Towards Real Time Team Optimization

Qinghai Zhou
University of Illinois at Urbana-Champaign
qinghai2@illinois.edu

Liangyue Li
Amazon
liliangy@amazon.com

Hanghang Tong
University of Illinois at Urbana-Champaign
htong@illinois.edu

***Abstract*—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). Consequently, the performance of the team might be changing due to such team dynamics. A natural question is how to plan the (re-)staffing actions (e.g., recruiting a new team member) at each time step so as to maximize the expected cumulative performance of the team. In this paper, we address the problem of real-time team optimization by intelligently selecting the best candidates towards increasing the similarity between the current team and the high-performance teams according to the team configuration at each time-step. The key idea is to formulate it as a Markov Decision process (MDP) problem and leverage recent advances in reinforcement learning to optimize the team dynamically. The proposed method bears two main advantages, including (1) *dynamics*, being able to model the dynamics of the team to optimize the initial team towards the direction of a high-performance team via performance feedback; (2) *efficacy*, being able to handle the large state/action space via deep reinforcement learning based value estimation. We demonstrate the effectiveness of the proposed method through extensive empirical evaluations.**

I. Introduction

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). It is hypothesized that 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 [1]. Although teams might take different paths towards maturity, research suggests that the effective cooperation and coordinations among team members generally bring the team from initial ineptness to the final levels of skilled performance [2]. In the context of sports teams and software development teams, research efforts have been on the relationship between team dynamics and team performances [3], [4].

Due to the team dynamics, the performance of the team is very likely to be changing over time. If a team struggles to achieve satisfactory performance or external demands for adjustments are required, changes to the team are necessary. A natural question is how to plan the team optimization/re-staffing actions (e.g., recruiting 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 [5] and team enhancement [6]) 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 a 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 be sub-optimal in terms of the long-term reward.

In this paper, 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 the recent advances in deep reinforcement learning 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 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. 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. We summarize our main contributions as follows,

978-1-7281-0858-2/19/$31.00

- **Problem Formulation** We formally define and formulate the real-time team optimization problem. Given the template teams with high performance, the key idea is to continuously maximize the long term performance of the optimized teams through a sequence of actions, e.g., adding/removing team members, establishing new collaborations.
- **Algorithm and Analysis** We propose a deep reinforcement learning based framework, which can continuously learn and update its team optimization strategy by incorporating both skill similarity and structural consistency.
- **Empirical Evaluations** We perform extensive experimental evaluations on real-world datasets to test the efficacy of our proposed framework in the task of real-time team optimization. The evaluations demonstrate that the proposed reinforcement learning framework can achieve significant performance in optimizing the initial team towards a high-performing one.

The rest of this paper is organized as follows. In Section II, we review the Deep Reinforcement Learning (DRL) as well as the Graph Neural Network family models as the preliminaries, after we formally define the problem of real-time team optimization. Section III introduces our proposed framework. We present the evaluation results in Section IV, and review related work in Section V. We finally conclude this paper in Section VI.

II. Problem Definitions and Preliminaries

In this section, we introduce the notations used throughout this paper (summarized in Table I), formally define real-time optimization problem and then present preliminaries on Deep Reinforcement Learning as well as Graph Neural Network.

A. Notations and Problem Definition

1) Notations. In this paper, we use bold lower case letters for vectors (e.g., $\mathbf{v}$), lower case letters for scalars (e.g., α), $\mathcal{T}_i$ for a team configuration at the i^{th} time step.

Symbols	Definition
$\mathbf{G} = \{\mathbf{V}, \mathbf{E}, \mathbf{N}\}$	an attributed social network
$\mathbf{V} = \{v_1, v_2, \ldots, v_n\}$	a set of n vertices of a network $\mathbf{G}$
$\mathbf{E} = \{(v_i, v_j) \vert v_i, v_j \in \mathbf{V}\}$	a set of edges
$\mathbf{N} \in \mathbb{R}^{n \times l}$	attribute matrix
$\mathcal{T} = \{a_1, a_2, \ldots, a_k \vert a_i \in \mathbf{V}\}$	a team of members from the network
$\mathbf{G}(\mathcal{T})$	the team network indexed by its members $\mathcal{T}$
$\mathbf{f}(\mathcal{T})$	skill representation of the team $\mathcal{T}$
$\mathbf{v}(v_i)$	the attribute/skill vector of vertex v_i
$\mu(v_i)$	vector representation of vertex v_i
l	the total number of skills
$n = \vert\mathbf{V}\vert$	the total number of individuals in $\mathbf{G}$
α, γ	learning rate and discount factor

TABLE I: Symbols and Definitions

2) Problem Definition In our problem setting, we are given a large social network of n individuals (i.e., $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$), consisting of subgraphs (i.e., teams represented by $\mathcal{T}$). The attribute/skill vector $\mathbf{v}(v_i)$ for an individual $v_i \in \mathbf{V}$ represents the strength of the skills possessed by the corresponding individual. We consider a team which has high performance to be a template team, denoted as $\mathcal{T}_s$. We detail how we choose template teams in the experimental setting in Section IV-B.

We study the real-time team optimization problem in which a team agent interacts with the environment by sequentially taking some enhancement actions (e.g., hiring a new team member, removing an existing team member) over a sequence of time steps, so as to maximize its cumulative reward (see Figure 1 for an illustrative 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 $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$ as follows:

- **State space** $\mathcal{S}$. The state $s_t \in \mathcal{S}$ is defined as the team configuration at time step t, i.e., $\mathcal{T}_t = \{a_1, a_2, \ldots\}$ where $a_1, a_2, \ldots$ are the members of the current team, $\mu(s_t)$ is the representation of the state:
$$\mu(s_t) = \sum_{v \in \mathcal{T}_t} \mu(v) \tag{1}$$
where $\mu(v)$ is the representation of the individual v in team $\mathcal{T}_t$. In this paper, we use two types of representations for individuals, including (1) the attribute/skill vector, i.e., $\mathbf{v}(v)$; and (2) the node embedding which is obtained by the graph neural network models, i.e., $\mu(v)$.
- **Action space** $\mathcal{A}$. The action $a_t \in \mathcal{A} \subseteq \mathbf{V}$ at the t^{th} time step 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 $\Delta\mathbf{E}(\mathcal{T}, \mathcal{T})$ (perturbation to the team network structure), $\Delta\mathbf{N}(\mathcal{T}, :)$ (perturbation to the team skill configuration), $+q$ (hiring q to join the team), and $-q$ (remove q from current team). In this paper, we focus on the enhancement actions of adding new members to the team and removing existing team members from the team if the member represented by the action a_t already exists in the team.
- **Reward** $\mathcal{R}$. After the team takes an action a_t under the state s_t, i.e., the team configuration changes at time t, the optimization agent receives rewards $r(s_t, a_t)$ according to the feedback it receives (e.g., performance evaluation, team cohesion). At time step t, the reward function is defined as,
$$r(s_t, a_t) = \text{sim}(\mu(\mathcal{T}_t), \mu(\mathcal{T}_s)) \tag{2}$$
where $\text{sim}(\mu(\mathcal{T}_t), \mu(\mathcal{T}_s))$ represents the similarity between the representation of the template team and the representation of the current team configuration. For example, when hiring a new team member, we often consider that a qualified candidate who has not only had collaborations with the existing team members (i.e., team structural consistency) but also possesses a set of skills required by the team function (i.e., attribute/skill similarity). The details of computing sim_f is presented in Section III.
- **Transition probability** $\mathcal{P}$: Transition probability $p(s_{t+1} \vert s_t, a_t)$ defines the probability of state transitioning from s_t to s_{t+1} when the team

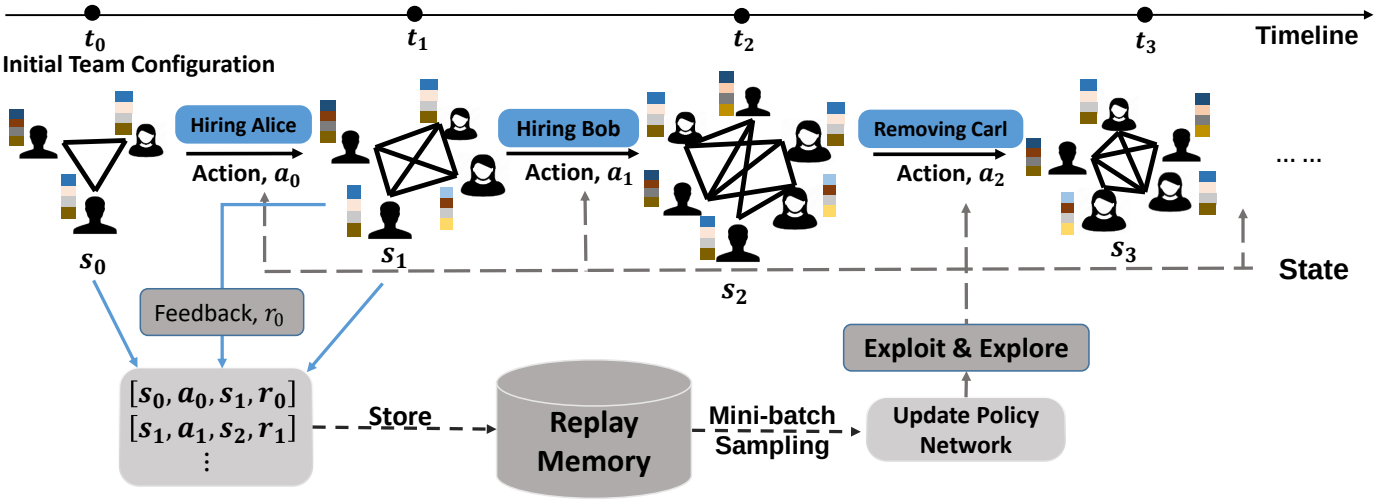


Fig. 1: An illustrative running example of real-time team optimization. A team optimization agent receives the feedback from the action made to the team and then updates the policy network which will affect the decision of actions.

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: $\gamma \in [0, 1]$ is the discount factor which is used to measure the present action-state value of future reward. In particular, when $\gamma = 0$, the team agent only considers the immediate reward and when $\gamma = 1$, all future rewards are fully taken into the consideration of calculating the action-state value of the current action a_t.
- **Terminal:** Once the team expands to have a certain number of members, the team agent stops exploring or selecting candidates from the network $\mathbf{G}$. For simplicity, we specify the size of the team with a fixed number in both training and testing.

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

Problem 1: Real Time Team Optimization

Given: given an initial configuration of a team (i.e., $\mathcal{T}_0$), the historical MDP, i.e., $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, a specified team size k, and a set of selected template teams $\mathbf{M} = \{\mathcal{T}_s^{(1)}, \mathcal{T}_s^{(2)}, \ldots\}$;

Find: an optimization policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ for a team agent to construct a team from the initial team configuration (i.e., $\mathcal{T}_0$) of size k which maximizes the similarity between the finalized team configuration, $\mathcal{T}_\pi$ and the given set of template teams.

We can formally formulate the above problem as,

$$\underset{\pi}{\text{argmax}}\, f = \sum_{\mathcal{T}_s^{(i)} \in \mathbf{M}} \text{sim}(\mathcal{T}, \mathcal{T}_s^{(i)})$$
$$\text{s.t. } \mathcal{T}_0 \subseteq \mathcal{T}_\pi \text{ and } |\mathcal{T}_\pi| = k$$

Given the above problem definition, a sample trajectory of this Markov Decision Process will be: $(s_0, a_0, r_0, \ldots, s_{m-1}, a_{m-1}, r_{m-1}, s_m)$, where $s_0 = \mathbf{G}(\mathcal{T}_0)$, $s_i = \mathbf{G}(\mathcal{T}_i), i \in \{1, \ldots, m-1\}$ and $s_m = \mathbf{G}(\mathcal{T}_\pi)$. After action a_i (e.g., adding a new member to the team) at every step i, we will receive the reward $r_i = r(s_i, a_i)$. Because this is a discrete optimization problem with a finite searching space (i.e., the number of candidates in a network $\mathbf{G}$ is finite), we propose to leverage Q-learning to learn this MDP, of which the technical details are described in Sec. III.

B. Preliminaries

1) Deep Reinforcement Learning. Research achievements have been made on utilizing reinforcement learning in many high-impact applications, ranging from recommender systems [7], [8], agent controlling in Atari games [9] to traffic signal control system [10]. In Q-Learning [11], the optimal action-state value function directly follows the *Bellman equation*, of which the intuition is: if the optimal value $Q^*(s', a')$ of the state s' at the next time-step is known for all possible actions a' in the action space $\mathcal{A}$, then the optimal strategy is to select the action a' which maximizes the expected value of $r + \gamma Q^*(s', a')$,

$$Q^*(s, a) = \mathbb{E}_{s' \sim \varepsilon}\big[r + \gamma \max_{a'} Q^*(s', a') | s, a\big] \quad (3)$$

where r is the immediate reward of action a' and γ is the discount factor. The basic idea behind many reinforcement learning algorithms is to estimate the action-state value function by iteratively updating the value, i.e., $Q_{t+1}(s, a) = \mathbb{E}\big[r + \gamma \max_{a'} Q_t(s', a') | s, a\big]$ and store the triple, (value, action, state) in a look-up table, denoted as Q-value table. However, the Q-value table will become extraordinarily large with the increasing number of states to be stored, making the traditional Q-learning method infeasible in the real world large-scale recommender systems. A follow-up work [9] proposes to leverage the convolutional neural network, i.e., Q-network, to approximate the action-state value function, which can be trained by minimizing the following loss function at each iteration i,

$$L_i(\theta_i) = \mathbb{E}_{s,a}\big[(y_i - Q(s, a; \theta_i))^2\big] \quad (4)$$

where $y_i = \mathbb{E}_{s' \sim \varepsilon}\big[r + \gamma \max_{a'} Q(s', a'; \theta_i) | s, a\big]$ is the target for the i-th iteration. As a result, the modified neural network serves as a non-linear approximator to the action-state value function, which provides a model-free method for reinforcement learning. The advantage of this method is that it does not require the calculation of the transition probability and the storage of the Q-value table. This promotes the flexibility of reinforcement learning to support a variety of application scenarios and also enriches the generalizability of the system compared to the traditional training approaches to estimate the action-state value function.

2) Graph Neural Network models.

The Graph Neural Networks (GNN) are useful neural network architectures in many applications, e.g., network embedding, node classification anomaly detection [12], [13]. For a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, we can obtain the vector representation of nodes $v \in \mathbf{V}$ by an iterative process,

$$\mu(v)^{(i)} = h^{(i)}(\big\{w(u,v), \mathbf{v}(u), \mu^{(i-1)}\big\}_{u \in N(v)}, \mathbf{v}(v), \mu(v)^{(i-1)}) \quad (5)$$

where h is a parametric function, $\mu(v)^{(i)}$ is the representation of node v at the i^{th} iteration, $i \in \{1, 2, 3, \dots\}$, $w(u, v)$ is the weight of edge $(u, v) \in \mathbf{E}$, $\mathbf{v}(u)$, $\mathbf{v}(v)$ are the attribute vectors of the node u, v respectively and $N(v)$ represents the set of neighborhood of node $v \in \mathbf{V}$ [14]. At each iteration, features information of the nodes of the neighborhood are used to computer the hidden representation of the specific node v. The initial node representation $\mu_v^{(0)} \in \mathbb{R}^d$ can be set to zero and for the simplicity of notation, after M iterations, we obtain the consequent node representation as $\mu(v) = \mu(v)^{(M)}$. After we get the node representations, we can further obtain the graph-level representation by applying pooling over all the obtained node embeddings. Recently, the Variational Graph Auto-encoder (VGAE), which utilizes a similar idea of the Variational Auto-encoder (VAE) on graph data, has shown impact on the node representation learning in an unsupervised setting. The Variational Graph Auto-Encoder is an unsupervised framework based on graph convolutional network for learning node representations. Similar to the variational autoencoder [15], in this framework, the encoder uses a two-layer GCN, denoted as function $g(\mathbf{N}, \mathbf{A}) = \widetilde{\mathbf{A}}\text{ReLU}(\widetilde{\mathbf{A}}\mathbf{N}\mathbf{W}_0)\mathbf{W}_1$ with two learnable matrices $\mathbf{W}_0$ and $\mathbf{W}_1$, where $\mathbf{A}$ is the adjacency matrix, $\widetilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix, $\mathbf{N}$ is the node attribute matrix and $\text{ReLU}(\cdot) = \max(0, \cdot)$, to compute the mean vector μ_i and the standard deviation vector σ_i^2 of a latent variable Z_i of normal distribution for each node i, i.e., $q(\mathbf{z}_i|\mathbf{N}, \mathbf{A}) = \mathcal{N}(\mathbf{z}_i|\mu_i, \sigma_i^2)$. The decoder uses the following equation to reconstruct the normalized adjacency matrix $\widetilde{\mathbf{A}}$ and for each entry $\mathbf{A}_{ij}$, we have $p(\mathbf{A}_{ij} = 1|\mathbf{z}_i, \mathbf{z}_j) = \text{sigmoid}(\mathbf{z}_i'\mathbf{z}_j)$. The loss we want to minimize is as follows,

$$\mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}\big[\log\, p(\mathbf{A}|\mathbf{Z})\big] - \text{KL}\big[q(\mathbf{Z}|\mathbf{X}, \mathbf{A})||p(\mathbf{Z})\big] \quad (6)$$

where $\text{KL}\big[q(\cdot)||p(\cdot)\big]$ is the Kullback-Leibler divergence between the two distribution.

III. 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.

A. The Proposed Real Time Team Optimization Framework

In order to achieve the optimal real time team optimization, we propose to take the enhancement actions (e.g., adding a member from the network $\mathbf{G}$ or removing a team member from the current team configuration $\mathcal{T}_t$) according to the feedback from the environment (i.e., the reward as defined in the MDP). In this paper, as mentioned above, we consider optimizing the current team to become similar to the template teams with outstanding performance through a sequence of actions (i.e., adding/removing members). More specifically, we give detailed description of the reward and the transition process of the MDP as follows,

- **Reward** $\mathcal{R}$. The reward $r(s_t, a_t)$ is defined as the average similarity between the vector representations of the team configuration at t^{th} time step and the representation of teams in the set of the template teams,

$$r(s_t, a_t) = \frac{1}{|\mathbf{M}|} \sum_{\mathcal{T}_s \in \mathbf{M}} \text{sim}(\mu(\mathcal{T}_t), \mu(\mathcal{T}_s)) \quad (7)$$

  where $\mu(\mathcal{T}_s)$ is the representation of one template team from the set $\mathbf{M}$ (i.e., $\mathbf{M} = \{\mathcal{T}_s^{(1)}, \mathcal{T}_s^{(2)}, \dots\}$) and $\mu(\mathcal{T}_t)$ is the current team representation. According to the definition in Eq. 1, the representation of a team is the linear aggregation of the representations of all members, we then have $\mu(\mathcal{T}_t) = \sum_{a_i \in \mathcal{T}_t} \mu(a_i)$. We leverage the cosine similarity to calculate the similarity between the two vectors, as presented below,

$$\text{sim}(\mu(\mathcal{T}_t), \mu(\mathcal{T}_s)) = \frac{\mu(\mathcal{T}_t)\mu(\mathcal{T}_s)}{\|\mu(\mathcal{T}_s)\|\|\mu(\mathcal{T}_s)\|} \quad (8)$$

- **Transition from** s_t **to** s_{t+1}. As mentioned, in this paper, we consider two types of team enhancement actions, i.e., adding a new member to the current team from the network $\mathbf{G}$ and removing an existing member from the current team. Therefore, the state at the next time step can be expressed as $\mu(s_{t+1}) = \mu(\mathcal{T}_{t+1}) = \mu(\mathcal{T}_t) + \mu(a_t)$ when a new team candidate a_t is selected and added to the current team and $\mu(s_{t+1}) = \mu(\mathcal{T}_{t+1}) = \mu(\mathcal{T}_t) - \mu(a_t)$ when an existing member in the current team is leaving.

B. The Standard DQN Model

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_t and action a_t. Using Bellman optimality equation [16], the optimal state-action function $Q^*(s, a)$ can be written as follows via one-step lookahead:

$$Q^*(s_t, a_t) = r(s_t, a_t) + \gamma \max_{a'} Q^*(s_{t+1}, a') \quad (9)$$

This implicitly indicates a greedy policy of selecting actions:

$$\pi(a_t|s_t; Q^*) = \underset{a_t}{\operatorname{argmax}}\, Q^*(s_t, a_t) \tag{10}$$

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

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)] \tag{11}$$

where α is the learning rate and s_{t+1} is the next state after action a_t is taken, e.g., $\mu(s_{t+1}) = \mu(s_t) + \mu(a_t)$.

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. In practice, the action-state value function is often highly nonlinear. We therefore refer to a non-linear approximator which is parameterized by neural networks to estimate the action-state value function, i.e., $Q^*(s, a) \approx Q(s, a; \theta)$ where θ is the parameter of the neural networks. Specifically, the Q function can be parameterized as,

$$Q^*(s_t, a_t) = \mathrm{ReLU}(W^{(2)}_{\theta_2}(\mathrm{ReLU}(W^{(1)\mathrm{T}}_{\theta_1}[\mu(s_t), \mu(a_t)]))) \tag{12}$$

where $W^{(1)}_{\theta_1}$, $W^{(2)}_{\theta_2}$ are the parameters of the corresponding layer in the network, $\mathrm{ReLU}(\cdot)$ is the activation function for each layer, $\mu(s_t)$, $\mu(a_t)$ are the vector representation of the current state s_t (i.e., current team configuration $\mu(s_t) = \mu(\mathcal{T}_t)$) and of the selected member a_t.

The Q-network which approximates the action-state value function is trained to minimize the following loss function $L(\theta)$,

$$L(\theta) = \mathbb{E}_{s_t, a_t, r, s_{t+1}}[(y - Q(s_t, a_t; \theta))^2] \tag{13}$$

where $y = \mathbb{E}_{s_{t+1}}[r + \gamma \max_{a'} Q(s_{t+1}, a'; \theta_t)|s_t, a_t]$ is the expected action-state value for the current episode and θ_t is the parameters from the last episode. The derivatives of the loss function $L(\theta)$ with respect to θ can be written as:

$$\nabla_\theta L(\theta) = \mathbb{E}_{s_t, a_t, r, s_{t+1}}[(r + \gamma \max_{a'} Q(s_{t+1}, a'; \theta) - Q(s_t, a_t; \theta))\nabla_\theta Q(s_t, a_t; \theta)] \tag{14}$$

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

C. Off-policy training

With the proposed Q-network, we train the parameters of the model from the offline log of different teams' development, including the actions the team agent takes and the reward it receives. To be specific, we train two separate team optimization agents with two types of representation for the individuals in the network, (1) attribute/skill vector and (2) node representation obtained by the VGAE model. For each episode in the training process, the agent starts with making actions, e.g., selecting candidates from the network $\mathbf{G}$, and then receive reward from the environment before updating the state. We also apply the $\epsilon-\mathrm{greedy}$ policy in selecting actions, which enables the agent to explore more qualified candidates rather than always selecting the best candidate at present. The value of ϵ decreases for every certain number of episodes, and the discount factor γ is set to 1. The off-policy training algorithm is presented in Algorithm 1.

Algorithm 1 Off-policy Training for Real Time Team Optimization

Input: (1) vector representations of all nodes in graph $\mathbf{G}$, (2) a set of template teams $\mathbf{M}$, (3) the learning rate α, (4) the discount factor γ, (5) initial and end probabilities of accepting the current selection, ϵ_i and ϵ_e (6) the size of a team k;

Output: a model of policy for real time team optimization.

```
Initialize replay memory D with the capacity of m;
Initialize action-state value function Q with random weights θ;
Initialize target action-state value function Q̃ with random weights θ̃ = θ;
for episode = 1, ..., T do
    Initialize a zero state representation μ(s_0) and a team configuration T_0 with empty members;
    while |T| ≠ k do
        Select a random action a_t with probability ε, otherwise select a_t = argmax_a Q(s_t, a; θ)
        Execute action a_t, compute reward r(s_t, a_t) using Eq. 7;
        Update team configuration by adding a_t to T if a_t ∉ T or removing a_t if a_t ∈ T and set s_{t+1} = s_t;
        Store the transition (s_t, a_t, s_{t+1}, r(s_t, a_t)) into the memory D;
        Sample a mini-batch of transitions (s_t, a_t, s_{t+1}, r) from D;
        Optimize the network parameter θ via stochastic gradient decent on the loss defined in Eq. 13;
    end while
    Set the network parameter Q̃ = Q every 10 episodes;
    Update the ε value as ε = ε_e + (ε_i − ε_e)e^{−episode/200};
end for
```

D. Other team optimization methods

1) Random Walk Graph Kernel [5] proposes to apply random walk graph kernel to recommend the best team candidate if one team member becomes unavailable. It selects the best candidate member c by choosing the one who maximizes the similarity between the old and the new teams embedded in social networks as follows,

$$\underset{c}{\operatorname{argmax}}\, \mathcal{K}(\mathbf{G}^*, \mathbf{G}_c) = \mathbf{q}'_\times \left(\mathbf{I} - \alpha \mathbf{A}_\times\right)^{-1} \mathbf{N}_\times \mathbf{p}_\times \tag{15}$$

where $\mathbf{G}^*$ is original team, $\mathbf{G}_c$ is the new team after selecting

member c, $\mathbf{A}_\times = \mathbf{N}_\times(\mathbf{A}^* \otimes \mathbf{A}_c)$ is the matrix of the two networks' Kronecker product, $\otimes$ represents the Kronecker product of two matrices, α is a decay factor, $\mathbf{q}_\times = \mathbf{q}^* \otimes \mathbf{q}_c$ and $\mathbf{p}_\times = \mathbf{p}^* \otimes \mathbf{p}_c$ are two vectors representing the stopping and starting probability of random walk respectively, $\mathbf{N}_\times$ is a diagonal matrix and can be expressed as $\mathbf{N}_\times = \sum_{i=1}^{l} \text{diag}(\mathbf{N}^*(:, i)) \otimes \text{diag}(\mathbf{N}_c(:, i))$ and $\mathbf{N} \in \mathbb{R}^{n \times d}$ is the attribute matrix. This team recommendation strategy works as follow, at each time-step, it selects the member from the entire social network which maximizes the similarity between the optimized team and the predefined high-performance team.

2) Skill based team recommendation selects a team candidate which maximizes the average cosine similarity between the skill representation of the current team $\mathbf{f}(\mathcal{T}_t)$ and that of the template teams, i.e., $\text{argmax}_c \frac{1}{|\mathbf{M}|} \sum_{\mathcal{T}' \in \mathbf{M}} \text{sim}(\mathbf{f}(\mathcal{T}_t) + \mathbf{v}(c), \mathbf{f}(\mathcal{T}'))$.

IV. Empirical Evaluations

In this section, we conduct extensive experiments to evaluate our proposed algorithms in order to answer the following two questions:

- **Structural consistency.** From the perspective of the topology of the network, do the members in the newly optimized teams have similar collaboration networks as the members in template teams?
- **Skill similarity.** Are the skills possessed by the new members close to the skills presented by the template teams?

A. Datasets

We use two real-world datasets, which are publicly available. Table II summarizes the statistics of these datasets and the detailed description of these datasets are as follows.

Dataset	Nodes	Edges	Attribute	# of teams
DBLP	18,674	64,089	7	30,145
Movie	13,678	416,874	20	2,578

TABLE II: Statistics of the graph datasets used for experiment

- **DBLP** is a dataset which provides bibliographic information in major computer journals and proceedings. It has (1) one co-authorship network where each node represents an author, and there is an edge between two nodes if two authors have ever published a paper together and the weight of this specific edge is the number of papers the two corresponding authors have co-authored. The authors' skills are represented by conferences in 7 different areas (i.e., DM, VIS, DB, NLP, AI, SYSTEM, MULTIMEDIA) where for a given author and a conference, the skill level of this author is defined as the percentage of the papers that s/he has published in this specific area; (2) one paper-author network which provides information of all the coauthors for a given paper and we treat the authors that have published the same paper as a team. In our experiment, we define the performance of the team as the number of citations that the corresponding paper has and we select a number of teams (i.e., papers) that have a good performance.
- **Movie** is an extension of the MovieLens dataset, which links movies from MovieLens to their corresponding IMDb webpages 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 establish the social network of the actors/actresses where each node represents one actor/actress and an edge exists if two actors/actresses have ever collaborated in the same movie with the corresponding weight being 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 and for a certain genre, the skill level of this specific person is defined as the percentage of the movies that belongs to this genre. Similarly, for a given movie, we treat all of its actors/actress as a team.

B. Experimental Design

In this subsection, we introduce the quantitative metrics and the experimental settings we apply in the evaluation of the proposed algorithm.

1) Evaluation metric. We quantify the effectiveness of the proposed algorithms by measuring the following two metrics, including (1) the consistency in the collaboration structure of the team members, where we compare the degree distributions of the members of the teams optimized by the trained agent, and (2) the similarity of the skills possessed by the optimized team and the template teams. The detailed description is as follows,

- *Structural consistency.* One essential objective of team optimization is to find a new team where the members have a similar collaboration network as the template teams (e.g., the number of collaborations that the members have in the newly optimized team is close to that of the members in the template teams). In real-world scenarios, the number of members in one team is comparatively small. For example, in *DBLP* dataset, the average number of authors in one paper (i.e., the size of team) is usually less than 10, and the number of actors/actresses in the cast of a movie is approximately 25 in *Movie* dataset. Therefore we present the distribution of the degrees of all members from the teams obtained by the proposed algorithms and comparison methods in the evaluation.
- *Skill similarity.* Another goal of team optimization is to construct a team which possesses a similar skill set as the template teams. In our experiment, we define the skill set of the team as the aggregation of the skill vectors of each member that belongs to the team, i.e., $\mathbf{f}(\mathcal{T}) = \sum_{v \in \mathcal{T}} \mathbf{v} \in \mathbb{R}^d$ where d is the dimension of the attribute/skill vector. We then use cosine similarity to measure the similarity between two teams w.r.t. attribute/skill, denoted as sim_f, where $\text{sim}_f(\mathcal{T}_1, \mathcal{T}_2) = \frac{\mathbf{f}(\mathcal{T}_1) \cdot \mathbf{f}(\mathcal{T}_2)}{\|\mathbf{f}(\mathcal{T}_1)\|_2 \|\mathbf{f}(\mathcal{T}_2)\|_2}$.

2) Experimental setting. In the experiment on *DBLP* dataset, we consider a paper (i.e., a team of authors) of which the

number of citations is larger than 40 to be a highly-cited publication and the authors in the corresponding paper are considered to be a high-performance team (denoted as one template team). In the training phase of the proposed algorithm, we randomly select 500 template teams and perform the off-policy training according to Algorithm 1. We apply two types of node representations, attribute/skill vectors and VGAE node embeddings respectively in both training and testing, and we use a suffix 'A' and 'G' to represent either variant. The average number of members in one template team is 4.56 and the average degree of the members in the template teams is 15.13. Accordingly, we set the parameter, $m = 5$, where one episode terminates after the number of selected members reaches 5. In the testing phase, we perform the evaluation of the trained team optimization agent in the following three types of settings of the initial configuration of the testing team, (1) two initial members with the same top skill that exist in our selected template teams (setting 1), (2) two initial members from non-template teams (setting 2) and (3) no initial member exists (i.e., starting from scratch) (setting 3). For each of the three settings, the evaluation is performed on 600 different randomly configured initial teams. The eventual team size (i.e., k) in the testing phase is set to be 4 and 6 respectively (i.e., 300 teams of size 4 and 6 respectively), the training and testing of the proposed algorithm are performed for ten times. The results presented in the next subsection are the average of evaluation metrics. For the experiment on the *Movie* dataset, a similar evaluation procedure is conducted. We consider a movie with an average rating of 8.0 as a top-rated movie and its actors/actresses in the cast are considered template movie teams. Because the statistics of the *Movie* dataset differs from the *DBLP* dataset, e.g., the average number of actors/actresses in one top-rated movie is 25.52 and the average number of collaborations for one individual actor/actress is 36.72. Thus in the training phase, we randomly select 360 movies to be the template teams and set $k = 24$ in order for one episode to terminate; in the testing phase, the same three settings of the initial team configuration are employed but the differences are (1) six initial members are given for the first two types of test settings and (2) the eventual size of testing teams is defined to be 24, which approximates the average size of template teams.

3) Repeatability and Machine Configuration. The experiments are performed on a virtual environment on Windows 10 - Intel Core i5-7300 at 2.60 GHz and 16GB RAM and are implemented in Python 3.6. All datasets are publicly available. We will release the code upon the publication of this paper.

C. Quantitative Results

1) Evaluation on *DBLP*. We first compare the degree distribution of all the members from the teams that are obtained by the proposed algorithms and the comparison methods. The result is summarized in Figure 2a. We can see the three lines (red dash-dotted, purple solid and green dashed) which represent the degree distributions of members from the teams obtained by Algorithm-G under three different settings. These three lines approach the degree distribution of the members from the template teams (i.e., the blue solid line) to a greater extent compared with the other three lines (yellow dashed, green solid and pink dotted), which represents the degree distribution of the members from the teams obtained by Algorithm-A. This is because in the process of obtaining the node representations using VGAE, the structural information is embedded in the node representation. As a result, the trained team agent is able to select team members having a more similar collaboration structure as the members from the template teams. Another observation is that, the degree distributions representing the first type of setting (i.e., orange dashed line and red dash-dotted line) have shown the best performance w.r.t structural consistency compared with other two types of settings. In addition, the distributions representing the third type of setting (i.e., pink and green dotted line) demonstrate a better structural consistency than that of the second type of setting (i.e., green and purple solid line). This is because compared with constructing a team from scratch, an initial team of members with mediocre performance will actually impede the team agent from optimizing a team towards the direction of selecting outstanding members. The result of the graph kernel (i.e., brown dotted line) shows that it has an inclination of selecting members with a larger number of collaborations. This is because as Eq. 15 shows, a network of team with more collaborations (i.e., higher degree of node) will have a larger graph kernel, and thus team candidates with large degrees are more likely to be selected by this strategy.

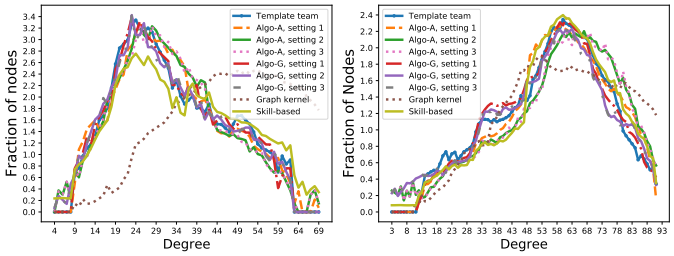
The results of the measurement of skill similarity are presented on the left side of Figure 3. We have the following observations, (1) under the same type of evaluation setting, leveraging the attribute vector in training can achieve a better similarity of skills. For example, 72.35% obtained by Algorithm-A (the orange bar), is larger than the score, 67.47% obtained by Algorithm-G (the red bar), this is consistent with our intuition because when we use the attribute/skill vector as the node representation, the objective of optimization becomes maximizing the skill similarity; (2) under setting 1, the skill similarity is higher than that under other two settings. For example, 72.35% (the orange bar) is higher than $70.58\%, 69.08\%$ (the green, pink bars respectively) and 67.47% (the red bar) is higher than $65.76\%, 63.98\%$ (the purple, grey bars respectively) where the latter two bars represent the skill similarity under the other two testing settings; (3) the graph kernel-based team optimization strategy achieves a skill similarity of 48.33%, which is consistent with the result of degree distribution in Figure 2a because it selects a large portion of candidates with large degree as well as skill vectors with large values, and thus causes a low score of skill similarity; (4) although the skill-based method achieves the best performance since it computes the similarity for each team candidate and selects the member with the largest value, it is very time-consuming.

2) Evaluation on *Movie*. The degree distribution of all members from the testing teams on the *Movie* dataset is presented in Figure 2b. We can see that even though the pattern of these distributions are different from those on *DBLP* dataset, leveraging VGAE node representations can still achieve a

closest degree distribution (i.e., the red dotted-dashed line) to the distribution of the template team members (i.e., the blue line). In addition, under the same type of evaluation setting, the model trained using VGAE node representation is more capable of optimizing the teams towards the template teams than the model which uses attribute/skill vectors. For example, the purple solid line is closer to the blue line than the green solid line (under setting 2), and the grey solid line is also closer than the pink dotted line (under setting 3). The graph kernel method shows a similar results to Figure 2a and there is a large portion of members who have higher degrees. For the skill similarity on *Movie*, from the right side of Figure 3, we have the following observations that are consistent with the results on *DBLP*: (1) the team agent trained using attribute/skill vector under setting 1 achieves the highest similarity in skill (i.e., 69.98% in orange bar) among the results obtained by other proposed methods; (2) the team agent trained with attribute/skill vectors always outperforms the agent trained with VGAE node representations, for example, under the second type of setting, 68.86% (the green bar) is higher than 64.32% (the purple bar) and 68.43% (the pink bar) is higher than 62.91% (the grey bar) under the third type of evaluation setting.

3) Parameter study. Figure 4 summarizes the comparison of degree distributions on the *Movie* dataset under two new types of settings which alter the evaluation parameter. We first fix the size of initial team configuration to be 12 and increase the number of members that are originally in the template teams from 0 to 12. From figure 4a, we can see that (1) when no members from the template teams exist in the initial team configuration, the degree distribution (i.e., the yellow dashed line) is apparently the most contrasting distribution comparing with the distribution of the template teams; (2) as we increase the number of members that are from the template teams, the distribution is gradually approaching the original distribution. Second, we alter the size of the initial team (from 3 to 12) with all members from the template teams and summarize the results in Figure 4b. It can be seen that as the size of initial team increases, the distribution is also getting closer to the original distribution (i.e., the blue line), which is consistent with our intuition. If we compare these two settings, we can see that when the size of the initial team is the same as the number of members from template teams in Figure 4a, the degree distribution of the former setting is closer to the original distribution. For example, the green solid line in Figure 4b approaches the original distribution to a greater extent than the red dotted-dashed line in Figure 4a because there are none non-template members in the initial members in the second setting.

We also compare the skill similarity when altering the parameters in the same way and Figure 5 present how the skill similarity changes. We have the following observations, (1) for both settings, as the number of members in the initial team configuration increases, the skill similarity also improves; (2) when the number of members from the template teams is the same in both settings, the initial teams with only members from the template teams achieves a higher score of skill similarity. For example, when the number of members from the template teams is 6 and the attribute vectors are leveraged in training, 69.98% (Figure 5b) is higher than 69.15% (Figure 5a).



(a) *DBLP* (b) *Movie*

Fig. 2: Aggregated degree distribution of the teams. (a) The number of seed (size of initial team) is randomly selected as one or two if existed and the eventual size of the team is defined as 4 and 6 respectively in the experiment ($k = 4, 6$). (b) Six members exist initially in every testing team and the size of the team is set as 24 ($k = 24$).

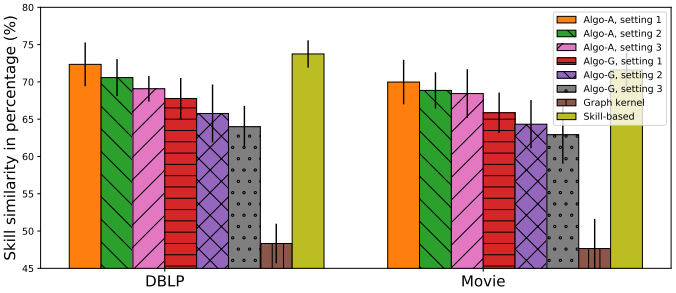
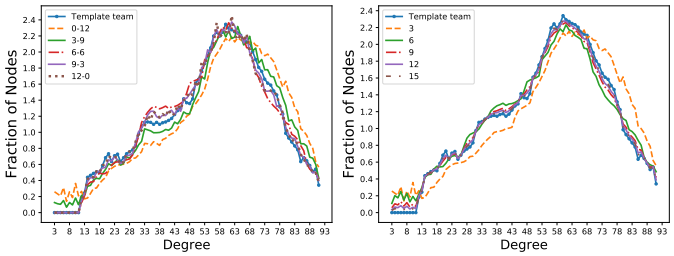
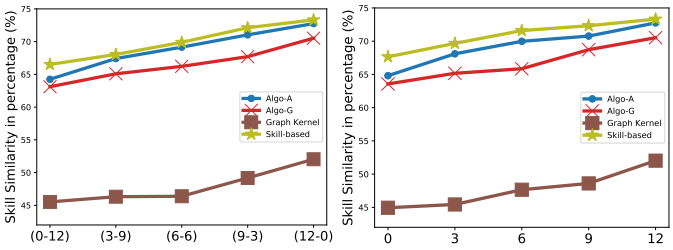


Fig. 3: The average skill similarity (in percentage) of the teams optimized by the proposed algorithms and the comparison methods. Higher is better. Best viewed in color.



(a) degree distribution vs. # of template members in $\mathcal{T}_0$. (b) degree distribution vs. # of template members in $\mathcal{T}_0$.

Fig. 4: Comparison of the degree distribution of the teams optimized by the proposed algorithm and the comparison methods. The more the line approaches the original distribution (i.e., the blue dotted line), the better. Best viewed in color. (a) the size of $\mathcal{T}_0$ is 12 of which the number of members from the template increases. (b) the size of $\mathcal{T}_0$ increases.



(a) Skill similarity vs. the fraction of members from the template teams (the size of initial team is fixed to be 12).

(b) Skill similarity vs. the number of members from the template teams (the size of initial team increases).

Fig. 5: Parameter analysis of skill similarity on *Movie* dataset. The higher the value is, the better. Best viewed in color.

D. Case study

1) Case study on *DBLP*. In this subsection, we present a case study of real-time team optimization using our proposed method. First, we apply the model trained with VGAE node representations to an initial team with two members, *Kunal Punera and Byron Dom*. After the optimization has finished, the team includes the following new members, *John Langford, Thomas Hofmann, Alexandar Smola, and Yi Chang*. We find that the new team largely overlaps with this template team, which includes *Hans Kriegel, Alexandar Smola, Kunal Punera, Yi Chang and Byron Dom*. We can see that both teams consist researchers in the area of *Machine Learning* and *AI*, among which, *Dr. Alexandar Smola* and *Dr. Byron Dom* are highly reputed researchers in these areas, which demonstrates that the proposed algorithm can recommend team candidates with similar collaboration network as well as similar types of skills present in the template team.

2) Case study on *Movie*. One representative example of team obtained by the proposed method includes the following actors/actresses, *Christian Bale, Gary Oldman, Morgan Freeman, Aaron Eckhart, Gwyneth Paltrow and Joseph Gordon-Levitt* (only a subgroup of actors/actresses in this team are listed). We can observe that the listed members in these team are popular movie actors/actresses and they have starred in several well-known highly rated movies. For example, *Christian Bale, Gary Oldman, Morgan Freeman and Aaron Eckhart* are in the main cast of the movie, *The Dark Knight* of which the rate is 9.0 on *IMDB*. *Morgan Freeman* is a *Golden Globe Award* winner with many highly rated movie, including *The Shawshank Redemption and Dark Knight Trilogy*, and he has also collaborated with *Gwyneth Paltrow and Joseph Gordon-Levitt* in the movie *Seven* and *The Dark Knight Rises*, respectively.

V. Related Work

In this section, we review the related work in terms of (a) reinforcement learning, and (b) team recommendation.

Reinforcement Learning studies the problem how an agent takes actions in an environment in order to maximize the cumulative reward until it reaches the terminal state. To date, reinforcement learning has been widely studied in the applications ranging from control system to recommendation. The MDP-based CF model can be viewed as approximating a partial observable MDP (POMDP) by applying a finite rather than an unbounded window of previous history to define the current state [7]. To better reduce the high computational and representational complexity of the POMDP, three methods have been successfully developed, including value function approximation [17], policy based optimization [18], [19], and stochastic sampling [20]. Furthermore, [21] proposes to adopt the reinforcement learning technique to observe the responses of users in a conversational recommender, with the goal of maximizing a numerical cumulative reward function which models the benefit that the users get from each recommendation session. [8], [22] model web page recommendation as a Q-Learning problem and learn to make recommendations according to the web usage data as the actions instead of mining explicit patterns from the data. To address the high-dimensional combinatorial state and action spaces, the solution based on deep reinforcement learning is introduced [23]. [9] adopts deep neural networks as an approximator of the action-value function (Q-function) and successfully implements the framework in the game of Atari. In order for better generalization as well as better performance, [24] proposes double Q-learning that can be generalized to work with large-scale function approximation.

Team Recommendation is a very active research area 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 specific field.This work is related to this in the sense that we consider the scenario of finding the optimal team recommendation strategy which could maximize the performance of the current team. To ensure a successful recommendation, the selected team members should possess the desired skills and have strong team cohesion, which is first illustrated in [25]. [26] studies forming teams to accommodate a sequence of tasks arriving in an online fashion. A popular method in recommendation (collaborative filtering) is latent factor model [27]–[29] where the fundamental idea is to identify the latent factors. The factorization based technique can be naturally extended by adding biases, temporal dynamics and varying confidence level. [30] proposes a decision support system based on a relational recommendation approach which provides an automated pre-selection of candidates that best match the future team members. In many areas, identifying experts in a research field is of great benefit. For example, assigning papers to the right reviewers in a peer-review process [31], [32], which can be done by building the co-author network [33] or using language model and topic-based model [34], [35]. In business, hiring the desired specialists is a cost-efficient way to facilitate the on-going project, and many methods have been proposed to search for an expert through an organization's document repository [36].

VI. Conclusion

In this paper, we formally define and formulate the problem of real time team optimization and propose deep reinforcement

learning based algorithms to model the dynamics of the team and the feedback from the environment. To demonstrate the effectiveness of the proposed algorithm, we conduct extensive empirical experiments and show that it can achieve the goal of optimizing teams towards the direction of a high-performing team. Future work includes extending the proposed algorithm to other recommendation scenarios, e.g., list-wise recommendation in online shopping and social recommendation.

ACKNOWLEDGEMENT

This work is supported by the National Science Foundation under Grant No. IIS-1651203, IIS-1715385, by Army Research Office under the contract number W911NF-16-1-0168, by the U.S. Department of Homeland Security under Grant Award Number 2017-ST-061-QA0001. The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] B. W. Tuckman, "Developmental sequence in small groups." *Psychological bulletin*, vol. 63, no. 6, p. 384, 1965.

[2] B. B. Morgan Jr, E. Salas, and A. S. Glickman, "An analysis of team evolution and maturation," *The Journal of General Psychology*, vol. 120, no. 3, pp. 277–291, 1993.

[3] S. Warner, M. T. Bowers, and M. A. Dixon, "Team dynamics: A social network perspective," *Journal of Sport Management*, vol. 26, no. 1, pp. 53–66, 2012.

[4] S. Dorairaj, J. Noble, and P. Malik, "Understanding team dynamics in distributed agile software development," in *International Conference on Agile Software Development*. Springer, 2012, pp. 47–61.

[5] L. Li, H. Tong, N. Cao, K. Ehrlich, Y. Lin, and N. Buchler, "Replacing the irreplaceable: Fast algorithms for team member recommendation," in *WWW*. ACM, 2015, pp. 636–646.

[6] L. Li, H. Tong, N. Cao, K. Ehrlich, Y.-R. Lin, and N. Buchler, "Enhancing team composition in professional networks: Problem definitions and fast solutions," *IEEE transactions on knowledge and data engineering*, vol. 29, no. 3, pp. 613–626, 2016.

[7] G. Shani, R. I. Brafman, and D. Heckerman, "An mdp-based recommender system," in *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 453–460. [Online]. Available: http://dl.acm.org/citation.cfm?id=2073876.2073930

[8] N. Taghipour and A. Kardan, "A hybrid web recommender system based on q-learning," in *Proceedings of the 2008 ACM Symposium on Applied Computing*, ser. SAC '08. New York, NY, USA: ACM, 2008, pp. 1164–1168. [Online]. Available: http://doi.acm.org/10.1145/1363686.1363954

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[10] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18. New York, NY, USA: ACM, 2018, pp. 2496–2505. [Online]. Available: http://doi.acm.org/10.1145/3219819.3220096

[11] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[12] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434*, 2018.

[13] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 2019, pp. 594–602.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[15] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[16] R. Bellman, *Dynamic programming*. Courier Corporation, 2013.

[17] M. Hauskrecht, "Incremental methods for computing bounds in partially observable markov decision processes," in *AAAI/IAAI*. Citeseer, 1997, pp. 734–739.

[18] A. Y. Ng and M. Jordan, "Pegasus: A policy search method for large mdps and pomdps," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2000, pp. 406–415.

[19] P. Poupart and C. Boutilier, "Vdcbpi: an approximate scalable algorithm for large pomdps," in *Advances in Neural Information Processing Systems*, 2005, pp. 1081–1088.

[20] M. Kearns, Y. Mansour, and A. Y. Ng, "A sparse sampling algorithm for near-optimal planning in large markov decision processes," *Machine learning*, vol. 49, no. 2-3, pp. 193–208, 2002.

[21] T. Mahmood and F. Ricci, "Improving recommender systems with adaptive conversational strategies," in *Proceedings of the 20th ACM conference on Hypertext and hypermedia*. ACM, 2009, pp. 73–82.

[22] N. Taghipour, A. Kardan, and S. S. Ghidary, "Usage-based web recommendations: a reinforcement learning approach," in *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 2007, pp. 113–120.

[23] P. Sunehag, R. Evans, G. Dulac-Arnold, Y. Zwols, D. Visentin, and B. Coppin, "Deep reinforcement learning with attention for slate markov decision processes with high-dimensional states and actions," *arXiv preprint arXiv:1512.01124*, 2015.

[24] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[25] T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 467–476.

[26] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi, "Online team formation in social networks," in *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012, pp. 839–848.

[27] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[28] G. Dror, N. Koenigstein, and Y. Koren, "Web-scale media recommendation systems," *Proceedings of the IEEE*, vol. 100, no. 9, pp. 2722–2736, 2012.

[29] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 448–456.

[30] J. Malinowski, T. Weitzel, and T. Keim, "Decision support for team staffing: An automated relational recommendation approach," *Decision Support Systems*, vol. 45, no. 3, pp. 429–447, 2008.

[31] D. Mimno and A. McCallum, "Expertise modeling for matching papers with reviewers," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 500–509.

[32] M. Karimzadehgan and C. Zhai, "Constrained multi-aspect expertise matching for committee review assignment," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 1697–1700.

[33] K. Balog, Y. Fang, M. de Rijke, P. Serdyukov, L. Si *et al.*, "Expertise retrieval," *Foundations and Trends® in Information Retrieval*, vol. 6, no. 2–3, pp. 127–256, 2012.

[34] H. Deng, I. King, and M. R. Lyu, "Formal models for expert finding on dblp bibliography data," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 163–172.

[35] S. H. Hashemi, M. Neshati, and H. Beigy, "Expertise retrieval in bibliographic network: a topic dominance learning approach," in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2013, pp. 1117–1126.

[36] K. Balog, L. Azzopardi, and M. De Rijke, "Formal models for expert finding in enterprise corpora," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006, pp. 43–50.