Replacing the Irreplaceable: Fast Algorithms for Team Member Recommendation

Liangyue Li
Arizona State University
liangyue@asu.edu

Hanghang Tong
Arizona State University
Hanghang.Tong@asu.edu

Nan Cao
IBM Research
nancao@us.ibm.com

Kate Ehrlich
IBM Research
katee@us.ibm.com

Yu-Ru Lin
University of Pittsburgh
yurulin@pitt.edu

Norbou Buchler
US Army Research Laboratory
norbou.buchler.civ@mail.mil

ABSTRACT

In this paper, we study the problem of Team Member Replacement : given a team of people embedded in a social network working on the same task, find a good candidate who can fit in the team after one team member becomes unavailable. We conjecture that a good team member replacement should have good *skill matching* as well as good *structure matching*. We formulate this problem using the concept of graph kernel. To tackle the computational challenges, we propose a family of fast algorithms by (a) designing effective pruning strategies, and (b) exploring the smoothness between the existing and the new team structures. We conduct extensive experimental evaluations on real world datasets to demonstrate the effectiveness and efficiency. Our algorithms (a) perform significantly better than the alternative choices in terms of both precision and recall; and (b) scale *sub-linearly*.

Objective

Alg.

1. INTRODUCTION

In his world-widely renowned book “The Science of the Artificial” [37], Nobel laureate Herbert Simon pointed out that it is more the complexity of the *environment*, than the complexity of the individual persons, that determines the complex behavior of humans. The emergence of online social network sites and web 2.0 applications provides a new connected environment/context, where people interact and collaborate with each other as a team to collectively perform some complex tasks.

Among others, the churn of team members is a common problem across many application domains. To name a few, an employee in a software or sales team might decide to leave the organization and/or be assigned to a new task. In a law enforcement mission, a SWAT team might lose certain task force due to the fatality or injury. In professional sports (e.g., NBA), the rotation tactic between the benches could play a key role on the game outcome. In all these cases, the loss of the key member (i.e., the irreplaceable) might bring the catastrophic consequence to the team performance. How can we find the best alternate when a team member becomes unavailable? However, despite the frequency with which people leave a team before a project/task is complete and the resulting disruption [43], replacements are often found opportunistically and are not necessarily optimal.

Background

We conjecture there will be less disruption when the team member who leaves is replaced with someone with similar relationships with the other team members. This conjecture is inspired by some recent research which shows that team members prefer to work with people they have worked with before [21] and that distributed teams perform better when members know each other [9]. Furthermore, research has shown that specific communication patterns amongst team members are critical for performance [7]. Thus, in addition to factors such as skill level, maintaining the same or better level of familiarity and communication amongst team members before and after someone leaves should reduce the impact of the departure. In other words, for team member replacement, the similarity between individuals should be measured in the context of the team itself. More specially, a good team member replacement should meet the following two requirements. First (*skill matching*), the new member should bring a similar skill set as the current team member to be replaced. Second (*structure matching*), the new member should have a similar network structure as the current team member in connecting the rest team members.

Reason

Armed with this conjecture, we formally formulate the Team Member Replacement problem by the notation of graph similarity/kernel. By modeling the team as a labeled graph, graph kernel/similarity provides a natural way to capture both the skill and structural match. However, for a network with n individuals, a straight-forward method would require $O(n)$ graph kernel computations for one team member replacement, which is computationally intractable. For example, for the *DBLP* dataset with almost 1M users (i.e., $n \approx 1,000,000$), we found that it would take 6,388s to find one replacement for a team of size 10. To address the computational challenges, we propose a family of fast algorithms by carefully designing the pruning strategies and exploring the smoothness between the existing and the new teams. We perform the extensive experimental evaluations to demonstrate the effectiveness and efficiency of our methods. Specially, we find that (1) by encoding both the skill and structural matching, it leads to a much better replacement result. Compared with the best alternative choices,

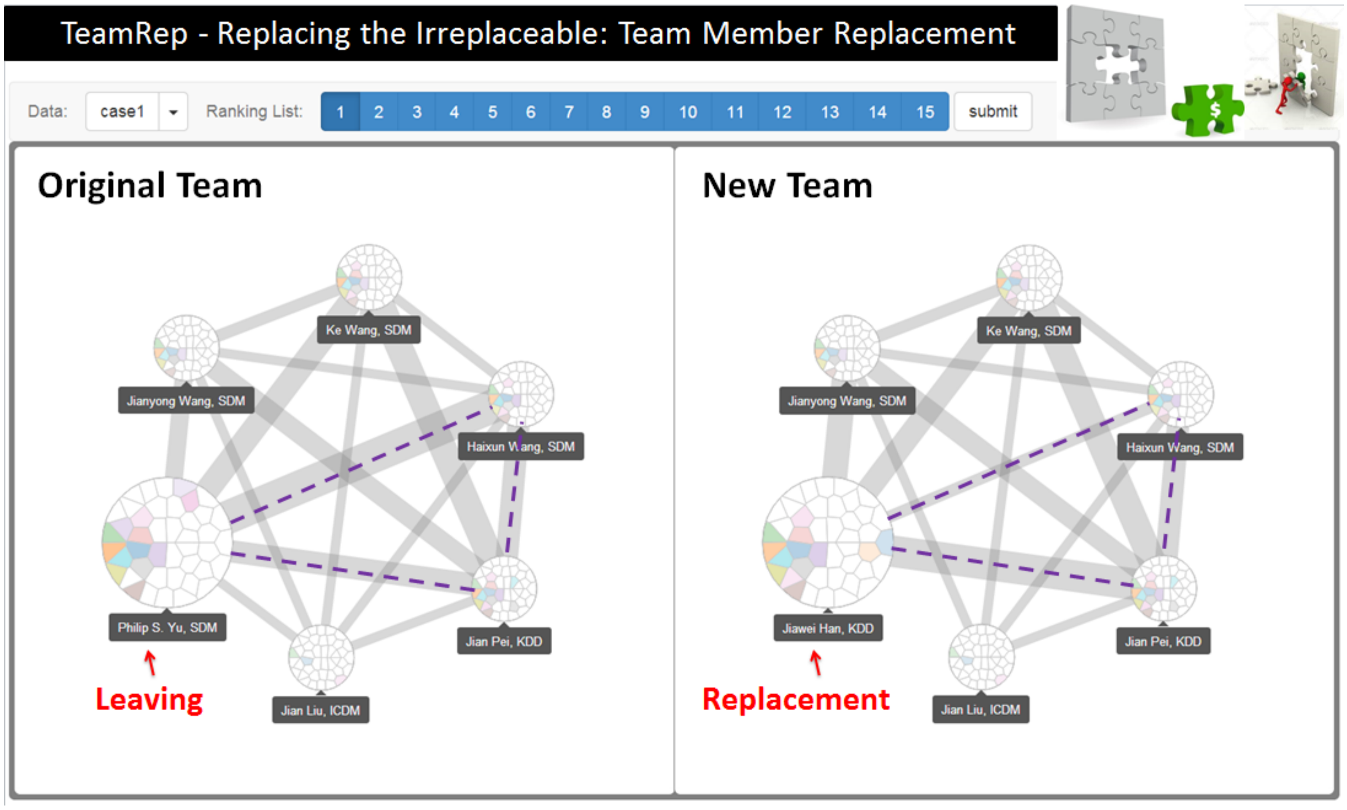

Figure 1: The team graphs before and after *Jiawei Han* takes *Philip S. Yu*'s place. See subsection 5.2 for detailed explanations.

we achieve 27% and 24% *net increase* in average recall and precision, respectively; (2) our fast algorithms are orders of magnitude faster and scale *sub-linearly*. For example, our pruning strategy alone leads up to 1,709× speed-up, without sacrificing any accuracy.

The main contributions of this paper are as follows.

1 **Problem Definitions.** We formally define the TEAM MEMBER REPLACEMENT problem, to recommend a good candidate after a team member is unavailable in the context of networks.

2 **Algorithms and Analysis.** We propose a family of effective and scalable algorithms for TEAM MEMBER REPLACEMENT ; and analyze its correctness and complexity.

3 **Experimental Evaluations**. We perform extensive experiments on real world datasets, to validate the effectiveness and efficiency of our methods. (See an example in Figure 1.)

The rest of the paper is organized as follows. Section 2 formally defines TEAM MEMBER REPLACEMENT problem. Section 3 presents our basic solutions. Section 4 addresses the computational challenges. Section 5 presents the experimental results. Section 6 reviews some related work. Section 7 concludes the paper.

2. PROBLEM DEFINITIONS

Table 1 lists the main symbols used throughout this paper. We describe the n individuals by an labelled social network $\mathbf{G} := \{\mathbf{A}, \mathbf{L}\}$, where $\mathbf{A}$ is an $n \times n$ adjacency matrix characterizing the connectivity among different individuals; and

Table 1: Table of symbols

Symbols	Definition
$\mathbf{G} := \{\mathbf{A}, \mathbf{L}\}$	the entire social network
$\mathbf{A}_{n\times n}$	the adjacency matrix of $\mathbf{G}$
$\mathbf{L}_{n\times l}$	skill indicator matrix
$\mathcal{T}$	the team member index
$\mathbf{G}(\mathcal{T})$	the team network indexed by its members $\mathcal{T}$
d_i	the degree of the i^{th} node in $\mathbf{A}$
l	the total number of skills
t	the team size, i.e., $t = \lvert\mathcal{T}\rvert$
n	the total number of individuals in $\mathbf{A}$
m	the total number of connections in $\mathbf{A}$

Def.

$\mathbf{L}$ is $n \times l$ skill indicator matrix. The i^{th} row vector of $\mathbf{L}$ describes the skill set of the i^{th} individual. For example, suppose there are only three skills in total, including {*data mining, databases, information retrieval*}. Then an individual with a skill vector $[1, 1, 0]$ means that s/he has both *data mining* and *databases* skills but no skill in terms of *information retrieval*. Also, we represent the elements in a matrix using a convention similar to Matlab, e.g., $\mathbf{A}(i, j)$ is the element at the i^{th} row and j^{th} column of the matrix $\mathbf{A}$, and $\mathbf{A}(:, j)$ is the j^{th} column of $\mathbf{A}$, etc.

We use the calligraphic letter $\mathcal{T}$ to index the members of a team, which includes a subset of $t = |\mathcal{T}|$ out-of n individuals. Correspondingly, we can represent the team by another labelled team network $\mathbf{G}(\mathcal{T}) := \{\mathbf{A}(\mathcal{T}, \mathcal{T}), \mathbf{L}(\mathcal{T}, :)\}$. Note that $\mathbf{A}(\mathcal{T}, \mathcal{T})$ and $\mathbf{L}(\mathcal{T}, :)\}$ are sub-matrices of $\mathbf{A}$ and $\mathbf{L}$, respectively. If we replace an existing member $p \in \mathcal{T}$ of a given team $\mathcal{T}$ by another individual $q \notin \mathcal{T}$, the new team members are indexed by $\mathcal{T}_{p\to q} := \{\mathcal{T}/p, q\}$; and the new team is represented by the labelled network $\mathbf{G}(\mathcal{T}_{p\to q})$.

With the above notations and assumptions, our problems can be formally defined as follows:

Objective

Problem 1. Team Member Replacement

Given: *(1) A labelled social network* $\mathbf{G} := \{\mathbf{A}, \mathbf{L}\}$*, (2) a team* $\mathbf{G}(\mathcal{T})$*, and (3) a team member* $p \in \mathcal{T}$*;*

Output: *A "best" alternate* $q \notin \mathcal{T}$ *to replace the person* p*'s role in the team* $\mathbf{G}(\mathcal{T})$*.*

3. PROPOSED SOLUTIONS

In this section, we present our solution for Problem 1. We start with the design objectives for the Team Member Replacement problem, present graph kernel as the basic solution to fulfill such design objectives; and finally analyze the main computational challenges.

3.1 Design Objectives

Generally speaking, we want to find a *similar* person q to replace the current team member p who is about to leave the team. That is, a good replacement q should not only have a similar skill set as team member p; but also would maintain the good chemistry of the team so that the whole team can work together harmonically and/or be less disrupted. In other words, the similarity between individuals should be measured in the context of the team itself. Often, the success of a team largely depends on the successful execution of several sub-tasks, each of which requires the cooperation among several team members with certain skill configurations. For example, several classic tactics often recurringly find themselves in a successful NBA team, including (a) *triangle offense* (which is featured by a sideline triangle created by *the center*, *the forward*, and *the guard*), (b) *pick and roll* (which involves the cooperation between two players - one plays as 'pivot' and the other plays as 'screen', respectively), etc. Generally speaking, team performance arises from the shared knowledge and experience amongst team members and their ability to share and coordinate their work. As noted in the introduction, a specific pattern of communication, is associated with higher team performance. Maintaining that communication structure should therefore be less disruptive to the team.

If we translate these requirements into the notations defined in Section 2, it naturally leads to the following two design objectives for a good Team Member Replacement . First (*skill matching*), the new member should bring a similar skill set as the current team member p to be replaced that are required by the team. Second (*structural matching*), the new member should have a similar network structure as team member p in connecting the rest team members.

3.2 Basic Solutions

In order to fulfill the above two design objectives, we need a similarity measure between two individuals in the context of the team itself, that captures both skill matching and the structural matching. We refer to this kind of similarity as *team context aware similarity.* Mathematically, the so-called graph kernel defined on the current and new teams provides a natural tool for such a team context aware similarity. That is, we want to find a replacement person q as

$$q = \text{argmax}_{j, j \notin \mathcal{T}} \ \ \text{Ker}(\mathbf{G}(\mathcal{T}), \mathbf{G}(\mathcal{T}_{p\to j})) \tag{1}$$

Objective

In Eq. (1), $\mathbf{G}(\mathcal{T})$ is the labelled team graph; and $\mathbf{G}(\mathcal{T}_{p\to j})$ is the labelled team graph after we replace a team member p by another individual j; and Ker(.) is the kernel between these two labelled graphs. Generally speaking, the basic idea of various graph kernels is to compare the similarity of the *sub-graphs* between the two input graphs and then aggregate them as the overall similarity between the two graphs. Let us explain the intuition/rationality of why graph kernel is a natural choice for team context aware similarity. Here, each subgraph in a given team (e.g., the dashed triangles in Figure 1) might reflect a specific skill configuration among a sub-group of team members that is required by a certain sub-task of that team. By comparing the similarity between two subgraphs, we implicitly measure the capability of the individual j to perform this specific sub-task. Thus, by aggregating the similarities of all the possible subgraphs between the two input graphs/teams, we get a goodness measure of the overall capability of the individual j to perform all the potential sub-tasks that team member p is involved in the original team. Note that the team replacement scenario is different from team formation [27, 1, 33]. These existing work on team formation aims to build a team from scratch by optimizing some pre-chosen metric (e.g., compatibility, diversity, etc). In contract, we aim to find a new team member such that the new team resembles the original team as much as possible.

Def.

Having this in mind, many of the existing graph kernels can be adopted in Eq. (1), such as random walk based graph kernel, sub-tree based graph kernels (See section 6 for a review). In this paper, we focus on random walk based graph kernel due to its mathematical elegancy and superior empirical performance. Given two labelled graphs $\mathbf{G}_i := \{\mathbf{A}_i, \mathbf{L}_i\}$, $i = 1, 2$, the random walk based graph kernel between them can be formally computed as follows [39]:

$$\text{Ker}(\mathbf{G}_1, \mathbf{G}_2) = \mathbf{y}'(\mathbf{I} - c\mathbf{A}_\times)^{-1}\mathbf{L}_\times\mathbf{x} \tag{2}$$

Def.

where $\mathbf{A}_\times = \mathbf{L}_\times(\mathbf{A}'_1 \otimes \mathbf{A}'_2)$ is the weight matrix of the two graphs' Kronecker product, $\otimes$ represents the Kronecker product between two matrices, c is a decay factor, $\mathbf{y} = \mathbf{y}_1 \otimes \mathbf{y}_2$ and $\mathbf{x} = \mathbf{x}_1 \otimes \mathbf{x}_2$ are the so-called starting and stopping vectors to indicate the weights of different nodes and are set uniform in our case, $\mathbf{L}_\times$ is a diagonal matrix where $\mathbf{L}_\times(i,i) = 0$ if the i^{th} row of $(\mathbf{A}'_1 \otimes \mathbf{A}'_2)$ is zeroed out due to label inconsistency of two nodes of the two graphs. $\mathbf{L}_\times$ can be expressed as $\mathbf{L}_\times = \sum_{k=1}^{l} \text{diag}(\mathbf{L}_1(:,k)) \otimes \text{diag}(\mathbf{L}_2(:,k))$.

3.3 Computational Challenges

Eq.(2) naturally suggests the following procedure for solving Team Member Replacement problem (referred to as TeamRep-Basic): for each individual $j \notin \mathcal{T}$, we compute its score score(j) by Eq.(2); and recommend the individual(s) with the highest score(s). However, this strategy (TeamRep-Basic) is computationally intensive since we need to compute *many* random walk based graph kernels and each of such computations could be expensive especially when the team size is large. To be specific, for a team $\mathcal{T}$ of size t and a graph $\mathbf{G}$ with n individuals in total, its time complexity is $O(nt^6)$ since we need to compute a random walk based graph kernel for each candidate who is not in the current team, each of which could cost $O(t^6)$ [39] . Even if we allow some approximation in computing each of these graph kernels, the best known algorithms (i.e., by [23]) would still give an overall time complexity as $O(n(lt^2r^4 + mr + r^6))$,

Obstacle

where r is reduced rank after low rank approximation, which is still too high. For example, on the *DBLP* dataset with 916,978 authors, for a team with 10 members, it would take 6,388s to find a best replacement.

In the next section, we present our solution to remedy these computational challenges.

4. SCALE-UP AND SPEED-UP

In this section, we address the computational challenges to scale-up and speed-up TEAMREP-BASIC. Step1 We start with an efficient pruning strategy to reduce the number of graph kernel computations, Step2 and then present two algorithms to speed-up the computation of individual graph kernel.

4.1 Scale-up: Candidate Filtering

Here, we propose an efficient pruning strategy to filter out those unpromising candidates. Recall that one of our design objectives for a good TEAM MEMBER REPLACEMENT is *structural matching*, i.e., the new member has a similar network structure as team member p in connecting the rest team members. Reason Since p is connected to at least some of the rest members, it suggests that if an individual does not have any connection to any of the rest team members, s/he might not be a good candidate for replacement.

Pruning Strategy: Filter out all the candidates who do not have any connections to any of the rest team members.

LEMMA 1. Effectiveness of Pruning. *For any two persons i and j not in $\mathcal{T}$, if i is connected to at least one member in $\mathcal{T}/p$ and j has no connections to any of the members in $\mathcal{T}/p$, we have that*

$$Ker(\mathbf{G}(\mathcal{T}), \mathbf{G}(\mathcal{T}_{p\to i})) \geq Ker(\mathbf{G}(\mathcal{T}), \mathbf{G}(\mathcal{T}_{p\to j})).$$

PROOF. Suppose that $\mathbf{G}(\mathcal{T}) := \{\mathbf{A}_0, \mathbf{L}_0\}$. Let $\mathbf{G}(\mathcal{T}_{p\to i}) := \{\mathbf{A}_1, \mathbf{L}_1\}$, and $\mathbf{G}(\mathcal{T}_{p\to j}) := \{\mathbf{A}_2, \mathbf{L}_2\}$.

By Taylor expansion of Eq. (2), we have

$\mathrm{Ker}(\mathbf{G}(\mathcal{T}), \mathbf{G}(\mathcal{T}_{p\to i})) = \sum_{z=0}^{\infty} c\mathbf{y}'(\mathbf{L}_{\times 1}(\mathbf{A}'_0 \otimes \mathbf{A}'_1))^z\mathbf{x}$, where $\mathbf{L}_{\times 1} = \sum_{k=1}^{l} \mathrm{diag}(\mathbf{L}_0(:,k)) \otimes \mathrm{diag}(\mathbf{L}_1(:,k))$,

$\mathrm{Ker}(\mathbf{G}(\mathcal{T}), \mathbf{G}(\mathcal{T}_{p\to j})) = \sum_{z=0}^{\infty} c\mathbf{y}'(\mathbf{L}_{\times 2}(\mathbf{A}'_0 \otimes \mathbf{A}'_2))^z\mathbf{x}$, where $\mathbf{L}_{\times 2} = \sum_{k=1}^{l} \mathrm{diag}(\mathbf{L}_0(:,k)) \otimes \mathrm{diag}(\mathbf{L}_2(:,k))$.

Therefore, it is sufficient to show that $(\mathbf{L}_{\times 1}(\mathbf{A}'_0 \otimes \mathbf{A}'_1))^z \geq (\mathbf{L}_{\times 2}(\mathbf{A}'_0 \otimes \mathbf{A}'_2))^z$ for any $z > 0$, where two matrices $\mathbf{A} \geq \mathbf{B}$ if $\mathbf{A}_{ij} \geq \mathbf{B}_{ij}$ holds for all possible (i,j). We prove this by induction.

(Base Case of Induction) When $z = 1$, we have

$$\begin{aligned} &\mathbf{L}_{\times 1}(\mathbf{A}'_0 \otimes \mathbf{A}'_1) \\ &= (\textstyle\sum_{k=1}^{l} \mathrm{diag}(\mathbf{L}_0(:,k)) \otimes \mathrm{diag}(\mathbf{L}_1(:,k)))(\mathbf{A}'_0 \otimes \mathbf{A}'_1) \\ &= \textstyle\sum_{k=1}^{l} (\mathrm{diag}(\mathbf{L}_0(:,k))\mathbf{A}'_0) \otimes (\mathrm{diag}(\mathbf{L}_1(:,k))\mathbf{A}'_1) \end{aligned} \tag{3}$$

Because $(\mathrm{diag}(\mathbf{L}_1(:,k))\mathbf{A}'_1) \geq (\mathrm{diag}(\mathbf{L}_2(:,k))\mathbf{A}'_2)$, we have $\mathbf{L}_{\times 1}(\mathbf{A}'_0 \otimes \mathbf{A}'_1) \geq \mathbf{L}_{\times 2}(\mathbf{A}'_0 \otimes \mathbf{A}'_2)$.

(Induction Step) Assuming $(\mathbf{L}_{\times 1}(\mathbf{A}'_0 \otimes \mathbf{A}'_1))^{z-1} \geq (\mathbf{L}_{\times 2}(\mathbf{A}'_0 \otimes \mathbf{A}'_2))^{z-1}$, we have that

$$\begin{aligned} (\mathbf{L}_{\times 1}(\mathbf{A}'_0 \otimes \mathbf{A}'_1))^z &\geq (\mathbf{L}_{\times 2}(\mathbf{A}'_0 \otimes \mathbf{A}'_2))^{z-1}(\mathbf{L}_{\times 1}(\mathbf{A}'_0 \otimes \mathbf{A}'_1)) \\ &\geq (\mathbf{L}_{\times 2}(\mathbf{A}'_0 \otimes \mathbf{A}'_2))^z \end{aligned}$$

where the first inequality is due to the induction assumption; and the second inequality is due to the base case. This completes the proof. □

Remarks. By Lemma 1, our pruning strategy is 'safe', i.e., it will not miss any potentially good replacements. In the meanwhile, we can reduce the number of graph kernel compuations from $O(n)$ to $O(\sum_{i\in\mathcal{T}/p} d_i)$, which is sub-linear in n.

4.2 Speedup Graph Kernel - Exact Approach

Here, we address the problem of speeding up the computation of an individual graph kernel. Let $\mathbf{G}(\mathcal{T}) := \{\mathbf{A}_1, \mathbf{L}_1\}$ and $\mathbf{G}(\mathcal{T}_{p\to q}) := \{\mathbf{A}_2, \mathbf{L}_2\}$, where $\mathbf{A}_1, \mathbf{A}_2$ are symmetric adjacency matrices of the two graphs.[1] Without loss of generality, let us assume that p is the last team member in $\mathcal{T}$. Compare $\mathbf{A}_1$ with $\mathbf{A}_2$, it can be seen that the only difference is their last columns and last rows. Therefore, we can rewrite $\mathbf{A}_2$ as $\mathbf{A}_2 = \mathbf{A}_c + \mathbf{A}_{d2}$, where $\mathbf{A}_c$ is $\mathbf{A}_1$ with its last row and column being zeroed out, and the nonzero elements of $\mathbf{A}_{d2}$ only appear in its last row and column reflecting the connectivity of q to the new team. Notice that $\mathbf{A}_{d2}$ has a rank at most 2, so it can be factorized into two smaller matrices as $\mathbf{A}_{d2} = \mathbf{E}_{t\times 2}\mathbf{F}_{2\times t}$.

Denote $\mathrm{diag}(\mathbf{L}_1(:,j))$ as $\mathbf{L}_1^{(j)}$ and $\mathrm{diag}(\mathbf{L}_2(:,j))$ as $\mathbf{L}_2^{(j)}$ for $j = 1, ..., l$. Compare $\mathbf{L}_1^{(j)}$ with $\mathbf{L}_2^{(j)}$, the only difference is the last diagonal element. Therefore, we can write $\mathbf{L}_2^{(j)}$ as $\mathbf{L}_2^{(j)} = \mathbf{L}_c^{(j)} + \mathbf{L}_{d2}^{(j)}$, where $\mathbf{L}_c^{(j)}$ is $\mathbf{L}_1^{(j)}$ with last element zeroed out, and $\mathbf{L}_{d2}^{(j)}$'s last element indicates q's strength of having the jth skill. $\mathbf{L}_2^{(j)}$'s rank is at most 1, so it can be factorized as $\mathbf{L}_2^{(j)} = \mathbf{e}_{t\times 1}^{(j)}\mathbf{f}_{1\times t}^{(j)}$. Therefore, the exact graph kernel for the labelled graph can be computed as:

$$\begin{aligned}
&\mathrm{Ker}(\mathbf{G}(\mathcal{T}), \mathbf{G}(\mathcal{T}_{p\to q})) \\
&= \mathbf{y}'(\mathbf{I} - c(\textstyle\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_2^{(j)})(\mathbf{A}'_1 \otimes \mathbf{A}'_2))^{-1}(\textstyle\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_2^{(j)})\mathbf{x} \\
&= \mathbf{y}'(\mathbf{I} - \underbrace{c(\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})(\mathbf{A}_1 \otimes \mathbf{A}_c)}_{\mathbf{Z}\text{: invariant w.r.t. } q} \\
&\quad \underbrace{-c(\sum_{j=1}^{l}(\mathbf{L}_1^{(j)} \otimes \mathbf{e}^{(j)})(\mathbf{I} \otimes \mathbf{f}^{(j)}))(\mathbf{A}_1 \otimes \mathbf{A}_c)}_{\mathbf{PQ}(\mathbf{A}_1 \otimes \mathbf{A}_c) = \mathbf{PY}_1} \\
&\quad \underbrace{-c(\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})(\mathbf{A}_1 \otimes \mathbf{E})(\mathbf{I} \otimes \mathbf{F})}_{\mathbf{X}_1\mathbf{Y}_2} \\
&\quad \underbrace{-c(\sum_{j=1}^{l}(\mathbf{L}_1^{(j)} \otimes \mathbf{e}^{(j)})(\mathbf{I} \otimes \mathbf{f}^{(j)}))(\mathbf{A}_1 \otimes \mathbf{E})(\mathbf{I} \otimes \mathbf{F})}_{\mathbf{X}_2\mathbf{Y}_2})^{-1} \\
&\quad (\textstyle\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_2^{(j)})\mathbf{x}
\end{aligned} \tag{4}$$

Each $\mathbf{L}_1^{(j)} \otimes \mathbf{e}^{(j)}$ is a matrix of size t^2 by t and $\mathbf{I} \otimes \mathbf{f}^{(j)}$ is a matrix of size t by t^2. We denote the matrix created by concatenating all $\mathbf{L}_1^{(j)} \otimes \mathbf{e}^{(j)}$ horizontally as $\mathbf{P}$, *i.e.*, $\mathbf{P} = [\mathbf{L}_1^{(1)} \otimes \mathbf{e}^{(1)}, \ldots, \mathbf{L}_1^{(l)} \otimes \mathbf{e}^{(l)}]$; denote the matrix created by stacking all $\mathbf{I} \otimes \mathbf{f}^{(j)}$ vertically as $\mathbf{Q}$, *i.e.*, $\mathbf{Q} = [\mathbf{I} \otimes \mathbf{f}^{(1)}; \ldots; \mathbf{I} \otimes \mathbf{f}^{(l)}]$. Obviously, $(\sum_{j=1}^{l}(\mathbf{L}_1^{(j)} \otimes \mathbf{e}^{(j)})(\mathbf{I} \otimes \mathbf{f}^{(j)}))$ is equal to $\mathbf{PQ}$. We denote $(\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})(\mathbf{A}_1 \otimes \mathbf{E})$ by $\mathbf{X}_1$; denote $(\sum_{j=1}^{l}(\mathbf{L}_1^{(j)} \otimes \mathbf{e}^{(j)})(\mathbf{I} \otimes \mathbf{f}^{(j)}))(\mathbf{A}_1 \otimes \mathbf{E})$ by $\mathbf{X}_2$; denote $\mathbf{Q}(\mathbf{A}_1 \otimes \mathbf{A}_c)$ by $\mathbf{Y}_1$ and denote $(\mathbf{I} \otimes \mathbf{F})$ by $\mathbf{Y}_2$. Let $\mathbf{X}$ be $[\mathbf{P}, \mathbf{X}_1, \mathbf{X}_2]$ and $\mathbf{Y}$ be $[\mathbf{Y}_1; \mathbf{Y}_2; \mathbf{Y}_2]$.

With these additional notations, we can rewrite Eq. (4) as

[1]Although we focus on the undirected graphs in this paper, our proposed algorithms can be generalized to directed graphs.

$$\begin{aligned}&\mathrm{Ker}(\mathbf{G}(\mathcal{T}), \mathbf{G}(\mathcal{T}_{p\to q})) = \mathbf{y}'(\mathbf{Z} - c\mathbf{X}\mathbf{Y})^{-1}(\textstyle\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_2^{(j)})\mathbf{x} \\ &= \mathbf{y}'(\mathbf{Z}^{-1} + c\mathbf{Z}^{-1}\mathbf{X}(\mathbf{I} - c\mathbf{Y}\mathbf{Z}^{-1}\mathbf{X})^{-1}\mathbf{Y}\mathbf{Z}^{-1}) \\ &((\textstyle\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})\mathbf{x} + (\sum_{j=1}^{l}(\mathbf{L}_1^{(j)} \otimes \mathbf{e}^{(j)})(\mathbf{I} \otimes \mathbf{f}^{(j)}))\mathbf{x})\end{aligned} \tag{5}$$

where the second equation is due to the matrix inverse lemma [18].

Remarks. In Eq. (5), $\mathbf{Z} = \mathbf{I} - c(\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})(\mathbf{A}_1 \otimes \mathbf{A}_c)$ does not depend on the candiate q. Thus, if we pre-compute its inverse $\mathbf{Z}^{-1}$, we only need to update $\mathbf{X}(\mathbf{I} - c\mathbf{Y}\mathbf{Z}^{-1}\mathbf{X})^{-1}\mathbf{Y}$ and $\mathbf{PQx}$ for every new candidate. Notice that compared with the original graph kernel (the first equation in Eq. (4)), $(\mathbf{I} - c\mathbf{Y}\mathbf{Z}^{-1}\mathbf{X})$ is a much smaller matrix of $(l+4)t \times (l+4)t$. In this way, we can accelerate the process of computing its inverse without losing the accuracy of graph kernel.

4.3 Speedup Graph Kernel - Approx Approach

Note that the graph kernel by Eq. (5) is exactly the same as the original method by the first equation in Eq. (4). If we allow some approximation error, we can further speed-up the computation.

Note that $\mathbf{A}_c$ is symmetric and its rank-r approximation can be written as $\hat{\mathbf{A}}_c = \mathbf{U}\mathbf{V}$, where $\mathbf{U}$ is a matrix of size t by r and $\mathbf{V}$ is a matrix of size r by t. $\mathbf{A}_1$ can be approximated as $\hat{\mathbf{A}}_1 = \hat{\mathbf{A}}_c + \mathbf{A}_{d1} = \mathbf{U}\mathbf{V} + \mathbf{E}_1\mathbf{F}_1 = \mathbf{X}_1\mathbf{Y}_1$, where $\mathbf{X}_1 = [\mathbf{U}, \mathbf{E}_1], \mathbf{Y}_1 = [\mathbf{V}; \mathbf{F}_1], \mathbf{E}_1 = [\mathbf{w}_1, \mathbf{s}], \mathbf{F}_1 = [\mathbf{s}'; \mathbf{w}_1']$, $\mathbf{s}$ is a zero vector of length t except that the last element is 1, and $\mathbf{w}_1$ is the weight vector from p to the members in $\mathcal{T}$. Similarly, after p is replaced by a candidate q, the weight matrix of the new team can be approximated as $\hat{\mathbf{A}}_2 = \mathbf{X}_2\mathbf{Y}_2$ where $\mathbf{X}_2 = [\mathbf{U}, \mathbf{E}_2], \mathbf{Y}_2 = [\mathbf{V}; \mathbf{F}_2], \mathbf{E}_2 = [\mathbf{w}_2, \mathbf{s}], \mathbf{F}_2 = [\mathbf{s}'; \mathbf{w}_2']$ and $\mathbf{w}_2$ is the weight vector from q to the members in the new team. The approximated graph kernel for labeled graphs can be computed as:

$$\begin{aligned}&\hat{\mathrm{Ker}}(\mathbf{G}(\mathcal{T}), \mathbf{G}(\mathcal{T}_{p\to q})) = \mathbf{y}^T(\mathbf{I} - c\mathbf{L}_\times(\hat{\mathbf{A}}_1' \otimes \hat{\mathbf{A}}_2'))^{-1}\mathbf{L}_\times\mathbf{x} \\ &= \mathbf{y}'(\mathbf{I} - c\mathbf{L}_\times(\mathbf{X_1Y_1}) \otimes (\mathbf{X_2Y_2}))^{-1}\mathbf{L}_\times\mathbf{x} \\ &= \mathbf{y}'(\mathbf{I} - c\mathbf{L}_\times(\mathbf{X_1} \otimes \mathbf{X_2})(\mathbf{Y_1} \otimes \mathbf{Y_2}))^{-1}\mathbf{L}_\times\mathbf{x} \\ &= \mathbf{y}'(\mathbf{I} + c\mathbf{L}_\times(\mathbf{X_1} \otimes \mathbf{X_2})\mathbf{M}(\mathbf{Y_1} \otimes \mathbf{Y_2}))\mathbf{L}_\times\mathbf{x} \\ &= \mathbf{y}'\mathbf{L}_\times\mathbf{x} + c\mathbf{y}'(\textstyle\sum_{j=1}^{l} \mathbf{L}_1^{(j)}\mathbf{X}_1 \otimes \mathbf{L}_2^{(j)}\mathbf{X_2})\mathbf{M}(\mathbf{Y}_1 \otimes \mathbf{Y}_2)\mathbf{L}_\times\mathbf{x} \\ &= (\textstyle\sum_{j=1}^{l}(\mathbf{y}_1'\mathbf{L}_1^{(j)}\mathbf{x}_1)(\mathbf{y}_2'\mathbf{L}_2^{(j)}\mathbf{x}_2)) + c \\ &(\textstyle\sum_{j=1}^{l} \mathbf{y}_1'\mathbf{L}_1^{(j)}\mathbf{X_1} \otimes \mathbf{y}_2'\mathbf{L}_2^{(j)}\mathbf{X_2})\mathbf{M}(\sum_{j=1}^{l} \mathbf{Y}_1\mathbf{L}_1^{(j)}\mathbf{x}_1 \otimes \mathbf{Y}_2\mathbf{L}_2^{(j)}\mathbf{x}_2)\end{aligned} \tag{6}$$

where $\mathbf{M} = (\mathbf{I} - c(\sum_{j=1}^{l} \mathbf{Y}_1\mathbf{L}_1^{(j)}\mathbf{X_1} \otimes \mathbf{Y}_2\mathbf{L}_2^{(j)}\mathbf{X_2}))^{-1}$, the second equation is due to the kronecker product property; the third equation is again due to the matrix inverse lemma, the fourth equation is by matrix multiplication distributivity and the last equation is due to the kronecker product property.

Remarks. The computation of $\mathbf{M}$ is much cheaper than the original graph kernel since it is a matrix inverse of size $(r+2)^2 \times (r+2)^2$. It was first proposed in [23] to explore the low-rank structure of the input graphs to speed-up graph kernel computations. However, in the context of TEAM MEMBER REPLACEMENT , we would need to estimate the low-rank approximation *many* times ($O(\sum_{i\in\mathcal{T}/p} d_i)$) when we directly apply the method in [23]. In contrast, we only need to compute top-r approximation *once* by Eq. (6). As our complexity analysis (subsection 4.4) and experimental evaluations (subsection 5.3) show, this brings a few times additional speed-up.

4.4 Putting Everything Together

Putting everthing together, we are ready to present our algorithms for TEAM MEMBER REPLACEMENT . Depending on the specific methods for computing the individual graph kernels, we propose two variants.

4.4.1 Variant #1: TEAMREP-FAST-EXACT

We first present our algorithm using the exact graph kernel computation in Eq. (5). The algorithm (TEAMREP-FAST-EXACT) is summarized in Algorithm 1. We only need to pre-compute and store $\mathbf{Z}^{-1}$, $\mathbf{R}$, $\mathbf{b}$ and $\mathbf{l}$ for later use to compute each candidate's score (step 2 and 3). In the loop, the key step is to update $\mathbf{M}$ involving matrix inverse of size $(l+4)t \times (l+4)t$ which is relatively cheaper to compute (step 17).

Algorithm 1: TEAMREP-FAST-EXACT

```
Input: (1) The entire social network $\mathbf{G} := \{\mathbf{A}, \mathbf{L}\}$, (2)
       original team members $\mathcal{T}$, (3) person $p$ who will
       leave the team, (4) starting and ending
       probability $\mathbf{x}$ and $\mathbf{y}$(be uniform by default),
       and (5) an integer $k$ (the budget)
Output: Top $k$ candidates to replace person $p$
1  Initialize $\mathbf{A}_c, \mathbf{L}_1^{(j)}, \mathbf{L}_2^{(j)}, j = 1, \ldots, l$ ;
2  Pre-compute
   $\mathbf{Z}^{-1} \leftarrow (\mathbf{I} - c(\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})(\mathbf{A}_1 \otimes \mathbf{A}_c))^{-1}$;
3  Set $\mathbf{R} \leftarrow (\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})\mathbf{x}$; $\mathbf{b} \leftarrow \mathbf{y}^T\mathbf{Z}^{-1}\mathbf{R}$; $\mathbf{l} \leftarrow c\mathbf{y}^T\mathbf{Z}^{-1}$;
4  for each candidate $q$ in $\mathbf{G}$ after pruning do
5      Initialize $\mathbf{s} \leftarrow$ a zero vector of length $t$ except the
       last element is 1;
6      Initialize $\mathbf{w} \leftarrow$ weight vector from $q$ to the new
       team members;
7      Set $\mathbf{E} \leftarrow [\mathbf{w}, \mathbf{s}]$; $\mathbf{F} \leftarrow [\mathbf{s}'; \mathbf{w}']$ ;
8      Set $\mathbf{e}^{(j)} \leftarrow$ a $t$ by 1 zero vector except the last
       element is 1, for $j = 1, \ldots, d_n$ ;
9      Set $\mathbf{f}^{(j)} \leftarrow$ a $1 \times t$ zero vector except the last
       element which is label $j$ assignment for $q$;
10     Set $\mathbf{P} \leftarrow [\mathbf{L}_1^{(1)} \otimes \mathbf{e}^{(1)}, \ldots, \mathbf{L}_1^{(l)} \otimes \mathbf{e}^{(l)}]$;
11     Set $\mathbf{Q} \leftarrow [\mathbf{I} \otimes \mathbf{f}^{(1)}; \ldots; \mathbf{I} \otimes \mathbf{f}^{(l)}]$;
12     Compute $\mathbf{X}_1 \leftarrow (\sum_{j=1}^{l} \mathbf{L}_1^{(j)}\mathbf{A}_1 \otimes \mathbf{L}_c^{(j)}\mathbf{E})$;
13     Compute $\mathbf{X}_2 \leftarrow (\sum_{j=1}^{l} \mathbf{L}_1^{(j)}\mathbf{A}_1 \otimes \mathbf{e}^{(j)}\mathbf{f}^{(j)}\mathbf{E})$;
14     Compute $\mathbf{Y}_1 \leftarrow \mathbf{Q}(\mathbf{A}_1 \otimes \mathbf{A}_c)$;
15     Compute $\mathbf{Y}_2 \leftarrow (\mathbf{I} \otimes \mathbf{F})$;
16     Set $\mathbf{X} \leftarrow [\mathbf{P}, \mathbf{X}_1, \mathbf{X}_2]$, $\mathbf{Y} \leftarrow [\mathbf{Y}_1; \mathbf{Y}_2; \mathbf{Y}_2]$;
17     Update $\mathbf{M} \leftarrow (\mathbf{I} - c\mathbf{Y}\mathbf{Z}^{-1}\mathbf{X})^{-1}$;
18     Compute $\mathbf{r}' \leftarrow \mathbf{Z}^{-1}\mathbf{PQx}$;
19     Compute score($q$) $= \mathbf{b} + \mathbf{y}^T\mathbf{r}' + \mathbf{lXMY}(\mathbf{Z}^{-1}\mathbf{R} + \mathbf{r}')$ ;
20 end
21 Return the top $k$ candidates with the highest scores.
```

The effectiveness and efficiency of TEAMREP-FAST-EXACT are summarized in Lemma 2 and Lemma 3, respectively. Compared with TEAMREP-BASIC, Algorithm 1 is much faster without losing any recommendation accuracy.

LEMMA 2. *Accuracy of* TEAMREP-FAST-EXACT. *Algorithm 1 outputs the same set of candidates as* TEAMREP-BASIC.

PROOF. (Sketch) First, according to Lemma 1, we will not miss a promising candidate during the pruning stage. Second, for each candidate after pruning, Algorithm 1 calculates its graph kernel exactly the same as Eq. (5), which is in turn the same as Eq. (4) and hence Eq. (2). Therefore, after ranking the scores, Algorithm 1 outputs the same set of candidates as TEAMREP-BASIC. □

Lemma 3. *Time Complexity of* TeamRep-Fast-Exact *Algorithm 1 takes $O((\sum_{i \in \mathcal{T}/p} d_i)(lt^5 + l^3t^3))$ in time.*

Proof. (Sketch) After pruning, the number of potential candidates (the number of loops in Algorithm 1) is $O(\sum_{i \in \mathcal{T}/p} d_i)$. In every loop, computing $\mathbf{X}_1$,$\mathbf{X}_2$ and $\mathbf{Y}_1$ take $O(lt^5)$; computing $\mathbf{M}$ takes $O(lt^5 + l^3t^3)$ and computing the score(q) takes $O(lt^3)$. Putting everything together, the time complexity of Algorithm 1 is $O((\sum_{i \in \mathcal{T}/p} d_i)(lt^5 + l^3t^3))$. □

4.4.2 *Variant #2:* TeamRep-Fast-Approx

By using Eq. (6) to compute the graph kernel instead, we propose an even faster algorithm (TeamRep-Fast-Approx), which is summarized in Algorithm 2. In the algorithm, we only need to compute the top r eigen-decomposition for $\mathbf{A}_c$ once (step 2), and use that to update the low rank approximation for every new team. Besides, when we update $\mathbf{M}$, a matrix inverse of size $(r+2)^2 \times (r+2)^2$ (step 14), the time is independent of the team size.

```
Algorithm 2: TeamRep-Fast-Approx
Input: (1) The entire social network $\mathbf{G} := \{\mathbf{A}, \mathbf{L}\}$, (2)
       original team members $\mathcal{T}$, (3) person $p$ who will
       leave the team, (4) starting and ending
       probability $\mathbf{x}$ and $\mathbf{y}$.(be uniform by default),
       and (5) an integer $k$ (the budget)
Output: Top $k$ candidates to replace person $p$
1  Initialize $\mathbf{A}_c, \mathbf{L}_1^{(j)}, \mathbf{L}_2^{(j)}, j = 1, \ldots, l$ ;
2  Compute top $r$ eigen-decomposition for $\mathbf{A}_c$:
   $\mathbf{U\Lambda U}' \leftarrow \mathbf{A}_c$ ;
3  Set $\mathbf{V} \leftarrow \mathbf{\Lambda U}'$;
4  Initialize $\mathbf{s} \leftarrow$ a zero vector of length $t$ except the last
   element is 1;
5  Initialize $\mathbf{w}_1 \leftarrow$ weight vector from $p$ to $\mathcal{T}$;
6  Set $\mathbf{E}_1 \leftarrow [\mathbf{w}_1, \mathbf{s}], \mathbf{F}_1 \leftarrow [\mathbf{s}'; \mathbf{w}_1']$ ;
7  Set $\mathbf{X}_1 \leftarrow [\mathbf{U}, \mathbf{E}_1]$ , $\mathbf{Y}_1 \leftarrow [\mathbf{V}; \mathbf{F}_1]$;
8  for each candidate $q$ in $\mathbf{G}$ after pruning do
9      Initialize $\mathbf{w}_2 \leftarrow$ weight vector from $q$ to the new
       team members ;
10     Set $\mathbf{E}_2 \leftarrow [\mathbf{w}_2, \mathbf{s}], \mathbf{F}_2 \leftarrow [\mathbf{s}'; \mathbf{w}_2']$ ;
11     Set $\mathbf{X}_2 \leftarrow [\mathbf{U}, \mathbf{E}_2]$ , $\mathbf{Y}_2 \leftarrow [\mathbf{V}; \mathbf{F}_2]$;
12     Compute $\mathbf{S} \leftarrow \sum_{j=1}^{l} \mathbf{y}_1'\mathbf{L}_1^{(j)}\mathbf{X}_1 \otimes \mathbf{y}_2'\mathbf{L}_2^{(j)}\mathbf{X}_2$;
13     Compute $\mathbf{T} \leftarrow \sum_{j=1}^{l} \mathbf{Y}_1\mathbf{L}_1^{(j)}\mathbf{x}_1 \otimes \mathbf{Y}_2\mathbf{L}_2^{(j)}\mathbf{x}_2)$;
14     Update $\mathbf{M} \leftarrow (\mathbf{I} - c(\sum_{j=1}^{l} \mathbf{Y}_1\mathbf{L}_1^{j}\mathbf{X}_1 \otimes \mathbf{Y}_2\mathbf{L}_2^{j}\mathbf{X}_2))^{-1}$;
15     Set score$(q) = (\sum_{j=1}^{l}(\mathbf{y}_1'\mathbf{L}_1^{(j)}\mathbf{x}_1)(\mathbf{y}_2'\mathbf{L}_2^{(j)}\mathbf{x}_2)) + c\mathbf{SMT}$ ;
16 end
17 Return the top $k$ candidates with the highest scores.
```

The effectiveness and efficiency of TeamRep-Fast-Approx are summarized in Lemma 4 and Lemma 5, respectively. Compared with TeamRep-Basic and TeamRep-Fast-Exact, Algorithm 2 is even faster; and the only place it introduces the approximation error is the low-rank approximation of $\mathbf{A}_c$ (step 2).

Lemma 4. *Accuracy of* TeamRep-Fast-Approx*. If $\mathbf{A}_c = \mathbf{U\Lambda U}'$ holds, Algorithm 2 outputs the same set of candidates as* TeamRep-Basic.

Proof. Omitted for brevity. □

Lemma 5. *Time Complexity of* TeamRep-Fast-Approx *Algorithm 2 takes $O((\sum_{i \in \mathcal{T}/p} d_i)(lt^2r + r^6))$ in time.*

Proof. Omitted for brevity. □

Table 2: Summary of Datasets.

Data	n	m	# of teams
DBLP	916,978	3,063,244	1,572,278
Movie	95,321	3,661,679	10,197
NBA	3,924	126,994	1,398

5. EXPERIMENTAL EVALUATIONS

In this section, we present the experimental evaluations. The experiments are designed to answer the following questions:

- *Effectiveness*: How accurate are the proposed algorithms for Team Member Replacement ?
- *Efficiency:* How scalable are the proposed algorithms?

5.1 Datasets

DBLP. DBLP dataset[2] provides bibliographic information on major computer science journals and proceedings. We use it to build a co-authorship network where each node is an author and the weight of each edge stands for the number of papers the two corresponding authors have co-authored. The network constructed has $n = 916,978$ nodes and $m = 3,063,244$ edges. We use the conferences (*e.g.,* KDD, SIGMOD, CVPR, etc) as the skill of the authors. For a given paper, we treat all of its co-authors as a team. Alternatively, if a set of authors co-organize an event (such as a conference), we also treat them as a team.

Movie. This dataset[3] is an extension of MovieLens dataset, which links movies from MovieLens with their corresponding IMDb webpage and Rotten Tomatoes review system. It contains information of 10,197 movies, 95,321 actors/actress and 20 movie genres (*e.g.,* action, comedy, horror, etc.). Each movie has on average 22.8 actors/actress and 2.0 genres assignments. We set up the social network of the actors/actresses where each node represents one actor/actress and the weight of each edge is the number of movies the two linking actors/actresses have co-stared. We use the movie genres that a person has played as his/her skills. For a given movie, we treat all of its actors/actress as a team.

NBA. The NBA dataset[4] contains NBA and ABA statistics from the year of 1946 to the year of 2009. It has information of 3,924 players and 100 teams season by season. We use players' positions as their skill labels, including *guard*, *forward* and *center*. The edge weight of the player network stands for the number of seasons that the two corresponding nodes/individuals played in the same team.

The statistics of these three datasets are summarized in Table 2. All the experiments are run on a Windows machine with 16 GB memory and Intel i7-2760QM CPU.

Repeatability of Experimental Results. All the three datasets are publicly available. We will release the code of the proposed algorithms through authors' website.

5.2 Effectiveness Results

A. Qualitative Evaluations. We first present some case studies on the three datasets to gain some intuitions.

Case studies on DBLP. Let us take a close look at Fig. 1, which shows a screenshot of our current demo system. The original team is shown on the left side and the person leaving the team (*Philip S. Yu*) is represented by a node (diagram)

[2]http://arnetminer.org/citation
[3]http://grouplens.org/datasets/hetrec-2011/
[4]http://www.databasebasketball.com

with larger radius. If the user clicks a replacement from the recommendation list (on the top), the system will show the new team on the right side. Here, We introduced a novel visualization technique to represent authors' relationships and their expertise within one unique graph visualization. Particularly, in this visualization, the authors are shown as voronoi diagrams [14]. The authors' expertise is visualized as the voronoi cells inside the diagram, that is, each cell indicates a type of expertise. We use different color hues to identify different expertise types and use the color saturations to encode the author's strength in that expertise type. For example, if *KDD* is represented in orange, a bright orange cell in a voronoi diagram means the author has a strong expertise in *KDD*. In contrast, a white cell indicates the author's lacking of the corresponding expertise. To facilitate visual comparison of different authors, we fix the position of these expertise cells across different diagrams so that, for example, *KDD* is always shown at the left side of the author diagrams. These voronoi diagrams are connected by links indicating the authors' relationships. The strength of the relationship are presented by the line thickness.

Fig. 1 visualizes the team structures before and after *Philip S. Yu* becomes unavailable in the team writing [32]. Our algorithm's top recommendation is *Jiawei Han*. As we can see, both *Han* and *Yu* posses very similar skills and are renowned for their extraordinary contributions in the data mining and databases community. Moreover, *Han* has collaborated with almost each of the rest authors/team members. Looking more closely, we find *Han* preserves several key triangle substructures in the original team: one with *Ke Wang* and *Jian Pei*, and the other with *Haixun Wang* and *Jian Pei*. These triangle sub-structures might play a critical role in accomplishing sub-tasks in writing the paper.

We also consider a bigger team, i.e, the organizing committee of *KDD 2013*. After filtering those not in *DBLP*, we have 32 people in the committee team. We use their co-authorship network as their social network. Suppose one of the research track co-chairs *Inderjit Dhillon* becomes unavailable and we are searching for another researcher who can fill in this critical role in organizing *KDD 2013*. The top five candidates our algorithm recommends are *Philip S. Yu*, *Jiawei Han*, *Christos Faloutsos*, *Bing Liu* and *Wei Wang*. The results are consistent with the intuitions - all of these recommended researchers are highly qualified - not only have they made remarkable contributions to the data mining field, but also they have strong ties with the remaining organizers of *KDD 2013*. For example, *Liu* is the current chair of KDD executive committee; *Wang* is one of the research track program chairs for *KDD 2014*; and *Faloutsos* was the PC co-chair of *KDD 2003*, etc.

Case studies on Movie. Assuming actor *Matt Damon* became unavailable when filming the epic war movie *Saving Private Ryan* (1998) and we need to find an alternative actor who can play *Ryan*'s role in the movie. The top five recommendations our algorithm gives are: *Samuel L. Jackson*, *Steve Buscemi*, *Robert De Niro*, *Christopher Walken*, *Bruce Willis*. As we know, *Saving Private Ryan* is a movie of *action* and *drama* genres. Notice that both *Damon* and *Jackson* have participated in many movies of *drama*, *thriller* and *action* genres, hence *Jackson* has the acting skills required to play the role in this movie. Moreover, *Jackson* has co-played with *Tom Sizemore*, *Vin Diesel*, *Dale Dye*, *Dennis Farina*, *Giovanni Ribisi* and *Ryan Hurst* in the crew before.

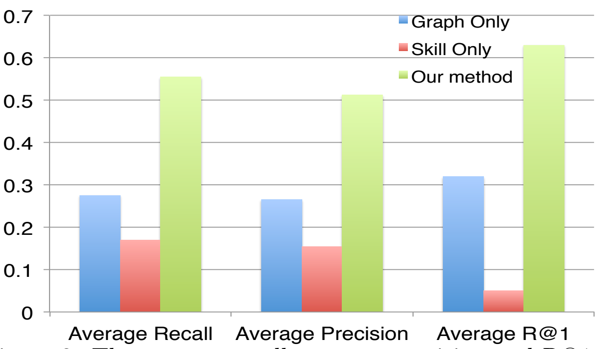

Figure 2: The average recall, average precision and R@1 of the three comparison methods. Higher is better.

The familiarity might increase the harmony of filming the movie with others.

Case studies on NBA. Let us assume that *Kobe Bryant* in Los Angeles Lakers was hurt during the regular season in 1996 and a bench player is badly wanted. The top five replacements our algorithm recommends are: *Rick Fox*, *A.c. Green*, *Jason Kidd*, *Brian Shaw* and *Tyronn Lue*. As we know, *Bryant* is a guard in NBA. Among the five recommendations, *Kidd*, *Shaw* and *Lue* all play as guards. More importantly, *Jason*, *Brian* and *Tyronn* have played with 9, 7 and 9 of the rest team members on the same team in the same season for multiple times. Therefore, it might be easier for them to maintain the moment and chemistry of the team which is critical to winning the game.

B. Quantitative Evaluations. Besides the above case studies, we also perform quantitative evaluations to compare the proposed algorithms with the alternative choices, including (a) only with *structure matching* and not including $\mathbf{L}_\times$ in Eq. (2) (Graph Only), and (b) only with *skill matching* and using cosine similarity of skill vectors as scores (Skill Only).

User studies. We perform a user study as follows. we choose 10 papers from various fields, replace one author of each paper and run the three comparison methods, and each of them recommends top five candidates. Then, we mix the outputs (15 recommendations in total) and ask users to (a) mark exactly one best replacement; (b) mark all good replacements from the list of 15 recommended candidates. The results are presented in Fig. 2, Fig. 3 and Fig. 4, respectively. As we can see from these figures, the proposed method (the green bar) is best in terms of both precision and recall. For example, the average recalls by our method, by 'Graph Only' and by 'Skill Only' are 55%, 28%, 17%, respectively. As for different papers, our method wins 9 out-of 10 cases (except for 'paper 2' where 'Skill Only' is best).

Author alias prediction. In *DBLP*, some researchers might have multiple name identities/alias. For example, in some papers, *Alexander J. Smola* might be listed as *Alex J. Smola*, *Zhongfei (Mark) Zhang* might be listed as *Zhongfei Zhang*, etc. For such an author, we run the team replacement algorithm on those papers s/he was involved to find top-k replacement. If his/her other alias appears in the top-k recommended list, we treat it as a *hit*. The average accuracy of different methods is shown in Fig. 5. Again, our method performs best.

5.3 Efficiency Results

A. The speed-up by pruning. To demonstrate the benefit of our pruning strategy, we run TeamRep-Basic with and without pruning on the three datasets and compare

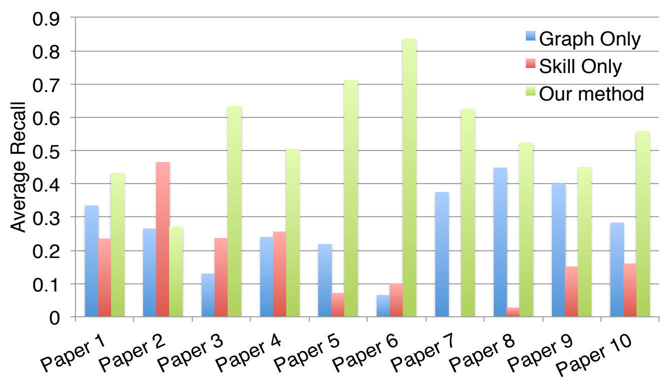

Figure 3: Recall for different papers. Higher is better.

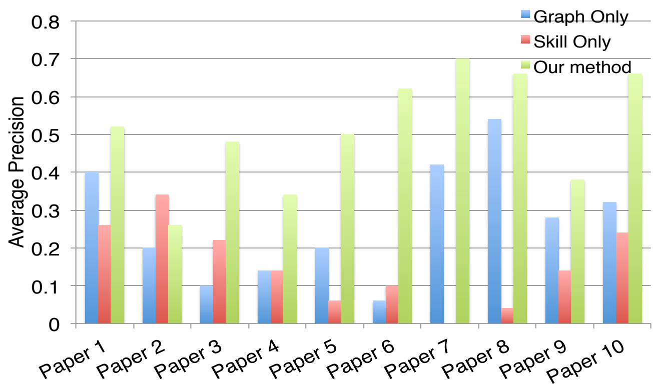

Figure 4: Precision for different papers. Higher is better.

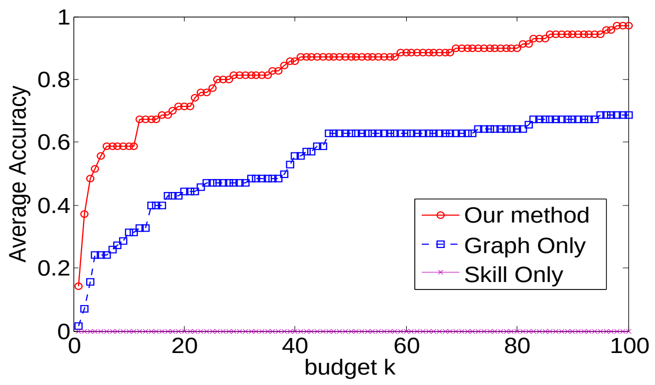

Figure 5: Average accuracy vs. budget k. Higher is better.

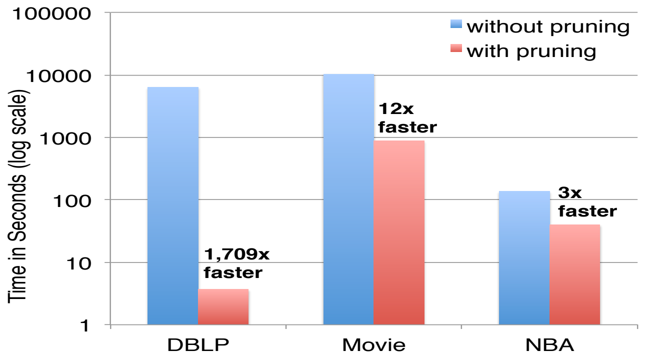

Figure 6: Time Comparisons before and after pruning on three datasets. Notice time is in log-scale.

their running time. For *DBLP*, we choose the authors of paper [28] (6 authors); for *Movie*, we select the film crew of *Titanic* (1997) (22 actors/actresses); for *NBA*, we pick the players on Los Angeles Lakers in year 1996 (17 players). The result is presented in Fig. 6. As we can see, the pruning step itself brings significant savings in terms of running time, especially for larger graphs (e.g., *DBLP* and *Movie*). Notice that according to Lemma 1, we do not sacrifice any recommendation accuracy by pruning.

B. Further speedup. Next, we vary the team sizes and compare the running time of TeamRep-Basic with TeamRep-Fast-Exact(exact methods); and Ark-L [23] with TeamRep-Fast-Approx (approximate methods). For TeamRep-Basic and Ark-L, we apply the same pruning step as their preprocessing step. The results are presented in Fig. 7 and Fig. 8, respectively. We can see that the proposed TeamRep-Fast-Exact and TeamRep-Fast-Approx are much faster than their alternative choices, especially when team size is large. Notice that Ark-L is the best known methods for approximating random walk based graph kernel.

C. Scalability. To test the scalability of our TeamRep-Fast-Exact and TeamRep-Fast-Approx algorithms, we sample a certain percentage of edges from the entire *DBLP* network and run the two proposed algorithms on teams with different sizes. The results are presented in Fig. 9 and Fig. 10, respectively. As we can seen, both algorithms enjoy a *sub-linear* scalability w.r.t. the total number of edges of the input graph (m).

6. RELATED WORK

In this section, we review the related work in terms of (a) team formation, (b) recommendation and expert finding, and (c) graph kernel.

Team Formation. Team formation studies the problem of assembling a team of people to work on a project. To ensure success, the selected team members should possess the desired skills and have strong team cohesion, which is first studied in [27]. As follow-up work, Anagnostopoulos et al [1] studies forming teams to accommodate a sequence of tasks arriving in an online fashion and Rangapuram et al [33] allows incorporating many realistic requirements into team formation based on a generalization of the densest subgraph problem. With the presence of the underlying social network, the set cover problem is complicated by the goal of lowering the communication cost at the same time. Bogdanov et al [3] studies how to extract a diversified group pulled from strong cliques given a network; this ensures that the group is both comprehensive and representative of the whole network. Cummings and Kiesler [9] find that prior working experience is the best predictor of collaborative tie strength. To provide insights into designs of online communities and organizations, the systematic differences in appropriating social softwares among different online enterprise communities is analyzed in [31]. The patterns of informal networks and communication in distributed global software teams using social network analysis is also investigated in [8]. Specific communication structures are proven critical to new product development delivery performance and quality [7]. For assessing the skills of players and teams in online multi-player games and team-based sports, "team chemistry" is also accounted for in [11, 10].

Recommendation and Expert Finding. Recommen-

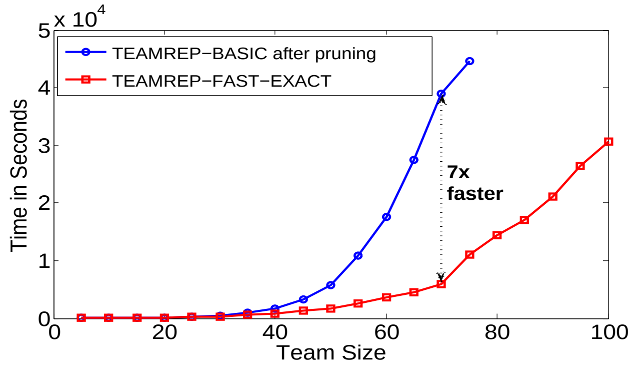

Figure 7: Time Comparison between TEAMREP-BASIC and TEAMREP-FAST-EXACT. TEAMREP-FAST-EXACT is on average 3× faster. TEAMREP-BASIC takes more than 10 hours when team size = 70.

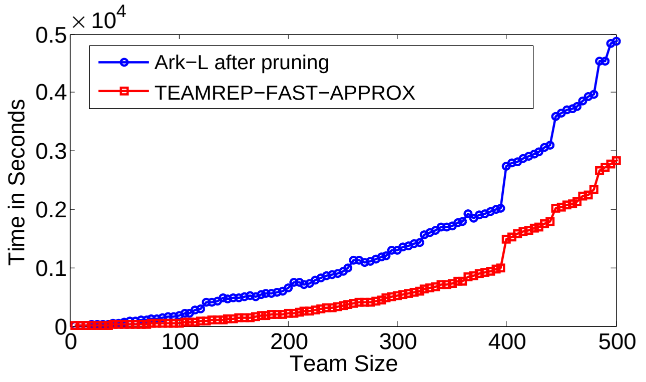

Figure 8: Time Comparisons between Ark-L[20] and TEAMREP-FAST-APPROX. TEAMREP-FAST-APPROX is on average 3× faster.

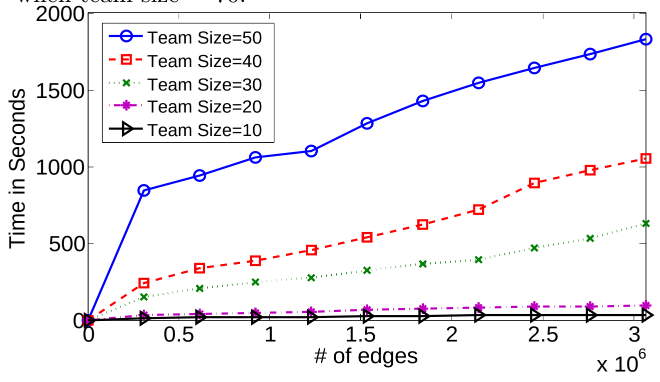

Figure 9: Running time of TEAMREP-FAST-EXACT vs. graph size. TEAMREP-FAST-EXACT scales sub-linearly w.r.t. the number of edges of the input graph.

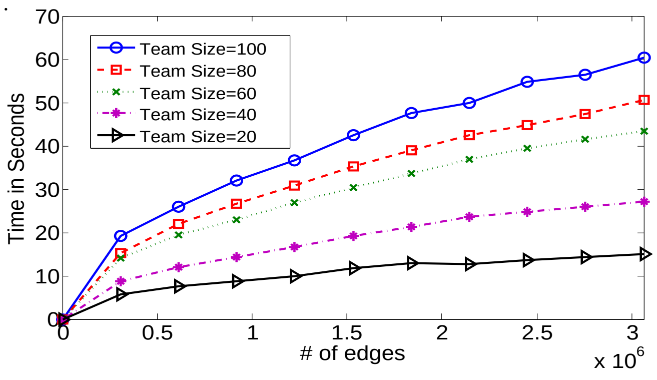

Figure 10: Running time vs. graph size. TEAMREP-FAST-APPROX scales sub-linearly w.r.t. the number of edges of the input graph.

dation and expert finding is a very active research topic in data mining and information retrieval, either to recommend products a user is mostly interested in or to identify the most knowledgeable people in a field. Our work is related to this in the sense that we aim to recommend top candidates who are most suitable for the vacancy. A popular method in recommendation (collaborative filtering) is latent factor model [26, 13, 41]. The basic idea is to apply matrix factorization to user-item rating data to identify the latent factors. The factorization technique can be naturally extended by adding biases, temporal dynamics and varying confidence levels. In question-answering sites, *e.g.*, Quora and Stack Overflow, an important task is to route a newly posted question to the 'right' user with appropriate expertise and several methods based on link analysis have been proposed [44, 6, 45]. In academia, identifying experts in a research field is of great value, *e.g.*, assigning papers to the right reviewers in a peer-review process [30, 24], which can be done by either building the co-author network [28] or using language model and topic-based model [12, 19]. For enterprises, finding the desired specialist can greatly reduce costs and facilitate the ongoing projects. Many methods have been proposed to expert search through an organization's document repository [2, 42].

Graph Kernel. Graph kernel measures the similarity between two graphs. Typical applications include automated reasoning [38], bioinformatics/chemoinformatics [15, 36]. Generally speaking, graph kernels can be categorized into three classes: kernels based on walks [16, 39, 40, 17, 4], kernels based on limited-sized subgraphs [22, 35, 25] and kernels based on subtree patterns [29, 34, 20]. Graph kernels based on random walk is one of the most successful choices [5]. The idea is to perform simultaneous walks on the two graphs and count the number of matching walks. One challenge of random walk based graph kernel lies in computation. The straight-forward method needs $O(n^6)$ in time. For unlabelled graphs, the time complexity can be reduced to $O(n^3)$ by reducing to the problem of solving a linear system [39, 40]. With low rank approximation, the computation can be further accelerated with high approximation accuracy [23].

7. CONCLUSION

In this paper, we study the problem of TEAM MEMBER REPLACEMENT to recommend replacement when a critical team member becomes unavailable. To our best knowledge, we are the first to study this problem. The basic idea of our method is to adopt graph kernel to encode both *skill matching* and *structural matching*. To address the computational challenges, we propose a suite of fast and scalable algorithms. Extensive experiments on real world datasets validate the effectiveness and efficiency of our algorithms. To be specific, (a) by bringing skill matching and structural matching together, our method is significantly better than the alternative choices in terms of both average precision (24% better) and recall (27% better); and (b) our fast algorithms are orders of magnitude faster while enjoying a *sub-linear* scalability.

In the future, we would like to expand team replacement to team enhancement and team composition. For instance, given the structure of a high-grossing movie (e.g., *Saving Private Ryan*) of a particular genre, we want to develop effective algorithm to suggest a team of actors.

References

[1] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi. Online team formation in social networks. In *WWW*, pages 839–848, 2012.

[2] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR*, pages 43–50, 2006.

[3] P. Bogdanov, B. Baumer, P. Basu, A. Bar-Noy, and A. K. Singh. As strong as the weakest link: Mining diverse cliques in weighted graphs. In *ECML/PKDD (1)*, pages 525–540, 2013.

[4] K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *ICDM*, pages 74–81, 2005.

[5] K. M. Borgwardt, H.-P. Kriegel, S. V. N. Vishwanathan, and N. Schraudolph. Graph kernels for disease outcome prediction from protein-protein interaction networks. In *Pacific Symposium on Biocomputing*, 2007.

[6] M. Bouguessa, B. Dumoulin, and S. Wang. Identifying authoritative actors in question-answering forums: the case of yahoo! answers. In *KDD*, pages 866–874, 2008.

[7] M. Cataldo and K. Ehrlich. The impact of communication structure on new product development outcomes. In *CHI*, pages 3081–3090, 2012.

[8] K. Chang and K. Ehrlich. Out of sight but not out of mind?: Informal networks, communication and media use in global software teams. In *CASCON*, pages 86–97, 2007.

[9] J. N. Cummings and S. B. Kiesler. Who collaborates successfully?: prior experience reduces collaboration barriers in distributed interdisciplinary research. In *CSCW*, pages 437–446, 2008.

[10] C. DeLong and J. Srivastava. Teamskill evolved: Mixed classification schemes for team-based multi-player games. In *PAKDD (1)*, pages 26–37, 2012.

[11] C. DeLong, L. G. Terveen, and J. Srivastava. Teamskill and the nba: applying lessons from virtual worlds to the real-world. In *ASONAM*, pages 156–161, 2013.

[12] H. Deng, I. King, and M. R. Lyu. Formal models for expert finding on dblp bibliography data. In *ICDM*, pages 163–172, 2008.

[13] G. Dror, N. Koenigstein, and Y. Koren. Web-scale media recommendation systems. *Proceedings of the IEEE*, 100(9): 2722–2736, 2012.

[14] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: applications and algorithms. *SIAM review*, 41 (4):637–676, 1999.

[15] A. Feragen, N. Kasenburg, J. Petersen, M. de Bruijne, and K. M. Borgwardt. Scalable kernels for graphs with continuous attributes. In *NIPS*, pages 216–224, 2013.

[16] T. Gärtner, P. A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *COLT*, pages 129–143, 2003.

[17] T. Gärtner, J. W. Lloyd, and P. A. Flach. Kernels and distances for structured data. *Machine Learning*, 57(3):205–232, 2004.

[18] G. Golub and C. Van Loan. Matrix computations. 1996.

[19] S. H. Hashemi, M. Neshati, and H. Beigy. Expertise retrieval in bibliographic network: a topic dominance learning approach. In *CIKM*, pages 1117–1126, 2013.

[20] S. Hido and H. Kashima. A linear-time graph kernel. In *ICDM*, pages 179–188, 2009.

[21] P. J. Hinds, K. M. Carley, D. Krackhardt, and D. Wholey. Choosing work group members: Balancing similarity, competence, and familiarity. In *Organizational Behavior and Human Decision Processes*, pages 226–251, 2000.

[22] T. Horváth, T. Gärtner, and S. Wrobel. Cyclic pattern kernels for predictive graph mining. In *KDD*, pages 158–167, 2004.

[23] U. Kang, H. Tong, and J. Sun. Fast random walk graph kernel. In *SDM*, pages 828–838, 2012.

[24] M. Karimzadehgan and C. Zhai. Constrained multi-aspect expertise matching for committee review assignment. In *CIKM*, pages 1697–1700, 2009.

[25] R. I. Kondor, N. Shervashidze, and K. M. Borgwardt. The graphlet spectrum. In *ICML*, page 67, 2009.

[26] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42 (8):30–37, 2009.

[27] T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. In *KDD*, pages 467–476, 2009.

[28] J.-Z. Li, J. Tang, J. Zhang, Q. Luo, Y. Liu, and M. Hong. Eos: expertise oriented search using social networks. In *WWW*, pages 1271–1272, 2007.

[29] P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Graph kernels for molecular structure-activity relationship analysis with support vector machines. *Journal of Chemical Information and Modeling*, 45(4):939–951, 2005.

[30] D. M. Mimno and A. McCallum. Expertise modeling for matching papers with reviewers. In *KDD*, pages 500–509, 2007.

[31] M. Muller, K. Ehrlich, T. Matthews, A. Perer, I. Ronen, and I. Guy. Diversity among enterprise online communities: collaborating, teaming, and innovating through social media. In *CHI*, pages 2815–2824, 2012.

[32] J. Pei, H. Wang, J. Liu, K. Wang, J. Wang, and P. S. Yu. Discovering frequent closed partial orders from strings. *IEEE Trans. Knowl. Data Eng.*, 18(11):1467–1481, 2006.

[33] S. S. Rangapuram, T. Bühler, and M. Hein. Towards realistic team formation in social networks based on densest subgraphs. In *WWW*, pages 1077–1088, 2013.

[34] N. Shervashidze and K. M. Borgwardt. Fast subtree kernels on graphs. *NIPS*, 2009.

[35] N. Shervashidze, S. V. N. Vishwanathan, T. Petri, K. Mehlhorn, and K. M. Borgwardt. Efficient graphlet kernels for large graph comparison. *Journal of Machine Learning Research - Proceedings Track*, 5:488–495, 2009.

[36] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

[37] H. A. Simon. *The Sciences of the Artificial (3rd Ed.)*. MIT Press, Cambridge, MA, USA, 1996. ISBN 0-262-69191-4.

[38] E. Tsivtsivadze, J. Urban, H. Geuvers, and T. Heskes. Semantic graph kernels for automated reasoning. In *SDM*, pages 795–803, 2011.

[39] S. V. N. Vishwanathan, K. M. Borgwardt, and N. N. Schraudolph. Fast computation of graph kernels. In *NIPS*, pages 1449–1456, 2006.

[40] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 99:1201–1242, 2010.

[41] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *KDD*, pages 448–456, 2011.

[42] S. Wu, J. Sun, and J. Tang. Patent partner recommendation in enterprise social networks. In *WSDM*, pages 43–52, 2013.

[43] R. Zadeh, A. D. Balakrishnan, S. B. Kiesler, and J. N. Cummings. What's in a move?: normal disruption and a design challenge. In *CHI*, pages 2897–2906, 2011.

[44] J. Zhang, M. S. Ackerman, and L. A. Adamic. Expertise networks in online communities: structure and algorithms. In *WWW*, pages 221–230, 2007.

[45] G. Zhou, S. Lai, K. Liu, and J. Zhao. Topic-sensitive probabilistic model for expert finding in question answer communities. In *CIKM*, pages 1662–1666, 2012.