GENIUS: Subteam Replacement with Clustering-based Graph Neural Networks

Chuxuan Hu*, Qinghai Zhou*, Hanghang Tong*

Abstract

The state of the art for *subteam replacement*, based on random walk graph kernels, encounter the following limitations: (1) ineffective in capturing fine-grained node feature correlations, (2) inefficient without proper pruning mechanisms, and (3) limited applicability to single-member or equal-sized subteam replacements. In this paper, we address these limitations by proposing GENIUS, a clustering-based graph neural network (GNN) framework that (1) captures team social network knowledge for *subteam replacement* by deploying team-level attention GNNs (*TAGs*) and self-supervised *positive team contrasting* training scheme, (2) generates unsupervised team social network member clusters to prune candidates for fast computation, and (3) incorporates a *subteam recommender* that selects new subteams of flexible sizes. We demonstrate the efficacy of the proposed method in terms of (1) *effectiveness*: being able to select better subteam members that significantly increase the similarity between the new and original teams, and (2) *efficiency*: achieving more than 600× speed-up in average running time.

Keywords– Subteam Replacement, Social Network Analysis, Graph Neural Networks

1 Introduction

The emergence of network science of teams has revolutionized the way to characterize, predict, and optimize teams that are embedded in large-scale social networks [18]. Among others, given a team of people working on the same task (e.g., launching a targeted web service, proceeding towards a specific research topic), *subteam replacement* is dedicated to finding the optimal set of people who can best perform the function of a subset of the original team (i.e., subteam), if the subteam becomes unavailable for certain reasons. The applications of *subteam replacement* are abundant. To name a few, in a conference organization committee, there are always some returning members each year, and in the meanwhile, new members should be invited; in a software development team, a subteam of employees might be assigned to other departments due to new business requirements; in artistic groups like choirs or dance troupes, a subgroup of artists might be unable to show up in a play due to scheduling and/or staging conflicts. In these situations, the goal of *subteam replacement* is to avoid the potential deterioration of the team's performance due to the absence of the leaving subteam members.

The state of the art adopts the following two principles in designing *subteam replacement* algorithms. First (*structural similarity*), to reduce the disruption to the current team, the recommended new subteam should have a similar collaboration relationship as the unavailable subteam in terms of the connections with the residual subteam members. Second (*skill similarity*), the new members should possess a similar skill set as the members from the unavailable subteam to optimally meet the team-level task requirements. Most, if not all, of the existing works leverage random walk graph kernel [25] as the cornerstone for *subteam replacement* to capture both the structural and skill match, where the key idea is to select members that can achieve the largest similarity between the newly constructed and the original teams [19, 21].

Although these approaches recommend reasonably well-qualified new subteams in *subteam replacement*, there are three major limitations with the graph kernel-based methods. First, from the perspective of effectiveness, in quantifying the similarity between teams, graph kernel-based methods separately exploit the node attributes (i.e., skills), which inevitably ignores the potential correlations between various skills. For example, in research team replacement (e.g., on the DBLP dataset), the researchers' skills correspond to the research areas in computer science (e.g., data mining, NLP, machine learning, etc.). Graph kernel-based methods will separately evaluate the closeness between two teams regarding an individual research area (i.e., skill), without considering the essential correlation among different areas (e.g., machine learning versus data mining), which will sabotage the performance of graph kernel-based approaches. It is worth noting that we observe a

*University of Illinois at Urbana-Champaign, ({chuxuan3, qinghai2, htong}@illinois.edu)

drastic performance drop w.r.t. multiple graph similarity as the skill set scales up (e.g., keywords in publications), and we have a detailed discussion in Sections 3 and 4. Second, graph kernel-based methods could incur a high time and space cost since they iteratively calculate the similarity between the original team and the new team with candidate members from an enormous search space. Furthermore, existing *subteam replacement* methods generate new subteams of the same size as the unavailable subteam. Nonetheless, in real-world scenarios, recommending a compact set of members that can undertake the same tasks would be preferable due to the limited hiring budget.

In this paper, we thoroughly address the aforementioned limitations and propose a novel graph neural network (GNN) framework based on clustering, namely Genius, to efficiently recommend effective new subteams for *subteam replacement*. Specifically, given the team social network within which the teams are embedded, the proposed Genius framework first utilizes a GNN-backboned team representation learner to encode the knowledge of collaboration structures and skills into member-level embeddings, which will be employed for optimal *subteam replacement*. Apart from generalizing Genius to all message-passing GNNs, we also designate the first structurally optimized team-level attention GNNs (*TAGs*) for *subteam replacement* to foster more effective team embeddings. To further improve representation learning, we design a self-supervised *positive team contrasting* training scheme where the key idea is to enforce the similarity between the subteam representation and that of the corresponding team. For optimization, we aim to minimize the disparity between the recommended new subteam and the original team in both collaboration structures and skills. We further propose a novel clustering-based method, which (1) can speed up the *subteam replacement* by pruning the unqualified candidates, and (2) allows flexible sizes of recommended subteams without harming its performance to meet the requirements of cases where new teams of smaller sizes are expected. To summarize, our main contributions are three-fold:

- **Problem:** we expand upon prior work on the *subteam replacement* problem [21] and extend its scope to include replacements of flexible sizes. We formally define the flexible-sized *subteam replacement* problem in Section 2.
- **Algorithms and Analysis:** to the best of our knowledge, we are the first to introduce a trainable framework Genius for team member replacement problems. We also put forward the first structurally optimized GNNs specially cultivated for team social networks, referred to as *TAGs*.
- **Empirical Evaluations:** we conduct extensive experiments w.r.t. multiple graph similarity metrics to fully demonstrate the superiority of our model over the state of the art.

2 Problem Definition

Subteam replacement aims to find a set of new subteam members to replace the unavailable members in the original team so that the new team can perform the same functionalities as the original team. In this paper, the entire team social network, within which the teams are embedded, can be denoted as either the set of all its members $\mathcal{G}$ or as the combination of its adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and the node attribute matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. Here, n represents the size of the team social network, and d stands for the number of node attributes. The original team of a group of individual members is represented by $\mathcal{T}$, and the subteam to be replaced is denoted as $\mathcal{R}$. Different from the existing methods that find the same number of new members [21], we investigate the flexible-sized replacement by recommending new team members $\mathcal{S}$ where $|\mathcal{S}| \leq |\mathcal{R}|$ from the residual individuals of $\mathcal{G}$ (i.e., $\mathcal{M} = \mathcal{G} \setminus \mathcal{T}$), which is more applicable in real-world scenarios (e.g., limited hiring budget).

We formally define the flexible-sized *subteam replacement* problem as follows:

Problem 1. ***Subteam Replacement***

Given: *(1) a team social network $\mathcal{G} = \{\mathbf{A}, \mathbf{X}\}$; (2) the original team $\mathcal{T}$; (3) the subteam of individuals to be replaced $\mathcal{R} \subset \mathcal{T}$.*

Find: *a subteam replacement model, which is capable of capturing the team-level knowledge to replace the unavailable subteam $\mathcal{R}$ by recommending a new set of members $\mathcal{S}$ of flexible size where $|\mathcal{S}| \leq |\mathcal{R}|$. Ideally, the new team should be similar to the original team from the perspectives of collaboration structures and skills. Formally, the objective of the problem can be defined as the maximization of* $\text{GraphSim}(\mathcal{T}, \mathcal{T} \setminus \mathcal{R} \cup \mathcal{S})$*, where* GraphSim *quantifies the graph-level similarity between the two input sets.*

3 Proposed Approach

In this section, we introduce the details of our proposed framework, Genius, for flexible-sized *subteam replacement*. To be specific, Genius addresses the limitations of existing methods by integrating a **G**NN-backboned **EN**coder for with**I**n-cl**US**ter subteam search. We present the two major building blocks of Genius, including (1) a *team network encoder* that captures the essential team social network knowledge to fulfill the requirements of structural and skill match (i.e., the *training* phase), as well as *TAGs*, a novel GNN structure

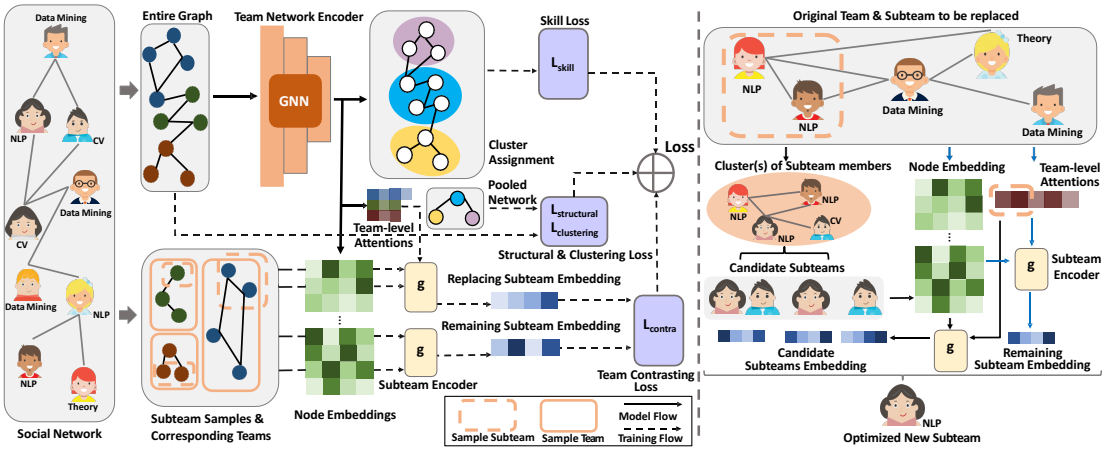


Figure 1: Overview of the GENIUS Framework. (Left) The data flow of the *team network encoder* and its training procedure. (Right) Within-cluster search performed by the *subteam recommender*.

cultivated for team social networks, in Section 3.1, and (2) a *subteam recommender* for recommending new subteam members to replace the unavailable subteam members (i.e., the *inference* phase) in Section 3.2. The superiority of GENIUS in capturing key correlations among node attributes, as compared to graph kernel-based methods, is comprehensively analyzed in Section 3.3. An overview of the proposed GENIUS framework is provided in Figure 1.

3.1 Team Encoder The general idea of *subteam replacement* is to select new subteam members that possess similar characteristics as the original team members in terms of collaborations and skills. Therefore, the quality of the replacement is considerably dependent on the informativeness of the learned representations of the network. We propose a *team network encoder* composed of two sub-components, including (1) a GNN-based team representation learner to capture the correlations between subteams and their corresponding original teams, and (2) a clustering layer to generate node cluster assignments for efficient member recommendation.

To construct a high-quality *team network encoder*, specifically, we exploit GNNs [11, 15] to map each node in the team social network to a low-dimensional latent space. Specifically, we stack multiple GNN layers in order to extract the long-range node dependencies in the network, which can be represented as $\mathbf{H}^1 = \text{GNN}^1(\mathbf{A}, \mathbf{X}), \ldots, \mathbf{H}^L = \text{GNN}^L(\mathbf{A}, \mathbf{H}^{L-1})$, where $\mathbf{H}^L$ denotes the node embedding matrix at the L-th GNN layer. To alleviate the over-smoothing issue in GNNs (i.e., all nodes have similar embeddings) [5], we obtain the final node representations by concatenating the intermediate embeddings from each GNN layer [26]:

$$\mathbf{Z} = \text{concat}(\mathbf{H}^{(1)}, ..., \mathbf{H}^{(L)}), \tag{3.1}$$

where $\mathbf{Z} \in \mathbb{R}^{n \times d'}$ denotes learned node embeddings from the *team network encoder* and encodes the topological and attribute information of the team social network, with d' representing the dimension for the node embeddings. In *subteam replacement*, the two fundamental principles require that the recommended new subteam members have (1) a close collaboration relationship (i.e., structural match), and (2) similar attributes (i.e., skill match), as the original team members. Through the proposed GNN-backboned encoder, we can quantitatively evaluate whether the candidate subteams are well-qualified in the embedding space by assessing the similarity between the corresponding vector representations. It is worth noting that the *team network encoder* is compatible with any standard GNN-based architecture [7, 12, 15, 24], and here we apply Graph Convolutional Networks (GCNs) [15] in our implementation.

Furthermore, motivated by the principles in *subteam replacement*, we speculate that the recommended subteam members and team members with similar representations in the embedding space should form clusters. Hence, we exploit a clustering layer to group members in the team social network according to the generated representations. Specifically, the clustering layer first performs a non-linear transformation on the embedding matrix (i.e., $\mathbf{Z}$) as

$$\mathbf{E} = \sigma(\mathbf{Z}\mathbf{W}_c), \tag{3.2}$$

where $\mathbf{W}_c \in \mathbb{R}^{d' \times c}$ is the learnable weight matrix and c is the number of clusters. σ is the ReLU activation function. The transformed matrix will be used to generate a (soft) cluster assignment matrix $\mathbf{C} \in \mathbb{R}^{n \times c}$ where the elements of the corresponding row i are computed as follows:

$$\mathbf{C}[i, m] = \frac{e^{\mathbf{E}[i,m]}}{\sum_{j=1}^{c} e^{\mathbf{E}[i,j]}}, \tag{3.3}$$

where $m \in \{1, \ldots, c\}$, $\mathbf{C}[i, :]$ is the cluster assignment vector for node v_i.

Having obtained the cluster assignment matrix, we further compute the (hard) assignment vector $\mathbf{h} \in \mathbb{R}^n$, where $\mathbf{h}[i] = \text{argmax}_{j\in\{1,\ldots,c\}} \mathbf{C}[i,j]$, and we can construct the cluster container $\mathcal{K}$ as $\mathcal{K}[m] = \{v_i|\mathbf{h}[i] = m, i \in \{1,2,\ldots,n\}\}, m \in \{1,2,\ldots,c\}$, which will be further utilized for generating candidates.

The clustering layer offers two advantages. First, it ensures that nodes with similar embeddings are grouped together, aligning with the requirements for structural and skill match in *subteam replacement*. Second, the clustering results can be leveraged to significantly speed up the replacements because they largely reduce the search space of *subteam recommender* by pruning the unqualified members, as illustrated in Section 3.2.

Team-level Attention GNNs (*TAGs*). Members have various levels of contributions to the team's performance, and to capture their importance to the team, an attention mechanism is applied. Specifically, we propose a novel GNN structure with team-level attention, namely *TAGs*, which generates attention vectors along with team member embeddings. *TAGs* learn a vector set $\mathcal{A} = \{\mathbf{a}_\mathcal{F} \in \mathbb{R}^{|\mathcal{F}|}|\mathcal{F} \in \mathcal{P}\}$, where $\mathcal{P}$ refers to the set of all teams $\mathcal{F}$ within the team social network $\mathcal{G}$. To maintain uniformity, we apply similar operations to the vectors as Equation (3.3), $\mathbf{a}_\mathcal{F}[t] = \frac{e^{\mathbf{a}_\mathcal{F}[t]}}{\sum_{i=1}^{|\mathcal{F}|} e^{\mathbf{a}_\mathcal{F}[i]}}$.

Training. The objective of our proposed GENIUS framework is to recommend well-qualified new subteam members that are close to the members of existing teams, based on the learned node representations. To navigate the model training, we first propose a self-supervised contrasting scheme, namely *positive team contrasting*, to enhance the node representation learning. In addition, we introduce the training objectives for satisfying both structural and skill match in *subteam replacement*.

Positive team contrasting. Contrastive learning aims to empower representation learning by maximizing the agreement between similar examples in each pair [6]. Intuitively, a subteam as well as its replacement should be similar to the original team in terms of collaboration structures and skills because they work on the same and well-specified task. To address this, we propose *positive team contrasting* to improve team-level representation learning through self-contrasting. The key idea is to minimize the disparity between the embedding of the recommended subteam (i.e., $\mathcal{R}$) and that of the residual subteam (i.e., $\mathcal{T} \setminus \mathcal{S}$).

We define the team-level representation (e.g., $\mathcal{T}$) as the weighted aggregation of the members' embeddings:

$$g(\mathcal{T}) = \sum_{t\in\mathcal{T}} w_t \mathbf{Z}[t,:], \qquad \sum_{t\in\mathcal{T}} w_t = 1, \tag{3.4}$$

where w_t denotes the weights for aggregation. In this paper, we set the weight w_t as $\frac{1}{|\mathcal{R}|}$ for standard GNNs, and $\mathbf{a}_\mathcal{R}[t]$ when deploying *TAGs*.

Having obtained the team embeddings from Equation (3.4), we can now define the *team contrasting loss* as follows:

$$L_{\text{contra}} = \frac{1}{s}\sum_{i=1}^{s} \text{Diff}(g(\mathcal{R}_i), g(\mathcal{T}_i \setminus \mathcal{R}_i)), \tag{3.5}$$

where s is the number of sample teams, $\mathcal{R}_i$ denotes the sample subteam to be replaced in the i-th sample team (i.e., $\mathcal{T}_i$) from the given social network $\mathcal{G}$. $\text{Diff}(\cdot,\cdot)$ calculates the first-order norm of the difference between the two input vectors. L_{contra} calculates the average difference between the replaced subteams and the residual subteams. By minimizing the *team contrasting loss*, we enforce the matching among team-level embeddings regarding collaboration structures and skills.

According to the two fundamental principles in designing *subteam replacement* algorithms, i.e., structural and skill match, we further define the *skill loss*, *structural loss*, and *clustering loss* to optimize the clustering layer. We specify these training objectives as follows.

Skill loss. The nodes within the same cluster should exhibit similarities in skills. We develop a *skill loss* function to train the model to achieve this objective. The basic idea is to enhance the pairwise similarity among skill vectors (i.e., node attributes) in relation to their cluster assignments. We first apply a pairwise similarity measure to the attribute matrix (i.e., $\mathbf{X}$) and the cluster assignment matrix (i.e., $\mathbf{C}$) as $\mathbf{Y}_1 = \text{PairSim}(\mathbf{X},\mathbf{X}), \mathbf{Y}_2 = \text{PairSim}(\mathbf{C},\mathbf{C})$, where $\text{PairSim}(\mathbf{P},\mathbf{Q}) = \hat{\mathbf{P}}\hat{\mathbf{Q}}^T$ for $\mathbf{P},\mathbf{Q} \in \mathbb{R}^{n\times d}$, and $\hat{\mathbf{P}}$ and $\hat{\mathbf{Q}}$ denote the row-normalized $\mathbf{P}$ and $\mathbf{Q}$, respectively. The matrices $\mathbf{Y}_i, i \in \{1,2\}$, represent the similarity between all pairs of nodes regarding node attributes and cluster assignments, respectively. Specifically, $\mathbf{Y}_i(j,k), i \in \{1,2\}, j,k \in \{1,2,\ldots,n\}$, denotes the corresponding similarity between node j and k. We then compare the two matrices $\mathbf{Y}_1$ and $\mathbf{Y}_2$ using PairSim as $\mathbf{Y}_3 = \text{PairSim}(\mathbf{Y}_1,\mathbf{Y}_2)$, where $\mathbf{Y}_3$ encodes the closeness between the similarity measures on node attributes and cluster assignments. Therefore, by aggregating the diagonal elements of $\mathbf{Y}_3$, where $\mathbf{Y}_3(i,i), i \in \{1,2,\ldots,n\}$, denotes the similarity between the node attributes and the cluster assignment of node i, we can obtain the skill similarity loss from the clustering results as follows,

$$L_{\text{skill}} = -\text{tr}(\mathbf{Y}_3), \tag{3.6}$$

where $\text{tr}(\cdot)$ denotes the trace of the input matrix. Through minimizing L_{skill}, the clustering layer is optimized to assign nodes with similar skills (i.e., high attribute similarity) to the same cluster.

Structural loss. Nodes that are topologically closer should be more likely to be clustered together. Inspired by previous studies on supervised node clustering [29],

we exploit the *structural loss* $L_{\text{structural}}$ to encode structural similarity for the cluster assignments. Specifically, we define the objective as follows:

$$L_{\text{structural}} = \|\mathbf{A} - \mathbf{C}\mathbf{C}^T\|_F, \tag{3.7}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

Clustering loss. A cluster assignment vector with approximately equal values (i.e., a near-uniform distribution) implies a high degree of uncertainty in the corresponding node's cluster assignment. Therefore, it is preferable for cluster assignment vectors to closely resemble one-hot vectors. We thus define the *clustering loss* as:

$$L_{\text{clustering}} = \frac{1}{n}\sum_{i=1}^{n} \text{Entropy}(\mathbf{C}[i,:]), \tag{3.8}$$

where Entropy denotes the entropy function, which is calculated as $\text{Entropy}(\mathbf{C}[i,:]) = -\sum_{j=1}^{c} \mathbf{C}[i,j]\log \mathbf{C}[i,j]$,

We combine the aforementioned objectives as

$$L = L_{\text{contra}} + b_1 L_{\text{skill}} + b_2 L_{\text{structural}} + b_3 L_{\text{clustering}}, \tag{3.9}$$

where L represents the total loss and $b_i, i \in \{1,2,3\}$ are the regularization parameters that balance the scales of training objectives for different team social networks. L is applied to guide the training of the entire GENIUS framework. The four losses prevent the overfitting of one another.

3.2 Subteam Recommender Having the proposed *team network encoder* model, we are equipped with effective representations of teams and their members. Given an original team $\mathcal{T}$ and the subteam to be replaced $\mathcal{R} \subset \mathcal{T}$, we propose a *subteam recommender* that performs an efficient within-cluster search based on subteam representation (Algorithm 1), where g is the predefined function (Equation (3.4)) for calculating subteam embeddings, and f computes the cosine similarity. Set is a deduplication function for containers, and Product generates Cartesian products on the assigned clusters. When calculating new subteam embedding $\mathbf{n}$ in line 8, $g' = g$ for traditional GNNs, and $g'(\mathcal{E}) = \alpha * \sum_{t\in\mathcal{E}} \mathbf{Z}[t,:]$, where $\alpha = \sum_{t\in\mathcal{R}} \mathbf{a}_{\mathcal{R}}[t]$ when deploying *TAGs*.

Subteam recommender generates candidates only from clusters where subteam members to be replaced are assigned. Compared to generating candidates across the entire dataset, this pruning technique significantly improves the efficiency of the algorithm (Lemma 1). A stable number of team members within clusters, implying a modest candidate search space, is achieved by adjusting the number of clusters c in accordance with the size of the team social network n. This effectively maintains the efficiency of the *subteam recommender* as team social networks scale. Additionally, the proposed GENIUS framework ensures that the optimal solution is included in the candidate new subteams $\mathcal{Q}$ since nodes with similar embeddings will be clustered together and thus, it is valid to prune unqualified candidates through clustering.

Algorithm 1 Subteam Recommender

Input: (1) original team $\mathcal{T}$; (2) a subteam to be replaced $\mathcal{R}$; (3) cluster hard assignment vector $\mathbf{h}$; (4) node embedding matrix $\mathbf{Z}$; (5) residual social network $\mathcal{M}$; (6) clusters $\mathcal{K}$.

Output: Optimal new subteam $\mathcal{S} \subset \mathcal{M}$, $|\mathcal{S}| \le |\mathcal{R}|$.

1: Initialize similarity $y = 0$, $\mathcal{S} = \{\}$
2: Compute residual subteam embedding $\mathbf{r} \leftarrow g(\mathcal{T} \setminus \mathcal{R})$
3: Compute clusters assigned to $\mathcal{R}$, $\mathbf{c} \leftarrow \mathbf{h}[\mathcal{R}]$
4: Generate candidate subteams $\mathcal{Q} \leftarrow \text{Product}(\mathcal{K}[\mathbf{c}])$
5: **for** each candidate $\mathcal{D}$ **in** $\mathcal{Q}$ **do**
6: Remove original team $\mathcal{T}$ members $\mathcal{E} \leftarrow \mathcal{D} \setminus \mathcal{T}$
7: Deduplicate candidate $\mathcal{E} \leftarrow \text{Set}(\mathcal{E})$
8: Compute new subteam embedding $\mathbf{n} \leftarrow g'(\mathcal{E})$
9: Compute similarity $t \leftarrow f(\mathbf{r}, \mathbf{n})$
10: **if** $t > y$ **then**
11: Update $\mathcal{S} \leftarrow \mathcal{E}$, $y \leftarrow t$
12: **end if**
13: **end for**

Lemma 1. *Within-cluster search is $O(c^{|\mathcal{R}|})$ more efficient in time than searching over the entire dataset of size n. The values are dependent on datasets, and the exact figures can be referred to in Table 1 and Section 4. Typically, $c \approx 1e3$, $|\mathcal{R}| \approx 5$, and $n \approx 2e4$.*

Lemma 2. *The search algorithm is guaranteed to produce $\mathcal{S}$ where $|\mathcal{S}| \le |\mathcal{R}|$ by the nature of the Cartesian product.*

3.3 Analysis Instead of being limited by a fixed set of genres, nodes with more specialized descriptions are becoming dominant these days, highlighting the limitations of graph kernel-based methods in both *effectiveness* and *efficiency* as compared to GENIUS.

Effectiveness. As the number of node attributes increases, each node attribute dimension no longer represents a general class (i.e., coarse-grained attributes) where little correlation exists among different dimensions. This remarkably increases the difficulty for graph kernel-based methods to capture the potential correlations among various node attributes.

Given two labeled graphs $\mathbf{G}_i = \{\mathbf{A}_i, \mathbf{L}_i\}, i = 1,2$, random walk graph kernel [3, 13, 19] is computed as $\text{Ker}(\mathbf{G}_1, \mathbf{G}_2) = \mathbf{y}(\mathbf{I} - a\mathbf{A}_\times)^{-1}\mathbf{L}_\times\mathbf{x}$, where $\mathbf{A}_\times = \mathbf{L}_\times(\mathbf{A}_1 \otimes \mathbf{A}_2)$, $\otimes$ representing the Kronecker product of two matrices, $\mathbf{x}$ and $\mathbf{y}$ represent starting and stopping probability vectors for the random walk, which are usually set as uniform, and a is a decay factor. $\mathbf{L}_\times$ is a diagonal matrix calculated as

$$\mathbf{L}_\times = \sum_{i=1}^{d} \text{diag}(\mathbf{L}_1(:,i)) \otimes \text{diag}(\mathbf{L}_2(:,i)), \tag{3.10}$$

Table 1: Statistics of evaluation datasets.

Datasets	# nodes	# edges	# features	# teams
DBLP	26,351	54,044	7,551	7,052
IMDB	13,472	665,166	2,040	818

where diag represents the diagonalization operation. It represents matches of labels (i.e., node attributes) among nodes.

Random walk graph kernel bases on paths. The $\mathbf{L}_\times$ term in Equation (3.10) eliminates paths between nodes where labels are different, regardless of the extent they vary. Most importantly, random walk graph kernel fails to recognize correlations among different node attributes. Although some previous study includes skill pair matrix in the algorithm [21], the essence of those calculations is still a rough match of skill vectors. For instance, in the DBLP dataset, the attributes *pattern recognition* and *feature extraction* are strongly correlated, which isn't the case with *security*. However, current methods, by the nature of the Kronecker product, inaccurately equate such relationships, leading to less precise team member replacements.

Genius, on the other hand, is capable of handling graphs with massive node attributes. By including sample teams in training, the model learns to identify correlations among node attributes in team social networks and make optimal replacement choices.

Efficiency. With the remaining parameters (network size n, team size, etc.) held constant and given d node attributes, graph kernel-based methods have an optimal time complexity of $O(d^2)$ [21], whereas the time complexity of Genius is $O(d)$. It is also worth stressing that with the entire dataset fed into *team network encoder* once, all teams and their subteams to be replaced can be directly passed into *subteam recommender*, which significantly reduces the amortized complexity of Genius to a very low level.

Lemma 3. *The amortized time complexity for the training of the* team network encoder *is in* $O(\frac{nET}{2^n})$*, where* E *denotes the number of training epochs, and* T *represents the time spent in each epoch.*

Proof. The time complexity for training a *team network encoder* is in $O(nET)$, and the total number of replacement is 2^n. Thus, the amortized complexity for training is in $O(\frac{nET}{2^n})$. □

4 Experiments

In this section, we perform empirical evaluations to demonstrate the effectiveness and efficiency of Genius.

4.1 Experimental Setup

Datasets. We utilize two widely employed public real-world datasets, DBLP and IMDB, which have consistently remained the predominant and sole choices in prior research [19,21]. We follow similar experimental setups, albeit with around 150× more abundant node attributes. Table 1 summarizes the statistics of each dataset. Detailed descriptions can be found as follows.

- **DBLP**[1] provides computer science bibliographic information. We build a graph where nodes represent authors, edges represent co-authorship, and node attributes are each author's paper keywords. Teams refer to the authors of each paper.
- **IMDB**[2] contains statistics of actors and movies, where nodes represent actors, edges represent the number of movies where actors co-starred, and node attributes are tags of movies in which actors played. Teams are represented by crews of actors in a movie.

Comparison Methods. We compare Genius with two categories of baseline methods for solving *subteam replacement*, including (1) *graph kernel*-based method [20, 21], which prior investigations take as the optimal solution, and (2) *supervised encoder*-based method [10], which the capability of Genius to learn team- and member- level representations is compared to. To validate the superiority of *TAGs*, we compare the performance of Genius with standard GNNs and that with *TAGs*, namely *TAG*-Genius. By default, Genius in this section refers to that with standard GNNs.

Implementation Details. For each dataset, 60% of the teams are selected as training samples (number of sample teams $s = 0.6|\mathcal{P}|$), 20% are for validation while the remaining 20% (labeled as $\mathcal{Z}, \mathcal{Z} \subset \mathcal{G}$) are for testing. Sample subteams $\mathcal{R}_i$ are randomly selected from sample teams $\mathcal{T}_i$, where $i \in \{1, 2, \ldots, s\}$. For balancing parameters in Equation (3.9), b_1, b_2, and b_3 are set as $100, 100$, and 10 respectively for both DBLP and IMDB to scale the losses to a uniform level and ensure they are collectively updated for effective optimization.

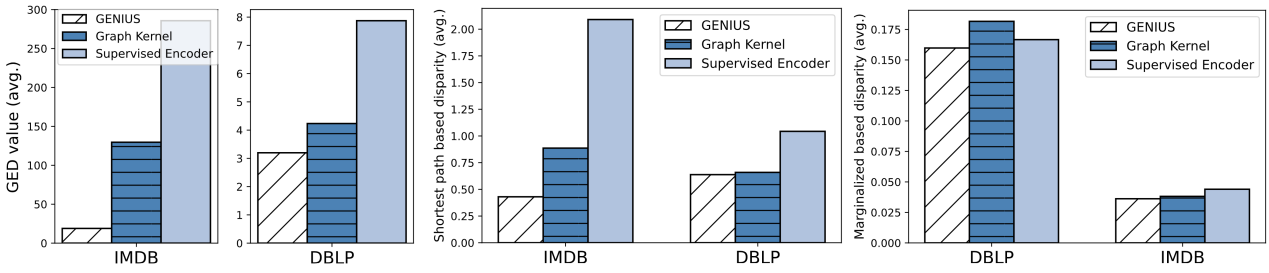
Evaluation Metrics. There is no single golden criterion for quantitatively evaluating the effectiveness of the replacement results. Thus, we examine the disparity between the new teams and the original teams using multiple graph similarity metrics to ensure comprehensive comparisons. In this paper, we employ graph disparity calculated based on three metrics: (1) *Graph Edit Distance (GED)* [1], (2) *Shortest Path Graph Kernel (SP)* [4], and (3) *Marginalized Graph Kernel (MGK)* [14,23]. Lower values indicate a higher similarity between the original teams and the new teams.

4.2 Effectiveness Results

Overall Comparison. We first conduct experiments to examine the general performance of Genius and compare the results w.r.t. baselines with the same set of test cases. The results are visualized in Figure 2. Accordingly, we have the following observations, includ-

[1] http://arnetminer.org/citation

[2] https://grouplens.org/datasets/hetrec-2011/



(a) Disparity based on GED. (b) Disparity based on SP. (c) Disparity based on MGK.

Figure 2: Performance evaluation w.r.t. different datasets.

ing: **(1)** Based on various graph similarity metrics and across different datasets, GENIUS generates results with significantly lower graph disparity w.r.t. original teams: for instanace, SP-based graph disparity for GENIUS in IMDB is only 1/5 of that for the *supervised encoder*-based method (Figure 2b); GED-based graph disparity for GENIUS in IMDB is only 1/15 of that for the *supervised encoder*-based method and 1/7 of that for the *graph kernel*-based method (Figure 2a). **(2)** The graph disparity of baseline models increases drastically when the average team size increases: for instance, on IMDB (average team size is around 17), both baselines generate results with average GED-based graph disparity rising to around 100× higher than that of DBLP (average team size is around 3) while GENIUS has a steady performance (Figure 2a). **(3)** The *supervised encoder*-based method has the worst performance of the three, indicating that training with labels is not appropriate for team recommendation tasks. In short, GENIUS outperforms baselines to a great extent.

Impact of team-level attention. We conduct experiments on the DBLP dataset, and as can be viewed from Figure 3, *TAG*-GENIUS generates new teams of higher similarity to original teams than GENIUS based on all graph similarity metrics.

Performance as the size of the skill set varies. This part addresses our discussion on the limitations of graph kernels as the skill set scales up in Section 3.3. We pick DBLP as the evaluation dataset because it allows a wider range of node attribute variations. The same experiment settings are applied except that different numbers of node attributes are randomly selected from the original dataset, ranging from $50(1\%)$ to $7,551(100\%)$. Both GENIUS and the *graph kernel*-based method are tested with the same set of skills and test cases. Figure 4 visualizes the results, where as the size of the skill set increases, the disparity between the new teams and original teams increases drastically for the *graph kernel*-based method while GENIUS has stable performance. The results confirm our previous statement that the performance of the *graph kernel*-based method is impaired by its inability to recognize potential correlations among various skills.

Table 2: Average time to find a solution in seconds

Methods	**DBLP**	**IMDB**
GENIUS	2.27 (0.04)	11.74 (0.13)
Graph Kernel	1335.20	4432.01
Supervised Encoder	7.32 (7.21)	216.36 (215.87)

* Numbers in brackets (if any) represent inference time

Table 3: Average training time for GENIUS per epoch in seconds w.r.t. the size of skill set (DBLP)

# Node attributes	50	500	3,000	5,400	7,551
Average Training Time	3.59	2.77	3.32	2.93	3.00

Table 4: Average time to find a solution in seconds w.r.t. the size of skill set (DBLP)

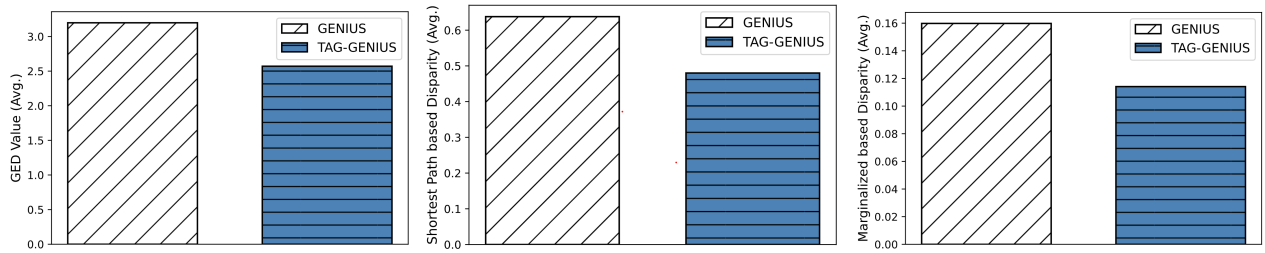
Methods	50	500	3,000	5,400	7,551
GENIUS	2.77 (0.23)	2.12 (0.15)	2.41 (0.05)	2.10 (0.03)	2.17 (0.04)
Graph Kernel	12.40	96.10	570.12	924.34	1335.20

* Numbers in brackets (if any) represent inference time

4.3 Efficiency Results For models that require training (GENIUS and the *supervised encoder*-based method), we take total running time as the sum of training time (total training time for 2000 epochs divided by the number of test samples, i.e., $|\mathcal{Z}|$) and inference time. For the method that does not require training (*graph kernel*-based method), the total running time is taken as the time required to obtain results. We investigate the efficiency results from the following aspects:

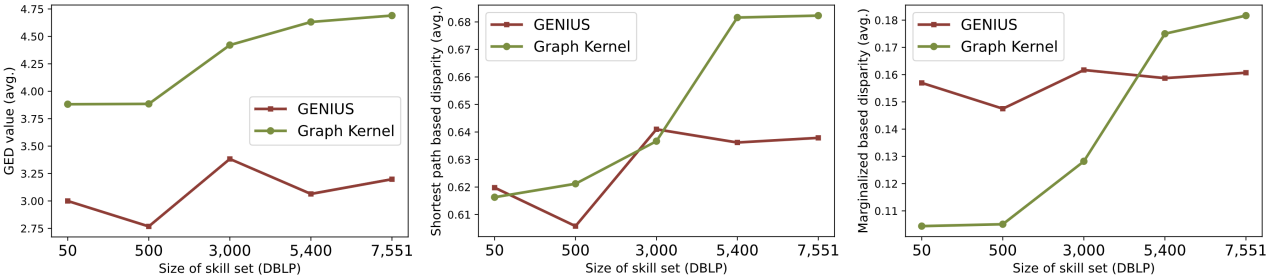
The overall efficiency differences between GENIUS *and baselines (Table 2).* It is obvious that GENIUS performs much faster than the two baselines due to its low amortized time complexity and the crucial improvements brought by the within-cluster search algorithm. GENIUS has around 600× higher efficiency than the *graph kernel*-based method on DBLP and around 400× higher efficiency on IMDB; GENIUS has around 3× higher efficiency than the *supervised encoder*-based method on DBLP and around 20× higher efficiency on IMDB.

Training time differences of GENIUS *as the size of the skill set varies (Table 3).* From the results, we can



(a) Disparity based on GED. (b) Disparity based on SP. (c) Disparity based on MGK.

Figure 3: Performance comparisons between GENIUS and *TAG*-GENIUS.



(a) Disparity based on GED. (b) Disparity based on SP. (c) Disparity based on MGK.

Figure 4: Performance variation w.r.t. size of skill set (DBLP).

observe tiny fluctuations in the average training time per epoch, i.e., 3.12 ± 0.29 seconds, across different numbers of node attributes (coefficient of variation $c_v \approx 9\%$). Thus, although variations in the number of node attributes give rise to different numbers of parameters to be trained, the actual training time per epoch remains within a small range.

Efficiency differences of GENIUS *compared to the graph kernel-based method as the size of the skill set varies (Table 4).* This part addresses our discussion on the inefficiency of graph kernels as the skill set scales up in Section 3.3. We can observe that as the number of node attributes increases, the *graph kernel*-based method takes a drastically increasing amount of time to obtain results. Meanwhile, the average inference time to find a solution using GENIUS drops as the number of node attributes increases. This is because with more detailed skill information, nodes become more scattered, resulting in fewer candidates generated by within-cluster searches. For every size of skill set, GENIUS finds replacements significantly faster than the *graph kernel*-based method.

5 Related Work

We review the related work in terms of (1) team recommendations and (2) graph neural networks.

Team Recommendations. Team recommendation addresses forming effective teams for tasks with constraints. It considers skill matches, team member connectivity [17, 28, 30], and the balance in between [9]. Individual contributions to teams have also been studied [20]. *Team replacement* uses graph kernels [19] or reinforcement learning in dynamic scenarios [33] for single-member replacement. For multiple-member replacements (*subteam replacement*), graph kernels are the primary scheme [21]. However, these methods yield fixed-size teams, potentially problematic in real-world scenarios with budget constraints. Zhou et al. [31] introduce influence functions [16,32] to interpret results from graph kernel-based team recommendation algorithms.

Graph Neural Networks. Various graph neural networks (GNNs) [7, 12, 15, 24] have been developed and widely applied in numerous fields. Within these models, node representations are passed through layers in a consecutive fashion [11]. Each kind of them has its own strengths and performs well in different tasks including social network analysis [12, 15] and graph anomaly detection [8, 22]. Graph convolutional networks (GCNs) [15] originate from the graph spectral theory and generate promising results by capturing information from the entire graph to encode each node. Graph Attention Networks (GAT) [24] is able to learn the relationships of each node with its neighbors, and GraphSAGE [12] successfully deals with unseen nodes. Generalized pagerank GNN (GPRGNN) [7] and Jumping Knowledge Net (JKNet) [26] further solve the problem of over-smoothing. Existing GNNs mainly focus on the member level [2, 15, 27], while *subteam replacement* requires effective team-level representations. We introduce *TAGs* to address this need.

6 Conclusion

In this paper, we investigate the challenging problem of *subteam replacement*. To tackle this problem, we propose a novel framework GENIUS, which incorporates a GNN-backboned *team network encoder* that learns

essential team-level representations with *positive team contrasting*, a training technique specifically developed for team-related problems. Besides generalizing GENIUS for all popular GNNs, we also develop *TAGs*, the first GNN structure designated for team social networks. To further improve the efficacy of GENIUS, members with similar skills and collaboration structures are clustered together, by which a *subteam recommender* performs a within-cluster search that prunes unqualified candidates. Through extensive experiments, we demonstrate the superiority of GENIUS over existing methods. Our model can be further extended to multiple problems, including subgraph matching in large team social network datasets, subgraph similarities, etc.

Acknowledgement

This work is supported by NSF (1947135), NIFA (2020-67021-32799), and DHS (17STQAC00001-07-00), The content of the information in this document does not necessarily reflect the position or the policy of the Government or Amazon, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] Z. Abu-Aisheh, R. Raveaux, J.-y. Ramel, and P. Martineau, "An Exact Graph Edit Distance Algorithm for Solving Pattern Recognition Problems," 2015.

[2] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," 2017.

[3] K. Borgwardt, N. Schraudolph, and S. Vishwanathan, "Fast computation of graph kernels," 2006.

[4] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," 2005.

[5] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," 2019.

[6] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020.

[7] E. Chien, J. Peng, P. Li, and O. Milenkovic, "Adaptive universal generalized pagerank graph neural network," 2020.

[8] K. Ding, Q. Zhou, H. Tong, and H. Liu, "Few-shot network anomaly detection via cross-network meta-learning," 2021.

[9] C. Dorn and S. Dustdar, "Composing near-optimal expert teams: A trade-off between skills and connectivity," 2010.

[10] X. Fu, J. Zhang, Z. Meng, and I. King, "MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding," 2020.

[11] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," 2017.

[12] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2017.

[13] U. Kang, H. Tong, and J. Sun, "Fast random walk graph kernel," 2012.

[14] H. Kashima, K. Tsuda, and A. Inokuchi, "Marginalized kernels between labeled graphs," 2003.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016.

[16] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," 2017.

[17] T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," 2009.

[18] L. Li and H. Tong, *Computational Approaches to the Network Science of Teams*, 2020.

[19] L. Li, H. Tong, N. Cao, K. Ehrlich, Y.-R. Lin, and N. Buchler, "Replacing the irreplaceable: Fast algorithms for team member recommendation," 2015.

[20] L. Li, H. Tong, Y. Wang, C. Shi, N. Cao, and N. Buchler, "Is the whole greater than the sum of its parts?" 2017.

[21] Z. Li, X. Pi, M. Wu, and H. Tong, "Reform: Fast and adaptive solution for subteam replacement," 2021.

[22] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A comprehensive survey on graph anomaly detection with deep learning," 2021.

[23] P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert, "Extensions of marginalized graph kernels," 2004.

[24] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2018.

[25] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," 2010.

[26] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," 2018.

[27] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," 2016.

[28] X. Yin, C. Qu, Q. Wang, F. Wu, B. Liu, F. Chen, X. Chen, and D. Fang, "Social connection aware team formation for participatory tasks," *IEEE Access*, 2018.

[29] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," 2018.

[30] A. ZAKARIAN and A. KUSIAK, "Forming teams: an analytical approach," 1999.

[31] Q. Zhou, L. Li, N. Cao, N. Buchler, and H. Tong, "Extra: Explaining team recommendation in networks," 2018.

[32] Q. Zhou, L. Li, N. Cao, L. Ying, and H. Tong, "Adversarial attacks on multi-network mining: Problem definition and fast solutions," 2021.

[33] Q. Zhou, L. Li, and H. Tong, "Towards real time team optimization," 2019.