Methodology for Identifying Social Groups within a Transactional Graph

Maxence Morin[1,2][0009-0006-3789-9904], Baptiste Hemery[2][0000-0003-3923-1829], Fabrice Jeanne[2], and Estelle Pawlowski-Cherrier[1][0009-0008-9872-3667]

[1] Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France
`maxence.morin@unicaen.fr`
`estelle.pawlowski@ensicaen.fr`
[2] Orange Innovation, 14000 Caen, France
`{baptiste.hemery,fabrice.jeanne}@orange.com`

Abstract. Social network analysis is pivotal for organizations aiming to leverage the vast amounts of data generated from user interactions on social media and other digital platforms. These interactions often reveal complex social structures, such as tightly-knit groups based on common interests, which are crucial for enhancing service personalization or fraud detection. Traditional methods like community detection and graph matching, while useful, often fall short of accurately identifying specific groups of users. This paper introduces a novel framework specifically designed to identify groups of users within transactional graphs by focusing on the contextual and structural nuances that define these groups.

Keywords: Social Network Analysis · Transactional Graphs · Community Detection · Graph Matching · SubGraph of Interest.

1 Introduction

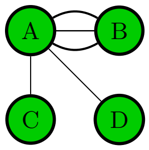
Social network analysis is subject to increasing attention. Relational data are obtained from the interactions of users of a service. They are of great value. Whether it comes from social media, telecommunications services, or financial transactions, companies in these sectors have understood the importance of extracting information from relational data. They exploit information to enrich and personalize services, improve trust by combating fraud and improper use, and so on. A common method to analyze relational data is to transform data into transactional graphs. Nodes represent users, and edges represent transactions. This abstraction allows us to solve some issues in social network analysis thanks to graph theory.

In these graphs, we search for groups of users that present interesting characteristics. Users exchange in a manner that creates communities. These can be identified using community detection algorithms. However, in this paper we are not interested in community detection. We are looking for other specific types of groups in transactional graphs.

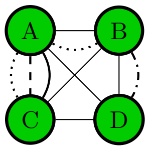
We define a new concept that we call Group of Interest (GoI). A GoI is a social group composed of users who share social relations. The difference between communities and GoI is explained in the following paragraphs. As with communities, GoI can overlap. For example, a user can be a member of a family and a member of a group of fraudsters at the same time. We define the term "social truth" to encompass all social groups present in transactional graphs. In the real world, we cannot know the entirety of the social truth. Interviewing people costs money. Thus, we only know a part of the social truth that we call *samples*.

We refer to all the exchanges observed in a service as the "transactional truth". Transactions are traces of underlying social relationships. However, transactional truth is bound by a period of time. This restriction makes transactional truth incomplete, and it may reflect a part or the entirety of the social truth. If we extend the period of time, every social relationship may induce a transaction. In this paper, we aim to uncover the social truth by using only the transactional truth.

One could naturally think of traditional community detection methods to solve this specific problem, but these methods are not sufficient for two main reasons. The first reason is that these algorithms identify communities, which are groups of nodes that are highly interconnected relative to the rest of the graph. The transactions induced by a GoI do not create a highly connected subgraph. For example, in the subgraph shown in Figure 1a, each node $A, B, C,$ or D does not engage in transactions with everyone. We are far from having a densely connected subgraph. Nonetheless, the users represented by these nodes may know each other and all share a relationship. The underlying relationships may be dense, as illustrated in Figure 1b. The second reason is that community detection methods cover the entire graph and attempt to identify communities of any type, whereas we are interested in only specific subgraphs.



(a) Observed exchanges in a transactional service



(b) Underlying social relationships between the users. Different types of social relationships are represented by different line styles.

Fig. 1: Differences between the observed transactional truth and the underlying and inaccessible social truth.

The second natural solution one might also think of could belong to the graph matching research domain. Graph matching methods usually require a query to find topologically equivalent subgraphs. But the query does not consider any context surrounding it. We consider that the context is important to increase the performance of the model. Nevertheless, community detection methods as well as graph matching methods can represent a part of our solution to social truth identification. We give some details in Section 3.1 to adapt these methods in a new and complete framework.

In this paper, we introduce the notion of a SubGraph of Interest (SGI). Furthermore, we propose a new framework that encompasses the social truth identification problem and formalizes several methods for solving it. We also propose a method to evaluate the framework's performance. The experimental evaluation of the framework will be presented in a further paper.

The rest of this paper is organized as follows. Section 2 is devoted to the problem statement. The framework is presented in Section 3. Section 4 describes how to evaluate the framework. Conclusions and further work are discussed in Section 5.

2 Problem Statement

2.1 Notations

Let $G = (V_G, E_G)$ be a multigraph comprised of a set of nodes V_G and edges E_G. Each vertex $v \in V_G$ represents a user. Each edge $(u, v) \in E_G$ denotes a relationship such as a financial transaction, a communication, or a friendship. Let $S = (V_S, E_S)$ be a subgraph, which implies $S \subseteq G$ if $V_S \subseteq V_G$ and $E_S \subseteq E_G$.

2.2 Problem Formalization

We define a new concept that we call SubGraph of Interest (SGI). An SGI is a subgraph that represents the transactions made by a GoI. Let $\mathcal{S}_{type}$ be an SGI that represents transactions of a certain type of GoI. Similar to GoIs, SGIs can overlap. Thus, a node can belong to several SGIs. Let $\mathbb{S}_{type}$ be the set containing all the $\mathcal{S}_{type}$ of a unique type. The set $\mathbb{S}_{type}$ is representing one GoI type present in the social truth. The type can be families, groups of colleagues, fraudsters, and so on. For the sake of clarity, we use $\mathbb{S}$ and $\mathcal{S}$ instead of $\mathbb{S}_{type}$ and $\mathcal{S}_{type}$ as we are only interested in a unique type of GoI at a time. Recovering the set $\mathbb{S}$ for each type means, in the end, recovering the entire social truth.

Let $\mathbb{S}_n \subset \mathbb{S}$ be the set of known $\mathcal{S}$ in $\mathbb{S}$ that represents our *samples*. The set $\mathbb{S}_n$ has a fixed size n. These n subgraphs are used as training samples. As mentioned in the introduction, we have to deal with a set $\mathbb{S}_n$ containing very few samples.

The aim of this paper is to recover the entire set $\mathbb{S}$ using only the training sample set $\mathbb{S}_n$ for one given type. The contribution of this paper is twofold:

1. Propose a framework to tackle the previously mentioned problem of social truth identification.
2. Evaluate this framework.

3 Framework Definition

Given a multigraph G and a set of samples $\mathbb{S}_n$, we want the framework to output a set $\hat{\mathbb{S}}$. $\hat{S}$ stands for the elements of $\hat{\mathbb{S}}$. We want this set $\hat{\mathbb{S}}$ to be as close as possible to the set $\mathbb{S}$.

We illustrate the framework as follows:

$$(G, \mathbb{S}_n) \rightarrow \boxed{method} \rightarrow \hat{\mathbb{S}} \text{ such that } \hat{\mathbb{S}} \simeq \mathbb{S} \tag{1}$$

The relation $\hat{\mathbb{S}} \simeq \mathbb{S}$ means that we want the set $\hat{\mathbb{S}}$ to be as similar as possible to the social truth $\mathbb{S}$ (which is only part of the social truth, i.e. for a specific type). We precise this point in Section 4.

Now we define three *method* proposals in framework (1). The first two *methods* share the same basis and are gathered in the first approach, detailed in Section 3.1. The last *method* relies on a new pruning approach, detailed in Section 3.2.

3.1 First Approach

The *method* of the first approach consists of three steps.

1. The first step is a function named SUBGRAPHS. This function takes a multigraph G as input and must return a set of clusters. The role of this function can be fulfilled by already existing functions. We chose functions from either community detection or graph matching research domains.
2. The second step is the FEATURES function. This function takes a subgraph as input and should return a fixed-size vector that encodes information about the given subgraph. Details about this function are provided in a further paragraph.
3. The last step of the first approach is the selection step. The clusters returned by the SUBGRAPHS function are not necessarily relevant. We need to select those that are similar enough to the available *samples* in $\mathbb{S}_n$.

We present this FIRSTAPPROACH function through the Algorithm 1. We go through the details of each step now.

Subgraphs Function The role of the SUBGRAPHS function is to create subgraphs from the given multigraph G. It seems natural to think of community detection and graph matching methods as good functions for creating clusters.

On the one hand, community detection algorithms [4] yield subgraphs, also referred to as communities. They identify either overlapping or non-overlapping communities. Because SGI can overlap, we are interested in overlapping community detection.

Among the overlapping community detection algorithms, we find algorithms using edge betweenness [6] or modularity based algorithms [20]. We propose to use an overlapping variant of the Label Propagation algorithm [13, 28] for its

Algorithm 1 FirstApproach and check functions.

```
function FirstApproach(G, $\mathbb{S}_n$, subgraphs, $\Gamma$)
    $\hat{\mathbb{S}} \leftarrow$ subgraphs(G)
    $results \leftarrow$ List()
    for all $\hat{\mathcal{S}} \in \hat{\mathbb{S}}$ do
        $\hat{\mathcal{F}} \leftarrow$features($\hat{\mathcal{S}}$)
        if check($\hat{\mathcal{F}}, \mathbb{S}_n, \Gamma$) then
            $results \leftarrow results + \hat{\mathcal{S}}$
            break
        end if
    end for
    return $results$
end function
function check($\mathcal{F}_\kappa, \Psi, \Gamma$)
    for all $\psi \in \Psi$ do
        $\mathcal{F}_\psi \leftarrow$features($\psi$)
        if $d_c(\mathcal{F}_\kappa, \mathcal{F}_\psi) < \Gamma$ then
            return $true$
        end if
    end for
    return $false$
end function
```

inherent usage of the neighborhood. It creates communities by considering nodes interactions with their neighbors. The algorithm uses a threshold to determine a node's membership in a community. It also analyzes the frequency to set the dominant label to propagate.

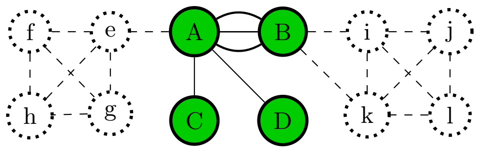
Nevertheless, we identified two limitations to community detection as a solution for identifying the social truth. The first limitation stems from the definition of a community itself. Often, a community is referred to as a subgraph that is highly connected among its nodes and that shares only a few edges with the rest of the graph [4].

The second limitation is the design of community detection algorithms. These algorithms create clusters of any type, covering the entire graph and providing subgraphs even if they are not relevant. In a sense, community detection algorithms provide more information than we require, as our problem is confined to a specific type of SGI.

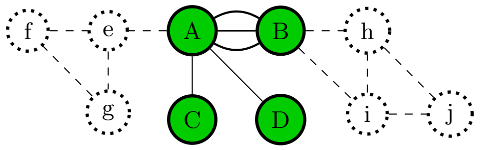
On the other hand, graph matching [26] consists in finding all occurrences of a query subgraph within a graph. An occurrence is a subgraph that has the same topology as the query. The problem that consists of ensuring that a subgraph has the same topology as a query is named the isomorphism problem [7]. This problem is NP-complete in the general case [3].

As well as for community detection based methods, considering the problem of social truth identification, one can point out three limitations to graph matching based methods. The first limitation is that a query cannot represent an SGI in $\mathbb{S}$. A query has a fixed number of nodes and edges. SGIs in $\mathbb{S}$ have high variability in the number of nodes and edges. To overcome this limitation, we can look for Approximate Graph Matching [5,26]. Approximate Graph Matching methods find subgraphs that are similar enough to a query. The similarity can be computed using the graph edit distance. The graph edit distance is the required distortion to obtain the subgraph by inserting, removing, or substituting some nodes or edges. However, this metric is generally NP-hard to compute [27].

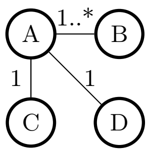
The second limitation of this approach in our use case is the lack of context around the query subgraph. In a social network, a group of people is defined by its structure, its personal pieces of information, and the way it connects to the rest of the graph. For example, the query in Figure 2c matches both types A and B in Figure 2. The only way to distinguish them is by considering their extended neighbors. For this reason, a query should be sampled from the graph to take the context around it into account. To the best of our knowledge, such an approach does not exist in the literature.



(a) SGI of type A



(b) SGI of type B



(c) Example of query that matches both types A and B.

Fig. 2: The first two subgraphs represent two different types of SGI. The difference is not in the topology of the subgraphs but in their context. A simple query like in the subgraph 2c is not enough to differentiate them.

The third limitation is that there are n samples in $\mathbb{S}_n$, whereas graph matching requires a unique query. We compensate for this limitation by analyzing subgraphs in $\mathbb{S}_n$ and computing their Maximum Common Subgraph (MCS) [17] to obtain a unique query. Then, we propose to do classic graph matching with this MCS as the query. This approach is the one we chose for the SUBGRAPHS function when considering graph matching.

Features Function A feature vector enables encoding information about a graph, a subgraph, a node, or an edge. This concept is related to element characterization in graphs. The objective of characterization is to generate a vector of fixed size that encodes information about an element [2, 8, 12, 15, 16]. In the FirstApproach function presented in Algorithm 1, the features function takes a subgraph as input. Thus, we are interested in subgraph characterization.

Subgraph characterization requires capturing the local structure, the subgraph's relative position, and nodes' specific information. We achieve this by defining a vector composed of some metrics, as mentioned in reference [12]. For example, we can use the following metrics:

– The number of nodes and edges in the subgraph. These metrics are straightforward and discriminate between subgraphs of different sizes.
– The minimum, maximum, and average of degrees, or clustering coefficients encode the internal structure of the graph.
– The minimum, maximum, and average of the shortest path length within the subgraph encode the diameter of the subgraph and can detect central nodes.
– Transitivity: This is the fraction of all possible triangles present in the subgraph. It encodes how tightly knitted the subgraph is and gives information about the subgraph's internal structure.
– Assortativity is the similarity of connections in the graph with respect to the node degree. This metric informs us how the connected nodes are similar to each other.

This list is non-exhaustive. The selection of one metric over another is related to the dataset and the SGI type we are searching for.

In the state of the art, we find deep learning-based methods to create feature vectors. We find methods such as SubGNN [2], SUGAR [22], or Sub2Vec [1].

Selection Step The subgraphs function returns subgraphs that might be irrelevant. For this reason, we have to select only those that could match a subgraph present in $\mathbb{S}$. We propose to resort to the cosine distance $d_c(A, B) = \frac{A \cdot B}{\|A\| \|B\|}$ and define an upper bound Γ for the distance to make sure that our subgraphs are relevant. This guarantees that the property $\hat{\mathbb{S}} \simeq \mathbb{S}$ in framework (1) is fulfilled. This distance is applied to the feature vectors previously defined. Other distances, such as the Euclidean distance or the Manhattan distance, could also be considered.

3.2 Second Approach

In this section, we propose a *method* with a completely different strategy. Instead of predicting where the subgraphs are and whether elements such as nodes or edges belong to an SGI or not, we approach the problem in reverse. We are interested in predicting elements that are not in an SGI. We then prune these

Algorithm 2 SECONDAPPROACH function.

```
1:  function SECONDAPPROACH(G, $\mathbb{S}_n$, $\Gamma_{node}, \Gamma_{edge}$)
2:      $V_{bad} \leftarrow V_G$
3:      $E_{bad} \leftarrow E_G$
4:      for all $u \in V_G$ do
5:          $\mathcal{F}_u \leftarrow$FEATURES$(u)$
6:          for all $\mathcal{S} \in \mathbb{S}_n$ do
7:              if CHECK$(\mathcal{F}_u, V_{\mathcal{S}}, \Gamma_{node})$ then
8:                  $V_{bad} \leftarrow V_{bad} - u$
9:                  break
10:             end if
11:         end for
12:     end for
13:     for all $e \in E_G$ do
14:         $\mathcal{F}_e \leftarrow$FEATURES$(e)$
15:         for all $\mathcal{S} \in \mathbb{S}_n$ do
16:             if CHECK$(\mathcal{F}_e, E_{\mathcal{S}}, \Gamma_{edge})$ then
17:                 $E_{bad} \leftarrow E_{bad} - e$
18:                 break
19:             end if
20:         end for
21:     end for
22:     $G_{pruned} \leftarrow \nabla(G, V_{bad}, E_{bad})$
23:     return CONNECTED$(G)$
24: end function
```

elements from the graph. We suppose that the resulting connected components are all SGIs in $\mathbb{S}$. This approach is presented in Algorithm 2.

We can relate this approach to the edge betweenness community detection algorithm [6]. The difference is that our approach is not limited to edge pruning. To prune elements, we first need to know which elements do not belong to an SGI in $\mathbb{S}$. This is achieved by comparing the features of elements of the multigraph with elements from samples in $\mathbb{S}_n$. Again, we need feature vectors for nodes and edges in the graph. These vectors can be obtained using two methods.

The first method involves using a simple hand-engineered feature vector, as presented in Section 3.1, applied at both the node and edge levels. We design feature vectors using users' and transactions' properties. The second method uses graph embedding techniques such as the Graph Isomorphism Network (GIN) [25] for node features and the Edge2Vec method [23] for edge features.

GIN is based on a Message Passing Neural Network [11]. It creates a feature vector based on the neighborhood of nodes. GIN is as powerful as the Weisfeiler-Lehman graph isomorphism test [24], which ensures that the topology around nodes is correctly encoded.

Edge2vec combines a deep autoencoder [18] to reduce the size of the feature vector and a Skipgram model [14] to ensure the vector correctly encodes the context around edges.

We compare the features of nodes and edges of the multigraph with the training samples in $\mathbb{S}_n$. We propose using again the cosine distance d_c as defined in Section 3.1. If a node or an edge differs too much from any of the training samples, we add it to the *bad* set. Let $\mathcal{F}$ represent the features of an element, such as a node or an edge. We consider thresholds $\varGamma_{node}$ and $\varGamma_{edge}$ as the maximum distance a node or an edge can be from a node or an edge of the training samples. The sets V_{bad} and E_{bad}, referred to as the *bad* sets, contain the elements to be removed from the multigraph G. We consider all the connected components obtained by the function CONNECTED to be the results $\hat{\mathbb{S}}$.

We define a pruning function ∇ that takes a multigraph G, the bad sets V_{bad} and E_{bad}, and returns a graph from which the nodes and edges from V_{bad} and E_{bad} have been removed. We express the ∇ function as follows:

$$\nabla(G, V_{bad}, E_{bad}) = G_{pruned} \tag{2}$$

We define four specific pruning functions: ∇_{simple}, ∇_{node}, ∇_{edge}, and $\nabla_{majority}$.

The ∇_{simple} pruning function removes all the nodes and edges present in the *bad* sets. We express the ∇_{simple} pruning function as follows:

$$\nabla_{simple}(G, V_{bad}, E_{bad}) = (V_G \setminus V_{bad}, E_G \setminus E_{bad}) \tag{3}$$

When we remove a node, we also remove all edges that were connected to that node.

The ∇_{node} is expressed as follows:

$$\nabla_{node}(G, V_{bad}, E_{bad}) = (V_G \setminus V_{bad}, E_G) \tag{4}$$

The weakness of the ∇_{node} pruning function is the case of two connected subgraphs. For example, in Figure 3, we cannot distinguish the subgraph $ABCD$ from the subgraph $EFGHI$ by removing nodes only. We need to remove the edge (C, E) to reveal the two connected components.

We formalize the ∇_{edge} pruning function as follows:

$$\nabla_{edge}(G, V_{bad}, E_{bad}) = (V_G, E_G \setminus E_{bad}) \tag{5}$$

The weakness of the ∇_{edge} pruning function is the multigraph context. We need to remove all edges between two nodes to create two disconnected components. For example, in Figure 3, in the case of multiple edges between nodes C and E represented by a dashed line, all edges between C and E must be removed.

The $\nabla_{majority}$ pruning function removes nodes and edges, but spares nodes if the majority of their edges are not in E_{bad}. We define the function as follows:

$$edgeMajority(v) = \frac{|E(v) \setminus E_{bad}|}{|E(v)|} \tag{6}$$

Here, $E(v)$ represents the set of all edges connected to node v, and $E(v) \setminus E_{bad}$ represents the set of edges connected to v that are not in E_{bad}. The ratio thus

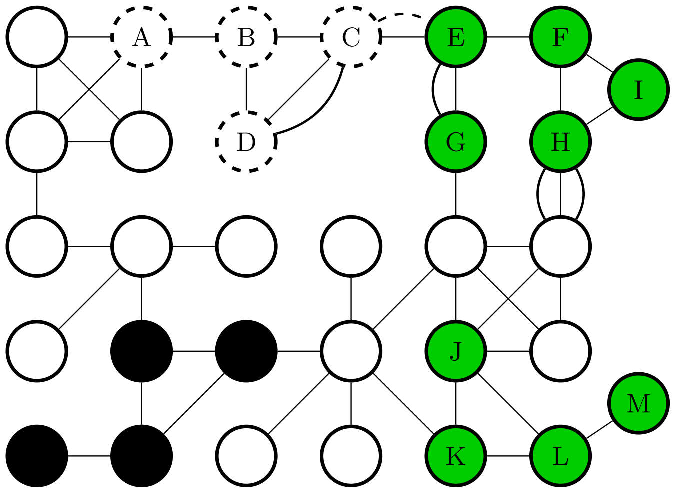


Fig. 3: An example of a multigraph. Dashed nodes are from a subgraph in $\mathbb{S}_n$. Dashed and green nodes compose three subgraphs from $\mathbb{S}$. Black nodes are from a subgraph that shares a similar topology but not the same context.

gives the proportion of good edges to total edges connected to v. Next, we define the set of all spared nodes as follows:

$$V_{spared} = \{v \in V_G \mid edgeMajority(v) \geq 0.5\} \tag{7}$$

In this expression, V_G represents the set of all nodes in the multigraph. The condition $edgeMajority(v) \geq 0.5$ ensures that only those nodes for which the majority of their edges are not in E_{bad} are included in V_{spared}. Finally, we can formalize the pruning function as follows:

$$\nabla_{majority}(G, V_{bad}, E_{bad}) = (V_G \setminus (V_{bad} \setminus V_{spared}), E_G \setminus E_{bad}) \tag{8}$$

The expression $(V_{bad} \setminus V_{spared})$ identifies the set of nodes that are bad and not spared. Essentially, it removes the spared nodes from the set of bad nodes, ensuring that nodes with a majority of good connections are not pruned.

This *method* of pruning nodes and edges from the multigraph is a new perspective on the problem of social truth identification. We reversed the problem of the SGIs position's prediction by focusing on the removal of elements from the multigraph to create connected components.

4 How to Evaluate the Framework

In this section, we propose an adaptation of the *precision* and *recall* metrics to evaluate the performance of the *method* in the framework (1). We need to compare the output $\hat{\mathbb{S}}$ of the framework with the social truth $\mathbb{S}$. We describe a *match* function that is used in the adaptation of the *precision* and the *recall*. The *match* function is a boolean function. It guarantees that two subgraphs are similar enough to be matched. It takes $\hat{\mathcal{S}} \in \hat{\mathbb{S}}$ and $\mathcal{S} \in \mathbb{S}$ as inputs and returns *true* if they match, *false* otherwise. The *match* function uses the three boolean functions φ_{extra}, $\varphi_{missing}$, and φ_{size} that we describe now.

1. The φ_{extra} boolean function checks that the ratio of the nodes found in $\hat{\mathcal{S}}$ that are not in $\mathcal{S}$ over the size of $V_{\mathcal{S}}$ is under the threshold $\varGamma_{extra}$. We express it as follows:
$$\varphi_{extra}(\hat{\mathcal{S}}, \mathcal{S}) = \frac{|V_{\hat{\mathcal{S}}} \setminus V_{\mathcal{S}}|}{|V_{\mathcal{S}}|} < \varGamma_{extra} \tag{9}$$
2. The $\varphi_{missing}$ boolean function checks that the ratio of nodes in $\mathcal{S}$ that are not present in $\hat{\mathcal{S}}$ over the number of nodes in $V_{\mathcal{S}}$ is under the threshold $\varGamma_{missing}$. We express it as follows:
$$\varphi_{missing}(\hat{\mathcal{S}}, \mathcal{S}) = \frac{|V_{\mathcal{S}} \setminus V_{\hat{\mathcal{S}}}|}{|V_{\mathcal{S}}|} < \varGamma_{missing} \tag{10}$$
3. The φ_{size} boolean function checks that the size difference between $\mathcal{S}$ and $\hat{\mathcal{S}}$ over the size of $V_{\mathcal{S}}$ is under a threshold $\varGamma_{size}$. We express it as follows:
$$\varphi_{size}(\hat{\mathcal{S}}, \mathcal{S}) = \frac{abs(|V_{\hat{\mathcal{S}}}| - |V_{\mathcal{S}}|)}{|V_{\mathcal{S}}|} < \varGamma_{size} \tag{11}$$

Every threshold is set according to the context of the problem.

We need to consider the three boolean functions together to declare two subgraphs as matching. We express the *match* function as follows:

$$match(\mathcal{S}, \hat{\mathcal{S}}) = \begin{cases} \varphi_{extra}(\hat{\mathcal{S}}, \mathcal{S}) & \& \\ \varphi_{missing}(\hat{\mathcal{S}}, \mathcal{S}) & \& \\ \varphi_{size}(\hat{\mathcal{S}}, \mathcal{S}) \end{cases} \tag{12}$$

Iverson bracket [10] is a notation that generalizes the Kronecker delta. This function is defined to take the value 1 if the statement is true and the value 0 otherwise. It is generally denoted as follows, where P is the statement:

$$[P] = \begin{cases} 1 \text{ if } P \text{ is true;} \\ 0 \text{ otherwise.} \end{cases} \tag{13}$$

We define a subgraph $\mathcal{S}_a$ as relevant if it matches at least one subgraph in a set of subgraphs $\varPsi$. The $\varPsi$ variable can take the value $\mathbb{S}_n$, $\mathbb{S}$, or $\hat{\mathbb{S}}$. We use the Iverson bracket to formalize the *relevant* boolean function as follows:

$$relevant(\mathcal{S}_a, \varPsi) = \Big(\sum_{\mathcal{S}_b \in \varPsi} [match(\mathcal{S}_a, \mathcal{S}_b)] \Big) \geq 1 \tag{14}$$

We use *precision* and *recall* metrics to evaluate the framework. The *precision* metric counts the proportion of relevant SGIs among selected SGIs. The *recall* metric counts the proportion of relevant SGIs selected among all selectable relevant SGIs. We define it as the ratio of $\hat{\mathcal{S}}$ that matches an element in $\mathbb{S}$ out of all the predicted elements in $\hat{\mathbb{S}}$. We formalize *precision* as follows:

$$precision = \frac{\sum_{\hat{\mathcal{S}} \in \hat{\mathbb{S}}} [relevant(\hat{\mathcal{S}}, \mathbb{S})]}{|\hat{\mathbb{S}}|} \tag{15}$$

A higher precision means that the function finds more subgraphs that are close enough to the social truth.

We formalize *recall* as follows:

$$recall = \frac{\sum_{\mathcal{S} \in \mathbb{S}} [relevant(\mathcal{S}, \hat{\mathbb{S}})]}{|\mathbb{S}|} \tag{16}$$

A higher recall means that a greater number of the social truth subgraphs are matched.

In conclusion, the adaptation of precision and recall metrics, integrated with the match function, provides a robust framework for evaluating the performance of the method in the framework (1). We can also calculate an F-measure score [9] such as an F1-score [21].

5 Conclusions and Future Work

In this paper, we introduce a novel framework designed to address the challenge of social truth identification within transactional graphs. This is a critical task for deciphering complex social interactions in large datasets. We present three distinct methods that combine traditional graph analysis techniques with innovative approaches to recover the hidden social truth. By adapting precision and recall metrics, we have established a method to evaluate the performance of the proposed framework. This ensures that the identified SubGraphs of Interest are relevant and representative of the underlying social structures.

For the future, we propose two interesting directions that involve integrating more advanced deep learning techniques into our framework:

1. Enhancement of Community Detection: Incorporating deep learning techniques into community detection algorithms could significantly improve their performance across diverse datasets. Exploring methods such as the Neural Overlapping Community Detection model [19] and other deep learning-based approaches could offer substantial improvements in detecting structures within multigraphs.
2. Deep Learning as a distance function: Applying deep learning to enhance the distance function d used with hand-crafted feature vectors and a pruning approach could increase the framework's robustness and performance. This integration would potentially enable more accurate identification of various types of SGIs, making the framework more effective across different scenarios.

Experiments using the framework are in progress and will be presented in a separate article. We will present the performance and efficiency in terms of time and memory of the framework with large datasets.

Acknowledgments. The authors would like to thank Orange and the ANRT for funding the thesis.

References

1. Adhikari, B., Zhang, Y., Ramakrishnan, N., Prakash, B.A.: Sub2Vec: Feature Learning for Subgraphs. In: Phung, D., Tseng, V.S., Webb, G.I., Ho, B., Ganji, M., Rashidi, L. (eds.) Advances in Knowledge Discovery and Data Mining. pp. 170–182. Lecture Notes in Computer Science, Springer International Publishing (2018). https://doi.org/10.1007/978-3-319-93037-4_14
2. Alsentzer, E., Finlayson, S., Li, M., Zitnik, M.: Subgraph Neural Networks. In: Advances in Neural Information Processing Systems. vol. 33, pp. 8017–8029 (2020). https://doi.org/10.48550/ARXIV.2006.10538
3. Babai, L.: Graph isomorphism in quasipolynomial time [extended abstract]. In: Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing. p. 684–697. STOC '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2897518.2897542
4. Fortunato, S.: Community detection in graphs. Physics Reports **486**(3–5), 75–174 (Feb 2010). https://doi.org/10.1016/j.physrep.2009.11.002
5. Gallagher, B.: Matching structure and semantics: A survey on graph-based pattern matching. In: AAAI Fall Symposium: Capturing and Using Patterns for Evidence Detection (2006), https://api.semanticscholar.org/CorpusID:12917214
6. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences **99**(12), 7821–7826 (Jun 2002). https://doi.org/10.1073/pnas.122653799
7. Grohe, M., Neuen, D.: Recent advances on the graph isomorphism problem. In: BCC (2020), https://api.semanticscholar.org/CorpusID:226237505
8. Grover, A., Leskovec, J.: Node2vec: Scalable Feature Learning for Networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (Aug 2016). https://doi.org/10.1145/2939672.2939754
9. Hand, D.J., Christen, P., Kirielle, N.: F*: an interpretable transformation of the f-measure. Machine Learning **110**(3), 451–456 (Mar 2021). https://doi.org/10.1007/s10994-021-05964-1
10. Iverson, K.E.: A programming language. John Wiley & Sons, Inc., USA (1962)
11. Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. In: International Conference on Learning Representations. Poster (2017). https://doi.org/10.48550/ARXIV.1609.02907, https://openreview.net/forum?id=SJU4ayYgl
12. Li, G., Semerci, M., Yener, B., Zaki, M.J.: Effective graph classification based on topological and label attributes. Statistical Analysis and Data Mining **5**(4), 265–283 (Jun 2012). https://doi.org/10.1002/sam.11153
13. Liu, Q., Su, Y., Peng, Q., Chen, K., Lu, Y.: An overlapping community detection algorithm for label propagation based on node influence. In: 2021 3rd International Conference on Advances in Computer Technology, Information Science and Communication (CTISC). pp. 183–187 (2021). https://doi.org/10.1109/CTISC52352.2021.00041
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. In: International Conference on Learning Representations. Poster (2013-01). https://doi.org/10.48550/ARXIV.1301.3781

15. Narayanan, A., Mahinthan, C., Venkatesan, R., Chen, L., Liu, Y., Jaiswal, S.: graph2vec: Learning Distributed Representations of Graphs. In: Proceedings of the 13th International Workshop on Mining and Learning with Graphs (MLG). Workshop on Mining and Learning with Graphs, Association for Computing Machinery (2017). https://doi.org/10.48550/ARXIV.1707.05005
16. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: Online Learning of Social Representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 701–710. Association for Computing Machinery (2014). https://doi.org/10.1145/2623330.2623732
17. Roy, I., Chakrabarti, S., De, A.: Maximum common subgraph guided graph retrieval: late and early interaction networks. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22, Curran Associates Inc., Red Hook, NY, USA (2024)
18. Salakhutdinov, R., Hinton, G.: Semantic hashing. International Journal of Approximate Reasoning **50**(7), 969–978 (2009). https://doi.org/10.1016/j.ijar.2008.11.006, https://www.sciencedirect.com/science/article/pii/S0888613X08001813, special Section on Graphical Models and Information Retrieval
19. Shchur, O., Günnemann, S.: Overlapping Community Detection with Graph Neural Networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Deep Learning on Graphs Workshop (DLG), Association for Computing Machinery (2019). https://doi.org/10.48550/ARXIV.1909.12201
20. Singh, D., Garg, R.: Ni-louvain: A novel algorithm to detect overlapping communities with influence analysis. Journal of King Saud University - Computer and Information Sciences **34**(9), 7765–7774 (2022). https://doi.org/10.1016/j.jksuci.2021.07.006, https://www.sciencedirect.com/science/article/pii/S1319157821001737
21. Sokolova, M., Japkowicz, N., Szpakowicz, S.: Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation. In: Sattar, A., Kang, B.h. (eds.) AI 2006: Advances in Artificial Intelligence. pp. 1015–1021. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
22. Sun, Q., Li, J., Peng, H., Wu, J., Ning, Y., Yu, P.S., He, L.: SUGAR: Subgraph Neural Network with Reinforcement Pooling and Self-Supervised Mutual Information Mechanism. In: Proceedings of the Web Conference 2021. pp. 2081–2091. WWW '21, Association for Computing Machinery (Apr 2021). https://doi.org/10.1145/3442381.3449822
23. Wang, C., Wang, C., Wang, Z., Ye, X., Yu, P.S.: Edge2vec: Edge-based social network embedding. ACM Trans. Knowl. Discov. Data **14**(4) (may 2020). https://doi.org/10.1145/3391298
24. Weisfeiler, B.Y., Leman, A.A.: The reduction of a graph to canonical form and the algebra which appears therein. Nauchno-Technicheskaya Informatsia, Series **2**(9), 12–16 (1968), https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf
25. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? (2019). https://doi.org/10.48550/arxiv.1810.00826
26. Yan, J., Yin, X.C., Lin, W., Deng, C., Zha, H., Yang, X.: A short survey of recent advances in graph matching. In: Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval. pp. 167–174. ICMR '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2911996.2912035
27. Zeng, Z., Tung, A.K.H., Wang, J., Feng, J., Zhou, L.: Comparing stars: on approximating graph edit distance. Proc. VLDB Endow. **2**(1), 25–36 (aug 2009). https://doi.org/10.14778/1687627.1687631

28. Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. Tech. rep., Carnegie Mellon University (2002), https://pages.cs.wisc.edu/~jerryzhu/pub/CMU-CALD-02-107.pdf