FINDING ANTAGONISTIC COMMUNITIES IN SIGNED UNCERTAIN GRAPHS

FINDING ANTAGONISTIC COMMUNITIES IN SIGNED UNCERTAIN GRAPHS

By QIQI ZHANG, BS

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the Requirements for the Degree Master of Science

McMaster University

MASTER OF SCIENCE (2023)

Hamilton, Ontario, Canada (Computing and Software)

TITLE:	Finding Antagonistic Communities in Signed Uncertain Graphs
AUTHOR:	Qiqi Zhang BS (Mathematics & Statistics), McMaster University, Hamilton, Canada
SUPERVISOR:	Dr. Lingyang Chu
NUMBER OF PAGES:	xiii, 56

Lay Abstract

Uncertain graph analysis is vital for various real-world applications, but traditional graph mining algorithms face challenges due to uncertain information. In this paper, we introduce a novel method to identify antagonistic communities in signed uncertain graphs. These communities consist of edges that are highly likely to be positive within the same community, and highly likely to be negative between different communities. By focusing on local parts of the graph, our method efficiently discovers significant groups of antagonistic communities. The proposed method is supported by solid theoretical foundations and is evaluated on real-world and synthetic datasets. Results demonstrate its exceptional effectiveness and efficiency in uncovering and analyzing antagonistic communities. This research paves the way for improved understanding of uncertain graphs and their impact on diverse applications.

Abstract

Uncertain graph analysis plays a crucial role in many real-world applications, where the presence of uncertain information poses challenges for traditional graph mining algorithms. In this paper, we propose a novel method to find antagonistic communities in signed uncertain graphs, where vertices in the same community have a large expectation of positive edge weights and the vertices in different communities have a large expectation of negative edge weights. By restricting all the computations on small local parts of the signed uncertain graph, our method can efficiently find significant groups of antagonistic communities. We also provide theoretical foundations for the method. Extensive experiments on five real-world datasets and a synthetic dataset demonstrate the outstanding effectiveness and efficiency of the proposed method.

Acknowledgements

I would like to express my heartfelt appreciation to my supervisor, Dr. Chu, for his unwavering guidance and support during the course of this research project. His expertise, patience, and encouragement have been instrumental in shaping the direction of my work and helping me overcome challenges. I am truly grateful for his valuable insights and constructive feedback at every stage of the project.

I would also like to extend my gratitude to my senior colleagues, Xin and Shaoxin, for their assistance and valuable insights during the course of this project.

Furthermore, I would like to thank my beloved alma mater, McMaster University, for providing an exceptional academic environment that fosters learning, growth, and research excellence. The resources, facilities, and opportunities provided by the university have been instrumental in the success of this project.

Finally, I would like to express my deep appreciation to my family for their unwavering support and encouragement throughout this journey. Their love, understanding, and belief in my abilities have been a constant source of motivation.

Lastly, I would like to thank my adorable cat, Jan. Despite being mischievous and often causing a ruckus, I am grateful for his companionship over the countless days and nights. Thank you for the comforting purrs after late-night work sessions and your soft furry belly and plush paws.

Table of Contents

List of Figures

List of Tables

Notation, Definitions, and Abbreviations

Notation

S	The set of vertex indices
$\mathcal{G}$	The signed uncertain graph
$\mathcal{G}_S$	The signed uncertain subgraph induced by S
G	The possible world / the signed static graph
G_S	The signed static subgraph induced by S
$\mathbb{G}$	The set of signed uncertain subgraphs
$\mathbb{P}(G)$	The probability of sampling G from $\mathcal{G}$
V	The set of vertices
V_S	The subset of V
v_i	The i-th vertex

(v_i, v_j)	The edge connecting the vertices v_i and v_j
A^+	The real-valued adjacency matrices to store the positive edge weights
$A^+_{i,j}$	The entry in the i-th row and j-th column of A^+
A^+_G	The adjacency matrices to store the sampled positive edge weights
$(A^+_G)_{i,j}$	The entry in the i-th row and j-th column of A^+_G
A^-	The real-valued adjacency matrices to store the negative edge weights
$A^-_{i,j}$	The entry in the i-th row and j-th column of A^-
A^-_G	The adjacency matrices to store the sampled neagtive edge weights
$(A^-_G)_{i,j}$	The entry in the i-th row and j-th column of A^-_G
P^+	The probability matrices to store the probability of being positive for each edge
$P^+_{i,j}$	The entry in the i-th row and j-th column of P^+
P^-	The probability matrices to store the probability of being negative for each edge
$P^-_{i,j}$	The entry in the i-th row and j-th column of P^-
$P^0_{i,j}$	The probability of not having an edge between v_i and v_j
$\mathbf{x}$	The embedding vector
X	The embedding matrix

$X_{:,j}$	The j-th column of X
$X_{i,j}$	The entry in the i-th row and j-th column of X
$\texttt{OF}(X, G_h)$	The overall friendship of X on a possible world G_h
$\texttt{OA}(X, G_h)$	The overall antagonism of X on a possible world G_h
$d_h^+(X_{:,j})$	The weighted average friendship of all the edges in the static subgraph induced by $X_{:,j}$ from G_h
$d_h^-(X_{:,j}, X_{:,l})$	The weighted average antagonism of all the edges between the two static subgraphs induced by $X_{:,j}$ and $X_{:,l}$ from G_h
$\texttt{SUM}(X, G_h)$	The sum of the overall friendship and overall antagonism of X on G_h
$\mathbb{E}\big(\texttt{OF}(X, G_h)\big)$	The expectation of $\texttt{OF}(X, G_h)$
$\mathbb{E}\big(\texttt{OA}(X, G_h)\big)$	The expectation of $\texttt{OA}(X, G_h)$
$\mathbb{E}\big(d_h^+(X_{:,j})\big)$	The expectation of $d_h^+(X_{:,j})$
$\mathbb{E}\big(d_h^-(X_{:,j}, X_{:,l})\big)$	The expectation of $d_h^-(X_{:,j}, X_{:,l})$
$\mathbb{E}\big(\texttt{SUM}(X, G_h)\big)$	The expectation of $\texttt{SUM}(X, G_h)$
$\texttt{SIG}(X)$	The significance of X on $\mathcal{G}$

Definitions

Signed uncertain graph

A signed uncertain graph consists of a set of vertices and a set of signed uncertain edges. The weight of a signed uncertain edge can be positive, negative or zero, where a positive weight shows a friendly relationship between the two vertices, a negative weight shows an antagonistic relationship, and a weight equal to zero means no edge exists between the two vertices. Additionally, each edge is also associated with a discrete probability distribution, showing the probabilities for the edge weight to be positive, negative, and zero.

Possible world

A signed uncertain graph $\mathcal{G}$ describes a distribution over a population of signed static graphs, each of which is called a possible world of $\mathcal{G}$.

Abbreviations

SUG	Signed uncertain graph
SSG	Signed static graph
FACSUG	Finding antagonistic communities in signed uncertain graph
PGD	Proximal gradient descend
LUA	Locate and update algorithm
KKT	Karush-Kuhn-Tucker

RD	Replicator dynamics
FPS	Fitness proportionate selection
SGA	Synthetic genetic array
SNAP	Stanford network analysis project
OF	Overall friendship
OA	Overall antagonism
HAM	Harmonic mean

Declaration of Academic Achievement

I, Qiqi Zhang, hereby declare that the academic achievement presented in this document is the result of my own original work and has been conducted under the guidance and supervision of Dr. Lingyang Chu. All sources used have been duly acknowledged and referenced.

I affirm that this academic achievement has not been submitted for any other degree or qualification at any other university or educational institution. The research conducted adheres to the ethical guidelines and standards set forth by McMaster University and relevant regulatory bodies.

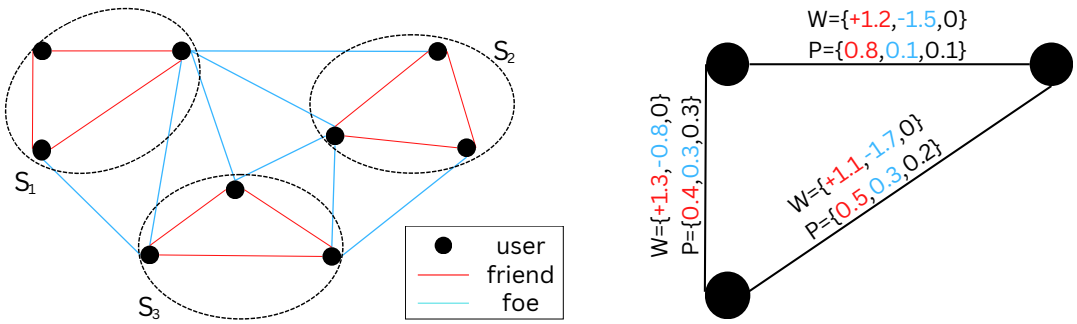
I take full responsibility for the accuracy and integrity of the information presented in this document. Any contributions or assistance received from others during the course of this academic achievement have been duly acknowledged.

I understand that any falsification or misrepresentation of the information contained herein may result in disciplinary action, including revocation of any degrees or qualifications awarded based on this work.

Chapter 1

Introduction

Finding antagonistic communities in signed graphs is a classic data mining task that enjoys many real-world applications [22, 35, 18]. For example, in the signed graph of a social network, where each vertex represents a user and the positive/negative edge weight represents the friend/foe relationship between users, finding groups of antagonistic communities can identify polarized social communities where most of the users in the same community are friends and the users in different communities are mostly foes [42, 3]. Figure 1.1(a) is an example of antagonistic communities in a signed static graph of a social network. $S1, S2, S3$ are three different antagonistic communities. In the signed graph of a gene interaction network, where each vertex represents a gene and the positive/negative edge weight represents the stimulatory and inhibitory interactions between genes, finding groups of antagonistic communities can identify functionally inhibitory gene groups [20, 32], where most of the genes in the same community stimulate each other and the genes in different communities inhibit each other. Finding such groups of gene communities can help in understanding the functional organization of the gene interaction network [31].



(a) (b)

Figure 1.1: Two illustrative examples. (a) shows an example of a group of three antagonistic communities in a signed static graph. (b) shows an example of signed uncertain graph.

Most existing methods [14, 6, 38, 15] find antagonistic communities by assuming the signed graph to be a static graph with constant edge weights. However, in many real-world networks, the signed relationships between vertices are often uncertain. For example, the friend/foe relationships between users in a social network can be uncertain due to many factors such as the dynamics [4], the timeliness [12], and the privacy [36] of social relationships. In the signed graph of a gene interaction network, the uncertainty of the stimulatory/inhibitory interactions between genes is directly caused by the uncertainty of biological experiments [17, 33]. As a result, it is more realistic to capture the uncertainty of relationships in a real-world signed network by a **signed uncertain graph (SUG)**, where each edge is independently associated with a discrete probability distribution of signed edge weights. Figure 1.1(b) shows an example of signed uncertain graph.

We are interested in tackling the novel problem of **finding antagonistic communities in signed uncertain graphs (FACSUG)**. The goal is to find groups of antagonistic communities, where the vertices in the same community have a large

expectation of positive edge weights and the vertices in different communities have a large expectation of negative edge weights.

As discussed later in Section 2, FACSUG is a challenging problem that cannot be straightforwardly solved by existing methods [14, 6, 38, 28, 10, 40, 9, 8, 19, 24, 43]. First, most of the signed graph clustering methods [14, 6, 38, 28, 10, 40, 9] are developed to find one or multiple groups of antagonistic communities in a signed static graph. These methods cannot be directly applied to find antagonistic communities in a signed uncertain graph due to the uncertainty of edge weights. Second, many uncertain graph clustering methods [8, 19, 24, 43] are designed to find cohesive subgraphs in an unsigned uncertain graph without negative edge weights, where each cohesive subgraph is expected to contain a large number of positive edge weights. These methods handle the uncertainty of edge weights well, however, they cannot incorporate the negative edge weights to model the antagonism between communities, which prohibits them from successfully tackling the FACSUG problem.

In this paper, we tackle the novel FACSUG problem and make the following contributions.

First, we model a group of antagonistic communities as a set of k signed uncertain subgraphs, where each subgraph represents a community, the vertices in each community have a large expectation of positive edge weights between each other, and the vertices in different communities are connected by the edges with a large expected negative weight. In this way, we model the FACSUG problem as a constrained quadratic programming problem.

Second, we show that the classic proximal gradient descent (PGD) method [29] is very inefficient in solving the constrained quadratic programming problem. Thus, we

solve the problem by developing a novel method named LUA, which is experimentally three orders of magnitudes faster than PGD. The key idea of LUA is to confine all the computational operations to small subgraphs instead of the whole graph, such that computations can be efficiently conducted on small sub-matrices of the adjacency matrix without sacrificing any computational accuracy.

Last, we conduct extensive experiments on five real-world datasets and a synthetic dataset. The results clearly show that our method is accurate, efficient and scalable in finding group(s) of antagonistic communities. We also conduct two case studies to show the performance of the proposed method in finding meaningful groups of antagonistic communities. One case study is conducted on the gene-interaction network extracted from BioGRID [33] to find groups of stimulatory genes that are inhibitory with each other. Another case study is conducted on the adjective network extracted from WordNet [27] to find groups of synonyms that are antonymous with each other.

Chapter 2

Related Works

Our work is generally related to the signed graph clustering methods [14, 6, 38, 28, 10, 40, 9] and the uncertain graph clustering methods in the literature.

The **signed graph clustering methods** [14, 6, 38, 28, 10, 40, 9] aim to find a group of conflicting cohesive subgraphs in a signed static graph, where "conflicting" means the vertices belonging to different subgraphs have large negative edge weights between each other, and "cohesive" means the vertices within the same subgraph have large positive edge weights between each other. Given a large signed static graph, EIGENSIGN [6] finds a single group of two conflicting cohesive subgraphs, each of which is a small subgraph in a local area of the signed graph. ATOM [28] finds multiple groups of subgraphs, where each group contains two conflicting cohesive subgraphs. KOCG [14] and SCG [38] attempt to find multiple groups of conflicting cohesive subgraphs, where each group may contain more than two subgraphs. SPONGE [15] partitions the signed graph into a group of conflicting cohesive subgraphs. PolarSeeds [40] uses two disjoint sets of seed vertices as a query to find a group of two conflicting cohesive subgraphs, where each subgraph contains one set of

seed vertices. MBCEnum [10] finds a maximum clique that can be partitioned into two conflicting cohesive subgraphs. MSCD [9] finds a maximum θ-clique, which is also possible to be partitioned into two conflicting cohesive subgraphs.

The above methods are related to our work because the group of conflicting cohesive subgraphs can be viewed as a group of antagonistic communities by regarding each cohesive subgraph as a community. However, these methods are customized for signed static graphs, and their definitions of conflicting cohesive subgraphs do not consider the uncertainty of edges in signed uncertain graphs. Therefore, they cannot be straightforwardly applied to find antagonistic communities in signed uncertain graphs.

Some **uncertain graph clustering methods** [8, 19, 24, 43] aim to find multiple cohesive subgraphs in an unsigned uncertain graph, where each edge has a probability of existing and the edge weights are non-negative. Ceccarello et al. [8] partition an unsigned uncertain graph into k subgraphs, where each subgraph has a central vertex that maximizes the minimum or the average connection probability to every vertex in the subgraph. Han et al. [19] solve the same problem as [8] by a more efficient approximation approach. AVPR [24] incorporates a reliability criterion into the classic k-means algorithm [25] to partition an unsigned uncertain graph into k reliable subgraphs that are not likely to be disconnected. SSCAN [43] proposes a stable structural clustering model to find stable subgraphs on an unsigned uncertain graph. These methods work well in finding stable or reliable cohesive subgraphs in an unsigned uncertain graph. However, since they are not designed to incorporate negative edge weights, they do not model the conflict between subgraphs. Therefore, these methods cannot be straightforwardly applied to find antagonistic communities

in signed uncertain graphs.

To the best of our knowledge, MIL [26] is the closest related work that aims to find a set of subgraphs in a signed uncertain graph by comprehensively considering the signs of edge weights and the uncertainty of edges. However, the problem tackled by MIL [26] is substantially different from the proposed FACSUG problem. Because MIL [26] aims to partition the entire signed uncertain graph into a single group of conflicting cohesive subgraphs, where every vertex in the signed uncertain graph must be contained in a cohesive subgraph. However, FACSUG aims to find multiple different groups of conflicting cohesive subgraphs, where a vertex is allowed not to be contained in any cohesive subgraph.

Chapter 3

Problem Definition

In this section, we first introduce how to model a signed uncertain graph and its subgraphs. Then, we introduce how to model a group of antagonistic communities in a signed uncertain graph. At last, we present the problem of finding antagonistic communities.

3.0.1 Modelling a Signed Uncertain Graph and Its Subgraphs

A signed uncertain graph consists of a set of vertices and a set of signed uncertain edges. The weight of a signed uncertain edge can be positive, negative or zero, where a positive weight shows a friendly relationship between the two vertices, a negative weight shows an antagonistic relationship, and a weight equal to zero means no edge exists between the two vertices. Additionally, each edge is also associated with a discrete probability distribution, showing the probabilities for the edge weight to be positive, negative, and zero.

We represent a **signed uncertain graph** by $\mathcal{G} = (V, A^+, A^-, P^+, P^-)$, where $V =$

$\{v_1, ..., v_n\}$ is the set of n vertices, A^+ and A^- are the n-by-n real-valued adjacency matrices to store the edge weights, and P^+ and P^- are the n-by-n probability matrices to store the probability distribution associated with each edge.

Denote by $A^+_{i,j}$, $A^-_{i,j}$, $P^+_{i,j}$, and $P^-_{i,j}$, respectively, the entries in the i-th row and j-th column of A^+, A^-, P^+, and P^-. Denote by (v_i, v_j) the edge connecting the vertices v_i and v_j. We have $A^+_{i,j} \geq 0$ and $A^-_{i,j} \geq 0$, where $A^+_{i,j}$ stores the positive edge weight of the edge (v_i, v_j) and $A^-_{i,j}$ stores the absolute value of the negative edge weight of the edge (v_i, v_j). We also have $0 \leq P^+_{i,j} \leq 1$, and $0 \leq P^-_{i,j} \leq 1$, where $P^+_{i,j}$ and $P^-_{i,j}$ are the probabilities for the edge (v_i, v_j) to have the weights $A^+_{i,j}$ and $-A^-_{i,j}$, respectively. We denote by $P^0_{i,j} = 1 - P^+_{i,j} - P^-_{i,j}$ to be the probability for the edge (v_i, v_j) to have a weight equal to zero, that is, the probability of not having an edge between v_i and v_j. Thus, if the edge (v_i, v_j) does not exist in $\mathcal{G}$, we have $P^0_{i,j} = 1$ and we set $A^+_{i,j}$, $A^-_{i,j}$, $P^+_{i,j}$, and $P^-_{i,j}$ to zero by default.

According to the classic model of possible worlds[34, 16], a signed uncertain graph $\mathcal{G}$ describes a distribution over a population of signed static graphs, each of which is called a **possible world** of $\mathcal{G}$. By independently sampling the weights of each edge in $\mathcal{G}$ from the associated probability distribution, we can generate a possible world, denoted by $G = (V, A^+_G, A^-_G)$, where V is the set of vertices in $\mathcal{G}$, and A^+_G and A^-_G are the n-by-n adjacency matrices to store the sampled edge weights.

Denote by $(A^+_G)_{i,j}$ and $(A^-_G)_{i,j}$ the entries at the i-th row and the j-th column of A^+_G and A^-_G, respectively. For each edge (v_i, v_j) in $\mathcal{G}$, the sampled edge weight has three possible outcomes, such as positive, negative, and zero. If the outcome is positive, then $(A^+_G)_{i,j} = A^+_{i,j}$ and $(A^-_G)_{i,j} = 0$. If the outcome is negative, then $(A^+_G)_{i,j} = 0$ and $(A^-_G)_{i,j} = A^-_{i,j}$. Otherwise, if the outcome is zero, then $(A^+_G)_{i,j} = 0$ and $(A^-_G)_{i,j} = 0$,

which means there is no edge between v_i and v_j. For a signed uncertain graph $\mathcal{G}$ containing m edges, there exists a population of 3^m unique possible worlds. The probability of sampling G from $\mathcal{G}$, denoted by $\mathbb{P}(G)$, can be computed by multiplying the probability of getting the sampled edge weight of each edge.

Next, we introduce how to represent a subgraph of $\mathcal{G}$.

Denote by $S \subseteq \{1, \ldots, n\}$ a set of vertex indices, a subgraph $\mathcal{G}_S$ of $\mathcal{G}$ is a **signed uncertain subgraph** that consists of the set of the vertices in $V_S = \{v_i \mid i \in S, v_i \in V\}$ and the subset of the signed uncertain edges in $\mathcal{G}$ between any pair of the vertices in V_S. Similarly, for any possible world G of $\mathcal{G}$, a subgraph G_S of G is a signed static subgraph that consists of the set of vertices in V_S and the subset of the edges in G between any pair of the vertices in V_S. Since $\mathcal{G}_S$ and G_S are determined by S, we say $\mathcal{G}_S$ and G_S are **induced** by S from $\mathcal{G}$ and G, respectively.

Following the routine of [30, 7, 13, 23], we map the set of vertex indices S into an n-dimensional **embedding vector** $\mathbf{x} = [x_1, x_2, ..., x_n]$, which is contained in the simplex $\Delta^n = \{\mathbf{x} \mid \sum_{i=1}^{n} x_i = 1, x_i \geq 0\}$. The i-th entry x_i represents the weight of v_i in V_S. If $x_i = 0$, then $i \notin S$, which further means the vertex v_i is not contained in V_S. The indices of all the non-zero entries in x constitute the set of vertex indices S, that is, $S = \{i \mid x_i > 0\}$. Similarly, we have $V_S = \{v_i \mid x_i > 0\}$. Since $\mathbf{x}$ determines S, which further determines $\mathcal{G}_S$ and G_S, we abuse the meaning of "induce" to say $\mathcal{G}_S$ and G_S are **induced** by $\mathbf{x}$.

When the context is clear, we write "signed uncertain graph", "signed uncertain subgraph", and "signed uncertain edges" in short as "graph", "subgraph" and "edges", respectively.

3.0.2 Modelling a Group of Antagonistic Communities

In this subsection, we first introduce how to model a group of antagonistic communities, then we discuss how to measure the significance of a group of antagonistic communities.

A **group of k antagonistic communities** is modelled by a set of k signed uncertain subgraphs, denoted by $\mathbb{G} = \{\mathcal{G}_{S_1}, \ldots, \mathcal{G}_{S_k}\}$, where $\mathcal{G}_{S_1}, \ldots, \mathcal{G}_{S_k}$ are the subgraphs induced by the sets of vertex indices $S_1, \ldots, S_k$, respectively. Each subgraph in $\mathbb{G}$ represents an antagonistic community. By mapping each of the $S_j, j \in \{1, \ldots, k\}$, into an embedding vector $\mathbf{x}_j \in \Delta^n$, we can put the k embedding vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k$ in an n-by-k embedding matrix, denoted by X. The j-th column of X, denoted by $X_{:,j}$, is the embedding vector of S_j, which induces the j-th community $\mathcal{G}_{S_j}$.

To measure the significance of a group of k antagonistic communities represented by X, we first measure the sum of the overall friendship and the overall antagonism of X on a possible world G_h sampled from $\mathcal{G}$, and then we take the expectation of the sum with respect to the probability distribution of the 3^m possible worlds of $\mathcal{G}$.

The **overall friendship** of X on a possible world G_h is measured by

$$\texttt{OF}(X, G_h) = \sum_{j=1}^{k} d_h^+(X_{:,j}), \tag{3.0.1}$$

where k is the number of communities and

$$d_h^+(X_{:,j}) = \sum_{a=1}^{n} \sum_{b=1}^{n} X_{a,j} X_{b,j} (A_{G_h}^+)_{a,b} \tag{3.0.2}$$

is borrowed from [30, 23, 7, 13] to measure the weighted average friendship of all the edges in the static subgraph induced by $X_{:,j}$ from G_h. The meaning of $d_h^+(X_{:,j})$ can be

derived from the facts that the positive edge weight $(A^+_{G_h})_{a,b}$ represents the strength of friendship between the vertices v_a and v_b in G_h, and the scalar values $X_{a,j}$ and $X_{b,j}$ are the non-negative weights of v_a and v_b in the community $X_{:,j}$. In summary, $\texttt{OF}(X, G_h)$ is the sum of the weighted average friendship within the k subgraphs induced by X from G_h.

The **overall antagonism** of X is measured by

$$\texttt{OA}(X, G_h) = \sum_{j \neq l} d^-_h(X_{:,j}, X_{:,l}), \tag{3.0.3}$$

where the operation $\sum_{j \neq l}$ means the summation over all the cases of (j, l) in $\{(j, l) \mid j \in \{1, \ldots, k\}, l \in \{1, \ldots, k\}, j \neq l\}$, and the term

$$d^-_h(X_{:,j}, X_{:,l}) = \sum_{a=1}^{n} \sum_{b=1}^{n} X_{a,j} X_{b,l} (A^-_{G_h})_{a,b} \tag{3.0.4}$$

measures the weighted average antagonism of all the edges between the two static subgraphs induced by $X_{:,j}$ and $X_{:,l}$ from G_h. In summary, $\texttt{OA}(X, G_h)$ is the sum of the weighted average antagonism between every pair of the subgraphs induced by X from G_h.

The **sum of the overall friendship and overall antagonism** of X on G_h is

$$\texttt{SUM}(X, G_h) = \texttt{OF}(X, G_h) + \alpha\, \texttt{OA}(X, G_h), \tag{3.0.5}$$

where $\alpha > 0$ is a hyperparameter to balance the importance of overall friendship and overall antagonism when measuring the significance of X.

Since $\mathcal{G}$ describes a probability distribution of 3^m possible worlds, we compute the

significance of X on $\mathcal{G}$ as the expectation of $\texttt{SUM}(X, G_h)$, that is,

$$\texttt{SIG}(X) = \sum_{h=1}^{3^m} \mathbb{P}(G_h)\, \texttt{SUM}(X, G_h), \tag{3.0.6}$$

where m is the number of edges in $\mathcal{G}$ and $\mathbb{P}(G_h)$ is the probability of sampling the possible world G_h from $\mathcal{G}$,

3.0.3 The Problem of Finding Antagonistic Communities in Signed Uncertain Graphs

The significance $\texttt{SIG}(X)$ measures the expected sum of the overall friendship and overall antagonism of a group of k antagonistic communities. Recall that the goal of our task is to find multiple groups of antagonistic communities that are expected to have a high friendship within each community and a high antagonism between different communities. This goal can be mathematically interpreted as finding the local maximum points of $\texttt{SIG}(X)$, because the group of k antagonistic communities induced by a local maximum point of $\texttt{SIG}(X)$ has a locally maximum expected sum of the overall friendship and overall antagonism.

Following the above insight, we formulate the problem of **finding antagonistic communities in signed uncertain graphs** (FACSUG) as the following constrained quadratic programming problem

$$\begin{aligned} &\max_{X \in \Delta^{n \times k}} F(X) \\ &F(X) = \texttt{SIG}(X) - \beta \sum_{j \neq l} X_{:,j}^{\top} X_{:,l}, \end{aligned} \tag{3.0.7}$$

where $X \in \Delta^{n \times k}$ means each column of X is an embedding vector in Δ^n, the term $\sum_{j \neq l} X_{:,j}^\top X_{:,l}$ penalizes the overlap between different communities in X, and $\beta > 0$ is a hyperparameter controlling the strength of penalization. Since our goal is to find non-overlapping communities, we use a trivially large value $\beta = 50$ by default to force the term $\sum_{j \neq l} X_{:,j}^\top X_{:,l}$ to be zero.

We regard every local maximum point of $F(X)$ as a group of k antagonistic communities. However, different local maximum points may have different significance. In real-world applications, we are more interested in finding the groups of antagonistic communities with large values of significance $F(X)$. Since every **KKT point** satisfying the KKT conditions [21] is a potential local maximum point of $F(X)$, we focus on designing a solution to find the KKT points with large values of $F(X)$.

Chapter 4

Solution

In this section, we first convert the FACSUG problem into a more compact matrix version that is easier to solve. Then, we analyze several important properties of the KKT points of $F(X)$, which makes it possible to quickly find the KKT points. At last, we present an efficient algorithm to find the KKT points.

4.0.1 Converting the FACSUG Problem

The original form of the FACSUG problem in Equation (3.0.7) is not directly tractable in practical time because the term $\texttt{SIG}(X)$ requires enumerating the 3^m possible worlds of $\mathcal{G}$. To tackle this issue, we conduct the following transformation to convert the FACSUG problem into a matrix version that is easier to solve.

Since $\texttt{SIG}(X)$ is the expectation of $\texttt{SUM}(X, G_h)$ with respect to the probability distribution $\mathcal{D}$ of the possible worlds of $\mathcal{G}$, we can rewrite it as

$$\texttt{SIG}(X) = \mathbb{E}\big(\texttt{SUM}(X, G_h)\big), \tag{4.0.1}$$

where $\mathbb{E}$ represents the expectation with respect to the probability distribution $\mathcal{D}$. By plugging Equations (3.0.1)-(3.0.5) into Equation (4.0.1), we can derive

$$\begin{aligned} \texttt{SIG}(X) &= \mathbb{E}\big(\texttt{OF}(X, G_h)\big) + \alpha\, \mathbb{E}\big(\texttt{OA}(X, G_h)\big) \\ &= \sum_{j=1}^{k} \mathbb{E}\big(d_h^+(X_{:,j})\big) + \alpha \sum_{j \neq l} \mathbb{E}\big(d_h^-(X_{:,j}, X_{:,l})\big), \end{aligned} \tag{4.0.2}$$

where

$$\begin{aligned} \mathbb{E}\big(d_h^+(X_{:,j})\big) &= \sum_{a=1}^{n} \sum_{b=1}^{n} X_{a,j} X_{b,j} \mathbb{E}\big((A^+_{G_h})_{a,b}\big) \\ &= \sum_{a=1}^{n} \sum_{b=1}^{n} X_{a,j} X_{b,j} A^+_{i,j} P^+_{i,j} \\ &= X_{:,j}^\top (A^+ \circ P^+) X_{:,j}, \end{aligned} \tag{4.0.3}$$

and

$$\begin{aligned} \mathbb{E}\big(d_h^-(X_{:,j}, X_{:,l})\big) &= \sum_{a=1}^{n} \sum_{b=1}^{n} X_{a,j} X_{b,l} \mathbb{E}\big((A^-_{G_h})_{a,b}\big) \\ &= \sum_{a=1}^{n} \sum_{b=1}^{n} X_{a,j} X_{b,l} A^-_{a,b} P^-_{a,b} \\ &= X_{:,j}^\top (A^- \circ P^-) X_{:,l}. \end{aligned} \tag{4.0.4}$$

Here, the operator $\circ$ represents the elementwise multiplication between two matrices.

By plugging Equations (4.0.2)-(4.0.4) into Equation (3.0.7), we can rewrite the

FACSUG problem as

$$\begin{aligned} &\max_{X \in \Delta^{n \times k}} F(X) \\ &F(X) = \sum_{j=1}^{k} X_{:,j}^{\top} (A^{+} \circ P^{+}) X_{:,j} + \sum_{j \neq l} X_{:,j}^{\top} H X_{:,l}, \end{aligned} \tag{4.0.5}$$

where $H = \alpha A^{-} \circ P^{-} - \beta I$ is an n-by-n matrix.

The above form of the FACSUG problem does not require enumerating of the 3^m possible worlds of $\mathcal{G}$. Thus it is easier to solve compared to the original form of the FACSUG problem. A straightforward idea to find the KKT points of $F(X)$ in the FACSUG problem is to use a classic gradient-based method such as proximal gradient descent [29]. However, such a method generally requires to operate with the entire n-by-n matrices such as A^{+}, A^{-}, P^{+}, and P^{-}, where n is the number of vertices in $\mathcal{G}$. When the graph $\mathcal{G}$ contains a large number of vertices, the matrices will be huge. Thus a classic gradient-based method often cannot work very efficiently to find the KKT points of $F(X)$.

In the following subsection, we discuss some properties of the KKT points of $F(X)$. These properties will be used later to develop an efficient algorithm to find the KKT points.

4.0.2 The Properties of KKT Points

For any $j \in \{1, \ldots, k\}$, denote by $M_{:,j} = \sum_{j \neq l} H X_{:,l}$ the j-th column of an n-by-k matrix M, we can represent the sum of the terms in $F(X)$ that are related to $X_{:,j}$ as

$$f(X_{:,j}) = X_{:,j}^{T} (A^{+} \circ P^{+}) X_{:,j} + 2 X_{:,j}^{\top} M_{:,j} \tag{4.0.6}$$

Next, we analyze the relationship between the KKT points of $f(X_{:,1}), \ldots, f(X_{:,k})$ and a KKT point of $F(X)$ in the following theorem.

Theorem 1. *Given an embedding matrix $X^* \in \Delta^{n\times k}$, if for all $j \in [1,k]$, $X^*_{:,j}$ is a KKT point of $f(X_{:,j})$, then X^* is a KKT point of $F(X)$.*

Proof. Recall that the constraint $X \in \Delta^{n\times k}$ implies, for each $j \in \{1,\ldots,k\}$, $X_{:,j} \in \Delta^n$. We can write the Lagrangian function of $F(X)$ as

$$\mathcal{L}_F = F(X) + \sum_{j=1}^{k}\Big(\sum_{i=1}^{n}\mu_{i,j}X_{i,j} - \lambda_j(\sum_{i=1}^{n}X_{i,j} - 1)\Big), \tag{4.0.7}$$

where $\mu_{i,j}$ and λ_j are Lagrangian multipliers. Similarly, since $X_{:,j} \in \Delta^n$, the Lagrangian function of $f(X_{:,j})$ is

$$\mathcal{L}_f = f(X_{:,j}) + \sum_{i=1}^{n}\mu_{i,j}X_{i,j} - \lambda_j(\sum_{i=1}^{n}X_{i,j} - 1). \tag{4.0.8}$$

Next, we show that the following KKT conditions hold for X^* to be a KKT point of $F(X)$.

Stationarity. Since $f(X_{:,j})$ is the sum of all the terms in $F(X)$ that are related to $X_{:,j}$, we have

$$\nabla_{X_{:,j}}F(X) = \nabla_{X_{:,j}}f(X_{:,j}), \tag{4.0.9}$$

where ∇ is the gradient operator. Then, we can derive from Equations (4.0.7)-(4.0.9) that

$$\nabla_{X_{:,j}}\mathcal{L}_F = \nabla_{X_{:,j}}\mathcal{L}_f, \tag{4.0.10}$$

where $\nabla_{X_{:,j}}\mathcal{L}_F$ is exactly the j-th column of $\nabla_X\mathcal{L}_F$, denoted by $(\nabla_X\mathcal{L}_E)_{:,j}$. Since $\forall j \in \{1,\ldots,k\}$, $X^*_{:,j}$ is a KKT point of $f(X_{:,j})$, we have $\nabla_{X^*_{:,j}}\mathcal{L}_f = \mathbf{0}, \forall j \in \{1,\ldots,k\}$,

which further implies $\nabla_{X^*}\mathcal{L}_E = \mathbf{0}$. Thus, the stationarity holds.

Primal feasibility. Since $X_{:,j} \in \Delta^n$, $\forall j \in \{1, \ldots, k\}$, we have $X \in \Delta^{n\times k}$. Thus, the primal feasibility holds.

Dual feasibility. Since $\forall j \in \{1, \ldots, k\}$, $X^*_{:,j}$ is a KKT point of $f(X_{:,j})$, we have $\forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, k\}, \mu_{i,j} \geq 0$, which means the dual feasibility holds.

Complementary slackness. Since $\forall j \in \{1, \ldots, k\}$, $X^*_{:,j}$ is a KKT point of $f(X_{:,j})$, we have $\forall j \in \{1, \ldots, k\}, \sum_{i=1}^{n} \mu_{i,j} X^*_{i,j} = 0$, which further implies $\sum_{j=1}^{k} \sum_{i=1}^{n} \mu_{i,j} X^*_{i,j} = 0$. Thus, the complementary slackness holds. The theorem follows. □

According to Theorem 1, we can find a KKT point of $F(X)$ by finding the KKT points of $f(X_{:,1}), \ldots, f(X_{:,k})$. This allows us to divide the original FACSUG problem into k sub-problems, each focusing on finding a KKT point $X^*_{:,j}$ of $f(X_{:,j})$. However, directly solving each sub-problem by a classic gradient-based method is still not efficient because it also requires manipulating the large matrices such as A^+, A^-, P^+, and P^-, which is inefficient when the graph contains many vertices.

Next, we introduce some new notations and discuss a useful property of a KKT point of $f(X_{:,j})$, which makes it possible to develop an efficient algorithm to find a KKT point of $f(X_{:,j})$.

For any $X_{:,j} \in \Delta^n$, denote by

$$R(X_{:,j}) = (A^+ \circ P^+)X_{:,j} + M_{:,j} \tag{4.0.11}$$

an n-dimensional column vector, where $R_i(X_{:,j})$ represents the i-th entry in $R(X_{:,j})$. Denote by

$$Q(X_{:,j}) = (X_{:,j})^\top R(X_{:,j}) \tag{4.0.12}$$

a real valued scalar, and by $U = \{1, \ldots, n\}$ the set of indices of all the vertices in $\mathcal{G}$.

The following theorem shows a sufficient and necessary condition to examine a KKT point of $f(X_{:,j})$.

Theorem 2. *$X^*_{:,j} \in \Delta^n$ is a KKT point of $f(X_{:,j})$ if and only if*

$$R_i(X^*_{:,j}) \begin{cases} = Q(X^*_{:,j}) & \text{if} \quad i \in S^*_j \\ \leq Q(X^*_{:,j}) & \text{if} \quad i \in U \setminus S^*_j, \end{cases} \tag{4.0.13}$$

where $S^*_j = \{i \in U \mid X^*_{i,j} > 0\}$ is the set of vertex indices determined by $X^*_{:,j}$.

Proof. We first write the Lagrangian function of $f(X_{:,j})$ as

$$\mathcal{L}_f = f(X_{:,j}) + \sum_{i=1}^{n} \mu_{i,j} X_{i,j} - \lambda_j (\sum_{i=1}^{n} X_{i,j} - 1). \tag{4.0.14}$$

where $\mu_{i,j}$ and λ_j are Lagrangian multipliers. Then, we prove the necessity and sufficiency as follows.

Necessity. When Equation (4.0.13) holds, we prove $X^*_{:,j}$ is a KKT point of $f(X_{:,j})$ by constructing the following Lagrangian multipliers to make the KKT conditions hold.

Let $\lambda_j = 2Q(X^*_{:,j})$ and

$$\mu_{i,j} = \begin{cases} 0 & \text{if} \quad i \in S^*_j \\ \lambda_j - 2R_i(X^*_{:,j}) & \text{if} \quad i \in U \setminus S^*_j, \end{cases} \tag{4.0.15}$$

we prove the KKT conditions hold as follows.

Stationarity. Recall that $\nabla_{X^*_{:,j}} \mathcal{L}_f = 2R_i(X^*_{:,j}) + \mu_{i,j} - \lambda_j$. When $i \in U \setminus S^*_j$, we have

$\mu_{i,j} = \lambda_j - 2R_i(X^*_{:,j})$ by Equation (4.0.15), which implies $\nabla_{X^*_{:,j}} \mathcal{L}_f = 0$. When $i \in S^*_j$, we have $\mu_{i,j} = 0$ by Equation (4.0.15), which means $\nabla_{X^*_{:,j}} \mathcal{L}_f = 2R_i(X^*_{:,j}) - 2Q(X^*_{:,j})$. By Equation (4.0.13), $R_i(X^*_{:,j}) = Q(X^*_{:,j})$ since $i \in S^*_j$, thus $\nabla_{X^*_{:,j}} \mathcal{L}_f = 0$. In sum, $\nabla_{X^*_{:,j}} \mathcal{L}_f = 0, \forall i \in \{1, \ldots, n\}$, thus the stationarity condition holds.

Primal feasibility. The primal feasibility condition trivially holds because $X^*_{:,j} \in \Delta^n$.

Dual feasibility. When $i \in S^*_j$, we have $\mu_{i,j} = 0$ by Equation (4.0.15). When $i \in U \setminus S^*_j$, we have $\mu_{i,j} = \lambda_j - 2R_i(X^*_{:,j}) = 2Q(X^*_{:,j}) - 2R_i(X^*_{:,j})$. By Equation (4.0.13), $R_i(X^*_{:,j}) \leq Q(X^*_{:,j})$ since $i \in S^*_j$, thus $\mu_{i,j} \geq 0$. In sum, $\mu_{i,j} \geq 0, \forall i \in \{1, \ldots, n\}$, thus the dual feasibility condition holds.

Complementary slackness. When $i \in S^*_j$, we have $\mu_{i,j} = 0$ by Equation (4.0.15). When $i \in U \setminus S^*_j$, we have $X^*_{i,j} = 0$. Therefore, $\sum_{i=1}^n \mu_{i,j} X^*_{i,j} = \sum_{i \in S^*_j} 0 \cdot X^*_{i,j} + \sum_{i \in U \setminus S^*_j} \mu_{i,j} \cdot 0 = 0$, which means the complementary slackness condition holds.

In conclusion, $X^*_{:,j} \in \Delta^n$ is a KKT point of $f(X^*_{:,j})$.

Sufficiency. Since $X^*_{:,j} \in \Delta^n$ is a KKT point of $f(X_{:,j})$, $X^*_{:,j}$ satisfies the KKT conditions. By the stationarity condition, we have $\forall i \in \{1, \ldots, n\}$,

$$2\big((A^+ \circ P^+)X^*_{:,j}\big)_i + 2M_{i,j} + \mu_{i,j} - \lambda_j = 0, \tag{4.0.16}$$

which can be rewritten as

$$2R_i(X^*_{:,j}) + \mu_{i,j} - \lambda_j = 0. \tag{4.0.17}$$

According to the complementary slackness condition, $\sum_{i=1}^n \mu_{i,j} X^*_{i,j} = 0$. Since $X^*_{i,j} \geq 0$, $\mu_{i,j} \geq 0$ due to the dual feasibility condition, and $X^*_{i,j} > 0, \forall i \in S^*_j$, by the

definition of S_j^*, we know $\forall i \in S_j^*,\ \mu_{i,j} = 0$.

Therefore, if $i \in S_j^*$, then $\mu_{i,j} = 0$ and we can derive from Equation (4.0.17) that $R_i(X_{:,j}^*) = \frac{\lambda_j}{2}$. If $i \in U \setminus S_j^*$, then we have $2R_i(X_{:,j}^*) - \lambda_j = -\mu_{i,j} \leq 0$, which means $R_i(X_{:,j}^*) \leq \frac{\lambda_j}{2}$.

By definition of $Q(X_{:,j}^*)$ and the primal feasibility condition, we can derive

$$\begin{aligned} Q(X_{:,j}^*) &= \sum_{i=1}^{n} X_{i,j}^* R_i(X_{:,j}^*) \\ &= \sum_{i \in S_j^*} X_{i,j}^* \cdot \frac{\lambda_j}{2} + \sum_{i \in U \setminus S_j^*} R_i(X_{:,j}^*) \cdot 0 \\ &= \frac{\lambda_j}{2}. \end{aligned} \tag{4.0.18}$$

Now, we can summarize the above to get

$$R_i(X_{:,j}^*) \begin{cases} = Q(X_{:,j}^*) & \text{if} \quad i \in S_j^* \\ \leq Q(X_{:,j}^*) & \text{if} \quad i \in U \setminus S_j^* \end{cases} \tag{4.0.19}$$

□

We can learn from Theorem 2 that if a point $X_{:,j}^*$ satisfies the conditions in Equation (4.0.13), then it is a KKT point of $f(X_{:,j})$ in $\mathcal{G}$. This property also applies to any subgraph $\mathcal{G}_{S_j}$ induced by a set of vertex indices S_j. Specifically, denote by $\hat{S}_j = \{i \mid \hat{X}_{i,j} > 0\}$ a subset of S_j, and $\hat{S}_j$ is determined by a point $\hat{X}_{:,j} \in \triangle^n$ where $\hat{X}_{i,j} = 0$ when $i \notin S_j$. If $\hat{X}_{:,j}$ satisfies

$$R_i(\hat{X}_{:,j}) \begin{cases} = Q(\hat{X}_{:,j}) & \text{if} \quad i \in \hat{S}_j \\ \leq Q(\hat{X}_{:,j}) & \text{if} \quad i \in S_j \setminus \hat{S}_j, \end{cases} \tag{4.0.20}$$

then we can derive from Theorem 2 that $\hat{X}_{:,j}$ is a KKT point of $f(X_{:,j})$ in $\mathcal{G}_{S_j}$.

However, since the condition term $S_j \setminus \hat{S}_j$ in Equation (4.0.20) is not equal to the condition term $U \setminus S_j^*$ in Equation (4.0.13), the point $\hat{X}_{:,j}$ satisfying Equation (4.0.20) may not satisfy Equation (4.0.13). This means the KKT point $\hat{X}_{:,j}$ of $f(X_{:,j})$ in $\mathcal{G}_{S_j}$ may not be a KKT point of $f(X_{:,j})$ in $\mathcal{G}$.

Next, we show a necessary and sufficient condition that ensures a KKT point $\hat{X}_{:,j}$ in $\mathcal{G}_{S_j}$ is also a KKT point in $\mathcal{G}$.

Corollary 1. *A KKT point $\hat{X}_{:,j}$ of $f(X_{:,j})$ in a subgraph $\mathcal{G}_{S_j}$ of the graph $\mathcal{G}$ is also a KKT point of $f(X_{:,j})$ in $\mathcal{G}$ if and only if $\Omega_j = \{i \in U \setminus \hat{S}_j \mid R_i(\hat{X}_{:,j}) > Q(\hat{X}_{:,j})\} = \emptyset$.*

Proof. **Necessity.** We prove necessity by contradiction, that is, we aim to prove if $\Omega_j \neq \emptyset$ then $\hat{X}_{:,j}$ is not a KKT point of $f(X_{:,j})$ in $\mathcal{G}$. The proof is as follows. If $\Omega_j \neq \emptyset$, then $\exists i \in U \setminus \hat{S}_j$ such that $R_i(\hat{X}_{:,j}) > Q(\hat{X}_{:,j})$. This means the second condition in Equation (4.0.13) does not hold for $\hat{X}_{:,j}$. Therefore, we can derive from Theorem 2 that $\hat{X}_{:,j}$ is not a KKT point of $f(X_{:,j})$ in $\mathcal{G}$.

Sufficiency. If $\Omega_j = \emptyset$, then $\forall i \in U \setminus \hat{S}_j$, $R_i(\hat{X}_{:,j}) \leq Q(\hat{X}_{:,j})$. Additionally, since $\hat{X}_{:,j}$ is a KKT point of $f(X_{:,j})$ in $\mathcal{G}_{S_j}$, we can derive from Theorem 2 and Equation (4.0.20) that $\forall i \in \hat{S}_j$, $R_i(\hat{X}_{:,j}) = Q(\hat{X}_{:,j})$. In summary, $\hat{X}_{:,j}$ satisfies

$$R_i(\hat{X}_{:,j}) \begin{cases} = Q(\hat{X}_{:,j}) & \text{if} \quad i \in \hat{S}_j \\ \leq Q(\hat{X}_{:,j}) & \text{if} \quad i \in U \setminus \hat{S}_j. \end{cases} \tag{4.0.21}$$

Therefore, we can derive from Theorem 2 that $\hat{X}_{:j}$ is also a KKT point of $f(X_{:,j})$ in $\mathcal{G}$. □

In the next subsection, we illustrate how to leverage the previously shown properties of KKT points to quickly find a KKT point of $f(X_{:,j})$ in $\mathcal{G}$. Later, in Subsection 4.0.4, we will summarize the complete algorithm to efficiently find the KKT points of $F(X)$.

4.0.3 Finding a KKT Piont of $f(X_{:,j})$ in $\mathcal{G}$

In this subsection, we aim to find a KKT point of the following optimization problem.

$$\begin{aligned} &\max_{X_{:,j} \in \Delta^n} f(X_{:,j}) \\ &f(X_{:,j}) = X_{:,j}^\top B_j X_{:,j}, \end{aligned} \tag{4.0.22}$$

where $B_j = (A^+ \circ P^+) + \boldsymbol{e} M_{:,j}^\top + M_{:,j} \boldsymbol{e}^\top$ is a matrix with n columns and n rows, and $\boldsymbol{e}$ is an n-dimensional column vector with all the entries equal to 1.

When B_j is materialized (i.e., computed and stored), the above optimization problem can be solved by many existing methods [30, 7, 23]. However, since B_j is not a sparse matrix, it is challenging to materialize B_j when $\mathcal{G}$ contains a large number of vertices. In this case, if B_j is not materialized, we cannot straightforwardly apply the existing methods [30, 7, 23] to efficiently find a solution to the optimization problem in Equation (4.0.22).

To tackle this challenge caused by B_j, we propose the **locate and update algorithm (LUA)** to efficiently find a KKT point of $f(X_{:,j})$ in $\mathcal{G}$ without fully materializing all the entries in B_j. The key idea is to iteratively conduct a **locate stage** and an **update stage**, where each iteration only materializes a small proportion of the entries in B_j.

More specifically, given an initial $X_{:,j}$ that induces an initial subgraph $\mathcal{G}_{S_j}$, the locate stage finds a KKT point $\hat{X}_{:,j}$ within the subgraph $\mathcal{G}_{S_j}$ and reduces $\mathcal{G}_{S_j}$ into its subgraph $\mathcal{G}_{\hat{S}_j}$, where $\hat{S}_j = \{i \mid \hat{X}_{i,j} > 0\}$ is a subset of S_j. The update stage further increases the value of $f(X_{:,j})$ by adding the vertex indices in Ω_j to $\hat{S}_j$. This produces the updated S_j to induce a new subgraph $\mathcal{G}_{S_j}$, which is then used to conduct the locate stage in the next round of iteration. The above iteration continues until $\Omega_j = \emptyset$, then, according to Corollary 1, $\hat{X}_{:,j}$ is a KKT point of $f(X_{:,j})$ in $\mathcal{G}$.

Since each of the above iterations is conducted on a small subgraph $\mathcal{G}_{S_j}$, we only need to materialize the small set of entries in B_j that corresponds to the edges in $\mathcal{G}_{S_j}$. This avoids the challenge of fully materializing all the entries in B_j.

Next, we introduce the details of the locate stage and the update stage of LUA.

Locate stage. Given an initial value of $X_{:,j}$ that induces a subgraph $\mathcal{G}_{S_j}$, the locate stage finds a KKT point $\hat{X}_{:,j}$ of $f(X_{:,j})$ in $\mathcal{G}_{S_j}$ by the Replicator Dynamics (RD) method [39]. The t-th iteration of RD is conducted by

$$X_{i,j}^{t+1} = X_{i,j}^{t} \frac{(B_j X_{:,j}^{t})_i}{(X_{:,j}^{t})^\top B_j X_{:,j}^{t}}, \tag{4.0.23}$$

where $(B_j X_{:,j}^{t})_i$ is the i-th entry of the n-dimensional vector $B_j X_{:,j}^{t}$. According to [39], the above iteration converges to a KKT point $\hat{X}_{:,j} \in \Delta^n$, which induces a subgraph $\mathcal{G}_{\hat{S}_j}$ contained in $\mathcal{G}_{S_j}$ and $\hat{S}_j = \{i \mid \hat{X}_{i,j} > 0\}$ is a subset of $S_j = \{i \mid X_{i,j} > 0\}$.

We can derive from Equation (4.0.23) that, if $\exists t$ such that $X_{i,j}^{t} = 0$, then $X_{i,j}^{t+1} = 0$. This means the computation is confined within the subgraph $\mathcal{G}_{S_j}$ induced by the initial value of $X_{:,j}$. Since $\forall i \notin S_j, X_{i,j}^{t} = 0$, we only need to compute and store a small sub-matrix of B_j, denoted by B_{S_j}, which corresponds to the set of vertex indices in S_j. This avoids the challenge of fully materializing B_j, thus largely promotes the

efficiency of the locate stage of LUA.

By Corollary 1, if $\Omega_j = \emptyset$, then the KKT point $\hat{X}_{:,j}$ found in $\mathcal{G}_{S_j}$ by the locate stage is also a KKT point in graph $\mathcal{G}$. In this case, the iteration of LUA stops. If $\Omega_j \neq \emptyset$, then we can derive from Corollary 1 that $\hat{X}_{:,j}$ is not a KKT point in $\mathcal{G}$. Therefore, we conduct the following update stage to further update $\hat{X}_{:,j}$.

Update stage. In this stage, we update $\hat{X}_{:,j}$ by an n-dimensional vector $\mathbf{b}$ and an optimal step size $\sigma > 0$, such that $f(\hat{X}_{:,j} + \sigma\mathbf{b})$ is larger than $f(\hat{X}_{:,j})$ and the constraint $(\hat{X}_{:,j} + \sigma\mathbf{b}) \in \Delta^n$ is satisfied.

Denote by $\mathbf{b} = [b_1, \ldots, b_n]$ and by $s = \sum_{i\in\Omega_j} b_i$. We define the i-th entry of $\mathbf{b}$ as

$$b_i = \begin{cases} R_i(\hat{X}_{:,j}) - Q(\hat{X}_{:,j}) & \text{if} \quad i \in \Omega_j \\ -s\hat{X}_{i,j} & \text{if} \quad i \in \hat{S}_j \\ 0 & \text{otherwise} \end{cases} \tag{4.0.24}$$

Next, we show that the constraint $(\hat{X}_{:,j} + \sigma\mathbf{b}) \in \Delta^n$ always holds for all step size $\sigma \in [0, \frac{1}{s}]$.

Observation 1. *If $\sigma \in [0, \frac{1}{s}]$, then $(\hat{X}_{:,j} + \sigma\mathbf{b}) \in \Delta^n$.*

Proof. Since $\sum_{i\in\hat{S}_j} \hat{X}_{i,j} = 1$, we have

$$\sum_{i=1}^{n} b_i = \sum_{i\in\Omega_j} b_i + \sum_{i\in\hat{S}_j} b_i = s - s = 0,$$

which means

$$\sum_{i=1}^{n} (\hat{X}_{i,j} + \sigma b_i) = \sum_{i=1}^{n} \hat{X}_{i,j} + \sigma \sum_{i=1}^{n} b_i = 1.$$

According to the definition of Ω_j in Corollary 1, we have $\forall i \in \Omega_j, R_i(\hat{X}_{:,j}) - Q(\hat{X}_{:,j}) > 0$. Then, we can derive from Equation (4.0.24) that $\forall i \in \Omega_j, b_i > 0$, which means $s = \sum_{i\in\Omega_j} b_i > 0$. Since $\sigma \in [0, \frac{1}{s}]$ and $s > 0$, we can derive from Equation (4.0.24) that

$$\hat{X}_{i,j} + \sigma b_i \geq 0, \forall i \in [1, n].$$

In conclusion, we have $(\hat{X}_{:,j} + \sigma\mathbf{b}) \in \Delta^n$. □

Now, we derive the optimal step size $\sigma^* \in [0, \frac{1}{s}]$ to maximize $f(\hat{X}_{:,j}+\sigma\mathbf{b}) - f(\hat{X}_{:,j})$ and show that $f(\hat{X}_{:,j} + \sigma^*\mathbf{b})$ is always larger than $f(\hat{X}_{:,j})$.

We first rewrite $f(\hat{X}_{:,j} + \sigma\mathbf{b}) - f(\hat{X}_{:,j})$ as the following quadratic form of σ.

$$\begin{aligned}
&f(\hat{X}_{:,j} + \sigma\mathbf{b}) - f(\hat{X}_{:,j}) \\
&= \mathbf{b}^\top(A^+ \circ P^+)\mathbf{b}\sigma^2 + 2\mathbf{b}^\top\big((A^+ \circ P^+)\hat{X}_{:,j} + M_{:,j}\big)\sigma \\
&= \mathbf{b}^\top(A^+ \circ P^+)\mathbf{b}\sigma^2 + 2\mathbf{b}^\top R(\hat{X}_{:,j})\sigma \\
&= \mathbf{b}^\top(A^+ \circ P^+)\mathbf{b}\sigma^2 + 2(\sum_{i\in\Omega_j} b_i^2)\sigma
\end{aligned} \tag{4.0.25}$$

Since $\Omega_j \neq \emptyset$, we know $\sum_{i\in\Omega_j} b_i^2 > 0$. In this case, the optimal step size σ^* is

$$\sigma^* = \begin{cases} \min\left(\frac{1}{s}, -\frac{\sum_{i\in\Omega_j} b_i^2}{\mathbf{b}^\top(A^+\circ P^+)\mathbf{b}}\right) & \text{if} \quad \mathbf{b}^\top(A^+ \circ P^+)\mathbf{b} < 0 \\ \frac{1}{s} & \text{if} \quad \mathbf{b}^\top(A^+ \circ P^+)\mathbf{b} \geq 0. \end{cases} \tag{4.0.26}$$

We can easily derive from the quadratic form of σ in Equation (4.0.25) that $f(\hat{X}_{:,j} + \sigma^*\mathbf{b})$ is always larger than $f(\hat{X}_{:,j}) > 0$.

Specifically, when $\mathbf{b}^\top(A^+ \circ P^+)\mathbf{b} < 0$, setting

$$\sigma^* = \min\left(\frac{1}{s}, -\frac{\sum_{i\in\Omega_j} b_i^2}{\mathbf{b}^\top(A^+ \circ P^+)\mathbf{b}}\right)$$

ensures $f(\hat{X}_{:,j} + \sigma^*\mathbf{b}) - f(\hat{X}_{:,j}) > 0$. When $\mathbf{b}^\top(A^+ \circ P^+)\mathbf{b} \geq 0$, setting $\sigma^* = \frac{1}{s}$ also ensures $f(\hat{X}_{:,j} + \sigma^*\mathbf{b}) - f(\hat{X}_{:,j}) > 0$. In conclusion, by using the optimal step size σ^* in Equation (4.0.26), $f(\hat{X}_{:,j} + \sigma^*\mathbf{b})$ is always larger than $f(\hat{X}_{:,j})$.

Considering that $\sigma^* > 0$ and $\forall i \in \Omega_j, b_i > 0$, updating $\hat{X}_{:,j}$ by $\hat{X}_{:,j} + \sigma^*\mathbf{b}$ is essentially assigning a positive value to the i-th entry in $\hat{X}_{:,j} + \sigma^*\mathbf{b}$ for all $i \in \Omega_j$. This is essentially adding the vertex indices in Ω_j to $\hat{S}_j$. Therefore, when $\Omega_j \neq \emptyset$, the update stage increases the value of $f(\hat{X}_{:,j})$ by adding the vertex indices in Ω_j to $\hat{S}_j$.

We summarize the pseudocode of LUA in Algorithm 1. The computation of LUA is very efficient because both the locate stage and the update stage are conducted on a small subgraph $\mathcal{G}_{S_j}$ instead of the entire graph $\mathcal{G}$.

The LUA algorithm converges because each step of iteration monotonically increases the value of $f(X_{:,j})$ and the function value of $f(X_{:,j})$ has a constant upper bound. When LUA converges, we have $\Omega_j = \emptyset$. Thus, by Corollary 1, the $X^*_{:,j}$ returned by LUA is a KKT point of $f(X_{:,j})$ in $\mathcal{G}$. In conclusion, LUA converges to a KKT point $X^*_{:,j}$ of $f(X_{:,j})$ in $\mathcal{G}$.

In the following subsection, we introduce how to use LUA to find the KKT points of $F(X)$ in Equation (4.0.5).

Algorithm 1 Locate and Update Algorithm

Input: An initial value $X^0_{:,j} \in \Delta^n$.
Output: A KKT point $X^*_{:,j}$ of $f(X_{:,j})$ in $\mathcal{G}$.

1: Set $\hat{X}_{:,j} = X^0_{:,j}$.
2: **repeat**
3: **Update stage:** if $\Omega_j \neq \emptyset$, then $X_{:,j} \leftarrow (\hat{X}_{:,j} + \sigma^* \mathbf{b})$.
4: **Locate stage:** Use $X_{:,j}$ as the initial value to find a KKT point $\hat{X}_{:,j}$ by the RD method in Equation 4.0.23.
5: **until** $\Omega_j = \emptyset$.
6: $X^*_{:,j} \leftarrow \hat{X}_{:,j}$.
7: **return** A KKT point $X^*_{:,j}$ of $f(X_{:,j})$ in $\mathcal{G}$.

Algorithm 2 Finding a KKT Point of $F(X)$

Input: An initial value of $X^0 \in \Delta^{n \times k}$
Output: A KKT point X^* of $F(X)$ in Equation (4.0.5).

1: set $X^* = X^0$.
2: **repeat**
3: **for** each $j \in \{1, \ldots, k\}$ **do**
4: Find a KKT point $X^*_{:,j}$ of $f(X_{:,j})$ in $\mathcal{G}$ by LUA.
5: Update the j-th column of X^* by $X^*_{:,j}$.
6: **end for**
7: **until** The point $X^*_{:,j}$ is a KKT point of $f(X_{:,j})$ in $\mathcal{G}$ for all $j \in \{1, \ldots, k\}$.
8: **return** X^*

4.0.4 Finding the KKT points of $F(X)$

According to Theorem 1, we can find a KKT point of $F(X)$ by iteratively finding the KKT points of $f(X_{:,j})$ for all $j \in \{1, \ldots, k\}$. This can be done by using LUA to iteratively maximize $f(X_{:,j})$ with respect to each column $X_{:,j}$ of X. We summarize our approach in Algorithm 2.

The iterations of Algorithm 2 starts from an initialization X^0. Since $F(X)$ is a non-convex function and Algorithm 2 is finding a KKT point of $F(X)$, properly initializing X^0 often gives a better chance of getting a good KKT point X^* that has

Algorithm 3 Initializing X^0 for Algorithm 2

Input: The matrices A^+, A^-, P^+, and P^- of $\mathcal{G}$.
Output: The initial value of X^0 for Algorithm 2.

1: Set X^0 as an n-by-k matrix with all the entries to be zero.
2: Initialize a set $\eta = \emptyset$ to store selected vertex indices.
3: Compute $\mathbf{c}^+$ and select an index $h = \texttt{Select}(\mathbf{c}^+)$.
4: Update η by $\eta \leftarrow \eta \cup h$ and set $X^0_{h,1} = 1$.
5: **for** each $j \in \{2, \ldots, k\}$ **do**
6: Compute $\mathbf{c}^-$ and select an index $h = \texttt{Select}(\mathbf{c}^-)$.
7: Update η by $\eta \leftarrow \eta \cup h$ and set $X^0_{h,j} = 1$.
8: **end for**
9: **return** X^0 as the initialization for Algorithm 2.

a large value of $F(X^*)$. Therefore, we propose a heuristic method in Algorithm 3 to find a practically good initialization of X^0.

The key idea of Algorithm 3 is to randomly select k indices of seed vertices as the initializations for the k sets of vertex indices $S_1, \ldots, S_k$. In step 3, we first compute a vector $\mathbf{c}^+ = [c_1^+, \ldots, c_n^+]$, where the i-th entry $c_i^+ = \sum_{j=1}^n (A^+ \circ P^+)_{i,j}$ is the **expected positive edge degree** of the i-th vertex in $\mathcal{G}$. Then, we use the *fitness proportionate selection* (FPS) approach[1] to select the index of the first seed vertex by $h = \texttt{Select}(\mathbf{c}^+)$, which randomly selects an index h with a probability $\mathbb{P}(h) = \frac{c_i^+}{\sum_{i=1}^n c_i^+}$. The FPS approach is more likely to select the vertices with larger expected positive degrees because such vertices are heuristically more likely to be a member of a significant group of antagonistic communities.

In the vector $\mathbf{c}^- = [c_1^-, \ldots, c_n^-]$ of step 6, the i-th entry $c_i^- = \frac{1}{\eta} \sum_{j \in \eta} (A^- \circ P^-)_{i,j}$ is the **expected antagonism** between the vertex v_i and all the other vertices identified by the indices in η. The for loop in steps 5-8 uses FPS to select the other $(k-1)$ indices of the seed vertices under the criterion that the expected antagonism between

[1] `https://en.wikipedia.org/wiki/Fitness_proportionate_selection`

seed vertices should be large. Again, heuristically, such seed vertices are more likely to belong to different antagonistic communities, where the group of communities possesses a large inter-community antagonism.

Algorithm 3 provides a good initial point for Algorithm 2 to detect a KKT point X^* with a large value of $F(X^*)$. Such a KKT point often induces a significant group of k antagonistic communities.

Since $F(X)$ is a non-convex function, it is impractical to compute all the KKT points of $F(X)$. However, in real applications, we are often interested in only the significant KKT points of large values of $F(X)$. Similar to many previous works [30, 23, 13], we adopt the "peeling-off" method to find multiple KKT points of $F(X)$. Specifically, when finding a KKT point X^*, which induces a group of k antagonistic communities $\mathbb{G} = \{\mathcal{G}_{S_1}, \ldots, \mathcal{G}_{S_k}\}$, we record $\mathbb{G}$ in a candidate set and remove it from $\mathcal{G}$. Then, we call Algorithm 2 again to find another KKT point X^* in the remaining graph of $\mathcal{G}$. Such a process iterates until all vertices and edges in graph $\mathcal{G}$ are removed. Then, we search the candidate set to return the significant groups of k antagonistic communities with large values of $F(X)$ as our final answer.

Chapter 5

Experiments

In this section, we conduct extensive experiments to evaluate the performance of the proposed method named FACSUG. We first introduce the datasets, baseline methods, and evaluation measures in Subsections 5.0.1, 5.0.2, and 5.0.3, respectively. Then, we compare the performance of FACSUG with the baseline methods in Subsection 5.0.4 and conduct an extensive parameter analysis on FACSUG in Subsection 5.0.5. At last, we demonstrate the outstanding scalability of FACSUG on a large synthetic signed uncertain graph in Subsection 5.0.6 and show some interesting case studies on the discovered antagonistic communities in Subsection 5.0.7.

5.0.1 Datasets

We use the following datasets to conduct our experiments.

BioGRID dataset [33]. This is a public dataset provided by thebiogrid.org [1]. It records the interactions between genes in the form of synthetic genetic array (SGA) scores produced by biological experiments. For a pair of genes, multiple SGA scores

Table 5.1: Statistics of the real-world datasets.

Dataset Name	Type	# Vertices	# Edges
BioGRID [33]	Gene interaction network	5,293	306,253
WikiElec [22]	Social network	7,118	924,863
Slashdot [22]	Social network	82,144	50,0481
Epinions [22]	Social network	131,828	711,210
WordNet [27]	Network of adjectives	5,251	8,461

Table 5.2: Statistics of the synthetic dataset.

ID of the signed uncertain graphs	1	2	3	4	5
# Vertices ($\times 10^6$)	0.6	0.9	1.1	1.3	1.5
# Edges ($\times 10^6$)	4.0	8.0	12.1	15.9	20.0

are produced by using different alleles in biological experiments. Depending on the allele used in the biological experiments, the interaction between the same pair of genes can exhibit both positive and negative behaviours, which produce positive and negative SGA scores, respectively. Due to the complex nature of genetic relationships and the uncertainty of biological experimental outcomes, each SGA score is associated with a probability. Therefore, the BioGRID dataset can be naturally modelled as a signed uncertain graph, where each vertex represents a gene, the sign of each edge shows the positive and negative interactions between two genes, and the probability of each uncertain edge captures the inherent uncertain relationships between genes.

WikiElec dataset [22]. This is a public dataset provided by the Stanford network analysis project (SNAP) [2]. It is a signed static graph derived from the Wikipedia who-votes-on-whom network. By using a signed edge prediction method named SLF [41] to predict the probability of each edge in the signed static graph, we

obtain a signed uncertain graph where each vertex represents a user, the sign of each edge shows the friendship and antagonism between two users, and the probability of each uncertain edge shows the uncertainty of the relationships between two users.

Slashdot dataset [22]. This is a public dataset provided by SNAP [2]. It is a signed static graph capturing the friend/foe relationship between the users of the Slashdot website in February 2009. By using SLF[41] to predict the probability of each edge in the signed static graph, we obtain a signed uncertain graph, where each vertex represents a user of the Slashdot website, the sign of each edge shows the friendship and antagonism between two users, and the probability of each uncertain edge shows the uncertainty of the relationships between two users.

Epinions dataset [22]. This is a public dataset provided by SNAP [2]. It is a signed static graph derived from the Epinions who-trusts-whom network. By utilizing SLF[41] to predict the probability of each edge in the signed static graph, we obtain a signed uncertain graph, where each vertex represents a user of the Epinions website, the sign of each edge shows the trust and distrust relationship between two users, and the probability of each uncertain edge shows the uncertainty of the relationships between two users.

Synthetic dataset [11]. This dataset contains a set of five signed uncertain graphs used to evaluate the scalability of the proposed method in dealing with large graphs. We first use the graph generator proposed by Chiang et al. [11] to generate multiple signed static graphs with different sizes. Then, we use SLF [41] to generate the probabilities of each edge, which converts each of the signed static graphs into a signed uncertain graph. The size of each generated graph is reported in Table 5.2.

WordNet dataset [27]. This is a signed uncertain graph constructed from a

Table 5.3: The Properties of the Compared Methods. SUG means signed uncertain graph, and SSG means signed static graph.

Method	Works on	Partition	# Communities per group (k)	# Groups
FACSUG	SUG	No	≥ 2	≥ 1
KOCG [14]	SSG	No	≥ 2	≥ 1
EIGENSIGN [6]	SSG	No	$= 2$	$= 1$
SCG [38]	SSG	No	≥ 2	$= 1$
SPONGE [15]	SSG	Yes	≥ 2	$= 1$
MIL [26]	SUG	Yes	≥ 2	$= 1$

subset of the adjectives in the WordNet database [27]. Each vertex represents an adjective, the sign of each edge indicates the synonymous and antonymous relationship between two adjectives, and the probability of each uncertain edge is derived from the Wu-Palmer similarity score [5], which captures the semantic relatedness between a pair of adjectives. We use this dataset to conduct an interesting case study in Subsection 5.0.7.

5.0.2 Baseline Methods and Implementation Details

We compare the performance of the proposed method named FACSUG and five baseline methods, such as KOCG [14], EIGENSIGN [6], SCG [38], SPONGE [15], and MIL [26]. The properties of the baseline methods are listed in Table 5.3. A "Yes" in the column of "Partition" means the corresponding method partitions the entire graph into a single group of k antagonistic communities, and it requires every vertex to belong to a community; a "No" means the method does not partition the graph into a single group of antagonistic communities, and it allows a vertex not to be contained

in any group of antagonistic communities. The "# Communities per group" shows the number of communities contained in each group of antagonistic communities; we denote this by k when reporting the experimental results in the rest of this section. The "# Groups" shows how many groups of antagonistic communities can be found by a method.

Since the baseline methods KOCG, EIGENSIGN, SCG, and SPONGE are designed to work on signed static graphs (SSG), they cannot be directly applied to find antagonistic communities in a signed uncertain graph (SUG). We tackle this issue by converting the SUG into an SSG. The key idea is to convert each uncertain edge into a static edge by taking the edge weight with the maximum probability. For example, for an edge (v_i, v_j), if $P^+_{i,j} >= P^-_{i,j}$, then we convert it into a static edge with the positive edge weight $A^+_{i,j}$. Otherwise, if $P^+_{i,j} < P^-_{i,j}$, then we convert it into a static edge with the negative edge weight $A^-_{i,j}$. After converting the SUG into an SSG, we run each of the baseline methods on the SSG to find antagonistic communities.

We apply the KOCG, EIGENSIGN, SCG, and SPONGE methods on the converted SSG, while FACSUG and MIL are run on the original SUG. Subsequently, we compute the performance measures for the results of each method using the original SUG.

The source codes of the baseline methods are provided by the authors of the corresponding papers, and we use their default hyper-parameters in our experiments. The proposed FACSUG method is implemented in MATLAB, and we use $\alpha = 0.3$ and $\beta = 50$ by default. All experiments are conducted on a server with 64GB main memory, a 7200 RPM hard drive, and an Intel(R) Core(TM) i9-10900K CPU @3.70GHz.

5.0.3 Performance Measures

We measure the performance of each method by the significance of the group(s) of antagonistic communities discovered by the method. Let $\mathbb{G} = \{\mathcal{G}_{S_1}, \ldots, \mathcal{G}_{S_k}\}$ be a group of k antagonistic communities, where the j-th community $\mathcal{G}_{S_j}$ is induced by the set of vertex indices S_j, $j \in \{1, \ldots, k\}$. We measure the significance of $\mathbb{G}$ by the harmonic mean (HAM) of the overall friendship (OF) and the overall antagonism (OA) of $\mathbb{G}$.

Since some of the baseline methods, such as EIGENSIGN, SCG, SPONGE and MIL, do not use the matrix X to induce a group of antagonistic communities, we cannot evaluate the OF and OA of these methods by the $\texttt{OF}(X, G_h)$ in Equation (3.0.1) and the $\texttt{OA}(X, G_h)$ in Equation (3.0.3), respectively.

Considering that all the methods use a set of vertex indices S_j, $j \in \{1, \ldots, k\}$, to induce the j-th community $\mathcal{G}_{S_j}$ in the group of antagonistic communities $\mathbb{G}$, we use $\{S_1, \ldots, S_k\}$ to develop the measures of OF, OA, and HAM as follows.

First, we measure the **overall friendship (OF)** by

$$\text{OF} = \frac{1}{k} \sum_{j=1}^{k} \texttt{EF}(S_j), \tag{5.0.1}$$

where

$$\begin{aligned} \texttt{EF}(S_j) &= \mathbb{E}\Big(\frac{1}{|S_j|(|S_j|-1)} \sum_{h \in S_j} \sum_{l \in S_j, l \neq h} A^+_{h,l} \Big) \\ &= \frac{1}{|S_j|(|S_j|-1)} \sum_{h \in S_j} \sum_{l \in S_j, l \neq h} A^+_{h,l} P^+_{h,l}, \end{aligned} \tag{5.0.2}$$

is the expectation of the average positive edge weight for the signed uncertain subgraph $\mathcal{G}_{S_j}$ induced by S_j. The term inside the expectation operation in the first row

of Equation (5.0.2) is a widely used measure of intra-subgraph cohesion on static graphs [37]. Since a positive edge means a friendly relationship, $\texttt{EF}(S_j)$ measures the expected friendship in the community induced by S_j, and OF is the average of the expected friendship in the communities induced by $S_1, \ldots, S_k$, respectively.

Second, we measure the **overall antagonism (OA)** by

$$\mathrm{OA} = \frac{1}{k(k-1)} \sum_{j=1}^{k} \sum_{h=1}^{k} \texttt{EA}(S_j, S_h), \tag{5.0.3}$$

where

$$\begin{aligned} \texttt{EA}(S_j, S_h) &= \mathbb{E}\Big(\frac{1}{|S_j||S_h|} \sum_{r \in S_j} \sum_{l \in S_h} A^-_{r,l} \Big) \\ &= \frac{1}{|S_j||S_h|} \sum_{r \in S_j} \sum_{l \in S_h} A^-_{r,l} P^-_{r,l}, \end{aligned} \tag{5.0.4}$$

is the expectation of the average negative edge weight between the signed uncertain graphs $\mathcal{G}_{S_j}$ and $\mathcal{G}_{S_h}$. Since a negative edge means an antagonistic relationship, $\texttt{EA}(S_j, S_h)$ is the expected antagonism between the two communities induced by S_j and S_h, respectively. Therefore, OA measures the average of the expected antagonism between each pair of the communities induced by $S_1, \ldots, S_k$.

Finally, we measure the **harmonic mean (HAM)** of OF and OA by

$$\mathrm{HAM} = \frac{2 \times (\mathrm{OF} \cdot \mathrm{OA})}{\mathrm{OF} + \mathrm{OA}}. \tag{5.0.5}$$

A large HAM means a large OF and a large OA, which implies a high significance of the group of antagonistic communities $\mathbb{G}$. Thus, a larger HAM means a better performance.

5.0.4 The Performance of Compared Methods

In this subsection, we first show in Figure 5.1 the performance of each method with a fixed "# communities per group", denoted by k. Then, we show in Figure 5.2 the performance of each method with different values of k.

In Figure 5.1, we use $k = 3$ for FACSUG, KOCG, SCG and SPONGE. Since EIGENSIGN cannot find a group of more than 2 communities, we use its default parameter $k = 2$. MIL is designed to automatically compute the value of k, and it does not allow manually setting the value of k. Thus we use its computed value of k, which is shown in the legend of each subfigure in Figure 5.1.

As shown in Figure 5.1, since EIGENSIGN, SCG, SPONGE, and MIL can only find a single group of antagonistic communities, each of their performance is shown as a single point. Both FACSUG and KOCG can find multiple groups of antagonistic communities. Thus their performances are shown as curves, where a point on each curve shows the index of a group of antagonistic communities and its corresponding performance. To ease the difficulty of viewing the results, we only report the groups with HAM no smaller than 0.01 and we sort the index of the groups in descending orders of HAM, OF, and OA correspondingly.

We can see from Figure 5.1 that the leftmost points on the curves of FACSUG and KOCG are much higher than the single points of the other baseline methods. This means the top-1 group of communities found by each of FACSUG and KOCG has a much higher significance than the group of communities discovered by each of the other baseline methods.

The inferior performance of SPONGE and MIL is largely due to their strategy of partitioning the entire graph into a single group of antagonistic communities. This

requires every vertex to belong to a community. Thus the discovered communities are usually large in size and contain many neutral vertices that do not have a strong positive or negative edge connection with the other vertices. This leads to sparse positive edge connections within each community and sparse negative edge connections between different communities. Therefore, SPONGE and MIL cannot achieve a high performance of HAM, OF, or OA.

The methods SCG and EIGENSIGN do not require every vertex to belong to a community, which excludes the neutral vertices from the discovered group of antagonistic communities. However, they measure the overall friendship based on the sum of positive edge weights inside each community and measure the overall antagonism based on the sum of negative edge weights between different communities. This often leads to communities that are large in size but not dense in edge connections. Thus SCG and EIGENSIGN cannot produce a high performance of HAM, OF, or OA. In comparison, both FACSUG and KOCG achieve a much better performance because they derive the overall friendship and overall antagonism based on the notion of dense subgraph [30, 23, 7, 13], which tends to find smaller communities with denser positive connections within each community and denser negative connection between different communities.

We can also see from Figure 5.1 that the performance of FACSUG is almost always better than KOCG. This is because KOCG is designed to work on static signed graphs, thus, it cannot effectively find highly significant groups of antagonistic communities in a signed uncertain graph. We also notice that in Figures 5.1(a) and 5.1(e), the HAM and OF of FACSUG are slightly lower than that of KOCG on BioGRID when the index of groups is larger than 60. This is because the top 60 groups found by FACSUG

cover more vertices to form more significant group(s) of communities than KOCG. When these vertices are removed by the "peeling-off" method in Subsection 4.0.4, there are fewer remaining vertices to form highly significant groups of communities.

Next, we discuss the performance of each method with different values of k in Figure 5.2.

Since each of EIGENSIGN, SCG, SPONGE, and MIL finds a single group of antagonistic communities, but FACSUG and KOCG find many groups of communities, we report the HAM, OF, and OA of the top-1 group of communities with the largest HAM as the performance of FACSUG and KOCG. Since EIGENSIGN and MIL do not allow manually setting the value of k, we report the performance of EIGENSIGN only when $k = 2$ and the performance of MIL only at its computed value of k. For SCG, SPONGE, KOCG and FACSUG, we use the values of k including $\{2, 3, 5, 7, 10, 50\}$ and the k computed by MIL.

As shown in Figure 5.2, FACSUG almost always achieves a better HAM than the baseline methods when $k \in \{2, 3, 5, 7, 10, 50\}$. The reasons for the inferior performance of the baseline methods are mentioned previously, thus, we skip them here to avoid redundancy. For all the methods, the performance when $k = 50$ is lower than when $k \in \{2, 3, 5, 7, 10\}$, because it is unlikely for any of the datasets to contain a significant group of 50 antagonistic communities. Due to the same reason, we can see that the performances of all the methods are almost zero at the large value of k computed by MIL.

5.0.5 Parameter Analysis

In this subsection, we focus on analyzing the effect of the parameter α on the performance of FACSUG. We skip the analysis on the parameter β in Equation (3.0.7) because it is set to a trivially large value (i.e., $\beta = 50$) to prevent overlap between communities, and the specific value of β does not have a significant effect on the performance of FACSUG.

Figure 5.3 shows the effect of α on the Average HAM, Average OF, and Average OA of FACSUG. We can see from Figures 5.3(i)-5.3(l) that the Average OA increases when α increases. This is because α is the weight of the term $\texttt{OA}(X, G_h)$ in Equation (3.0.5), thus increasing α encourages FACSUG to find the groups of communities with higher overall antagonism. Meanwhile, increasing α puts more weight on the term $\texttt{OA}(X, G_h)$, which reduces the relative importance of the term $\texttt{OF}(X, G_h)$ in Equation (3.0.5). Therefore, we can observe from Figures 5.3(e)-5.3(h) that the Average OF drops when α increases.

The Average HAM in Figures 5.3(a)-5.3(d) show different trends on different datasets due to the different distributions of positive edge weights and negative edge weights. The datasets of BioGRID and WikiElect contain fewer negative edges than Slashdot and Epinions, thus increasing α brings more drop of OF than the increase of OA, which reduces the Average HAM on BioGRID and WikiElect. For Slashdot and Epinions, increasing α from 0.2 to 0.6 increases the Average HAM because it brings more increase in OA than the drop of OF. However, the gain of HAM diminishes when increasing α to be overly large.

In summary, the parameter α controls the tradeoff between the overall friendship and the overall antagonism of the groups of communities.

5.0.6 The Scalability of FACSUG

In this subsection, we compare the scalability of FACSUG and the proximal gradient descend (PGD) method [29] in solving the optimization problem in Equation (3.0.7) on the synthetic dataset. Both methods are implemented in MATLAB and run in the same software and hardware environment. We use the same initialization point X^0 for both methods.

Figure 5.4(a) shows the objective value $F(X^*)$ of the KKT point X^* detected by each of FACSUG and PGD. The objective values are close, which shows FACSUG and PGD are finding comparably good solutions. Figure 5.4(b) shows the running time of FACSUG and PGD. We can see that FACSUG is about three orders of magnitudes faster than PGD because PGD is a generic solution that is not well customized to solve the optimization problem in Equation (3.0.7). Specifically, PGD calculates the gradient of $F(X)$ in each iteration, which is computationally expensive when the number of edges is large. In comparison, due to the well-designed LUA algorithm in Algorithm 1, all the iterations of FACSUG are performed on small local graphs, which leads to its fast speed in processing large graphs.

5.0.7 Case Studies

In this subsection, we present two interesting case studies on BioGRID and WordNet, respectively, to show the groups of antagonistic communities discovered by FACSUG.

Case study 1. This case study explores the application of FACSUG on the gene interaction network of BioGRID, where each vertex is a gene, and each signed uncertain edge captures the uncertain positive or negative relationship between genes. Running FACSUG in this network finds groups of antagonistic communities, where

each community contains a set of positively related genes and the genes from different communities in the same group are negatively related to each other. Finding such groups of antagonistic gene communities can provide meaningful insights into the mutually inhibitory relationship between different sets of functionally similar genes.

Take Figure 5.5 as an example, we can see four groups of antagonistic gene communities, where each vertex shows the name of a gene and the vertices with the same shape (i.e., circle and square) belong to the same community. On each edge, we show the edge weights by a tuple $w = (w_1, w_2)$ and the probability of weights by a tuple $p = (p_1, p_2)$. The elements w_1 and w_2 show the weights of positive and negative relationships, respectively. The elements p_1 and p_2 show the probabilities for the positive and negative relationships, respectively. The sum of the probabilities p_1 and p_2 are smaller than 1 because $1 - p_1 - p_2$ is the probability of not having an edge between two vertices. The colour of an edge is determined by the sign of the expected edge weight, denoted by $\texttt{eew} = p_1 * w_1 + p_2 * w_2$. If $\texttt{eew} >= 0$, we draw the edge in green to represent a positive expected relationship between two genes; otherwise, we draw the edge in red to represent a negative expected relationship. As shown in Figure 5.5, the groups of gene communities show strong positive relationships within each community and strong negative relationships between different communities. This demonstrates the effectiveness of FACSUG in discovering significant groups of antagonistic gene communities.

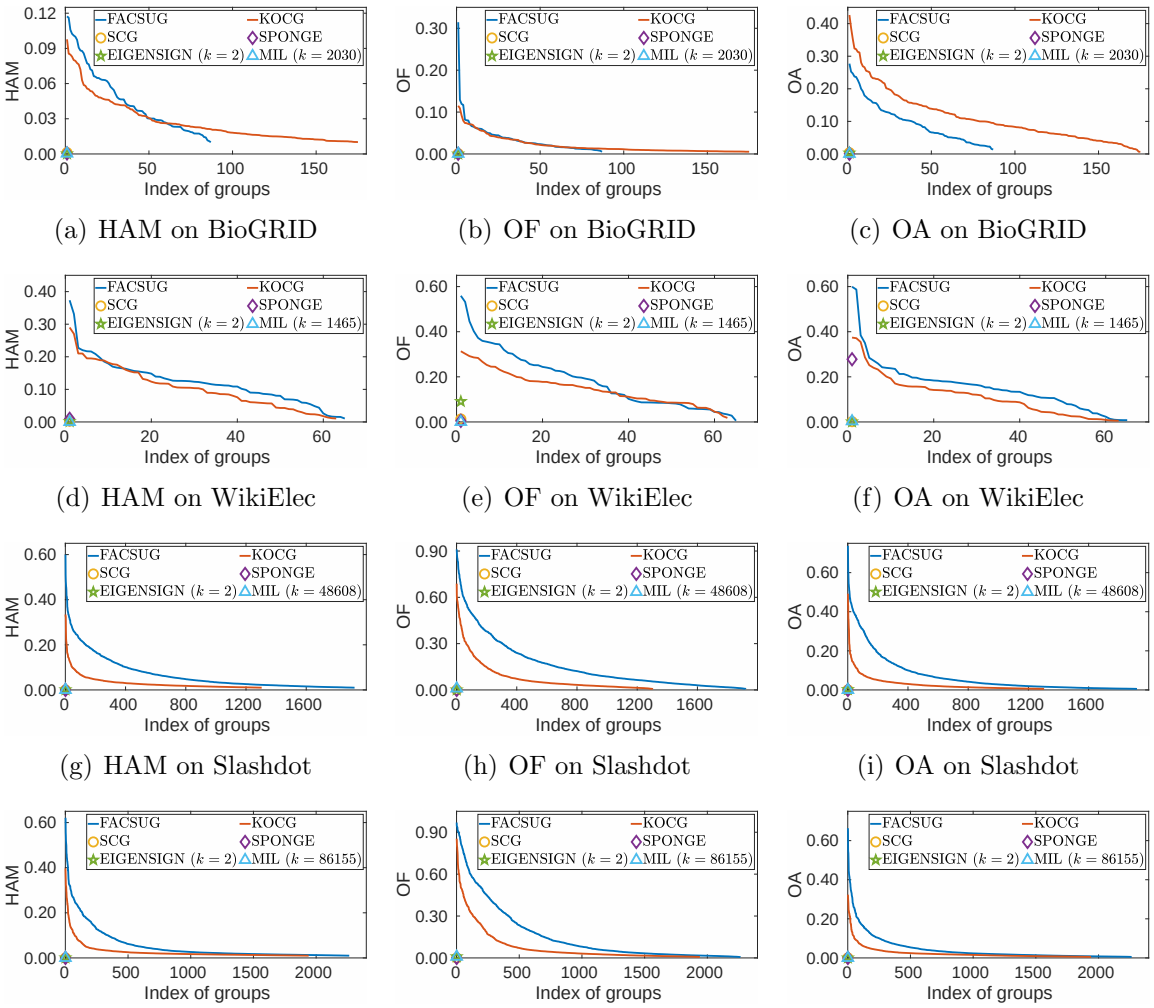
The application of FACSUG is beyond gene interaction networks. Next, we introduce another interesting case study that runs FACSUG to find two sets of synonyms with opposite meanings in a network of adjectives.

Case study 2. This case study explores the application of FACSUG on the

Table 5.4: The groups of antagonistic adjective communities discovered in the adjective network of WordNet when $k = 2$ and $\alpha = 0.3$.

ID	Community 1	Community 2
1	impermissible, inadmissible	permissible, admissible
2	impolite, discourteous	polite, courteous
3	tractable, docile	intractable, stubborn
4	unconventional, unorthodox	conventional, orthodox
5	conquerable, surmountable	unconquerable, insurmountable
6	pluperfect, perfect	imperfect, imperfectible
7	unconcerned, uninvolved	concerned, involved
8	unimportant, insignificant, superficial	profound, significant
9	unaccustomed, unusual	accustomed, usual
10	correct, accurate	inaccurate, incorrect

adjective network of WordNet, where each vertex is an adjective, and the positive and negative weights of a signed uncertain edge show the synonymous and antonymous relationships, respectively, between two adjectives. Running FACSUG in this network finds groups of antagonistic adjective communities, where each community contains a set of synonyms and the adjectives from different communities in the same group have opposite meanings. Table 5.4 shows the top 10 groups of antagonistic adjective communities discovered by FACSUG. Each row shows one group of two adjective communities with opposite meanings. These results further verify that finding groups of antagonistic communities reveals interesting patterns in the signed uncertain graph of adjectives in the WordNet dataset.



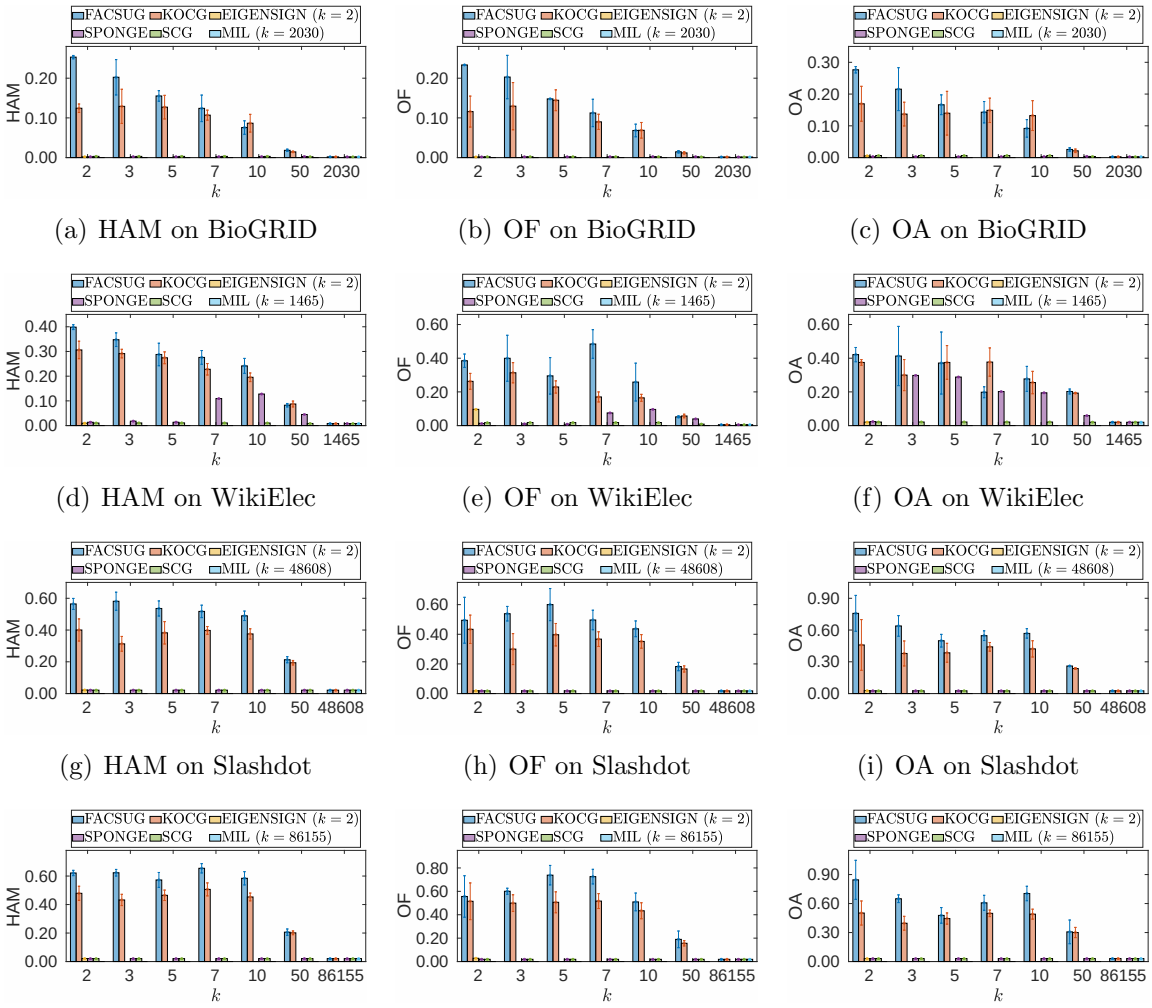
(a) HAM on BioGRID (b) OF on BioGRID (c) OA on BioGRID

(d) HAM on WikiElec (e) OF on WikiElec (f) OA on WikiElec

(g) HAM on Slashdot (h) OF on Slashdot (i) OA on Slashdot

(j) HAM on Epinions (k) OF on Epinions (l) OA on Epinions

Figure 5.1: The HAM, OF, and OA of the group(s) of antagonistic communities discovered by each method.



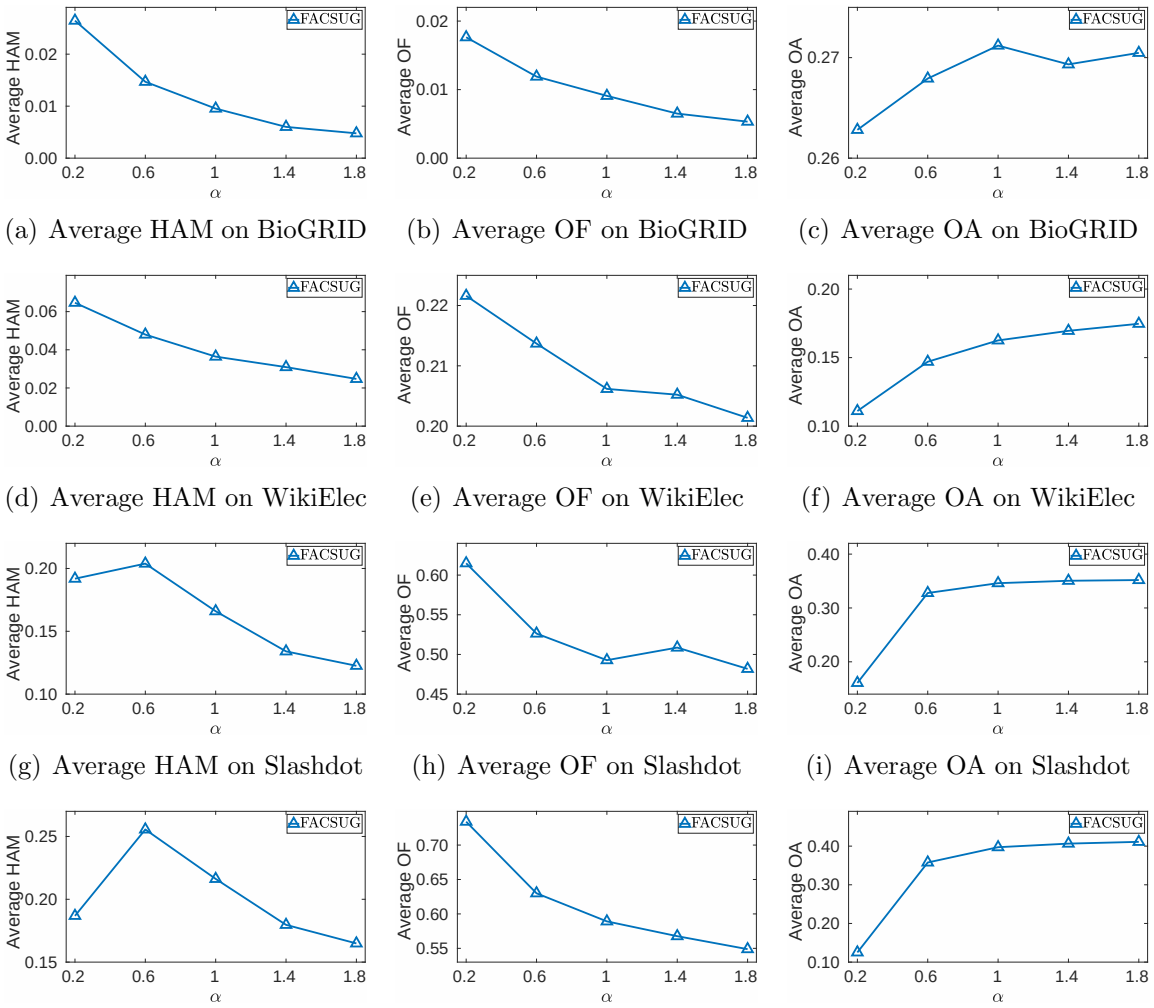
(a) HAM on BioGRID (b) OF on BioGRID (c) OA on BioGRID

(d) HAM on WikiElec (e) OF on WikiElec (f) OA on WikiElec

(g) HAM on Slashdot (h) OF on Slashdot (i) OA on Slashdot

(j) HAM on Epinions (k) OF on Epinions (l) OA on Epinions

Figure 5.2: The HAM, OF, and OA of the top-1 group of antagonistic communities with the largest HAM for different values of k. The error bar shows the standard deviation of 5 independent runs.



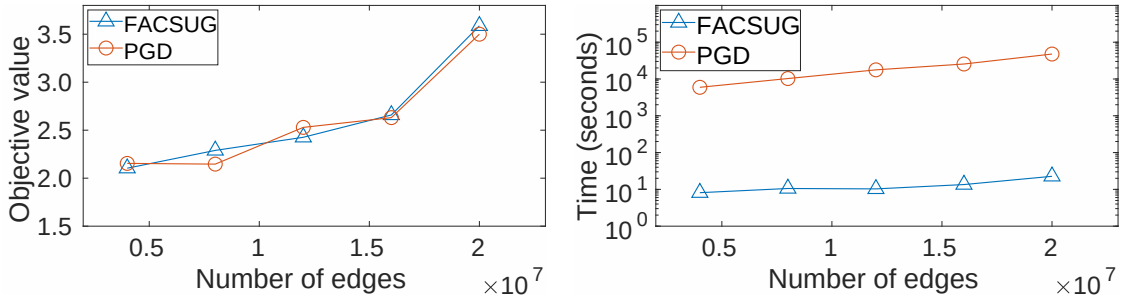
(a) Average HAM on BioGRID (b) Average OF on BioGRID (c) Average OA on BioGRID

(d) Average HAM on WikiElec (e) Average OF on WikiElec (f) Average OA on WikiElec

(g) Average HAM on Slashdot (h) Average OF on Slashdot (i) Average OA on Slashdot

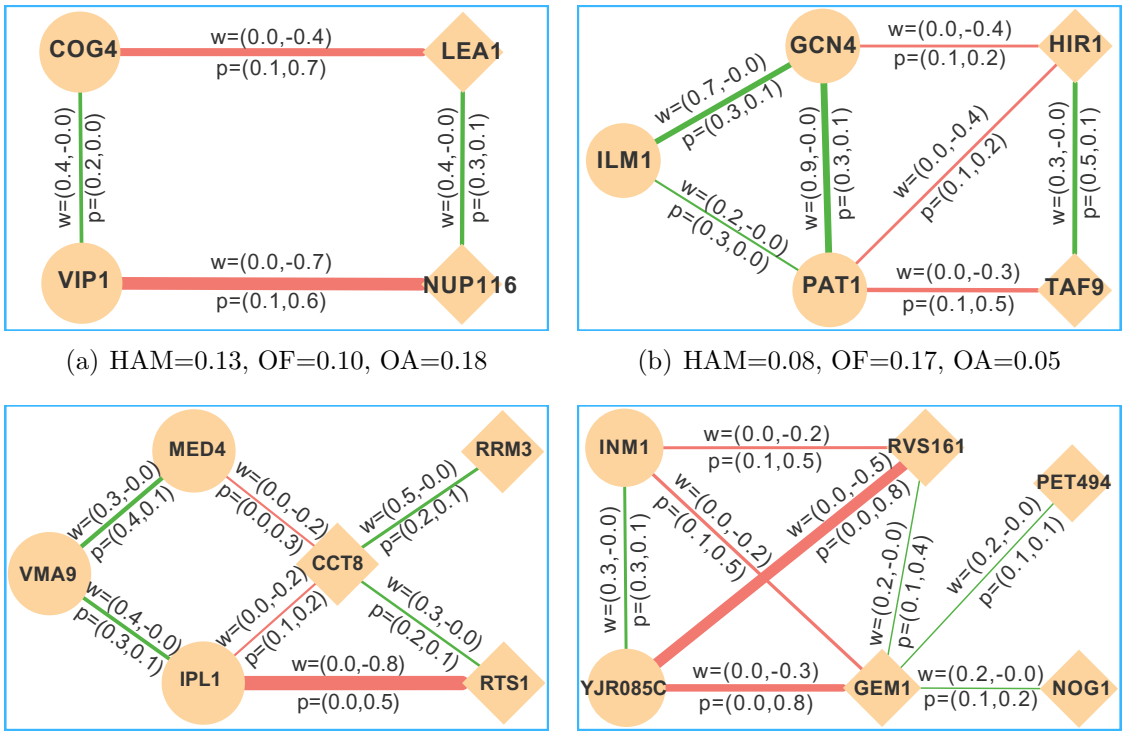
(j) Average HAM on Epinions (k) Average OF on Epinions (l) Average OA on Epinions

Figure 5.3: The performance of FACSUG when $\alpha \in \{0.2, 0.6, 1.0, 1.4, 1.8\}$, $\beta = 50$, and $k = 10$. The Average HAM, Average OF, and Average OA are the mean values of HAM, OF, and OA, respectively, computed from the groups of communities with top-50 largest HAM, OF, and OA, correspondingly.



(a) Objective value (b) Running time

Figure 5.4: Scalability analysis on the synthetic dataset.



(a) HAM=0.13, OF=0.10, OA=0.18 (b) HAM=0.08, OF=0.17, OA=0.05

(c) HAM=0.06, OF=0.07, OA=0.06 (d) HAM=0.07, OF=0.05, OA=0.11

Figure 5.5: The groups of antagonistic gene communities discovered in the gene interaction network of BioGRID when $k = 2$ and $\alpha = 0.1$.

Chapter 6

Conclusion

In this paper, we tackled the novel problem of finding groups of antagonistic communities in signed uncertain graphs. We formulate the problem as a constrained quadratic programming problem and solve it with an efficient algorithm named FACSUG. Extensive experiments on five real-world datasets and one synthetic dataset show the outstanding performance and efficiency of FACSUG in finding significant groups of antagonistic communities. In future work, we will attempt to extend FACSUG to find groups of antagonistic communities in dynamic uncertain graphs where the weight distributions of uncertain edges change over time.

Bibliography

[1] Biogird. URL `https://thebiogrid.org/`.

[2] Snap. URL `https://snap.stanford.edu/data/`.

[3] L. A. Adamic and N. Glance. The political blogosphere and the 2004 us election: Divided they blog. *Proceedings of the International Workshop on Link Discovery*, pages 36–43, 2005.

[4] V. Arnaboldi, M. Conti, A. Passarella, and R. Dunbar. Dynamics of personal social relationships in online social networks: A study on twitter. *Proceedings of the ACM Conference on Online Social Networks*, pages 15–26, 2013.

[5] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc., 2009.

[6] F. Bonchi, E. Galimberti, A. Gionis, B. Ordozgoiti, and G. Ruffo. Discovering polarized communities in signed networks. *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 961–970, 2019.

[7] S. R. Bulo, M. Pelillo, and I. M. Bomze. Graph-based quadratic optimization: A fast evolutionary approach. *Computer Vision and Image Understanding*, pages 984–995, 2011.

[8] M. Ceccarello, C. Fantozzi, A. Pietracaprina, G. Pucci, and F. Vandin. Clustering uncertain graphs. *Proceedings of the VLDB Endowment*, pages 472–484, 2017.

[9] C. Chen, Y. Wu, R. Sun, and X. Wang. Maximum signed θ-clique identification in large signed graphs. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[10] Z. Chen, L. Yuan, X. Lin, L. Qin, and J. Yang. Efficient maximal balanced clique enumeration in signed networks. *Proceedings of The Web Conference*, pages 339–349, 2020.

[11] K.-Y. Chiang, J. J. Whang, and I. S. Dhillon. Scalable clustering of signed networks using balance normalized cut. *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 615–624, 2012.

[12] L. Christiansen, B. Mobasher, and R. Burke. Using uncertain graphs to automatically generate event flows from news stories. *HT (Extended Proceedings)*, 2017.

[13] L. Chu, S. Wang, S. Liu, Q. Huang, and J. Pei. ALID: scalable dominant cluster detection. *Proc. VLDB Endow.*, pages 826–837, 2015.

[14] L. Chu, Z. Wang, J. Pei, J. Wang, Z. Zhao, and E. Chen. Finding gangs in war from signed networks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1505–1514, 2016.

[15] M. Cucuringu, P. Davies, A. Glielmo, and H. Tyagi. Sponge: A generalized eigenproblem for clustering signed networks. *The International Conference on Artificial Intelligence and Statistics*, pages 1088–1098, 2019.

[16] N. Dalvi and D. Suciu. Management of probabilistic data: Foundations and challenges. *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 1–12, 2007.

[17] V. Fionda. Networks in biology. *Encyclopedia of Bioinformatics and Computational Biology*, pages 915–921, 2019.

[18] N. Girdhar and K. Bharadwaj. Signed social networks: a survey. *Advances in Computing and Data Sciences: First International Conference*, pages 326–335, 2017.

[19] K. Han, F. Gui, X. Xiao, J. Tang, Y. He, Z. Cao, and H. Huang. Efficient and effective algorithms for clustering uncertain graphs. *Proceedings of the VLDB Endowment*, pages 667–680, 2019.

[20] E. L. Huttlin, M. P. Jedrychowski, J. E. Elias, T. Goswami, R. Rad, S. A. Beausoleil, J. Villén, W. Haas, M. E. Sowa, and S. P. Gygi. A tissue-specific atlas of mouse protein phosphorylation and expression. *Cell*, pages 1174–1189, 2010.

[21] H. W. Kuhn and A. W. Tucker. Nonlinear programming. *Traces and Emergence of Nonlinear Programming*, pages 247–258, 2013.

[22] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1361–1370, 2010.

[23] H. Liu and S. Yan. Robust graph mode seeking by graph shift. *Proceedings of*

the International conference on Machine Learning (ICML-10), pages 671–678, 2010.

[24] L. Liu, R. Jin, C. Aggarwal, and Y. Shen. Reliable clustering on uncertain graphs. *International Conference on Data Mining*, pages 459–468, 2012.

[25] D. J. MacKay. An example inference task: Clustering. *Information Theory, Inference and Learning Algorithms*, pages 284–292, 2003.

[26] D. Mandaglio, A. Tagarelli, and F. Gullo. In and out: Optimizing overall interaction in probabilistic graphs under clustering constraints. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1371–1381, 2020.

[27] G. A. Miller. Wordnet: A lexical database for English. *Communications of the ACM*, pages 39–41, 1995.

[28] J. Niu and A. E. Sarıyüce. On cohesively polarized communities in signed networks. *Companion Proceedings of the ACM Web Conference*, pages 1339–1347, 2023.

[29] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends® in Optimization*, pages 127–239, 2014.

[30] M. Pavan and M. Pelillo. Dominant sets and pairwise clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 167–172, 2006.

[31] W. Peng, J. Wang, J. Cai, L. Chen, M. Li, and F.-X. Wu. Improving protein function prediction using domain and protein complexes in PPI networks. *BMC Systems Biology*, pages 1–13, 2014.

[32] V. Sanz-Moreno, G. Gadea, J. Ahn, H. Paterson, P. Marra, S. Pinner, E. Sahai, and C. J. Marshall. Rac activation and inactivation control plasticity of tumor cell movement. *Cell*, pages 510–523, 2008.

[33] C. Stark, B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers. Biogrid: A general repository for interaction datasets. *Nucleic Acids Research*, pages D535–D539, 2006.

[34] D. Suciu and N. N. Dalvi. Foundations of probabilistic answers to queries. *SIGMOD Conference*, 2005.

[35] L. Tang and H. Liu. *Community detection and mining in social media*. Morgan & Claypool Publishers, 2010.

[36] Y. Tian, J. Yan, J. Hu, and Z. Wu. A privacy preserving model in uncertain graph mining. *International Conference on Networking and Network Applications*, pages 102–106, 2018.

[37] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 104–112, 2013.

[38] R.-C. Tzeng, B. Ordozgoiti, and A. Gionis. Discovering conflicting groups in signed networks. *Advances in Neural Information Processing Systems*, pages 10974–10985, 2020.

[39] J. W. Weibull. *Evolutionary Game Theory*. MIT press, 1997.

[40] H. Xiao, B. Ordozgoiti, and A. Gionis. Searching for polarization in signed graphs: A local spectral approach. *Proceedings of The Web Conference*, pages 362–372, 2020.

[41] P. Xu, W. Hu, J. Wu, and B. Du. Link prediction with signed latent factors in signed social networks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1046–1054, 2019.

[42] S.-H. Yang, A. J. Smola, B. Long, H. Zha, and Y. Chang. Friend or frenemy? Predicting signed ties in social networks. *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 555–564, 2012.

[43] D. Yu, D. Wang, Q. Luo, Y. Zheng, G. Wang, and Z. Cai. Stable structural clustering in uncertain graphs. *Information Sciences*, pages 596–610, 2022.