Balance Maximization in Signed Networks via Edge Deletions

Kartik Sharma  
IIT Delhi  
Kartik.Sharma.cs117@cse.iitd.ac.in

Iqra Altaf Gillani  
IIT Delhi  
iqraaltf@cse.iitd.ac.in

Sourav Medya  
Northwestern University  
sourav.medya@kellogg.northwestern.edu

Sayan Ranu  
IIT Delhi  
sayanranu@cse.iitd.ac.in

Amitabha Bagchi  
IIT Delhi  
bagchi@cse.iitd.ac.in

ABSTRACT

In signed networks, each edge is labeled as either positive or negative. The edge sign captures the polarity of a relationship. Balance of signed networks is a well-studied property in graph theory. In a balanced (sub)graph, the vertices can be partitioned into two subsets with negative edges present only across the partitions. Balanced portions of a graph have been shown to increase coherence among its members and lead to better performance. While existing works have focused primarily on finding the largest balanced subgraph inside a graph, we study the network design problem of maximizing balance of a target community (subgraph). In particular, given a budget \( b \) and a community of interest within the signed network, we aim to make the community as close to being balanced as possible by deleting up to \( b \) edges. Besides establishing NP-hardness, we also show that the problem is non-monotone and non-submodular. To overcome these computational challenges, we propose heuristics based on the spectral relation of balance with the Laplacian spectrum of the network. Since the spectral approach lacks approximation guarantees, we further design a greedy algorithm, and its randomized version, with provable bounds on the approximation quality. The bounds are derived by exploiting pseudo-submodularity of the balance maximization function. Empirical evaluation on eight real-world signed networks establishes that the proposed algorithms are effective, efficient, and scalable to graphs with millions of edges.

1 INTRODUCTION AND RELATED WORK

Graphs can model various complex systems such as knowledge graphs [33], road networks [27], communication networks [29], and social networks [19]. Typically, nodes represent entities, and edges characterize relationships between pairs of entities. Signed graphs further enhance the representative power of graphs by capturing the polarity of a relationship through positive and negative edge labels [15, 17, 31]. For example, if a graph represents social interactions, a positive edge would denote friendly interaction, and a negative edge would indicate a hostile relationship. Similarly, in a collaboration network, positive edges may indicate complementary skill sets, whereas negative edges would indicate disparate skills.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WSDM ’21, March 08–12, 2021, Jerusalem, Israel
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06...
https://doi.org/10.1145/3428699.3430391

Signed graphs were first studied by Harary et al. [15] with particular focus on their balance. A balanced signed graph is one in which the vertices can be partitioned into two sets such that all edges inside each partition have a positive sign and all the negative signed edges are across the partitions. Balance is correlated with both positive and negative side-effects on a community. On the positive side, balanced communities are positively correlated with performance in financial networks where edges represent trading links [3, 12]. On the negative side, in social networks, balanced communities often promote “echo-chambers”, reduce diversity of opinions, and ultimately lead to more polarized viewpoints [13].

Owing to the correlation of balance with several higher-order functional traits, it is natural to measure how far a community is from being balanced. For example, in financial networks, it is important to evaluate how the community may be engineered to further improve its balance. On the other hand, in social networks, an adversary, such as a political party, may be interested in polarizing the community in its favor by further increasing its balance. To avoid such adversarial attacks, it is important to know the weak links in a community so that they can be safeguarded.

In this paper, we address these applications by studying the problem of maximizing balance via edge deletions (MBED). In the MBED problem, we are given a graph, a target community within this graph, and a budget \( b \). Our goal is to remove \( b \) edges, such that the community gets as close to being balanced as possible. We formally define the notion of balance closeness in § 2. Deleting an edge would correspond to actions such as unfollowing or blocking a connection. If increasing balance is desirable, then MMBED provides a mechanism towards achieving the goal. On the other hand, MMBED also measures how susceptible a community is to adversarial attacks by revealing how much the balance can be increased through a small number of deletions, and which are these critical edges that must be protected.

1.1 Related Work

The problem we study falls in the class of network design problems. In network design, the goal is to modify the network so that an objective function modeling a desirable property is optimized. Examples of such objective functions include optimizing shortest path distances (traffic and sustainability improvement) [11, 23, 27, 28], increasing centrality of target nodes by adding a small set of edges [7, 18, 26], optimizing the k-core [24, 39], manipulating node similarities [10], and boosting/containing influence on social networks [5, 20, 25].

While several works exist on finding balanced subgraphs [9, 12, 15, 17, 31], work on optimizing balance through network design is rather limited. The only work is by Akiyama et al. [1], where they study the minimum number of sign flips needed to make a graph...
balanced. However our work is different for several reasons. First, [1] does not have any notion of a budget constraint. Second, the cascading impact of a sign flip and an edge deletion on the balance of a graph is significantly different. Third, [1] lacks evaluation on large real world graphs containing millions of edges. Finally, from a practicality viewpoint, selectively flipping the sign of an edge is difficult since the edge sign encodes the nature of interaction between the two entities (endpoints) of the edge. In contrast, deleting an edge is a more lightweight task as it only involves stopping further interactions with a chosen node.

Several studies related to identifying large balanced subgraphs exist. Poljak and Turzík addressed the problem of finding a maximum weight balanced subgraph and showed an equivalence with max-cut in a graph with a general weight function [35]. Other approaches include finding balanced subgraphs with the maximum number of vertices [12, 31] and edges [9] in the context of biological networks. Hüffner et al. [17] gave an exact algorithm for finding maximum weight balanced subgraph and showed an equivalence with any induced balanced subgraph that is connected and balanced. The largest connected induced balanced subgraph is denoted by $S(\Gamma)$, and thus, $\Delta(\Gamma) = |V(S(\Gamma))|$. It is worth noting that the largest connected induced balanced subgraph might not be unique.

We solve a network design problem where the balance is maximized via edge deletions. The modified graph is denoted as $\Gamma_X$ after the deletion operation of edge set $X$ on $\Gamma$. Deletion of an edge (positive or negative) may increase the balance of a graph.

**Example 2.** The current balance of the graph in Fig. 1(a) is 6. Deleting any negative or positive edge increases the balance to 8 (Fig. 1(b)-(d)). Note that deleting an edge may initiate a cascading impact and bring in multiple nodes into the balanced subgraph.

**Problem 1 (Maximizing Balance via Edge Deletion (Mbed)).** Given a signed (sub) graph $\Gamma$, the current balance $\Delta(\Gamma)$ is the maximum number of nodes in any induced subgraph that is connected and balanced. The largest connected induced balanced subgraph is denoted by $S(\Gamma)$, and thus, $\Delta(\Gamma) = |V(S(\Gamma))|$.

We envision $H$ to be the target community where we would like to maximize balance. $\mathcal{C}$ denotes the edges that may be deleted, which may be the entire edge set of $H$.

**2 Problem Characterization**

**Theorem 1.** The Mbed problem is NP-hard.

**Lemma 1.** The optimization function $f(B)$ of Mbed is non-monotonic, i.e., an edge deletion may lead to a decrease in current balance.

**Proof.** Consider the path $a-b-c-d$ with only edge $(b, c)$ being negative. The current balance is 4 since the entire graph is balanced. If we delete any edge, the balance decreases to at most 3.

A function $f(\cdot)$ is submodular [19] if the marginal gain by adding an element $e$ to a subset $S$ is equal or higher than the same in a superset $T$. Mathematically, it satisfies:

$$f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$$

for all elements $e$ and all pairs of sets $S \subseteq T$ and $e \notin S, e \notin T$.

**Lemma 2.** $f(B)$ is not submodular.

1We show a stronger result that it is not even proportionally-submodular in Sec. 8.2.
Balance Maximization in Signed Networks via Edge Deletions WSDM ’21, March 08–12, 2021, Jerusalem, Israel

\[ \Gamma = ((V, E), \sigma) \]
\[ S(\Gamma) \]
\[ \Delta(\Gamma) [V(\Gamma)] \]
\[ C \]
\[ b \]
\[ L(\Gamma) \]
\[ \lambda_1(\Gamma) \]
\[ \mathbf{u}, \mathbf{v} \]
\[ \mathcal{H}_X \]
\[ \text{ce}(H, x) \]

Table 1: Frequently used symbols

**3 THE SPECTRAL APPROACH**

Given a signed graph \( \Gamma = ((V, E), \sigma) \), let \( A \) be its adjacency matrix where \( A_{ij} = \sigma(i, j) \) for \( (i, j) \in E \) and 0 otherwise. Furthermore, let \( D \) be the diagonal degree matrix defined as \( D_{ii} = d(i) \), where \( d(i) \) is the vertex degree, i.e., the total number of edges incident on vertex \( i \). We define the corresponding signed Laplacian as follows.

**Definition 4 (Signed Laplacian).** The Laplacian of a signed graph \( \Gamma = ((V, E), \sigma) \), denoted as \( L(\Gamma) \), is the symmetric matrix \( [V(\Gamma)] \times [V(\Gamma)] \) matrix defined as \( L(\Gamma) = D(\Gamma) - A(\Gamma) \), i.e., \( L_{ii} = d_i \) and \( L_{ij} = -\sigma(i, j) \) if \( (i, j) \in E \) and 0 otherwise for \( i \neq j \).

**Lemma 3 (\cite{16}).** Given a signed graph \( \Gamma = ((V, E), \sigma) \), \( \Gamma \) is balanced iff the smallest eigenvalue of the Laplacian \( \lambda_1(\Gamma) = 0 \).

It has been further shown that \( \lambda_1(\Gamma) \) is a measure of how “far” the graph is from being balanced \( \cite{4, 22} \).

**Lemma 4 (\cite{4}).** Given a signed graph \( \Gamma = ((V, E), \sigma) \) with \( \lambda_1(\Gamma) \) as the smallest eigenvalue of the corresponding Laplacian,

\[ \lambda_1(\Gamma) \leq \nu(\Gamma) \leq \epsilon(\Gamma) \]

where \( \nu(\Gamma) (\epsilon(\Gamma)) \) denotes the frustration number (frustration index), i.e., the minimum number of vertices (edges) to be deleted such that the signed graph is balanced.

Note that \( \Delta(\Gamma) = |V| - \nu(\Gamma) \). Through Lemma 4, for any given subgraph \( H \), we have:

\[ \Delta(H) = |V(H)| - \nu(H) \leq |V(H)| - \lambda_1(H) \]

**3.1 An Upperbound Based Algorithm**

Since directly maximizing \( \Delta(H) \) is NP-hard, we turn our focus to the upperbound provided by Eq. (2). It is evident that maximizing the upper bound is equivalent to minimizing \( \lambda_1(H) \). To minimize \( \lambda_1(H) \), we first derive the following upper bound.

**Lemma 5.** Given a signed graph \( \Gamma \), a subgraph \( H \), a candidate edge set \( C \), for a set \( X \subseteq C \), we have

\[ \lambda_1(H_X) \leq \lambda_1(H) - \sum_{(i, j) \in X} (v_i - \sigma(i, j)v_j)^2 \]

where \( \mathbf{v} \) denotes the unit eigenvector of Laplacian \( L(H) \) corresponding to the minimum eigenvalue \( \lambda_1(H) \) and \( v_i \) denotes the \( i \)-th entry of \( \mathbf{v} \). Recall, \( H_X \) is the subgraph formed due to removal of edge set \( X \) from \( H \).

**Proof.** Given a signed graph \( \Gamma \) with \( L(\Gamma) \) being its corresponding Laplacian. We know for any \( \mathbf{u} \in \mathbb{R}^{|V|} \),

\[ \mathbf{u}^T L(\Gamma) \mathbf{u} = \sum_{(i, j) \in E^+} (u_i - u_j)^2 + \sum_{(i, j) \in E^-} (u_i + u_j)^2 \]

Now, using Eq. (4) for \( (\mathbf{u}, \mathbf{v}) \) and (unit) eigenvector \( \mathbf{v} \) of \( L(H) \) corresponding to \( \lambda_1(H) \), we get

\[ \mathbf{v}^T L(H_X) \mathbf{v} = \sum_{(i, j) \in E(H_X)} (v_i - \sigma(i, j)v_j)^2 \]

\[ = \sum_{(i, j) \in E(H)} (v_i - \sigma(i, j)v_j)^2 - \sum_{(i, j) \in X} (v_i - \sigma(i, j)v_j)^2 \]

\[ = \mathbf{v}^T L(H) \mathbf{v} - \sum_{(i, j) \in X} (v_i - \sigma(i, j)v_j)^2 \]

Note that as \( \lambda_1(H_X) = \min_{\mathbf{v}^T L(H_X) \mathbf{v}} \frac{\mathbf{v}^T L(H) \mathbf{v}}{\mathbf{v}^T \mathbf{v}} \), \( \lambda_1(H_X) \leq \frac{\mathbf{v}^T L(H) \mathbf{v}}{\mathbf{v}^T \mathbf{v}} \).

Substituting \( \lambda_1(H_X) = \lambda_1(H) \) and \( \mathbf{v}^T \mathbf{v} = 1 \), the result is proved.

We denote the upper bound as the function \( g \), where \( g \) is

\[ g(X) = \lambda_1(H) - \sum_{(i, j) \in X} (v_i - \sigma(i, j)v_j)^2 \]

The upper bound \( g(X) \) is easier to optimize than minimizing \( \lambda_1(H) \). In particular, \( g(X) \) is a modular function and hence greedily choosing the top-b edges will achieve an optimal solution \( \cite{30} \).

**Lemma 6.** \( g(X) \) is modular (submodular and supermodular).

**Proof.** The proof is in Section 8.3.

**Algorithm:** Since \( g(X) \) is modular, we simply compute \( g(\{e\}) \), for each edge \( e = (i, j) \in C \) and select the top-b edges based on the value of \( (v_i - \sigma(i, j)v_j)^2 \), where \( b \) is the budget.

The algorithm involved in this approach requires to compute the smallest eigenpair of \( L(H) \) only once. So, we can use the Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) method proposed by Knizhner 

\[ \text{Method proposed by Knizhner} \]

This method has theoretical guarantee on linear convergence, and the costs per iteration and the memory use are competitive with those of the Lanczos method.

**3.3 Perturbation & Iterative Algorithm**

We extend the described upper bound in Lemma 5 into a tighter expression and design another way to solve MBED in an iterative fashion. Similarly, the main idea is to compute change in the smallest eigenvalue \( \lambda_1(H) \) of the Laplacian with a single edge deletion. We drop \( H \) and use \( \lambda_1(H) = \lambda_1 \) where the context is understood.

Let \( \hat{\lambda}_1 \) be the (exact) smallest eigenvalue of \( \tilde{L}(H) \), where \( \tilde{L}(H) \) is the perturbed version of \( L(H) \) obtained by deleting a single edge \( (i, j) \in E(H) \). Let \( \delta = \lambda_2 - \lambda_1 \) be the eigengap of \( L(H) \). For graphs that have sufficiently large eigengaps, we show the following result.

**Lemma 7.** Given \( \lambda_3 \) is the smallest eigenvalue of \( L(H) \) and \( \mathbf{v} \) is the corresponding unit eigenvector, for \( \delta \geq 4 \) we have \( \hat{\lambda}_1 = \lambda_1 - \frac{1}{2} (v_i - \sigma(i, j)v_j)^2 + O(1) \).

**Proof.** See App. 7.1

\text{3Lanczos algorithm \cite{32} (with Fast Multpole method \cite{6}) has a time complexity of} \( O(d_{\text{avg}} |V(H)| k) \) where \( d_{\text{avg}} \) is the average number of nonzero elements in a row of the matrix and \( k \) is the number of iterations of the algorithm.
3.2.1 Algorithm: We use Lemma 7 to design an iterative algorithm (Alg. 1). Given \( \sigma \) as unit eigenvector corresponding to the smallest eigenvalue \( \lambda_1 \), we define score of an edge \( e = (i, j) \in E(H) \) as \((\sigma_i - \sigma_j)\varepsilon_j^2\). We use this score to subsequently find the best edge from the candidate edge set \( C \) (lines 4–6). In subsequent iterations (lines 2–8) of the algorithm, we recompute the eigenvector (line 3) corresponding to the minimum eigenvalue of the perturbed matrix after the deletion of the best edge (line 7) and use LOPCG method for all such iterations to achieve faster convergence.

3.2.2 Limitations: Alg. 1 does not provide any approximation guarantee and does not directly optimize the objective in MBED. Rather, it minimizes the smallest eigenvalue. Although it is known that in a balanced graph, \( \lambda_1 = 0 \), no result is known on the bounds leads to significantly better guarantees [36]. First, we prove so that \( \sigma(1, x) = \sigma(x, i) \).

Example 3. In Fig. 1(a), nodes \( v_1 \) and \( v_2 \) are not part of the balanced subgraph \( H(\Gamma) \) due to condition (1) and condition (2) respectively.

In previous sections, we implicitly assume only one edge being considered for deletion at a node \( y \) where \( \{\text{cep}(H, y)\} > 0 \). This result, however, we explicitly assume any edge being considered for deletion is a peripheral edge. Note, however, that following an edge deletion, the set of peripheral edges changes. Empowered with these observations, we next establish pseudo-submodularity.

4 APPROXIMATION ALGORITHMS

In § 2.1, we showed that MBED is non-monotonic. We next show that if the set of deleted edges \( X \) is selected strategically, then monotonicity can be guaranteed. If the optimization function is monotonic and pseudo-submodular, then greedy algorithms can produce approximation bounds. The rest of the section builds towards this result.

Observation 1. If the set of deleted edges \( X \) is chosen such that \( H_X \) and \( H \) have same number of connected components, then the objective function \( f \) is monotonic, i.e., \( f(S) \geq f(T) \) for \( S \subseteq T \) if \( e \) is deleted.

Proof. For all of the subsequent discussions, we will use \( S(H) \) to denote the largest balanced subgraph of \( H \) with the two vertex sets being \( V_1 \) and \( V_2 \). The deleted edge \( e \) can fall in one of three categories. (1) Both endpoints lie in \( V_1 \) (or equivalently \( V_2 \)), in which case \( \Delta(H) = \Delta(H_X) \) since \( H_X \) and \( H \) have same number of connected components. (2) One endpoint lies in \( V_1 \) and the other in \( V_2 \). Even in this case \( \Delta(H) = \Delta(H_X) \). (3) One endpoint in \( V_1 \) (or \( V_2 \)) and the other in \( V \setminus \{V_1, V_2\} \). In this case, the node in \( V_1 \) continues to stay there while the other endpoint may move into \( V_2 \) or \( V_2 \) and thus \( \Delta(H) \leq \Delta(H_X) \).

Choosing \( X \) is in our control. Hence, we may assume that MBED is monotonic by ensuring that \( X \) satisfies the constraint outlined in Obs. 1. We next establish that although MBED is not submodular (Lem. 2), it is pseudo-submodular (Thm. 2).

4.1 Pseudo-Submodularity

We first prove that our objective function is pseudo-submodular (Thm. 2) and then provide approximations (Thms. 3 and 4) via Randomized Greedy and Greedy algorithms.

**Definition 5 (Contradictory Edge-pair).** Given a subgraph \( H \) with largest balanced subgraph \( H' \) having balance partition \( (V_1, V_2) \) two edges \( e_1, e_2 \) form a contradictory edge-pair if any of these conditions hold for some \( u, u' \in V_1 \) and \( w, w' \in V_2 \), and \( x \notin V_1 \cup V_2 \) : 
1. \( e_1 = (u, u') \) and \( e_2 = (w, w') \) such that \( \sigma((x, u)) = \sigma((x, w')) \).
2. \( e_1 = (u, u') \) and \( e_2 = (u, w') \) such that \( \sigma((x, u)) = -\sigma((x, w')) \).
3. \( e_1 = (u, w) \) and \( e_2 = (w, w') \) such that \( \sigma((x, w)) = -\sigma((x, w')) \).

We use cep \( (H, x) \) to denote the set of contradictory edge-pairs for subgraph \( H \) with one end at node \( x \). A contradictory edge pair restricts node \( x \) from contributing to the balance. This property is more formally expressed as follows.

**Observation 2.** A node \( x \) will not be part of \( S(H) \) if some of the following conditions hold: (1) \( \{\text{cep}(H, x)\} > 0 \), (2) the node \( x \) is connected to \( S(H) \) only via paths ending at a node \( y \) where \( \{|\text{cep}(H, y)|\} > 0 \).

Example 3. In Fig. 1(a), nodes \( v_1 \) and \( v_2 \) are not part of the balanced subgraph \( H(\Gamma) \) due to condition (1) and condition (2) respectively.

Obs. 2 allows us to formally define when an edge deletion increases the balance.

**Observation 3.** \( f(\{e\}) > 0 \) iff \( e, e' \in \text{cep}(H, x) \) for some \( e' \in E, x \in V(H), \) and \( \{|\text{cep}(H, x)|\} > 0 \), i.e., following deletion of \( e \) does not associate with any contradictory edge pair.

From Obs. 3, it follows that only the deletion of a peripheral edge may result in increase of balance. A peripheral edge has one endpoint within \( S(H) \) and the other outside \( S(H) \). Owing to this result, hereon, we implicitly assume any edge being considered for deletion is a peripheral edge. Note, however, that following an edge deletion, the set of peripheral edges changes. Empowered with these observations, we next establish pseudo-submodularity.

4.1.1 Local Pseudo-Submodularity.

**Definition 6 (Pseudo-submodularity [36]).** Given a scalar \( 0 \leq \gamma \leq 1 \), a function \( f \) is pseudo-submodular if \( \sum_{e \in R} [f(Q \cup \{e\}) - f(Q)] \geq \gamma(f(Q \cup R) - f(Q)) \) for any pair of disjoint sets \( Q, R \subset C \).

Note that the pseudo-submodularity ratio \( \gamma \) is a pessimistic bound over all pairs of disjoint sets. Instead of using \( \gamma \), we compute approximation bounds on a local submodularity ratio [36] defined on two sets, \( Q, R \), i.e., a non-negative \( \gamma_{Q,R} \) satisfying \( \sum_{e \in R} [f(Q \cup \{e\}) - f(Q)] \geq \gamma_{Q,R} f(Q \cup R) - f(Q) \). It has been shown that using local bounds leads to significantly better guarantees [36]. First, we prove a lower bound for \( \gamma_{Q,R} \) as follows:

**Theorem 2.** For two disjoint sets \( Q, R \),
\[ \sum_{e \in R} [f(Q \cup \{e\}) - f(Q)] \geq \gamma_{Q,R} [f(Q \cup R) - f(Q)] \]
where \( \gamma_{Q,R} \geq \frac{1}{4+\Delta(H_Q)(|R|^{-1})} \).

Proof. See App. 7.2.

This theorem proves a lower bound for \( \gamma_{Q,R} \) for any disjoint sets \( Q \) and \( R \). Obs. 1 and Thm. 2 show that the monotonicity and local pseudo-submodularity holds for our objective function. We next leverage these properties to design a randomized greedy algorithm with approximation guarantees.

4.2 Randomized Greedy (RG)

**Lemma 8 ([36]).** Assuming \( 0 \leq \gamma \leq 1 \) for \( i \in \{0, 1, 2, \ldots, k - 1\} \) so that \( \sum_{e \in \text{OPT}} [f(S_i \cup \{e\}) - f(S_i)] \geq \gamma_i [f(S_i \cup \text{OPT}) - f(S_i)] \) (local pseudo-submodularity) throughout the execution of the RG algorithm, where \( f \) is monotonic, OPT denotes the optimal set of edges,
and $S_i$ denotes the set of chosen elements after the $i$-th iteration (i.e. $|S_i| = 1$); then $Rg$ obtains an approximation of $1 - \exp \left( -\frac{1}{k} \sum_{i=1}^{k} y_i \right)$ with a high probability.

We can directly apply this lemma in our setting. The $Rg$ Algorithm is described as Algorithm 2.

**Theorem 3.** For MBED, the $Rg$ algorithm obtains an approximation of $1 - e^{-y'}$, and $y' \geq 4 \cdot \frac{1}{4 + \lambda^R (b-1)}$, where $b$ and $\lambda^R$ denote the budget and the balance after deleting the optimal set of edges respectively.

Proof. Let us denote the optimal set of $b$ edges as $B^*$. By monotonicity, we get $\Delta(H_{S_b}) \leq \Delta(H_{S_{b'}}) \leq \Delta(H_{S_{b'}}) \leq \Delta^R$. From Theorem 2, $y_{S_b, B^*} \geq \frac{4}{4 + \Delta(H_{S_b})(b-1)} \geq \frac{4}{4 + \lambda^R (b-1)}$. Now, by substituting $y_i$ with $y_{S_b, B^*}$ in Lem. 8 we get the desired result. \qed

**Improved Bounds:** The lower bound of $y'$ in Thm. 3 can be tightened. In particular, $y' \geq \frac{4}{4 + \lambda^S (b-1)}$, where $\lambda^S$ denotes the balance after deleting the solution set of $b$ edges produced by the $Rg$.

The bound could be further improved as $y' \geq \frac{4}{4 \lambda^R + \Delta(\arg max \{ f(e) \})}$, where $\psi^R$ is the summation of marginal gains of the edges in the optimal solution set over the solution set produced by $Rg$ (see App. 7.3). Table 2 summarizes the additional lower bounds of $y'$ (where the approximation guarantee is $1 - e^{-y'}$) that can be derived on the $Rg$.

**Implementation:** Alg. 2 first computes the set of peripheral edges of the initial balanced subgraph $S(H)$ (line 3). After that, for all peripheral candidate edges, $f(\{e\})$ is computed (lines 4 – 5). Using these values, the subset of peripheral edges of cardinality $b$ maximizing the sum of $f(\{e\})$ is chosen and a random edge from this subset is selected for deletion (lines 6 – 8). Following this edge deletion, the balanced subgraph $S(H)$ is updated to include the newly compatible nodes (line 9). The peripheral edge set for the updated $S(H)$ is recomputed (line 3) and this process continues in an iterative manner for $b$ iterations.

### 4.3 The Greedy Approach

The only difference with Alg. 2 is that instead of choosing a random edge from the top $b$ edges with the highest sum of $f(\{e\})$ (lines 6 – 7), the greedy algorithm (Greedy) chooses the edge with the highest $f(\{e\})$, i.e. $e_g = \arg \max_{e \in C} \{ f(\{e\}) \}$.

**Theoretical Bounds:** We derive the approximation of Greedy in App. 7.4. Table 2 summarizes the different lower bounds of $y'$ (where the approximation guarantee is $1 - e^{-y'}$).

### 4.4 Time Complexity

Alg. 2 comprises of three main dominating parts with respect to the time complexity: (i) calls to compute function $f(\{e\})$ for all candidate edges, (ii) computing peripheral edge set (line 3) and (iii) finally updating the balanced subgraph $S(H)$ (line 9).
5.2 Efficacy and Efficiency

5.2.1 Small budget on all datasets: Fig. 2 shows the percentage increase in balance (IB) for eight datasets achieved by each algorithm. Greedy and Rg outperform all the baselines by up to 12%. Besides having approximation guarantees (Thms. 3 and 4), Greedy and Rg directly optimize the objective function in an iterative fashion. In contrast, the baselines choose solution edges depending on other criterion. In particular, the spectral methods Isa and Spec-Top do not perform well since it chooses edges based on an upper-bound to minimize the minimum eigenvalue of the corresponding Laplacian. Though the balanced graph has minimum eigenvalue of the Laplacian as 0, the rate at which the edge deletions move towards achieving it, might still be low. We also observe that Greedy, in general, performs better than Rg. It would be wrong, however, to draw the conclusion that Greedy is always better. In subsequent experiments where we choose k-cores as the input subgraphs, we will see that Rg performs better. We will revisit the topic of Greedy vs Rg while discussing that experiment.

5.2.2 Larger budget on large datasets: To further demonstrate the efficacy of our methods we vary the budget as a function of C, i.e., all edges in H. Fig. 3 shows the percentage increase in balance (IB) for the four largest datasets. Consistent with previous experiments, Rg and Greedy outperform all baselines (better by up to 6% points). More interestingly, we observe that a substantial increase in balance is feasible (9% or up to 4000 nodes) by deleting only 0.1% of edges (∼500 edges). In other words, improvement in balance-dependent community functions, such as team performance or stability, may be significantly improved through minor adjustments to the network.

5.2.3 Scalability: Table 4 shows the running times of all algorithms against budget in the three largest datasets. Although Rg and Greedy are slower than the other baselines, they finish within a few minutes even on a million edges’ network. Thus, scalability to large networks is not a concern. A more interesting behavior is witnessed in the correlation between efficacy and efficiency. More specifically, we observe that the better performance of an algorithm in IB%, the higher is its running time. When an algorithm performs better, it means in each iteration, the algorithm produces a larger cascading impact following an edge deletion. Higher cascading impact leads to a larger number of new peripheral edges coming into consideration. Consequently, the running time goes up.
5.3 Impact of Community Density

In this experiment, we systematically vary the density of the input community $H$ and analyze its impact on the performance. To control the density of $H$, we use $k$-core [38] as the input subgraph. As $k$ increases, $H$ gets denser. Table 5 shows the maximum and minimum $k$-core sizes along with their balance for each dataset. We vary the value of $k$ depending on the $k$-core distribution of the graph. As high $k$-cores contain fewer nodes, the highest value of $k$ is chosen such that the size of the $k$-core is at least 10% of the original graph size in terms of number of nodes.

Fig. 4 presents the results. In this section, we only consider the three best-performing algorithms of Greedy, Rg and Min-Cep. Greedy and Rg continue to be the best performers. Another interesting behavior we observe is that, the higher the $k$, and therefore density, the smaller is the gap between Greedy and Rg. In some cases, Rg performs better than Greedy. This behavior is a direct consequence of how Rg and Greedy operates. Greedy deterministically chooses the edge with the highest marginal gain. Consequently, when the gradient of the marginal gains in the sorted order is high, choosing the highest edge produces a good result. However, when the gradient is small and several edges provide similarly high marginal gains, Rg performs better.

5.4 Visualizations on Bitcoin Network

In the next experiment, we visually inspect the impact of edge deletions on increasing balance in the BitcoinOTC data. Fig. 5 presents the gradual increase in the size of the balanced component following 5 and 10 edge deletions. It shows that: (1) both positive and negative edges are chosen for deletion, and (2) there may be significant cascading impact of a single deletion (as visible in the appearance of several new green squares in Fig. 5(c)).

6 CONCLUSIONS

In this paper, we studied the problem of maximizing the balance in signed networks via edge deletion. While existing studies have focused primarily on finding the largest balanced subgraph, we adopted a network design approach to improve balance inside a subgraph. We proved that the problem is NP-hard, non-submodular, and non-monotonic. To overcome the resultant computational challenges, we designed an efficient heuristic based on the relation of Laplacian eigenvalues with the balance in corresponding signed graphs. Since these heuristics do not exhibit approximation guarantees, we leverage pseudo-submodularity of the objective function to design greedy algorithms with provable approximation guarantees. Through an extensive set of experiments, we showed that the proposed approximation algorithms outperform the baseline algorithms while being scalable to large graphs. An interesting future direction would be to explore alternative network design mechanisms such as node deletion and edge-sign flips to improve balance. From a theoretical perspective, we also aim to investigate the parameterized complexity of balance-related design problems.
7 APPENDIX

7.1 Proof of Lemma 7

Proof. Given a signed graph $G = (V, E, \sigma)$, a subgraph $H$, let $\lambda_1$, $\lambda_2$ be the eigenvalues of $L(H)$ and the perturbed matrix $\tilde{L}(H)$ (after single edge $(i, j)$ deletion) respectively where $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_m$.

We have $\tilde{L}(H) = L(H) + P$, and perturbation matrix $P = D + S$, where $D$ is a diagonal matrix with $D_{ii} = D_{jj} = -1$ and 0 otherwise. $S_{ij} = S_{ji} = \sigma(i, j)$ for the perturbed edge $(i, j) \in E$ and otherwise 0. Given $\sigma$ as the unit eigenvector corresponding to $\lambda_1$ we have,

$$\tilde{\lambda}_1 = \lambda_1 + \sigma^T D \sigma + O(||P||_F^2) = \lambda_1 + \sigma^T D \sigma + O(||P||_F^2) = \lambda_1 + \sum_{k=1}^n \lambda_k + \sum_{i,j} \sigma(i,j) \nu_i \nu_j + o(1),$$

$$\tilde{\lambda}_i = \lambda_i - \sum_{k=1}^n \lambda_k + \sum_{i,j} \sigma(i,j) \nu_i \nu_j + o(1),$$

$$\tilde{\lambda}_i = \lambda_i - \sigma(i,j) \nu_i \nu_j + o(1).$$

Now, to show that $\tilde{\lambda}_1(H)$ is indeed the smallest eigenvalue of $\tilde{L}(H)$, using matrix perturbation theory (p. 203 [37]), we have

$$\tilde{\lambda}_1 \leq \lambda_1 + ||P||_F \leq \lambda_1 + 2$$

Since the spectral gap $\delta = \lambda_2 - \lambda_1 \geq 4$, we have $\tilde{\lambda}_1 \geq \lambda_1$. So, we have $\lambda_1 = \tilde{\lambda}_1$ is the smallest eigenvalue of $\tilde{L}(H)$. \(\square\)

7.2 Details for proof for Theorem 2

Before proving Thm. 2, we derive a few results. Let $\mathcal{V}$ be the node set that gets added in the maximum balanced subgraph $S(H)$ after deleting $B$ edges. We know that $\forall u \in \mathcal{V}$ there exists $v \in S(H)$ such that $(u, v) \in B$. The inclusion of one node may lead to including more nodes in the balanced portion. Let $C_u$ be the size of component that gets added with $u \in \mathcal{V}$ and $C^* = \max\{C_u, u \in \mathcal{V}\}$.

Observation 4.

$$C^* + 1 \leq \frac{\Delta(H)}{2} \quad (6)$$

(a) Initial subgraph (b) After 5 deletions (c) After 10 deletions

Figure 5: Visualization of the impact of edge deletions by GREEDY. Green and orange denote the two partitions of the balanced subgraph $S(H)$; grey denotes the component outside $S(H)$. The solid red and blue edges are positive and negative, respectively, while the dashed edges in (b) and (c) are the ones being deleted. (b) and (c) also show the new components being added to the balanced portion through green and orange squares.

Proof by Contradiction. If $[C_u] > \frac{\Delta(H) - 2}{2}$, then the initial $S(H)$ would consist of the larger among $V_1$ and $V_2$ (which would be at least of size $\frac{\Delta(H)}{2}$) along with $\{u\} \cup C_u$.

Choice of $a(B)$ and Peripheral Edges (PE): Let $a(B)$ be the number of nodes $x$ satisfying: (1) $\exists e \in E$ and $\exists x \in V$ such that $|\text{cep}(H, x)| > 0$. We use Obs. 3 to restrict $a(B)$ to always belong to the periphery of the current balanced subgraph. An upperbound of $f(B)$ is as follows.

Lemma 9.

$$f(B) \leq \sum_{i=1}^b f(\{e_i\}) + (C^* + 1)a(B). \quad (7)$$

Proof. This is proved using induction (Sec. 8.4). \(\square\)

7.2.1 Final proof for Theorem 2

Proof. Note that $f(Q \cup R) - f(Q) = \Delta(H_Q | R) - \Delta(H_Q)$. We can write this as $\Delta(H_Q') - \Delta(H'_Q)$, where $H'_Q = H_Q$. That means marginal gain in balance of deleting the set $R$ over $Q$ is same as the marginal gain in balance of deleting the set $R$ from $H_Q$. We can thus use $f'(R) = \Delta(H'_R) - \Delta(H'^R)$ in place of $f$. Thus, by Lem. 9:

$$f'(B) \leq \sum_{i=1}^b f'(\{e_i\}) + (C^* + 1)a(B) \quad (8)$$

where $C^*$ and $\alpha$ are defined accordingly to new initial subgraph $H'^Q = H_Q$. Next, we propose an upper bound of $\alpha(\cdot)$ as follows:

$$\alpha(B) \leq \frac{|B| - 1}{2} \quad (9)$$

This is true since we need at least two edges for one node to be counted in $\alpha(B)$.

Now we have:

$$\sum_{e \in R} \frac{f(Q \cup \{e\}) - f(Q)}{f(Q \cup R) - f(Q)} \geq \frac{1}{1 + \frac{a(B)(C^* + 1)}{\sum_{e \in R} f'(\{e\})}} \quad (\text{Replace } f'(R) \text{ using Eq. 8})$$

$$\geq \frac{1}{1 + \frac{a(B)(C^* + 1)}{\sum_{e \in R} f'(\{e\})}} \quad (\text{Using the upper bound of } \alpha \text{ in Eq. 9})$$

$$\geq \frac{1}{1 + \frac{\Delta(H_Q)(|R| - 1)}{2 \sum_{e \in R} f'(\{e\})}} \quad (\text{Eq. 6})$$

\(\square\)

We also show a construction for the tight lower bound in Thm. 2 (Sec. 8.5).

7.3 Proof with bound $\frac{4\psi'}{4\psi' + \Delta(H_Q(b-1))}$

In proof of Thm. 2, we have $\sum_{e \in R} f'(\{e\}) \geq 1$. However, $\sum_{e \in R} f'(\{e\}) \geq \psi'$, where $\psi'$ is the summation of marginal gains of the elements in the optimal solution set (i.e., $R$) over the solution set produced by Rg (i.e., $Q$). Now replacing $\sum_{e \in R} f'(\{e\})$ as $\psi'$ we get, $\psi' \geq \frac{4\psi'}{4\psi' + \Delta(H_Q(b-1))}$ according to Thm. 3.
7.4 Approximation by Greedy

**Lemma 10 ([8]).** Given $f$ is a non-negative and monotone set function, budget $b$, and $\sum_{e \in R} (f(S \cup \{e\}) - f(S)) \geq \gamma \cdot (f(S \cup R) - f(R))$ where $S^\circ$ is the final set selected by the Greedy Algorithm, then the algorithm has the following approximation guarantee of $\left(1 - e^{-\gamma}b^*\right)$ where $S^\circ, b^* = \min|y|$ for any $R, S^\circ \subseteq R = \Phi$.

We apply this result in our problem setting:

**Theorem 4.** For the MBED problem, Greedy algorithm obtains an approximation of $1 - e^{-\gamma}$ and $y^* \geq \frac{4\Delta(1-b)}{1+\Delta}$ where $b$ and $\Delta$ denote the budget and the balance after deleting the optimal set of edges respectively.

Proof. Let the optimal set of $b$ edges be $B^*$ and let $S^\circ$ denote the final edge set by the Greedy algorithm. Also, let $\Delta^*$ denote the balance after deleting the optimal set of edges, then by its definition we have $\Delta(H_{S^\circ}) \leq \Delta^*$. From Theorem 2, $y^* \geq \frac{\gamma S^\circ, B^*}{3\Delta(H_{S^\circ}) + |B^*| - 1} \geq \frac{\Delta}{2\Delta^*}$. So, substituting $S^\circ, b^*$ in Lem. 10 as $S^\circ, |B^*|$ (for $y^*$), we get the desired approximation of $\frac{\Delta}{4\Delta^*}$.

The other lower bounds for $y^*$ (where the approximation produced by Greedy is $1 - e^{-\gamma}$) as $\frac{\gamma}{4\Delta^* + 1}$ and $\frac{\gamma}{4\Delta^* (1-b)}$ can be derived in similar ways as in the case of $\Delta$.

**REFERENCES**

[1] Jin Akiyama, David Avis, Vasěk Chvátal, and Hiroshi Era. 1981. Balancing signed graphs. *Discrete Applied Mathematics*, 3 (4), 1981, 227–233.

[2] Ashwin Arulselvan. 2014. A note on the set union knapsack problem. *Discrete Applied Mathematics*, 164 (2014), 214–218.

[3] O. Askarisichani, J. Ng Lane, F. Bullo, N. E. Friedkin, A. K. Singh, and B. Uzzi. 2019. Integration of AI and OR Techniques in Constraint Programming. *Springer*, 76–91.

[4] Francesco Belardo. 2014. Balancedness and the least eigenvalue of Laplacian of signed graphs. *Linear and Multilinear Algebra*, 2, 2 (1953), 143–146.

[5] Francisco Belardo and Vladimir Koolen. 2013. A fast divide-and-conquer algorithm for computing the spectra of real symmetric tridiagonal matrices. *Applied and Computational Harmonic Analysis*, 34, 3 (2013), 379–414.

[6] Ed S Coale and Vladimir Rokhlin. 2013. A fast divide-and-conquer algorithm for computing the spectra of real symmetric tridiagonal matrices. *Applied and Computational Harmonic Analysis*, 34, 3 (2013), 379–414.

[7] Pierluigi Crescenzi, Gianlorenzo D’Angelo, Lorenzo Severini, and Yilka Velas. 2015. Greedily Improving Our Own Centrality in A Network. In *SEA*, 11, 9 (2018), 74–107.

[8] Bhaskar DasGupta, German Andres Enciso, Eduardo Sontag, and Yi Zhang. 2007. The set-union problem. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. 169–175.

[9] S. Medya, A. Silva, and A. Singh. 2020. Approximate Algorithms for Data-driven Influence Limitation. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[10] Sourav Medya, Arlezi Silva, Ambuj Singh, Prithwish Basu, and Ananthan Swami. 2018. Group centrality maximization via network design. In *Proc. 24th SIAM International Conference on Data Mining, SIAM*, 126–134.

[11] Sourav Medya, Jithin Vachery, Sayan Ranu, and Ambuj Singh. 2018. Noticeable network delay minimization via node upgrades. *Proceedings of the VLDB Endowment*, 11, 9 (2018), 988–1001.

[12] Alejandro Ordozgoiti, Antonis Matozakis, and Aristides Gionis. 2020. Finding large balanced subgraphs in signed networks. In *Proceedings of The Web Conference 2020*, 1378–1388.

[13] Bruno Ordozgoiti, Antonis Matozakis, and Aristides Gionis. 2020. Finding large balanced subgraphs in signed networks. In *Proceedings of The Web Conference 2020*, 1378–1388.

[14] Hauke Pausch. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8, 3 (2017), 489–508.

[15] Chengbin Peng, Tamara G Kolda, and Ali Pinar. 2014. Accelerating community detection by using k-core subgraphs. *arXiv preprint arXiv:1403.2226* (2014).

[16] Svatopluk Poljak and Daniel Turzík. 1986. A polynomial time heuristic for certain graph: Applications and solution approaches. *Springer*, 76–91.

[17] George L Nemhauser and Laurence A Wolsey. 1978. Best algorithms for approximating maximum of a submodular set function. *Mathematics of operations research*, 3, 1 (1978), 177–188.

[18] Bruno Ordozgoiti, Antonis Matozakis, and Aristides Gionis. 2020. Finding large balanced subgraphs in signed networks. In *Proceedings of The Web Conference 2020*, 1378–1388.

[19] Lorenzo Orcioni, Sushant Sachdeva, and Nisheeth K Vishnoi. 2012. Approximating the exponential, the Lanczos method and an O (m) time spectral algorithm for balanced separator. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, 1141–1160.

[20] Richard Santiago and Yuichi Yoshida. 2020. Weakly Submodular Function Maximization. *Machine Learning Research*, 11, 9 (2020), 1141–1160.

[21] Richard Santiago and Yuichi Yoshida. 2020. Weakly Submodular Function Maximization. *Machine Learning Research*, 11, 9 (2020), 1141–1160.

[22] Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2017. Finding multiple source-destination pairs. *GeoInformatica*, 20, 2 (2016), 365–404.

8 ADDITIONAL PROOFS

8.1 NP-hardness

Proof. Let $Sk(U, S, P, W, q)$ be an instance of the Set Union Knapsack Problem [14], where $U = \{u_1, \ldots, u_q\}$ is a set of items, $S = \{S_1, \ldots, S_m\}$ is a set of subsets $(S_i \subseteq U, P : S \rightarrow R_+)$ is a subset profit function, $W : U \rightarrow R_+$ is an item weight function, and $q \in R_+$ is the budget. For a subset $A \subseteq S$, the weighted union of set $A$ is $W(A) = \sum_{i \in A} S_i w_e$ and $P(A) = \sum_{i \in A} P_i$. The problem is to find a subset $A^* \subseteq S$ such that $W(A^*) \leq q$ and $P(A^*)$ is maximized. Sk is NP-hard to approximate within a constant factor [2]. We reduce a version of Sk with equal profits and weights (also NP-hard) to the Mbed problem. We define a corresponding Mbed problem instance via constructing a graph $G$ as follows.
For each $S_i \in S$ and $u_j \in U$ we create nodes $x_i$ and $y_j$ respectively. We also add a node $v$ with a large connected component $L$ of size $l$ only with positive edges attached to it. The node $v$ has negative edges with every node $x_i$, $\forall i \in [m]$ and every node $y_j$, $\forall j \in [n]$. Additionally, if $u_j \in S_i$, a negative edge $(x_i, y_j)$ will be added to the edge set $E$.

In $\mathcal{M}_{\mathcal{A}ed}$, the number of edges to be removed is the budget, $b = q$. The candidate set, $\mathcal{C} = \{(v, y_j)\vert y_j \in [n]\}$. Note that the initial largest connected balanced component is $\{v \cup L\} \cup \{y_j \vert y_j \in [n]\}$ if $l > m + 1$ (assuming $n > m$). Our claim is that, for any solution $\mathcal{A}$ of an instance of $\mathcal{S}K$ there is a corresponding solution set of edges, $B$ (where $|B| = b$) in the graph $\Gamma$ of the $\mathcal{M}_{\mathcal{A}ed}$ version, such that $f(B) = P(\mathcal{A}) + n + l + 1$ if $B = \{(v, y)\vert y \in \mathcal{A}\}$ are removed.

In the new balanced graph, we aim to build two partitions ($W_1$ and $W_2$) as follows. One partition $W_1$ consists of $\{v \cup L\}$ initially. Our goal is to delete edges from $\mathcal{C}$ and add the nodes $y_j$'s in $W_1$. If $(n, y_j)$ for any $j$' does not get deleted then it would be in $W_2$. If there is any node $x_i$ that is connected with only nodes in $\mathcal{A}$ besides connected with $v$, then removing all the edges in $B$ would put the node $x_i$ in $W_2$. Thus removing edges in $\mathcal{A}$ would put $P(\mathcal{A})$ nodes in $W_2$. Thus, $f(B) = P(\mathcal{A}) + n + l + 1$.

\section{8.2 Proportionally Submodular}

\textbf{Lemma 8.1.} The objective function $f$ is not proportionally submodular [36]. In other words, there exists $S, T \in E$ for some graph $H$ such that $|T|f(S) + |S|f(T) < |S \cap T|f(S \cup T) + |S \cup T|f(S \cap T)$.

\textbf{Proof.} Consider a balanced subgraph of $H$, $(S, H(S))$ has a partition $V_1$ and $V_2$. A node $v$ is outside $S(H)$ and it is connected to $V_1$ with positive edges $e_1$ and $e_2$, $V_2$ with another positive edge $e_3$. Thus the node $v$ cannot be the part of $S(H)$. Consider an edge $e_4$ inside $V_1$ which can be removed without making the graph disconnected. Let us assume $S = \{e_1, e_2\}, T = \{e_2, e_4\}$. Then, $f(\{(e_1, e_2)\}) = 0$ and $f(\{(e_2, e_1)\}) = 0$, even after removing any of these edges it is not possible to add the node $v$ to $S(H)$. Note that $f(S \cap T) = f(\{(e_1)\}) = 0$. However, $f(S \cup T) = f(\{(e_1, e_2)\}) = 1$ since the node $v$ can be added. Substituting these values, we get $|T|f(S) + |S|f(T) < |S \cap T|f(S \cup T) + |S \cup T|f(S \cap T)$.

\section{8.3 Proof of Lemma 6}

We denote $g_X(Y)$ as the marginal gain of the set of edges $Y$ over the set $X$, i.e., $g_X(Y) = g(X \cup Y) - g(X)$. To prove modularity, we need to show $g_X(Y) = \sum_{e \in Y} g_X(e)$, i.e. the marginal gain of the set of $Y$ over $X$ is the summation of the marginal gains of each individual in $Y$ over $X$ for any $X, Y$.

\textbf{Proof.} We can write $g_X(Y)$ as follows.

$$g_X(Y) = -\sum_{(i,j) \in X \setminus Y} (v_i - \sigma(i, j)v_j)^2 - \sum_{(i,j) \in X} (v_i - \sigma(i, j)v_j)^2$$

$$= -\sum_{(i,j) \in Y} (v_i - \sigma(i, j)v_j)^2 - \sum_{e \in Y} g_X(e)$$

\section{8.4 Proof of Lemma 9}

\textbf{Proof.} We prove this by induction on the number of edges, $b$. Let us denote $B_k \subseteq B$ as {$e_1, \ldots, e_k$}. We construct $B$ by only considering peripheral edges $e_{k+1}$ such that, for all $k \leq b$: $(e_{k+1}, e') \in cep(H_{B_k}, x)$, for some node $x$ and edge $e'$. 

\textbf{Base case ($b = 1$):} $f(\{(e_1)\}) \leq f(\{(e_1)\})$. Also, $\alpha(\{(e_1)\}) = 0$.

\textbf{Inductive hypothesis (IH):} Suppose the equation holds for $b = k$, i.e., $f(B_k) \leq \sum_{i=1}^{k} f(\{(e_i)\}) + (C^* + 1)\alpha(B_k)$.

\textbf{Inductive step ($b = k + 1$):} We present different cases for $e_{k+1}$.

\textbf{Case 1:} $(e_{k+1}, e') \in cep(H_{B_k}, x)$ and $\text{cep}(H_{e_{k+1}})$, $x = 0$, i.e., after deleting $e_{k+1}$, $x$ moves into the balanced subgraph. Then, we must also have $\text{cep}(H_{B_{k+1}}, x) = 0$. Hence, $f(B_k \cup \{(e_{k+1})\}) = f((e_{k+1}))$ and the inequality holds.

\textbf{Case 2:} Either (1) $(e_{k+1}, e') \in cep(H_{B_k}, x)$ and $\text{cep}(H_{e_{k+1}}, x) > 0$ or (2) $(e_{k+1}, e') \notin cep(H_{B_k}, x)$.

Thus, by Observation 2, we have $f(\{(e_{k+1})\}) = 0$.

\textbf{Case 2a:} Suppose $\text{cep}(H_{B_{k+1}}, x) = 0$. Then by definition of $\alpha$, $C^*$, we have $\alpha(B_{k+1}) = \alpha(B_k) + 1$, and $f((B_k \cup \{(e_{k+1})\})) - f(B_k) \leq C^* + 1$.

Substituting this, we get $f(B_{k+1}) \leq C^* + 1 + \sum_{i=1}^{k} f(\{(e_i)\}) + (C^* + 1)\alpha(B_{k+1})$.

\textbf{Case 2b:} In other cases, $f(B_k \cup \{(e_{k+1})\}) - f(B_k) = f((e_{k+1})) = 0$. This exhausts our cases and the claim is true $\forall b, b > 0$.

\section{8.5 Construction for the tight lower bound in Thm. 2}

One can construct a graph $H$ and the sets $Q, R$ where equality holds. In particular, let $R$ be of an arbitrary size $b$. Consider $H_Q$ to have the MBS partition as $V_1, V_2$ each of size $\frac{\Delta(H_Q)}{2}$. Nodes of type 1 (Obs. 2) are attached to these each with the sole connected component of size $\frac{\Delta(H_Q) - 2}{2}$. Let these nodes have 3 such connections (thus, removing two will help - any two such that our "connected assumption" holds are in the set $R$). We have another node of type 1 such that only two such connections are connected and one of these is in $R$ and the connected component $C$ to it is of size 0. This completes the set $R$. Thus, $\sum_{e \in R} f(Q \cup \{e\}) - f(Q) = 1$ and $f(Q \cup R) - f(Q) = 1 + \left(\frac{\Delta(H_Q) - 2}{2} + 1\right) \frac{b-1}{2}$. 

\end{document}