Harnessing Teamwork in Networks:

Prediction, Optimization, and Explanation

by

Liangyue Li

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved October 2018 by the
Graduate Supervisory Committee:

Hanghang Tong, Chair
Chitta Baral
Huan Liu
Norbou Buchler

ARIZONA STATE UNIVERSITY

December 2018

ABSTRACT

Teams are increasingly indispensable to achievements in any organizations. Despite the organizations' substantial dependency on teams, fundamental knowledge about the conduct of team-enabled operations is lacking, especially at the *social, cognitive* and *information* level in relation to team performance and network dynamics. The goal of this dissertation is to create new instruments to *predict*, *optimize* and *explain* teams' performance in the context of composite networks (i.e., social-cognitive-information networks).

Understanding the dynamic mechanisms that drive the success of high-performing teams can provide the key insights into building the best teams and hence lift the productivity and profitability of the organizations. For this purpose, novel predictive models to forecast the long-term performance of teams (*point prediction*) as well as the pathway to impact (*trajectory prediction*) have been developed. A joint predictive model by exploring the relationship between team level and individual level performances has also been proposed.

For an existing team, it is often desirable to optimize its performance through expanding the team by bringing a new team member with certain expertise, or finding a new candidate to replace an existing under-performing member. I have developed graph kernel based performance optimization algorithms by considering both the structural matching and skill matching to solve the above enhancement scenarios. I have also worked towards real time team optimization by leveraging reinforcement learning techniques.

With the increased complexity of the machine learning models for predicting and optimizing teams, it is critical to acquire a deeper understanding of model behavior. For this purpose, I have investigated *explainable prediction* – to provide explanation behind a performance prediction and *explainable optimization* – to give reasons why

the model recommendations are good candidates for certain enhancement scenarios.

DEDICATION

This dissertation is dedicated to my parents.

ACKNOWLEDGMENTS

First of all, I want to express my deep gratitude to my advisor Dr. Hanghang Tong, for his advice, encouragement, patience, and support. This dissertation is impossible without the help from him. Prof. Tong led me to the exciting field of data mining and mentored me through every step of the way towards conducting excellent research: from finding a good research problem to coming up with innovative solutions, from writing good research papers to giving well-prepared presentations with dedication to every detail. I enjoyed every discussion with him and I have benefited tremendously from his sharp insights, wise suggestions and grand visions. More than that, he is also a great friend with a lot of caring. His encouragement during the moments when my research was not going as planned, when my paper got rejected, and during my job hunting makes my PhD journey much less stressful. He is hands down the best advisor ever.

I would like to thank my thesis committee members, Chitta Baral, Huan Liu and Norbou Buchler for their helpful suggestions and insightful comments. I took Dr. Baral's course on natural language processing, which broadened my horizon on the challenging tasks for understanding human languages. Dr. Liu, with his deep and broad knowledge in the field, always challenges me to see the problems in the big picture. Dr. Buchler is nothing but a great collaborator, who generously shares with me his deep understanding in the computational approaches to teaming.

I was very lucky to work as an intern in LinkedIn with amazing colleagues and mentors: Qi He, How Jing, Jaewon Yang, Bee-Chung Chen, Qin Wang, Siyuan Zhang. Special thanks to How, who was my mentor during the internship, for offering me tremendous help on my project.

During my PhD study, I have received consistent support and encouragement from my friends and colleagues. I am especially grateful to all the colleagues at the Data

Lab and Star Lab: Chen Chen, Xing Su, Si Zhang, Boxin Du, Qinghai Zhou, Jian Kang, Zhe Xu, Scott Freitas, Haichao Yu, Ruiyue Peng, Rongyu Lin, Xiaoyu Zhang, Dawei Zhou, Yao Zhou, Arun Reddy, Xu Liu, Xue Hu, Jun Wu, Lecheng Zheng, Pei Yang. I am also thankful to all my friends both at and outside of ASU, whom make the journey a pleasant one.

Finally, I am deeply indebted to my dear mother and father for their unconditional love and strong support during my graduate study in the US. This dissertation is dedicated to them.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER Page

CHAPTER Page

Chapter 1

INTRODUCTION

1.1 Motivations

In defining the essence of professional teamwork, Hackman and Katz [51] stated that teams function as 'purposive social systems', defined as people who are readily identifiable to each other by role and position and who work interdependently to accomplish one or more collective objectives. Teams are increasingly indispensable to achievement in any organization. This is perhaps most evident in multinational organizations where communication technology has transformed the geographically dispersed teams and networks. Business operations in the large organizations now involve large, interactive, and layered networks of teams and personnel communicating across hierarchies and countries during the execution of complex and multifaceted international businesses. Despite the organizations' substantial dependency on teams, fundamental knowledge about the conduct of team-enabled operations is lacking, especially at the *social, cognitive* and *information* level in relation to team performance and network dynamics. What do high-performing engineering/design/sale teams share in common with respect to their communication patterns? How to predict a team's performance before it starts to work on the assigned project? How to foster productive behavioral changes of team members and leaders in order to optimize performance?

1.2 Research Objectives and Key Challenges

Generally speaking, the **team performance** can be viewed as the **composite** of the following three aspects, including (1) its **users**, (2) **tasks** that the team performs and (3) the **networks** that the team is embedded in or operates on, i.e.,

$$\text{team performance} = f(\text{users, tasks, networks}) \tag{1.1}$$

The goal of my Ph.D. dissertation is to create new instruments to *predict*, *optimize* and *explain* teams' performance in the context of composite networks (i.e., social-cognitive-information networks). This research objective involves a number of key challenges, many of which can be attributed to **the complexity of teams**. Specifically, the complexity of the teams comes from all the following five components of Eq. (1.1).

- *Challenge 1: the complexity of the users.* There are three basic types of users, including the individuals (e.g., team members), team leaders (e.g., project managers), the "owners" of human resource (e.g., HR in an organization). While in general, different types of users are collaborative in nature, their goals are not always consistent with each other. For certain tasks, the team members or its leader might have to make a decision within a short time period, with incomplete and partial knowledge of its embedded environment/networks, and possibly under great stress.

- *Challenge 2: the complexity of tasks.* Within an organization, there are often multiple teams for a variety of different types of tasks, such as engineering teams, support teams, business teams, planning teams, etc. Each type of teams might have its own "secret recipe" for success. For example, a successful engineering team might heavily rely on its execution of plan, while a planning team might

need more innovation. Some tasks might be collaborative, while others might be competitive with each other. How to optimize the performance of a target team in the presence of an adversarial team? From an organization perspective, how to strengthen an existing team (e.g., by expanding the team size) without hurting others?

- *Challenge 3: the complexity of networks.* The challenges come from the environment that the team is embedded in or operates on, i.e., the fact that such networks are often *big*, meaning that they are large in size (*volume*), highly volatile in dynamics (*velocity*), spreading over multiple channels/layers/platforms (*variety*); and noisy and incomplete (*veracity*). In the dissertation, we assume the networks are undirected, but the proposed methods can be easily extended to handle the directionality of networks.

- *Challenge 4: the complexity of performance.* There is no single performance measure of the team, but rather a set of inter-correlated metrics. For example, the impact metrics for research teams include citation-based number of citations, h-index, online usage based view counts, download counts and network based centrality, all of which might be correlated with each other [12].

- *Challenge 5: the complexity of composite (e.g., $f()$).* The composite itself, which composes different aspects/metrics into the performance measure(s), is far-beyond a many-to-one linear process. Instead, it is likely to be a many (aspect) to many (performance measures) non-linear process.

1.3 Research Tasks Overview

In this dissertation, I take a multi-disciplinary approach, consisting of supervised learning, visualization and optimization, to tackle three complementary research

tasks.

Task 1: Team Performance Prediction. Understanding the dynamic mechanisms that drive the success of high-performing teams can provide the key insights into building the best teams and hence lift the productivity and profitability of the organizations. From the algorithmic perspective, the interesting problems are to forecast the long-term performance of teams (*point prediction*) as well as the pathway to impact (*trajectory prediction*). For research teams, early prediction of their performance has many important implications, ranging from personal career development and recruitment search, to the jurisdiction of research resources. The impact pathway often provides a good indicator of the shift of the research frontier and can also help trigger an early intervention should the impact trajectory step down in the near future. On the other hand, the ancient Greek philosopher Aristotle articulated more than 2000 years ago that "*the whole is more than the sum of its parts*", it is worthwhile to quantitatively examine the relationship between the team level and individual level performances and leverage that to build a joint predictive model.

Task 2: Team Performance Optimization. In this task, we focus on the problem of optimizing/enhancing an existing team. For example, if the team leader perceives the need to enhance certain expertise of the entire team, who shall we bring into the team (i.e., *team expansion*); if we need to reduce the size of an existing team (e.g., for the purpose of cost reduction), who shall leave the team (i.e., *team shrinkage*) so that the remaining team is least impacted; if the team leader sees a conflict between certain team members, how shall we resolve it (i.e., *team conflict resolution*); in case the desired team configuration changes over time, how to reflect such dynamics in the team enhancement process (i.e., *team evolution*)? We propose to solve all these enhancement scenarios based on a team member replacement algorithm we developed recently [75]. On the other hand, teams can be often viewed as a dynamic system.

We propose to plan the sequential optimization actions to maximize the cumulative performance using reinforcement learning.

Task 3: Team Performance Explanation. The basics of team effectiveness were identified by J. Richard Hackman, who uncovered a groundbreaking insight: what matter most to collaboration are certain enabling conditions. Recent studies found that three of Hackman's conditions – a compelling direction, a strong structure, and a supportive context – continue to be particularly critical to team success [50]. In this task, we aim to reveal the "secret recipe" for success by developing an explanation model for the above team performance prediction models as well as the performance optimization models. Such explanations can provide insights to *why* some teams are predicted to be successful and *why* we should bring a certain member to the team. Understanding the reasons behind predictions and recommendations is critical in assessing *trust*, which is especially fundamental if decisions (e.g., funding allocations) need to be made based on a prediction.

1.4 Impacts and Benefits

In the context of composite networks, this research will establish effective algorithms and tools for the performance prediction and optimization of teams along with explanations. This research will help organizations make a better decision to perform certain tasks that need collaborative effort within a team. Based on our work in this dissertation, we will build a system of team enhancement (i.e., prediction, optimization, explanation). The visualization component of this system can be used to track individual and team performance over time, and provide feedback to individuals to foster productive behavior change. To the best of our knowledge, this is the first comprehensive effort that integrates interactive visualization mechanisms, machine learning models and advanced network analysis algorithms for optimizing

teams. The preliminary results (e.g., publications, presentations and prototype systems) are available at `team-net-work.org`.

Chapter 2

LITERATURE SURVEY

In this chapter, we will briefly introduce some of the state-of-the-arts for performance/impact prediction, optimization, and explanation.

2.1 Literature Survey on Performance Prediction

Impact/popularity prediction: As a pilot study, Yan et al. [119, 118] identify effective features to address citation count prediction problem. Davletov et al. [34] address the same problem by first clustering papers according to their temporal change in citation counts over time and assigning a polynomial to each cluster for regression. In light of the difficulty posed by power law distribution of citations, Dong et al. [38] instead consider whether a paper can increase the primary author's h-index. Yu et al. [124] address predicting citation relations in heterogeneous bibliographical networks.

A close line of work is to predict the popularity of other online contents, e.g., posts, videos, TV series. Yao et al. [121] predict the long-term impact of questions/answers. Notice that in terms of methodology, the method in [121] can be conceptually viewed as a special case of our iBall model when there are only two domains and the instance-level correspondence across different domains (e.g., question-answers association) is known. Li et al. [71] conduct an study on popularity forecast of videos shared in social networks. They consider both the intrinsic attractiveness of a video and the influence from the underlying diffusion structure. Chang et al. [25] are the first to comprehensively study for predicting the popularity of online serials with autoregressive models. As online serials have strong sequence dependence and release date

dependence, they develop an autoregressive model to capture the dynamic behaviors of audiences. Though the focus of this research is to propose a tailored method to predict the long-term citation counts, our method could be naturally applied to other related applications, e.g., popularity prediction.

Multi-task learning: Our joint model iBall is also related to multi-task learning as we jointly learn the models for each domain (task). Multi-task learning aims to improve the generalization performance of a learning task with the help of other related tasks. A key challenge in multi-task learning is to exploit the relationship among different tasks to allow information shared across tasks. One way is by sharing of parameters. In neural networks, hidden units are shared across tasks [21, 85]. It can also be induced by assuming that the parameters used by all tasks are close to each other by minimizing the Frobenius norms of their differences in methods based on convex optimization formulations [43]. In Bayesian hierarchical models, parameter sharing can be imposed by assuming a common prior they share [123]. A second way is assuming a common basis of the parameter space. A low-rank and sparse structure of the underlying predictive hypothesis has been applied to capture the tasks relatedness as well as outlier tasks [28, 29, 57]. Our method is directly applicable when the correlation/similarity among different tasks is known and enjoys a closed-form solution. In terms of computation, we also provide an efficient way to track the joint predictive model in the dynamic setting.

Multi-label Learning. Multi-label learning is a machine learning paradigm where each data instance is associated with a set of labels. For example, in image classification, an image could be tagged as nature, ocean and sky; in document categorization, a text might belong to politics and foreign affairs. The algorithms developed for multi-label learning can be roughly categorized into two groups by a recent survey [129]: *problem transformation methods*, to fit data to existing algo-

rithms; and *algorithm adaptation methods*, to adapt existing learning technique to fit the multi-label data. In the first category, binary relevance [17] trains an individual classifier for each of the labels separately, which ignores label correlations and might suffer class imbalance issue. Classifier chains [97] on the other hand incrementally build classifier for each of the labels by augmenting the feature space using preceding predicted labels. The multi-label problems can be also modeled as a label ranking problem through the technique of pairwise comparison [45], essentially binary classifiers trained in one-vs-one fashion. In the second category, multi-label k-nearest neighbor algorithm [128] combines kNN and Bayesian reasoning to make prediction based on labeling information in the neighbors. Decision tree has also been adopted to handle multi-label data by computing the multi-label entropy [30]. Rank-SVM [42] employs maximum margin strategy to define linear models that minimize the ranking loss while having a large margin and enjoying non-linear extension through kernel trick. Recently, there is a line of work focused on exploiting the relationship among the labels to improve the learning performance. Zhang and Yeung [130] propose a probabilistic model for multi-label learning by assuming that the model parameters follow a matrix-variant normal distribution and the label relationship learning becomes solving for the column covariance matrix in the maximum a posteriori (MAP) solution. Huang and Zhou [56] notice that some label correlations are not shared globally and propose an approach that allows correlation sharing in a subset of instances. Ji et al. [59] assume that the model parameters share a low-dimensional subspace and formulate a regularized optimization problem.

2.2 Literature Survey on Team Performance Optimization

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 [67]. As follow-up work, Anagnostopoulos et al [2] studies forming teams to accommodate a sequence of tasks arriving in an online fashion and Rangapuram et al [96] allows incorporating many realistic requirements into team formation based on a generalization of the densest subgraph problem. Beyond that, minimizing the tensions among the team members is considered [48]. 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. Cao et al [20] develop an interactive group mining system that allows users to efficiently explore the network data and from which to progressively select and replace candidate members to form a team. Bogdanov et al [11] 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 [33] 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 [90]. The patterns of informal networks and communication in distributed global software teams using social network analysis is also investigated in [26]. Specific communication structures are proven critical to new product development delivery performance and quality [24]. To assess the skills of players and teams in online multi-player games and team-based sports, "team chemistry" is also accounted for in [36, 35].

Recommendation and Expert Finding. Recommendation 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 recom-

mendation (collaborative filtering) is latent factor model [66, 40, 113]. 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 [127, 16, 131]. 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 [88, 61], which can be done by either building the co-author network [72] or using language model and topic-based model [37, 52]. 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 [6, 116].

Graph Kernel. Graph kernel measures the similarity between two graphs. Typical applications include automated reasoning [108], bioinformatics and chemoinformatics [44, 102]. Generally speaking, graph kernels can be categorized into three classes: kernels based on walks [46, 111, 112, 47, 13], kernels based on limited-sized subgraphs [55, 103, 65] and kernels based on subtree patterns [86, 101, 53]. Graph kernels based on random walk is one of the most successful choices [14]. 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 for labelled graphs take $O(lr't^3)$ time by reducing to the problem of solving a linear system [111, 112]. With low rank approximation, the computation can be further accelerated with high approximation accuracy [60].

2.3 Literature Survey on Team Performance Explanation

Intelligible Models. Some machine learning models are inherently intelligible

and are proposed to strike a good balance between model complexity and intelligibility [83, 84, 22]. The Generalized Additive Models (GAMs) [83] assume that the target is a linear combination of potentially nonlinear single-feature models (i.e., shape functions). Popular shape functions are splines, decision trees and boosted trees. GA^{2}M models [84] incorporate the pairwise interactions between the features on top of GAMs. In addition, the sparse partially linear additive model (SPLAM) [82] is proposed to address two model selection challenges, i.e., what features to include in the model and which of these features should be treated nonlinearly. The limitations with this line of work is that the intelligible models might fail when the number of features grows into the millions.

Model Explanation. With increased complexity of machine learning models, many research efforts have been on acquiring a deeper understanding of model behavior. There are mainly two paths for explaining model's performances, namely, explaining through features and training samples. The first approach examines the importance of different features to model predictions. To work for any complicated model, LIME [98] is proposed as a model-agnostic explanation model by learning an interpretable model locally around the prediction for a specific test sample. In some cases, the features may have indirect influence to the model prediction via other related features. Such indirect influence can be quantified based on a differential analysis of feature influence before and after obscuring the feature influence on the model outcome [1]. The second popular approach to model interpretability is to generate explanations through the lens of training examples. Influence functions, as a classic technique from robust statistics, are used to trace a model's prediction through the learning algorithm back to its training data [64]. The key idea is to compute the change of the loss at a test sample should a training example is upweighted by some small ϵ. Graph signal process has also been used for influential sample analysis

where the influence metric is used as a function at the nodes in the data graph [4]. The most influential samples would be those critical to the recover of high frequency components of the function.

Chapter 3

TEAM PERFORMANCE PREDICTION

In this chapter, we introduce our work on team performance prediction, including long-term performance prediction [73] and performance trajectory forecasting [79]. We also explore the relationship between the team level and individual level performances and design a joint prediction model for the prediction of both. We would describe their problem definitions and the key ideas behind our solutions. We focus on research teams for the performance prediction purpose.

3.1 Long-term Performance Forecasting

Understanding the dynamic mechanisms that drive the high-impact scientific work (e.g., research papers, patents) is a long-debated research topic and has many important implications, ranging from personal career development and recruitment search, to the jurisdiction of research resources. Scholars, especially junior scholar, who could master the key to producing high-impact work would attract more attentions as well as research resources; and thus put themselves in a better position in their career developments. High-impact work remains as one of the most important criteria for various organization (e.g. companies, universities and governments) to identify the best talents, especially at their early stages. It is highly desirable for researchers to judiciously search the right literature that can best benefit their research.

Recent advances in characterizing and modeling scientific success have made it possible to forecast the long-term impact of scientific work. Wuchty et al. [117] observe that papers with multiple authors receive more citations than solo-authored ones. Uzzi et al. [110] find that the highest-impact science work is primarily grounded in

atypical combinations of prior ideas while embedding them in conventional knowledge frames. Recently, Wang et al. [114] develop a mechanistic model for the citation dynamics of individual papers. In data mining community, efforts have also been made to predict the long-term success. Carlos et al. [23] estimate the number of citations of a paper based on the information of past articles written by the same author(s). Yan et al. [119] design effective content (e.g., topic diversity) and contextual (e.g., author's *h*-index) features for the prediction of future citation counts. Despite much progress, the following four key algorithmic challenges in relation to predicting long-term scientific impact have largely remained open.

C1 *Scholarly feature design:* many factors could affect scientific work's long-term impact, e.g., research topics, author reputations, venue ranks, citation networks' topological features, etc. Among them, which bears the most predictive power?

C2 *Non-linearity:* the effect of the above scholarly features on the long-term scientific impact might be way beyond a linear relationship.

C3 *Domain heterogeneity:* the impact of scientific work in different fields or domains might behave differently; yet some closely related fields could still share certain commonalities. Thus, a one-size-fits-all or one-size-fits-one solution might be sub-optimal.

C4 *Dynamics:* with the rapid development of science and engineering, a significant number of new research papers are published each year, even on a daily basis with the advent of arXiv[1]. The predictive model needs to handle such stream-like data efficiently, to reflect the recency of the scientific work.

In this study, we propose a joint predictive model–Impact Crystal Ball (iBall in

[1] arxiv.org

short) – to forecast the long term scientific impact at an early stage by collectively addressing the above four challenges. First (for C1), we found that the citation history of a scholarly entity (e.g., paper, researcher, venue) in the first three years (e.g., since its publication date) is a strong indicator of its long-term impact (e.g., the accumulated citation count in ten years); and adding additional contextual or content features brings little marginal benefits in terms of prediction performance. This not only largely simplifies the feature design, but also enables us to forecast the long-term scientific impact at its early stage. Second (for C2), our joint predictive model is flexible, being able to characterize both the linear and non-linear relationship between the features and the impact score. Third (for C3), we propose to jointly learn a predictive model to differentiate distinctive domains, while taking into consideration the commonalities among these similar domains (see an illustration in Figure 3.1). Fourth (for C4), we further propose a fast on-line update algorithm to adapt our joint predictive model efficiently over time to accommodate newly arrived training examples (e.g., newly published papers).

Our main contributions can be summarized as follows:

- **Algorithms:** we propose a joint predictive model –iBall– for the long-term scientific impact prediction problem, together with its efficient solvers.
- **Proofs and analysis:** we analyze the correctness, the approximation quality and the complexity of our proposed algorithms.
- **Empirical evaluations:** we conduct extensive experiments to demonstrate the effectiveness and efficiency of our proposed algorithms.

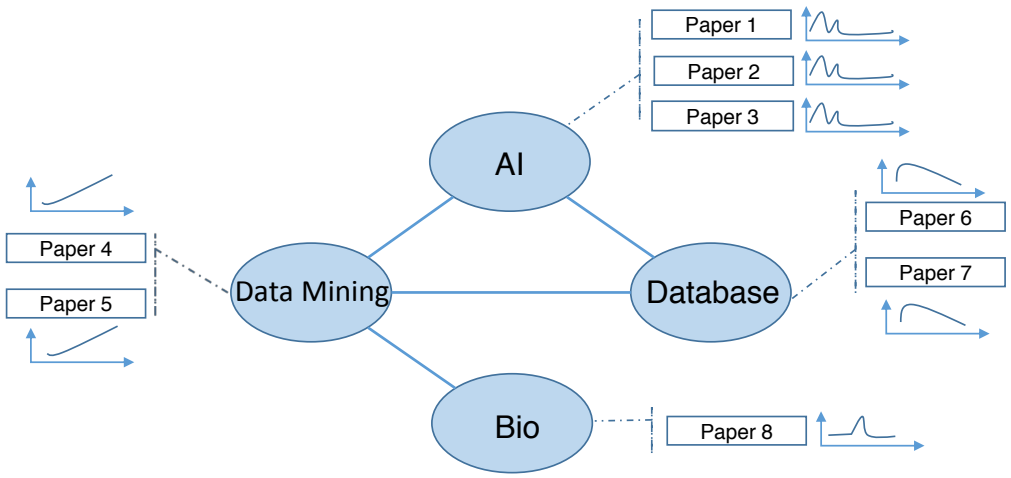


Figure 3.1: An illustrative example of the proposed joint predictive model. Papers from the same domain (e.g., AI, Databases, Data Mining and Bio) share similar patterns in terms of attracting citations over time. Certain domains (e.g., AI and Data Mining) are more related with each other than other domains (e.g., AI and Bio). We want to jointly learn four predictive models (one for each domain), with the goal of encouraging the predictive models from more related domains (e.g., AI and Data Mining) to be 'similar' with each other.

3.1.1 Problem Statement

In this section, we first present the notations and then formally define the long-term scientific impact prediction for scholarly entities (e.g., research papers, researchers, conferences).

Table 3.1 lists the main symbols used. We use bold capital letters (e.g., $\mathbf{A}$) for matrices, bold lowercase letters (e.g., $\mathbf{w}$) for vectors, and lowercase letters (e.g., λ) for scalars. For matrix indexing, we use a convention similar to Matlab as follows, e.g., we use $\mathbf{A}(i, j)$ to denote the entry at the i-th row and j-th column of a matrix $\mathbf{A}$, $\mathbf{A}(i, :)$ to denote the i-th row of $\mathbf{A}$ and $\mathbf{A}(:, j)$ to denote the j-th column of $\mathbf{A}$.

Symbols	Definition
n_d	number of domains
n_i	number of training samples in the i-th domain
m_i	number of new training samples in the i-th domain
d	feature dimensionality
$\mathbf{X_t^{(i)}}$	feature matrix of training samples from the i-th domain at time t
$\mathbf{x_{t+1}^{(i)}}$	feature matrix of new training samples from the i-th domain at time $t+1$
$\mathbf{Y_t^{(i)}}$	impact vector of training samples from the i-th domain at time t
$\mathbf{y_{t+1}^{(i)}}$	impact vector of new training samples from the i-th domain at time $t+1$
$\mathbf{A}$	adjacency matrix of domain relation graph
$\mathbf{w^{(i)}}$	model parameter for the i-th domain
$\mathbf{K^{(i)}}$	kernel matrix of training samples in the i-th domain
$\mathbf{K^{(ij)}}$	cross domain kernel matrix of training samples in the i-th and j-th domains

Table 3.1: Symbols for iBall

Besides, we use prime for matrix transpose, e.g., $\mathbf{A}'$ is the transpose of $\mathbf{A}$.

To differentiate samples from different domains at different time steps, we use superscript to index the domain and subscript to indicate timestamp. For instance, $\mathbf{X_t^{(i)}}$ denotes the feature matrix of all the scholarly entities in the i-th domain at time t and $\mathbf{x_{t+1}^{(i)}}$ denotes the feature matrix of new scholarly entities in the i-th domain at time $t+1$. Hence, $\mathbf{X_{t+1}^{(i)}} = [\mathbf{X_t^{(i)}}; \mathbf{x_{t+1}^{(i)}}]$. Similarly, $\mathbf{Y_t^{(i)}}$ denotes the impact vector of scholarly entities in the i-th domain at time t and $\mathbf{y_{t+1}^{(i)}}$ denotes the impact vector of

new scholarly entities in the i-th domain at time $t+1$. Hence, $\mathbf{Y}^{(\mathbf{i})}_{\mathbf{t+1}} = [\mathbf{Y}^{(\mathbf{i})}_{\mathbf{t}}; \mathbf{y}^{(\mathbf{i})}_{\mathbf{t+1}}]$. We will omit the superscript and/or subscript when the meaning of the matrix is clear from the context.

With the above notations, we are ready to define the long-term impact prediction problem in both static and dynamic settings as follows:

Problem 1. *Static Long-term Scientific Impact Prediction*

Given: *feature matrix* $\mathbf{X}$ *and impact* $\mathbf{Y}$ *of scholarly entities*

Predict: *the long-term impact of new scholarly entities*

We further define the dynamic impact prediction problem as:

Problem 2. *Dynamic Long-term Scientific Impact Prediction*

Given: *feature matrix* $\mathbf{X_t}$ *and new training feature matrix* $\mathbf{x_{t+1}}$ *of scholarly entities, the impact vector* $\mathbf{Y_t}$*, and the impact vector of new training samples* $\mathbf{y_{t+1}}$

Predict: *the long-term impact of new scholarly entities*

3.1.2 Empirical Observations

In this subsection, we perform an empirical analysis to highlight some of the key challenges on *AMiner* citation network [105]. This is a rich real dataset for bibliography network analysis and mining. The dataset contains 2,243,976 papers, 1,274,360 authors, and 8,882 computer science venues. For each paper, the dataset provides its titles, authors, references, publication venue and publication year. The papers date from year 1936 to 2013. In total, the dataset has 1,912,780 citation relationships extracted from ACM library.

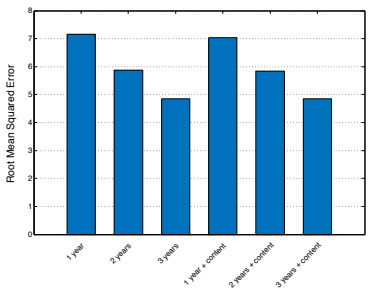


Figure 3.2: Prediction error comparison with different features.

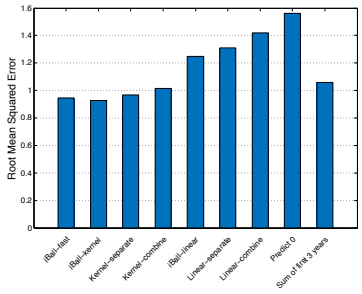


Figure 3.3: RMSE comparisons using different methods. The citation count is normalized in this figure. See Section 3.1.4 for normalization details.

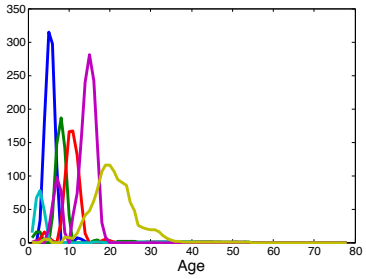


Figure 3.4: Visualization of papers' citation behavior. Different colors encodes different citation behaviors.

Feature design Prior work [23, 119] has proposed some effective features for citation count prediction, e.g., topic features (topic rank, diversity), author features (h-index, productivity), venue features (venue rank, venue centrality). Other work [114] make predictions only on the basis of the early years' citation data and find that the future impact of majority papers fall within the predicted citation range. We conduct experiment to compare performance of different features. Figure 3.2 shows the root mean squared error using different features with a regression model for the prediction of 10 years' citation count. For example, '3 years' means using the first 3 years' citation as feature, and '3 years + content' means using the first 3 years' citation along with content features (e.g., topic, author features). The result shows that adding content features (the right three bars in the figure) brings little improvement for citation prediction.

Non-linearity To see if the feature has linear relationship with the citation, we compare the performance of different methods using only the first 3 years' citation history. In Figure 3.3, the non-linear models (iBall-fast, iBall-kernel, Kernel-combine) all outperform the linear models (iBall-linear, Linear-separate, Linear-combine). See Section 3.1.3 and 3.1.4 for details of these models. It is clear that complex relationship between the features and the impact cannot be well characterized by a simple linear model - the prediction performance for all the linear models is even worse than the baseline method (using the summation of the first 3 years' citation counts).

Domain heterogeneity To get a sense of the dynamic patterns of the citation count, we construct a paper-age citation matrix $\mathbf{M}$, where $\mathbf{M_{ij}}$ indicates the number of citations the i-th paper receives in the j-th year after it gets published. The matrix $\mathbf{M}$ is then factorized as $\mathbf{M} \approx \mathbf{WH}$ using Non-negative Matrix Factorization (NMF) [70]. We visualize the first six rows of $\mathbf{H}$ in Figure 3.4, which can give us different clustering citation dynamic patterns. As can be seen from the figure, the cyan line has a very small peak in the first 3 years and then fades out very quickly; the blue line picks up very fast in the early years and then fades out; the yellow line indicates a delayed pattern where the scientific work only receives some amount of attentions decades after it gets published. This highlights that impact of scientific work from different domains behaves differently.

3.1.3 Proposed Algorithms

In this subsection, we present our joint predictive model to forecast the long-term scientific impact at an early stage. We first formulate it as a regularized optimization problem; then propose effective, scalable and adaptive algorithms; followed up by theoretical analysis in terms of the optimality, the approximation quality as well as

the computational complexity.

iBall – Formulations Our predictive model applies to different types of scholarly entities (e.g., papers, researchers and venues). For the sake of clarity, we will use paper citation prediction as an example. As mentioned earlier, research papers are in general from different domains. We want to jointly learn a predictive model for each of the domains, with the design objective to leverage the commonalities between related domains. Here, the commonalities among different domains is described by a non-negative $\mathbf{A}$, i.e., if the i-th and j-th domains are closely related, its corresponding $\mathbf{A}_{ij}$ entry will have a higher numerical value. Denote feature matrix for papers in the i-th domain by $\mathbf{X^{(i)}}$, citation count of papers in the i-th domain by $\mathbf{Y^{(i)}}$ and the model parameter for the i-th domain by $\mathbf{w^{(i)}}$, we have the following joint predictive model:

$$\min_{\mathbf{w^{(i)}}, i=1,\ldots,n_d} \sum_{i=1}^{n_d} \mathcal{L}[f(\mathbf{X^{(i)}}, \mathbf{w^{(i)}}), \mathbf{Y^{(i)}}] + \theta \sum_{i=1}^{n_d} \sum_{j=1}^{n_d} \mathbf{A_{ij}} g(\mathbf{w^{(i)}}, \mathbf{w^{(j)}}) + \lambda \sum_{i=1}^{n_d} \Omega(\mathbf{w^{(i)}}) \tag{3.1}$$

where $f(\mathbf{X^{(i)}}, \mathbf{w^{(i)}})$ is the prediction function for the i-th domain, $\mathcal{L}(.)$ is a loss function, $g(\mathbf{w^{(i)}}, \mathbf{w^{(j)}})$ characterizes the relationship between the model parameters of the i-th and j-th domains, $\Omega(\mathbf{w^{(i)}})$ is the regularization term for model parameters and θ, λ are regularization parameters to balance the relative importance of each aspect.

As can be seen, this formulation is quite flexible and general. Depending on the loss function we use, our predictive model can be formulated as regression or classification task. Depending on the prediction function we use, we can have either linear or non-linear models. The core of our joint model is the second term that relates parameters of different models. If $\mathbf{A}_{ij}$ is large, meaning the i-th and j-th domains are closely related to each other, we want the function value $g(.)$ that characterizes the relationship between the parameters to be small.

iBall– linear formulation: if the feature and the output can be characterized by a linear relationship, we can use a linear function as the prediction function and the Euclidean distance for the distance between model parameters. The linear model can be formulated as follows:

$$\min_{\mathbf{w^{(i)}},i=1,\ldots,n_d} \quad \sum_{i=1}^{n_d} \|\mathbf{X^{(i)}}\mathbf{w^{(i)}} - \mathbf{Y^{(i)}}\|_2^2 + \theta \sum_{i=1}^{n_d}\sum_{j=1}^{n_d} \mathbf{A_{ij}}\|\mathbf{w^{(i)}} - \mathbf{w^{(j)}}\|_2^2 + \lambda \sum_{i=1}^{n_d} \|\mathbf{w^{(i)}}\|_2^2 \tag{3.2}$$

where θ is a balance parameter to control the importance of domain relations, and λ is a regularization parameter. From the above objective function we can see that, if the i-th domain and j-th domain are closely related, i.e., $\mathbf{A}_{ij}$ is a large positive number, it encourages a smaller Euclidean distance between $\mathbf{w^{(i)}}$ and $\mathbf{w^{(j)}}$. The intuition is that for a given feature, it would have a similar effect in predicting the papers from two similar/closely related domains.

iBall– non-linear formulation: As indicated in our empirical studies (Figure 3.3), the relationship between the features and the output (citation counts in ten years) is far beyond linear. Thus, we further develop the kernelized counterpart of the above linear model. Let us denote the kernel matrix of papers in the i-th domain by $\mathbf{K^{(i)}}$, which can be computed as $\mathbf{K^{(i)}}(a,b) = k(\mathbf{X^{(i)}}(a,:), \mathbf{X^{(i)}}(b,:))$, where $k(\cdot,\cdot)$ is a kernel function that implicitly computes the inner product in a high-dimensional reproducing kernel Hilbert space (RKHS) [5] . Similarly, we define the cross-domain kernel matrix by $\mathbf{K^{(ij)}}$, which can be computed as $\mathbf{K^{(ij)}}(a,b) = k(\mathbf{X^{(i)}}(a,:), \mathbf{X^{(j)}}(b,:))$, reflecting the similarity between papers in the i-th domain and j-th domain. Different from the linear model where the model parameters in different domains share the same dimensionality (i.e., the dimensionality of the raw feature), in the non-linear case, the dimensionality of the model parameters are the same as the number of training samples in each domain, which is very likely to be different across different domains. Thus, we cannot use the same distance function for $g(.)$. To address this issue, the key is to realize that the predicted value of a test sample using kernel methods is

a linear combination of the similarities between the test sample and all the training samples. Therefore, instead of restricting the model parameters to be similar, we impose the constraint that the predicted value of a test sample using the training samples in its own domain and using training samples in a closely related domain to be similar. The resulting non-linear model can be formulated as follows:

$$\min_{\mathbf{w^{(i)}}, i=1,\ldots,n_d} \sum_{i=1}^{n_d} \|\mathbf{K^{(i)}w^{(i)}} - \mathbf{Y^{(i)}}\|_2^2 + \theta \sum_{i=1}^{n_d}\sum_{j=1}^{n_d} \mathbf{A_{ij}} \|\mathbf{K^{(i)}w^{(i)}} - \mathbf{K^{(ij)}w^{(j)}}\|_2^2 + \lambda \sum_{i=1}^{n_d} \mathbf{w^{(i)\prime}K^{(i)}w^{(i)}} \tag{3.3}$$

where θ is a balance parameter to control the importance of domain relations, and λ is a regularization parameter. From the above objective function we can see that, if the i-th domain and j-th domain are closely related, i.e., $\mathbf{A}_{ij}$ is a large positive number, the predicted value of papers in the i-th domain computed using training samples from the i-th domain ($\mathbf{K^{(i)}w^{(i)}}$) should be similar to that using training samples from the j-th domain ($\mathbf{K^{(ij)}w^{(j)}}$).

iBall – Closed-form Solutions It turns out that both iBall linear and non-linear formulations have the following closed-form solutions:

$$\mathbf{w} = \mathbf{S}^{-1}\mathbf{Y} \tag{3.4}$$

iBall linear formulation. In the linear case, we have that $\mathbf{w} = [\mathbf{w^{(1)}}; \ldots; \mathbf{w^{(n_d)}}]$, $\mathbf{Y} = [\mathbf{X^{(1)\prime}Y^{(1)}}; \ldots; \mathbf{X^{(n_d)\prime}Y^{(n_d)}}]$, and $\mathbf{S}$ is a block matrix composed of $n_d \times n_d$ blocks, each of size $d \times d$, where d is the feature dimensionality. $\mathbf{S}$ can be computed as follows:

$$\begin{array}{c} \\ \text{i-th block row} \end{array} \begin{array}{c} \begin{array}{ccc} \text{i-th block column} & & \text{j-th block column} \end{array} \\ \begin{bmatrix} \ldots & \ldots & \ldots \\ \ldots & \mathbf{X^{(i)\prime}X^{(i)}} + (\theta \sum_{j=1}^{n_d} \mathbf{A_{ij}} + \lambda)\mathbf{I} & -\theta \mathbf{A_{ij}}\mathbf{I} \\ \ldots & \ldots & \ldots \end{bmatrix} \end{array} \tag{3.5}$$

iBall non-linear formulation. In the non-linear case, we have that $\mathbf{w} = [\mathbf{w^{(1)}}; \ldots; \mathbf{w^{(n_d)}}]$, $\mathbf{Y} = [\mathbf{Y^{(1)}}; \ldots; \mathbf{Y^{(n_d)}}]$, and $\mathbf{S}$ is a block matrix composed of $n_d \times n_d$ blocks with the (i, j)-th block of size $n_i \times n_j$, where n_i is the number of training samples in the i-th domain. $\mathbf{S}$ can be computed as follows:

$$\begin{matrix} & \text{i-th block column} & \text{j-th block column} & \\ \begin{bmatrix} \ldots & \ldots & \ldots \\ \ldots & (1 + \theta \sum_{j=1}^{n_d} \mathbf{A_{ij}})\mathbf{K^{(i)}} + \lambda \mathbf{I} & -\theta \mathbf{A_{ij}} \mathbf{K^{(ij)}} \\ \ldots & \ldots & \ldots \end{bmatrix} & \begin{matrix} \text{i-th block} \\ \\ \text{row} \end{matrix} \end{matrix} \tag{3.6}$$

iBall – Scale-up with Dynamic Update The major computation cost for the closed-form solutions lies in the matrix inverse $\mathbf{S}^{-1}$. In the linear case, the size of $\mathbf{S}$ is $(dn_d) \times (dn_d)$; and so its computational cost is manageable. However, this is not the case for non-linear closed-form solution since the matrix $\mathbf{S}$ in Eq. (3.6) is of size $n \times n$, where $n = \sum_{i=1}^{n_d} n_i$, which is the number of all the training samples. It would be very expensive to store this dense matrix ($O(n^2)$ space) and to compute its inverse ($O(n^3)$ time); especially when the number of training samples is very large, and the model receives new training examples constantly over time (dynamic update). In this subsection, we devise an efficient algorithm to scale up the non-linear closed-form solution and efficiently update the model to accommodate the new training samples over time. The key of the iBall algorithm is to use the low-rank approximation of the $\mathbf{S}$ matrix to approximate the original $\mathbf{S}$ matrix to *avoid* the matrix inversion; and at each time step, efficiently update the low-rank approximation itself.

After new papers in all the domains are seen at time step $t + 1$, the new $\mathbf{S}_{t+1}$

computed by Eq. (3.6) becomes:

$$\begin{bmatrix} & \text{i-th block column} & \text{j-th block column} & \\ \dots & \dots & \dots & \\ \dots & (1+\theta\sum_{j=1}^{n_d}\mathbf{A_{ij}})\mathbf{K_{t+1}^{(i)}}+\lambda\mathbf{I} & -\theta\mathbf{A_{ij}}\mathbf{K_{t+1}^{(ij)}} & \text{i-th block row} \\ \dots & \dots & \dots & \end{bmatrix} \tag{3.7}$$

where $\mathbf{K_{t+1}^{(i)}}$ is the new within-domain kernel matrix for the i-th domain and $\mathbf{K_{t+1}^{(ij)}}$ is the new cross domain kernel matrix for the i-th and j-th domains. The two new kernel matrix can be computed as follows:

$$\mathbf{K_{t+1}^{(i)}}=\begin{bmatrix}\mathbf{K_t^{(i)}} & (\mathbf{k_{t+1}^{(i)}})' \\ \mathbf{k_{t+1}^{(i)}} & \mathbf{h_{t+1}^{(i)}}\end{bmatrix} \quad \mathbf{K_{t+1}^{(ij)}}=\begin{bmatrix}\mathbf{K_t^{(ij)}} & \mathbf{k_{t+1}^{(ij_*)}} \\ \mathbf{k_{t+1}^{(i_*j)}} & \mathbf{h_{t+1}^{(i_*j_*)}}\end{bmatrix} \tag{3.8}$$

where $\mathbf{k_{t+1}^{(i)}}$ is the matrix characterizing the similarity between new training samples and old training samples and can be computed as: $\mathbf{k_{t+1}^{(i)}}(a,b)=k(\mathbf{x_{t+1}^{(i)}}(a,:),\mathbf{X_t^{(i)}}(b,:))$; $\mathbf{h_{t+1}^{(i)}}$ is the similarity matrix among new training samples and can be computed as: $\mathbf{h_{t+1}^{(i)}}(a,b)=k(\mathbf{x_{t+1}^{(i)}}(a,:),\mathbf{x_{t+1}^{(i)}}(b,:))$. $\mathbf{k_{t+1}^{(i_*j)}}$ is the matrix characterizing the similarity between new training samples in the i-th domain and old training samples in the j-th domain and can be computed as: $\mathbf{k_{t+1}^{(i_*j)}}(a,b)=k(\mathbf{x_{t+1}^{(i)}}(a,:)),\mathbf{X_t^{(j)}}(b,:)$. Similarly, $\mathbf{k_{t+1}^{(ij_*)}}$ measures the similarity between old training samples in the i-th domain and new training samples in the j-th domain and can be computed as: $\mathbf{k_{t+1}^{(ij_*)}}=k(\mathbf{X_t^{(i)}}(a,:),\mathbf{x_{t+1}^{(j)}}(b,:))$; $\mathbf{h_{t+1}^{(i_*j_*)}}$ is the similarity matrix between new training samples from both i-th and j-th domains and is computed as: $\mathbf{h_{t+1}^{(i_*j_*)}}=k(\mathbf{x_{t+1}^{(i)}}(a,:),\mathbf{x_{t+1}^{(j)}}(b,:))$.

Given that $\mathbf{S}_t$ is a symmetric matrix, we can approximate it using top-r eigen-decomposition as: $\mathbf{S}_t\approx\mathbf{U}_t\mathbf{\Lambda}_t\mathbf{U}_t'$, where $\mathbf{U}_t$ is an $n\times r$ orthogonal matrix and $\mathbf{\Lambda}_t$ is an $r\times r$ diagonal matrix with the largest r eigenvalues of $\mathbf{S}_t$ on the diagonal. If we can directly update the eigen-decomposition of $\mathbf{S}_{t+1}$ after seeing the new training

samples from all the domains, we can efficiently compute the new model parameters as follows:

$$\mathbf{w}_{t+1} \quad = \mathbf{S}_{t+1}^{-1}\mathbf{Y}_{t+1} = \mathbf{U}_{t+1}\mathbf{\Lambda}_{t+1}^{-1}\mathbf{U}_{t+1}'\mathbf{Y}_{t+1} \tag{3.9}$$

where $\mathbf{Y}_{t+1} = [\mathbf{Y_t^{(1)}}; \mathbf{y_{t+1}^{(1)}}; \ldots; \mathbf{Y_t^{(n_d)}}; \mathbf{y_{t+1}^{(n_d)}}]$. Here, $\mathbf{\Lambda}_{t+1}^{-1}$ a $r \times r$ diagonal matrix, whose diagonal entries are the reciprocals of the corresponding eigenvalues of $\mathbf{\Lambda}_{t+1}$. In this way, we avoid the computationally costly matrix inverse in the closed-form solution.

Compare $\mathbf{S}_{t+1}$ with $\mathbf{S}_t$, we find that $\mathbf{S}_{t+1}$ can be obtained by inserting into $\mathbf{S}_t$ at the right positions with some rows and columns of the kernel matrices involving new training samples, i.e.,$\mathbf{k_{t+1}^{(i)}}$, $\mathbf{h_{t+1}^{(i)}}$,$\mathbf{k_{t+1}^{(i_*j)}}$,$\mathbf{k_{t+1}^{(ij_*)}}$,$\mathbf{k_{t+1}^{(i_*j_*)}}$. From this perspective, $\mathbf{S}_{t+1}$ can be seen as the sum of the following two matrices:

$$\underbrace{\begin{bmatrix} \cdots & \overset{\text{i-th block column}}{\cdots} & \overset{\text{j-th block column}}{\cdots} & \\ \cdots & \begin{bmatrix} \alpha_i\mathbf{K_t^{(i)}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} & \begin{bmatrix} -\theta\mathbf{A_{ij}}\mathbf{K_t^{(ij)}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} & \begin{matrix}\text{i-th block} \\ \\ \text{row}\end{matrix} \\ \cdots & \cdots & \cdots & \end{bmatrix}}_{\tilde{\mathbf{S}}_t} \tag{3.10}$$

$$+ \underbrace{\begin{bmatrix} \cdots & \overset{\text{i-th block column}}{\cdots} & \overset{\text{j-th block column}}{\cdots} & \\ \cdots & \begin{bmatrix} \mathbf{0} & \alpha_i(\mathbf{k_{t+1}^{(i)}})' \\ \alpha_i\mathbf{k_{t+1}^{(i)}} & \alpha_i\mathbf{h_{t+1}^{(i)}} + \lambda\mathbf{I} \end{bmatrix} & \begin{bmatrix} \mathbf{0} & -\theta\mathbf{A_{ij}}\mathbf{k_{t+1}^{(ij_*)}} \\ -\theta\mathbf{A_{ij}}\mathbf{k_{t+1}^{(i_*j)}} & -\theta\mathbf{A_{ij}}\mathbf{h_{t+1}^{(i_*j_*)}} \end{bmatrix} & \begin{matrix}\text{i-th block} \\ \\ \text{row}\end{matrix} \\ \cdots & \cdots & \cdots & \end{bmatrix}}_{\Delta\mathbf{S}} \tag{3.11}$$

$$\overset{\text{def}}{=} \tilde{\mathbf{S}}_t + \Delta\mathbf{S} \tag{3.12}$$

where we denote $1 + \theta\sum_{j=1}^{n_d}\mathbf{A}_{ij}$ by α_i. The top-r eigen-decomposition of $\tilde{\mathbf{S}}_t$ can be directly written out from that of $\mathbf{S}_t$ as: $\tilde{\mathbf{S}}_t \approx \tilde{\mathbf{U}}_t\mathbf{\Lambda}_t\tilde{\mathbf{U}}_t'$, where $\tilde{\mathbf{U}}_t$ can be obtained by

inserting into $\mathbf{U}_t$ corresponding rows of 0, the same row positions as we insert into $\mathbf{S}_t$ the new kernel matrices. We propose Algorithm 1 to update the eigen-decomposition of $\mathbf{S}_{t+1}$, based on the observation that $\mathbf{S}_{t+1}$ can be viewed as $\tilde{\mathbf{S}}_t$ perturbed by a low-rank matrix $\Delta\mathbf{S}$. In line 5 of Algorithm 1, the only difference between the partial QR decomposition and the standard one, is that since $\tilde{\mathbf{U}}_t$ is already orthogonal, we only need to perform the Gram-Schmidt procedure starting from the first column of $\mathbf{P}$.

Algorithm 1: Eigen update of $\mathbf{S}_{t+1}$

Input: (1)eigen pair of $\mathbf{S}_t$: $\mathbf{U}_t$, $\mathbf{\Lambda}_t$;

(2)feature matrices of new papers in each domain: $\mathbf{x}_{\mathbf{t+1}}^{(\mathbf{i})}, i = 1, \ldots, n_d$;

(3)adjacency matrix of domain relation graph $\mathbf{A}$;

(4)balance parameters θ, λ

Output: eigen pair of $\mathbf{S}_{t+1}$: $\mathbf{U}_{t+1}$, $\mathbf{\Lambda}_{t+1}$

1 Obtain $\tilde{\mathbf{U}}_t$ by inserting into $\mathbf{U}_t$ rows of 0 at the right positions ;

2 Compute $\mathbf{k}_{\mathbf{t+1}}^{(\mathbf{i})}$, $\mathbf{h}_{\mathbf{t+1}}^{(\mathbf{i})}$,$\mathbf{k}_{\mathbf{t+1}}^{(\mathbf{i_*j})}$,$\mathbf{k}_{\mathbf{t+1}}^{(\mathbf{ij_*})}$,$\mathbf{k}_{\mathbf{t+1}}^{(\mathbf{i_*j_*})}$ for $i = 1, \ldots, n_d, j = 1, \ldots, n_d$;

3 Construct sparse matrix $\Delta\mathbf{S}$;

4 Perform eigen decomposition of $\Delta\mathbf{S}$: $\Delta\mathbf{S} = \mathbf{P}\mathbf{\Sigma}\mathbf{P}'$;

5 Perform partial QR decomposition of $[\tilde{\mathbf{U}}_t, \mathbf{P}]$:$[\tilde{\mathbf{U}}_t, \Delta\mathbf{Q}]\mathbf{R} \leftarrow \mathrm{QR}(\tilde{\mathbf{U}}_t, \mathbf{P})$;

6 Set $\mathbf{Z} = \mathbf{R}[\mathbf{\Lambda}_t\ \mathbf{0}; \mathbf{0}\ \mathbf{\Sigma}]\mathbf{R}'$;

7 Perform full eigen decomposition of $\mathbf{Z}$: $\mathbf{Z} = \mathbf{V}\mathbf{L}\mathbf{V}'$;

8 Set $\mathbf{U}_{t+1} = [\tilde{\mathbf{U}}_t, \Delta\mathbf{Q}]\mathbf{V}$ and $\mathbf{\Lambda}_{t+1} = \mathbf{L}$;

9 **Return**: $\mathbf{U}_{t+1}$, $\mathbf{\Lambda}_{t+1}$.

Building upon Algorithm 1, we have the fast iBall algorithm (Algorithm 2) for scaling up the non-linear solution with dynamic model update.

Algorithm 2: iBall –scale-up with dynamic update

Input: (1)eigen pair of $\mathbf{S}_t$: $\mathbf{U}_t$, $\mathbf{\Lambda}_t$;

(2)feature matrices of new papers in each domain: $\mathbf{x}_{\mathbf{t+1}}^{(\mathbf{i})}, i = 1, \ldots, n_d$;

(3)citation count vectors of new papers in each domain: $\mathbf{y}_{\mathbf{t+1}}^{(\mathbf{i})}, i = 1, \ldots, n_d$;

(4)adjacency matrix of domain relation graph $\mathbf{A}$;

(5)balance parameters θ, λ

Output: (1) updated model parameters $\mathbf{w}_{t+1}$, (2) eigen pair of $\mathbf{S}_{t+1}$: $\mathbf{U}_{t+1}$, $\mathbf{\Lambda}_{t+1}$

1 Update the eigen-decomposition of $\mathbf{S}_{t+1}$ using Algorithm 1 as:

$\mathbf{S}_{t+1} \approx \mathbf{U}_{t+1}\mathbf{\Lambda}_{t+1}\mathbf{U}'_{t+1}$;

2 Compute the new model parameters: $\mathbf{w}_{t+1} = \mathbf{U}_{t+1}\mathbf{\Lambda}_{t+1}^{-1}\mathbf{U}'_{t+1}\mathbf{Y}_{t+1}$;

3 **Return**: $\mathbf{w}_{t+1}$, $\mathbf{U}_{t+1}$ and $\mathbf{\Lambda}_{t+1}$.

iBall – Proofs and Analysis In this subsection, we will provide some analysis regarding the optimality, the approximation quality as well as the computational complexity of our proposed algorithms.

A - Correctness of the closed-form solutions of the iBall linear and non-linear formulations: In Lemma 1, we prove that the closed-form solution given in Eq. (3.4) with $\mathbf{S}$ computed by Eq. (3.5) is the fixed-point solution to the linear formulation in Eq. (3.2) and the closed-form solution given in Eq. (3.4) with $\mathbf{S}$ computed by Eq. (3.6) is the fixed-point solution to the non-linear formulation in Eq. (3.3).

Lemma 1. *(Correctness of closed-form solution of the iBall linear and non-linear formulations.) For the closed-form solution given in Eq. (3.4), if* $\mathbf{S}$ *is computed by Eq. (3.5), it is the fixed-point solution to the objective function in Eq. (3.2); and if*

$\mathbf{S}$ *is computed by Eq. (3.6), it is the fixed-point solution to the objective function in Eq. (3.3).*

Proof. Omitted for brevity. See [74] for detail. □

B - Correctness of the eigen update of $\mathbf{S}_{t+1}$: The critical part of Algorithm 2 is the subroutine Algorithm 1 for updating the eigen-decomposition of $\mathbf{S}_{t+1}$. According to Lemma 2, the only place that approximation error occurs is the initial eigen-decomposition of $\mathbf{S}_0$. The eigen updating procedure won't introduce additional error.

Lemma 2. *(Correctness of Algorithm 1.) If $\mathbf{S}_t = \mathbf{U}_t\mathbf{\Lambda}_t\mathbf{U}_t'$ holds, Algorithm 1 gives the exact eigen-decomposition of $\mathbf{S}_{t+1}$.*

Proof. Omitted for brevity. See [80] for details. □

C - Approximation Quality: We analyze the approximation quality of Algorithm 2 to see how much the learned model parameters deviate from the parameters learned using the exact iBall non-linear formulation. The result is summarized in Theorem 1.

Theorem 1. *(Error bound of Algorithm 2.) In Algorithm 2, if $\frac{\sum_{i\notin\mathcal{H}}\lambda_t^{(i)}}{\sum_i \lambda_{t+1}^{(i)}} < 1$, the error of the learned model parameters is bounded by:*

$$\|\mathbf{w}_{t+1} - \hat{\mathbf{w}}_{t+1}\|_2 \leq \frac{\sum_{i\notin\mathcal{H}}\lambda_t^{(i)}}{(\sum_i \lambda_{t+1}^{(i)})^2(1-\delta)}\|\mathbf{Y}_{t+1}\|_2 \tag{3.13}$$

where $\mathbf{w}_{t+1}$ is the model parameter learned by the exact iBall non-linear formulation at time $t+1$, $\hat{\mathbf{w}}_{t+1}$ is the updated model parameter output by Algorithm 2 from time t to $t+1$, $\lambda_t^{(i)}$ and $\lambda_{t+1}^{(i)}$ are the largest i-th eigenvalues of $\mathbf{S}_t$ and $\mathbf{S}_{t+1}$ respectively, $\delta = \|(\tilde{\mathbf{U}}_t\mathbf{\Lambda}_t\tilde{\mathbf{U}}_t' + \Delta\mathbf{S})^{-1}(\tilde{\mathbf{S}}_t - \tilde{\mathbf{U}}_t\mathbf{\Lambda}_t\tilde{\mathbf{U}}_t')\|_F$, $\mathcal{H}$ is the set of integers between 1 and r, i.e., $\mathcal{H} = \{a|a \in [1, r]\}$.

Proof. Suppose we know the exact $\mathbf{S}_t$ at time t and its top-r approximation: $\hat{\mathbf{S}}_t = \mathbf{U}_t\mathbf{\Lambda}_t\mathbf{U}_t'$. After one time step, we can construct $\Delta\mathbf{S}$ and the exact $\mathbf{S}_{t+1}$ can be computed as $\mathbf{S}_{t+1} = \tilde{\mathbf{S}}_t + \Delta\mathbf{S}$. The model parameters learned by the exact non-linear model is:

$$\mathbf{w}_{t+1} \quad = \mathbf{S}_{t+1}^{-1}\mathbf{Y}_{t+1} = (\tilde{\mathbf{S}}_t + \Delta\mathbf{S})^{-1}\mathbf{Y}_{t+1} \tag{3.14}$$

If we allow approximation as in Algorithm 2, the approximated model parameter is:

$$\hat{\mathbf{w}}_{t+1} \quad = \hat{\mathbf{S}}_{t+1}^{-1}\mathbf{Y}_{t+1} = (\tilde{\mathbf{U}}_t\mathbf{\Lambda}_t\tilde{\mathbf{U}}_t' + \Delta\mathbf{S})^{-1}\mathbf{Y}_{t+1} \tag{3.15}$$

Denote $\tilde{\mathbf{S}}_t + \Delta\mathbf{S}$ by $\mathbf{B}$ and $\tilde{\mathbf{U}}_t\mathbf{\Lambda}_t\tilde{\mathbf{U}}_t' + \Delta\mathbf{S}$ by $\mathbf{C}$,we have the following:

$$\|\mathbf{B} - \mathbf{C}\|_F \quad = \|\tilde{\mathbf{S}}_t - \tilde{\mathbf{U}}_t\mathbf{\Lambda}_t\tilde{\mathbf{U}}_t'\|_F \leq \textstyle\sum_{i\notin\mathcal{H}} \lambda_t^{(i)} \tag{3.16}$$

where the last inequality is due to the following fact:

$$\begin{aligned} \|\textstyle\sum_i a_i\mathbf{u}_i\mathbf{u}_i'\|_F \quad &= \sqrt{\mathrm{tr}(\textstyle\sum_i a_i^2\mathbf{u}_i\mathbf{u}_i')} = \sqrt{\textstyle\sum_i a_i^2\mathrm{tr}(\mathbf{u}_i\mathbf{u}_i')} \\ &= \sqrt{\textstyle\sum_i a_i^2} \leq \textstyle\sum_i |a_i| \end{aligned} \tag{3.17}$$

Denote $\|\mathbf{C}^{-1}(\mathbf{B} - \mathbf{C})\|_F$ by δ, we know that

$$\delta \quad \leq \|\mathbf{C}^{-1}\|_F\|\mathbf{B} - \mathbf{C}\|_F \leq \frac{\sum_{i\notin\mathcal{H}}\lambda_t^{(i)}}{\sum_i \lambda_{t+1}^{(i)}} < 1 \tag{3.18}$$

From matrix perturbation theory [49], we will reach the following:

$$\begin{aligned} \|\mathbf{w}_{t+1} - \hat{\mathbf{w}}_{t+1}\|_2 \quad &= \|\mathbf{B}^{-1}\mathbf{Y}_{t+1} - \mathbf{C}^{-1}\mathbf{Y}_{t+1}\|_2 \\ &\leq \|\mathbf{B}^{-1} - \mathbf{C}^{-1}\|_F\|\mathbf{Y}_{t+1}\|_2 \\ &\leq \frac{\|\mathbf{C}^{-1}\|_F^2\|\mathbf{B}-\mathbf{C}\|_F}{1-\delta}\|\mathbf{Y}_{t+1}\|_2 \\ &\leq \frac{\sum_{i\notin\mathcal{H}}\lambda_t^{(i)}}{(\sum_i \lambda_{t+1}^{(i)})^2(1-\delta)}\|\mathbf{Y}_{t+1}\|_2 \end{aligned} \tag{3.19}$$

□

D - Complexities: Finally, we analyze the complexities of Algorithm 1 and Algorithm 2. In terms of time complexity, the savings are two-folds: (1) we only need

to compute the kernel matrices involving new training samples; (2) we avoid the time consuming large matrix inverse operation. In terms of space complexity, we don't need to maintain the huge $\mathbf{S}_t$ matrix, but instead store its top-r eigen pairs which is only of $O(nr)$ space.

Theorem 2. *(Complexities of Algorithm 1 and Algorithm 2.) Algorithm 1 takes* $O((n+m)(r^2+r'^2))$ *time and* $O((n+m)(r+r'))$ *space. Algorithm 2 also takes* $O((n+m)(r^2+r'^2))$ *time and* $O((n+m)(r+r'))$ *space, where* m *is total number of new training samples.*

Proof. Time complexity of Algorithm 1: Step 1-3 take $O(nm)$ time, where n is total number of training samples from previous step, and m is the total number of new training samples. Eigen decomposition of $\Delta\mathbf{S}$ in step 4 takes $O(nmr')$, where r' is the rank of $\Delta\mathbf{S}$, since $\Delta\mathbf{S}$ is sparse matrix with $O(nm)$ non-zero entries. QR decomposition in step 5 takes $O((n+m)r'^2)$ since we only need to start from the columns in $\mathbf{P}$. Step 6 and 7 both take $O((r+r')^3)$ time. The last line takes at most $O((n+m)(r+r')^2)$. The overall time complexity is $O((n+m)(r^2+r'^2))$.

Space complexity of Algorithm 1: The storage of eigen pairs requires $O((n+m)r)$ space. Step 1-3 take $O(mn)$ space. Eigen decomposition of $\Delta\mathbf{S}$ in step 4 takes $O((n+m)r')$ space. QR decomposition in step 5 needs $O((n+m)(r+r'))$ space. Step 6 and 7 take $O((r+r')^2)$ space and line 8 needs $O((n+m)(r+r'))$. The overall space complexity is $O((n+m)(r+r'))$.

Time complexity of Algorithm 2: Update eigen decomposition of $\mathbf{S}_{t+1}$ in step 1 takes $O((n+m)(r^2+r'^2))$ time and computing the new learning parameter in step 2 takes $O(n+m)r$ time. The overall time complexity is $O((n+m)(r^2+r'^2))$.

Space complexity of Algorithm 2: Update eigen decomposition of $\mathbf{S}_{t+1}$ in step 1 takes $O((n+m)(r+r'))$ and computing the new learing parameter in step 2 takes

$O((n+m)r)$ space. The overall space complexity is $O((n+m)(r+r'))$.

□

3.1.4 Experiments

In this subsection, we design and conduct experiments mainly to inspect the following aspects:

- *Effectiveness:* How accurate are the proposed algorithms for predicting scholarly entities' long-term impact?
- *Efficiency:* How fast are the proposed algorithms?

Experiment Setup We use the real-world citation network dataset AMiner[2] to evaluate our proposed algorithms. The statistics and empirical observations are described in Section 3.1.1. Our primary task is to predict a paper's citations after 10 years given its citation history in the first three years. Thus, we only keep papers published between year 1936 and 2000 to make sure they are at least 10 years old. This leaves us 508,773 papers. Given that the citation distribution is skewed, the 10-year citation counts are normalized to the range of $[0, 7]$. Our algorithm is also able to predict citation counts for other scholarly entities including researchers and venues. We keep authors whose research career (when they publish the first paper) begin between year 1960 and 2000 and venues that are founded before year 2002. This leaves us 315,340 authors and 3,783 venues.

For each scholarly entity, we represent it as a three dimensional feature vector, where the i-th dimension is the number of citations the entity receives in the i-th year after its life cycle begins (e.g., paper gets published, researchers publish the first paper). We build a k-nn graph ($k = 5$) among different scholarly entities; use METIS [62]

[2]http://arnetminer.org/billboard/citation

to partition the graph into balanced clusters; and treat each cluster as a domain. We set the domain number (n_d) to be 10 for both papers and researchers; and 5 for venues. The Gaussian kernel matrix of the cluster centroids is used to construct the domain-domain adjacency matrix $\mathbf{A}$.

To simulate the dynamic scenario where training samples come in stream, we start with a small initial training set and at each time step add new training samples to it. The training samples in each domain are sorted by starting year (e.g., publication year). In the experiment, for papers, we start with 0.1% initial training data and at each update add another 0.1% training samples. The last 10% samples are reserved as test samples, i.e., we always use information from older publications for the prediction of the latest ones. For authors, we start with 0.2% initial training data and at each update add another 0.2% training data and use the last 10% for testing. For venues, we start with 20%, add 10% at each update and use last 10% for testing.

The root mean squared error (RMSE) between the the actual citation and the predicted one is adopted for accuracy evaluation. All the experiments were performed on a Windows machine with four 3.5GHz Intel Cores and 256GB RAM.

Repeatability of Experimenal Results: The AMiner citation dataset is publicly available. We have released the code of the proposed algorithms through authors' website. For all the results reported in this section, we set $\theta = \lambda = 0.01$ in our joint predictive model. Gaussian kernel with $\sigma = 5.1$ is used in the non-linear formulations.

Effectiveness Results We perform the effectiveness comparisons of the following nine methods:

1 *Predict 0:* directly predict 0 for test samples since majority of the papers have 0 citations.

2 *Sum of the first 3 years:* assume the total number of citations doesn't change after three years.

3 *Linear-combine:* combine training samples of all the domains for training using linear regression model.

4 *Linear-separate:* train a linear regression model for each domain separately.

5 *iBall-linear:* jointly learn the linear regression models as in our linear formulation.

6 *Kernel-combine:* combine training samples of all the domains for training using kernel ridge regression model [99].

7 *Kernel-separate:* train a kernel ridge regression model for each domain separately.

8 *iBall-kernel:* jointly learn the kernel regression models as in our non-linear formulation.

9 *iBall-fast :* proposed algorithm for speeding up the joint non-linear model.

A - Overall paper citation prediction performance. The RMSE result of different methods for test samples from all the domains is shown in Figure 3.5. We have the following observations: (1) the non-linear methods (iBall-fast, iBall-kernel, Kernel-separate, Kernel-combine) outperform the linear methods (iBall-linear, Linear-separate, Linear-combine) and the straightforward 'Sum of first 3 years' is much better than the linear methods, which reflects the complex non-linear relationship between the features and the impact. (2) The performance of iBall-fast is very close to iBall-kernel and sometimes even better, which confirms the good approximation quality of the model update and the possible de-noising effect offered by the low-rank approximation. (3) The iBall family of joint models is better than their separate versions (Kernel-separate, Linear-

	Predict 0	Linear-combine	Linear-separate	iBall-linear	Sum of first 3 years	Kernel-combine	Kernel-separate	iBall-fast
iBall-kernel	0	5.53e-16	6.12e-17	1.16e-13	1.56e-219	1.60e-72	8.22e-30	3.39e-14

Table 3.2: p-value of statistical significance

separate). To evaluate the statistical significance, we perform a t-test using 1.4% of the training samples and show the p-values in Table 3.2. From the result, we see that the improvement of our method is significant. To investigate parameter sensitivity, we perform parametric studies with three parameters in iBall-fast, namely, θ, λ and r. Figure 3.8 shows that the proposed method is stable in a large range of the parameter space.

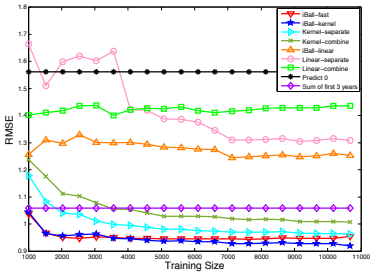


Figure 3.5: Overall paper citation prediction performance comparisons. Lower is better.

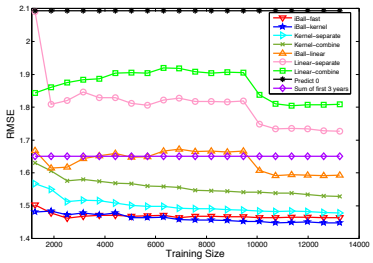


Figure 3.6: Author citation prediction performance comparison. Lower is better.

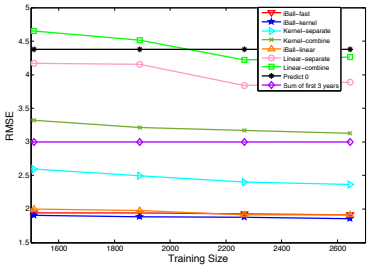
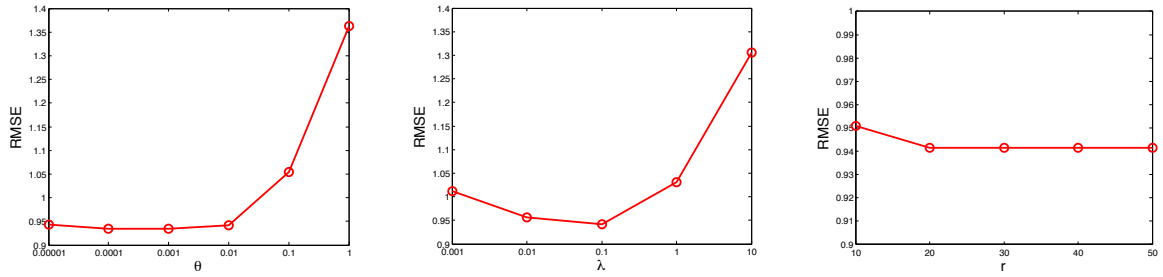


Figure 3.7: Venue citation prediction performance comparison. Lower is better.

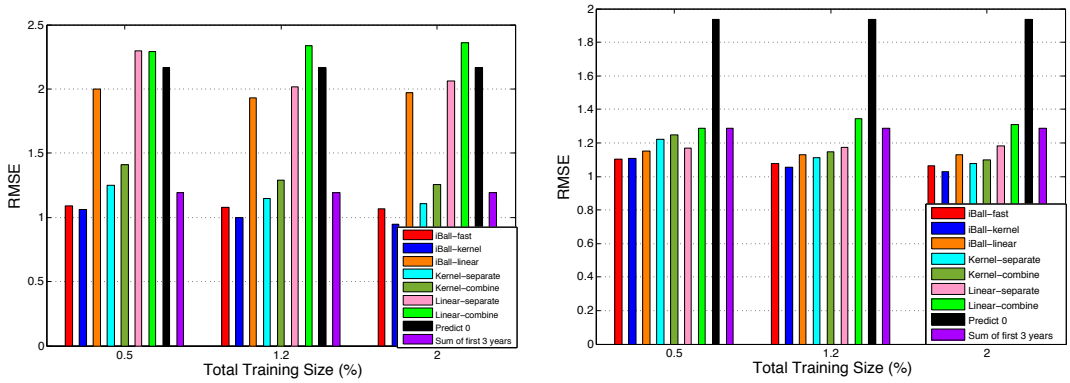
B - Domain-by-domain paper citation prediction performance. In Figure 3.9 we show the RMSE comparison results for two domains with different total training sizes. iBall-kernel and its fast version iBall-fast consistently outperform other methods in both of the domains. In the first domain, some linear methods (Linear-separate and Linear-combine) perform even worse than the baseline ('Predict 0').

C - Prediction error analysis. We visualize the actual citation vs. the predicted



(a) RMSE vs. θ (b) RMSE vs. λ (c) RMSE vs. r

Figure 3.8: Sensitivity study on iBall-fast: study the effect of the parameters θ, λ and r in terms of RMSE.



(a) Prediction performance comparison in the first domain.

(b) Prediction performance comparison in the second domain.

Figure 3.9: Paper citation prediction performance comparison in two domains.

citation using iBall as a heat map in Figure 3.10. The (x, y) square means among all the test samples with actual citation y, the percentage that have predicted citation x. We observe a very bright region near the $x = y$ diagonal. The prediction error mainly occurs in a bright strip at $x = 1, y \geq 1$. This is probably due to the delayed high-impact of some scientific work, as suggested by the blue and green lines in Figure 3.4, i.e., some papers only pick up attentions many years after they were published.

D - Author and venue citation prediction performance. We also show the RMSE com-

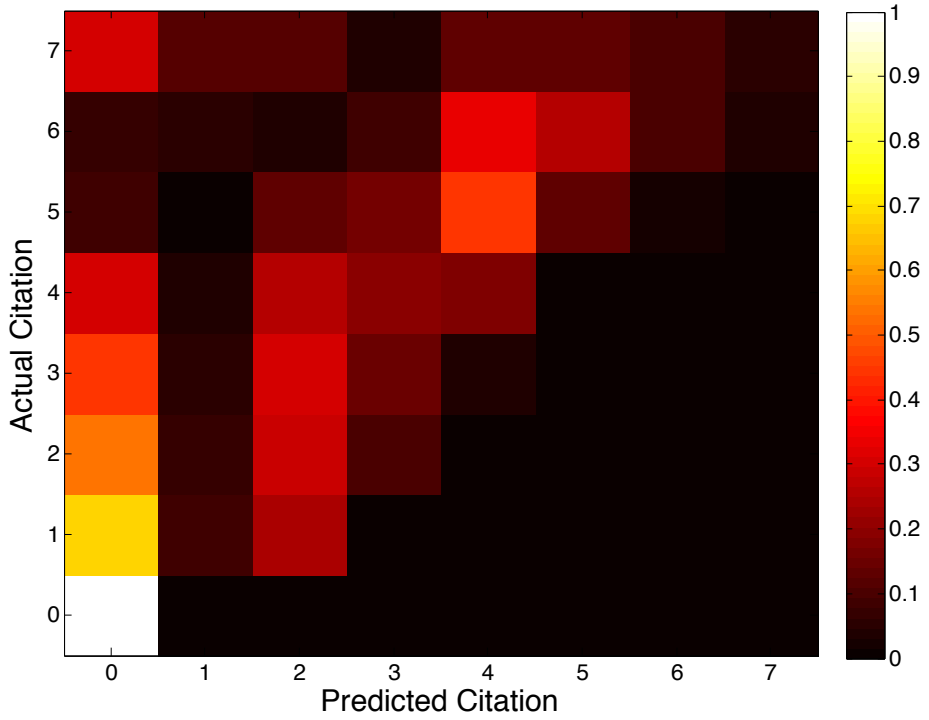


Figure 3.10: Prediction error analysis: actual citation vs. predicted citation. Best viewed in color.

parison results for the impact prediction of authors and venues in Figure 3.6 and 3.7 respectively. Similar observations can be made as the paper impact prediction, except that for the venue citation prediction, iBall-linear can achieve the similar performance as iBall-fast and iBall-kernel. This is probably due to the effect that venue citation (which involves the aggregation of the citations of all of its authors and papers) prediction is at a much coarser granularity, and thus a relatively simple linear model is sufficient to characterize the correlation between features and outputs (citation counts).

Efficiency Results

A - Running time comparison: We compare the running time of different methods with different training sizes and show the result in Figure 3.11 with time in log scale.

All the linear methods are very fast ($< 0.01s$) as the feature dimensionality is only 3. Our iBall-fast outperforms all other non-linear methods and scales linearly.

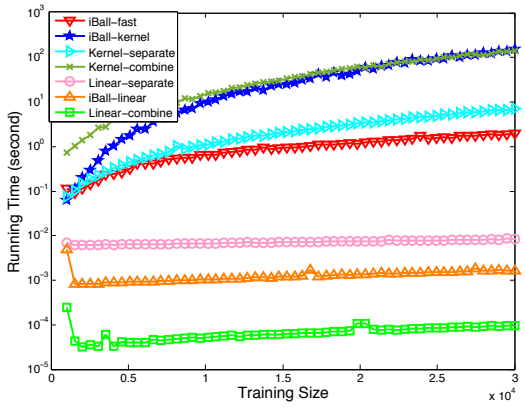


Figure 3.11: Comparison of running time of different methods. The time axis is of log scale.

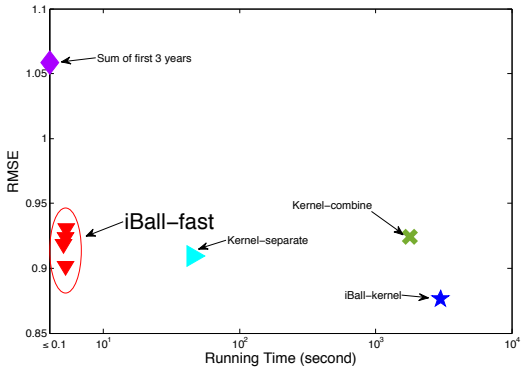


Figure 3.12: Quality vs. speed with 88,905 training samples.

B - Quality vs. speed: Finally, we evaluate how the proposed methods balance between the prediction quality and speed. In Figure 3.12, we show the RMSE vs. running time of different methods with 88,905 total training samples. For iBall-fast, we show its results using different rank r for the low-rank approximation. Clearly, iBall-fast achieves the best trade-off between quality and speed as its results all lie in the bottom left corner.

3.2 Performance Trajectory Forecasting

The emerging research area on the "science of science" (e.g., understanding the intrinsic mechanism that drives high-impact scientific work, foreseeing the success of scientific work at an early stage), has been attracting extensive research attention in the recent years, most of which are centered around the citation counts of the scholarly entities (e.g., researchers, venues, papers, institutes) [110, 73, 114]. From

the prediction perspective, more often than not, it is of key importance to forecast the pathway to impact for scholarly entities (e.g., how many citations a research paper will attract in each of several consecutive years in the future). The impact pathway often provides a good indicator of the shift of the research frontier. For instance, the rapid citation count increase of the deep learning papers reveals an emerging surge of this topic. The impact pathway can also help trigger an early intervention should the impact trajectory step down in the near future. Research resources could be more judiciously allocated if the impact pathway can be forecast at an early stage. For example, the research management agency could proactively allocate more resources to those rising fields.

The state of the art has mainly focused on modeling the long-term scientific impact for the early prediction. For example, Wang et al. [114] integrate preferential attachment, a temporal citation trend and the underlying "fitness" of the paper into designing a generative model for the citation dynamics of individual papers. Yan et al. [119] focus on designing effective scholarly features (e.g., content features, author features, venue features) for the future citation count prediction. Li et al. [73] propose a joint predictive model to encourage similar research domains to share similar model parameters.

Despite their own success, all the existing work on impact forecasting are essentially for *point prediction*, to predict the number of cumulative citations of a paper in the future. They are not directly applicable to forecasting the impact pathway, e.g., citation counts in each of the next 10 years. One baseline solution is to treat the impacts across different years independently and to train a separate model for each of the impacts. This treatment might ignore the inherent relationship among different impacts across different years, and thus might lead to sub-optimal performance. Having this in mind, a better way could be to apply the existing multi-label learning [130] or

multi-task learning [29] methods to exploit the relation among impacts across different years. Nonetheless, these general-purpose multi-label/multi-task learning approaches might overlook some unique characteristics of the impact pathway prediction, which is exactly the focus of this work.

In this work, we aim to develop a new predictive model tailored for scholarly entity impact pathway prediction. To be specific, our model will focus on the following two design objectives:

- **D1. Prediction Consistency.** Intuitively, the scholarly impacts at certain years might be correlated with each other, which, if vetted carefully, could boost the prediction performance (i.e., multi-label or multi-task learning). Here, one difficulty for impact pathway prediction is that such a relation structure is often not accurately known a priori. Thus a good predictive model should be capable of simultaneously inferring the impact relation structure from the training data and leveraging such (inferred) relation to improve the prediction performance.

- **D2. Parameter Smoothness.** For a given feature of the predictive model, we do not expect its effect on the impacts of adjacent years would change dramatically. For instance, the effect of "fitness" defined in [114], capturing a scientific work's perceived novelty and importance, is unlikely to change greatly but rather gradually fade away over years. A good predictive model should be able to capture such temporal smoothness.

We propose a new predictive model (*iPath*) to simultaneously fulfill these two design objectives. First, we propose to exploit the prediction consistency (i.e., D1) in the *output* space. Second, to encode the parameter smoothness (i.e., D2) between adjacent time steps, we impose a linear transition process in the *parameter space* from one time step to the next. We formulate it as a regularized optimization problem and

propose an effective alternating strategy to solve it. Our method is flexible, being able to handle both linear and non-linear models.

The main contributions can be summarized as follows:

- **Problem Definitions.** We define a novel scholarly impact pathway prediction problem, to predict the impact of a scholarly entity at several consecutive time steps in the future.
- **Algorithm and Analysis.** We propose and analyze a new predictive model (*iPath*) for the impact pathway forecasting problem.
- **Empirical Evaluations.** We conduct extensive experiments to validate the effectiveness of the proposed algorithm.

3.2.1 Problem Definition

In this subsection, we first present the notations used (summarized in Table 3.3) and then formally define the pathway to impact forecasting problem.

We use bold upper-case letters for matrices (e.g., $\mathbf{A}$), bold lowercase letters for vectors (e.g., $\mathbf{v}$), and lowercase letters (e.g., α) for scalars. For matrix indexing, we use a convention similar to Matlab's syntax as follows. We use $\mathbf{A}(i, j)$ to denote the entry at the intersection of the i-th row and j-th column of matrix $\mathbf{A}$, $\mathbf{A}(i, :)$ to denote the i-th row of $\mathbf{A}$ and $\mathbf{A}(:, j)$ to denote the j-th column of $\mathbf{A}$. Besides, we use prime for matrix transpose (e.g., $\mathbf{A}'$ is the transpose of $\mathbf{A}$).

For a given scholarly entity (e.g., research papers, researchers, conferences), after observing the impacts in the first few years, we want to forecast its impacts in the next several years (e.g., 10 or 20 years) into the future. Formally, denote $\mathbf{x} \in \mathbb{R}^d$ as the impacts observed in the first d time steps, we want to predict the impact pathway $\mathbf{y} = (y_1, y_2, \ldots, y_l)'$ afterwards, where y_i is the citation count in the i-th

Symbols	Definition
n	number of scholarly entities
d	feature dimension, i.e., number of time steps observed
l	length of the forecasting horizon into the future
$\mathbf{w}_i$	model parameter for predicting the i-th impact
$\mathbf{X}$	feature matrix
$\mathbf{Y}$	impact matrix
$\mathbf{A}$	adjacency matrix of the impact graph
$\mathbf{A}_0$	prior knowledge of the impact graph structure
$\mathbf{B}$	transition matrix
$\mathbf{K}$	kernel matrix
E	energy function
$\Phi_c(\cdot)$	the potential defined on a maximal clique c

Table 3.3: Symbols for *iPath*

future time step, and l is the length of the horizon we want to look into the future. Mathematically, the task is to learn a predictive function $f : \mathbf{x} \to \mathbf{y}$ from the training set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)|i = 1, 2, \ldots, n\}$, where n is the number of training samples. For convenience, let $\mathbf{X}$ be the feature matrix by stacking all the features (i.e., impact values of the first d time steps) of the n scholarly entities as its rows, that is, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]'$. Similarly, let $\mathbf{Y}$ be the impact matrix by stacking all the impacts (i.e., values of all the l future time steps) of the n scholarly entities as its rows, that is, $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n]'$.

With the above notations, we formally define the pathway to impact forecasting

problem as follows:

Problem 3. *Pathway to Impact Forecasting*

Given: *feature matrix* $\mathbf{X}$ *and impact matrix* $\mathbf{Y}$ *of* n *scholarly entities.*

Predict: *the impacts in each of the continuous future time steps of a new scholarly entity.*

Remarks: At the high-level, this problem setting bears some similarities to the classic multi-label learning [130] or multi-task learning [29] (i.e., predicting each impact is treated as a task). Nonetheless, the impact pathway of a scholarly entity brings several unique characteristics as outlined in the Introduction, which in turn calls for a new method to solve it.

3.2.2 Proposed Algorithms

In this subsection, we present a predictive model to forecast the pathway to impact. We first formulate it as a regularized optimization problem, and then propose an effective alternating optimization algorithm to solve it.

iPath **Formulations** Let us first summarize the key ideas behind our proposed formulation. First, we want to leverage the relation across the impacts at different time steps, so that closely related impacts are likely to have consistent predicted outputs. The relation among the impacts at different time steps is encoded in a non-negative matrix $\mathbf{A}$, where the entry $\mathbf{A}_{ij}$ is a large positive value if the i-th impact and j-th impact are closely related. The matrix $\mathbf{A}$ can be regarded as the weight matrix of the impact relationship graph, where vertices are impacts at different time steps and edge exists between two similar impacts. Second, one difficulty is that the impact relation might not be accurately known a prior. We address this by inferring

a good relation that can benefit the prediction performance, while not deviating too far from the (noisy) prior knowledge of the relation. Third, as we mentioned in the problem definition, we focus on the impact pathway forecasting, where the effect of features on the impacts at adjacent time steps is expected to transition smoothly. To realize such smoothness, we impose a linear transition process $\mathbf{B}$ between model parameters of adjacent time steps $\mathbf{w}_t$ and $\mathbf{w}_{t+1}$.

Putting all the above aspects together, our model can be formulated as follows:

$$\begin{aligned}\min_{\mathbf{W},\mathbf{B},\mathbf{A}} \quad & \underbrace{\mathcal{L}[f(\mathbf{X},\mathbf{W}),\mathbf{Y}]}_{\text{empirical loss}} + \underbrace{\alpha \sum_{i=1}^{l}\sum_{j=1}^{l} \mathbf{A}_{ij} g(\mathbf{w}_i,\mathbf{w}_j)}_{\text{prediction consistency}} + \underbrace{\beta \sum_{t=2}^{l} \|\mathbf{w}_t - \mathbf{B}\mathbf{w}_{t-1}\|_2^2}_{\text{parameter smoothness}} \\ & \underbrace{+ \gamma\|\mathbf{B}-\mathbf{I}\|_F^2 + \delta \sum_{i=1}^{l} \Omega(\mathbf{w}_i) + \epsilon\|\mathbf{A}-\mathbf{A}_0\|_F^2}_{\text{regularizations}}\end{aligned} \tag{3.20}$$

where $\mathbf{W}$ is the parameter matrix of the prediction parameters for all the impacts as $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_l]$; $f(\mathbf{X},\mathbf{W})$ is the prediction function, which could be linear or non-linear; $\mathcal{L}(\cdot)$ is the empirical loss between the predicted impacts and actual impacts; $g(\mathbf{w}_i,\mathbf{w}_j)$ characterizes the prediction consistency between the i-th impact and the j-th impact; $\|\mathbf{w}_t - \mathbf{B}\mathbf{w}_{t-1}\|_2^2$ instantiates the parameter smoothness; the rest terms are regularizations on $\mathbf{B}$, $\mathbf{W}$ and $\mathbf{A}$ respectively; $\mathbf{A}_0$ is the noisy prior knowledge about the impact/label relation; and α, β, γ, δ and ϵ are the trade-off parameters.

Remarks: the second term models the prediction consistency. If the i-th impact and the j-th impact are similar, i.e., $\mathbf{A}_{ij}$ is a large positive number, then the function value $g(\cdot)$ that measures the consistency between the predicted values for the i-th and j-th impacts should be small. In addition, to address the challenge of inferring a good relation, we are learning a relation $\mathbf{A}$ in the model by regularizing it not to deviate too far from our prior knowledge of the impact relation ($\mathbf{A}_0$). The third term models the parameter smoothness by assuming a linear transition process between model

parameters at two adjacent time steps. More specifically, the model parameter for time step t, $\mathbf{w}_t$ is close (in the form of Euclidean distance) to the model parameter for the last time step with some linear transition, $\mathbf{B}\mathbf{w}_{t-1}$. When $\mathbf{B}$ is an identity matrix, such smoothness will be a small Euclidean distance between the two parameters themselves. Our model will learn the model parameters $\mathbf{W}$, linear transition process $\mathbf{B}$ and the impacts relation $\mathbf{A}$ jointly.

iPath – linear formulation: in the linear case, the predictions are made by a linear weighted combination of the features, where the offset is absorbed by adding a constant to the feature. The linear model can be formulated as follows:

$$\begin{aligned}\min_{\mathbf{W},\mathbf{B},\mathbf{A}} \quad & \|\mathbf{X}\mathbf{W}-\mathbf{Y}\|_F^2+\alpha\sum_{i=1}^{l}\sum_{j=1}^{l}\mathbf{A}_{ij}\|\mathbf{X}\mathbf{w}_i-\mathbf{X}\mathbf{w}_j\|_2^2 \\ & +\beta\sum_{t=2}^{l}\|\mathbf{w}_t-\mathbf{B}\mathbf{w}_{t-1}\|_2^2+\gamma\|\mathbf{B}-\mathbf{I}\|_F^2+\delta\sum_{i=1}^{l}\|\mathbf{w}_i\|_2^2+\epsilon\|\mathbf{A}-\mathbf{A}_0\|_F^2\end{aligned} \tag{3.21}$$

In this linear formulation, if $\mathbf{A}_{ij}$ is a large positive number, meaning the i-th impact and the j-th impact are similar, then the predicted values for the i-th impact $\mathbf{X}\mathbf{w}_i$ and that for the j-th impact $\mathbf{X}\mathbf{w}_j$ are consistent.

iPath – non-linear formulation: in the non-linear case, the predicted impact is no longer a linear combination of the features, but the linear combination of the *similarities* between the test sample and all the training samples, where the similarities are expressed in the kernel matrix $\mathbf{K}$. The (i,j)-th entry of $\mathbf{K}$ can be computed as $\mathbf{K}(i,j)=\kappa(\mathbf{X}(i,:),\mathbf{X}(j,:))$, where $\kappa(\cdot,\cdot)$ is a kernel function that implicitly computes the inner product in the reproducing kernel Hilbert space (RKHS) [5]. The non-linear model can be formulated as follows:

$$\begin{aligned}\min_{\mathbf{W},\mathbf{B},\mathbf{A}} \quad & \|\mathbf{K}\mathbf{W}-\mathbf{Y}\|_F^2+\alpha\sum_{i=1}^{l}\sum_{j=1}^{l}\mathbf{A}_{ij}\|\mathbf{K}\mathbf{w}_i-\mathbf{K}\mathbf{w}_j\|_2^2 \\ & +\beta\sum_{t=2}^{l}\|\mathbf{w}_t-\mathbf{B}\mathbf{w}_{t-1}\|_2^2+\gamma\|\mathbf{B}-\mathbf{I}\|_F^2+\delta\sum_{i=1}^{l}\mathbf{w}_i'\mathbf{K}\mathbf{w}_i+\epsilon\|\mathbf{A}-\mathbf{A}_0\|_F^2\end{aligned} \tag{3.22}$$

From the objective function, we can see that if $\mathbf{A}_{ij}$ is a large positive number, meaning the i-th impact and the j-th impact are similar, then the predicted values

for the i-th impact $\mathbf{Kw}_i$ and that for the j-th impact $\mathbf{Kw}_j$ are consistent.

iPath **Optimization Solutions** In this subsection, we introduce an effective alternating optimization strategy to solve *iPath*. Since the optimization for linear and non-linear formulations are very similar, we will focus on the non-linear case and omit the linear case (referred to as *iPath*-lin) due to space limit. In non-linear case, we need to solve Eq. (3.22), which involves the optimization for $\mathbf{W}$, $\mathbf{B}$ and $\mathbf{A}$. We apply an alternating strategy and each time optimize for one group of variables while fixing the others. The details are as follows:

#1. Optimize for W while others are fixed: when others are fixed, the objective function becomes:

$$\begin{aligned}\min_{\mathbf{W}} \quad & \|\mathbf{KW}-\mathbf{Y}\|_F^2+\alpha\sum_{i=1}^{l}\sum_{j=1}^{l}\mathbf{A}_{ij}\|\mathbf{Kw}_i-\mathbf{Kw}_j\|_2^2 \\ & +\beta\sum_{t=2}^{l}\|\mathbf{w}_t-\mathbf{Bw}_{t-1}\|_2^2+\delta\sum_{i=1}^{l}\mathbf{w}_i'\mathbf{Kw}_i\end{aligned}$$

As it turns out, it has the following fixed point solution:

$$vec(\mathbf{W})=\mathbf{S}^{-1}vec(\mathbf{K}'\mathbf{Y}) \tag{3.23}$$

where $vec(\cdot)$ is the vectorization operation on a matrix by stacking the columns of a matrix into one column vector, and $\mathbf{S}$ is a block matrix composed of $l\times l$ blocks. The (i,j)-th block of $\mathbf{S}$, $\mathbf{S}_{ij}$ can be written as follows:

$$\mathbf{S}_{ii}=\begin{cases}(1+\alpha\sum_{j=1}^{l}\mathbf{A}_{ij})\mathbf{K}'\mathbf{K}+\beta\mathbf{B}'\mathbf{B}+\delta\mathbf{K}, & \text{if } i=1\\ (1+\alpha\sum_{j=1}^{l}\mathbf{A}_{ij})\mathbf{K}'\mathbf{K}+\delta\mathbf{K}, & \text{if } i=l\\ (1+\alpha\sum_{j=1}^{l}\mathbf{A}_{ij})\mathbf{K}'\mathbf{K}+\beta(\mathbf{I}+\mathbf{B}'\mathbf{B})+\delta\mathbf{K}, & \text{otherwise}\end{cases} \tag{3.24}$$

$$\mathbf{S}_{ij}=\begin{cases}-\alpha\mathbf{A}_{ij}\mathbf{K}'\mathbf{K}-\beta\mathbf{B}', & \text{if } i=j-1\\ -\alpha\mathbf{A}_{ij}\mathbf{K}'\mathbf{K}-\beta\mathbf{B}, & \text{if } i=j+1\\ -\alpha\mathbf{A}_{ij}\mathbf{K}'\mathbf{K}, & \text{otherwise}\end{cases} \tag{3.25}$$

#2. Optimize for B while others are fixed: when others are fixed, the objective function becomes:

$$\min_{\mathbf{B}} \quad \beta \sum_{t=2}^{l} \|\mathbf{w}_t - \mathbf{B}\mathbf{w}_{t-1}\|_2^2 + \gamma \|\mathbf{B} - \mathbf{I}\|_F^2$$

It has the following fixed point solution:

$$\mathbf{B} = (\beta \sum_{t=2}^{l} \mathbf{w}_t \mathbf{w}'_{t-1} + \gamma \mathbf{I})(\beta \sum_{t=2}^{l} \mathbf{w}_{t-1} \mathbf{w}'_{t-1} + \gamma \mathbf{I})^{-1} \tag{3.26}$$

#3. Optimize for A while others are fixed: when others are fixed, the objective function becomes:

$$\min_{\mathbf{A}} \quad \alpha \sum_{i=1}^{l} \sum_{j=1}^{l} \mathbf{A}_{ij} \|\mathbf{K}\mathbf{w}_i - \mathbf{K}\mathbf{w}_j\|_2^2 + \epsilon \|\mathbf{A} - \mathbf{A}_0\|_F^2$$

It has the following fixed point solution:

$$\mathbf{A} = \mathbf{A}_0 - \mathbf{D}, \text{where } \mathbf{D}_{ij} = \|\mathbf{K}\mathbf{w}_j - \mathbf{K}\mathbf{w}_i\|_2^2. \tag{3.27}$$

The optimization solution for the non-linear model can be summarized as in Algorithm 3.

3.2.3 Analysis and Comparisons

In this subsection, we will first analyze the complexity of the proposed *iPath*, present some variants of it, and then provide a probabilistic interpretation for it, followed up by the comparisons with some existing work.

Complexity Analysis We summarize the time complexity of *iPath*-lin and *iPath*-ker in Theorem 3.

Theorem 3. *(Time Complexity). iPath-lin takes $O(N \cdot (ndl^2 + d^3l^3))$ time, and iPath-ker (Algorithm 3) takes $O(N \cdot (n^3l^3 + n^2l^2))$ time, where N is the number of iterations.*

Algorithm 3: *iPath*-ker – forecasting the pathway to impact

Input: (1)feature matrix $\mathbf{X}$;

(2)impact matrix $\mathbf{Y}$;

(3)prior knowledge of the relation $\mathbf{A}_0$;

(4)balance parameters α, β, γ, δ and ϵ;

Output: model parameters $\mathbf{w}_i, i = 1, \ldots, l$

Initialize $\mathbf{W}$, $\mathbf{B}$ and $\mathbf{A}$;

Construct kernel matrix $\mathbf{K}$ from $\mathbf{X}$;

while *not converged* **do**

    Update model parameters $\mathbf{W}$ by Eq. (3.23);

    Update linear transition matrix $\mathbf{B}$ by Eq. (3.26);

    Update impact relation $\mathbf{A}$ by Eq. (3.27);

end

Output model parameters $\mathbf{W}$.

Proof. Omitted for brevity. □

Remarks: in both *iPath*-lin and *iPath*-ker, the number of iterations is small in practice (typically in 5-10 iterations, see Sec. 3.2.4 for details). In *iPath*-lin, each iteration only takes linear time w.r.t. n. In *iPath*-ker, the major computational cost in each iteration is the inverse of a large matrix $\mathbf{S}$ in Eq. (3.23), which is of size nl by nl. One way to speed up is by low-rank approximation on such large matrix [73]. A top-r eigen-decomposition on $\mathbf{S}$ takes $O(n^2l^2r)$, where r is the rank. Then the inverse will become the multiplication of the eigenvector matrices and the inverse of the eigenvalue diagonal matrix, which is very easy to compute. Another way to speed up is to filter out those unpromising training samples. When new training samples arrive, we can first treat them as test samples and make predictions on them using

the existing trained model. Those samples whose prediction error is smaller than a specified threshold will be discarded. In this way, the size of matrix $\mathbf{S}$ will also be reduced.

Variants The proposed *iPath* model is comprehensive in handling both the *prediction consistency* as well as the *parameter smoothness*. In the case when one or both aspects are not necessary for the prediction in some applications, our model can be naturally adapted to accommodate such special cases. In this subsection, we will discuss two of the variants.

Variant #1: known relation. If the relation among the impacts are accurately known a priori, we can fix the relation in the model instead of learning it. We can do this by setting ϵ to 0 and plug in the known relation matrix $\mathbf{A}$. In the optimization solution, we only need to optimize for $\mathbf{W}$ and $\mathbf{B}$ in this variant.

Variant #2: known relation without parameter smoothness. In some cases, the parameter smoothness might not hold and we do not need to consider the linear transition process between adjacent parameters. We can set β, γ and ϵ to 0. This degenerates to the *iBall* model proposed in [73]. It is a special case of our *iPath* model without considering parameter smoothness and with known relation. Another difference is that *iPath* imposes the prediction consistency in the output space, instead of in the parameter space.

Probabilistic Interpretation In this subsection, we will provide a probabilistic interpretation for *iPath*. Our algorithm can be represented by the graphical model shown in Figure 3.13. The shaded nodes $\mathbf{Y}_i$ are the impacts observed, and in the linear formulation they are linear combination of the features with a multivariate

Gaussian noise:

$$\begin{aligned} \mathbf{Y}_i &= \mathbf{X}\mathbf{w}_i + \mathbf{e} \\ \mathbf{e} &\sim \mathcal{N}(\mathbf{0}, \sigma_y^2 \mathbf{I}) \\ \mathbf{Y}_i | \mathbf{w}_i &\sim \mathcal{N}(\mathbf{X}\mathbf{w}_i, \sigma_y^2 \mathbf{I}) \end{aligned} \tag{3.28}$$

For the model parameters $\mathbf{w}_t$, we assume it is a linear transition of the parameter for the last time step $\mathbf{w}_{t-1}$, with a multivariate Gaussian noise:

$$\begin{aligned} \mathbf{w}_t &= \mathbf{B}\mathbf{w}_{t-1} + \epsilon \\ \epsilon &\sim \mathcal{N}(\mathbf{0}, \sigma_w^2 \mathbf{I}) \\ \mathbf{w}_t | \mathbf{w}_{t-1} &\sim \mathcal{N}(\mathbf{B}\mathbf{w}_{t-1}, \sigma_w^2 \mathbf{I}) \end{aligned} \tag{3.29}$$

The relation among the impacts is represented as an undirected graph of different impacts $\mathbf{Y}_i$, with $\mathbf{A}$ as the weight matrix. If the i-th impact $\mathbf{Y}_i$ and the j-th impact $\mathbf{Y}_j$ are similar to each other, then the (i, j)-th entry $\mathbf{A}_{ij}$ is a large positive number. To define the distribution over this undirected graph of impacts, we refer to Hammersley-Clifford theorem in Markov Random Field (MRF) [9] and express it in terms of an energy function E and clique potentials defined on maximal cliques of the undirected graph as:

$$p(\mathbf{Y}) = \frac{1}{Z} \exp(-E(\mathbf{Y})), \text{where } E(\mathbf{Y}) = \sum_{c \in \mathcal{C}} \Phi_c(\mathbf{Y}_c). \tag{3.30}$$

Here $\mathcal{C}$ is the set of maximal cliques of the impact graph, Φ_c is a non-negative function defined on the random variables in the clique and Z is the partition function to ensure that the distribution sums to 1. If we only consider the potentials defined on the edge of the graph, as follows:

$$\Phi_{e=(\mathbf{Y}_i, \mathbf{Y}_j)} = \mathbf{A}_{ij} \|\mathbf{Y}_i - \mathbf{Y}_j\|_2^2 = \mathbf{A}_{ij} \|\mathbf{X}\mathbf{w}_i - \mathbf{X}\mathbf{w}_j\|_2^2 \tag{3.31}$$

Then, the distribution over the label graph is:

$$p(\mathbf{Y}) = \frac{1}{Z}\exp(-\sum_{i=1}^{l}\sum_{j=1}^{l}\mathbf{A}_{ij}\|\mathbf{X}\mathbf{w}_i - \mathbf{X}\mathbf{w}_j\|_2^2) \tag{3.32}$$

With these distributions defined, we aim to maximize the joint distribution described as follows:

$$\arg\max_{\mathbf{Y},\mathbf{X},\mathbf{W}} = p(\mathbf{w}_1)\prod_{t=2}^{l} p(\mathbf{w}_t|\mathbf{w}_{t-1})\prod_{i=1}^{l} p(\mathbf{Y}_i|\mathbf{w}_i)p(\mathbf{Y}) \tag{3.33}$$

where we assume $p(\mathbf{w}_1) \sim \mathcal{N}(\mathbf{0}, \sigma_1^2\mathbf{I})$. If we maximize the above joint distribution, we can obtain the empirical loss, prediction consistency and parameter smoothness terms in *iPath*.

Comparison with Existing Work As we point out in Sec. 3.2.3, *iBall* [73] is a special case of our *iPath* model. The idea of *iBall* is to leverage the relation among impacts in the parameter space, i.e., if $\mathbf{Y}_i$ and $\mathbf{Y}_j$ are similar, then the parameters $\mathbf{w}_i$ for predicting $\mathbf{Y}_i$ and $\mathbf{w}_j$ for predicting $\mathbf{Y}_j$ are similar. The multi-label learning method *MLRL* [130] also exploits such relation in the parameter space via maximum a posterior inference by assuming that $\mathbf{W}$ follows a matrix-variate normal distribution, but ignores the parameter smoothness. Our model *iPath* instead applies such relation in the output space and defines a linear transition process between two parameters at adjacent time steps.

3.2.4 Empirical Evaluations

In this subsection, we empirically evaluate the effectiveness of the proposed algorithms for forecasting the pathway to impact.

Datasets To evaluate the performance of the proposed *iPath* algorithms, we conduct experiments on the citation network dataset provided by AMiner [105] [3], which

[3]https://aminer.org/billboard/citation

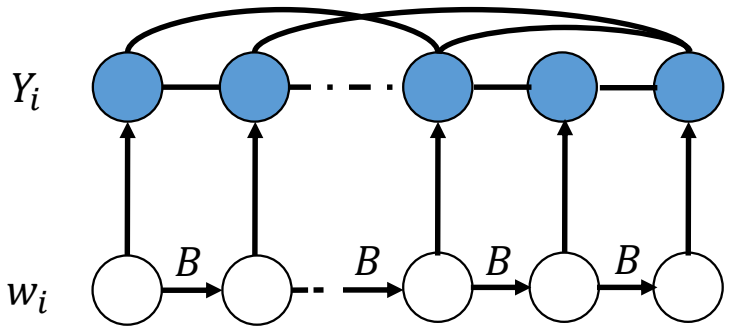


Figure 3.13: Graphical model representation of *iPath*.

is a rich dataset for bibliography network analysis and mining. The dataset contains information of 2,243,976 papers, 1,274,360 authors and 8,882 computer science venues. The information about a paper includes its title, authors, references, venue and publication year. The papers date from year 1936 to year 2013. From these, we can extract the number of citations each paper/author obtains in each year ever since its publication year.

Experiment Setup Our primary task is to forecast a paper's yearly citations from year 6 to year 15 after its publication, with the first five years' citation history observed. To ensure the papers are at least 15 years old, we only keep papers published between year 1960 and 1998. We process the author data in a similar way and keep those whose research career begins (when they publish the first paper) between year 1960 and 1990. For each scholarly entity (paper and author), we represent it as a five dimensional feature vector, which is the yearly citation counts in the first five years. To evaluate our algorithm, we sort the scholarly entities by their starting year (e.g., publication year), and train the model in the older entities and always test on the latest ones. In the experiment, we incrementally add the training samples by this chronological order, and for the paper impact pathway prediction, we reserve the

latest 10% samples as the test set; and for the author impact pathway prediction, we reserve the latest 6% samples as the test set.

Root mean squared error (RMSE) between the actual citations and the predicted ones is used as our accuracy evaluation. All the parameters, including the Gaussian kernel's bandwidth, are chosen through a grid search. All the experiments are run on a Windows machine with four 3.5 GHz Intel Cores and 256 GB RAM.

Results and Analysis

1. Paper and author impact pathway prediction performance. We compare the prediction accuracy of the following methods:

- *ind-linear:* train a liner ridge regression model for the impact in each year separately.
- *ind-kernel:* train a kernel ridge regression model for the impact in each year separately.
- *MTL-robust:* treat predicting the impact in each year as a task and apply the robust multi-task learning algorithm proposed in [29].
- *MLRL:* the multi-label learning method proposed in [130], where model parameters are assumed to conform matrix-variate normal distribution.
- *iBall-linear:* jointly learn the linear regression models as in [73].
- *iBall-kernel:* jointly learn the kernel ridge regression models as in [73].
- *iPath-lin:* the proposed linear predictive model with prediction consistency and parameter smoothness.
- *iPath-ker:* the proposed non-linear predictive model with prediction consistency and parameter smoothness.

The RMSE results of the above methods for predicting the impact pathway of both research papers and authors are in Figure 3.14 and 3.15, respectively. We can make the following observations: (1) the non-linear methods (ind-kernel, *iBall-kernel* and *iPath*-ker) generally perform better than the linear methods (ind-linear, MTL-robust, MLRL, *iBall-linear* and *iPath*-lin), which reflects that the impacts could be over simplified by a linear combination of the features. (2) Among the linear methods, we find that MTL-robust does not help improve the prediction over ind-linear. The possible reason is that MTL-robust has the assumption that the model parameters admit a low-rank and sparse component, which might not be true in our case. The *iBall-linear* performs better than ind-linear, which shows that the impact relation exploitation can indeed help the forecasting. (3) Furthermore, learning a good relation can further boost the performance, as MLRL has lower RMSE than *iBall-linear*. Our *iPath*-lin performs the best among all the linear models, by integrating prediction consistency and parameter smoothness. It is even comparable with ind-kernel when training size is greater than 30% for the paper impact pathway prediction. (4) We can make the similar observation in the non-linear case, as our *iPath*-ker performs better than *iBall-ker*, which itself is better than ind-kernel.

To evaluate the statistical significance, we perform a t-test between *iPath*-ker and the best competitor *iBall-kernel* with 30% of the training papers in the paper impact pathway prediction, and the p-value is 0.01, which suggests the significance of the improvement.

2. Sensitivity analysis. To investigate parameter sensitivity, we perform parametric studies with the two most important parameters in *iPath*, namely, α that controls the importance of prediction consistency, and β that controls the importance of parameter smoothness. Figure 3.16 shows that the proposed model is stable in a large range of both parameter spaces.

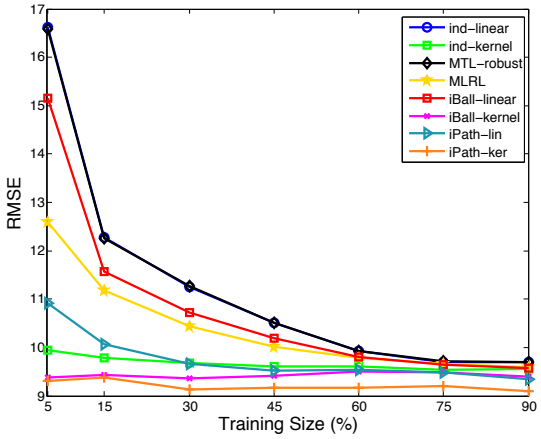


Figure 3.14: RMSE comparison of all the methods for paper impact pathway prediction.

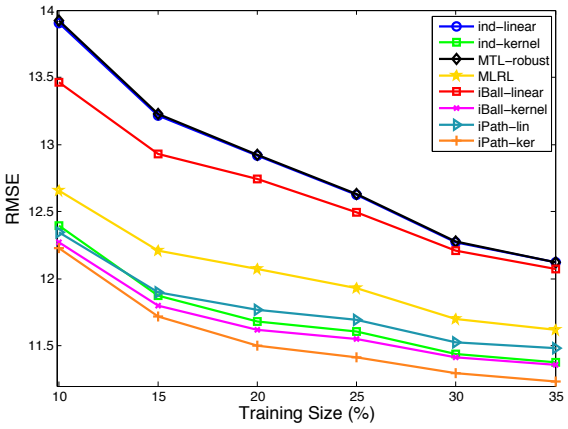
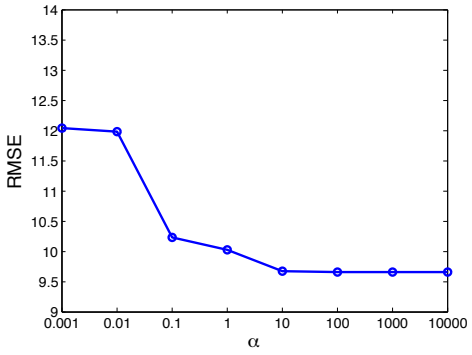
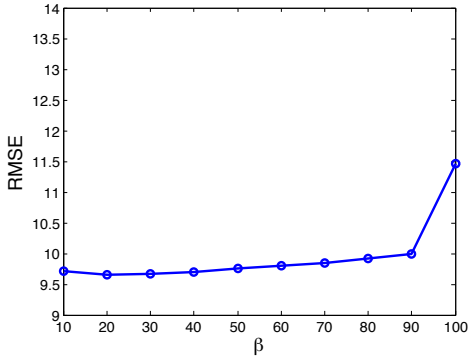


Figure 3.15: RMSE comparison of all the methods for author impact pathway prediction.



(a) RMSE vs. α



(b) RMSE vs. β

Figure 3.16: Sensitivity study on *iPath*-lin: study the effect of the parameters α and β in terms of RMSE.

3. Performance gain analysis. Let us take a closer investigation on where the performance gain of the proposed *iPath* stems from. As we mention above, *iPath* integrates both *prediction consistency* and *parameter smoothness*. We analyze how they contribute to the performance gain. Table 3.4 shows the results of *iPath*-ker methods on both the paper (60% training) and author (25% training) impact pathway prediction. 'Basic form' sets α, β, γ and ϵ all to zero, essentially ind-kernel method;

'Basic form + relation' incorporates the relations among impacts; 'Basic form + relation + transition' incorporates a known relation and the linear transition in the parameter space; 'Basic form + relation + transition + inferring' considers them all with an inferred relation. From the table, we can see that as we incrementally incorporate the elements, the RMSE decreases gradually, which confirms that all these elements are beneficial in improving the prediction performance.

RMSE	Paper Impact	Author Impact
Basic form	9.602	11.608
Basic form + relation	9.507	11.548
Basic form + relation + transition	9.335	11.489
Basic form + relation + transition + inferring	9.171	11.391

Table 3.4: Performance gain analysis of *iPath*. Smaller is better.

4. Robustness to noise in label graph. As *iPath* can learn a good relation for the prediction from our prior knowledge about the relation, we want to see how robust it is wrt the noise level in our prior knowledge. To this end, we input the same relation matrix with noise to *iBall* (the matrix $\mathbf{A}$) and *iPath* (the matrix $\mathbf{A}_0$). The noise is added to each entry of the label matrix with value $0.1 \times \text{NoiseLevel} \times \text{rand}$, where rand is a random number from 0 to 1. Figure 3.17 shows the RMSE results of both *iBall* and *iPath* under different noise levels for paper impact pathway prediction with 30% training samples. We observe a sharp performance drop of *iBall* when noise is added. In contrast, the proposed *iPath* degenerates gradually with the noise level. This shows that *iPath* can learn a relatively good relation even if our prior knowledge is noisy.

5. Convergence analysis. To see how fast the proposed *iPath* converges in practice, we plot the objective function value vs. number of iterations for both paper (15%

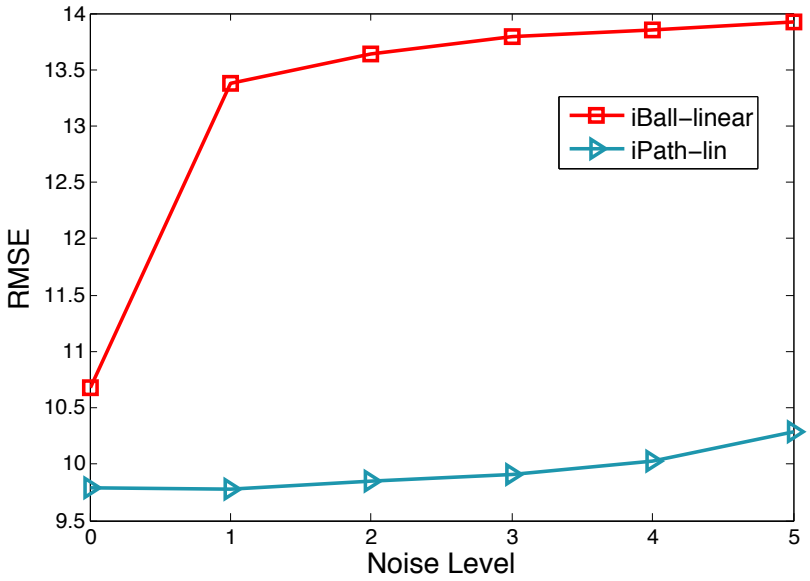
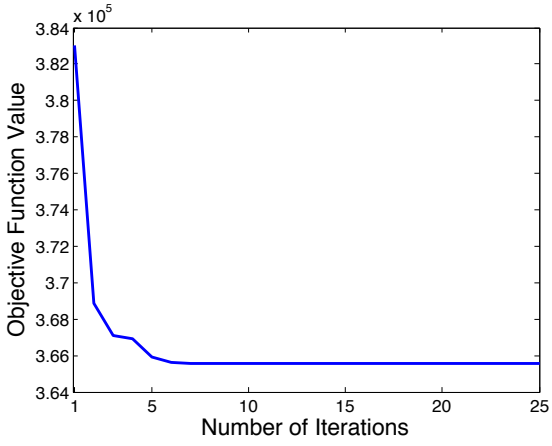
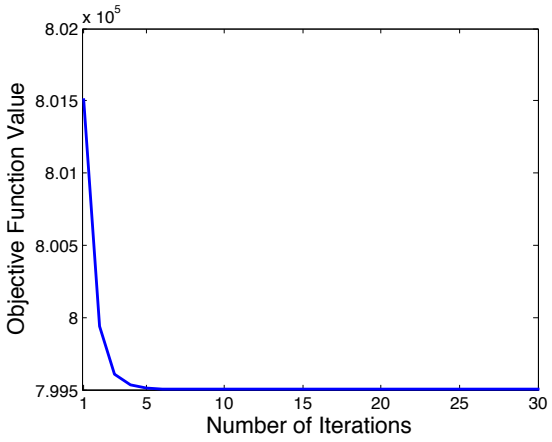


Figure 3.17: Robustness to noise on the label graph.

training samples) and author (10% training samples) impact pathway forecasting as in Figure 3.18. We observe that *iPath* converges after 5-10 iterations.



(a) Objective function value vs. # iterations on paper impact pathway forecasting.



(b) Objective function value vs. # iterations on author impact pathway forecasting.

Figure 3.18: Convergence analysis of *iPath*.

3.3 Part-Whole Outcome Prediction

The great Greek philosopher Aristotle articulated more than 2,000 years ago that "*the whole is greater than the sum of its parts*". This is probably most evident in

teams, which, through appropriate synergy, promise a collective outcome (i.e., team performance) that is superior than the simple addition of what each individual team member could achieve (i.e., individual productivity). For example, in the scientific community, the new breakthrough is increasingly resulting from the teamwork, compared with individual researcher's sole endevour [117]; in professional sports (e.g., NBA), the peak performance of a grass-root team is often attributed to the harmonic teamwork between the team players rather than the individual player's capability. Beyond teams, the *part-whole* relationship also routinely finds itself in other disciplines, ranging from crowdsourcing (e.g., Community-based Question Answering (CQA) sites [121]), collective decision-making in autonomous system (e.g., a self-orchestrated swarm of drones[4]), to reliability assessment of a networked system of components [122, 27].

From the algorithmic perspective, an interesting problem is to predict the outcome of the whole and/or parts [58]. In organizational teams, it is critical to appraise the individual performance, its contribution to the team outcome as well as the team's overall performance [81]. In the emerging field of the "science of science", the dream of being able to predict breakthroughs, e.g. predicting the likelihood of a researcher making disruptive contributions and foreseeing the future impact of her research products (e.g., manuscripts, proposals, system prototypes) pervades almost all aspects of modern science [31]. In Community-based Question Answering (CQA) sites, predicting the long-term impact of a question (*whole*) and its associated answers (*parts*) enables users to spot valuable questions and answers at an early stage. Despite much progress has been made, the existing work either develop separate models for predicting the outcome of whole and parts without explicitly utilizing the part-whole

[4]CBS 60 minutes report: `http://www.cbsnews.com/news/60-minutes-autonomous-drones-set-to-revolutionize-military-technology/`

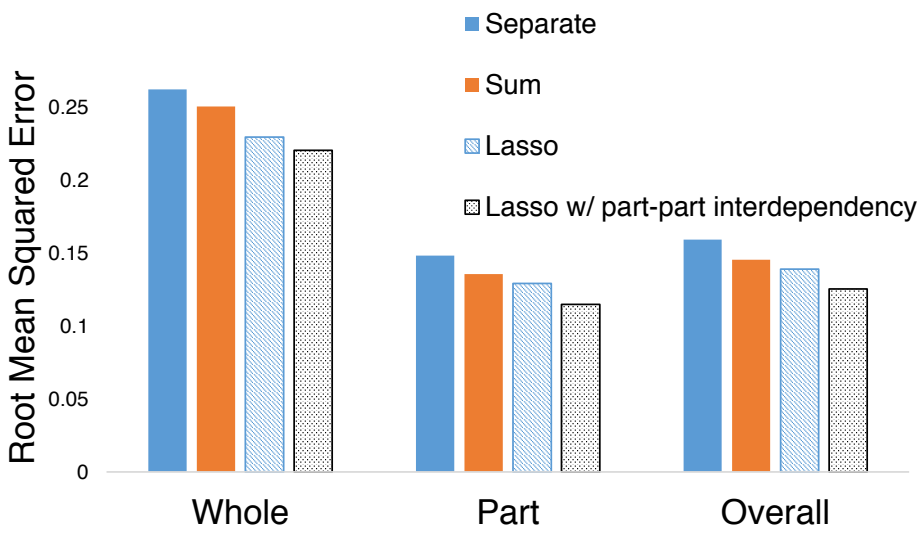


Figure 3.19: Prediction error comparison on *Movie* dataset. Lower is better. Best Viewed in Color. The right two bars are the proposed methods, which encode the non-linear part-whole relationship and the non-linearity with part-part interdependency respectively.

relationship [73, 79], or implicitly assume the outcome of the whole is a *linear* sum of the outcome of the parts [121], which might oversimplify the complicated part-whole relationships (e.g., non-linearity).

The key to address these limitations largely lies in the answers to the following questions, i.e., to what extent does the outcome of parts (e.g., individual productivity) and that of the whole (e.g., team performance) correlated, beyond the existing linear, independency assumption? How can we leverage such potentially non-linear and interdependent 'coupling' effect to mutually improve the prediction of the outcome of the whole and parts collectively? This is exactly the focus of this work, which is highly challenging for the following reasons. First (*Modeling Challenge*), the relationship between the parts outcome and whole outcome might be complicated, beyond the simple addition or linear combination. For example, the authors in [120]

empirically identified a non-linear correlation between the impacts of questions and the associated answers, that is, the impact of a question is much more strongly correlated with that of the *best* answer it receives, compared with the *average* impact of its associated answers. However, how to leverage such non-linear relationship between the parts and whole outcome has largely remained open. For teams, the team performance might be mainly dominated by a few top-performing team members, and/or be hindered by one or more struggling team members (i.e., the classic Wooden Bucket Theory, which says that "A bucket (whole) can only fill with the volume of water the shortest plank (parts) allows"). Moreover, the composing parts of the whole might not be independent with each other. In a networked system, the composing parts are connected with each other via an underlying network. Such part-part interdependency could have a profound impact on both the part outcome correlation as well as each part's contribution to the whole outcome. How can we mathematically encode the non-linear part-whole relationship as well as part-part interdependency? Second (*Algorithmic Challenge*), the complicated part-whole relationship (i.e., non-linearity and interdependency) also poses an algorithmic challenge, as it will inevitably increase the complexity of the corresponding optimization problem. How can we develop scalable algorithms whose theoretic properties are well-understood (e.g., the convergence, the optimality, and the complexity)?

To address these challenges, in this dissertation, we propose a joint predictive model named `PAROLE` to simultaneously and mutually predict the part and whole outcomes. First, *model generality*, the proposed model is flexible in admitting a variety of linear as well as non-linear relationships between the parts and whole outcomes, including *maximum aggregation, linear aggregation, sparse aggregation, ordered sparse aggregation* and *robust aggregation*. Moreover, it is able to characterize part-part interdependency via a graph-based regularization, which encourages the tightly connected

parts to share similar outcomes as well as have similar effect on the whole outcome. Second, *algorithm efficacy*, we propose an effective and efficient block coordinate descent optimization algorithm, which converges to the coordinate-wise optimum with a linear complexity. The main contributions of this section can be summarized as follows:

- **Models.** We propose a joint predictive model (`PAROLE`) that is able to admit a variety of linear as well as non-linear part-whole relationships and encode the part-part interdependency.
- **Algorithms and Analysis.** We propose an effective and efficient block coordinate descent optimization algorithm that converges to the coordinate-wise optimum with a linear complexity in both time and space.
- **Empirical Evaluations.** We conduct extensive empirical studies on several real-world datasets and demonstrate that the proposed `PAROLE` achieves consistent prediction performance improvement and scales linearly. See Fig. 3.19 for some sampling results.

3.3.1 Problem Definition

The main symbols are summarized in Table 3.5. We use bold capital letters (e.g., $\mathbf{A}$) for matrices and bold lowercase letters (e.g, $\mathbf{w}$) for vectors. We index the elements in a matrix using a convention similar to Matlab, e.g., $\mathbf{A}(:, j)$ is the j^{th} column of $\mathbf{A}$, etc. The vector obtained by sorting the components in non-increasing order of $\mathbf{x}$ is denoted by $\mathbf{x}_{\downarrow}$. Such sorting operation can be defined by a permutation matrix $\mathbf{P}_{\mathbf{x}}$, i.e., $\mathbf{P}_{\mathbf{x}}\mathbf{x} = \mathbf{x}_{\downarrow}$. We use $\mathcal{K}_{m+}$ to denote the *monotone non-negative cone*, i.e., $\mathcal{K}_{m+} = \{\mathbf{x} \in \mathbb{R}^n : x_1 \geq x_2 \geq \ldots x_n \geq 0\} \subset \mathbb{R}^n_+$. Similarly, we use $\mathcal{K}_m$ for the monotone cone.

Symbols	Definition
$\mathbf{F}^o$, $\mathbf{F}^p$	feature matrices for whole and part entities
$\mathbf{y}^o$, $\mathbf{y}^p$	impact vectors for whole and part entities
$\mathcal{O} = \{o_1, o_2, \ldots, o_{n_o}\}$	set of whole entities
$\mathcal{P} = \{p_1, p_2, \ldots, p_{n_p}\}$	set of part entities
$\phi(\cdot)$	whole to parts mapping function
G^p	the network connectivity among part entities
a_j^i	the contribution of part p_j to whole o_i
n_o/n_p	number of whole/part entities
$\text{Agg}(\cdot)$	the function that aggregates parts outcome
e_i	predicted whole outcome using whole feature vs. predicted whole outcome using aggregated parts outcome

Table 3.5: Symbols for PAROLE

We consider predicting the outcome for both the whole and their composing parts. Fig. 3.20 presents an illustrative example, which aims to predict the popularity (e.g., Facebook likes) of a particular movie (*whole*) and the popularities of the participating actors/actresses (*parts*). We denote the set of whole entities by $\mathcal{O} = \{o_1, o_2, \ldots, o_{n_o}\}$, and denote the set of part entities by $\mathcal{P} = \{p_1, p_2, \ldots, p_{n_p}\}$, where n_o and n_p are the number of the whole and parts, respectively. To specify the part-whole associations, we also define a mapping function ϕ that maps a whole entity to the set of its composing parts, e.g., $\phi(o_i) = \{p_{i_1}, p_{i_2}, \ldots, p_{i_{n_i}}\}$ (i.e., the edges between a movie and actors/actresses in Fig. 3.20). Note that the two sets $\phi(o_i)$ and $\phi(o_j)$ might have overlap. In the example of movies as whole entities, one actor could participate in multiple movies. Let $\mathbf{F}^o$ be the feature matrix for the whole entities, where the i^{th} row

$\mathbf{F}^o(i,:)$ is the feature vector for the i^{th} whole entity. Similarly, let $\mathbf{F}^p$ be the feature matrix for the part entities, where the j^{th} row $\mathbf{F}^p(j,:)$ is the feature vector for the j^{th} part entity. The outcome vector of the whole entities is denoted as $\mathbf{y}^o$ and the outcome vector of the part entities is denoted as $\mathbf{y}^p$. In addition, we might also observe a network connectivity among the part entities, denoted as G^p. In the movie example, the network G^p could be the collaboration network among the actors/actresses (the connections among the actors/actresses in Fig. 3.20).

With the above notations, we formally define our Part-Whole Outcome Prediction problem as follows:

Problem 4. Part-Whole Outcome Prediction

Given: *the feature matrix for the whole/part entities $\mathbf{F}^o$/$\mathbf{F}^p$, the outcome vector for the whole/part entities $\mathbf{y}^o$/$\mathbf{y}^p$, the whole to part mapping function ϕ, and the parts' network G^p (optional);*

Predict: *the outcome of new whole and parts' entities.*

3.3.2 Proposed Model – PAROLE

In this subsection, we present our joint predictive model `PAROLE` to simultaneously and mutually predict the outcome of the whole and parts. We first formulate it as a generic optimization problem, and then present the details on how to instantiate the part-whole relationship and part-part interdependency, respectively.

A Generic Joint Prediction Framework In order to fully embrace the complexity of the part-whole and part-part relationship, our joint predictive model should meet the following desiderata.

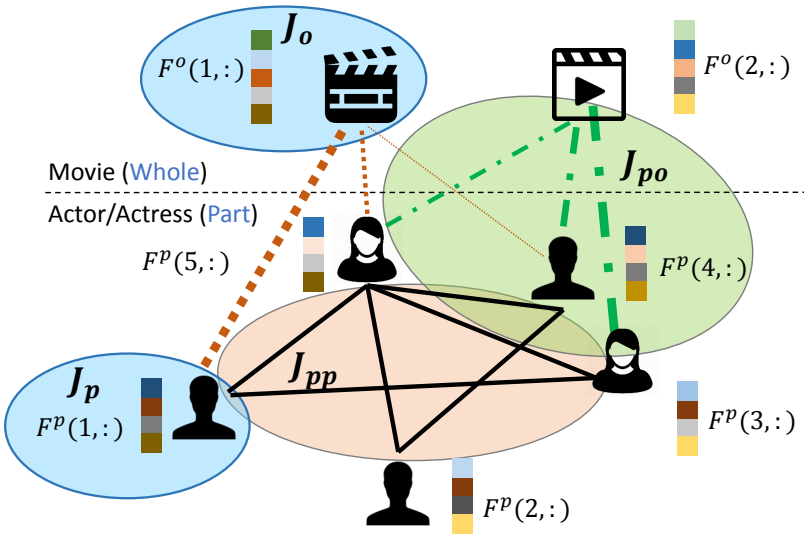


Figure 3.20: An illustrative example of part-whole outcome prediction where movies are the whole entities and the actors/actresses are the part entities. The four shadowed ellipses correspond to the key sub-objectives in our proposed `PAROLE` model (Eq. (3.34)).

First (*part-whole relationship*), the outcome of the whole and that of the parts might be strongly correlated with each other. For example, the team outcome is usually a collective effort of the team members. Consequently, the team performance is likely to be correlated/coupled with each individual's productivity, which might be beyond a simple linear correlation. This is because a few top-performing team members might dominate the overall team performance, or reversely, a few struggling team members might drag down the performance of the entire team. Likewise, in scientific community, a scientist's reputation is generally built by one or a few of her highest-impact work. Our joint predictive model should have the capability to encode such non-linear part-whole relationships, so that the prediction of the parts outcome and that of the whole can mutually benefit from each other.

Second (*part-part interdependency*), the composing parts of a whole entity might

be interdependent/interconnected via an underlying network, e.g., the collaboration network among the actors/actresses. The part-part interdependency could have a profound impact on the part-whole outcome prediction performance. That is, not only might the closely connected parts have similar effect on the whole outcome, but also these parts are very likely to share similar outcomes between themselves. Therefore, it is desirable to encode the part-part interdependency in the joint model to boost the prediction performance.

With these design objectives in mind, we propose a generic framework for the joint predictive model as follows:

$$\begin{aligned}\min_{\mathbf{w}^o,\mathbf{w}^p} \mathcal{J} = & \underbrace{\frac{1}{n_o}\sum_{i=1}^{n_o}\mathcal{L}[f(\mathbf{F}^o(i,:),\mathbf{w}^o),\mathbf{y}^o(i))]}_{\mathcal{J}_o\text{: predictive model for whole entities}} + \underbrace{\frac{1}{n_p}\sum_{i=1}^{n_p}\mathcal{L}[f(\mathbf{F}^p(i,:),\mathbf{w}^p),\mathbf{y}^p(i))]}_{\mathcal{J}_p\text{: predictive model for part entities}} \\ & + \underbrace{\frac{\alpha}{n_o}\sum_{i=1}^{n_o} h(f(\mathbf{F}^o(i,:),\mathbf{w}^o),\text{Agg}(\phi(o_i)))}_{\mathcal{J}_{po}\text{: part-whole relationship}} \\ & + \underbrace{\frac{\beta}{n_p}\sum_{i=1}^{n_p}\sum_{j=1}^{n_p} G^p_{ij} g(f(\mathbf{F}^p(i,:),\mathbf{w}^p), f(\mathbf{F}^p(j,:),\mathbf{w}^p))}_{\mathcal{J}_{pp}\text{: part-part interdependency}} + \underbrace{\gamma(\Omega(\mathbf{w}^o)+\Omega(\mathbf{w}^p))}_{\mathcal{J}_r\text{: parameter regularizer}}\end{aligned} \tag{3.34}$$

where the objective function is a sum of five sub-objective functions. The first two sub-objectives $\mathcal{J}_o$ and $\mathcal{J}_p$ (the two blue shadowed ellipses in Fig. 3.20) minimize the training loss for whole and parts outcome predictions, where $f(\cdot,\cdot)$ is the prediction function parameterized by $\mathbf{w}^o$ and $\mathbf{w}^p$. The prediction function could be either linear or non-linear; and $\mathcal{L}(\cdot)$ is a loss function, e.g., squared loss for regression or logistic loss for classification. The core of the objective function is the third term $\mathcal{J}_{po}$ (the green shadowed ellipse in Fig. 3.20) and the fourth term $\mathcal{J}_{pp}$ (the pink shadowed ellipse in Fig. 3.20). $\mathcal{J}_{po}$ characterizes the part-whole relationship, where $\text{Agg}(\cdot)$ is a function that aggregates the predicted outcomes of all the composing parts for the whole to a

single outcome, e.g., maximum, summation/mean or more complicated aggregations; and $h(\cdot)$ function measures the correlation between the predicted whole outcome and the aggregated predicted parts outcome. In $\mathcal{J}_{pp}$, the function $g(\cdot)$ characterizes the relationship of the predicted outcomes of parts i and j based on their connectivity G^p_{ij}, such that tightly connected parts would share similar outcomes. Lastly, $\mathcal{J}_r$ regularizes $\mathbf{w}^o$ and $\mathbf{w}^p$ to prevent overfitting. The regularization parameters α, β and γ are used to balance the relative importance of each aspect.

Remarks: Depending on the specific choices of the aggregation function $\text{Agg}(\cdot)$ and the $h(\cdot)$ function, the proposed model in Eq. (3.34) is able to admit a variety of part-whole relationships, which we elaborate below.

Modeling Part-Whole Relationships

Overview. In this subsection, we give the instantiations for a variety of part-whole relationships. For each whole entity o_i, define e_i as follows:

$$e_i = \mathbf{F}^o(i,:)\mathbf{w}^o - \text{Agg}(o_i) \tag{3.35}$$

which measures the difference between the predicted whole outcome using whole features (i.e., $\mathbf{F}^o(i,:)\mathbf{w}^o$) and predicted whole outcome using aggregated parts outcome (i.e., $\text{Agg}(o_i)$). Our proposed model will be able to characterize a variety of part-whole relationship, by using (a) different aggregation functions $\text{Agg}(\cdot)$ with augmented regularizations; and (b) different loss functions on e_i (e.g., squared loss or robust estimator).

Maximum aggregation. Let us first consider using maximum as the aggregation function, which can model the correlation between the whole outcome and the maximum parts outcome. Given that the max function is not differentiable, we propose to approximate it with a differentiable function that will largely facilitate

the optimization process. In details, we propose to use the smooth "soft" maximum function, which was first used in economic literature for consumer choice [100]: $\max(x_1, x_2, \ldots, x_n) \approx \ln(\exp(x_1) + \exp(x_2) + \ldots + \exp(x_n))$, where the maximum is approximated by summing up the exponential of each item followed by a logarithm. With this, we define the maximum aggregation function as follows:

$$\text{Agg}(o_i) = \ln(\sum_{j \in \phi(o_i)} \exp(\mathbf{F}^p(j,:)\mathbf{w}^p)) \tag{3.36}$$

which approximates the maximum predicted parts outcome. The part-whole relationship with maximum aggregation can be formulated as follows:

$$\mathcal{J}_{po} = \frac{\alpha}{2n_o} \sum_{i=1}^{n_o} e_i^2 \tag{3.37}$$

where we use the squared loss to measure the difference between the predicted whole outcome and the predicted approximated maximum parts outcome.

For the remaining part-whole relationships, we instantiate $\text{Agg}(o_i)$ using a linear function as follows:

$$\text{Agg}(o_i) = \sum_{j \in \phi(o_i)} a_j^i \mathbf{F}^p(j,:)\mathbf{w}^p \tag{3.38}$$

where each a_j^i is the weight of a particular part j's contribution to the whole o_i's outcome. Defining $\mathbf{a}_i$ as the vector whose components are a_j^i, $j \in \phi(o_i)$ and by imposing (i) different loss functions on e_i, and/or (ii) different norms on $\mathbf{a}_i$, we can model either linear or nonlinear part-whole relationships.

Linear aggregation. In this scenario, the whole outcome is a weighted linear combination of the parts outcome, where the weights determine each individual part's contribution to the whole outcome. The intuition of linear aggregation is that , in contributing to the final whole outcome, some parts play more important roles than the others. This part-whole relationship can be formulated as follows:

$$\mathcal{J}_{po} = \frac{\alpha}{2n_o} \sum_{i=1}^{n_o} e_i^2 \tag{3.39}$$

where we use the squared loss to measure the difference between the whole outcome and the aggregated parts outcome.

Remark: this formulation generalizes several special part-whole relationships. The expression that "*the whole is the sum of its parts*" is a special case of Eq. (3.39) where various a_j^i is 1, which we refer to as *Sum* in the empirical study. The average coupling formulated in [121] is also its special case with $a_j^i = \frac{1}{|o_i|}$. Instead of fixing the weights, Eq. (3.39) allows the model to learn to what extent each part contributes to the prediction of the whole outcome. Nonetheless, in all these variants, we have assumed that the part outcomes always have a *linear* effect on the whole outcome.

Sparse aggregation. The above linear aggregation assumes that each part would contribute to the whole outcome, which might not be the case as some parts have little or no effect on the whole outcome. This scenario can be seen in large teams, where the team performance could be primarily determined by a few members, who could either make or break the team performance. To encourage such a sparse selection among the composing parts of a whole entity, a natural choice is to introduce the l_1 norm on the vector $\mathbf{a}_i$ [106]:

$$\mathcal{J}_{po} = \frac{\alpha}{n_o} \sum_{i=1}^{n_o} (\frac{1}{2} e_i^2 + \lambda |\mathbf{a}_i|_1) \tag{3.40}$$

where the l_1 norm can shrink some part contributions to exactly zero and the parameter λ controls the degree of sparsity.

Ordered sparse aggregation. In some cases, the team performance (i.e., the whole outcome) is determined by not only a few key members, but also the structural hierarchy between such key members within the organization. To model such parts performance ranking in addition to the sparse selection, we adopt the ordered weighted l_1 norm (OWL) [126] that is able to give more weights to those parts with bigger effect on the whole outcome. Such part-whole relationship can be formulated

as follows:

$$\mathcal{J}_{po} = \frac{\alpha}{n_o} \sum_{i=1}^{n_o} (\frac{1}{2} e_i^2 + \lambda \Omega_{\mathbf{w}}(\mathbf{a}_i)) \tag{3.41}$$

where $\Omega_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{n} |x|_{[i]} w_i = \mathbf{w}^T |\mathbf{x}|_{\downarrow}$ is the ordered weighted l_1 norm, where $|x|_{[i]}$ is the i-th largest component of the vector $|\mathbf{x}|$ and $\mathbf{w} \in \mathcal{K}_{m+}$ is a vector of non-increasing non-negative weights.

Robust aggregation. In all the above formulations, we model the difference between the whole outcome and the aggregated parts outcome using squared loss, which is prone to outlying parts/wholes. To address this issue, we employ robust regression models [68] to reduce the effect of outliers as follows:

$$\mathcal{J}_{po} = \frac{\alpha}{n_o} \sum_{i=1}^{n_o} \rho(e_i) \tag{3.42}$$

where $\rho(\cdot)$ is a nonnegative and symmetric function that gives the contribution of each residual e_i to the objective function. In this work, we consider two robust estimators, namely Huber and Bisquare estimators as follows:

Method \ Case	$\lvert e\rvert \leq t$	$\lvert e\rvert > t$
Huber $\rho_H(e)$	$\frac{1}{2}e^2$	$t\lvert e\rvert - \frac{1}{2}t^2$
Bisquare $\rho_B(e)$	$\frac{t^2}{6}\left\{1 - [1 - (\frac{e}{t})^2]^3\right\}$	$\frac{t^2}{6}$

where the value t is a tuning constant. Smaller t values have more resistance to outliers.

Modeling Part-Part Interdependency As mentioned in Sec. 3.3.2, the part-part interdependency, if exists, can play two roles in the part-whole outcome predictions, i.e., closely connected parts would (A) have similar effect on the whole outcome and (B) share similar part outcomes between themselves.

A - The effect on the whole outcome: the closely connected parts might have similar impact on the whole outcome. It turns out we can use the same method to model such a part-part effect for various aggregation methods. Let us take *sparse aggregation* as an example and instantiate the term $\mathcal{J}_{po}$ in Eq. (3.34) as follows:

$$\mathcal{J}_{po} = \frac{\alpha}{n_o} \sum_{i=1}^{n_o} \left[\frac{1}{2} e_i^2 + \lambda |\mathbf{a}_i|_1 + \frac{1}{2} \sum_{k,l \in \phi(o_i)} G_{kl}^p (a_k^i - a_l^i)^2 \right] \tag{3.43}$$

where if the two parts k and l of o_i are tightly connected, i.e., $G_{k,l}^p$ is large, then the difference between their impacts on the whole outcome, a_k^i and a_l^i, is small.

B - The effect on the parts' outcomes: the tightly connected parts might share similar outcomes themselves. Such parts outcome similarity can be instantiated by a graph regularization as follows:

$$\mathcal{J}_{pp} = \frac{\beta}{2n_p} \sum_{i=1}^{n_p} \sum_{j=1}^{n_p} G_{ij}^p (\mathbf{F}^p(i,:)\mathbf{w}^p - \mathbf{F}^p(j,:)\mathbf{w}^p)^2 \tag{3.44}$$

where tightly connected two parts i and j with large $G_{k,l}^p$ is encouraged to be closer to each other in the output space, i.e., with similar predicted outcomes.

3.3.3 Optimization Algorithm

In this subsection, we propose an effective and efficient block coordinate descent optimization algorithm to solve the joint prediction framework in Eq. (3.34), followed by the convergence and complexity analysis.

Block Coordinate Descent Algorithm The proposed Eq. (3.34) is general, being able to admit a variety of different separate models ($\mathcal{J}_o$ and $\mathcal{J}_p$) as well as part-whole relationship ($\mathcal{J}_{po}$). Let us first present our algorithm to solve a specific instance of Eq. (3.34) by instantiating it using linear predictive functions, squared loss and sparse

aggregation as follows:

$$\begin{aligned}\min_{\mathbf{w}^o,\mathbf{w}^p} &\frac{1}{2n_o}\sum_{i=1}^{n_o}(\mathbf{F}^o(i,:)\mathbf{w}^o-\mathbf{y}^o(i))^2+\frac{1}{2n_p}\sum_{i=1}^{n_p}((\mathbf{F}^p(i,:)\mathbf{w}^p-\mathbf{y}^p(i))^2\\ &+\frac{\beta}{2n_p}\sum_{i=1}^{n_p}\sum_{j=1}^{n_p}G^p_{ij}(\mathbf{F}^p(i,:)\mathbf{w}^p-\mathbf{F}^p(j,:)\mathbf{w}^p)^2+\frac{\gamma}{2}(\|\mathbf{w}^o\|_2^2+\|\mathbf{w}^p\|_2^2)\\ &+\frac{\alpha}{n_o}\sum_{i=1}^{n_o}\left[\frac{1}{2}e_i^2+\lambda|\mathbf{a}_i|_1+\frac{1}{2}\sum_{k,l\in\phi(o_i)}G^p_{kl}(a^i_k-a^i_l)^2\right]\end{aligned} \tag{3.45}$$

In the formulation, we identify three coordinate blocks, namely $\mathbf{w}^o$, $\mathbf{w}^p$ and various a^i_j. We propose a block coordinate descent (BCD) algorithm to optimize Eq. (3.45) by updating one coordinate block while fixing the other two.

1. Updating $\mathbf{w}^o$ while fixing others: Observing that only $\mathcal{J}_o$, $\mathcal{J}_{po}$ and $\mathcal{J}_r$ are functions of $\mathbf{w}^o$, we have

$$\begin{aligned}\frac{\partial\mathcal{J}}{\partial\mathbf{w}^o}&=\frac{\partial\mathcal{J}_o}{\partial\mathbf{w}^o}+\frac{\partial\mathcal{J}_{po}}{\partial\mathbf{w}^o}+\frac{\partial\mathcal{J}_r}{\partial\mathbf{w}^o}\\ &=\frac{1}{n_o}(\mathbf{F}^o)'(\mathbf{F}^o\mathbf{w}^o-\mathbf{y}^o)+\gamma\mathbf{w}^o+\frac{\alpha}{n_o}(\mathbf{F}^o)'(\mathbf{F}^o\mathbf{w}^o-\mathbf{M}\mathbf{F}^p\mathbf{w}^p)\end{aligned} \tag{3.46}$$

where $\mathbf{M}$ is a n_o by n_p sparse matrix with $\mathbf{M}(i,j)=a^i_j$, for $j\in\phi(o_i)$. We then update $\mathbf{w}^o$ as $\mathbf{w}^o\leftarrow\mathbf{w}^o-\tau\frac{\partial\mathcal{J}}{\partial\mathbf{w}^o}$, where τ is the step size.

2. Updating $\mathbf{w}^p$ while fixing others: The sub-objective functions that are related to $\mathbf{w}^p$ are $\mathcal{J}_p$, $\mathcal{J}_{pp}$, $\mathcal{J}_{po}$ and $\mathcal{J}_r$. Therefore,

$$\begin{aligned}\frac{\partial\mathcal{J}}{\partial\mathbf{w}^p}&=\frac{\partial\mathcal{J}_p}{\partial\mathbf{w}^p}+\frac{\partial\mathcal{J}_{pp}}{\partial\mathbf{w}^p}+\frac{\partial\mathcal{J}_{po}}{\partial\mathbf{w}^p}+\frac{\partial\mathcal{J}_r}{\partial\mathbf{w}^p}\\ &=\frac{1}{n_p}(\mathbf{F}^p)'(\mathbf{F}^p\mathbf{w}^p-\mathbf{y}^p)+\frac{\beta}{n_p}(\mathbf{F}^p)'\mathcal{L}^p\mathbf{F}^p\mathbf{w}^p+\gamma\mathbf{w}^p\\ &\quad-\frac{\alpha}{n_o}(\mathbf{F}^p)'\mathbf{M}'(\mathbf{F}^o\mathbf{w}^o-\mathbf{M}\mathbf{F}^p\mathbf{w}^p)\end{aligned} \tag{3.47}$$

where $\mathcal{L}^p$ is the Laplacian of the graph G^p [3]. Similarly, $\mathbf{w}^p$ can be updated by $\mathbf{w}^p\leftarrow\mathbf{w}^p-\tau\frac{\partial\mathcal{J}}{\partial\mathbf{w}^p}$.

3. Updating a_j^i while fixing others: Let us fix a whole o_i and the sub-problem with respect to $\mathbf{a}_i$ becomes:

$$\min_{\mathbf{a}_i} \frac{1}{2} e_i^2 + \lambda |\mathbf{a}_i|_1 + \frac{1}{2} \sum_{k,l \in \phi(o_i)} G_{kl}^p (a_k^i - a_l^i)^2 \tag{3.48}$$

Observing that the sub-problem is a composite of a non-smooth convex function ($\lambda |\mathbf{a}_i|_1$) and a differentiable convex function (the remaining terms), we update $\mathbf{a}_i$ using the proximal gradient descent method [7]. We first take a gradient step by moving $\mathbf{a}_i$ along the negative direction of the derivative of the smooth part w.r.t. $\mathbf{a}_i$, as follows:

$$\mathbf{z} = \mathbf{a}_i - \tau [e_i (-\mathbf{F}^p(\phi(o_i), :)\mathbf{w}^p) + \mathcal{L}_i^p \mathbf{a}_i] \tag{3.49}$$

where $\mathcal{L}_i^p$ is a shorthand notation for the Laplacian of the subgraph $G^p(\phi(o_i), \phi(o_i))$. Next, we compute the proximal-gradient update for the l_1 norm using soft-thresholding as $\mathbf{a}_i \leftarrow \mathcal{S}_{\tau\lambda}(\mathbf{z})$, where the soft-thresholding operator is defined as follows:

$$[\mathcal{S}_t(\mathbf{z})]_j = \text{sign}(\mathbf{z}_j)(|\mathbf{z}_j| - t)_+, \tag{3.50}$$

where we use $(x)_+$ as a shorthand for $\max\{x, 0\}$.

We will cycle through the above three steps to update the three coordinate blocks until convergence. The algorithm is summarized in Algorithm 4.

Remarks: we want to emphasize that Algorithm 4 provides a general optimization framework that not only works for the formulation with *sparse aggregation* in Eq. (3.45), but is also applicable to the other part-whole relationships introduced in Sec. 3.3.2. The only difference is that, since $\mathcal{J}_{po}$ varies for each part-whole relationship, its derivatives w.r.t. the coordinate blocks would also change. Next, for each of the other part-whole relationships, we give their derivative or proximal gradient w.r.t. the three coordinate blocks.

Algorithm 4: PAROLE - Part-Whole Outcome Predictions

Input: (1) the feature matrix for whole/part entities $\mathbf{F}^o/\mathbf{F}^p$, (2) outcome vector for the whole/part entities $\mathbf{y}^o/\mathbf{y}^p$, (3) the whole to parts mapping function ϕ, (4) the part-part network G^p (optional), (5) parameters $\alpha, \beta, \gamma, \lambda, \tau$.

Output: Model parameters $\mathbf{w}^o$ and $\mathbf{w}^p$.

Initialize $\mathbf{w}^o$ and $\mathbf{w}^p$ and $a_j, j \in \phi(o_i), i = 1, \ldots, n_o$;

while *Not converged* **do**

  Update $\mathbf{w}^o \leftarrow \mathbf{w}^o - \tau \frac{\partial \mathcal{J}}{\partial \mathbf{w}^o}$;

  Update $\mathbf{w}^p \leftarrow \mathbf{w}^p - \tau \frac{\partial \mathcal{J}}{\partial \mathbf{w}^p}$;

  Update $\mathbf{a}_i$ via proximal gradient descent for $i = 1, \ldots, n_o$;

end

1. Maximum aggregation: the derivatives of $\mathcal{J}_{po}$ w.r.t. $\mathbf{w}^o$ and $\mathbf{w}^p$ are as follows:

$$\frac{\partial \mathcal{J}_{po}}{\partial \mathbf{w}^o} = \frac{\alpha}{n_o} \sum_{i=1}^{n_o} e_i (\mathbf{F}^o(i,:))'$$
$$\frac{\partial \mathcal{J}_{po}}{\partial \mathbf{w}^p} = \frac{\alpha}{n_o} \sum_{i=1}^{n_o} e_i \cdot \frac{\sum_{j \in \phi(o_i)} (\mathbf{F}^p(j,:))' \tilde{\mathbf{y}}_i^p}{\sum_{j \in \phi(o_i)} \tilde{\mathbf{y}}_i^p}$$

where we denote $\tilde{\mathbf{y}}_i^p = \exp(\mathbf{F}^p(j,:)\mathbf{w}^p)$.

2. Linear aggregation: the derivatives of $\mathcal{J}_{po}$ w.r.t. $\mathbf{w}^o$ and $\mathbf{w}^p$ are the same as in the *sparse aggregation* case. Its derivative w.r.t. $\mathbf{a}_i$ is same as in Eq. (3.49) without the following proximal-gradient update.

3. Ordered sparse aggregation: the only difference from the *sparse aggregation* lies in the proximal-gradient update for the OWL norm, which can be computed as follows [126]:

$$\text{prox}_{\Omega_{\mathbf{w}}}(\mathbf{v}) = \text{sign}(\mathbf{v}) \odot \left(\mathbf{P}_{|\mathbf{v}|}^T \text{proj}_{\mathbb{R}_+^n} (\text{proj}_{\mathcal{K}_m} (|\mathbf{v}|_\downarrow - \mathbf{w})) \right) \tag{3.51}$$

In the above equation, to compute $\text{prox}_{\Omega_{\mathbf{w}}}(\mathbf{v})$, we first compute the Euclidean project of $(|\mathbf{v}|_{\downarrow} - \mathbf{w})$ onto $\mathcal{K}_m$ using the linear time *pool adjacent violators* (PAV) algorithm [87]. This is followed by a projection onto the first orthant by a clipping operation. The resulting vector is sorted back according to the permutation matrix $\mathbf{P}_{|\mathbf{v}|}$ and then element-wisely multiplied by the signs of $\mathbf{v}$.

4. Robust aggregation: we compute the gradient of $\mathcal{J}_{po}$ using chain rule as follows:

$$\frac{\partial \mathcal{J}_{po}}{\partial \mathbf{w}^o} = \frac{\alpha}{n_o}\sum_{i=1}^{n_o}\frac{\partial \rho(e_i)}{\partial e_i}\frac{\partial e_i}{\partial \mathbf{w}^o}, \quad \frac{\partial \mathcal{J}_{po}}{\partial \mathbf{w}^p} = \frac{\alpha}{n_o}\sum_{i=1}^{n_o}\frac{\partial \rho(e_i)}{\partial e_i}\frac{\partial e_i}{\partial \mathbf{w}^p},$$
$$\frac{\partial \mathcal{J}_{po}}{\partial \mathbf{a}_i} = \frac{\alpha}{n_o}[\frac{\partial \rho(e_i)}{\partial e_i}\frac{\partial e_i}{\partial \mathbf{a}_i} + \mathcal{L}_i^p \mathbf{a}_i]$$

where $\frac{\partial e_i}{\partial \mathbf{w^o}} = \mathbf{F}^o(i,:)'$, $\frac{\partial e_i}{\partial \mathbf{w^p}} = -\sum_{j\in\phi(o_i)} a_j \mathbf{F}^p(j,:)'$, and $\frac{\partial e_i}{\partial \mathbf{a}_i} = -\mathbf{F}^p(\phi(o_i),:)\mathbf{w}^p$; and the gradient of the Huber and Bisquare estimator can be computed as follows:

Method \ Case	$\lvert e\rvert \le t$	$\lvert e\rvert > t$
Huber $\frac{\partial \rho_H(e)}{\partial e}$	e	$t\cdot \text{sign}(e)$
Bisquare $\frac{\partial \rho_B(e)}{\partial e}$	$e[1-(e/t)^2]^2$	0

Proofs and Analysis In this subsection, we analyze the proposed `PAROLE` algorithm in terms of its convergence, optimality and complexity.

First, building upon the proposition from [107], we have the following theorem regarding the proposed Algorithm 4, which says that under a mild assumption, it converges to a local optimum (i.e., coordinate-wise minimum) of Eq. (3.45).

Theorem 4. *(*Convergence and Optimality of `PAROLE`). As long as $-\gamma$ is not an eigenvalue of $\frac{\alpha+1}{n_o}\mathbf{F}^{o\prime}\mathbf{F}^o$ or $\frac{1}{n_p}\mathbf{F}^{p\prime}\mathbf{F}^p + \beta\mathbf{F}^{p\prime}\mathcal{L}^p\mathbf{F}^p + \frac{\alpha}{n_o}\mathbf{F}^p\mathbf{M}'\mathbf{M}\mathbf{F}^p$, Algorithm 4 converges to a coordinate-wise minimum point.

Proof. Omitted for brevity. □

Next, we analyze the complexity of Algorithm 4, which is summarized in Lemma 3.

Lemma 3. *(*Complexity of `PAROLE`). Algorithm 4 takes $O(T(n_o d_o + n_p d_p + m_{po} + m_{pp}))$ time for *linear aggregation*, *maximum aggregation*, *sparse aggregation*, and *robust aggregation*, and it takes $O(T(n_o d_o + n_p d_p + C n_p d_p + m_{pp}))$ for *ordered sparse aggregation*, where d_o and d_p are the dimensionality of the whole and part feature vectors, $m_{po} = \sum_i |\phi(o_i)|$ is the number of associations between the whole and parts and m_{pp} is the number of edges in the part-part network, T is the number of iterations, $C = \max_i \log(|\phi(o_i)|)$ is a constant. The space complexity for Algorithm 4 is $O(n_o d_o + n_p d_p + m_{po} + m_{pp})$ for all the part-whole relationships.

Proof. Omitted for brevity. □

Remarks: suppose we have a conceptual part-whole graph $G = \{\mathcal{O}, \mathcal{P}\}$, which has n_o nodes for the whole entities and n_p nodes for the part entities, m_{po} links from whole nodes to their composing parts nodes and m_{pp} links in the part-part networks. The Lemma 3 says that `PAROLE` scales linearly w.r.t. the size of this part-whole graph in both time and space.

3.3.4 Experiments

In this subsection, we present the empirical evaluation results. The experiments are designed to evaluate the following aspects:

- *Effectiveness:* how accurate is the proposed `PAROLE` algorithm for predicting the outcomes of parts and whole?
- *Efficiency:* how fast and scalable is the proposed `PAROLE` algorithm?

Datasets The real-world datasets used for evaluations are as follows:

CQA. We use Mathematics Stack Exchange (*Math*) and Stack Overflow (*SO*) data from [121]. The questions are whole and answers are parts both with voting scores as outcome. For each question, we treat all the answers it receives as its composing parts. The extracted features are described in [121].

DBLP. DBLP dataset provides the bibliographic information of computer science research papers. We treat authors as whole with h-index as outcome and papers as parts with citation counts as outcome. For each author, his/her composing parts are the papers s/he has co-authored. Paper features include temporal attributes and author features include productivity and social attributes.

Movie. We crawl the metadata of 5,043 movies with budget information[5] from IMDb website. The meta information includes movie title, genres, cast, budget, etc. We treat movies as whole and the actors/actresses as parts both with the number of Facebook likes as the outcome. For each movie, we treat its cast as the composing parts. Movie features include contextual attributes and actors/actresses features include productivity and social attributes.

The statistics of these datasets are summarized in Table 3.6. For each dataset, we first sort the whole in chronological order, gather the first x percent of whole and their corresponding parts as training examples and always test on the last 10% percent of whole and their corresponding parts. The percentage of training x could vary. The root mean squared error (RMSE) between the actual outcomes and the predicted ones is adopted for effectiveness evaluation. The parameters are set for each method on each dataset via a grid search.

Repeatability of experimental results: all the datasets are publicly available. We have released the datasets and code of the proposed algorithms through authors'

[5] `http://www.the-numbers.com/movie/budgets/all`

Data	Whole	Part	# of whole	# of part
Math	Question	Answer	16,638	32,876
SO	Question	Answer	1,966,272	4,282,570
DBLP	Author	Paper	234,681	129,756
Movie	Movie	Actors/Actresses	5,043	37,365

Table 3.6: Summary of Datasets for PAROLE.

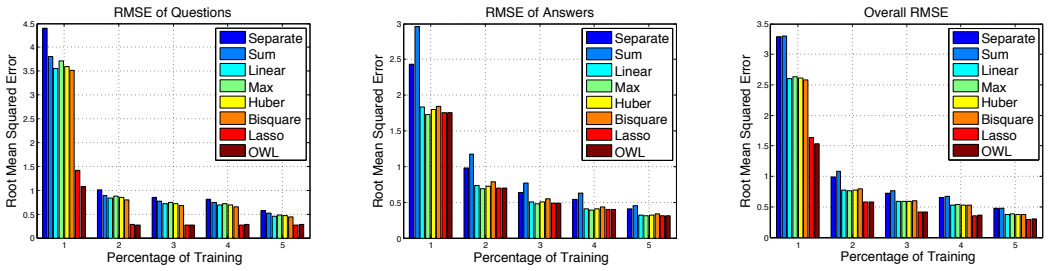
website. The experiments are performed on a Windows machine with four 3.5GHz Intel Cores and 256GB RAM.

Effectiveness Results We compare the effectiveness of the following methods:

1. *Separate:* train a linear regression model for parts and whole separately.
2. *Sum:* a joint model with Sum part-whole relationship.
3. *Linear:* our PAROLE with linear aggregation.
4. *Max:* our PAROLE with maximum aggregation.
5. *Huber:* our PAROLE with robust Huber estimator.
6. *Bisquare:* our PAROLE with robust Bisquare estimator.
7. *Lasso:* our PAROLE with sparse aggregation.
8. *OWL:* our PAROLE with ordered sparse aggregation.

A - Outcome prediction performance: the RMSE results of all the comparison methods for predicting the outcomes of parts and whole on all the datasets are shown from Fig. 3.21 to Fig. 3.24. We draw several interesting observations from these results. First, all the joint prediction models outperform the separate model

in most cases, which suggests that the part outcome indeed has a profound impact on the whole outcome, and vice versa. Second, among the joint prediction models, in general, the linear methods (*Sum* and *Linear*) are not as good as the non-linear counterparts (*Max*, *Huber*, *Bisqaure*, *Lasso* and *OWL*), and in some cases (Fig. 3.21b, Fig. 3.23b), the linear joint models are even worse than the separate method, which indicates that the part-whole relationship is indeed more complicated than the linear aggregation. Third, among the non-linear methods, a consistent observation across all the datasets is that *Lasso* and *OWL* are the best two methods in almost all the cases. This suggests that the whole outcome is mostly dominated by a few, often high-performing, parts.



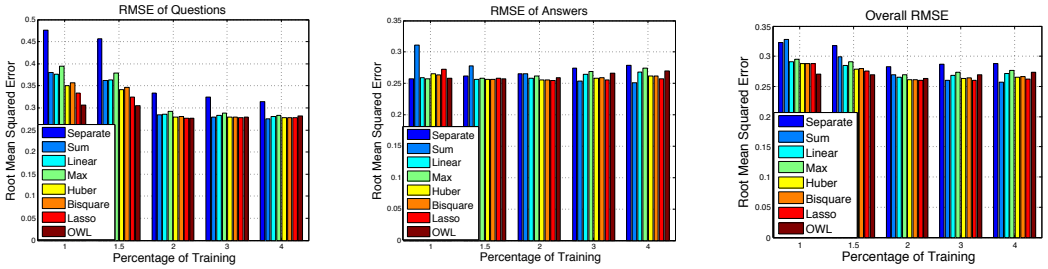
(a) RMSE of question outcome prediction.

(b) RMSE of answer outcome prediction.

(c) Overall RMSE.

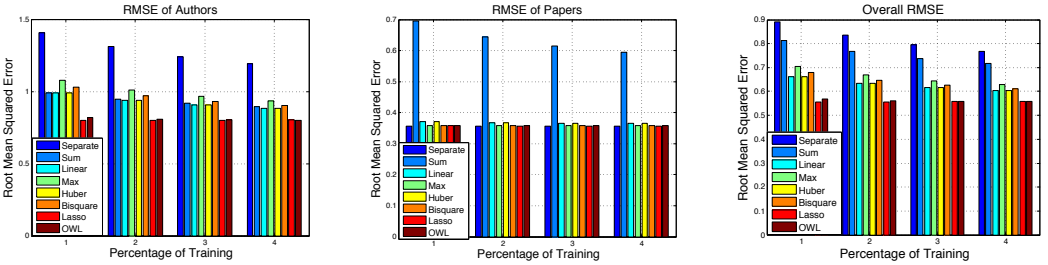
Figure 3.21: RMSE comparisons on *Math*. Best viewed in color. From left to right: *Separate*, *Sum*, *Linear*, *Max*, *Huber*, *Bisquare*, *Lasso* and *OWL*.

B - The effect of part-part interdependency: in the proposed joint prediction model, we have hypothesized that the part-part interdependency might help boost the predictions in two ways, i.e., regularizing the parts' contribution to the whole outcome as well as part outcome correlation. Here, we verify and validate to what extent these two aspects contribute to the performance gain, when such part-part interdependency information is available. Fig. 3.25 shows the results of *Lasso*



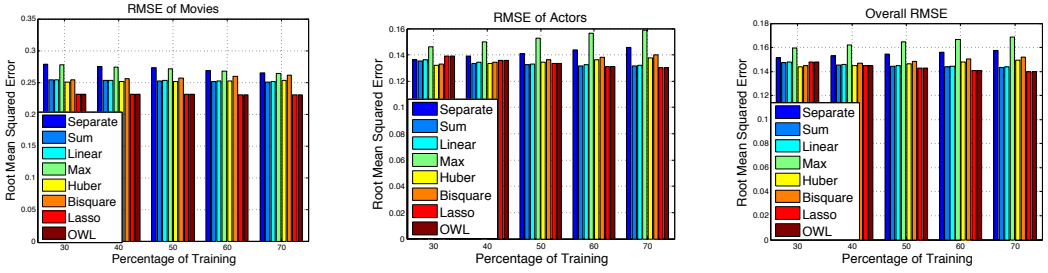
(a) RMSE of question outcome prediction. (b) RMSE of answer outcome prediction. (c) Overall RMSE.

Figure 3.22: RMSE comparisons on *SO*. Best viewed in color. From left to right: *Separate*, *Sum*, *Linear*, *Max*, *Huber*, *Bisquare*, *Lasso* and *OWL*.



(a) RMSE of author outcome prediction. (b) RMSE of paper outcome prediction. (c) Overall RMSE.

Figure 3.23: RMSE comparisons on *DBLP*. Best viewed in color. From left to right: *Separate*, *Sum*, *Linear*, *Max*, *Huber*, *Bisquare*, *Lasso* and *OWL*.



(a) RMSE of movie outcome prediction. (b) RMSE of actors/actress outcome prediction. (c) Overall RMSE.

Figure 3.24: RMSE comparisons on *Moive*. Best viewed in color. From left to right: *Separate*, *Sum*, *Linear*, *Max*, *Huber*, *Bisquare*, *Lasso* and *OWL*.

on the *Movie* dataset with 50% training data. The network among the parts, i.e., actors/actresses, is their collaboration network. The "PAROLE-Basic" does not use the network information. The "PAROLE-GraphForWhole" applies the graph regularization on the parts' contribution to the whole, which brings a 8% overall prediction error reduction. On top of that, "PAROLE-GraphForWhole&Parts" uses the graph regularization on the parts' outcome, which brings a 14.5% decrease in the overall prediction error.

C - Convergence analysis: Fig. 3.26 shows the objective function value vs. the number of iterations on the *SO* dataset using *OWL* with 5% training data. As we can see, the proposed PAROLE algorithm converges very fast, after 25-30 iterations.

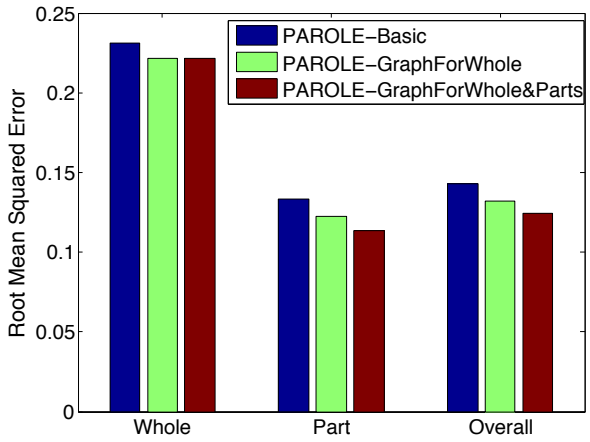


Figure 3.25: Performance gain analysis on *Movie*.

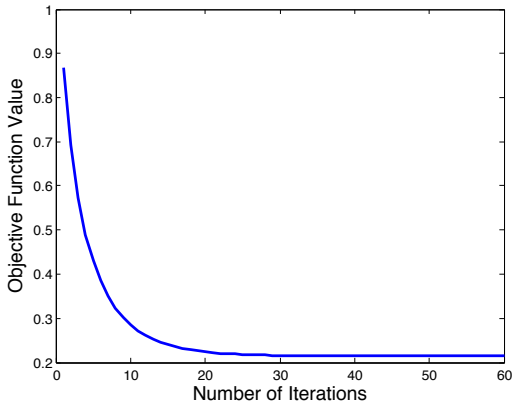


Figure 3.26: Convergence analysis on *SO*.

D - Sensitivity analysis: to investigate the parameter sensitivity, we perform parametric studies with the two most important parameters in PAROLE, i.e., α that controls the importance of part-whole relationship and β that controls the importance of part-part interdependency on the parts outcome. The bowl shaped surface in Fig. 3.27 suggests that the proposed model can achieve good performance in a large volume of the parameter space.

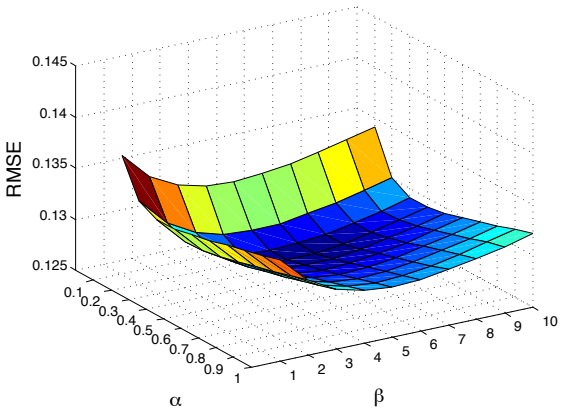


Figure 3.27: RMSE with varying α and β of *Lasso* on *Movie*.

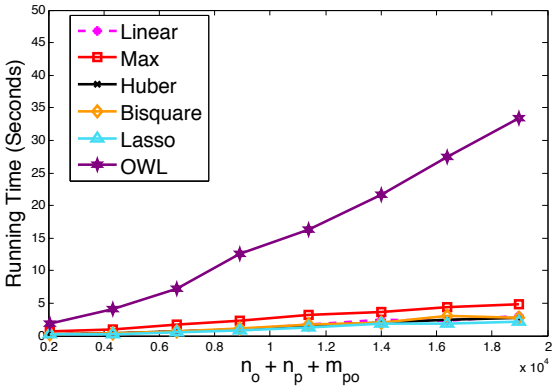


Figure 3.28: Scalability plot on *SO*.

Efficiency Results Fig. 3.28 shows the running time of all the proposed methods with varying size of training data ($n_o + n_p + m_{po}$). We can see that all the proposed methods scale linearly, which is consistent with Lemma 3. *OWL* takes the longest time due to the additional sorting operation in the proximal-gradient update for the OWL norm.

Chapter 4

TEAM PERFORMANCE OPTIMIZATION

In this chapter, we introduce our work on team performance optimization [75, 77, 76, 20]. We start with the problem of team member replacement, to recommend a good candidate to replace a churning team member and introduce a graph kernel based algorithm that considers both the structural matching and skill matching. We then extend this solution to other team enhancement scenarios. We also work towards real-time team optimization by leveraging reinforcement learning techniques.

4.1 Team Member Replacement

In his world-widely renowned book "The Science of the Artificial" [104], 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.

A promising algorithmic approach to team composition treats a team as a subgraph embedded in a larger social network. Prior research has focused on assembling a team from scratch while satisfying the skill requirements at minimum communication cost (e.g., diameter and minimum spanning tree) [67]. If the tasks arrive in an online fashion, the workload balance among the people needs to be considered [2]. In practical scenarios, there are more realistic requirements in the team formation, e.g., inclusion of a designated leader and size of a team [96]. With the increasing constraints, the team formation problem is NP-complete. Prior work to formulate

automated ways of forming a team has used heuristic approaches (e.g., RarestFirst and SteinerTree) but so far lacks efficient solutions [67]. Our work differs from previous efforts in three ways: (1) we alter the composition of an existing team based on new requirements; (2) we solve the problem in a principled approach with the notation of graph kernel; and (3) we design a set of efficient algorithms.

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 (e.g., from the other members within the organization), when a team member becomes unavailable?* This is the central question this work aims to answer. However, despite the frequency with which people leave a team before a project/task is complete and the resulting disruption [125], replacements are often found opportunistically and are not necessarily optimal.

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 [54] and that distributed teams perform better when members know each other [33]. Furthermore, research has shown that specific communication patterns amongst team members are critical for performance [24]. 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 specifically, 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 of the team members.

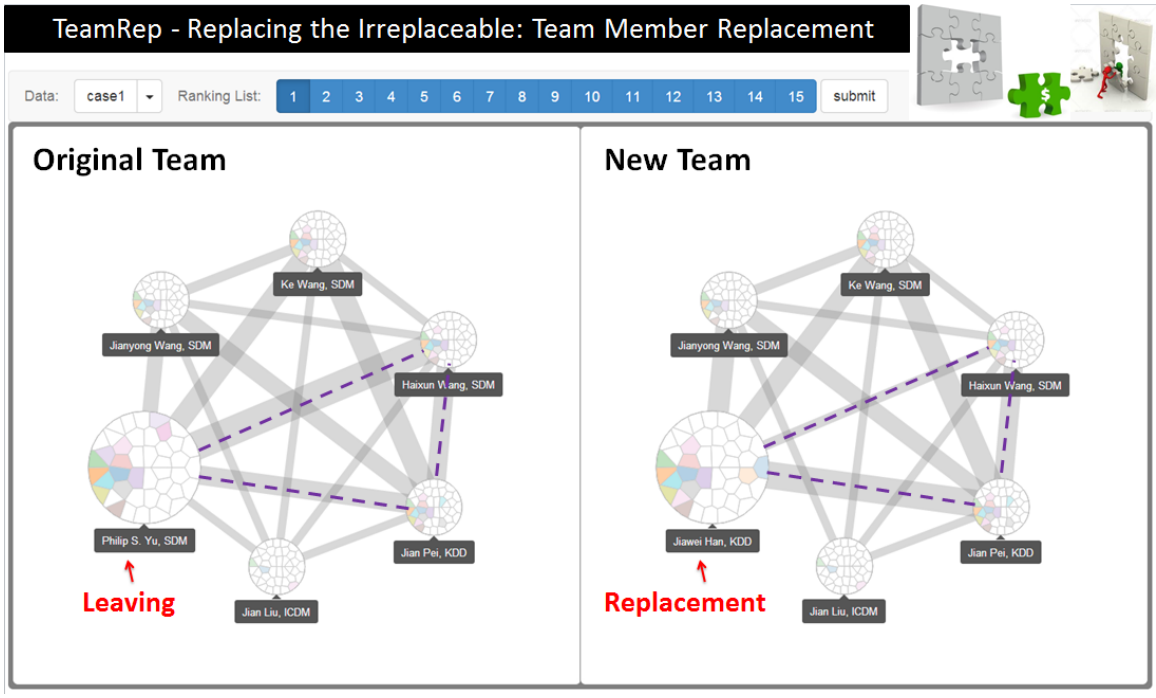


Figure 4.1: The team graphs before and after *Jiawei Han* takes *Philip S. Yu*'s place. See subsection 4.1.4 for detailed explanations.

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, the graph kernel provides a natural way to capture both the skill and structure match as well as the interaction of both. However, for a network with n individuals, a straightforward 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. Specifically, 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, 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 work are as follows.

1 **Problem Formulation.** We formally define the Team Member Replacement problem, to recommend a good candidate when a team member is unavailable in the context of networks where nodes carrying on multiple labels (skills) and edges representing social structures.

2 **Algorithms and Analysis.** We solve the problem by introducing graph kernels and 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, including user studies and case studies, on real world datasets, to validate the effectiveness and efficiency of our methods. (See an example in Figure 4.1.)

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 $\mathbf{A}$

Table 4.1: Symbols of Team Member Replacement

4.1.1 Problem Definitions

Table 4.1 lists the main symbols used throughout this work. We describe the n individuals by a 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 $\mathbf{L}$ 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:

Problem 5. 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* $\mathbf{G}(\mathcal{T})$*.*

4.1.2 Proposed Solutions

In this subsection, we present our solution for Problem 5. 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.

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 4.1.1, it naturally leads to the following two design objectives for a good Team Member Replacement:

- *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.
- *Structural matching*: the new member should have a similar network structure as team member p in connecting the rest of the team members.

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 as well as the interaction of both. 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

$$q = \text{argmax}_{j, j \notin \mathcal{T}} \ \ \text{Ker}(\mathbf{G}(\mathcal{T}), \mathbf{G}(\mathcal{T}_{p \to j})) \tag{4.1}$$

In Eq. (4.1), $\mathbf{G}(\mathcal{T})$ 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. As such, graph kernel is able to simultaneously capture both the skill matching and the structure matching, beyond the simple ad-hoc combination between the two (e.g., weighted linear combination, multiplicative combination, sequential filtering, etc). We would like to emphasize that this treatment is important - as we will show in the experimental section, it leads to much better performance over all the alternative choices. 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 4.1) 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 [67, 2, 96]. The existing work on team formation aims to build a team from scratch by optimizing some pre-chosen metric (e.g., compatibility, diversity, etc). In contrast, we aim to find a new team member such that the new team resembles the original team as much as

possible.

Having this in mind, many of the existing graph kernels can be adopted in Eq. (4.1), such as random walk based graph kernel, sub-tree based graph kernels (See Section 2.2 for a review). In this study, 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 [111]:

$$\text{Ker}(\mathbf{G}_1, \mathbf{G}_2) = \mathbf{y}'(\mathbf{I} - c\mathbf{A}_\times)^{-1}\mathbf{L}_\times\mathbf{x} \tag{4.2}$$

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 $\mathbf{L}_\times = \sum_{k=1}^{l} \text{diag}(\mathbf{L}_1(:, k)) \otimes \text{diag}(\mathbf{L}_2(:, k))$.

Computational Challenges Eq.(4.2) 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.(4.2); 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 $O(nt^3)$ 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 $O(t^3)$ [111]. Even if we allow some approximation in computing each of these graph kernels, the best known algorithms (i.e., by [60]) would still give an overall time complexity as $O(n(lt^2r^4 + mr + r^6))$, 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.

4.1.3 Scale-up and Speed-up

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

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. 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 4. 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. (4.2), we have

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

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

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

$$\begin{aligned} &\mathbf{L}_{\times 1}(\mathbf{A}'_0 \otimes \mathbf{A}'_1) = (\textstyle\sum_{k=1}^{l} \text{diag}(\mathbf{L}_0(:, k)) \otimes \text{diag}(\mathbf{L}_1(:, k)))(\mathbf{A}'_0 \otimes \mathbf{A}'_1) \\ &= \textstyle\sum_{k=1}^{l} (\text{diag}(\mathbf{L}_0(:, k))\mathbf{A}'_0) \otimes (\text{diag}(\mathbf{L}_1(:, k))\mathbf{A}'_1) \end{aligned} \tag{4.3}$$

Because $(\text{diag}(\mathbf{L}_1(:, k))\mathbf{A}'_1) \geq (\text{diag}(\mathbf{L}_2(:, k))\mathbf{A}'_2)$, 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 4, 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 computations from $O(n)$ to $O(\sum_{i \in \mathcal{T}/p} d_i)$, which is sub-linear in n.

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 $\text{diag}(\mathbf{L}_1(:, j))$ as $\mathbf{L}_1^{(j)}$ and $\text{diag}(\mathbf{L}_2(:, j))$ 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 j^{th} 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

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

labelled graph can be computed as:

$$\begin{aligned}
&\mathrm{Ker}(\mathbf{G}(\mathcal{T}), \mathbf{G}(\mathcal{T}_{p\to q})) = \mathbf{y}'(\mathbf{I} - c(\textstyle\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_2^{(j)})(\mathbf{A}_1' \otimes \mathbf{A}_2'))^{-1}(\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_2^{(j)})\mathbf{x} \\
&= \mathbf{y}'(\underbrace{\mathbf{I} - c(\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})(\mathbf{A}_1 \otimes \mathbf{A}_c)}_{\mathbf{Z}\text{: invariant w.r.t. } q} \underbrace{-c(\sum_{j=1}^{l} (\mathbf{L}_1^{(j)} \otimes \mathbf{e}^{(j)})(\mathbf{I} \otimes \mathbf{f}^{(j)}))(\mathbf{A}_1 \otimes \mathbf{A}_c)}_{\mathbf{PQ}(\mathbf{A}_1 \otimes \mathbf{A}_c) = \mathbf{P}\mathbf{Y}_1} \\
&\underbrace{-c(\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})(\mathbf{A}_1 \otimes \mathbf{E})(\mathbf{I} \otimes \mathbf{F})}_{\mathbf{X}_1\mathbf{Y}_2} \\
&\underbrace{-c(\sum_{j=1}^{l} (\mathbf{L}_1^{(j)} \otimes \mathbf{e}^{(j)})(\mathbf{I} \otimes \mathbf{f}^{(j)}))(\mathbf{A}_1 \otimes \mathbf{E})(\mathbf{I} \otimes \mathbf{F}))^{-1}}_{\mathbf{X}_2\mathbf{Y}_2}(\textstyle\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_2^{(j)})\mathbf{x}
\end{aligned} \tag{4.4}$$

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. (4.4) as

$$\begin{aligned}
&\mathrm{Ker}(\mathbf{G}(\mathcal{T}), \mathbf{G}(\mathcal{T}_{p\to q})) = \mathbf{y}'(\mathbf{Z} - c\mathbf{X}\mathbf{Y})^{-1}(\textstyle\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_2^{(j)})\mathbf{x} \\
&= \mathbf{y}'(\mathbf{Z}^{-1} + c\mathbf{Z}^{-1}\mathbf{X}(\mathbf{I} - c\mathbf{Y}\mathbf{Z}^{-1}\mathbf{X})^{-1}\mathbf{Y}\mathbf{Z}^{-1}) \\
&((\textstyle\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})\mathbf{x} + (\sum_{j=1}^{l} (\mathbf{L}_1^{(j)} \otimes \mathbf{e}^{(j)})(\mathbf{I} \otimes \mathbf{f}^{(j)}))\mathbf{x})
\end{aligned} \tag{4.5}$$

where the second equation is due to the matrix inverse lemma [49].

Remarks. In Eq. (4.5), $\mathbf{Z} = \mathbf{I} - c(\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})(\mathbf{A}_1 \otimes \mathbf{A}_c)$ does not depend on the candidate 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. (4.4)), $(\mathbf{I} - c\mathbf{Y}\mathbf{Z}^{-1}\mathbf{X})$ 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.

Speedup Graph Kernel - Approx Approach Note that the graph kernel by Eq. (4.5) is exactly the same as the original method by the first equation in Eq. (4.4). 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 $\hat{\mathbf{A}}_c = \mathbf{UV}$, 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 $\hat{\mathbf{A}}_1 = \hat{\mathbf{A}}_c + \mathbf{A}_{d1} = \mathbf{UV} + \mathbf{E}_1\mathbf{F}_1 = \mathbf{X}_1\mathbf{Y}_1$, 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:

$$
\begin{aligned}
&\hat{\text{Ker}}(\mathbf{G}(\mathcal{T}), \mathbf{G}(\mathcal{T}_{p\to q})) = \mathbf{y}^T(\mathbf{I} - c\mathbf{L}_\times(\hat{\mathbf{A}}_1' \otimes \hat{\mathbf{A}}_2'))^{-1}\mathbf{L}_\times\mathbf{x} \\
&= \mathbf{y}'(\mathbf{I} - c\mathbf{L}_\times(\mathbf{X_1}\mathbf{Y_1}) \otimes (\mathbf{X_2}\mathbf{Y_2}))^{-1}\mathbf{L}_\times\mathbf{x} = \mathbf{y}'(\mathbf{I} - c\mathbf{L}_\times(\mathbf{X_1} \otimes \mathbf{X_2})(\mathbf{Y}_1 \otimes \mathbf{Y}_2))^{-1}\mathbf{L}_\times\mathbf{x} \\
&= \mathbf{y}'(\mathbf{I} + c\mathbf{L}_\times(\mathbf{X_1} \otimes \mathbf{X_2})\mathbf{M}(\mathbf{Y}_1 \otimes \mathbf{Y}_2))\mathbf{L}_\times\mathbf{x} \\
&= \mathbf{y}'\mathbf{L}_\times\mathbf{x} + c\mathbf{y}'(\textstyle\sum_{j=1}^{l} \mathbf{L}_1^{(j)}\mathbf{X_1} \otimes \mathbf{L}_2^{(j)}\mathbf{X_2})\mathbf{M}(\mathbf{Y}_1 \otimes \mathbf{Y}_2)\mathbf{L}_\times\mathbf{x} \\
&= (\textstyle\sum_{j=1}^{l}(\mathbf{y}_1'\mathbf{L}_1^{(j)}\mathbf{x}_1)(\mathbf{y}_2'\mathbf{L}_2^{(j)}\mathbf{x}_2)) + c(\sum_{j=1}^{l} \mathbf{y}_1'\mathbf{L}_1^{(j)}\mathbf{X_1} \otimes \mathbf{y}_2'\mathbf{L}_2^{(j)}\mathbf{X_2})\mathbf{M}(\sum_{j=1}^{l} \mathbf{Y}_1\mathbf{L}_1^{(j)}\mathbf{x}_1 \otimes \mathbf{Y}_2\mathbf{L}_2^{(j)}\mathbf{x}_2)
\end{aligned}
\tag{4.6}
$$

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 [60] 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 [60]. In contrast, we only need to compute top-r approximation *once* by Eq. (4.6). As our complexity analysis (subsection 4.1.3) and experimental evaluations (subsection 4.1.4) show, this brings a few times additional speed-up.

Putting Everything Together Putting everything 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.

Variant #1: TeamRep-Fast-Exact

We first present our algorithm using the exact graph kernel computation in Eq. (4.5). The algorithm (TEAMREP-FAST-EXACT) is summarized in Algorithm 5. 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).

The effectiveness and efficiency of TEAMREP-FAST-EXACT are summarized in Lemma 5 and Lemma 6, respectively. Compared with TEAMREP-BASIC, Algorithm 5 is much faster without losing any recommendation accuracy.

Lemma 5. *Accuracy of* TEAMREP-FAST-EXACT. *Algorithm 5 outputs the same set of candidates as* TEAMREP-BASIC.

Proof. (Sketch) First, according to Lemma 4, we will not miss a promising candidate during the pruning stage. Second, for each candidate after pruning, Algorithm 5 calculates its graph kernel exactly the same as Eq. (4.5), which is in turn the same as Eq. (4.4) and hence Eq. (4.2). Therefore, after ranking the scores, Algorithm 5 outputs the same set of candidates as TEAMREP-BASIC. □

Lemma 6. *Time Complexity of* TeamRep-Fast-Exact *Algorithm 5 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 5) 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 5 is $O((\sum_{i\in\mathcal{T}/p} d_i)(lt^5 + l^3t^3))$. □

Variant #2: TeamRep-Fast-Approx

By using Eq. (4.6) to compute the graph kernel instead, we propose an even faster algorithm (TeamRep-Fast-Approx), which is summarized in Algorithm 6. 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.

The effectiveness and efficiency of TeamRep-Fast-Approx are summarized in Lemma 7 and Lemma 8, respectively. Compared with TeamRep-Basic and TeamRep-Fast-Exact, Algorithm 6 is even faster; and the only place it introduces the approximation error is the low-rank approximation of $\mathbf{A}_c$ (step 2).

Lemma 7. *Accuracy of* TeamRep-Fast-Approx. *If* $\mathbf{A}_c = \mathbf{U}\mathbf{\Lambda}\mathbf{U}'$ *holds, Algorithm 6 outputs the same set of candidates as* TeamRep-Basic.

Proof. Omitted for brevity. □

Lemma 8. *Time Complexity of* TeamRep-Fast-Approx *Algorithm 6 takes* $O((\sum_{i\in\mathcal{T}/p} d_i)(lt^2r + r^6))$ *in time.*

Proof. Omitted for brevity. □

4.1.4 Experimental Evaluations

In this subsection, 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?

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

Table 4.2: Summary of Datasets for Team Member Replacement.

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) to reflect authors' skills (*e.g., data mining, data base, computer vision, etc*) and for a given author and conference, we define his/her skill level as the percentage of the papers s/he publishes in that conference. 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.

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

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 datasets are summarized in Table 4.2. 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 have released the code of the proposed algorithms through authors' website.

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. 4.1, which shows a screenshot of our current demo system. This prototype system has been developed to a fully functional system and deployed to real users [20]. The original team is shown

[3]http://grouplens.org/datasets/hetrec-2011/

[4]http://www.databasebasketball.com

on the left side and the person leaving the team (*Philip S. Yu*) is represented by a node (diagram) 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 [41]. 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 is presented by the line thickness.

Fig. 4.1 visualizes the team structures before and after *Philip S. Yu* becomes unavailable in the team writing [94]. 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 sub-structures 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*. Af-

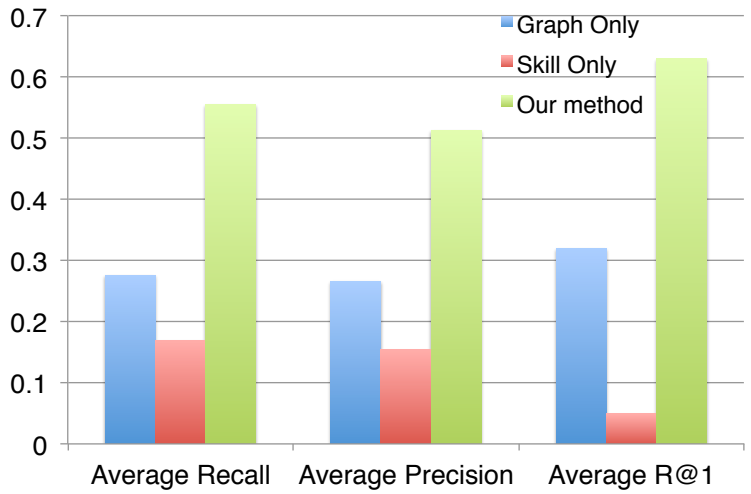


Figure 4.2: The average recall, average precision and R@1 of the three comparison methods. Higher is better.

ter 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*, *Christo-*

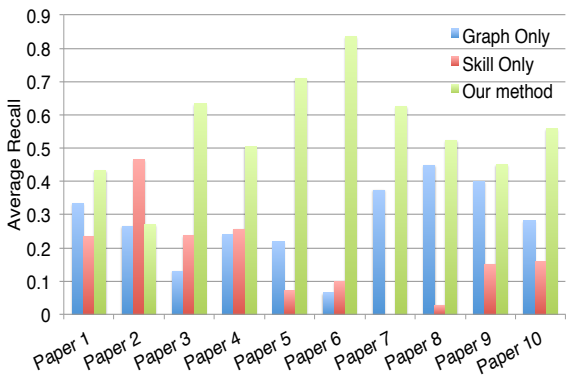


Figure 4.3: Recall for different papers. Higher is better.

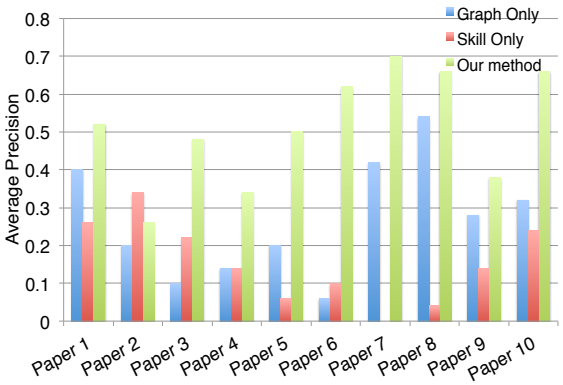


Figure 4.4: Precision for different papers. Higher is better.

pher 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. 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. Recall that we have two design objectives for our TEAM MEMBER REPLACEMENT problem, including both the *skill match* and the *structural match*. Our quantitative evaluations focus on the following two aspects. First, we examine whether simultaneously considering both design objectives outperform only considering one of them. Second, we evaluate to what extent our graph kernel formulation outperforms other alternative choices, in order to fulfill both design objectives (i.e., the skill match and the structural match). To be specific, we compare to the following alternative methods, including (a) only with *structure matching* and not including $\mathbf{L}_\times$ in Eq. (4.2) (Graph Only),(b) only with *skill matching* and using cosine similarity of skill vectors as scores (Skill Only), (c) using the weighted sum of scores by 'Skill Only' and 'Graph Only' (Linear Combination), (d) using the multiplication of the two (Multiplicative Combination), and (e) first picking those with high 'Skill Only' scores and then ranking them by 'Graph Only' scores (Sequential Filtering).

User studies. We perform a user study with 20 people aged from 22 to 35 as follows. We choose 10 papers from various fields, replace one author of each paper, run our method and the first two 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. 4.2, Fig. 4.3 and Fig. 4.4, 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

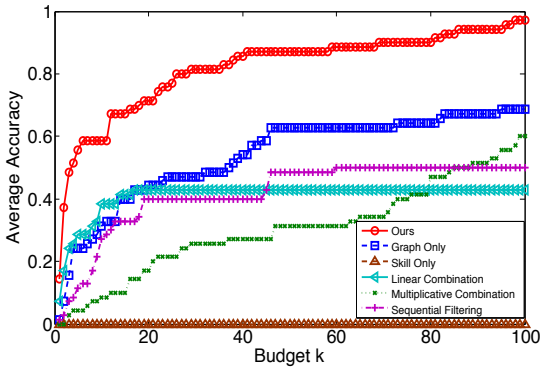


Figure 4.5: Average accuracy vs. budget k. Higher is better.

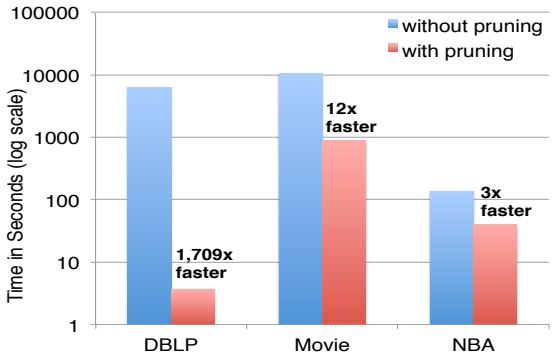


Figure 4.6: Time Comparisons before and after pruning on three datasets. Notice time is in log-scale.

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. 4.5. Again, our method performs best - it outperforms both the methods that consider only one design objective ('skill only' and 'graph only'); and that use alternative ad-hoc methods to combine both skill and structural match ('linear combination', 'multiplicative combination' and 'sequential filtering').

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 their running time. For *DBLP*, we choose the authors of paper [72] (6 authors); for *Movie*, we select the film crew of *Titanic* (1997) (22 actors/actresses); for *NBA*, we pick the players on the Los Angeles Lakers in year 1996 (17 players). The result is presented in Fig. 4.6. As we can see, the pruning step itself brings significant savings

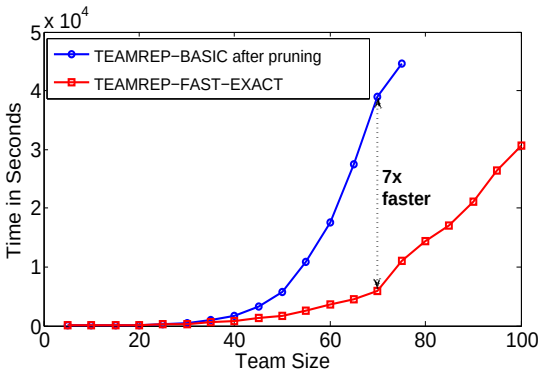


Figure 4.7: Time Comparison between TeamRep-Basic and TeamRep-Fast-Exact. TeamRep-Fast-Exact is on average 3× faster. TeamRep-Basic takes more than 10 hours when team size = 70.

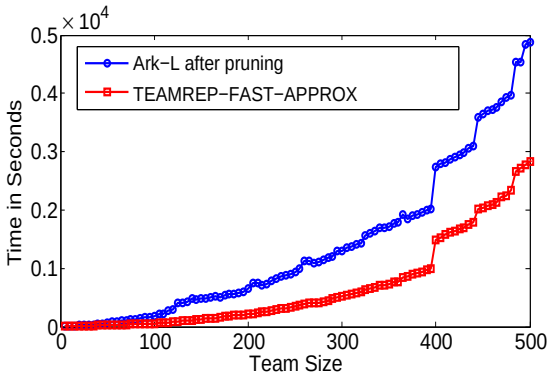


Figure 4.8: Time Comparisons between Ark-L[20] and TeamRep-Fast-Approx. TeamRep-Fast-Approx is on average 3× faster.

in terms of running time, especially for larger graphs (e.g., *DBLP* and *Movie*). Notice that according to Lemma 4, 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 [60] with TeamRep-Fast-Approx (approximate methods). For TeamRep-Basic and Ark-L, we apply the same pruning step as their pre-processing step. The results on *DBLP* are presented in Fig. 4.7 and Fig. 4.8, 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 method 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

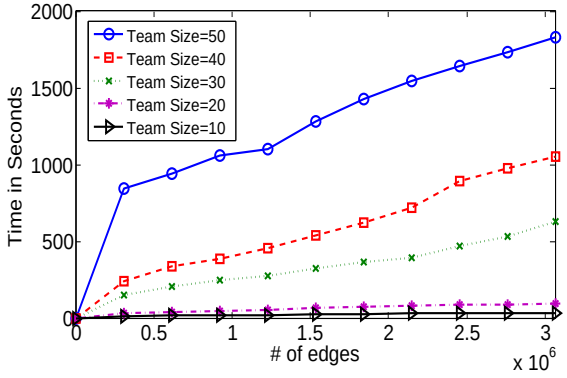


Figure 4.9: Running time of TeamRep-Fast-Exact vs. graph size. TeamRep-Fast-Exact scales sub-linearly w.r.t. the number of edges of the input graph.

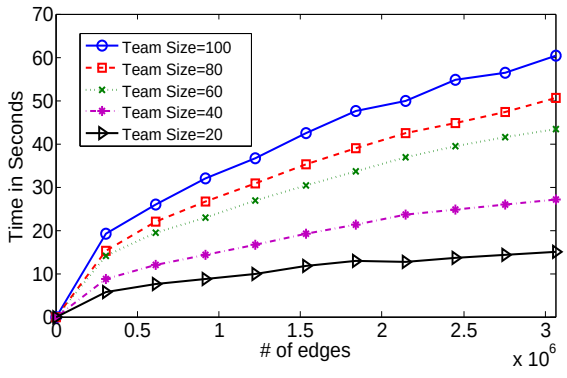


Figure 4.10: Running time vs. graph size. TeamRep-Fast-Approx scales sub-linearly w.r.t. the number of edges of the input graph.

DBLP network and run the two proposed algorithms on teams with different sizes. The results are presented in Fig. 4.9 and Fig. 4.10, 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).

4.2 Beyond Team Member Replacement

Different from Team Member Replacement, Team Refinement considers refining a team by replacing one member with another with the desired skill sets and communication connections. In the above two problems, the team size remains the same. In Team Expansion, we want to expand the team by adding a member with certain skill sets and communication structure. For instance, a software project team wants to develop a new feature of natural language search and a new member with Natural Language Processing (NLP) skill will be recruited. On the contrary, in Team Shrinkage, the size of a team needs to be reduced in response to new challenge such as a shortage of the available resource (e.g., a budget cut). In all cases, the resulting

disruption [125] should be minimized.

By careful inspection, we identify the problem similarity between Team Refinement, Team Expansion and Team Member Replacement and propose these problems can be formulated in a way to share common technical solutions. In Team Refinement, one team member is edited to a desired skill and network structure configuration. Since such edited member might not exist in the rest of the network, we call it a 'virtual member'. By replacing this 'virtual member' as in Team Member Replacement, we can solve Team Refinement. Similarly, in Team Expansion, the desired new member might also be a 'virtual member'. After adding this 'virtual member' to the current team and then replacing the 'virtual member', we can solve Team Expansion. We propose to reduce the disruption induced by the team alteration by maintaining the team-level similarity (between the original and the new teams), which includes skill similarity as well as structural similarity. The proposition is backed by some recent studies which show that team members prefer to work with people they have worked with before [54] and that distributed teams perform better when members know each other [33]. Furthermore, research has shown that specific communication patterns amongst team members are critical for performance [24].

4.2.1 Problem Definitions

In addition to the notations defined in Sec. 4.1.1, we define for the i^{th} individual, the associated skill vector as $\mathbf{l} = \mathbf{L}(i,:)$ and communication structure vector as $\mathbf{a} = \mathbf{A}(i,:)$. If we lay off an existing member $p \in \mathcal{T}$ of a given team $\mathcal{T}$, the new team members are indexed by $\mathcal{T}_{/p} := \{\mathcal{T}/p\}$; and the new team is represented by the labelled network $\mathbf{G}(\mathcal{T}_{/p})$.

With the above notations and assumptions, the other team enhancement problems can be formally defined as follows:

Problem 6. Team Refinement

Given: *(1) A labelled social network* $\mathbf{G} := \{\mathbf{A}, \mathbf{L}\}$*, (2) a team* $\mathbf{G}(\mathcal{T})$*, (3) a team member* $p \in \mathcal{T}$*, and (4) desired skill* $\mathbf{l}$ *and communication structure* $\mathbf{a}$ *for* p*;*

Recommend: *A candidate* $q \notin \mathcal{T}$ *with skill* $\mathbf{l}$ *and communication structure* $\mathbf{a}$ *to refine the person* p*'s role in the team* $\mathbf{G}(\mathcal{T})$*.*

Problem 7. Team Expansion

Given: *(1) A labelled social network* $\mathbf{G} := \{\mathbf{A}, \mathbf{L}\}$*, (2) a team* $\mathbf{G}(\mathcal{T})$*, and (3) desired skill* $\mathbf{l}$ *and communication structure* $\mathbf{a}$ *for a new member;*

Recommend: *A new member* $q \notin \mathcal{T}$ *with skill* $\mathbf{l}$ *and communication structure* $\mathbf{a}$ *to join the team* $\mathbf{G}(\mathcal{T})$*.*

Problem 8. Team Shrinkage

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

Recommend: *A member* $p \in \mathcal{T}$ *to leave the team* $\mathbf{G}(\mathcal{T})$*.*

4.2.2 Beyond Team Member Replacement*:* Team Refinement*,* Team Expansion *and* Team Shrinkage

In this subsection, we discuss how the techniques for Team Member Replacement can be applied to the other team enhancement scenarios, including Team Refinement, Team Expansion and Team Shrinkage. We note that the fast solutions developed in Section 4.1.3 also apply to these scenarios, and thus omit the detailed discussions.

Team Refinement In TEAM REFINEMENT, we want to edit a current team member p to have the desired skill $\mathbf{l}$ and communication structure vector $\mathbf{a}$. As the person with the exact skill and communication requirements might not exist in the network, we aim to find a best-effort match. We define a 'virtual member' v to be the person with skill $\mathbf{l}$ and network structure $\mathbf{a}$ and a 'virtual team' $\mathcal{T}'$ to be $\mathcal{T}_{p\to v}$. Using graph kernel, the best-effort match q can be found as:

$$q = \text{argmax}_{j, j\notin\mathcal{T}} \ \ \text{Ker}(\mathbf{G}(\mathcal{T}'), \mathbf{G}(\mathcal{T}'_{v\to j})) \tag{4.7}$$

Team Expansion In TEAM EXPANSION, we want to add a team member with the desired skill $\mathbf{l}$ and communication structure vector $\mathbf{a}$. Again, because the exact match might not exist, we instead find a best-effort match. We define a 'virtual member' v to be the person with skill $\mathbf{l}$ and network structure $\mathbf{a}$ and a 'virtual team' $\mathcal{T}'$ to be $\{\mathcal{T}, v\}$. Using graph kernel, the best-effort match q can be found as:

$$q = \text{argmax}_{j, j\notin\mathcal{T}} \ \ \text{Ker}(\mathbf{G}(\mathcal{T}'), \mathbf{G}(\mathcal{T}'_{v\to j})) \tag{4.8}$$

Team Shrinkage In TEAM SHRINKAGE, we want to remove a current team member with minimum disruption. Since graph kernel can characterize the team-level similarity, it can also be applied to TEAM SHRINKAGE. The idea is to find a current team member p so that the new team after p leaves is most similar to the old team. That is, we want to find a member $p \in \mathcal{T}$ such that:

$$p = \text{argmax}_{j\in\mathcal{T}} \ \ \text{Ker}(\mathbf{G}(\mathcal{T}), \mathbf{G}(\mathcal{T}_{/j})) \tag{4.9}$$

where $\mathbf{G}(\mathcal{T}_{/j})$ is the labelled team graph after a team member j leaves. Note that in TEAM SHRINKAGE, the search space is no longer the rest network but the team itself, which is much smaller.

4.2.3 Experimental Evaluations

Case studies on TEAM EXPANSION. Suppose we want to expand the organizing committee of *KDD* 2013 by hiring a researcher with strong expertise in Artificial Intelligence, and preferably who has collaborated with as many researchers on the committee as possible. The top five candidates found by our algorithm are: *Qiang Yang*, *Zoubin Ghahramani*, *Eric Horvitz*, *Thomas G. Dietterich* and *Raymond J. Mooney*. All the candidates have made significant contributions to the field of artificial intelligence and *Yang*, *Horvitz*, *Dietteirch* and *Mooney* are the current AAAI fellows. Among them, *Yang* has collaborated with some previous KDD organizing committee members (e.g., *Jian Pei*, *Ying Li*, *Geoff Webb* and *Dou Shen*).

Team Shrinkage. In *DBLP*, we select teams with over 10 members and manually inject a "noisy" individual to the team such that the individual is connected with all the team members with random edge weights and has randomly generated skill vectors. Recall that, in team shrinkage we want to find the "best" member to leave the team without much disruption to the team. In our setting, we treat the "noisy" individual as the "best" candidate. For "Skill Only", we first compute the similarity matrix among all team members using inner product of their skill vectors and then apply max-pooling as their score. Figure 4.11 shows the result of our method, "Graph Only" as well as "Skill Only". Our method achieves the best Precision@1, Recall@1 and F@1.

4.3 Towards Real Time Team Optimization

Teams can be often viewed as a dynamic system where the team configuration evolves over time (e.g., new members join the team; existing members leave the team; the skills of the members improve over time, etc.). It is hypothesized that

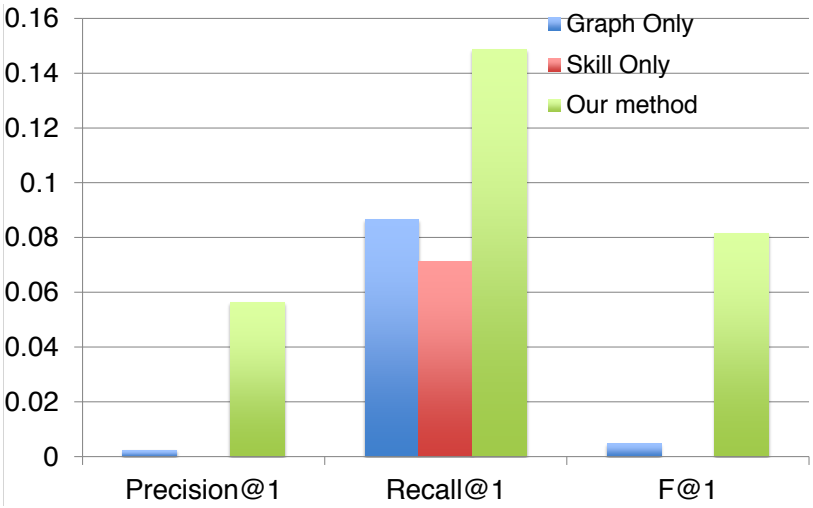


Figure 4.11: Precision@1, Recall@1 and F@1 of the three comparison methods for Team Shrinkage. Higher is better.

newly formed teams evolve through a series of development stages, notably *forming*, where the formation of the team starts; *storming*, where team members explore the situation; *norming*, where members accommodate, form and accept roles; and *performing*, where the team produces effective outcomes [109]. Although teams might take different paths towards maturity, research suggests that the effective cooperation and coordinations among team members generally brings the team from initial ineptness to the final levels of skilled performance [89]. In the context of sports teams and software development teams, research efforts have been on the relationship between team dynamics and team performances [115, 39].

Due to the team dynamics, the performance of the team is very likely to be changing. If a team fails to achieve satisfactory performance or adjustment to environmental demands are required, changes to the team are necessitated. A natural question is how to plan the team optimization/re-staffing actions (e.g., recruit a new team member) at each time step so as to maximize the expected cumulative performance of the team. Most existing work on team optimization (e.g., team replacement [75]

and team enhancement [77]) treat teams as a *static* system and recommend a single action to optimize a *short-term* objective. However, these approaches might fail due to the unique challenges brought about by the dynamics in team processes. First (*team dynamics*), the teams are constantly changing in their compositions and existing methods are not designed to learn the kind of changes that are effective in producing the teams' high performances. A straightforward way of applying existing methods for team optimization is to recommend one action at one time. However, this treatment is problematic in two ways: (1) the existing methods are optimizing a different objective and they cannot adjust their strategy based on the feedback (e.g., performance evaluation, team cohesion) to the team; and (2) the existing methods can not be computed on the fly in situations where real-time decisions are required. Second (*long-term reward*), teams are expected to deliver constantly good performance in the long run. The actions recommended by existing methods are purposed to optimize the short-term feedback, but might not lead to a long-term reward.

In this work, we treat the actions a team takes during its development cycles as sequential interactions between the team agent and the environment and propose to leverage Reinforcement Learning (RL) to automatically learn the optimal staffing strategies. Such team optimization based on reinforcement learning have two advantages. First, it is able to continuously update its staffing strategy during the interactions from the feedback at each time step, until it converges to the optimal strategy. Second, the models are trained via estimating the current value for a state-action pair with delayed rewards. The optimal strategy is able to maximize the expected cumulative rewards from the environment. In other words, it might recommend an action with small short-term rewards but have a big impact of the team performance in the long run. One challenge here is that the state/action space (e.g., the possible enhancement operations and their combinations over time) could be large. It is thus

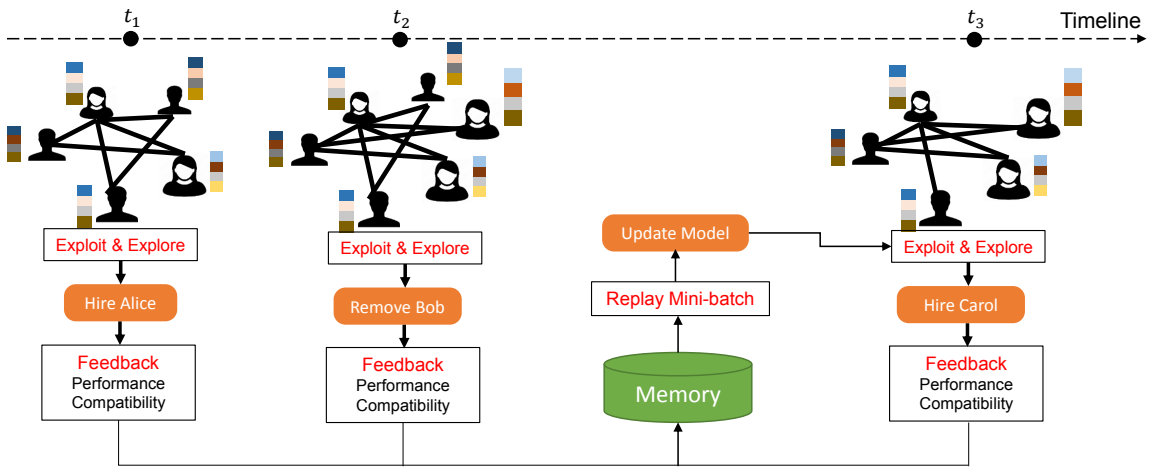


Figure 4.12: Running example of real time team optimization.

infeasible to evaluate the value for every state-action pair. Instead, we leverage value based approach and use a function approximator to estimate the state-action value in RL. This model-free approach does not estimate the transition probability nor explicitly store the Q-value table, making it flexible to handle the large state/action space in the team optimization scenarios.

4.3.1 Problem Definition

We describe the n individuals by a 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 $\mathbf{L}$ is $n \times l$ skill indicator matrix. 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.

We study the real-time team optimization problem in which a team (agent) interacts with environment by sequentially taking enhancement actions (e.g., hiring a new

team member) over a sequence of time steps, so as to maximize its cumulative reward (see Fig. 4.12 for an example). We model this problem as a Markov Decision Process (MDP), which includes a sequence of states, actions and rewards. More formally, MDP consists of a tuple of five elements $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ as follows:

- **State space** $\mathcal{S}$: A state $s_t \in \mathcal{S}$ is defined as the team configuration at time step t, which can be described by $\mathbf{G}(\mathcal{T}) := \{\mathbf{A}(\mathcal{T}, \mathcal{T}), \mathbf{L}(\mathcal{T}, :)\}$.

- **Action space** $\mathcal{A}$: The action $a_t \in \mathcal{A}$ is to take enhancement actions to the team, e.g., expand/shrink the team, establish collaboration between two team members, etc. Formally, a_t could be $\Delta\mathbf{A}(\mathcal{T}, \mathcal{T})$ (perturbation to the team network structure), $\Delta\mathbf{L}(\mathcal{T}, :)$ (perturbation to the team skill configuration), $+q$ (hiring q to join the team), and $-q$ (remove q from current team).

- **Reward** $\mathcal{R}$: After the team takes an action a_t at the state s_t, i.e., the team configuration changes at time t, the team receives rewards $r(s_t, a_t)$ according to the feedback it receives (e.g., performance evaluation, team cohesion).

- **Transition probability** $\mathcal{P}$: Transition probability $p(s_{t+1}|s_t, a_t)$ defines the probability of state transitioning from s_t to s_{t+1} when the team takes action a_t. We assume that the MDP satisfies $p(s_{t+1}|s_t, a_t, \ldots, s_1, a_1) = p(s_{t+1}|s_t, a_t)$.

- **Discount factor** γ: $\gamma \in [0, 1]$ defines the discount factor when we measure the present value of future reward. In particular, when $\gamma = 0$, it only considers the immediate reward; when $\gamma = 1$, all future rewards can be fully taken into account.

With the notations and definitions above, the problem of real time team optimization can be formally defined as follows:

Problem 9. *Real-Time Team Optimization*

Given: *the historical MDP, i.e., $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$*

Find: *a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, which can maximize the cumulative reward of the team*

4.3.2 Proposed Model

In this section, we introduce our proposed model based on reinforcement learning framework for the purpose of real-time team optimization. We propose to use a function approximator to estimate the state-action value without explicitly storing them into a lookup table.

The Classic Model – Q-Learning We follow the standard assumption that delayed rewards are discounted by a factor of γ per time step, and define the state-action value function $Q(s, a)$ as the expected rewards from state s and action a. Using Bellman optimality equation [8], the optimal state-action function $Q^*(s, a)$ can be written as follows via one-step look-ahead:

$$Q^*(s, a) = \mathbb{E}_\pi[r + \gamma \max_{a'} Q^*(s', a')|s, a] \tag{4.10}$$

We can use Q-learning control algorithm to update the Q values toward the optimal ones at each step of each episode as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \tag{4.11}$$

where α is the step size.

The limitations with the above standard reinforcement learning model are two folds: (1) in the real-time team optimization scenarios, the state/action space are enormous, which makes it infeasible to estimate $Q^*(s, a)$ for every state-action pair using the above update equation; and (2) many state and action pairs may not appear

in the log of the team development, in which case we will not have an accurate estimate for them.

The proposed Value Function Approximation Framework We propose to use a parameterized function to approximate the value of $Q^*(s,a)$ as $Q^*(s,a) \approx q(s,a;\theta)$ parameterized by θ. If we use a linear approximation function, we can represent $q(s,a;\theta)$ as $q(s,a;\theta) = x(s,a)^T\theta$, where $x(s,a)$ is the feature vector that describe the state-action pair. If the function cannot be well approximated by a linear one, we can represent it as a non-linear function, e.g., a deep neural network. The parameters of the value function approximator can be trained via minimizing the following loss function $L(\theta)$ as:

$$L(\theta) = \mathbb{E}_{s,a,r,s'}[(y - q(s,a;\theta))^2], \tag{4.12}$$

where $y = \mathbb{E}_\pi[r + \gamma \max_{a'} q(s',a';\theta^t)|s,a]$ is the target for the current iteration and θ^t is the parameters from the last iteration. The derivatives of the loss function $L(\theta)$ with respect to θ can be written as:

$$\Delta_\theta L(\theta) = \mathbb{E}_{s,a,r,s'}[(r + \gamma \max_{a'} q(s',a';\theta^t) - q(s,a;\theta))\Delta_\theta q(s,a;\theta)] \tag{4.13}$$

To optimize the loss function, it is more efficient to apply the stochastic gradient descent instead of the full expectations in the above gradient.

Features for state-action: suppose at state s_t, the team can be described by $\mathbf{G}(\mathcal{T}) := \{\mathbf{A}(\mathcal{T},\mathcal{T}), \mathbf{L}(\mathcal{T},:)\}$ and the action a_t is q, i.e., hiring q to join the team. The features we consider that can describe this state-action pair are:

- the average skill vector of the team at t
- the maximum skill vector of the team at t
- clustering coefficient of the team at t

- the average degree of the team at t
- the average network embeddings of the team at t
- the skill vector of q
- the local clustering coefficient of q
- the network embedding of q

Note that we use DeepWalk [95] on the entire social network within the organization to get the embeddings of all the individuals.

Off-policy training: We train the parameters of the model from the offline log of different teams' development, including the actions the team takes and the reward it gets. The off-policy training algorithm is presented in Algorithm 7.

Algorithm 5: 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

Initialize $\mathbf{A}_c, \mathbf{L}_1^{(j)}, \mathbf{L}_2^{(j)}, j = 1, \ldots, l$;

Pre-compute $\mathbf{Z}^{-1} \leftarrow (\mathbf{I} - c(\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})(\mathbf{A}_1 \otimes \mathbf{A}_c))^{-1}$;

Set $\mathbf{R} \leftarrow (\sum_{j=1}^{l} \mathbf{L}_1^{(j)} \otimes \mathbf{L}_c^{(j)})\mathbf{x}$; $\mathbf{b} \leftarrow \mathbf{y}^T\mathbf{Z}^{-1}\mathbf{R}$; $\mathbf{l} \leftarrow c\mathbf{y}^T\mathbf{Z}^{-1}$;

for *each candidate* q *in* $\mathbf{G}$ *after pruning* **do**

  Initialize $\mathbf{s} \leftarrow$ a zero vector of length t except the last element is 1;

  Initialize $\mathbf{w} \leftarrow$ weight vector from q to the new team members;

  Set $\mathbf{E} \leftarrow [\mathbf{w}, \mathbf{s}]; \mathbf{F} \leftarrow [\mathbf{s}'; \mathbf{w}']$;

  Set $\mathbf{e}^{(j)} \leftarrow$ a t by 1 zero vector except the last element is 1, for $j = 1, \ldots, d_n$;

  Set $\mathbf{f}^{(j)} \leftarrow$ a $1 \times t$ zero vector except the last element which is label j assignment for q;

  Set $\mathbf{P} \leftarrow [\mathbf{L}_1^{(1)} \otimes \mathbf{e}^{(1)}, \ldots, \mathbf{L}_1^{(l)} \otimes \mathbf{e}^{(l)}]$;

  Set $\mathbf{Q} \leftarrow [\mathbf{I} \otimes \mathbf{f}^{(1)}; \ldots; \mathbf{I} \otimes \mathbf{f}^{(l)}]$;

  Compute $\mathbf{X}_1 \leftarrow (\sum_{j=1}^{l} \mathbf{L}_1^{(j)}\mathbf{A}_1 \otimes \mathbf{L}_c^{(j)}\mathbf{E})$;

  Compute $\mathbf{X}_2 \leftarrow (\sum_{j=1}^{l} \mathbf{L}_1^{(j)}\mathbf{A}_1 \otimes \mathbf{e}^{(j)}\mathbf{f}^{(j)}\mathbf{E})$;

  Compute $\mathbf{Y}_1 \leftarrow \mathbf{Q}(\mathbf{A}_1 \otimes \mathbf{A}_c)$;

  Compute $\mathbf{Y}_2 \leftarrow (\mathbf{I} \otimes \mathbf{F})$;

  Set $\mathbf{X} \leftarrow [\mathbf{P}, \mathbf{X}_1, \mathbf{X}_2], \mathbf{Y} \leftarrow [\mathbf{Y}_1; \mathbf{Y}_2; \mathbf{Y}_2]$;

  Update $\mathbf{M} \leftarrow (\mathbf{I} - c\mathbf{Y}\mathbf{Z}^{-1}\mathbf{X})^{-1}$;

  Compute $\mathbf{r}' \leftarrow \mathbf{Z}^{-1}\mathbf{P}\mathbf{Q}\mathbf{x}$;

  Compute score$(q) = \mathbf{b} + \mathbf{y}^T\mathbf{r}' + \mathbf{l}\mathbf{X}\mathbf{M}\mathbf{Y}(\mathbf{Z}^{-1}\mathbf{R} + \mathbf{r}')$;

end

Return the top k candidates with the highest scores.

Algorithm 6: 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

Initialize $\mathbf{A}_c, \mathbf{L}_1^{(j)}, \mathbf{L}_2^{(j)}, j = 1, \ldots, l$;

Compute top r eigen-decomposition for $\mathbf{A}_c$: $\mathbf{U}\mathbf{\Lambda}\mathbf{U}' \leftarrow \mathbf{A}_c$;

Set $\mathbf{V} \leftarrow \mathbf{\Lambda}\mathbf{U}'$;

Initialize $\mathbf{s} \leftarrow$ a zero vector of length t except the last element is 1;

Initialize $\mathbf{w}_1 \leftarrow$ weight vector from p to $\mathcal{T}$;

Set $\mathbf{E}_1 \leftarrow [\mathbf{w}_1, \mathbf{s}], \mathbf{F}_1 \leftarrow [\mathbf{s}'; \mathbf{w}_1']$;

Set $\mathbf{X}_1 \leftarrow [\mathbf{U}, \mathbf{E}_1]$, $\mathbf{Y}_1 \leftarrow [\mathbf{V}; \mathbf{F}_1]$;

for *each candidate* q *in* $\mathbf{G}$ *after pruning* **do**

  Initialize $\mathbf{w}_2 \leftarrow$ weight vector from q to the new team members ;

  Set $\mathbf{E}_2 \leftarrow [\mathbf{w}_2, \mathbf{s}], \mathbf{F}_2 \leftarrow [\mathbf{s}'; \mathbf{w}_2']$;

  Set $\mathbf{X}_2 \leftarrow [\mathbf{U}, \mathbf{E}_2]$, $\mathbf{Y}_2 \leftarrow [\mathbf{V}; \mathbf{F}_2]$;

  Compute $\mathbf{S} \leftarrow \sum_{j=1}^{l} \mathbf{y}_1' \mathbf{L}_1^{(j)} \mathbf{X}_1 \otimes \mathbf{y}_2' \mathbf{L}_2^{(j)} \mathbf{X}_2$;

  Compute $\mathbf{T} \leftarrow \sum_{j=1}^{l} \mathbf{Y}_1 \mathbf{L}_1^{(j)} \mathbf{x}_1 \otimes \mathbf{Y}_2 \mathbf{L}_2^{(j)} \mathbf{x}_2)$;

  Update $\mathbf{M} \leftarrow (\mathbf{I} - c(\sum_{j=1}^{l} \mathbf{Y}_1 \mathbf{L}_1^{j} \mathbf{X}_1 \otimes \mathbf{Y}_2 \mathbf{L}_2^{j} \mathbf{X}_2))^{-1}$;

  Set score$(q) = (\sum_{j=1}^{l} (\mathbf{y}_1' \mathbf{L}_1^{(j)} \mathbf{x}_1)(\mathbf{y}_2' \mathbf{L}_2^{(j)} \mathbf{x}_2)) + c\mathbf{SMT}$;

end

Return the top k candidates with the highest scores.

Algorithm 7: Off-policy Training for Real Time Team Optimization

Initialize the capacity of replay memory $\mathcal{D}$;
Initialize action-value function q with random weights;
for *episode* $= 1, \ldots, M$ **do**
    Initialize s_0 from some previous episode;
    for $t = 1, \ldots, T$ **do**
        Observe state s_t;
        Execute action a_t following the off-policy and observe reward r_t;
        Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$;
        Sample transitions (s, a, r, s') from $\mathcal{D}$;
        Minimize $(y - q(s, a; \theta))^2$ according to Eq. (4.13)
    end
end

Chapter 5

TEAM PERFORMANCE EXPLANATION

In this chapter, we introduce our work on team performance explanation [132, 78]. We start with the explanation model we build for understanding networked prediction systems, i.e., the team performance prediction models introduced in Chapter 3, and then continue with our effort on explaining the team performance optimization model introduced in Chapter 4.

5.1 Towards Explainable Networked Prediction

Networked prediction has attracted lots of research attention in recent years. Networks, as a natural data model that captures the relationship among different objects, domains and learning components, provide powerful contextual information in modeling networked systems, including network of networks [91, 92, 27], network of time series [18, 19], network of learning models [73, 63]. Networked prediction has been successfully applied in many application domains, ranging from bioinformatics, environmental monitoring, infrastructure networks, to team science.

By leveraging the intrinsic relationship among the networked learning components, it often brings significant performance improvement to the mining tasks. In a network of networks, each node of the main network is itself another domain network. For example, in the candidate gene prioritization problem where a disease similarity network is given, a tissue-specific protein interaction network can be associated with a corresponding disease. The main network can contextualize the mining tasks in each domain-specific network by providing the consistency constraints across networks for both ranking [91] and clustering [92]. In the network of coevolving time series, e.g., a

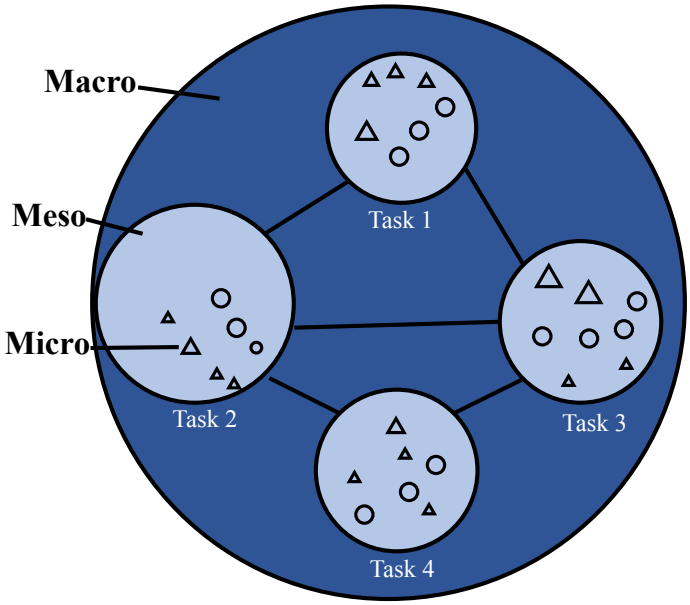


Figure 5.1: An illustration of networked prediction system.

sensor network where sensors measure the temperature time series in different locations of a building, encoding the network constraints finds similar latent factors from similar time series for imputation and classification tasks [18, 19]. In the network of learning models, the scientific domain similarity network provides natural constraints to the citation prediction models for each domain such that similar domains would share a similar regression model [73]; or in the multi-task learning setting, the task network enables the graph regularized multi-task learning formulation [63].

Despite its superior prediction power, networked prediction is often hard to understand for end users. Compared with traditional learning setting, networked prediction is even harder to explain due to its coupled, multi-level nature. The learning process propagates top-down through the underlying network from the macro level (i.e., the entire learning system), to meso level (i.e., learning tasks), and to micro level (i.e., individual learning examples). See Fig. 5.1.

- *Macro level.* At this highest level, we want to study the whole networked learning system to gain a global view of how the system works. What are the ingredients that are essential to the system characteristics, e.g., the parameters of the entire system?

- *Meso level.* At this level, we focus on one specific learning task and aim to understand its own learning behavior, e.g., how its own training samples and those from other learning tasks affect its model parameters via the network as the bridge.

- *Micro level.* At this finest granularity, we focus on one specific test example and want to understand the reasons behind the prediction of this test example given by the learned models, e.g., how the training examples from the same task and from the other related tasks affect its prediction.

On the other hand, we envision that the networked prediction setting also offers rich context to explain the learning process through the lens of various aspects as follows:

- *Example aspect.* Each training example could potentially shape the learned model of the same task and that of the other tasks via the underlying network. We want to identify the most influential examples at the different levels (i.e., macro, meso and micro levels) to have a comprehensive understanding of the roles the training examples play in the learning process.

- *Task aspect.* A learning task, if viewed as the aggregation of its training examples, would affect the learning process of the whole system as well as each of the other learning tasks. We seek to identify the important learning tasks at the three different levels.

- *Network aspect.* A task network is essential in the networked learning system and plays a unique role as it acts as a bridge to connect all the learning tasks together. Changing the task network would inevitably influence the learning results of the whole system as well as each individual learning task.

Following the above discussion, we propose a multi-aspect, multi-level approach to explain networked prediction. The key idea is to efficiently quantify the influence on different levels of the learning system due to the perturbation of various aspects. More concretely, the influence score is measured by the changes in the entire learning system's parameters (*macro*), one task's model parameters (*meso*), and the loss function value at a test sample (*micro*) in response to the changes made to the training examples, a learning task and the task network, respectively.

The key advantages are (1) *multi-aspect, multi-level:* we are able to provide a comprehensive explanation to the workings of networked prediction from the perspective of multiple aspects at multiple levels, essentially through the influences of example-task-network aspects with respect to macro-meso-micro levels; and (2) *efficiency:* with the help of influence functions which is rooted in robust statistics [32], we can efficiently evaluate the influences of changes to the networked prediction without retraining the whole learning system, which is often time consuming. Furthermore, we observe that the majority of the training examples have negligible influences at the three different levels, paving the way for us to design a safe pruning strategy to filter out those examples to further speed up the computations.

The main contributions of this work can be summarized as:

- **Problem Definitions.** We formulate the problem to demystify the mechanisms behind networked predictions from multiple aspects (example-task-network) at multiple levels (macro-meso-micro).

- **Algorithms and Analysis.** We propose an algorithm (`NEPAL`) to measure the influence of examples, tasks and network at the macro, meso, and micro levels, and design an effective pruning strategy to filter out the examples with negligible influence. We also provide theoretical analysis regarding the complexity and correctness of the proposed algorithm.

- **Empirical Evaluations.** We carefully design the empirical evaluations on real world datasets and demonstrate the effectiveness of the proposed multi-aspect, multi-level approach for understanding the networked prediction.

5.1.1 Problem Definition

In this subsection, we present the notations used throughout the work (summarized in Table 5.1), and formally define the Explainable Networked Prediction problem. We use bold capital letters (e.g., $\mathbf{A}$) for matrices and bold lowercase letters (e.g., $\mathbf{w}$) for vectors.

Let us consider a networked learning system with T supervised learning tasks, for example, recognizing objects from images or predicting the sentiment from texts. The training data we have for each task is given as $\{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^{n_t} \subset \mathbb{R}^d \times \mathbb{R}$, $t = 1, \ldots, T$, where n_t is the number of available training examples for the t-th task, and d is the dimensionality of the input space, which is assumed to be shared across the tasks. In this work, we assume a task relationship network described by a non-negative matrix $\mathbf{A}$ is available. In this network, each node represents a task and the edges represent the relatedness between the connected tasks, i.e., $\mathbf{A}(i, j)$ has a higher numerical value if the i-th and j-th tasks are closely related. The goal of networked prediction is to learn a prediction function parameterized by θ_t as $f_t(\mathbf{x}_i^t; \theta_t)$ for each task jointly in

Symbols	Definition
T	the number of tasks
d	feature dimensionality
$\{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^{n_t}$	training examples for the t-th task
$f_t(\mathbf{x}_i^t; \theta_t)$	prediction function of the t-th task parameterized by θ_t
$L(\cdot, \cdot)$	loss function
$\mathbf{A}$	task relationship network
$\mathcal{I}_G(\mathbf{x}^t)$, $\mathcal{I}_G(f_t)$, $\mathcal{I}_G(\mathbf{A}_{ij})$	macro-level influences of a training sample, a learning task and task network
$\mathcal{I}_s(\mathbf{x}^t)$, $\mathcal{I}_s(f_t)$, $\mathcal{I}_s(\mathbf{A}_{ij})$	meso-level influences of a training sample, a learning task and task network w.r.t. the s-th task
$\mathcal{I}_{\mathbf{x}_{\text{test}}^s}(\mathbf{x}^t)$, $\mathcal{I}_{\mathbf{x}_{\text{test}}^s}(f_t)$, $\mathcal{I}_{\mathbf{x}_{\text{test}}^s}(\mathbf{A}_{ij})$	micro-level influences of a training sample, a learning task and task network w.r.t. a test example $\mathbf{x}_{\text{test}}^s$

Table 5.1: Symbols of NEPAL

order to minimize the regularized empirical loss as follows:

$$\min_{\theta_1,\ldots,\theta_T} \sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} L(f_t(\mathbf{x}_i^t; \theta_t), y_i^t) + \lambda \sum_{i=1}^{T} \sum_{j=1}^{T} \mathbf{A}_{ij} \|\theta_i - \theta_j\|^2 \tag{5.1}$$

where $L(\cdot, \cdot)$ is the loss function, e.g., squared loss for regression task or cross entropy loss for classification task, and the last term is to regularize the model parameters through the task relationships.

Our goal is to demystify the networked learning system by understanding how the learning process is propagated at different levels from various aspects. In particular, given the learned models for all the tasks, we want to quantify the influence on

different levels of the learning system due to the perturbation of various aspects. More concretely, the influence score is measured by the changes in the whole learning system's parameters, one task's model parameters, and the loss function value at a test sample in response to the changes made to the training examples, a learning task and the task network.

With the above notations, the problem of explaining the networked prediction can be formally defined as follows:

Problem 10. Explainable Networked Prediction

Given: *the training data of all the tasks* $\{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^{n_t}$*, the learned models through joint training* $f_t(\cdot, \theta_t^*)$*, a query test sample from the* t*-th task* $\mathbf{x}_{\text{test}}^t$*;*

Compute: *the influence scores of the training samples, the learning tasks and the task network on the learning system's parameters, each learning task's parameters and on the prediction w.r.t.* $\mathbf{x}_{\text{test}}^t$*.*

5.1.2 Proposed Model

In this subsection, we present our explanation model `NEPAL` to help explain networked prediction by measuring the influence of the various aspects (i.e., example, task, network) at multiple levels (i.e., macro/system, meso/task, micro/example). We start with a brief review of influence functions, and then present our multi-aspect, multi-level approach to networked prediction, followed by the proof and analysis.

Preliminaries: Influence Function Influence function has been used in a single learning task to efficiently evaluate the change in model parameters due to the removal of a training sample without retraining the model [64]. For a single learning task, the objective is to minimize the empirical loss as $\theta^* = \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} L(f(\mathbf{x}_i; \theta), y_i)$. The

key idea is to compute the parameter change should a training sample is upweighted by some small ϵ, giving us new parameters $\theta^*_\epsilon = \arg\min_\theta \frac{1}{n}\sum_{i=1}^n L(f(\mathbf{x}_i;\theta),y_i) + \epsilon L(f(\mathbf{x};\theta),y)$. The influence of upweighting $\mathbf{x}$ on the parameters θ is given by

$$\mathcal{I}_\theta(\mathbf{x}) = \left.\frac{d\theta^*_\epsilon}{d\epsilon}\right|_{\epsilon=0} = -\mathbf{H}^{-1}_{\theta^*}\nabla_\theta L(f(\mathbf{x};\theta),y)$$

where $\mathbf{H}_{\theta^*} = \frac{1}{n}\sum_{i=1}^n \nabla^2_\theta L(f(\mathbf{x}_i;\theta),y_i)$ is the Hessian. Removing a training sample is equivalent to upweighting it by $\epsilon = -\frac{1}{n}$, the parameter change $(\theta^*_{-\mathbf{x}} - \theta^*)$ after the removal of the training sample $\mathbf{x}$ can be approximated by $-\frac{1}{n}\mathcal{I}_\theta(\mathbf{x})$ [64].

`NEPAL` – **Building Blocks** In this work, we introduce influence functions in the setting of a networked learning system, in order to evaluate the influences of multiple aspects at different levels. We first introduce how to use influence function to measure the learning system's parameter change due to perturbation of training examples and task network as two key building blocks.

The influence of training sample on model parameters: Removing a training example from one task would change the parameters of the task itself, but also the parameters of other tasks through the task network. We apply the similar idea as above to upweight a training example $\mathbf{x}^t$ from the t-th task and compute the changes in all the tasks' model parameters. Define the new parameters of the entire learning system after such upweighting as $\theta^*_\epsilon \stackrel{\text{def}}{=} (\theta^*_{1,\epsilon},\ldots,\theta^*_{T,\epsilon})$ and that $\theta^*_\epsilon = \arg\min \sum_{t=1}^T \frac{1}{n_t}\sum_{i=1}^{n_t} L(f_t(\mathbf{x}^t_i;\theta_t),y^t_i) + \lambda\sum_{i=1}^T\sum_{j=1}^T \mathbf{A}_{ij}\|\theta_i-\theta_j\|^2 + \epsilon L(f_t(\mathbf{x}^t;\theta_t),y^t)$. The influence of the upweighting on all the tasks' model parameters can be computed as

$$\mathcal{I}_\theta(\mathbf{x}^t) \stackrel{\text{def}}{=} \left.\frac{d\theta^*_\epsilon}{d\epsilon}\right|_{\epsilon=0} = -\mathbf{H}^{-1}_{\theta^*}\nabla_\theta L(f_t(\mathbf{x}^t;\theta_t),y^t) \tag{5.2}$$

where $\mathbf{H}_{\theta^*}$ is the Hessian of the objective function defined in Eq. (5.1). Since removing the training example $\mathbf{x}^t$ from the t-th task is the same as upweighting it by $\epsilon = -\frac{1}{n_t}$, we

can approximate the change of the parameters in the whole learning system ($\theta^*_{-\mathbf{x}^t} - \theta^*$) by $-\frac{1}{n_t}\mathcal{I}_\theta(\mathbf{x}^t)$. We show the detailed derivation in Sec. 8.

The influence of task network on model parameters: The changes in the task network $\mathbf{A}$ would also affect the whole learning system's parameters. To measure the influence of task network on model parameters, we upweight the task connection between task i and task j, i.e., $\mathbf{A}_{ij}$, and use the influence function to compute the changes of the model parameters. Define the new parameters after such upweighting as $\theta^*_\epsilon \stackrel{\text{def}}{=} (\theta^*_{1,\epsilon}, \ldots, \theta^*_{T,\epsilon})$ and that $\theta^*_\epsilon = \arg\min \sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} L(f_t(\mathbf{x}_i^t; \theta_t), y_i^t) + \lambda \sum_{i=1}^{T} \sum_{j=1}^{T} \mathbf{A}_{ij} \|\theta_i - \theta_j\|^2 + \epsilon \mathbf{A}_{ij} \|\theta_i - \theta_j\|^2$. The influence of the upweighting on all the tasks' model parameters can be computed as

$$\mathcal{I}_\theta(\mathbf{A}_{ij}) \stackrel{\text{def}}{=} \left.\frac{d\theta^*_\epsilon}{d\epsilon}\right|_{\epsilon=0} = -\mathbf{H}^{-1}_{\theta^*} \nabla_\theta \mathbf{A}_{ij} \|\theta_i - \theta_j\|^2 \tag{5.3}$$

where $\mathbf{H}_{\theta^*}$ is the Hessian of the objective function defined in Eq. (5.1). Since removing the connection between task i and task j is equivalent to upweighting $\mathbf{A}_{ij}$ by $\epsilon = -\lambda$, we can approximate the change of the parameters in the whole learning system ($\theta^*_{-\mathbf{x}^t} - \theta^*$) by $-\lambda \mathcal{I}_\theta(\mathbf{A}_{ij})$. We show the detailed derivation in Sec. 8.

`NEPAL` – **Multi-Aspect, Multi-Level** Based on the different aspects (i.e., training example, task, and task network) in the learning system, we can answer questions regarding the influences at different levels. For example, what are the most influential training samples in the whole learning system? What are the most influential learning tasks w.r.t. a test sample? See Table 5.2 for an overview.

Macro-level influences of training examples, tasks, and task network: At this macro level, we are interested in what the most influential training samples, tasks and task network connections are w.r.t. the whole learning system. We propose to use the l_2-norm of the change in the whole learning system's parameters as the measure of the macro-level influence should a training sample, training samples from a

Aspect \ Level	Macro/System	Meso/Task	Micro/Test example
Training example $\mathbf{x}^t$	globally influential training sample ($\mathcal{I}_G(\mathbf{x}^t)$)	task specific influential training sample ($\mathcal{I}_s(\mathbf{x}^t)$)	test specific influential training sample ($\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(\mathbf{x}^t)$)
Learning task f_t	globally influential task ($\mathcal{I}_G(f_t)$)	task specific influential task ($\mathcal{I}_s(f_t)$)	test specific influential task ($\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(f_t)$
Task network $\mathbf{A}$	globally influential task connections ($\mathcal{I}_G(\mathbf{A}_{ij})$)	task specific influential task connections ($\mathcal{I}_s(\mathbf{A}_{ij})$)	test specific influential task connections ($\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(\mathbf{A}_{ij})$)

Table 5.2: Multi-Aspect, Multi-Level Explanation in Networked Prediction

task, or a task connection is removed.

(1) Macro-level influence of a training sample $\mathbf{x}^t$. We use the l_2-norm of the change in all tasks' model parameters as the measure of macro-level influence of $\mathbf{x}^t$ as follows:

$$\mathcal{I}_G(\mathbf{x}^t) = \frac{1}{n_t}\|\mathcal{I}_\theta(\mathbf{x}^t)\|_2$$

(2) Macro-level influence of a learning task. For one learning task f_t, we use the average of the macro-level influences of the training samples from this task as the macro-level influence of this learning task, which is given as:

$$\mathcal{I}_G(f_t) = \frac{1}{n_t^2}\sum_{i=1}^{n_t}\|\mathcal{I}_\theta(\mathbf{x}_i^t)\|_2$$

(3) Macro-level influence of task network connection $\mathbf{A}_{ij}$. We use the l_2-norm of the change in all tasks' model parameters as the measure of macro-level influence of $\mathbf{A}_{ij}$ as follows:

$$\mathcal{I}_G(\mathbf{A}_{ij}) = \lambda\|\mathcal{I}_\theta(\mathbf{A}_{ij})\|_2$$

Meso-level influences of training examples, tasks, and task network: At this meso level, we are interested in what the most influential training samples, tasks and task

network connections are w.r.t. a specific learning task. We propose to use the l_2-norm of the change in the parameters corresponding to this learning task as the measure of the meso-level influence should a training sample, training samples from a task, or a task connection is removed. Recall that we approximate the change of the parameters in the whole learning system $(\theta^*_{-\mathbf{x}^t} - \theta^*)$ by $-\frac{1}{n_t}\mathcal{I}_\theta(\mathbf{x}^t)$. Let us denote $-\frac{1}{n_t}\mathcal{I}_{\theta_s}(\mathbf{x}^t)$ as the change corresponding to the parameters only in the s-th task.

(1) Meso-level influence of a training sample $\mathbf{x}^t$. The l_2-norm of the change in the s-th task's parameters is used as the measure of the meso-level influence of $\mathbf{x}^t$ to this task as follows:

$$\mathcal{I}_s(\mathbf{x}^t) = \frac{1}{n_t}\|\mathcal{I}_{\theta_s}(\mathbf{x}^t)\|_2$$

(2) Meso-level influence of a learning task. For one learning task s, we use the average of the meso-level influences of the training samples from the t-th task as the meso-level influence of learning task t to task s, which is given as:

$$\mathcal{I}_s(f_t) = \frac{1}{n_t^2}\sum_{i=1}^{n_t}\|\mathcal{I}_{\theta_s}(\mathbf{x}_i^t)\|_2$$

(3) Meso-level influence of task network connection $\mathbf{A}_{ij}$. The l_2-norm of the change in the s-th task's parameters is used as the measure of the meso-level influence of $\mathbf{A}_{ij}$ to task s as follows:

$$\mathcal{I}_s(\mathbf{A}_{ij}) = \lambda\|\mathcal{I}_{\theta_s}(\mathbf{A}_{ij})\|_2$$

Micro-level influences of training examples, tasks, and task network: Both the removal of a training sample and the task network connections can potentially change the parameters of all the tasks' models, which would in turn change the loss at a particular test sample $\mathbf{x}^s_{\text{test}}$ from the s-th task. We can apply chain rule to measure the influence of upweighting a training sample or task network connections on the loss function value at the test sample.

(1) Micro-level influence of a training sample $\mathbf{x}^t$. Let us first consider upweighting a training sample from the t-th task and its influence on the loss at $\mathbf{x}^s_{\text{test}}$ can be given as

$$\mathcal{I}_\theta(\mathbf{x}^t, \mathbf{x}^s_{\text{test}}) \stackrel{\text{def}}{=} \left.\frac{dL(f_s(\mathbf{x}^s_{\text{test}}; \theta^*_{s,\epsilon}), y^s_{\text{test}})}{d\epsilon}\right|_{\epsilon=0} = \nabla_\theta L(f_s(\mathbf{x}^s_{\text{test}}; \theta^*_s), y^s_{\text{test}})^T \left.\frac{d\theta^*_\epsilon}{d\epsilon}\right|_{\epsilon=0}$$
$$= -\nabla_\theta L(f_s(\mathbf{x}^s_{\text{test}}; \theta^*_s), y^s_{\text{test}})^T \mathbf{H}^{-1}_{\theta^*} \nabla_\theta L(f_t(\mathbf{x}^t; \theta_t), y^t)$$

The change of the loss function value at the test sample due to the removal of the training sample is used as the micro-level influence of $\mathbf{x}^t$ to $\mathbf{x}^s_{\text{test}}$ and can be approximated as

$$\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(\mathbf{x}^t) = -\frac{1}{n_t}\mathcal{I}_\theta(\mathbf{x}^t, \mathbf{x}^s_{\text{test}})$$

We show the algorithm for computing the micro-level influences of the training samples from all the tasks in Algorithm 8. Note that it is both computation and memory intensive to compute the inverse of the Hessian matrix especially for the large-scale networked learning problems. Instead, we use conjugate gradient optimization method to efficiently compute the inverse of the Hessian multiplied by a vector (Step 2).

(2) Micro-level influence of a learning task. For one test sample $\mathbf{x}^s_{\text{test}}$, we use the average of the micro-level influences of the training samples from the t-th task as the micro-level influence of this learning task, which is given as:

$$\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(f_t) = -\frac{1}{n_t^2}\sum_{i=1}^{n_t} \mathcal{I}_\theta(\mathbf{x}^t_i, \mathbf{x}^s_{\text{test}})$$

(3) Micro-level influence of task network connection $\mathbf{A}_{ij}$. Similarly, we can compute the influence of upweighting the task network connection $\mathbf{A}_{ij}$ on the loss at $\mathbf{x}^s_{\text{test}}$ as follows

$$\mathcal{I}_\theta(\mathbf{A}_{ij}, \mathbf{x}^s_{\text{test}}) \stackrel{\text{def}}{=} \left.\frac{dL(f_s(\mathbf{x}^s_{\text{test}}; \theta^*_{s,\epsilon}), y^s_{\text{test}})}{d\epsilon}\right|_{\epsilon=0} = \nabla_\theta L(f_s(\mathbf{x}^s_{\text{test}}; \theta^*_s), y^s_{\text{test}})^T \left.\frac{d\theta^*_\epsilon}{d\epsilon}\right|_{\epsilon=0}$$
$$= -\nabla_\theta L(f_s(\mathbf{x}^s_{\text{test}}; \theta^*_s), y^s_{\text{test}})^T \mathbf{H}^{-1}_{\theta^*} \nabla_\theta \mathbf{A}_{ij} \|\theta_i - \theta_j\|^2$$

The change of the loss function value at the test sample due to the removal of the connection between task i and j is used as the micro-level influence of $\mathbf{A}_{ij}$ and can be approximated as

$$\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(\mathbf{A}_{ij}) = -\lambda \mathcal{I}_\theta(\mathbf{A}_{ij}, \mathbf{x}^s_{\text{test}})$$

Remarks: The above micro-level influences of training samples, tasks and task network, i.e., $\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(\mathbf{x}^t)$, $\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(f_t)$ and $\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(\mathbf{A}_{ij})$, can be either positive or negative. The sign of the influence value indicates whether it helps the prediction of the test sample (i.e., reduce the loss at this test sample) or harms the prediction of the test sample (i.e., increase the loss at the test sample). The magnitude of the influence value, i.e., $|\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(\mathbf{x}^t)|$, $|\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(f_t)|$ and $|\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(\mathbf{A}_{ij})|$, indicates how much the influence is, be it positive or negative, on the test sample.

Proofs and Analysis In this subsection, we analyze the proposed `NEPAL` algorithm by giving the complexity analysis, the derivation of the key equations and its characteristics with some common loss functions for classification.

A – Complexity analysis: we analyze the time complexity of Algorithm 8 where the learning models for each task is logistic regression, i.e., $L(f_t(\mathbf{x}^t; \theta_t), y^t) = \log(1 + \exp(-y^t \theta_t^T \mathbf{x}^t))$.

Theorem 5. *(*Time complexity of `NEPAL`*). Algorithm 8 takes $O(nT^2d^2)$ with logistic regression model for each task, where $n = \sum_{t=1}^{T} n_t$ is the total number of training samples in all tasks and T is the number of tasks.*

Proof. The gradient of the loss function in logistic regression is computed as $\nabla_{\theta_t} L(f_t(\mathbf{x}^t; \theta_t), y^t) = -\sigma(-y^t \theta_t^T \mathbf{x}^t) y^t \mathbf{x}^t$. Step 1 for computing the gradient of the loss at $\mathbf{x}^s_{\text{test}}$ takes $O(d)$ time. In Step 2, the size of the Hessian matrix $\mathbf{H}_\theta$ is Td by Td, where T is the number of tasks. In the worst case scenario conjugate gradient algorithm (CG) will require Td iterations to converge, thus requiring at most the evaluation

Algorithm 8: `NEPAL` - Networked Prediction Explanation

Input: (1) the training data of all the tasks $\{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^{n_t}$, (2) the learned models through joint training $f_t(\cdot, \theta_t^*)$, (3) a query test sample from the s-th task $\mathbf{x}_{\text{test}}^s$.

Output: the micro-level influences of the training samples of all the tasks on the prediction w.r.t. $\mathbf{x}_{\text{test}}^s$.

1 Compute gradient of the loss at the test sample w.r.t. model parameters: $\mathbf{v} \leftarrow \frac{\partial L(f_s(\mathbf{x}_{\text{test}}^s;\theta_s^*),y_{\text{test}}^s)}{\partial \theta}$;

2 Compute $\mathbf{x} = \mathbf{H}_\theta^{-1}\mathbf{v}$ by solving $\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T\mathbf{H}_\theta\mathbf{x} - \mathbf{v}^T\mathbf{x}$ using conjugate gradient method, where the Hessian-vector product can be exactly computed using the $\mathcal{R}_{\mathbf{v}}\{\cdot\}$ operator [93];

3 **for** *each task t in all tasks* **do**

4 **for** $i = 1, \ldots, n_t$ **do**

5 Compute the gradient of the objective function at training sample $\mathbf{x}_i^t$ w.r.t. model parameters: $\mathbf{u} \leftarrow \frac{\partial L(f_t(\mathbf{x}_i^t;\theta_t^*),y_i^t)}{\partial \theta}$;

6 Compute the influence score of $\mathbf{x}_i^t$ as $\mathcal{I}_{\mathbf{x}_{\text{test}}^s}(\mathbf{x}_i^t) = \frac{1}{n_t}\mathbf{u}^T\mathbf{x}$;

7 **end**

8 **end**

of Td Hessian-vector multiplications, each of which takes $O(nTd)$ without explicitly forming the Hessian, where $n = \sum_{t=1}^T n_t$. In total, Step 2 takes $O(nT^2d^2)$. In the for-loops (Line 3 - Line 6), it takes $O(d)$ for each training sample, which totals $O(nd)$. In summary, the total time complexity is $O(nT^2d^2)$. □

B – Derivation of the influence functions $\mathcal{I}_\theta(\mathbf{x}^t)$ and $\mathcal{I}_\theta(\mathbf{A}_{ij})$:

Lemma 9. *(*Correctness of Eq (5.2)*). Denote* $\mathcal{J}(\theta) = \sum_{t=1}^T \frac{1}{n_t}\sum_{i=1}^{n_t} L(f_t(\mathbf{x}_i^t;\theta_t), y_i^t) + \lambda\sum_{i=1}^T\sum_{j=1}^T A_{ij}\|\theta_i - \theta_j\|^2$, *where* $\theta \stackrel{def}{=} (\theta_1, \ldots, \theta_T)$. *Assuming* $\mathcal{J}(\theta)$ *to be twice-*

differentiable and strictly convex, the influence of upweighting training sample $\mathbf{x}^t$ *on the parameters* θ *can be computed by* $\mathcal{I}_\theta(\mathbf{x}^t)$.

Proof. The Hessian matrix of $\mathcal{J}(\theta)$ is defined as:

$$\mathbf{H}_\theta \stackrel{\text{def}}{=} \nabla^2 \mathcal{J} = \sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} \nabla_\theta^2 L(f_t(\mathbf{x}_i^t; \theta_t), y_i^t) + \lambda \sum_{i=1}^{T} \sum_{j=1}^{T} \nabla_\theta^2 A_{ij} \|\theta_i - \theta_j\|^2$$

Let us upweight a training example $\mathbf{x}^t$ from the t-th task and the new parameters after such upweighting is written as $\theta_\epsilon^* = \arg\min \mathcal{J}(\theta) + \epsilon L(f_t(\mathbf{x}^t; \theta_t), y^t)$. Define the parameter change $\Delta_\epsilon = (\theta_\epsilon^* - \theta^*)$ and since θ^* does not depend on ϵ, we have the following

$$\frac{d\theta_\epsilon^*}{d\epsilon} = \frac{d\Delta_\epsilon}{d\epsilon}$$

By the first-order optimality conditions, we have

$$\nabla \mathcal{J}(\theta_\epsilon^*) + \epsilon \nabla L(f_t(\mathbf{x}^t; \theta_{t,\epsilon}^*), y^t) = 0$$

Because $\theta_\epsilon^* \to \theta^*$ as $\epsilon \to 0$, we can apply Taylor expansion to the left-hand side and get

$$[\nabla \mathcal{J}(\theta^*) + \epsilon \nabla L(f_t(\mathbf{x}^t; \theta_t^*), y^t)] + [\nabla^2 \mathcal{J}(\theta^*) + \epsilon \nabla^2 L(f_t(\mathbf{x}^t; \theta_t^*), y^t)] \Delta_\epsilon \approx 0$$

Since $\nabla \mathcal{J}(\theta^*) = 0$, we can solve for Δ_ϵ as

$$\Delta_\epsilon \approx -[\nabla^2 \mathcal{J}(\theta^*) + \epsilon \nabla^2 L(f_t(\mathbf{x}^t; \theta_t^*), y^t)]^{-1} \epsilon \nabla L(f_t(\mathbf{x}^t; \theta_t^*), y^t)$$

It can be further simplified if we only keep the $O(\epsilon)$ terms:

$$\Delta_\epsilon \approx -\nabla^2 \mathcal{J}(\theta^*)^{-1} \epsilon \nabla L(f_t(\mathbf{x}^t; \theta_t^*), y^t)$$

The influence of the upweighting can be computed as

$$\mathcal{I}_\theta(\mathbf{x}^t) \stackrel{\text{def}}{=} \left. \frac{d\theta_\epsilon^*}{d\epsilon} \right|_{\epsilon=0} = -\mathbf{H}_\theta^{-1} \nabla L(f_t(\mathbf{x}^t; \theta_t^*), y^t)$$

□

Lemma 10. *(*Correctness of Eq (5.3)*). Assuming $\mathcal{J}(\theta)$ to be twice-differentiable and strictly convex, the influence of upweighting task network connection $\mathbf{A}_{ij}$ on the parameters θ can be computed by $\mathcal{I}_\theta(\mathbf{A}_{ij})$.*

Proof. Similarly, let us upweight the task network connection and define the new parameters after such upweighting as $\theta^*_\epsilon = \arg\min \mathcal{J}(\theta) + \epsilon \mathbf{A}_{ij}\|\theta_i - \theta_j\|^2$.

By the first-order optimality conditions, we have

$$\nabla \mathcal{J}(\theta^*_\epsilon) + \epsilon \nabla \mathbf{A}_{ij}\|\theta_i - \theta_j\|^2 = 0$$

Because $\theta^*_\epsilon \to \theta^*$ as $\epsilon \to 0$, we can apply Taylor expansion to the left-hand side and get

$$[\nabla \mathcal{J}(\theta^*) + \epsilon \nabla \mathbf{A}_{ij}\|\theta_i - \theta_j\|^2] + [\nabla^2 \mathcal{J}(\theta^*) + \epsilon \nabla^2 \mathbf{A}_{ij}\|\theta_i - \theta_j\|^2)]\Delta_\epsilon \approx 0$$

Since $\nabla \mathcal{J}(\theta^*) = 0$, we can solve for Δ_ϵ as

$$\Delta_\epsilon \approx -[\nabla^2 \mathcal{J}(\theta^*) + \epsilon \nabla^2 \mathbf{A}_{ij}\|\theta_i - \theta_j\|^2]^{-1} \epsilon \nabla \mathbf{A}_{ij}\|\theta_i - \theta_j\|^2$$

It can be further simplified if we only keep the $O(\epsilon)$ terms:

$$\Delta_\epsilon \approx -\nabla^2 \mathcal{J}(\theta^*)^{-1} \epsilon \nabla \mathbf{A}_{ij}\|\theta_i - \theta_j\|^2$$

The influence of the upweighting can be computed as

$$\mathcal{I}_\theta(\mathbf{A}_{ij}) \stackrel{\text{def}}{=} \left.\frac{d\theta^*_\epsilon}{d\epsilon}\right|_{\epsilon=0} = -\mathbf{H}_\theta^{-1} \nabla \mathbf{A}_{ij}\|\theta_i - \theta_j\|^2$$

□

C – Analysis for Classification Loss: we analyze the influences from the aspect of examples at different levels with some common classification loss functions used for each learning task.

Let us first consider hinge loss used in Support Vector Machine for each task, i.e., $L(f_t(\mathbf{x}^t; \theta_t), y^t) = \max(0, 1 - y^t \theta_t^T \mathbf{x}^t)$. Let us consider the non-support vectors in

learning task t, i.e., the training samples that satisfy $y^t\theta_t^T\mathbf{x}^t > 1$. For these training samples, we know $L(f_t(\mathbf{x}^t;\theta_t), y^t) = 0$ and hence $\frac{\partial L(f_t(\mathbf{x}^t;\theta_t),y^t)}{\partial\theta_t} = \mathbf{0}$. The influence of the non-support vectors $\mathbf{x}^t$ on the parameters is therefore $\mathcal{I}_\theta(\mathbf{x}^t) = \mathbf{0}$. The macro-level influence of a non-support vector is $\mathcal{I}_G(\mathbf{x}^t) = 0$, and the meso-level influence of a non-support vector is $\mathcal{I}_s(\mathbf{x}^t) = 0$. This matches our intuition that since non-support vectors are known to have no influence on the resulting classifiers, removing them should not change the parameters in the learning task itself and also other tasks. By the same argument, the micro-level influence of the non-support vectors is $\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(\mathbf{x}^t) = 0$.

For logistic loss, $L(f_t(\mathbf{x}^t;\theta_t), y^t) = \log(1 + \exp(-y^t\theta_t^T\mathbf{x}^t))$, its gradient w.r.t. θ_t is $\nabla_{\theta_t} L(f_t(\mathbf{x}^t;\theta_t), y^t) = -\sigma(-y^t\theta_t^T\mathbf{x}^t)y^t\mathbf{x}^t$. For the training samples with large positive $y^t\theta_t^T\mathbf{x}^t$, $\sigma(-y^t\theta_t^T\mathbf{x}^t)$ is very small (close to 0), their influences at different levels are expected to be much smaller than other training samples.

In practice, from the example aspect, we are only interested in inspecting the training samples with high influence. To speed up the computation, we can first filter out those examples with large $y^t\theta_t^T\mathbf{x}^t$. Such analysis matches our empirical observation in Fig. 5.5 (macro-level) and Fig. 5.9 (micro-level).

5.1.3 Empirical Evaluations

In this subsection, we present the empirical evaluation results. The experiments are designed to evaluate the following aspects:

- *Effectiveness:* how does the proposed `NEPAL` algorithm help us understand the networked predictions?
- *Efficiency:* how fast and scalable is the proposed `NEPAL` algorithm?

Datasets The real world datasets used for evaluation are as follows:

MNIST. MNIST [69] is a commonly used handwritten digit dataset, containing images of handwritten numerals (0-9) represented by 28×28 pixels in grayscale. We construct the networked prediction system using logistic regression for three tasks, where task 1 distinguishes digit 1 from 7, task 2 differentiates digit 2 from 7 and task 3 classifies digit 6 from 9. In the task network, we connect task 1 with task 2 with $\mathbf{A}_{12} = 1$ and connect task 2 with task 3 with $\mathbf{A}_{23} = 0.1$.

Semantic Scholar [1]. This open research corpus offers over 20 million published research papers in computer science, neuroscience, and biomedical. We consider papers published between 1975 and 2005 with their features generated using information available only up to 2005. We build networked prediction models for papers published in venues in *data mining*, *computer vision*, *NLP*, *AI* and *computer networks* and a paper is classified as positive if its accumulative citation counts from 2006 to 2015 exceeds the median accumulative citation counts of the training examples from the respective domains. The features include author impact (e.g., author h-index), venue impact (e.g., venue rank). The task network is constructed based on the relevance between different research domains.

Sentiment [2]. This multi-domain sentiment dataset contains product reviews from Amazon.com for many product types [10]. We build networked prediction models for reviews from *music*, *video*, *DVD*, *book* and *magazine* and a review is labeled as positive if its rating is greater than 3 and negative if its rating is below 3. We extract both unigram and bigram features from the review text. The task network is constructed based on the relevance between different product domains.

Repeatability of experimental results: all the datasets are publicly available. We release the datasets and code of the proposed algorithms through authors' website.

[1] `http://labs.semanticscholar.org/corpus/`

[2] http://www.cs.jhu.edu/ mdredze/datasets/sentiment/

The experiments are performed on a MacBook Pro with two 2.4GHz Intel Cores and 8GB RAM.

Results on MNIST

A – Macro-level Influences. For this set of experiments, we study how the training examples, tasks and task network influences the entire learning system (i.e., the second column in Table 5.2).

(1) *Macro-level influences of training examples*: We compute the macro-level influences $\mathcal{I}_G(\mathbf{x}^t)$ for training examples of all the three learning tasks and plot their distributions in Fig. 5.2. In all the tasks, the great majority of the examples have no or negligible influences on the entire learning system and only a few can exert significant influence. The top 10 globally influential training examples measured by $\mathcal{I}_G(\mathbf{x}^t)$ is shown in Fig. 5.3 with 7 of them from the second task. The top 2 examples are the same images of digit 7 from task 2 and 1, respectively. To see how the globally influential examples affect the learning system's prediction performance, we flip the labels of the most influential examples at macro-level, retrain the model and compute the classification accuracy on the test set. We also test the random picking strategy with 30 repetitions. Fig. 5.4 shows that flipping the labels of the most influential examples exert larger interruption to the learning system as demonstrated by the significant drop in the test accuracy for all the three tasks. Our analysis in Sec. 8 shows that for the training samples with large positive $y^t\theta_t^T\mathbf{x}^t$, their macro-level influences are expected to be much smaller than other training samples. Fig. 5.5 provides empirical verification for this analysis.

(2) *Macro-level influence of a learning task*: the macro-level influences of the three learning tasks are $\mathcal{I}_G(f_1) = 0.0028$, $\mathcal{I}_G(f_2) = 0.0040$ and $\mathcal{I}_G(f_3) = 0.0037$. The second task has the highest influence on the entire system as it links task 1 and task 3.

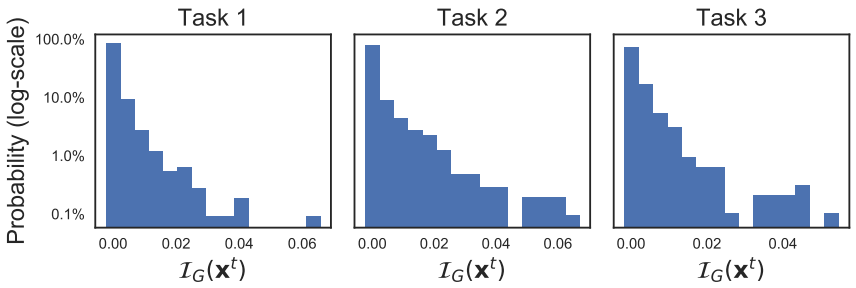


Figure 5.2: The distribution of the macro-level influences of training examples in each of the three tasks.

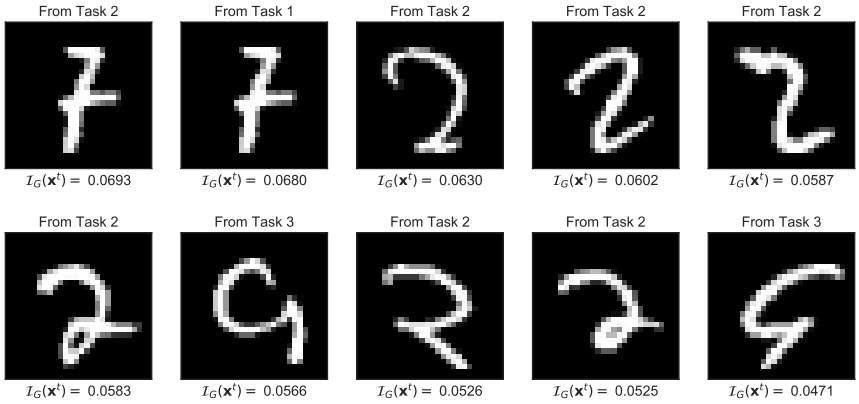


Figure 5.3: The top 10 globally influential training samples.

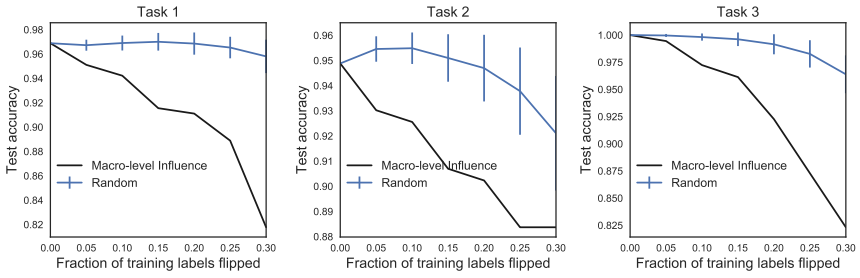


Figure 5.4: Fraction of training labels flipped vs. test accuracy.

(3) *Macro-level influence of task network*: the macro-level influences of the two links in the task network are $\mathcal{I}_G(\mathbf{A}_{12}) = 0.1898$ and $\mathcal{I}_G(\mathbf{A}_{23}) = 0.2484$. $\mathbf{A}_{23}$ is of more influence as it connects the two more relevant learning tasks.

B – Meso-level Influences. For this set of experiments, we study how the

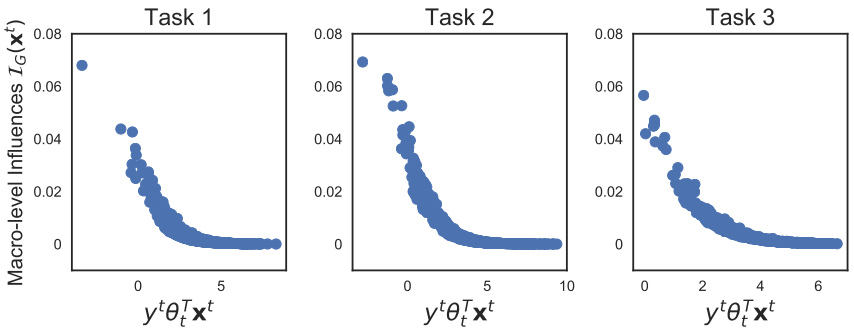


Figure 5.5: The macro-level influences of training samples vs. their $y\theta_t^T\mathbf{x}$.

training examples, tasks and task network affects the behavior of each learning task (i.e., the third column in Table 5.2).

(1) *Meso-level influences of training examples*: The distribution of the meso-level influences of the training examples w.r.t. the three learning tasks is similar to that in Fig. 5.2. We omitted these plots due to space limitation. For all the tasks, the majority of the examples have no or negligible influences on the learning tasks. Examples from the task itself tend to have a larger influence on this task. The top 5 influential training examples specific to each learning task are shown in Fig. 5.6.

(2) *Meso-level influence of a learning task*: We compute the meso-level influences of each learning task and observe that generally the most influential task for one specific task is the task itself, except that for the first task, task 2 has about the same influence on it as the task itself possibly due to the same negative training examples of digit 7 they share. For task 2, the first task has about half the influence as the task itself.

(3) *Meso-level influences of task network*: We compute the meso-level influences of the task network specific to each task. The results are consistent with our intuition. For task 1, the connection $\mathbf{A}_{12}$ has larger influence than $\mathbf{A}_{23}$; for task 2, the two task connections have about the same influences; and for task 3, the connection $\mathbf{A}_{23}$ has larger influence.

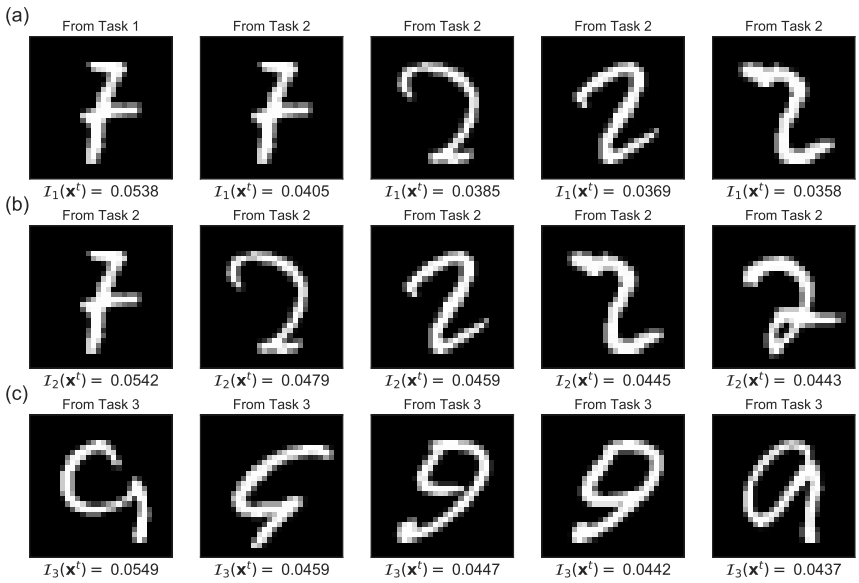


Figure 5.6: The top 5 influential training examples specific to each of the three learning tasks in (a), (b) and (c), respectively. The influence score of these training examples w.r.t. the tasks are shown under the example images.

C – Micro-level Influences. For this set of experiments, we randomly select one test example from each of the three learning tasks and study how the training examples, tasks and task network affects their predictions (i.e., the last column in Table 5.2).

(1) *Micro-level influence of training examples*: The distribution of the micro-level influences of the training examples w.r.t. the test examples is also similar to that in Fig. 5.2. We omitted these plots due to space limitation.

The top 4 influential training examples specific to each test example are shown in Fig. 5.7. For the purpose of validation, we want to compare how accurately $\mathcal{I}_{\mathbf{x}^s_{\text{test}}}(\mathbf{x}^t)$ can approximate $L(f_s(\mathbf{x}^s_{\text{test}}; \theta_{-\mathbf{x}^t})) - L(f_s(\mathbf{x}^s_{\text{test}}; \theta))$, i.e., the change of loss at the test sample after retraining without training example $\mathbf{x}^t$. We randomly pick a test example $\mathbf{x}^s_{\text{test}}$ from the first task, and show the predicted and actual changes for the top 100 influential training examples from each of the three tasks in Fig. 5.8. We can see that

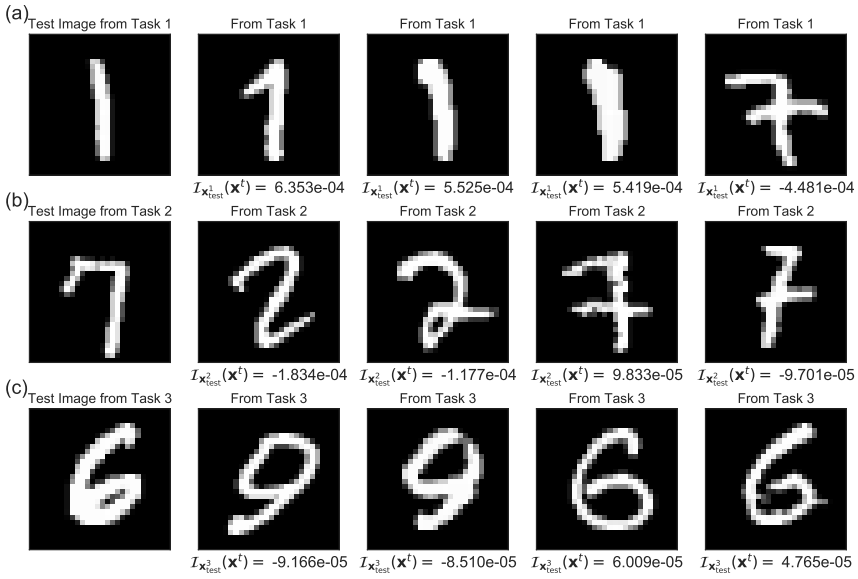


Figure 5.7: The top 4 influential training examples specific to each of the three test examples from each of the three tasks in (a), (b) and (c), respectively. The influence score of these training examples w.r.t. the test examples are shown under the example images.

the proposed method based on influence function can well approximate the change in losses, e.g., Pearson's R=0.99 and 0.98 in the first and second tasks, respectively.

Figure 5.8: Using influence function to approximate leave-one-out retraining test loss.

Fig. 5.9 plots the relationship between the micro-level influences of the training examples w.r.t. the test sample from task 1 and their $y^t\theta_t^T\mathbf{x}^t$ values. Note here that the micro-level influences could be either positive or negative, but their magnitude is very small when $y^t\theta_t^T\mathbf{x}^t$ has large positive value, which again supports our analysis

in Sec. 8. Fig. 5.10 shows the running time of Algorithm 8 with varying size of the

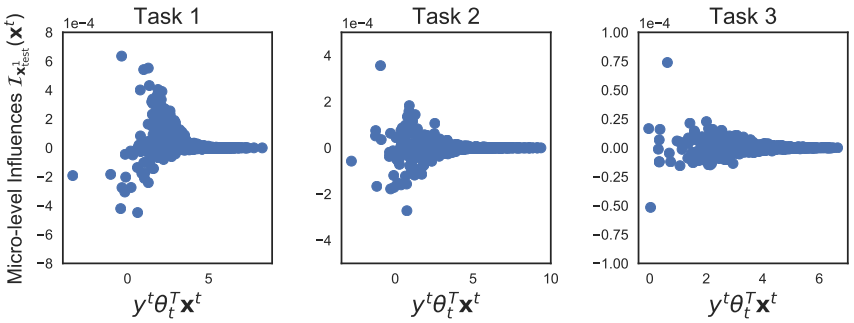


Figure 5.9: The micro-level influences of training samples vs. their $y\theta_t^T\mathbf{x}$

total number of training examples n. We can see that the proposed algorithm scales linearly, which is consistent with Theorem 5.

(2) *Micro-level influences of a learning task*: We compute the micro-level influences of each of the learning task w.r.t. the test examples and observe that generally the task that has the most influence on a particular test example is the one where the test example is from. One exception is that, for the test example from task 2, both task 1 and task 2 have similar positive influence possibly because task 1 that distinguishes between digit 1 and 7 can also help with the prediction of the test example of digit 7 in task 2.

(3) *Micro-level influences of task network*: We compute the micro-level influences of the task network specific to each test example. For the test example from the first task, $\mathbf{A}_{12}$ has a much larger negative impact than $\mathbf{A}_{23}$ possibly because connecting to task 2 does not help with the prediction of digit 1. For the test example of digit 7 from task 2, $\mathbf{A}_{12}$ has a larger positive impact than $\mathbf{A}_{23}$ since connecting to task 1 can help with this prediction.

Case Studies on Semantic Scholar Dataset We use the following test example from *data mining* area: **Constraint-Based Query Evaluation in Deductive**

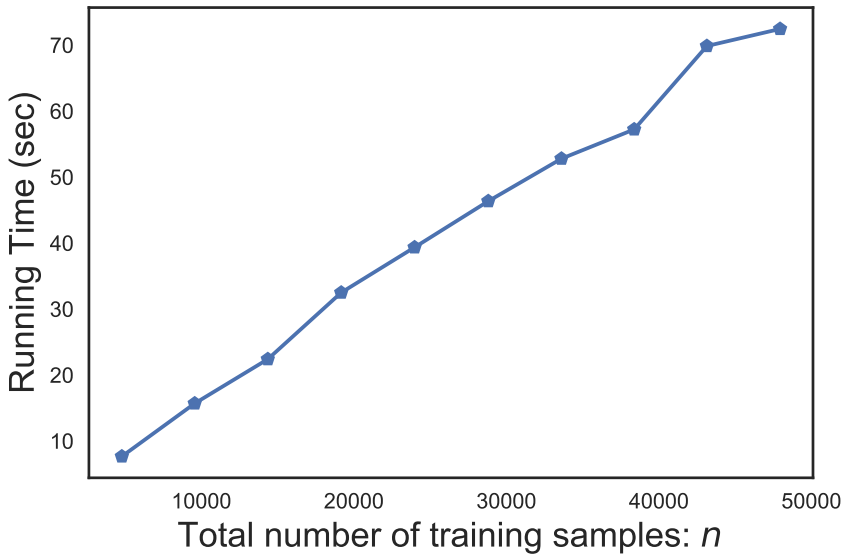


Figure 5.10: Running time vs. total number of training examples n for computing the micro-level influences of training examples.

Databases by *Prof. Jiawei Han* published in **TKDE** in 1994. We want to examine the most helpful training examples from each domain for this particular test example according to the micro-level influences.

The most helpful training examples from *data mining* are: **Structures, Semantics and Statistics** by *Alon Y. Halevy* published in VLDB in 2004 and **The architecture of complexity: the structure and the dynamics of networks from the web to the cell** by *Albert-László Barabási* published in KDD in 2005. The most helpful training examples from *computer vision* are: **SWIM: A Prototype Environment for Visual Media Retrieval** by *HongJiang Zhang* published in ACCV in 1995 and **Scene Reconstruction from Multiple Cameras** by *Richard Szeliski* published in ICIP in 2000. The most helpful training examples from *NLP* are: **Language Learning: Beyond Thunderdome** published in CoNLL in 2004 and **The segmentation problem in morphology learning** published in CoNLL in 1998 both by *Christopher D. Manning*. The most helpful training examples from *AI* are:

Robustness of Causal Claims by *Judea Pearl* published in UAI in 2004 and **Direct and Indirect Effects** by *Judea Pearl* published in UAI in 2001.

The helpful training examples from across domains are similar to this particular test example from *data mining* in the sense that they are all solo-authored papers by well-known researchers from respective domains. We want to emphasize that the influence function is not simply a Euclidean distance in the feature space as evident in Fig. 5.11, where we plot the micro-level influences $\mathcal{I}_{\mathbf{x}^1_{\text{test}}}(\mathbf{x}^t)$ vs. the Euclidean distance between the test example and the training examples from each task.

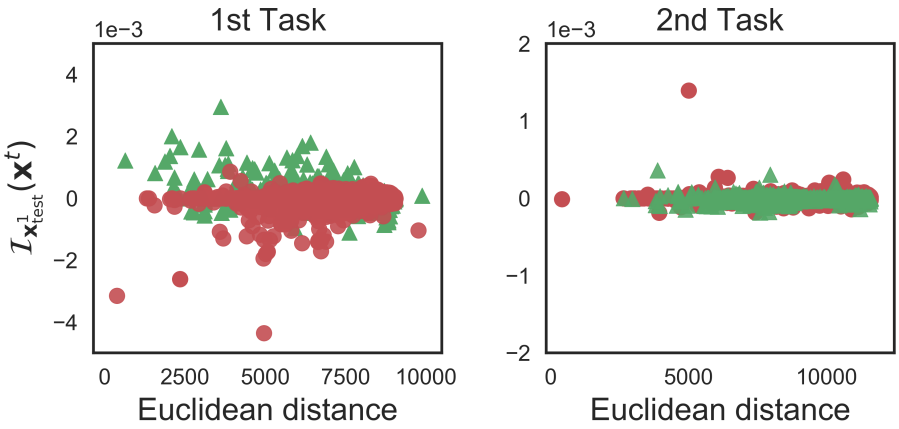


Figure 5.11: Euclidean distance vs. micro-level influence score on *Semantic Scholar*. We only show their relationship in *data mining* and *computer vision* as similar patterns are observed in other domains. Green triangles are training examples with the same label as the test example, and red dots are training examples with opposite label as the test example.

Case Studies on Sentiment Dataset We use the test example from *music* category and show the most influential training examples from other domains in Table 5.3. Comparing the test example and the helpful training examples from all the domains, it seems they are overall towards the positive sentiment despite some negative de-

scriptions about the products, e.g., the book "seems really dull" to average readers in the *book* domain. The harmful example from *music* is labeled as negative sentiment but the descriptions still sound largely positive.

	Review Text (positive sentiment is highlighted in bold font and negative sentiment is highlighted with underline). [...] is used to omit some sentences without altering the main meaning of the text.	**Label**
Test example from *music*	I was **instantly drawn into her music**. What I love about her songs is that they are so real. "You Give Me Love" is **so real and so strong**. "The Secret of Life" has taken some getting used to. It is not my absolute favorite on the CD. "Me" is the song that **means the most to me** since I have experienced trying to change for someone else in a relationship. [...] She has a big voice, and she **nails each song** on this CD. Give it a try.	+
Harmful example from *music*	**I liked this** when it first came out b/c I was 16.[...]This is **their best work** since the weirdness does get lame after a while. Favorite song is Mongoloid...totally strange **but rocking song**. "Uncontrollable Urge" **rocks** (in a weird way). Satisfaction is completely unique but its not a good song. [...] I saw them live (they were horrible) during their hey day. [...]**Anybody that gives this novelty group 5 stars is cheapening what true excellent music is.**	−
Helpful example from *book*	Here's a good litmus test to show **how good a book like "Breathing Lessons" is**–nothing extraordinary happens and yet I did not want to put the book down. [...]To the average reader this book probably would seem really dull. Heck, if someone told me the plot of this book I'd think it was really dull too, but **I didn't want to put it down**. [...] **It's hard for me to find any real faults with this book**, except for the lengthy flashback near the end that perhaps goes on too long. Some people may call this boring or dull, but I would call it purely exceptional. **I LOVED this book and highly recommend it**	+

Table 5.3: Case study on *Sentiment*.

5.2 Explaining Team Optimization in Networks

The emergence of network science has been significantly changing the landscape of team-based research in recent years. For example, if an existing team member becomes unavailable before the completion of the project, who shall we recommend to replace that individual's role so that the team would be least impacted due to the departure of this team member (*team member replacement*)? If the team leader perceives the need to expand the current team, who shall we bring into the existing team (*team expansion*)? Reversely, in case the team needs to be downsized (e.g., due to budget reduction), who should leave the team (*team shrinkage*)?

A cornerstone behind various team recommendation algorithms is random walk graph kernel [111]. By comparing and aggregating walks of the two input graphs, it naturally measures the graph similarity that captures both the topologies of the input graphs as well as the attributes associated with nodes and links. For instance, for team member replacement, by applying random walk graph kernel to the team networks before and after replacement, it encodes both the skill match, structure match as well as the interaction between the two during the replacement process [75]. Team member replacement further enables other team recommendation scenarios (e.g., team expansion, team shrinkage, etc.) [77]. Although being effective in answering questions like *who* is the best replacement, *what* is the best team expansion strategy, these existing methods lack intuitive ways to explain *why* the underlying algorithm gives the specific recommendation for a given team optimization scenario.

In this work, we present a prototype system (EXTRA), a paradigm shift from *what* to *why*, to explain the networked team recommendation results. On the algorithmic side, we propose an effective and efficient algorithm to explain random walk graph kernel given its central role in various team recommendation scenarios. The key idea

here is to identify the most influential network substructures and/or attributes whose removal or perturbation will impact the graph kernel/similarity the most. On the system side, our prototype system is able to explain various team recommendation scenarios (e.g., replacement, expansion, shrinkage) from a variety of different perspectives (e.g., edges, nodes and attributes). For example, given a candidate for team member replacement, we are able to tell what the key connections are between the candidate and the existing team members that might make him/her a potentially good replacement; for team expansion, we are able to identify the key skill sets that a candidate might bring to benefit the existing team the most.

5.2.1 Functionality Demonstration

In this section, we present the main functionalities of our prototype system (EXTRA) to explain three different team recommendation scenarios, including team member replacement, team expansion and team shrinkage. In our system, we model the underlying data as a large, attributed network, where nodes represent individuals, edges represent the relationship between different individuals, node attributes represent the skills of individuals, edge attributes represent the types of relationship (e.g., email communication, social friends, etc.) and a team is represented by an induced subgraph of its team members. The proposed EXTRA system provides explanations for different team recommendation scenarios through the lens of this underlying network from three different aspects, including edges, nodes and attributes. Table 5.4 summarizes the main functionalities of our system. The system allows users to explore team recommendation in the context of two common types of teams, including collaborative academic research teams and competitive sports teams. In addition, the system provides users with the option to manually assemble a team on-the-fly, and explore various team recommendation scenarios and the associated explanations.

	Team Replacement	**Team Expansion**	**Team Shrinkage**
Edges	important common collaborations shared by the candidate and the departure member	new collaborations that the new member might establish	the most important lacking collaborations the candidate should have
Nodes	most important existing team members that both the candidate and departure member collaborate with	key existing team members the new member will work with	key existing team members that the candidate should have collaborated with
Attributes	common and important skills shared by the candidate and the departure member	the unique skills the new team member brings that are critical to the team's new need	the most important skills that the candidate lacks

Table 5.4: Summary of system functionalities. Columns are different team recommendation scenarios and rows are different aspects for explanation.

Explaining Team Replacement A current team member might leave the team before the completion of the project for reasons like moving to another organization, being assigned to another project, etc. In this case, we need to find a good replacement for this member. In order to have the least impact on the entire team due to the member's departure so that the new team could continue to perform well, a team member replacement algorithm [75] often seeks to find a candidate who is most similar

to the departure member, in the sense of both skill match, structure match as well as the interaction between the two. Having this in mind, our prototype system identifies a few key (1) edges (the relationship between the candidate and other team members), (2) nodes (other team members) and (3) attributes (the skills of the candidate) that make the candidate and the departure member most similar. In this way, it could help the end-user (e.g., the team leader) understand why the underlying replacement algorithm thinks the given candidate is a potentially good replacement, based on which s/he can make a more informed decision.

Explaining Team Expansion If the team leader perceives the need to grow the current team based on the new requirement of the project, we need to find a best candidate to join the team. An effective team expansion recommendation algorithm often considers not only (1) if the new team member can bring critical skills to the team, but also (2) if the new member can collaborate efficiently with some of the existing team members with complementary skills [77]. Our prototype system provides the explanations for a recommended new team member from the following aspects, including (1) what are the unique new skills s/he brings to the team (i.e., attribute); and (2) what are the key collaborations the new team member might establish (i.e., edges) and with whom (i.e., nodes).

Explaining Team Shrinkage On the contrary to team expansion, the team leader might have to downsize the team (e.g., due to the budget reduction). In this scenario, a team shrinkage algorithm often chooses a least important team member to leave the team, so that the remaining team would maximally preserve the functionalities of the original team [77]. Our prototype system flags the *absent* skills and connections with existing team members that makes the candidate most insignificant to the current

team. In other words, we want to understand why the dismissal of this particular candidate would impose the least negative impact on the team.

5.2.2 Technical Details

In this subsection, we present key technical details behind the proposed EXTRA prototype system, including (1) the basics of random walk graph kernel, (2) how to use it for various team recommendation scenarios and (3) how to explain team recommendation.

Random Walk Graph Kernel Random walk graph kernel is a widely used computational model that provides a natural way to measure the similarity between two graphs [15]. Given two graphs $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ and $\mathbf{G}' = (\mathbf{V}', \mathbf{E}')$ (e.g., the two team networks before and after a replacement), let $\mathbf{W}$ and $\mathbf{W}'$ be the adjacency matrices of $\mathbf{G}$ and $\mathbf{G}'$, respectively. Their direct product graph $\mathbf{G}_\times = (\mathbf{V}_\times,\ \mathbf{E}_\times)$ is a graph with vertex set $\mathbf{V}_\times = \{(v, v') \mid v \in \mathbf{V}, v' \in \mathbf{V}'\}$ and edge set $\mathbf{E}_\times = \big\{ ((v_i,\ v'_r), (v_j,\ v'_s)) \mid (v_i, v_j) \in \mathbf{E}, (v'_r, v'_s) \in \mathbf{E}' \big\}$. We also represent the node attributes (e.g., skills of team members) as an $n \times l$ skill indicator matrix $\mathbf{L}$, where the i^{th} row vector of $\mathbf{L}$ describes the skills that the i^{th} team member has. Performing the simultaneous random walks on $\mathbf{G}$ and $\mathbf{G}'$ is equivalent to a random walk on the direct product graph. Let $\mathbf{p}$ and $\mathbf{p}'$ be the starting probabilities of the random walks on $\mathbf{G}$ and $\mathbf{G}'$, respectively. The stopping probabilities $\mathbf{q}$ and $\mathbf{q}'$ are defined similarly. Then, by imposing a decay factor c to longer walks and summing up all the common walks of different lengths, the random walk graph kernel for labelled graphs is computed as follows [111]:

$$k(\mathbf{G}, \mathbf{G}') = {\mathbf{q}_\times}^T (\mathbf{I} - c\mathbf{W}_\times)^{-1} \mathbf{L}_\times \mathbf{p}_\times \tag{5.4}$$

where $\mathbf{q}_\times = \mathbf{q} \otimes \mathbf{q}'$, $\mathbf{p}_\times = \mathbf{p} \otimes \mathbf{p}'$, $\otimes$ represents the Kronecker product operation, $\mathbf{W}_\times = \mathbf{L}_\times (\mathbf{W} \otimes \mathbf{W}')$ and $\mathbf{L}_\times = \sum_{k=1}^{l} \text{diag}(\mathbf{L}(:, k)) \otimes \text{diag}(\mathbf{L}'(:, k))$, where $\text{diag}(\mathbf{L}(:, k))$ is

a diagonal matrix where each entry indicates whether a team member has the k^{th} skill or not. $\mathbf{L}_{\times}(i,i) = 0$ if there is label inconsistency of two nodes from the two graphs (i.e. two team members have completely different skills), therefore the i^{th} row of $(\mathbf{W} \otimes \mathbf{W}')$ will be zeroed out. For plain graphs without node attributes, $\mathbf{L}_{\times}$ can be omitted from the above equation.

Graph Kernel for Team Recommendation It turns out random walk graph kernel is the core building block behind a variety of team recommendation scenarios. We summarize the key idea below. For details, please refer to [77].

In *team replacement*, the objective is to find a *similar* person m to replace the current team member r who is going to leave the team. A good replacement m should have a similar skill set as the current member r to continue the project and should have a similar collaboration structure with the existing team members so that the new team can still work together harmonically with little or no disruption.

Therefore, the similarity between the departure member and the candidate should be measured in the context of the team networks [75]. Mathematically, it aims to find a candidate m who satisfies: $m = \text{argmax}_{j, j \notin \mathbf{G}} k(\mathbf{G}, \mathbf{G}')$, where $\mathbf{G}$ is the current labelled team graph and $\mathbf{G}'$ is the team graph after replacement.

In *team shrinkage*, the objective is to find a current team member such that his/her dismissal will impact the current team as slightly as possible. In other words, we want the shrunk team as similar as possible to the original team, which leads to the following team shrinkage strategy, i.e., $m = \text{argmax}_{j, j \in \mathbf{G}} k(\mathbf{G}, \mathbf{G}_{-j})$, where $\mathbf{G}$ is the current labelled team graph and $\mathbf{G}_{-j}$ is the team graph without the current member j. In *team expansion*, the team leader often has an expectation for the configuration of the ideal new candidate, including the skills s/he has, and the way that s/he collaborates with the current team members. Thus, the team expansion strategy

adopts the following two steps. It first expands the current team by adding a *virtual* team member with the ideal configuration in terms of his/her skills as well as how to connect with the existing team members, and then it calls the team replacement algorithm to replace this virtual member with an actual person in the network, i.e., $n = \text{argmax}_{j, j \notin \mathbf{G}} k\left(\mathbf{G}', \mathbf{G}^{\mathbf{e}}\right)$, where $\mathbf{G}'$ is the newly expanded team with the actual candidate n and $\mathbf{G}^{\mathbf{e}}$ is the expanded team with the ideal, virtual team member.

Explaining Team Recommendation As mentioned in Section 5.2.1, a unique feature of our prototype system is to explain *why* a certain team recommendation algorithm gives a specific recommendation result, from various perspectives of the underlying network, including edges, nodes and attributes. Given the central role that random walk graph kernel plays in various team recommendation scenarios, we seek to understand the influence of various graph elements w.r.t. the corresponding graph kernels in Eq. (5.4), i.e., to what extent a given graph element (e.g., edge/node/attribute) would impact the graph kernel. An intuitive way to measure the influence of a given graph element would be first removing it from the team graph and then computing the change of the magnitude of the corresponding graph kernel. However, this method (referred to as 'direct computation') is computationally expensive, as we need to recompute the graph kernel for each possible graph element. To address this issue, we define the influence score of one specific element as the *rate* of the change in $k(\mathbf{G}, \mathbf{G}')$. For example, in order to calculate the influence score of the edge that connects the i^{th} and the j^{th} members in graph $\mathbf{G}$, let W_{ij} be the corresponding entry of the weighted adjacency matrix $\mathbf{W}$ of graph $\mathbf{G}$. Given random

walk graph kernel for labelled graph in Eq. (5.4) , we calculate its influence score as

$$\mathcal{I}(W_{ij}) = \frac{\partial k(\mathbf{G}, \mathbf{G}')}{\partial W_{ij}} = c\mathbf{q}_\times{}^T \mathbf{R}\mathbf{L}_\times \left(\frac{\partial \mathbf{W}}{\partial W_{ij}} \otimes \mathbf{W}' \right) \mathbf{R}\mathbf{L}_\times \mathbf{p}$$
$$= c\mathbf{q}_\times{}^T \mathbf{R}\mathbf{L}_\times \left((\mathbf{J}^{i,j} + \mathbf{J}^{j,i}) \otimes \mathbf{W}' \right) \mathbf{R}\mathbf{L}_\times \mathbf{p} \tag{5.5}$$

where $\mathbf{R} = (\mathbf{I} - c\ \mathbf{W}_\times)^{-1}$ and its computation can be accelerated using power method and $\mathbf{J}^{i,j}$ is a single-entry matrix with one at its (i,j)-th entry and zeros everywhere else. Edges (i.e., collaborations between team members) with the highest influence scores can be used to explain the corresponding team recommendation results. In team replacement, an ideal candidate should have these key collaborations as the departure member, indicating their similarity in terms of how they collaborate with existing members. In the scenario of team expansion, these are key collaborations that the team leader expects the new member to establish. In the team shrinkage scenario, these are important collaborations that the candidate might lack which in turns makes his/her dismissal of little negative impact on the team.

The node influence of the i^{th} member is defined as the aggregation of the influence of all the edges incident to this node, i.e., $\mathcal{I}(i) = \sum_{j|(i,j)\in\mathbf{E}} \mathcal{I}(W_{ij})$. Existing team members with the highest node influence scores are expected to be key members in the team. In the team replacement scenario, a good candidate should also collaborate with these key members as the departure member. In team expansion, these are the key team members the new member will work with. In team shrinkage, these are the key members that the candidate should have collaborated with.

Likewise, to compute the influence of a team member's attributes (e.g., skills) on the graph kernel, we take the derivative of the graph kernel w.r.t. the member's skill. Denote the k^{th} skill of the i^{th} team member in graph $\mathbf{G}$ as $L_k(i)$, the attribute influence can be calculated as

$$\mathcal{I}(L_k(i)) = \frac{\partial k(\mathbf{G}, \mathbf{G}')}{\partial L_k(i)} = \mathbf{q_x}{}^T \mathbf{R} \left(\frac{\partial \mathbf{L}_\times}{\partial L_k(i)} \right) (\mathbf{I} + c(\mathbf{W} \otimes \mathbf{W}')\mathbf{R}\mathbf{L}_\times)\, \mathbf{p}_\times$$

where $\frac{\partial \mathbf{L}_{\times}}{\partial L_k(i)} = \text{diag}(\mathbf{e}_i) \otimes \text{diag}(\mathbf{L}'(:,k))$ and $\mathbf{e}_i$ is a n by 1 vector with one at the i^{th} entry and zeros everywhere else. Skills with the highest attribute influence scores can be used to explain the corresponding team recommendation results. These are key skills that (1) the candidate and the departure member share in team replacement, (2) the new team member might bring in team expansion, and (3) the dismissal member might lack in team shrinkage.

5.2.3 *System Demonstration*

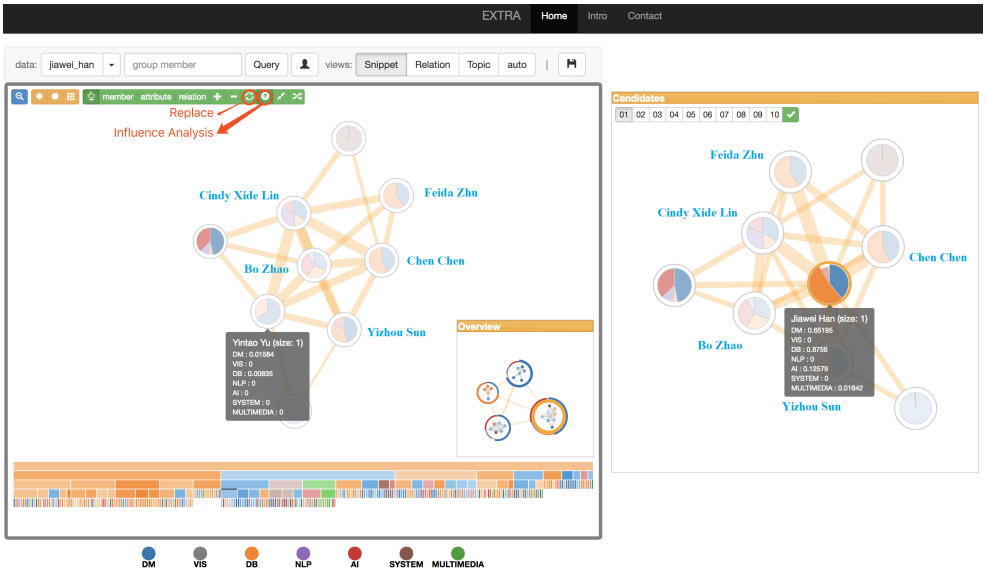


Figure 5.12: An illustrative example of influence analysis.

Figure 5.12 presents the user interface of EXTRA and an example of visualization of influence analysis for team member replacement on a co-authorship network[3], which suggests that Dr. *Jiawei Han* is the best replacement. The influence scores of all edges for both graphs are calculated by our proposed algorithm in Section 5.2.2 and the width of edge is proportional to the influence score. The top-*4* most influential

[3]A demo video of the system is available at https://youtu.be/D4gcI-QHtps. System website: http://144.202.123.224/system.html .

edges are those that connect Dr. *Yu* with the 4 existing members, *Lin, Zhao, Chen and Sun*, all of whom are considered as the key members and they also have strong collaborations with Dr. *Han*. After replacement, the top-*5* most influential edges with Dr. *Han* overlap with those of Dr. *Yu* in the same ranking order. The only exception is that no edge exists between Dr. *Yu* and Dr. *Zhu* because there is no prior collaboration between them. In addition, the system can provide explanations from the attribute perspective. The key skills (represented in pie chart) shared by Dr. *Han* and Dr. *Yu* are *databases* and *data mining* in this case, which makes the team replacement recommendation more understandable.

Chapter 6

CONCLUSION AND FUTURE WORK

In this chapter, we summarize our key research results and discuss promising future research directions.

6.1 Conclusion

In this dissertation, we establish effective algorithms and tools for the performance prediction and optimization of teams long with explanations, in the context of composite. We take a multi-disciplinary approach, consisting of supervised learning, visualization and optimization, to tackle three complementary research tasks, namely, *team performance prediction*, *team performance optimization*, and *team performance explanation*.

Team performance prediction For the prediction of long-term impact of scientific work given its citation history in the first few years, we propose iBall – a family of algorithms. The proposed algorithms collectively address a number of key algorithmic challenges in impact prediction (i.e., feature design, non-linearity, domain heterogeneity and dynamics). It is flexible and general in the sense that it can be generalized to both regression and classification models; and in both linear and non-linear formulations; it is scalable and adaptive to new training data. For forecasting the impact pathway of scholarly entities, we propose an effective method (*iPath*). The proposed *iPath* can collectively model two important aspects of the impact pathway prediction problem, namely, *prediction consistency* and *parameter smoothness*. It is flexible for handling both linear and non-linear models and empirical evaluations demonstrate

its effectiveness for forecasting the pathway to impact. To simultaneously and mutually predict the parts and whole outcomes, we propose a joint predictive model `NEPAL` . First, *model generality*, the proposed model is able to (i) admit a variety of linear as well as non-linear relationship between the parts and whole outcome and (ii) characterize part-part interdependency. Second, *algorithm efficacy*, we propose an effective and efficient block coordinate descent optimization algorithm that converges to the coordinate-wise optimum with a linear complexity in both time and space. The empirical evaluations on real-world datasets demonstrate that (i) by modeling the non-linear part-whole relationship and part-part interdependency, the proposed method leads to consistent prediction performance improvement, and (ii) the proposed algorithm scales linearly w.r.t. the size of the training data.

Team performance optimization We start with 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. Beyond Team Member Replacement, we have also considered a number of other team enhancement scenarios, namely, Team Refinement (to edit an existing member's skill and communication structure), Team Expansion (to hire a team member with desired skills and connections), and Team Shrinkage

(to remove an existing team member). All these enhancement scenarios can be solved using the same algorithm developed for TEAM MEMBER REPLACEMENT. The experimental evaluations show the effectiveness of the proposed algorithm. For real-time team optimization, i.e., to plan the team optimization/re-staffing actions at each time step so as to maximize the expected cumulative performance, we propose to leverage Reinforcement Learning to automatically learn the optimal staffing strategies.

Team performance explanation To demystify networked prediction (i.e., iBall model), we propose a multi-aspect, multi-level approach `PAROLE` by understanding how the learning process is diffused at different levels from different aspects. The key idea is to efficiently quantify the influence on different levels (i.e., macro/system, meso/task, micro/example) of the learning system due to the perturbation of the various aspects (i.e., example, task, network). The proposed approach offers two distinctive advantages: (1) *multi-aspect, multi-level*: we are able to provide a comprehensive explanation to the workings of the networked predictions; (2) *efficiency*: it has a linear complexity by efficiently evaluating the influences of changes to the networked prediction without retraining the whole learning system. The empirical evaluations on real-world datasets demonstrate the efficacy of the proposed `PAROLE` algorithm. As the first step towards explaining team recommendation through the lens of the underlying network where teams embed, we present a prototype system (EXTRA). The key algorithmic idea is to identify the most influential network substructures and/or attributes that account for the team recommendation results. The system is able to provide intuitive explanations from different perspectives (i.e., edges, nodes, attributes) for various team recommendation scenarios, including team replacement, team expansion and team shrinkage.

6.2 Future Work

As an emerging field, the network science of teams is still in its early stage and remains an active area of exploration. Future directions include modeling the hierarchical structure within organizations by extending our `PAROLE` model and modeling the heterogeneous goals among the team members. In our team optimization work, we have the implicit assumption that the original team is performing well and maintaining the similarity with the original team can promise a similar high performance. We want to point out that when the assumption does not hold, we can leverage the actual or predicted future performance as feedback to guide the team optimization process, as in the real time team optimization scenario. Other promising future directions are presented below:

- **Example-based Team Formation.** Given historical high-performing as well as struggling teams, we are interested in forming good teams from the patterns learned from these teams. We conjecture that a good team can be formed by maximizing a certain similarity function between the team we want to assemble and those previously successful teams that have worked on similar tasks; in the meanwhile, maximizing the distance from those struggling teams.

- **Multiple Persons Optimization.** In Chapter 4, we only consider single person in the various team optimization scenarios, including team replacement, team expansion, team shrinkage, etc. We are interested in extending these to multiple persons, for example, how to hire another three team members with various expertise into the team. The challenge now lies in the exponentially large solution space. A simple heuristic would be to apply our proposed algorithm for each person one at a time. However, this treatment would be sub-optimal and ignores the interactions among the multiple persons.

- **Multiple Teams Optimization.** We often need to optimize multiple teams within an organization and all of these teams are constrained by the same pool of human resources. For example, if we expand one team by hiring a new member from another team within the organization, it will inevitably impact the performance of the second team. We are interested in designing new algorithms for collectively optimizing multiple teams.

REFERENCES

[1] Adler, P., C. Falk, S. A. Friedler, G. Rybeck, C. Scheidegger, B. Smith and S. Venkatasubramanian, "Auditing black-box models for indirect influence", in "ICDM", pp. 1–10 (2016).

[2] Anagnostopoulos, A., L. Becchetti, C. Castillo, A. Gionis and S. Leonardi, "Online team formation in social networks", in "WWW", pp. 839–848 (2012).

[3] Ando, R. K. and T. Zhang, "Learning on graph with laplacian regularization", in "NIPS", pp. 25–32 (2006).

[4] Anirudh, R., J. J. Thiagarajan, R. Sridhar and T. Bremer, "Influential sample selection: A graph signal processing approach", arXiv preprint arXiv:1711.05407 (2017).

[5] Aronszajn, N., "Theory of reproducing kernels", Transactions of the American mathematical society pp. 337–404 (1950).

[6] Balog, K., L. Azzopardi and M. de Rijke, "Formal models for expert finding in enterprise corpora", in "SIGIR", pp. 43–50 (2006).

[7] Beck, A. and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems", SIAM journal on imaging sciences **2**, 1, 183–202 (2009).

[8] Bellman, R., *Dynamic programming* (Courier Corporation, 2013).

[9] Blake, A., P. Kohli and C. Rother, *Markov Random Fields for Vision and Image Processing* (The MIT Press, 2011).

[10] Blitzer, J., M. Dredze, F. Pereira *et al.*, "Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification", in "ACL", vol. 7, pp. 440–447 (2007).

[11] Bogdanov, P., 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)", pp. 525–540 (2013).

[12] Bollen, J., H. Van de Sompel, A. Hagberg and R. Chute, "A principal component analysis of 39 scientific impact measures", PloS one **4**, 6, e6022 (2009).

[13] Borgwardt, K. M. and H.-P. Kriegel, "Shortest-path kernels on graphs", in "ICDM", pp. 74–81 (2005).

[14] Borgwardt, K. M., 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).

[15] Borgwardt, K. M., C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola and H.-P. Kriegel, "Protein function prediction via graph kernels", in "ISMB (Supplement of Bioinformatics)", pp. 47–56 (2005).

[16] Bouguessa, M., B. Dumoulin and S. Wang, "Identifying authoritative actors in question-answering forums: the case of yahoo! answers", in "KDD", pp. 866–874 (2008).

[17] Boutell, M. R., J. Luo, X. Shen and C. M. Brown, "Learning multi-label scene classification", Pattern Recognition **37**, 9, 1757 – 1771 (2004).

[18] Cai, Y., H. Tong, W. Fan and P. Ji, "Fast mining of a network of coevolving time series", in "SDM", pp. 298–306 (2015).

[19] Cai, Y., H. Tong, W. Fan, P. Ji and Q. He, "Facets: Fast comprehensive mining of coevolving high-order time series", in "KDD", pp. 79–88 (2015).

[20] Cao, N., Y.-R. Lin, L. Li and H. Tong, "g-Miner: Interactive visual group mining on multivariate graphs", in "CHI", (2015).

[21] Caruana, R., "Multitask learning", Machine learning **28**, 1, 41–75 (1997).

[22] Caruana, R., Y. Lou, J. Gehrke, P. Koch, M. Sturm and N. Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission", in "KDD", pp. 1721–1730 (ACM, 2015).

[23] Castillo, C., D. Donato and A. Gionis, "Estimating number of citations using author reputation", in "String processing and information retrieval", pp. 107–117 (Springer, 2007).

[24] Cataldo, M. and K. Ehrlich, "The impact of communication structure on new product development outcomes", in "CHI", pp. 3081–3090 (2012).

[25] Chang, B., H. Zhu, Y. Ge, E. Chen, H. Xiong and C. Tan, "Predicting the popularity of online serials with autoregressive models", in "CIKM", pp. 1339–1348 (2014).

[26] Chang, K. and K. Ehrlich, "Out of sight but not out of mind?: Informal networks, communication and media use in global software teams", in "CASCON", pp. 86–97 (2007).

[27] Chen, C., H. Tong, L. Xie, L. Ying and Q. He, "Fascinate: Fast cross-layer dependency inference on multi-layered networks", in "KDD", (2016).

[28] Chen, J., J. Liu and J. Ye, "Learning incoherent sparse and low-rank patterns from multiple tasks", in "KDD", pp. 1179–1188 (2010).

[29] Chen, J., J. Zhou and J. Ye, "Integrating low-rank and group-sparse structures for robust multi-task learning", in "KDD", pp. 42–50 (ACM, 2011).

[30] Clare, A. and R. D. King, "Knowledge discovery in multi-label phenotype data", in "Principles of data mining and knowledge discovery", pp. 42–53 (Springer, 2001).

[31] Clauset, A., D. B. Larremore and R. Sinatra, "Data-driven predictions in the science of science", Science **355**, 6324, 477–480 (2017).

[32] Cook, R. D. and S. Weisberg, *Residuals and influence in regression* (New York: Chapman and Hall, 1982).

[33] Cummings, J. N. and S. B. Kiesler, "Who collaborates successfully?: prior experience reduces collaboration barriers in distributed interdisciplinary research", in "CSCW", pp. 437–446 (2008).

[34] Davletov, F., A. S. Aydin and A. Cakmak, "High impact academic paper prediction using temporal and topological features", in "CIKM", pp. 491–498 (2014).

[35] DeLong, C. and J. Srivastava, "Teamskill evolved: Mixed classification schemes for team-based multi-player games", in "PAKDD (1)", pp. 26–37 (2012).

[36] DeLong, C., L. G. Terveen and J. Srivastava, "Teamskill and the nba: applying lessons from virtual worlds to the real-world", in "ASONAM", pp. 156–161 (2013).

[37] Deng, H., I. King and M. R. Lyu, "Formal models for expert finding on dblp bibliography data", in "ICDM", pp. 163–172 (2008).

[38] Dong, Y., R. A. Johnson and N. V. Chawla, "Will this paper increase your *h*-index? scientific impact prediction", in "WSDM", (2015).

[39] Dorairaj, S., J. Noble and P. Malik, "Understanding team dynamics in distributed agile software development", in "International Conference on Agile Software Development", pp. 47–61 (Springer, 2012).

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

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

[42] Elisseeff, A. and J. Weston, "A kernel method for multi-labelled classification", in "NIPS", pp. 681–687 (2001).

[43] Evgeniou, T. and M. Pontil, "Regularized multi–task learning", in "ACM SIGKDD", pp. 109–117 (2004).

[44] Feragen, A., N. Kasenburg, J. Petersen, M. de Bruijne and K. M. Borgwardt, "Scalable kernels for graphs with continuous attributes", in "NIPS", pp. 216–224 (2013).

[45] Fürnkranz, J., E. Hüllermeier, E. L. Mencía and K. Brinker, "Multilabel classification via calibrated label ranking", Machine learning **73**, 2, 133–153 (2008).

[46] Gärtner, T., P. A. Flach and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives", in "COLT", pp. 129–143 (2003).

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

[48] Golshan, B. and E. Terzi, "Minimizing tension in teams", in "Proceedings of the 2017 ACM on Conference on Information and Knowledge Management", CIKM '17, pp. 1707–1715 (ACM, New York, NY, USA, 2017).

[49] Golub, G. H. and C. F. Van Loan, *Matrix Computations (3rd Ed.)* (Johns Hopkins University Press, Baltimore, MD, USA, 1996).

[50] Haas, M. and M. Mortensen, "The secrets of great teamwork.", Harvard business review **94**, 6, 70–6 (2016).

[51] Hackman, J. R. and N. Katz, *Group behavior and performance*, pp. 1208–1251 (Wiley, New York, 2010).

[52] Hashemi, S. H., M. Neshati and H. Beigy, "Expertise retrieval in bibliographic network: a topic dominance learning approach", in "CIKM", pp. 1117–1126 (2013).

[53] Hido, S. and H. Kashima, "A linear-time graph kernel", in "ICDM", pp. 179–188 (2009).

[54] Hinds, P. J., K. M. Carley, D. Krackhardt and D. Wholey, "Choosing work group members: Balancing similarity, competence, and familiarity", in "Organizational Behavior and Human Decision Processes", pp. 226–251 (2000).

[55] Horváth, T., T. Gärtner and S. Wrobel, "Cyclic pattern kernels for predictive graph mining", in "KDD", pp. 158–167 (2004).

[56] Huang, S.-J. and Z.-H. Zhou, "Multi-label learning by exploiting label correlations locally", in "AAAI", (2012).

[57] Jalali, A., S. Sanghavi, C. Ruan and P. K. Ravikumar, "A dirty model for multi-task learning", in "NIPS", pp. 964–972 (2010).

[58] Jasny, B. R. and R. Stone, "Prediction and its limits", Science **355**, 6324, 468–469 (2017).

[59] Ji, S., L. Tang, S. Yu and J. Ye, "A shared-subspace learning framework for multi-label classification", TKDD **4**, 2, 8 (2010).

[60] Kang, U., H. Tong and J. Sun, "Fast random walk graph kernel.", in "SDM", pp. 828–838 (2012).

[61] Karimzadehgan, M. and C. Zhai, "Constrained multi-aspect expertise matching for committee review assignment", in "CIKM", pp. 1697–1700 (2009).

[62] Karypis, G. and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs", SIAM J. Sci. Comput. **20**, 1 (1998).

[63] Kato, T., H. Kashima, M. Sugiyama and K. Asai, "Multi-task learning via conic programming", in "Advances in Neural Information Processing Systems", pp. 737–744 (2008).

[64] Koh, P. W. and P. Liang, "Understanding black-box predictions via influence functions", in "ICML", pp. 1885–1894 (2017).

[65] Kondor, R. I., N. Shervashidze and K. M. Borgwardt, "The graphlet spectrum", in "ICML", p. 67 (2009).

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

[67] Lappas, T., K. Liu and E. Terzi, "Finding a team of experts in social networks", in "KDD", pp. 467–476 (2009).

[68] Lawrence, K. D. and J. L. Arthur, *Robust regression: analysis and applications* (Marcel Dekker Inc, New York, 1990).

[69] LeCun, Y., L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE **86**, 11 (1998).

[70] Lee, D. D. and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization", Nature **401**, 6755, 788–791 (1999).

[71] Li, H., X. Ma, F. Wang, J. Liu and K. Xu, "On popularity prediction of videos shared in online social networks", in "CIKM", pp. 169–178 (2013).

[72] Li, J.-Z., J. Tang, J. Zhang, Q. Luo, Y. Liu and M. Hong, "Eos: expertise oriented search using social networks", in "WWW", pp. 1271–1272 (2007).

[73] Li, L. and H. Tong, "The child is father of the man: Foresee the success at the early stage", in "KDD", pp. 655–664 (2015).

[74] Li, L. and H. Tong, "The child is father of the man: Foresee the success at the early stage", CoRR **abs/1504.00948**, URL `http://arxiv.org/abs/1504.00948` (2015).

[75] Li, L., H. Tong, N. Cao, K. Ehrlich, Y. Lin and N. Buchler, "Replacing the irreplaceable: Fast algorithms for team member recommendation", in "WWW", pp. 636–646 (ACM, 2015).

[76] Li, L., H. Tong, N. Cao, K. Ehrlich, Y. Lin and N. Buchler, "TEAMOPT: interactive team optimization in big networks", in "CIKM", pp. 2485–2487 (2016).

[77] Li, L., H. Tong, N. Cao, K. Ehrlich, Y. Lin and N. Buchler, "Enhancing team composition in professional networks: Problem definitions and fast solutions", IEEE Trans. Knowl. Data Eng. **29**, 3, 613–626 (2017).

[78] Li, L., H. Tong and H. Liu, "Towards explainable networked prediction", in "CIKM", (2018).

[79] Li, L., H. Tong, J. Tang and W. Fan, "*iPath:* forecasting the pathway to impact", in "SDM", pp. 468–476 (SIAM, 2016).

[80] Li, L., H. Tong, Y. Xiao and W. Fan, "*Cheetah:* fast graph kernel tracking on dynamic graphs", in "SDM", pp. 280–288 (2015).

[81] Liu, L. and E. Zhao, "Team performance and individual performance: Example from engineering consultancy company in china", in "2011 International Conference on Management and Service Science", pp. 1–4 (2011).

[82] Lou, Y., J. Bien, R. Caruana and J. Gehrke, "Sparse partially linear additive models", Journal of Computational and Graphical Statistics **25**, 4, 1126–1140 (2016).

[83] Lou, Y., R. Caruana and J. Gehrke, "Intelligible models for classification and regression", in "Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 150–158 (ACM, 2012).

[84] Lou, Y., R. Caruana, J. Gehrke and G. Hooker, "Accurate intelligible models with pairwise interactions", in "Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 623–631 (ACM, 2013).

[85] Ma, J., Z. Zhao, X. Yi, J. Chen, L. Hong and E. H. Chi, "Modeling task relationships in multi-task learning with multi-gate mixture-of-experts", in "KDD", pp. 1930–1939 (2018).

[86] Mahé, P., 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).

[87] Mair, P., K. Hornik and J. de Leeuw, "Isotone optimization in r: pool-adjacent-violators algorithm (pava) and active set methods", Journal of statistical software **32**, 5, 1–24 (2009).

[88] Mimno, D. M. and A. McCallum, "Expertise modeling for matching papers with reviewers", in "KDD", pp. 500–509 (2007).

[89] Morgan Jr, B. B., E. Salas and A. S. Glickman, "An analysis of team evolution and maturation", The Journal of General Psychology **120**, 3, 277–291 (1993).

[90] Muller, M., 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", pp. 2815–2824 (2012).

[91] Ni, J., H. Tong, W. Fan and X. Zhang, "Inside the atoms: ranking on a network of networks", in "KDD", pp. 1356–1365 (2014).

[92] Ni, J., H. Tong, W. Fan and X. Zhang, "Flexible and robust multi-network clustering", in "KDD", pp. 835–844 (2015).

[93] Pearlmutter, B. A., "Fast exact multiplication by the hessian", Neural computation **6**, 1, 147–160 (1994).

[94] Pei, J., 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).

[95] Perozzi, B., R. Al-Rfou and S. Skiena, "Deepwalk: Online learning of social representations", in "Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 701–710 (ACM, 2014).

[96] Rangapuram, S. S., T. Bühler and M. Hein, "Towards realistic team formation in social networks based on densest subgraphs", in "WWW", pp. 1077–1088 (2013).

[97] Read, J., B. Pfahringer, G. Holmes and E. Frank, "Classifier chains for multi-label classification", in "Machine Learning and Knowledge Discovery in Databases", vol. 5782, pp. 254–269 (Springer Berlin Heidelberg, 2009).

[98] Ribeiro, M. T., S. Singh and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier", in "KDD", pp. 1135–1144 (ACM, 2016).

[99] Saunders, C., A. Gammerman and V. Vovk, "Ridge regression learning algorithm in dual variables", in "ICML", ICML '98, pp. 515–521 (1998).

[100] Schmeidler, D., "Subjective probability and expected utility without additivity", Econometrica: Journal of the Econometric Society pp. 571–587 (1989).

[101] Shervashidze, N. and K. M. Borgwardt, "Fast subtree kernels on graphs", NIPS (2009).

[102] Shervashidze, N., 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).

[103] Shervashidze, N., 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).

[104] Simon, H. A., *The Sciences of the Artificial (3rd Ed.)* (MIT Press, Cambridge, MA, USA, 1996).

[105] Tang, J., J. Zhang, L. Yao, J. Li, L. Zhang and Z. Su, "Arnetminer: extraction and mining of academic social networks", in "KDD", pp. 990–998 (2008).

[106] Tibshirani, R., "Regression shrinkage and selection via the lasso", Journal of the Royal Statistical Society. Series B (Methodological) pp. 267–288 (1996).

[107] Tseng, P., "Convergence of a block coordinate descent method for nondifferentiable minimization", Journal of optimization theory and applications **109**, 3, 475–494 (2001).

[108] Tsivtsivadze, E., J. Urban, H. Geuvers and T. Heskes, "Semantic graph kernels for automated reasoning", in "SDM", pp. 795–803 (2011).

[109] Tuckman, B. W., "Developmental sequence in small groups.", Psychological bulletin **63**, 6, 384 (1965).

[110] Uzzi, B., S. Mukherjee, M. Stringer and B. Jones, "Atypical combinations and scientific impact", Science **342**, 6157, 468–472 (2013).

[111] Vishwanathan, S. V. N., K. M. Borgwardt and N. N. Schraudolph, "Fast computation of graph kernels", in "NIPS", pp. 1449–1456 (2006).

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

[113] Wang, C. and D. M. Blei, "Collaborative topic modeling for recommending scientific articles", in "KDD", pp. 448–456 (2011).

[114] Wang, D., C. Song and A.-L. Barabási, "Quantifying long-term scientific impact", Science **342**, 6154, 127–132 (2013).

[115] Warner, S., M. T. Bowers and M. A. Dixon, "Team dynamics: A social network perspective", Journal of Sport Management **26**, 1, 53–66 (2012).

[116] Wu, S., J. Sun and J. Tang, "Patent partner recommendation in enterprise social networks", in "WSDM", pp. 43–52 (2013).

[117] Wuchty, S., B. F. Jones and B. Uzzi, "The increasing dominance of teams in production of knowledge", Science **316**, 5827, 1036–1039 (2007).

[118] Yan, R., C. Huang, J. Tang, Y. Zhang and X. Li, "To better stand on the shoulder of giants", in "JDCL", pp. 51–60 (2012).

[119] Yan, R., J. Tang, X. Liu, D. Shan and X. Li, "Citation count prediction: learning to estimate future citations for literature", in "CIKM", pp. 1247–1252 (2011).

[120] Yao, Y., H. Tong, T. Xie, L. Akoglu, F. Xu and J. Lu, "Joint voting prediction for questions and answers in cqa", in "ASONAM", pp. 340–343 (IEEE, 2014).

[121] Yao, Y., H. Tong, F. Xu and J. Lu, "Predicting long-term impact of cqa posts: a comprehensive viewpoint", in "KDD", pp. 1496–1505 (ACM, 2014).

[122] Yontay, P. and R. Pan, "A computational bayesian approach to dependency assessment in system reliability", Reliability Engineering & System Safety **152**, 104–114 (2016).

[123] Yu, K., V. Tresp and A. Schwaighofer, "Learning gaussian processes from multiple tasks", in "ICML", (2005).

[124] Yu, X., Q. Gu, M. Zhou and J. Han, "Citation prediction in heterogeneous bibliographic networks", in "SDM", pp. 1119–1130 (2012).

[125] Zadeh, R., A. D. Balakrishnan, S. B. Kiesler and J. N. Cummings, "What's in a move?: normal disruption and a design challenge", in "CHI", pp. 2897–2906 (2011).

[126] Zeng, X. and M. A. T. Figueiredo, "The ordered weighted l_1 norm: Atomic formulation, dual norm, and projections", CoRR **abs/1409.4271**, URL `http://arxiv.org/abs/1409.4271` (2014).

[127] Zhang, J., M. S. Ackerman and L. A. Adamic, "Expertise networks in online communities: structure and algorithms", in "WWW", pp. 221–230 (2007).

[128] Zhang, M.-L. and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning", Pattern recognition **40**, 7, 2038–2048 (2007).

[129] Zhang, M.-L. and Z.-H. Zhou, "A review on multi-label learning algorithms", TKDE **26**, 8, 1819–1837 (2014).

[130] Zhang, Y. and D.-Y. Yeung, "Multilabel relationship learning", TKDD **7**, 2, 7 (2013).

[131] Zhou, G., S. Lai, K. Liu and J. Zhao, "Topic-sensitive probabilistic model for expert finding in question answer communities", in "CIKM", pp. 1662–1666 (2012).

[132] Zhou, Q., L. Li, N. Cao, N. Buchler and H. Tong, "Extra: explaining team recommendation in networks", in "RecSys", pp. 492–493 (2018).

BIOGRAPHICAL SKETCH

Liangyue Li is currently a Ph.D. candidate at the School of Computing, Informatics and Decision Systems Engineering at Arizona State University. He received the B.Eng. degree in Computer Science from Tongji University in 2011. His current research interests include large-scale data mining and machine learning, especially for large graph data with application to social network analysis. He has published over 10 referred articles in top conferences and journals. He has served as a program committee member in top data mining and artificial intelligence venues (e.g., SIGKDD, WWW, SDM, AAAI, CIKM, etc). He has given tutorials at KDD 2018, WSDM 2018, and a keynote talk at CIKM 2016 Workshop on Big Network Analytics (BigNet 2016). For more details, please refer to his homepage at `http://www.public.asu.edu/~liangyue/`.