Identifying Smallest Unique Subgraphs in a Heterogeneous Social Network

Yen-Kai Wang, Wei-Ming Chen, Cheng-Te Li, Shou-De Lin
Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan
Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan
{r01922023, r02922010, sdlin}@csie.ntu.edu.tw, ctli@citi.sinica.edu.tw

***Abstract*—This paper proposes to study a novel problem, discovering a *Smallest Unique Subgraph* (SUS) for any node of interest specified by user in a heterogeneous social network. The rationale of the SUS problem lies in how a person is different from any others in a social network, and how to represent the identity of a person using her surrounding relational structure in a social network. To deal with the proposed SUS problem, we develop an *Ego-Graph Heuristic* (EGH) method to efficiently solve the SUS problem in an approximated manner. EGH intelligently examine whether one graph is not isomorphic to the other, instead of using the conventional subgraph isomorphism test. We also prove SUS is a NP-complete problem through doing a reduction from *Minimum Vertex Cover* (MVC) in a homogeneous tree structure. Experimental results conducted on a real-world movie heterogeneous social network data show both the promising efficiency and compactness of our method.**

Keywords—unique subgraph; smallest; unique, heterogeneous social network; subgraph isomorphism

I. INTRODUCTION

A social network is a graph in nature, where the nodes stand for actors (e.g. authors) and the edges between two actors represent their relationships (e.g. co-authorship). In social network analysis (SNA), people have proposed different measures for the graph structure to model general network phenomena or to capture some hidden properties, such as the six degree of separation and power-law degree distribution [2]. Analyzing a social network can not only assist experts in under-standing the social phenomenon but also help laymen manage their social circles. One of the most important topics of SNA is to identify strange individuals whose behaviors are anomalous, comparing to most of the usual nodes in the social network. For example, in an online social network, one might want to spot users who might be spammers or robots because their behaviors of making friends are abnormal in terms of both high frequency and large volume. In this paper, as a complement to the anomaly detection problem, we propose to study a novel problem, how to identify the uniqueness of a node of interest from its surrounding social structure in a big social network. Specifically, we aim to find a *Smallest Unique Subgraph* (SUS) which best describes the difference between itself and all the other nodes in a big social network.

The proposed SUS problem targets at a *heterogeneous* social network due to its power of representing diverse kinds of social relationships and interactions. There are many successful proposals for SNA on *homogeneous* social network for simplicity, where only single type of nodes and relations are in the network. However, in the real world, different types of objects can be connected through different kinds of relationships; it is more natural to define multiple types of entities and relations in a social network. In this sense, *heterogeneous* social network [3] is proposed to describe the complex relationships (i.e., typed edges) among entities. For example, a heterogeneous movie network shown in Figure 1 takes movies (M), directors (D), writers (W), and actors (A) as nodes, and their corresponding relationships as tuples such as <D_1, **direct**, M_1>, <M_1, **has actor**, A_1>, <M_3, **originate from**, M_4>, where the capital letter in the tuple stands for the type of source nodes, and the second element for the type of relations. We assume the behaviors of each node can be represented by its surrounding relational structure. For example, in Figure 1, consider the two-step neighboring relational structure as the behaviors of a node, two of actor A_1's behaviors are: he is the **spouse of** writer W_2 who had ever **written the script** for movie M_3. A1 is also an **actor of** movie M_3 which **originated from** movie M_4.

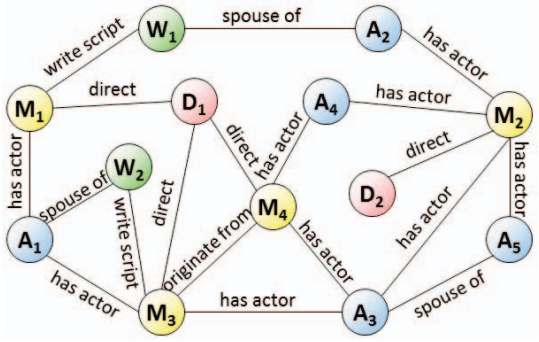


Fig 1. A heterogeneous movie social network. The capital letter on each node stands for its type: M (movie), D (director), A (actor), and W (writer). There are five relation types, including "write script", "has actor", "spouse of", "direct", and "originate from."

In this paper, given a node of interest, we aim to identify the *smallest unique subgraphs* in a heterogeneous social network. The underlying intuition is to find a subset of the surrounding relational structure of the query node (as its representative behaviors), such that the ego node can be uniquely distinguished from other nodes in the network. In more details, the desired subgraph describing the node of

978-1-4799-9926-2/15/$31.00 ©2015 IEEE

interest should satisfy two requirements. The first is *uniqueness*, which states that we require the mined subgraph is a unique one with respect to the mined subgraphs of all the other nodes in the social network. In other words, the identified subgraph (i.e., behaviors) of a node cannot be identical to any of the subgraphs of other nodes. The subgraph of the query node must to be able to exhibit the difference of behaviors between itself and all the other individuals. The second is *compactness*, which states that using least information description (i.e., the smallest unique subgraph) to represent the unique-ness of the query ego node. The compactness ensures us to not only include the most significant behaviors of the ego node, but also reduce both the space and time complexity of advanced analysis tasks. In fact we can easily find an arbitrary large unique subgraph containing the query node to describe itself. However, such large unique subgraph is less meaningful since (a) we need to use too much effort to describe the node, and (b) the relational structure far from the query node might be less irrelevant.

For anomaly detection, there are many relevant studies. For example, to identify the outlier users, some local friendship-based metrics are proposed to measure the differences from a certain user to all the other users [4]. In addition, the similarity among the friendship patterns is considered to match the users from across multiple social networks [5]. To the best of our knowledge, we are the first to deal with the task of identifying smallest unique subgraphs in a heterogeneous social network.

Finding the smallest unique subgraphs can enable several potential applications. For example, in the setting of social team formation [12], the uniqueness of experts can be used to find which ones are irreplaceable, and thus we can boost the effectiveness of the management of team members. The mined unique signatures of individuals can be also applied to identify outliers as well as terrorists in a social network using the egocentric abstraction technique [13], because the behaviors of such kind of individuals tend to be quite peculiar to others. In addition, one can put more efforts on the unique subgraphs of nodes when anonymizing a social network [14], since their special behaviors are more subject to be attacked to have higher accuracy of re-identification.

Identifying smallest unique subgraphs in a heterogeneous social network is a very challenging problem. The main reason is that solving SUS problem needs to perform the *subgraph isomorphism* tests between pairs of subgraphs. To make the subgraphs unique, in a heterogeneous social network $G = (V, E)$, for each node $v \in V$, we have to find the smallest connected subgraph$S$ such that there is no subgraph in the complement graph $CoG = G - S$ isomorphic to S. In other words, we have to compare each candidate subgraph to each of the mined unique subgraphs and ensure the candidate does not appear in the unique subgraphs of other nodes. As we know the complexity of subgraph isomorphism is NP-complete [10], we can further prove the hardness of the SUS problem to be NP-complete as well. Consequently, the SUS problem has no simple solution theoretically. We will provide the detailed analysis of hardness in the following sections.

To solve the proposed SUS problem, we propose an *Ego-Graph Heuristics* (EGH) approach, which is designed to be an efficient and effective method that escapes from the subgraph isomorphism test. The central idea is two-fold. First, while the subgraph isomorphism test is very time-consuming, it is much easier to tell whether or not one graph $G_1$is *not*a subgraph isomorphic to the other graph G_2.In other words, if G_1 possesses some relational features that cannot be found in the set of relational features of G_2, we can say G_1 is not isomorphic to G_2. Second, if a subgraph is validated to be unique (i.e., is not isomorphic to all the other possible subgraph candidates), according to the Apriori property, any of its supergraph is unique as well. Moreover, we will further show that the proposed EGH approach can be more efficient when the graph we target at is simpler, such as the tree structure. The EGH approach consists of two major stages. The first stage aims to find the smallest unique subgraphs for a small set of nodes. The second stage is, for any given node of interest v, to create a connection from node v to the nearest smallest unique subgraph S^* among the unique subgraphs derived at the first stage. The subgraph that contains nodes in the connection path as well as the subgraph S^* is considered as the resulting smallest unique subgraph for node v.

We summarize the contributions of this paper as follows.

- We propose to deal with a novel problem of identifying the *Smallest Unique Subgraph* (SUS) for any node of interest in a heterogeneous social network, which can be viewed as a complement problem of anomaly detection.
- Technically we develop an *Ego-Graph Heuristics* (EGH) approach to efficiently solve the SUS problem in an approximated manner. EGH intelligently examine whether one graph is not isomorphic to the other, instead of using the conventional subgraph isomorphism test.
- Theoretically we analyze the hardness of the SUS problem in terms of subgraph isomorphism, and prove that solving the SUS problem in a tree structure is NP-complete.
- Empirically the evaluation conducted on a large-scale movie heterogeneous social network shows the promising performance for time efficiency and SUS size, which is competitive to the optimal solution derived from the exhaustive method.

II. Problem Statement and Analysis

Definition 1: *Heterogeneous Social Network.* A heterogeneous social network $H = (V, E, L_V, L_E, f_V, f_E)$ is an undirected labeled graph, where V is a finite set of nodes, $E \subseteq (V \times V)$ is a finite set of edges, L_V is a finite set of node types, $L_E \subseteq (L_v \times L_v)$ is a finite set of edge types. And that $f_V: V \to L_V$ is the node label function which maps each node to its node type, and $f_E: E \to L_E$ is the edge label function which maps each edge to its edge type.

Definition 2: *Uniqueness.* A subgraph S of graph $H$is unique if and only if when all the nodes and edges in S are removed from Hand form the complement graph *CoS*, S is not a subgraph of *CoS*. In other words, $S = (V_s, E_S, L_V, L_E, f_V, f_E)$ is not a subgraph of $CoS = H \backslash S := (V \backslash V_s, E \backslash E_S, L_V, L_E, f_V, f_E)$.

Problem Definition: *Smallest Unique Subgraph* (SUS) *problem.* Given a heterogeneous social network H and a particular node of interest v, the SUS problem aims to find a

subgraph $S \subseteq H$ such that (a) $v \in$ S, (b) S is *unique* in H, and (c) there is no other subgraph S', whose size is less than S, satisfying the previous two requirements. The last requirement refers to that the mined unique subgraph S is preferable to be as small as possible.

As a decision problem, the SUS problem is free from the explosive number of the resulting unique subgraph. Nevertheless, SUS still suffers from the computational complexity problem, since we should perform the subgraph isomorphism test to determine whether or not there is a mapping of a candidate subgraph in the complement graph. We arrange the hardness of the isomorphism test in Table 1. It is NP-complete to test both subgraph and subtree isomorphism in a graph while it takes only polynomial time to complete the subtree isomorphism test in a tree structure. We can apparently find that the isomorphism test on tree structures seems to be relatively trivial. However, even equipped with the compact subtree isomorphism test, solving the SUS problem is still impractical due to the NP-complete hardness. We further provide the complexity of solving the SUS problem using either the subgraph or subtree isomorphism test in Table 2.

Table 1. Complexity of subgraph and subtree isomorphism

Complexity	Subgraph	Subtree
Graph	NP-complete [10]	NP-complete [11]
Tree	N/A	Polynomial [1]

Table 2. Complexity of smallest unique subgraph and subtree

Complexity	Subgraph	Subtree
Graph	NP-hard	NP-hard
Tree	N/A	NP-complete (appendix A)

III. Ego-Graph Heuristics Approach

A. Algorithm Overview

As aforementioned, since the subgraph isomorphism problem in a graph is NP-complete and is one of the components in the proposed SUS problem, the SUS problem is also NP-complete. To our knowledge, there is no efficient algorithm to do the subgraph isomorphism, that says, determine whether or not a subgraph is unique. In this work, therefore, we alternatively propose a heuristic algorithm, *Ego-Graph Heuristics* (EGH), which is developed to efficiently identify the smallest unique subgraph in an approximate manner. The proposed EGH algorithm consists of three steps. First, we define and extract k-layer subgraphs which are centered at all the nodes in H as the unique subgraph candidates. Second, for each unique subgraph candidate, we create an *Egocentric Information Table* (EIT), which represents distinct kinds of relational behaviors of each node. Third, we validate the uniqueness of each candidate by pair-wisely comparing the entries of tables. Some candidates are guaranteed to be unique while those candidates violating the uniqueness are removed. Finally, for each node v whose k-layer subgraph candidates violate the uniqueness, we connect node v to a nearest discovered unique subgraph, identified in the third step, as the corresponding final unique subgraph.

B. Extracting k-layer Subgraph as Candidates

In a heterogeneous social network, we assume that the behaviors of each node can be represented by its surrounding relational structure. We define the k-layer subgraphs centered at a node to capture its surrounding relational structure, as in the following. Then such k-layer subgraphs of a node are considered to be the candidates of unique subgraphs, which will be further validated in the following sub-sections.

Definition 3: *k-layer Subgraph*. Given a node v in a heterogeneous social network $H = (V, E, L_V, L_E, f_V, f_E)$, the k-layer subgraph of node v is the subgraph centered at v expanding outward up to k steps using the BFS manner. The k-layer subgraph of node v is denoted by $S^k[v] = (V^k[v], E^k[v], L_V, L_E, f_V, f_E)$, where $V^k[v] = \{u | u \in V, Length(path(v,u)) \leq k\}$, and $E^k[v] = \{e | e \in E, e \in path(v,u), Length(path(v,u)) \leq k\}$. Note that $path(v,u)$ is a path between node v and u in H, and $Length(path(v,u))$ is the length of $path(v,u)$.

For example, in Figure 1, the 1-layer subgraph of node A_1 is $S^1[A_1] = (V^1[A_1], E^1[A_1])$, where $V^1[A_1]$ = {A_1, M_1, M_3, W_2}and $E^1[A_1]$ = {(M_1, has_actor, A_1), (M_3, has_actor, A_1), (A_1, spouse_of, W_2)}. The 2-layer subgraph of A_1 is$S^2[A_1] = (V^2[A_1], E^2[A_1])$, where $V^2[A_1]$ = {A_1, M_1, M_3, W_2, W_1, D_1, M_4, A_3} and $E^2[A_1]$ = {(M_1, has_actor, A_1), (M_3, has_actor, A_1), (A_1, spouse_of, W_2), (W_2, write_script, M_3), (W_1, write_script, M_1), (D_1, direct, M_1), (W_1, direct, M_3), (M_3, originate_from, M_4), (M_3, has_actor, A_3)}.

Smaller k values refer to more essential behaviors of the ego node while larger k values include less relevant information. Note that in this work, we choose k to be 2. That says, for each node of interest has only two candidates of unique subgraphs, i.e., its 1-layer subgraph and 2-layer subgraph. Constraining the value k that determines the neighborhood of a node is reasonable since it is usually assumed farer away nodes do not have as significant influences as the closer ones. Moreover, since our goal is to find smallest unique subgraphs, a smaller k is natural to be better.

C. Egocentric Information Table

The k-layer subgraphs of a node are considered as the candidates of unique subgraph. Such candidates should be further validated to be truly unique in the heterogeneous social network. However, since it is infeasible to conduct the subgraph isomorphism test with an acceptable execution time, we alternatively resort to an approximate but efficient manner: if a subgraph candidate cannot be guaranteed to be not unique to any other candidates, then we do not identify it as a unique subgraph. To realize such idea, we have to decompose a subgraph candidate into a list of more fine-grained sub-structures and keep track of the some statistics of the sub-structures. We propose the *Egocentric Information Table* (EIT) to fulfill the idea of sub-structures as well as the corresponding statistics. Before introducing the idea of egocentric information table, we need to define the relational path in a heterogeneous social network.

Definition 4: *Relational Path*. A relational path, denoted by $rp = \langle l_1, l_2, \ldots, l_k \rangle$, is an ordered sequence of edge labels (i.e.,

types) in a heterogeneous social network $H = (V, E, L_V, L_E)$, where $l_i \in L_E$. For example, in Figure 1, one of the relational paths between A_1 and D_1 is $\langle has\_actor, direct \rangle$.

Definition 5: *Egocentric Information Table*. For each k-layer subgraph $S^k[v]$ whose centering at node v, we create an egocentric information table $EIT^k[v]$, which stores a list of pairs of *distinct feature* and the corresponding *feature value*. A feature is defined as a *relational path* rp while its feature value is the number of occurrence of rp in $S^k[v]$. Then an egocentric information table of $S^k[v]$ can be defined as $EIT^k[v] = \{(rp, freq_v(rp))\}$, where rp is each of the length-z relational paths $(z = 1,2, \dots, k)$ starting from node v, and $freq_v(rp)$ is the frequency of rp in $S^k[v]$. Consequently, each node v will have k EITs: $EIT^1[v]$, $EIT^2[v]$, ..., and $EIT^k[v]$, which is corresponding to the k-layer subgraphs $S^1[v]$, $S^2[v]$, ..., and $S^k[v]$.

Note that the number of distinct kinds of relational paths in $EIT^k[v]$ is $|L_E|^k$, where $|L_E|$ is the number of edge types in the social network. The total number of relational paths over all the egocentric information tables is $\sum_{i=1}^{k} |L_E|^k$. Although the space complexity grows exponentially with layer k, as aforementioned, k is chosen to be 2, which reduce much space concern.

We use the toy heterogeneous social network shown in Figure 2 to illustrate the idea of *Egocentric Information Table*, in which there are three kinds of edge types., Take node v_5 as the ego node and assume k=2. The egocentric information tables $EIT^1(v_5)$ and $EIT^2(v_5)$ are shown in Figure 3. For $EIT^1(v_5)$, since all of the three relational paths(i.e., $\langle l_1 \rangle$, $\langle l_2 \rangle$, and $\langle l_3 \rangle$) occur only once in $S^1[v_5]$, all their frequency values equal to1. For $EIT^2(v_5)$, for example, since the relational path $\langle l_3, l_3 \rangle$ occurs three times in $S^2[v_5]$, i.e., $\langle v_5, v_6, v_3 \rangle$, $\langle v_5, v_6, v_7 \rangle$ and $\langle v_5, v_6, v_5 \rangle$, the corresponding frequency is 3.

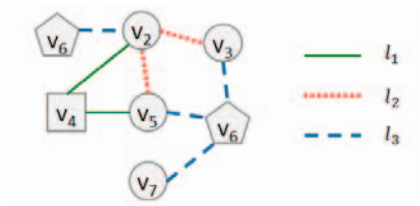


Figure 2: A toy heterogeneous social network.

$EIT^1(v_5)$

$\langle l_1 \rangle$	1
$\langle l_2 \rangle$	1
$\langle l_3 \rangle$	1

$EIT^2(v_5)$

$\langle l_1, l_1 \rangle$	2	$\langle l_2, l_3 \rangle$	1
$\langle l_1, l_2 \rangle$	0	$\langle l_3, l_1 \rangle$	0
$\langle l_1, l_3 \rangle$	0	$\langle l_3, l_2 \rangle$	0
$\langle l_2, l_1 \rangle$	1	$\langle l_3, l_3 \rangle$	3
$\langle l_2, l_2 \rangle$	2		

Figure 3. The egocentric information tables $EIT^1(v_5)$ (left) and $EIT^2(v_5)$ (right) for node $v_5$in the toy heterogeneous social network.

The pseudo code in Algorithm 1 shows the procedure of constructing $EIT^z[v]$ for each node v. We recursively call the function (line 5 to line 12) to count each distinct type of length-z relational path from each ego node v (line 2-3). We can simply repeat the procedure for z=1, 2, 3… k to finish the whole procedure of building all Egocentric Information Table. Besides, in fact we can build $EIT^1[v]$, $EIT^2[v]$, ..., $EIT^k[v]$ by only invoking recursive functions (with z=k) once to avoid duplicated works, but in order to interpret the ideas of EIT better, we choose to build one table per one time.

Algorithm 1. Construct EITs for z-layer Subgraphs

Input: H: a heterogeneous social network; z:layer
Output: $EIT^z[v]$: egocentric information tables for z-layer subgraphs of each node$v$ in H

```
1:   //EIT[v][z] is EIT^z[v]
2:   for v ←1 to |V| do
3:     recursive_count_path(EIT[v],H, v, z, Path={})
4:
5:   Function recursive_count_path(T, H, v, z, Path)
6:     if z = 0 do
7:       T[Path.length].add_one_path(Path)
8:     for each n in H.neighbor_nodes(v) do
9:       Path.add_node(n)
10:      recursive_count_path(T, H, n,z-1, Path)
11:      Path.remove_node(n)
12:  return
```

D. Uniqueness Validation and Discovery

Equipped with the egocentric information table for each candidate of k-layer subgraph, we develop an efficient algorithm to validate the uniqueness of each candidate and approximately find the unique subgraphs for some nodes in the heterogeneous social network. The central idea is to that if all the sub-structures (i.e., relational paths) of a subgraph candidate $S^k[v]$ are *contained by* those of another candidate $S^k[u]$, node v can be considered to be subgraph isomorphic to node u. In other words, if at least one of the relational paths of $S^k[v]$ is not covered by that of $S^k[u]$, node v is said to be unique to node u. Moreover, if $S^k[v]$ is unique to all the other subgraph candidates, $S^k[v]$ is considered to be the final unique subgraph of node v. We fulfill this idea by devising a two-step method. First, to find whether one candidate must not be subgraph isomorphic to all the other subgraph candidates, we compare the values of relational paths between their egocentric information tables in a pairwise manner. Based on the proof at bottom of this sub-section, we can ensure the proposed method can truly answer the subgraph isomorphism between subgraph candidates. Second, we store the pairwise validation results into a *validation topology*, which is a directed graph. The goal of the proposed validation topology is to examine if there is a subgraph candidate guaranteed to be not subgraph isomorphic to any other candidates, then this k-layer subgraph candidate must be unique in the network. In the following we first define the co-center subgraph isomorphism as well as the validation topology. The former is used to represent the subgraph isomorphism between two candidates centered at different nodes, while the latter is used to formally define *validation topology*.

Definition 6: *Co-center Subgraph Isomorphism*. Assume that V and U are the sets of nodes of k-layer subgraphs $S^k[v_i]$ and $S^k[u_j]$ respectively. $S^k[v_i]$ is *co-center subgraph isomorphic* to $S^k[u_j]$ if and only if there exists a subgraph isomorphic mapping $f: V \to U$ such that $f(v_i) = u_j$ and $f(v) = u, \forall v \in V \wedge u \in U$. That says, by enforcing the centers

of two k-layer subgraphs to be aligned, each of the remaining nodes should be able to find a subgraph isomorphic mapping.

Definition 7: *Validation Topology*. The validation topology is a directed graph $T = (V, E^T, f^T)$, in which the node set V is the same as the node set in the original heterogeneous social network H, and $E^T \subseteq V \times V$ is the set of directed edges. An edge $e = (v_i, v_j) \in E^T$ is a directed edge from node v_i to v_j, and $f^T: E^T \rightarrow \{1,2, \dots, k\}$ is a function mapping an edge e to a number z,which is the minimum layer that makes $S^m[v_i]$ must not be a subgraph of $S^m[v_j]$.

The first step of our method is *uniqueness validation*, which aims at testing whether or not a subgraph candidate $S^k[v_i]$ is co-center isomorphism to another candidate $S^k[v_j]$, in which their egocentric information tables are $EIT^k[v_i]$ and $EIT^k[v_j]$ respectively. By examining every relational path in $EIT^k[]$, if there exists a relational path rp such that $freq_{vi}(rp) > freq_{vj}(rp)$, we say that node v_i is ***not*** co-center subgraph isomorphic to node v_j in their k^{th} layer subgraphs. In other words, to ensure the k-layer subgraph candidate of v_i is unique to that of v_j, there must have *at least* one relational path rp of $S^k[v_i]$ that cannot be covered by $S^k[v_j]$.

Lemma 1. *EIT Examination*. *If there is a relational path* rp *such that* $freq_{vi}(rp) > freq_{vj}(rp)$ *in* $EIT^k[v_i]$ *and* $EIT^k[v_j]$, *then* $S^k[v_i]$ *is not co-center subgraph isomorphic to* $S^k[v_j]$.

Proof. This lemma is proved by contradiction. Assume that $S^k[v_i]$ is co-center subgraph isomorphic to $S^k[v_j]$, and there exists one kind of relational path rp *such that* $freq_{vi}(rp) > freq_{vj}(rp)$. Because $S^k[v_i]$ is co-center subgraph isomorphic to $S^k[v_j]$, the frequency of every relational path (i.e. $freq_{vi}(rp)$) starting from node v_j in $S^k[v_j]$ must be larger than or equal to the frequency of that kind of path (i.e., $freq_{vi}(rp)$) starting from node v_i in $S^k[v_i]$, i.e., $freq_{vi}(rp) \leq freq_{vj}(rp)$. It is obvious to contradict with premise of the assumption .The proof is done. ■

The second step of our method is *uniqueness discovery*, which aims to find the unique k-layer subgraphs for some nodes in the heterogeneous social network H. We perform the uniqueness validation process on *every* pair of nodes in the network, and capture the uniqueness between nodes into the validation topology T.We create a node in T for each node in H, construct an directed edge $e = (v_i, v_j)$ from node v_i to v_j with a number assignment $f^T(e) = z$ if there exists a minimum z-layer of subgraph such that v_i is unique to v_j (i.e., $\exists\, rp \in EIT^z[]$, *s.t.* $freq_{vi}(rp) > freq_{vj}(rp)$), where z is increased from 1 to k. If v_i is not unique (i.e., co-center subgraph isomorphic) to v_j for all the layers of interest (i.e., from 1st to k^{th} layer), no edge will be constructed between them. Based on the validation topology T constructed, we can identify a set of nodes, whose z-layer subgraphs $S^z[v_i]$ are unique, by the following two requirements: (a) $outDegree(v_i) = |V| - 1$and (b) $\min_{v_j \in V \setminus v_i} f^T((v_i, v_j)) = z$. The first condition refers to that if $S^z[v_i]$ is not a subgraph of any other subgraph candidates in H, it must be unique. The second condition indicates that to have smaller unique subgraphs, we select the subgraph candidates with the minimum layer number z to be the result. It is apparent that there could be only a small set $U$of nodesin H satisfying such two requirements, while we cannot find the unique subgraphs for most of other nodes in $V \setminus U$ from their k-layer subgraph candidates. We will deal with the discovery of unique subgraphs for nodes in $V \setminus U$ in the next sub-section.

Lemma 2. *Small Unique Subgraph Selection*. *Subgraph candidates* $S^z[v_i]$ *satisfying two requirements in the validation topology T are small unique subgraphs:* (a) $outDegree(v_i) = |V| - 1$ *and* (b) $\min_{v_j \in V \setminus v_i} f^T((v_i, v_j)) = z$.

Proof. This lemma is proved by contradiction. Assume $S^z[v_i]$ is a subgraph satisfying the two requirements but it is not a unique subgraph. Therefore $S^z[v_i]$ is subgraph isomorphic to $H - S^z[v_i]$ according to the definition of uniqueness. As a result, there exists a subgraph$S^z[v_x]$ in $H - S^z[v_i]$ such that $S^z[v_i]$ is co-center subgraph isomorphic to and there a mapping such that $f(v_i) = v_x$. It means that all the frequency values of rp in $EIT^z(v_x)$ are larger than or equal to those in $EIT^z(v_i)$. Consequently there will be no directed edge from v_i to v_x in the validation topology. As a result, the fact $outDegree(v_i) < |V| - 1$ contradicts to the definition of $S^z[v_i]$. ■

Algorithm 2. Uniqueness Validation and Discovery

Input: H: a heterogeneous social network; $EIT^z[v]$: egocentric information tables for each node v in H and for $1 \leq z \leq k$.
Output: US: the list of seed unique subgraphs

```
1:  T ← an directed graph with the same node set V of H,
2:          and edge set is empty
3:  for v1 ← 1 to |V| do
4:     v1NotUnique ← true
5:     for v2 ← 1 to |V| do
6:        for z ←1 to k do
7:           if table_compare(v1, v2, z)=true do
8:              T.add_edge(v1, v2, m=z)
9:              v1NotUnique ← false
10:             break
11:       if  v1NotUnique=true do
12:          break
13:
14: Function table_compare(v1, v2, z)
15:       for each rp in EIT^z[v1] do
16:          if freq_v1(rp) > freq_v2(rp) do
17:             return true
18: return false
19:
20: US = []
21: for v that T.out-degree(v) = |V| - 1 do
22:       m ← infinite
23:       for each e in H.OutGoingEdges(v) do
24:          m = min(m, T[e].m)
25:       US.add( NLayerSubgraph(H, v, m) )
26: return US
```

The pseudo code in Algorithm 2 shows the procedure of validating the uniqueness of each candidates. We compare two EIT for each possible pairs of nodes (line 3-12) and add edges

in Validation Topology if some subgraph must be unique to another subgraph. The procedure of comparing two EIT (line 14-17) is to compare the frequencies (i.e.times) of each relational path rp in that layer. If there exist one kind of relational path that the number of it in $S^z[v_i]$ is larger than the number of it in $S^z[v_j]$, then $S^z[v_i]$ must be unique to $S^z[v_j]$. After adding all necessary edges on VT, we can find out all unique subgraphs seeds (line 21-26) by counting the out-degree of certain node in VT. Having this unique subgraphs in hand, we can further expand them in the next step.

Algorithm 3. Finding unique subgraph for query node q

Input: H: a heterogeneous social network; q: the query node; US: the list of seed unique subgraphs
Output: US_q: the unique subgraph containing query node q.

```
1:  US.sort(key=us.numOfNodes(), ascending);
2:  queue = []
3:
4:  for each n in H.nodes() do
5:      H.node[n]['size'] = infinite
6:
7:  for each us in US do
8:      for i ←1 to |US| do
9:          if H.node[n]['size'] != infinite do
10:             H.node[n]['size'] = US[i].numOfNodes()
11:             H.node[n]['pre'] = i
12:             queue.enq(n)
13:
14: while(!queue.empty()) do
15:     now = queue.deq()
16:     size = H.node[now]['size']
17:     for each n in H.neighbor_nodes(now) do
18:         if(H.node[n]['size'] > size + 1) do
19:             H.node[n]['size'] = size + 1
20:             H.node[n]['pre'] = H.node[now]['pre']
21:             queue.enq(n);
22:
23: Function trace_back_subgraph(v)
24:     g = graph()
25:     pre = H.node[v]['pre']
26:     if pre <= |US| do
27:         g.extend(US[pre])
28:         return g
29:     else do
30:         g.add_edge(v, pre)
31:         return trace_back_subgraph(pre)
32:
33: return US_q= trace_back_subgraph(q)
```

E. Finding Unique Subgraphs

After testing the uniqueness of each subgraph candidates, we can obtain unique subgraphs for some nodes. However, those nodes which have unique subgraphs could find smaller unique subgraph, and rest of nodes cannot directly find unique subgraph. As a consequence, in this step, we not only aim at finding smaller unique subgraph for those nodes which already have unique subgraph, but also aim at finding unique subgraph for those nodes which have no unique subgraph originally.

By the definition of uniqueness in this paper, it is not difficult to prove that if we add nodes and edges to a unique subgraph, the new graph is still a unique subgraph. As a result, we take those unique subgraphs found in previous step, and adapt BFS (i.e. Breadth-First-Search) techniques to expand those unique subgraph seeds to whole graph.

First, for each node v we store two number (p, s), where p is the index of the unique subgraph that node v connect to, and s is the current minimal size(i.e. number of node) of new unique subgraph which contains the connected unique subgraph seed and the path to the candidate. As initialization, from largest to smallest unique subgraph seeds $S^z[v_i]$, we set s of each node in the seed graph to be Size($S^z[v_i]$) and a to be v_i, where Size(g) is the number of nodes in graph g. Second, we put all the nodes in unique subgraph seeds into the BFS's queue, and the order is from smallest subgraph seeds to largest subgraph seeds. After pushing nodes into queue, we start to run BFS algorithm. For each iteration, a node v is pulled from the queue, and then we use node v to walk outward to update (a, s).

The pseudo codes in Algorithm 3 shows the procedure of finding unique subgraph for each query node q. First, we initialize the minimal size s for the nodes which are not covered by unique subgraph seeds to be infinite and for the rest nodes to be the minimal size of unique subgraphs (line 1-12). Besides, we put the nodes of unique subgraphs into the queue for later usage. Second, we deploy BFS algorithm to expand unique subgraph seeds (line 14-21). Finally, we trace back the path from query node q to certain unique subgraph as the final result (line 23-33). Note that the subgraph which expands from a unique subgraph seed is still a unique subgraph because of the definition of uniqueness.

IV. Smallest Unique SubTree

In this section, we aim to prove the hardness of the proposed *Smallest Unique Subgraph* (SUS) problem in a heterogeneous social network. We approach the proof through relaxing the original SUS problem into a less complex graph setting that leads to a simpler problem: finding *Smallest Unique SubTree* (SUS-T) in a homogeneous social network. If we can prove that solving the SUS-T problem in a homogeneous network is NP-complete, since heterogeneous newtorks are more complex than homogeneous, and the fact that there is polynomial solution to determine whether a subtree is unique, we can say solving SUS problem is NP-complete as well. In the following we first define the SUS-T problem. Then the proof is approached through a reduction from *Minimum Vertex Cover* (MVC) problem.

A. Problem Statement

Problem Definition: *Smallest Unique SubTree* (*SUS-T*) *problem. Given a heterogeneous connected tree H^T and a particular node of interest v, the SUS-T problem aims to find an induced subtree $S \subseteq H^T$ such that* (a) $v \in S$, (b) *S is unique in H^T, and* (c) *there is no other induced subtree S', whose size is less than S, satisfying the previous two requirements.*

Since there exists an efficient algorithm to do the *subtree isomorphism test* [1], which can be applied to determine the uniqueness of subtree candidates, it might be natural to ask whether or not SUS-T can be solved efficiently as well. Unfortunately, in the following, we deduce that SUS-T is not a sample problem in general. In fact, both determining whether or not a subtree contains the query node and whether or not a

subtree is unique can be tackled in a polynomial time. However, when it comes to finding the *smallest* subtree among the candidates, there is no trivial solution. We validate such deduction by doing a reduction from *Minimum Vertex Cover* to SUS-T. Note that in fact SUS-T could be practical because trees can be generated from models of information diffusion, such as *Independent Cascade* (IC) model [18] and *Linear Threshold* (LT) model [19], in a heterogeneous social network. Hence, the mined smallest unique subtrees can be considered as the most informative pathways of ego nodes of interest.

SUS-T is a NP problem since we can have polynomial-time algorithms to determine both whether or not a subtree contains the query node and whether or not a subtree is unique in a heterogeneous connected tree. The former containment test can be achieved through simply scanning nodes in the subtree with linear time, while the latter uniqueness test needs to resort to the subtree isomorphism algorithm between a subtree S and its complement tree H^T–S. Therefore, we can say that subtrees satisfying the first two requirements can be found through non-deterministically testing on every possible subtree candidates. In other words, we can have polynomial-time solutions to determine whether or not a subtree of an ego is unique.

On the other hand, it is trivial and apparent that solving the SUS-T problem in a homogeneous social network (termed *homogeneous SUS-T*) can be reduced into solving SUS-T in a heterogeneous one termed *heterogeneous SUS-T*), through adding types/labels to nodes and edges. Consequently, any problems that can be reduced into homogeneous SUS-T can be reduced into heterogeneous SUS-T as well. Our ultimate goal aims to show that *Minimum Vertex Cover* (MVC), which had been proven to be a NP-complete problem, can be reduced to homogeneous SUS-T problem in a polynomial manner. If such reduction can be made, we can indirectly conclude that the original SUS problem is NP-complete.

B. Proof Sketch for the Reduction from MVC to SUS-T

Given an arbitrary MVC problem over a graph G with n nodes and m edges, with loss of generality, we give an index for each node v_1 to v_n, such that each node can be uniquely identified in the graph. We use G to construct a special connected tree (termed by *S-Tree*), which is designed to provide a mapping between MVC and homogeneous SUS-T. Specifically, the special tree *S-Tree* is required to raise a substitute solution to finding a minimum set of nodes such that finding the smallest unique subtree in *S-Tree* can yield the minimum set of nodes covering G. In the following we will first describe the construction of *S-Tree* for homogeneous SUS-T from the graph in the given MVC. Then based on the constructed *S-Tree*, we will show the required structure that has to be included in a unique subtree of homogeneous SUS-T. Finally we elaborate how to create the mapping between the given MVC and the SUS-T. Note that due to the space limit, we do not provide the detailed description about all the lemmas (lemma 3 to lemma 11).

1) *S-Tree Construction*

The *S-Tree* is constructed from the given graph G with n nodes and e edges in the MVC problem. *S-Tree* is designed to have two major components, *family* and *arm*. A *S-Tree* contains a number of *families*, where each of which is a tree structure. Each *arm* is a long path connecting two *families*. *Family* can be further divided into three categories of structures, (a) *target*, (b) *constraint*, and (c) *decision*, with different objectives. We will first introduce the basic components of a *family* and describe such three categories of *family*.

- **Family.** A *family* is a tree structure that contains four types of nodes *root*, *point*, *chosen*, and *index*. An illustration is shown in Figure 4. Each *family* contains one *root* node, at most n *point* nodes, at most n*chosen* nodes, and less than or equal to $n^2 + n$ *index* nodes. First, among these four types of nodes, *point* nodes are designed to have a mapping between *S-Tree* and nodes in the given graph of MVC. Second, each *chosen* node, which is connected to a *point* node, is used to mark whether or not such *point* node is selected in MVC. Third, the design of *index* structure is to provide a unique signature for each *point* node, i.e., each node in the given graph of MVC. The *index* structure of each *point* node is a tree with at most $n + 1$ nodes. *Index* structures of a family are derived by varying the depth from 1 to n and the branch from n to 1 at the same time. An illustration is shown in Figure 4. Finally, the *root* node is created to make those nodes in a *family* structure be connected. There are three categories of *family* structure, *target*, *constraint*, and *decision*, as elaborated in the following.

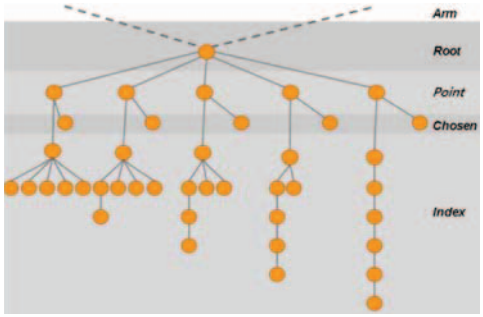
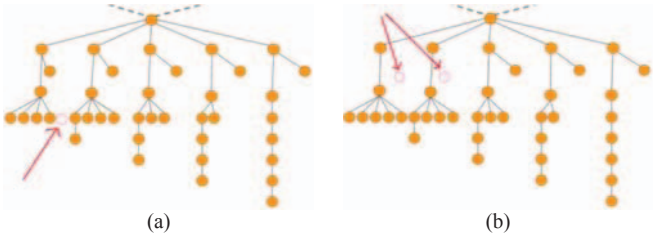


Figure 4: An illustration to the basic structure of a *Family*, which contains 41 nodes with 5 *point* nodes in MVC. In addition, there are two arms which are used to connect this family to other families.

- **Target.** As the most important one of the families, a *target* appears exactly the same as the original *family* structure mentioned above. An illustration is shown in Figure 4. A *target* family is the super-tree of any other *families*. A *target* contains one *root* node, n *point* nodes, n*chosen* nodes, and n*Index* structures. A *S-Tree* must contain exactly one *target*. Note that the *target* family could also be the smallest unique subtree in *S-Tree*.

- **Constraint.** A *constraint* family is nearly the same as *target* except for that one node is removed from the *index* structures. The removed node is restricted to be one of the leaf nodes in the *index* structure. Depended on various ways of leaf removal, for a *S-Tree*, there is a total of $2n - 2$ *constraints* families. And these *constraints*, though will not be considered as a part of the unique subtrees, aim to help regulate the structure of the potential unique subtrees.

- **Decision.** The goal of the design of *decision* is to reflect all the possible selections of nodes in the MVC problem. Therefore, each node in the given graph of MVC has a one-to-one mapping to a *point* node in a *decision* family. A *decision* family is nearly the same as *target* as well. The only difference is that a pair of *chosen* nodes is removed from the *target* family. A subtree of *S-Tree* possessing a missing edge of a pair of *chosen* nodes stands for that the edge of the corresponding *point* nodes in MVC is covered. In other words, *S-Tree* needs to contain m *decision* families that indicate all the possible combinations of edges cover in the graph of the MVC problem.



(a) (b)

Figure 5: An illustration to (a) the *constraint* family and (b) the *decision* family, with respect to the *target* family in Figure 4.

- **Arm.** An arm is a long path with $2 \times N_{tar}$ nodes, where N_{tar} is the number of nodes in a *target*. Both end nodes of an arm are connected to the *root* nodes of two family structures that could be *target*, *constraint*, or *decision*. We create arms to connect all the families such that *S-Tree* is constructed to be connected. Figure 6 shows a snapshot illustration of using arms to connect families in a *S-Tree*.

Figure 6: An illustration to *arms* connecting families in a *S-Tree*.

2) *Minimum Required Subtree in S-Tree*

Conceptually we intend to solve the homogeneous SUS-T problem in the constructed *S-Tree* and to prove that finding the smallest unique subtree in *S-Tree* is equivalent to selecting a smallest set of nodes covering all the edges of the graph in MVC. Here we aim to show that there is a *minimum required subtree R* that must be included in any possible unique subtrees. We first show the size and the required elements of the *minimum required subtree R*. This can be done by proving that the smallest unique subtree in *S-Tree* must not only have less than or equal to N_{tar} nodes and but also contain at least a node in the *target* family (Lemma 3 and 4). Second, we show that the *minimum required subtree R* must contain all the nodes of *root*, *point*, and *index* in the *target* family, through subtree matching among *target* and *constraint* families (Lemma 5 and 6). Note that since both *target* and *constraint* families contain all the *chosen* nodes, *chosen* nodes are not included in *R*. In addition, we also show that there is a restrict *root*-to-*root*, *point*-to-*point*, *index*-to-*index*, and *chosen*-to-*chosen* mapping between two subtrees (Lemma 7). Moreover, we show that it is unnecessary to include *arms* in *R* (Lemma 8). Finally, we show that any super-trees of the minimum required subtree have no mapping to any subtrees in the *constraint* families (Lemma 9). In other words, we do not need to worry about the potential mapping from *constraint* families as long as we keep the minimum required subtrees as a subtree of the smallest unique subtree. Consequently, what we only need to investigate is the mapping from subtrees, which are also the super-trees of the minimum required subtree *R*, in the *target* family to those in the *decision* families.

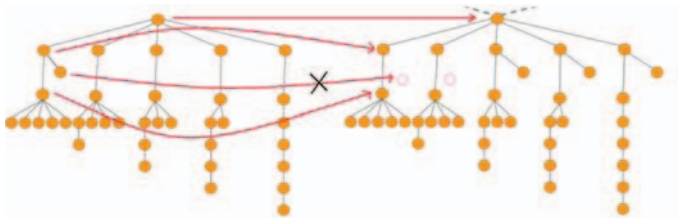


Figure 7. The rules of mapping from root to root, point to point, chosen to chosen, and index to index.

3) *Problem Mapping*

We aim to show the equivalence mapping between the given MVC and the homogeneous SUS-T in the constructed *S-Tree*. Based on the analysis of the minimum required subtree *R*, we need to come up with the subtree mappings from the *target* family to the *decision* families such that the subtree is unique. In other words, we aim to extend the minimum required structure to be a certain subtree *T** by adding *chosen* nodes such that *T** is unique. Recall the mapping restriction that *chosen* nodes are required to be matched to the corresponding *chosen* nodes from the *target* to a *decision* family (Lemma 7). We can show that if we do not want the unique subtree *T** to be a subtree of a *decision* family, *T** must contains at least one of the missing *chosen* node of that *decision* family (Lemma 11). In other words, if we are constructing a unique subtree *T'* from the minimum required structure *R*, *T'* is required not to be a subtree of any other families of *constraint* and *decision* (i.e., have no mappings between *T'* and any other families). Since the unique subtree *T'* must be the super-tree of *R*, its mapping to another family must follow the *root*-to-*root*, *point*-to-*point*, *chosen*-to-*chosen*, and *index*-to-*index* rules, as shown in Figure 7. Therefore, if *T'* contains a *chosen* node v_c, we will have no mapping from *T'* to the *decision* families without such *chosen* node v_c. Because the pair of missing *chosen* nodes in a *decision* family can be mapped to the two nodes connected by an edge in the graph of MVC, adding any of the missing *chosen* nodes to *R* means at least one of the two nodes is selected in MVC, and thus covering the edge (Lemma 10). Then since a *chosen* node missed in a *decision* family refers to covering an edge in the given graph of MVC, we can say that adding *chosen* nodes into the minimum required structure *R* is identical to covering edges in the graph of MVC. Eventually we turn to answer both the problems of MVC and homogeneous SUS-T by examining the rule that a *chosen* node in homogeneous SUS-T is added if and only if the corresponding *chosen* node is selected in MVC.

As a result, in order to be unique and smallest, all the potential mappings from subtrees in the *target* family to those in *decision* families should be examined by adding the least one *chosen* node. Such action is equivalent to finding a smallest set of nodes that cover every edge in MVC.

V. EXPERIMENTS

We aim to exhibit the performance of the proposed EGH approach. The main goal is to see (a) whether or not the run time of EGH is efficient, and (b) whether or not the sizes of the identified smallest unique subgraphs by EGH are small enough. We compare EGH the brute-force approach with different parameter settings. Details are provided in the following.

A. Dataset

The heterogeneous social network is constructed from extracting entities and relations from UCI KDD Archive movie dataset [13]. In this network, there are about 24,000 nodes representing movies (9,097), directors (3,233), actors (10,917), and some other movie-related persons (500) such as producers and writers (the numbers in parentheses show the number of different instances for each node type). We also extract 126,926 relations between these nodes. Totally, there are 44 different relation types in the movie network, which can be divided into three groups: relations between people (e.g. spouse and mentor), relations between movies (e.g. remake), and relations between a person and a movie (e.g. director and actor). The amount of diverse relations makes it a complicated heterogeneous social network for humans to analyze.

B. Evaluation Settings

We compare the proposed EGH to a brute-force approach which applies VF2 [17]. In brute-force approach, we enumerate all the possible subgraph candidates whose size is smaller than M (M=4 in our experiments), and employ VF2 for the subgraph isomorphism test between each pair of subgraph candidates. Due to the limitation of pages, details of the brute-force approach are not provided here. Those identified to be unique are sorted by their sizes, and the smallest one is considered as the resulting smallest unique subgraph for node v. Due to the high complexity, we restrict the size M of the enumerated subgraph candidates to be 1, 2, 3, and 4. That says, we compare the EGH with the number of layers L=1 and L=2 to the brute-forced approach with the size of subgraph candidates M= 2, 3, and 4.

We evaluate the performance from two perspectives. The first is the time efficiency (in second) that is required to complete the process of smallest unique subgraph identification by EGH and the brute-force methods. We expect the run time of EGH is drastically shorter than the brute-force method. The second is the size of the mined smallest unique subgraph. If the size of identified subgraphs by EGH is close to those found by the brute-force, the effectiveness of EGH can be guaranteed.

Also due to the high complexity of the brute-forced method, it is infeasible for us to use the entire heterogeneous social network for the experiments. Instead we sample connected node-induced subgraphs from the original heterogeneous social network, whose sizes range from 100 to 600 respectively. In order to reduce the variance of performance incurred by sampling, we conduct subgraph sampling up 20 times and report the average values for each evaluation measure.

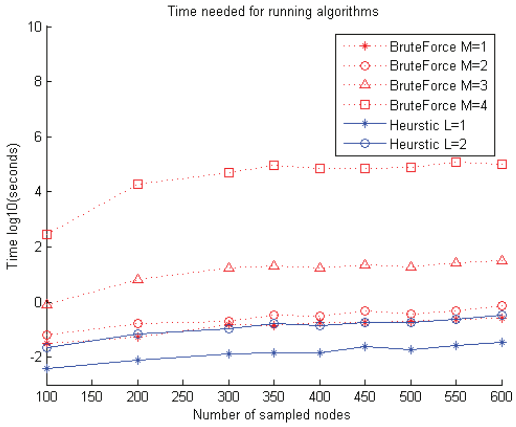


Figure 8: The average run time of EGH with L=1 and L=2 and the brute-force method with M= 2, 3, and 4, by varying the number of sampled nodes.

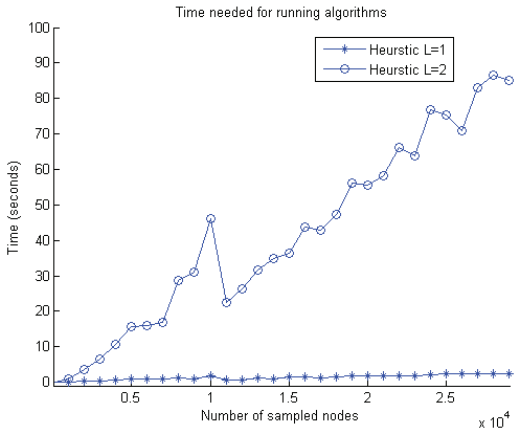


Figure 9: The average run time of EGH with L=1 and L=2, by varying the number of sampled nodes in the heterogeneous social network.

C. Experimental Results

Figure 8 shows that the average run time of EGH is apparently much lower than that of the brute-force approach, as the number of sampled nodes grows. Since the y-axis of Figure 8 is log-scaled, the run time is not linear but exponential. Therefore, EGH can be said to be singnificantly efficient, comparing to the brute-force. On the other hand, as the size of subgraph candidate increases, the run time of the brute-force method raises drastically. The run time of our EGH with the two-layer setting (L=2) is even competitive to the brute-force method with single-node subgraph candidate (M=1). Since EGH is very efficient, we wonder how the average run time increases when the number of sampled nodes grows far away from 600. We vary the number of sampled nodes from 1,000, 2,000, 3,000, ..., 28,000, and present the run time in Figure 9. We can find that for L=1, the run time is nearly fixed no matter how the number of sampled nodes goes up. As for L=2, though the growth in run time is linear, it is still under 100 seconds even when the sampling size is up to 28,000. Therefore, we can say EGH is able to efficiently output the unique subgraphs.

We present the average size of the identified subgraphs by EGH and the brute-force method, when the number of sampled

nodes increases. The results are shown in Figure 10. It is natural to find that the average size of unique subgraphs found by EGH is larger than that of the brute-force method. Nevertheless, the size of uniques subgraph output by EGH is around 10 to 20 in general. Such sizes are acceptable when we would to visualize the the unique subgraphs. In addition, though the unique subgraphs produced by the brute-force method is smaller (around 4 to 7 in general), their run time is much higher than EGH. The unacceptable run time of the brute-force method makes the identification process inpractical. Therefore, the proposed EGH can be an efficient alternative to find the unique and smaller subgraphs for practical uses.

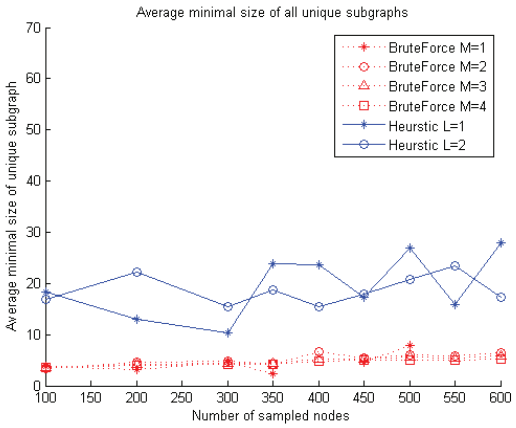


Figure 10: The average size of the identified unique subgraphs for EGH with L=1 and L=2 and the brute-force method with M= 2, 3, and 4, by varying the number of sampled nodes.

VI. Related Work

MSUS can be seen as a satellite problem to graph pattern mining (GPM). There are researches solving this problem in perspective of collecting frequent or rare patterns and separating collections of patterns [6]. For example in [7], a set of frequent patterns are collected with a random sampling and pattern clustering method. In [8] a pruning based system is constructed to filter out the less useful patterns from the pattern set. And in [9], a measurement is introduced to help cluster the patterns from different classes. However, as MSUS only want to decide whether any mapping of the pattern exists or not instead of building a pattern set, most of the systems in those previous works cannot be applied onto this problem.

For the goal of MSUS is to identify the anomalous pattern from an identity, it makes a similar idea to anomalous structure mining in anomaly detection (AD). In [15], the system collects the patterns which are similar to some predefine extreme structures. And in [16], a similarity measurement is setup in purpose to allocate the patterns far from the general cases. However in MSUS, what we want to find is the exact mappings of the patterns but not some similar cases. So there should be a more precise but still efficient measuring to the difference of each identities in the social network.

VII. Conclusion

In this research, to analyze the potential of obtaining of uncommon social structure around any identity, we invest into studying the unique patterns in a social network. In sight of compactness, the unique patterns are preferred to be smaller better. But we verify that finding the smallest unique patterns is hard as long as it is still connected even if the social network is simplified through removing the less important connections. We then propose an ego-graph heuristic method which can gather small unique patterns efficiently. In our experiment on a real movie heterogeneous social network, it is shown that the heuristic method can remain efficient even when the scale of data is up to about 30,000 nodes, while the average size of the pattern is only about a constant factor to the brute-force system.

Acknowledgement

This work was sponsored by Ministry of Science and Technology (MOST) of Taiwan under grant 104-2221-E-001-027-MY2. This work is also supported by Multidisciplinary Health Cloud Research Program: Technology Development and Application of Big Health Data, Academia Sinica, Taipei, Taiwan under grant MP10212-0318.

References

[1] Shamir, R. and Tsur, D. “Faster subtree isomorphism.” Journal of Algorithms, 33:267-280, 1999.

[2] Scott, J. “Social network analysis.” Sociology 22(1): 109-127, 1988.

[3] Law, J. “Notes on the theory of the actor-network: Ordering, strategy, and heterogeneity.” Systems practice 5(4): 379-393, 1992.

[4] Hassanzadeh, R., Nayak, R., and Stebila, D. “Analyzing the effectiveness of graph metrics for anomaly detection in online social networks.” In WISE, 624-630, 2012.

[5] Zhou, B, and Pei, J. “Preserving privacy in social networks against neighborhood attacks.” In IEEE ICDE, 506-515, 2008.

[6] Jiang, C., Coenen, F., and Zito, M. “A survey of frequent subgraph mining algorithms.” The Knowledge Engineering Review 1(1): 1-31, 2013.

[7] Zhang, S., Yang, J., and Li, S. “Ring: An integrated method for frequent representative subgraph mining.” In IEEE ICDM, 1082-1087, 2009.

[8] Thoma, M., Cheng, H., Gretton, A., Han, J., Kriegel, H., Smola, A., Song, L., Yu, P. S. and Yan, X., Borgwardt, K. M. “Discriminative frequent subgraph mining with optimality guarantees.” Statistical Analysis and Data Mining, 3(5): 302-318, 2010.

[9] Dhifli, W., Saidi, R., and Nguifo, E. M. “Mining Representative Unsubstituted Graph Patterns Using Prior Similarity Matrix.” In Data Mining meetings (GDR I3), 2013.

[10] Cook, S. A. “The complexity of theorem-proving procedures.” In STOC, 151-158, 1971.

[11] Heggernes, P., Hof, P., and Milanic, M. “Induced Subtrees in Interval Graphs.” In IWOCA, 230-243, 2013.

[12] Lappas, T., Liu, K., and Terzi. E. “Finding a Team of Experts in Social Networks.” In ACM KDD, 467-476, 2009.

[13] Li, C.-T. and Lin, S.-D. “Egocentric Information Abstraction for Heterogeneous Social Networks.” In IEEE/ACM ASONAM, 255-260, 2009.

[14] Backstrom, L., Dwork, C., and Kleinberg, J. “Wherefore Art Thou R3579X? Anonymized Social Networks, Hidden Patterns, and Structural Steganography.” In WWW, 181-190, 2007.

[15] Akoglu, L., McGlohon, M., and Faloutsos, C. “Oddball: Spotting anomalies in weighted graphs.” In PAKDD, 410-421, 2010.

[16] Heard, N. A., Weston, D. J., Platanioti, K., and Hand, D. J. “Bayesian anomaly detection methods for social networks.” The Annals of Applied Statistics 4(2): 645-662, 2010.

[17] Cordella, L. P., Foggia, P., Sansone, C., and Vento, M. “A (sub) graph isomorphism algorithm for matching large graphs.” IEEE Transactions on Pattern Analysis and Machine Intelligence 26(10): 1367-1372, 2004.

[18] Chen, W., Wang, C., and Wang, Y. “Scalable influence maximization for prevalent viral marketing in large-scale social networks.” In ACM KDD, 1029-1038, 2010.

[19] Chen, W., Yuan, Y., and Zhang, L. “Scalable influence maximization in social networks under the linear threshold model.” In IEEE ICDM, 88-97, 2010.