M') &= SE(v, M') - SE(v, M' \cup \{n\}) \\
 &\subseteq SE(v, M) - SE(v, M' \cup \{n\}) \\
 &\subseteq SE(v, M) - SE(v, M \cup \{n\}) \\
 &= SE_{gain}(v, n, M).
 \end{aligned}$$

□

The *Multiple-Neighbor* method starts from the state of $M = \phi$ and $SE(v, M) = N_2(v)$ given the query vertex v . It iteratively calculates the $SE_{gain}(v, n, M)$ of each neighbor n that is not included in M , adding the neighbor with the largest $|SE_{gain}(v, n, M)|$ into M and removing the vertices in $SE_{gain}(v, n, M)$ from the current $SE(v, M)$. It ends when there is no neighbor vertex with $|SE_{gain}(v, n, M)| > 0$. This method guarantees that the result UID has a minimal SE set.

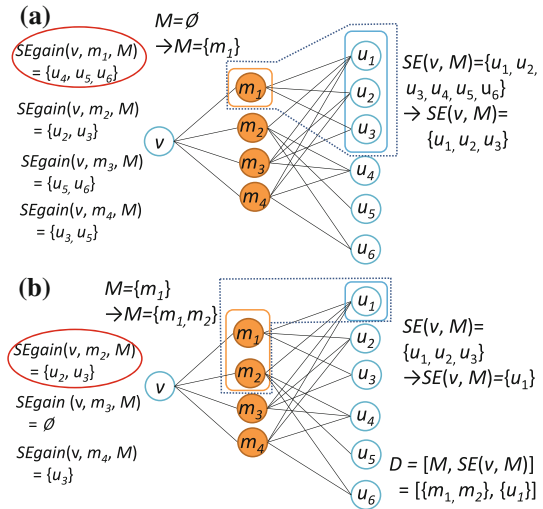
Theorem 4 The UID of query vertex v found by the *Multiple-Neighbor* method is guaranteed with minimal SE set, i.e., if there is no neighbor $n \in N(v) - M$ satisfying $|SE_{gain}(v, n, M)| > 0$, then $SE(v, M)$ is minimal.

Proof Let $SE_{min} = SE(v, N(v))$ be the minimal SE set and let the current state of the *Multiple-Neighbor* method be $SE(v, M_v)$. Suppose that $SE(v, M_v)$ is not minimal, then the following two conditions must both hold.

- (a) $\exists u$ such that $u \notin SE_{min}$ and $u \in SE(v, M_v)$, and
- (b) $\neg \exists n \in N(v) - M_v$ such that $SE_{gain}(v, n, M_v) > 0$.

Since $u \notin SE_{min}$, there exists $n \in N(v) - M$ such that $n \notin N(u)$. If we add n into M_v , then $u \notin SE(v, M_v \cup n)$. Because $u \in SE(v, M_v)$ and $u \notin SE(v, M_v \cup n)$, $u \in SE(v, M_v) - SE(v, M_v \cup n) = SE_{gain}(v, n, M_v)$.

Fig. 6 The UID of v found by the *Multiple-Neighbor* method **a** m_1 is chosen as the first vertex to be added into M **b** m_2 is chosen as the second vertex to be added into M



Algorithm 2: The Multiple-Neighbor method

```

Input: A Graph  $G = (V, E)$ , a query vertex  $v$ 
Output: UID  $D$  of  $v$ 
1  $SE \leftarrow N_2(v)$ 
2  $M \leftarrow \phi$ 
3 while  $|SE| > 0$  do
4    $m_{max} \leftarrow \text{argmax}_m |SEgain(v, m, M)|, m \in N(v)$ 
5   if  $m_{max} \leq 0$  and  $|M| \geq 1$  then
6     break;
7   end
8    $M \leftarrow M \cup m_{max}$ 
9    $SE \leftarrow SE - N(m_{max})$ 
10 end
11  $D \leftarrow [M, SE]$ 
12 output  $D;$ 
    
```

Figure 6 shows how the *Multiple-Neighbor* method is applied to the same example graph. In Fig. 6a, $SEgain(v, m_x, M)$ is generated for m_1, m_2, m_3 and m_4 , while m_1 is selected to be added into M because it has the largest $SEgain$. $SEgain(v, m_1, M) = \{u_4, u_5, u_6\}$ is then removed from $SE(v, M)$. In Fig. 6b, $SEgain$ of the rest of the neighbors are updated and m_2 is then added. After m_2 is added, it leaves only u_1 in $SE(v, M)$ and the algorithm halts because the condition $|SEgain(v, x, M)| > 0$ cannot be satisfied by adding any neighbor.

5 Finding MUIDs

In Sect. 4, we introduced four methods to find the UID given a vertex v . We then extend the three algorithms proposed in Sect. 4, except exhaustive search, to identify the MUID given a vertex v .

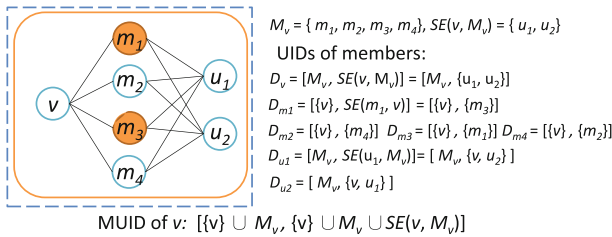


Fig. 7 The MUID of v found by the 1 -Hop+ by redefining M and SE set

Note that in MUID, not only the uniqueness of the query nodes but also that of the rest of the nodes in the set has to be uniquely identified. That is to say, after adding nodes into the set to uniquely identify the query vertex, one would then need to make sure the introduced nodes can be uniquely identified by adding more nodes. In the worst case, one would have to include every node in the graph into the MUID. The quality of MUID is measured by the average quality of UID of each vertex in the MUID. Note that the quality of UID is the same as the one defined in Definition 3.

For the 1 -Hop+ method, we first prove that the UID found given a vertex v can be an MUID for v by redefining M and SE set.

Theorem 5 Given a vertex v , and the UID identified by the 1 -Hop+ method $D = [M_v, SE(v, M_v)]$, $\forall u \in D$ we can find a UID $D_u = [M_u, SE(u, M_u)]$ such that $M_u \subseteq \{v\} \cup M_v$ and $SE(u, M_u) \subseteq \{v\} \cup M_v \cup SE(v, M_v)$.

Proof We have to show that for each $m \in M_v$ we can find UID $D_m = [M_m, SE(m, M_m)]$ such that $M_m \subseteq \{v\} \cup M_v$ and $SE(m, M_m) \subseteq \{v\} \cup M_v$, and the same for $u \in SE(v, M_v)$. This can be proved by the following two statements.

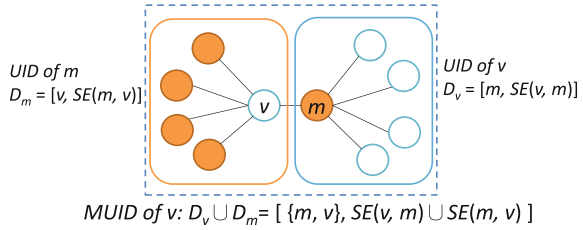
- (a) For $m \in M_v$, the UID of m can be found as $M_m = \{v\}$ and $SE(m, M_m) = \{x | x \in M_v - \{m\}, type(x) = type(m)\}$.
- (b) For $u \in SE(v, M_v)$, the UID of u can be found as $M_u = M_v$ and $SE(u, M_u) = SE(v, M_v) \cup \{v\} - \{u\}$.

From (a) and (b), we have shown that $\forall u \in D$, it is possible to identify u with UID, $D_u = [M_u, SE(u, M_u)]$, such that $M_u \subseteq \{v\} \cup M_v$ and $SE(u, M_u) \subseteq \{v\} \cup M_v \cup SE(v, M_v)$. Therefore the UID of v identified by the 1 -Hop+ method has been proven to be an MUID of v by redefining M and SE set. □

In Fig. 7 an example of redefining UID by 1 -Hop+ method is shown. The problem for the MUID obtained this way is that for each query node, all the neighbor nodes are included in the MUID. For nodes with large degrees, the MUID generated will be large in size, which is not preferable. Also for the neighbor nodes of v , they are SE to other neighbors with the same type, leaving large SE set in their own UIDs.

In the *One-Neighbor* method, given a vertex v , we obtain a UID, $D_v = [\{m\}, SE(v, m)]$. To extend the *One-Neighbor* method for MUID, we propose *One-Neighbor-MUID* method to further uniquely identify the newly introduced nodes. The proposal is to obtain the UID for m , D_m , in a similar manner. With $D_m = [\{v\}, SE(m, v)]$ as shown in Fig. 8, $D_v \cup D_m = [\{m\} \cup \{v\}, SE(v, m) \cup SE(m, v)]$

Fig. 8 The MUID of v found by the *One-Neighbor-MUID*, obtained by combining UIDs of v and m



becomes the MUID of v . The problem for the *One-Neighbor-MUID* is that for each node, there could be many SE nodes, which is not desirable in general.

Theorem 6 Given a vertex v and one of its neighbors m and the UID, $D_v = [\{m\}, SE(v, m)]$, $D_m = [\{v\}, SE(m, v)]$, can be identified by the *One-Neighbor* method. The union of UIDs, $D_v \cup D_m$, is an MUID of v .

Proof Given v and m , which are uniquely identified by UID, D_v and D_m , respectively, we have to show that for each element $u \in SE(v, m)$ and $u \in SE(m, v)$ we can find UID, $D_u = [M_u, SE(u, M_u)]$.

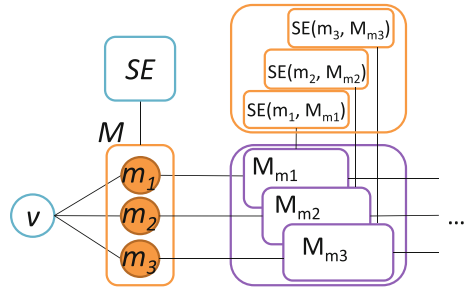
- (a) For each $u \in SE(v, m)$, its UID can be found as $M_u = \{m\}$ and $SE(u, M_u) = SE(v, m) \cup \{v\} - \{u\}$.
- (b) For each $u \in SE(m, v)$, its UID can be found as $M_u = \{v\}$ and $SE(u, M_u) = SE(m, v) \cup \{m\} - \{u\}$.

From (a) and (b), we can conclude that every vertex $u \in D_v \cup D_m$ can be identified by a UID D_u , where $D_u \subseteq D_v \cup D_m$. We prove that the result of *One-Neighbor-MUID*, $D_v \cup D_m$, satisfies the requirement as an MUID of v . □

We have showed that both *I-Hop+* and *One-Neighbor-MUID* can be extended to obtain MUID given a query vertex v . However, the quality of MUID produced by these methods is usually not optimal. To overcome such deficiency, we try to extend the *Multiple-Neighbor* method for the MUID problem. As proposed previously, the spirit of the *Multiple-Neighbor* method is to choose a small neighborhood subset M of a given query v , leading to a smallest $SE(v, M)$ to uniquely identify v . To extend the method for finding MUID as *Multiple-Neighbor-MUID*, given the query vertex v and its UID $D = M_v \cap SE(v, M_v)$, we have to make sure that all newly included $m \in M_v$ and $u \in SE(v, M_v)$ need to be uniquely identified, the process continues. It is not hard to show that each u belonging to $SE(v, M_v)$ obtained from our *Multiple-Neighbor* method has been uniquely identified. Unfortunately, their corresponding UIDs might not be optimal because there could exist another set M identifies better SE sets. Thus, we need to include more vertices into the MUID to identify the best result set.

We have proposed to use a heuristic $SEgain(v, n, M) = SE(v, M) - N(n)$ as a criterion to choose the next neighbor to be added in the *Multiple-Neighbor* method. This criterion calculates the number of nodes that can be removed from the UID if $n \in N(v)$ is added. However, when it comes to *Multiple-Neighbor-MUID*, all the newly added nodes in M and $SE(v, M)$ also have to be uniquely identified. As shown in Fig. 9, to estimate how many additional nodes would be introduced after adding m_1 ,

Fig. 9 Example of *Multiple-Neighbor-MUID*



we need to estimate how many vertices are required to identify M_{m_1} , and to identify the further introduced vertices.

Similar to the *Multiple-Neighbor* method for the UID problem, here we define a heuristic function to estimate the quality of neighbors as candidates to be added into M . The idea is to execute the UID *Multiple-Neighbor* method for one pass on all nodes and record the M and $SE(v, M)$ sets of each node v as $UID_{SE}(v)$ and $UID_M(v)$. When choosing the neighbors of v into M in *Multiple-Neighbor-MUID*, we give higher priority for neighbor n with smaller $|SE(v, M)|$ in UID, or $|UID_{SE}(n)|$. Given equal-sized $|UID_{SE}(n)|$, we then define the secondary criteria as minimizing $|UID_M(n) - M|$. This is because that exploiting neighbors already in M could potentially introduce fewer new vertices. Given equal-sized $|UID_{SE}(n)|$ and $|UID_M(n) - M|$ the tertiary criteria is larger $SEgain(v, m, M)$. Note that for n to be chosen into M , $|SEgain(v, n, M)|$ must be larger than 0.

Algorithm 3 describes the algorithm in details.

6 Experimental results

We introduce 3 real and 3 synthetic datasets for evaluation in Sect. 6.1. Then, in Sect. 6.2 we display the experimental results of finding UID and MUID, respectively. Finally, we report some case studies on MUID examples in Sect. 6.3.

6.1 Experiment settings

We exploit three real datasets and three synthetic datasets to evaluate our model. The statistics of all the data sets are shown in Table 2. The three real datasets include KDD movie data set, HepTh citation network, and academic co-author network. The KDD movie data set is a movie-centric heterogeneous information network. There are 35,311 nodes belonging to 20 different types. The node types include movie, actor, director, place, producer, ...etc. There are a total of 168,868 edges in this network connecting different entities. The second real data set encodes paper-author relationship among numerous scholars and published research papers. This network consists of 63,122 nodes and 770,155 edges. There are six types of nodes—student, professor, department, college, and Chinese and English keywords. For the third dataset, we choose a commonly used citation graph, Arxiv high energy physics theory (HEPTH), provided

Algorithm 3: *Multiple-Neighbor-MUID* method**Input:** A Graph $G = (V, E)$, a query vertex v , precomputed $SE(v, M)$ and M in UID**Output:** MUID set X of v

```

1  Get  $N(v) = \{m_1, m_2, \dots, m_d\}$ ;
2   $M_x \leftarrow \emptyset$ 
3   $SE_x \leftarrow \emptyset$ 
4   $UID_{SE}(v) \leftarrow$  Read  $SE(v, M)$  of  $v$ 's UID
5   $UID_M(v) \leftarrow$  Read  $M$  of  $v$ 's UID
6   $W$  is a stack and initially empty.
7  //  $W$  stores not yet uniquely identified vertices
8  push  $v$  into  $W$ 
9  while  $|W| > 0$  do
10    $w \leftarrow$  pop first element from  $W$ 
11    $SE_w \leftarrow SE(w, M_x)$   $M_w \leftarrow \emptyset$ 
12    $N_x \leftarrow N(v) - X$ 
13   while  $|SE_w| > 0$  do
14     for  $n \in N_x$  do
15       if  $|SE_{gain}(w, n, M_w)| = 0$  then
16          $N_x \leftarrow N_x - \{n\}$ 
17     end
18   end
19   if  $N_x = \emptyset$  then
20     break;
21   end
22    $minUID_{SE} \leftarrow$  max integer
23    $minUID_M \leftarrow$  max integer
24    $maxSEG \leftarrow$  min integer
25    $n_{opt} \leftarrow null$ 
26   for  $n \in N_x$  do
27     if  $(UID_{SE}(n) < minUID_{SE})$  or
28      $(UID_{SE}(n) = minUID_{SE} \text{ and } UID_M(n) < minUID_M)$  or
29      $(UID_{SE}(n) = minUID_{SE} \text{ and } UID_M(n) = minUID_M \text{ and } SE_{gain}(w, n, M_w) > maxSEG)$  then
30        $minUID_{SE} \leftarrow UID_{SE}(n)$ 
31        $minUID_M \leftarrow UID_M(n)$ 
32        $maxSEG \leftarrow SE_{gain}(w, n, M_w)$ 
33        $n_{opt} \leftarrow n$ 
34     end
35   end
36    $M_w \leftarrow M_w \cup \{n_{opt}\}$ 
37    $SE_w \leftarrow SE_w - N(n_{opt})$ 
38   push  $n_{opt}$  into  $W$ 
39   end
40   for  $u \in SE_w$  do
41     if  $N(u) \supset N(w)$  then
42       p
43     end
44     ush  $u$  into  $W$  //  $\exists M_u$  that  $w \notin SE(u, M_u)$ 
45   end
46    $M_x \leftarrow M_x \cup M_w$ 
47    $SE_x \leftarrow SE_x \cup SE_w$ 
48 end
49 Output  $X = [M_x, SE_x]$ ;

```

Table 2 Data sets statement

Name	IVI	IEI	Number of types
KDD movie dataset	35311	168868	20
TW academic network	63122	770155	6
HepTh citation network	41840	933149	4
Erdős-Rényi Model	50000	249128	3
Barabási-Albert Model	50000	249179	3
Watts-Strogatz Model	50000	300000	3

in KDD Cup 2003. There are 93,319 edges connecting 41,840 nodes with four types, author, paper, journal, and email domain. For each dataset, a giant connected component (GCC) is obtained for our experiments. We also create three synthetic datasets of 50,000 nodes each, based on three different social network generation models. We apply Erdős and Rényi (1961) model to produce a random graph; the Albert and Barabási (2002) model to produce scale-free networks; and the Watts and Strogatz (1998) model to produce graphs that can mimic the small world phenomena. For the Erdős-Rényi random graph (ER) model, we set the connection probability to be 0.0002. For the Barabási-Albert (BA) model, we set the number of initial nodes to 5. For the Watts-Strogatz (WS) model, we set the mean degree to 6, and the edge rewiring probability to 0.18. We assume there are three types of vertices in the network and randomly assign a type to each node.

In Table 3 we compare different models by averaging their rankings (1–3). Note that the ranking is generated using all vertices in the graph based on Definition 3 (i.e., we compare $|SE(v, M)|$ first, and the second criteria is $|M|$ given identical $|SE(v, M)|$). Finally, the methods with lower average rank are considered having better performance, namely obtaining better quality of UIDs.

For MUID, we compare the performance of the three greedy methods only because the exhaustive search method becomes intractable. Similar to our experiment on UID, Table 4 displays the comparison between the three greedy-based algorithms on MUID using the metrics described in Sect. 3.1. We return the average rank (calculated over all vertices) of each model.

6.2 Data analysis

We first discuss the results for UID in Table 3. In all datasets, the *Multiple-Neighbor* method obtains as good results as the exhaustive search method does. The *One-Neighbor* method, although generally cannot achieve the optimal solution, usually does better than the *1-Hop+* method. The *1-Hop+* method guarantees the minimal SE set at the cost of including all neighbors into M . The *One-Neighbor* method includes only one neighbor into M and therefore cannot guarantee minimizing the size of SE set. Therefore, it is not hard to show that for datasets where the optimal UID can be obtained with only one neighbor, the *One-Neighbor* method would outperform the *1-Hop+* method. The last row of the table records the execution time to generate the

Table 3 The UID experimental results of the datasets

Dataset	MN	ON	OH	EX
KDD movie	1.00	1.80	2.21	1.00
TW academic	1.01	2.65	3.63	1.00
HepTh	1.00	2.50	3.39	1.00
ER	1.00	2.70	3.70	1.00
BA	1.00	2.62	3.62	1.00
WS	1.00	2.30	3.29	1.00
Execution time of TW Academic (sec.)	383.06	271.84	256.22	3876.46

In the table, “MN”, “ON”, “OH”, “EX” stand for the *Multi-Neighbor* method, the *One-Neighbor* method, the *1-Hop+* method, and exhaustive search, respectively. The last row records the execution time on Taiwan Academic network

UIDs for all vertices on the Academic network dataset. It shows that the *Multiple-Neighbor* method, although requires slightly more time than the other two greedy methods, performs significantly more efficient than the exhaustive search does.

In Table 4 we discuss the quality of results of the MUID identified by each of the three methods. In both metrics, *Multiple-Neighbor-MUID* obtained the best result and it generally does a better job minimizing the *TSE* than *USE*. Table 5 shows the mean of union size as well as the *TSE* and *TM* on the Taiwan academic network dataset. The *Multiple-Neighbor-MUID* obtains the minimal *TSE* and *USE*. The *One-Neighbor-MUID* method always obtains the minimal *UM* as 2 (the query and its neighbor). Although the *1-Hop+* method guarantees the minimal SE set for the UID of the ego vertex, it includes full neighborhood of the ego vertex. The neighbors of the same type would be SE to each other, therefore it introduces not only a large set of *M* but also a large SE set. For the KDD Movie dataset, the *One-Neighbor-MUID* and the *1-Hop+* methods seem to perform closer to *Multiple-Neighbor-MUID* than with other datasets. It is because in this dataset more than 40% vertices have very low degrees. Also, we have observed that in the dataset with more node types, the performance of the *One-Neighbor-MUID* method improves because such data provides more chances for *One-Neighbor* method to uniquely identify the query node using only one neighbor node.

Furthermore, we compare the effectiveness of the heuristics introduced in Sect. 5: UID_{SE} , UID_M , SE_{gain} . In Table 6 the *Full* setting utilizes all three heuristics and performs the best. The setting, $H_A(UID_{SE}$ is excluded), generally performs the worst. It is because that using UID_M without UID_{SE} may include neighbors with many SE nodes. This also explains why on the HepTh dataset the H_A looks fine because there are fewer SE nodes. Also, it seems that UID_M is the least effective heuristic among the three.

6.3 Case study

Here we conduct a case study to demonstrate real-world results for MUID using the KDD Movie dataset. Figure 10 shows an example of three different MUIDs discovered

Table 4 The MUID experimental results in the datasets

Dataset	Metrics	MN	ON	OH
KDD movie	Union	1.33	1.61	2.07
	Sum	1.01	1.81	2.19
TW academic	Union	1.20	2.43	2.32
	Sum	1.00	2.55	2.40
HepTh	Union	1.04	1.92	2.79
	Sum	1.02	1.95	2.78
ER	Union	1.00	2.02	2.96
	Sum	1.00	2.14	2.84
BA	Union	1.00	2.00	2.98
	Sum	1.00	2.04	2.95
WS	Union	1.05	1.76	2.99
	Sum	1.05	1.78	2.97

“Union” and “Sum” represent the two metric ranks described in Sect. 3.1

Table 5 The MUID experimental results in mean of union size and mean of sum of size on Taiwan Academic network

Dataset	Metrics	MN	ON	OH
TW academic	USE	3.00	10.45	12.58
	UM	10.19	2.00	13.20
	TSE	80.04	425.27	1563.26
	TM	29.49	12.16	32.58

USE, UM, TSE, TM are defined in Sect. 3.1

Table 6 The comparison between heuristic sets in *Multiple-Neighbor-MUID*

Dataset	Metrics	Full	H_A	H_B	H_C
KDD movie	Union	1.07	2.00	1.32	1.63
	Sum	1.07	2.01	1.32	1.64
TW academic	Union	1.31	2.32	1.45	1.83
	Sum	1.36	2.34	1.45	1.83
HepTh	Union	1.35	1.58	1.78	1.82
	Sum	1.38	1.60	1.79	1.83

Full stands for the heuristic setting including the three heuristics. H_A excludes UID_{SE} , H_B excludes UID_M , and H_C excludes both of UID_{SE} and UID_M (i.e., H_C includes only SE_{gain})

by the three proposed algorithms. The *1-hop+* method simply obtains the neighbors of the ego vertex (i.e., movie Interiors marked in grey). *One-Neighbor* adds a single node, actor Kristin Griffith, first because this vertex has degree 1 and will not further produce SE nodes that need to be further identified. The *Multiple-Neighbor* method

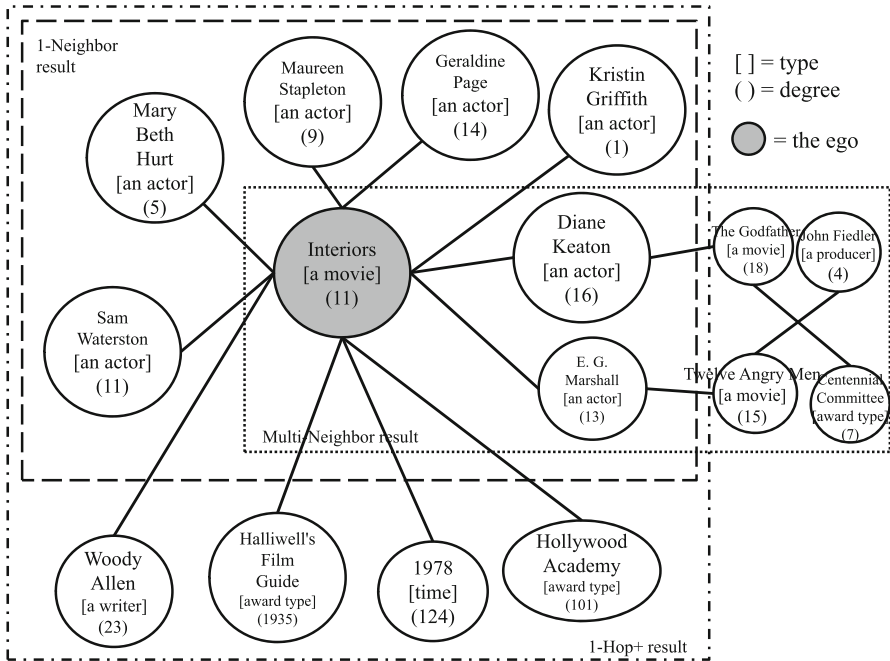


Fig. 10 Three different groups. A case that MN wins in all evaluation metrics

first selects two neighbor nodes since they are less likely to introduce too many nodes that require a large amount of vertices to identify.

Table 7 shows the size of MUID sets produced by different models. Note that for the *Multiple-Neighbor* method, from *TSE* we can realize each node in MUID can be uniquely identified by its M set without the requirement of an SE set, which implies the vertices in this set are more unique than those in other MUID sets.

After examining the MUID results in KDD Movie dataset, we have identified three interesting patterns: Outliers, high-degree nodes and movie series. We will discuss each pattern in detail below.

6.3.1 Outliers

Figure 11 displays a frequent pattern that contains some inactive individuals in the network. Firstly, a vertex with small degree (e.g., an actor named David Tom) is chosen as the ego. In this case, 1-Hop+ and One-Neighbor MUID methods return the same set of nodes. It is because that this ego has only one single neighbor (movie *Stepfather 3*), which has to be included by both methods. Then, it is necessary to include SE('David Tom', 'Stepfather 3') so that Tom can be unique. Then one can observe that the mission has been accomplished because all the 7 nodes in the set can be uniquely identified by the remaining nodes in the set. Generally speaking, in the movie dataset, we found plenty of MUID of such flavor: an outlier ego, with its neighbor and some SE nodes. The *Multiple-Neighbor* method first includes two more nodes to identify

Table 7 The MUID experimental results in union size and sum of size for the cases in this section

Queried vertex	Metrics	MN	ON	OH
Interiors	USE	0	7	9
	UM	7	2	12
	TSE	0	42	44
	TM	12	8	22
David Tom	USE	6	6	6
	UM	5	2	2
	TSE	20	30	30
	TM	13	7	7
Saving private Ryan	USE	0	2	8
	UM	6	2	13
	TSE	0	2	16
	TM	10	3	24
Blondie on a budget	USE	5	17	6
	UM	9	2	5
	TSE	9	272	30
	TM	49	18	28

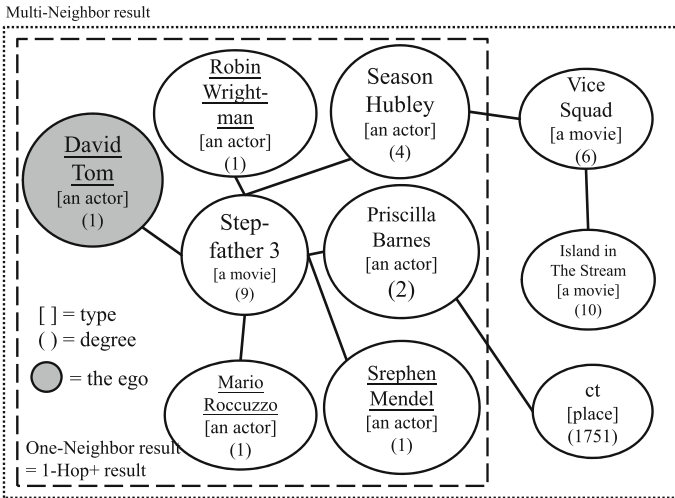


Fig. 11 Outliers. The MN performed best in TSE

Season Hubley and Priscilla Barnes so their own SE set can be minimized. Then, a movie *Island in The Stream* has to be includes to uniquely identify *Vice Squad*. Note that although the value $|SE(v, M)|+|M|$ of the *Multiple-Neighbor* method is not as small as the other methods in these examples, it generally produces the better union and sum of SE sets, which implies that some additional vertices might be added to reduce the size of the SE set.

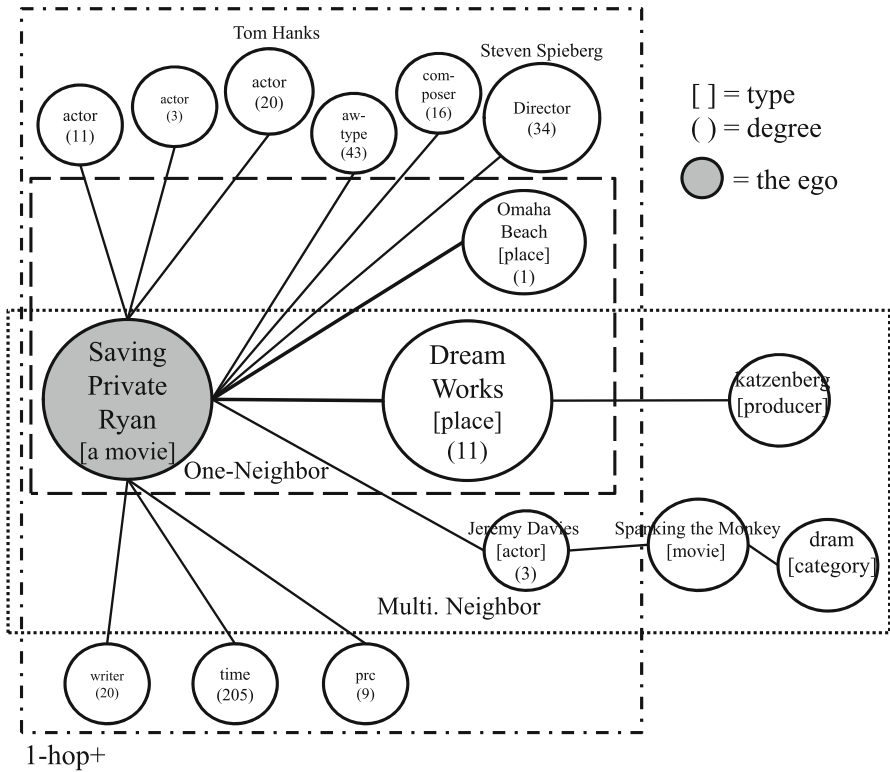


Fig. 12 High-degree nodes. The *MN* outperformed others in both SE sizes

6.3.2 High-degree nodes

An example of high degree nodes is illustrated in Fig. 12. By taking *Saving Private Ryan* as the ego, the *Multiple-Neighbor* select DreamWorks and Jeremy Davies to uniquely identify the ego node, indicating that this is the only movie performed by J. Davis and published by DreamWorks. To identifying both neighbors, we include three more nodes and finally produce a MUID with an empty SE set, which means all nodes in this set are unique given certain neighbors. The *One-Neighbor* algorithm chooses Omaha Beach to uniquely identify the ego. Then, it continues to find DreamWorks (place) to uniquely identify Omaha Beach. Although the total size of graph is small, it does not produce empty SE sets as both places have each other as the SE node. The *1-Hop+* method included all 12 neighbor nodes. In general, nodes with higher degree have higher chance to connect to vertices with rare type and fewer connections (e.g., Omaha beach).

6.3.3 Series of movies

The next example describes the cases where there exist many SE nodes. Figure 13 uses *Blondie on a Budget* as the ego, and shows that there are 5 nodes that are SE to this

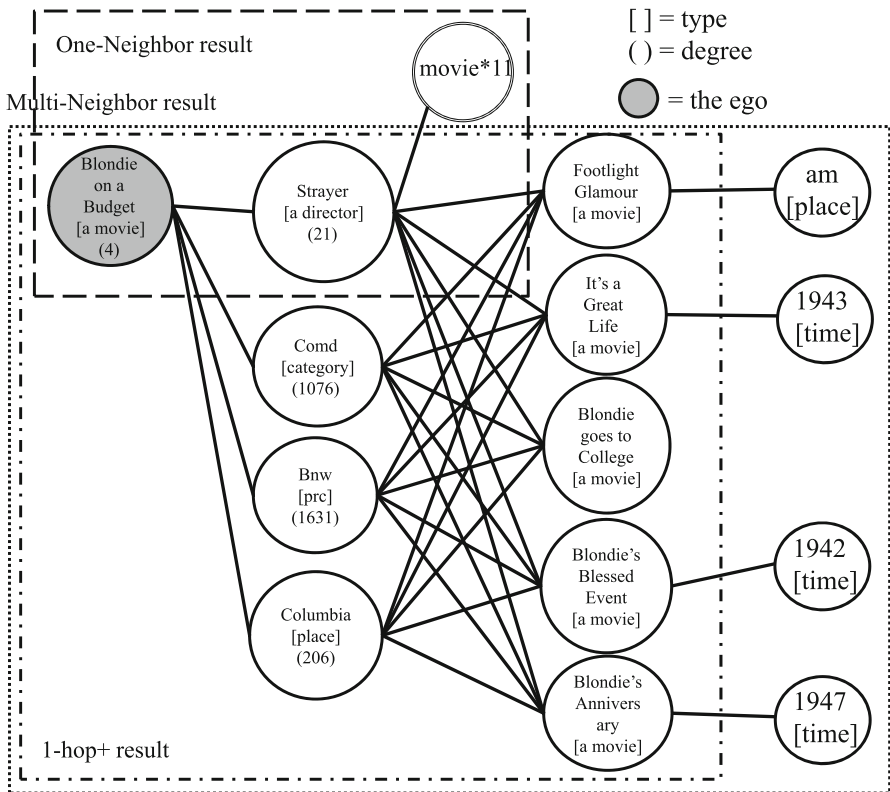


Fig. 13 Series of movies. A case that MN performed best; while ON performed worst

ego. It is inevitable for any algorithm to include these 5 nodes in order to distinguish the movie from others. The *Multiple-Neighbor* produces the set that includes some relevant information (e.g., director, category, etc.) about the movie, and those 5 SE nodes. For recommendation purpose, we may find those movies are suitable candidates to replace each other. Furthermore, additional information about the 5 added movies (e.g., place, time) is provided, so they can be identified with minimal SE sets.

7 Conclusions

This paper shows that the uniqueness of a node can be captured in many different ways: using all the neighbors plus some similar nodes (*1-Hop+*), using a combination of 1st degree and 2nd degree neighbors (*One-Neighbor method*), greedily adding vertices that are likely to be unique themselves (*Multiple-Neighbor method*). This paper proposes and proves some valid methods to identify a closure set of mutually identifiable nodes, and focus on finding one with small SE set size in UID or smaller union or summation SE set size in MUID. The future work lies in the investigation of some other metrics such as diversity of types, coverage, etc, and comes up with efficient and effective models to identify such sets and their complexity analysis accordingly.

Acknowledgements Mi-Yen Yeh's research is supported in part by National Science Council of Taiwan, R.O.C., under Contracts NSC101-2221-E-001-013. Shou-De Lin's research is supported by National Science Council, National Taiwan University and Intel Corporation under Grants NSC100-2911-I-002-001, NSC101-2628-E-002-028-MY2 and NTU102R7501. Jian Pei's research is supported in part by an NSERC Discovery Grant and a BCFRST NRAS Endowment Research Team Program project. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

- Albert R, Barabási A-L (2002) Statistical mechanics of complex networks. *Rev Modern Phys* 74:47–97
- Erdős P, Rényi A (1961) On the evolution of random graphs. *Bull Inst Int Stat* 38:343
- Fortunato S (2010) Community detection in graphs. *Phys Rep* 486(3–5):75–174
- Lappas T, Liu K, Terzi E (2009) Finding a team of experts in social networks. In *Proceedings of ACM SIGKDD international conference on knowledge discovery and data mining*
- Li C-T, Lin S-D (2009) Egocentric information abstraction for heterogeneous social networks. In *ASONAM, 2009*
- Li C-T, Shan M-K (2010) Team formation for generalized tasks in expertise social networks. In *IEEE SocialCom*
- Newman M (2004) Detecting community structure in networks. *Eur Phys J B Condens Matter Complex Syst* 38(2):321–330
- Sozio M, Gionis A (2010) The community-search problem and how to plan a successful cocktail party. In *Proceedings of ACM SIGKDD international conference on knowledge discovery and data mining*
- Tang J, Zhang J, Yao L, Li J, Zhang L, Su Z (2008) ArnetMiner: extraction and mining of academic social networks. In *Proceedings of ACM SIGKDD international conference on knowledge discovery and data mining*
- Watts D, Strogatz S (1998) Collective dynamics of small-world networks. *Nature* 363:202–204
- Yang D-N, Chen Y-L, Lee W-C, Chen M-S (2011) Social-temporal group query with acquaintance constraint. In *Proceeding of international conference on very large data, bases*
- Zhou B, Pei J (2008) Preserving privacy in social networks against neighborhood attacks. In *Proceedings of IEEE international conference on data, engineering*
- Zhu J, Nie Z, Liu X, Zhang B, Wen J-R (2009) StatSnowball: a statistical approach to extracting entity relationships. In *Proceedings of international world wide web conference*