A QUBO Framework for Team Formation

Karan Vombatkere[1] ✉, Theodoros Lappas[2], and Evimaria Terzi[1]

[1] Boston University {kvombat,evimaria}@bu.edu
[2] Satalia theodoros.lappas@satalia.com

Abstract. The team formation problem assumes a set of experts and a task, where each expert has a set of skills and the task requires some skills. The objective is to find a set of experts that maximizes coverage of the required skills while simultaneously minimizing the costs associated with the experts. Different definitions of cost have traditionally led to distinct problem formulations and algorithmic solutions. We introduce the unified TeamFormation formulation that captures all cost definitions for team formation problems that balance task coverage and expert cost. Specifically, we formulate three TeamFormation variants with different cost functions using quadratic unconstrained binary optimization (QUBO), and we evaluate two distinct general-purpose solution methods. We show that solutions based on the QUBO formulations of TeamFormation problems are at least as good as those produced by established baselines. Furthermore, we show that QUBO-based solutions leveraging graph neural networks can effectively learn representations of experts and skills to enable transfer learning, allowing node embeddings from one problem instance to be efficiently applied to another.

Keywords: Team Formation · Quadratic Binary Optimization (QUBO) · Graph Neural Network (GNN) · Combinatorial Optimization.

1 Introduction

The team formation problem is commonly defined as follows: given a set of experts, each possessing a set of skills, and a task that requires specific skills, the goal is to identify a subset of experts best suited to complete the task. A vibrant stream of literature has been dedicated to algorithmic solutions for addressing an ever-expanding universe of variants of this problem [1,2,13,16,18,24,34,35].

The fundamental requirements in most team formation problems is that the selected experts maximize the *coverage* of the required skills while minimizing their *cost*. Existing work on this problem combines these two requirements, by setting one as a constraint and the other as the objective. The cost of a team has many different definitions with each leading to a different problem formulation. Common cost functions include a linear sum of individual expert costs or a network-based cost that accounts for the structural connectivity of the selected experts within an underlying social graph.

Inspired by recent work [25,34], we integrate both the coverage and cost objectives aiming to find a team $\mathbf{x}$ for task J such that $\lambda Cov(J \mid \mathbf{x}) - Cost(\mathbf{x})$ is

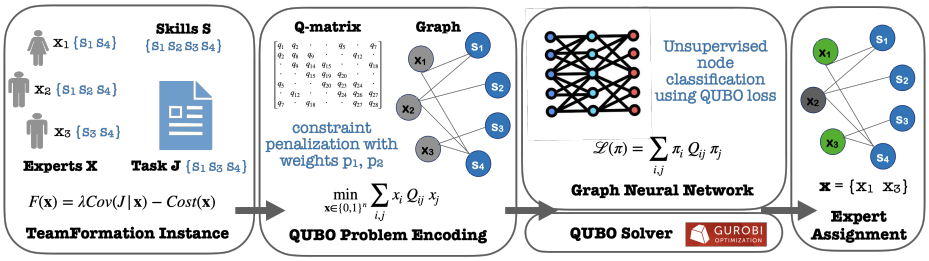

Fig. 1: High-level flowchart of our QUBO framework for TeamFormation.

maximized. We call this general problem TeamFormation. In this formulation, λ is a normalization factor that balances the two components of the objective. This formulation is general and can incorporate direct costs associated with experts or more complex cost functions, e.g., coordination costs.

In this paper, we examine three variants of the TeamFormation problem resulting from different cost functions, and show that they can be expressed as quadratic unconstrained binary optimization (QUBO) problems. This perspective enables us to frame team formation as an energy minimization problem, drawing parallels with physics-based combinatorial optimization techniques.

We explore two classes of solution methods: one using QUBO solvers [12] and another leveraging graph neural networks (GNNs) [29]. QUBO solvers provide exact or near-optimal solutions. However, they operate as black-box solvers that do not provide any insight into the underlying space of experts and skills, and their computational complexity grows significantly with problem size.

Motivated by these limitations, and inspired by recent work on deep learning for combinatorial optimization problems [6,30], we introduce a GNN-based approach. This approach models the problem as an unsupervised node classification task; the classification process assigns each expert a binary decision (selected or not selected in the team) and the GNN learns to classify the experts by optimizing a QUBO-based loss function that corresponds to maximizing the TeamFormation objective. Apart from learning good solutions, the embeddings learned by the GNN provide a semantic representation of the expert-skill space, where node proximity reflects relationships between skills and experts.

To the best of our knowledge, we are the first to provide a unified QUBO-based framework (see Fig. 1) for team formation, enabling a consistent algorithmic approach across different TeamFormation variants. In our experimental evaluation, we utilize real-world datasets from diverse domains, including collaboration networks of artists and scientists, and online labor market data. Our results demonstrate that our general algorithms consistently find high-quality solutions, often outperforming combinatorial baselines designed specifically for certain problem variants. Furthermore, our experiments highlight the potential for transfer learning, where GNNs trained on one problem instance can be effectively used to solve related instances with minimal additional computation.

2 Related Work

Our QUBO-based formulation for the TeamFormation problem applies to all variants requiring a balance between coverage and cost. In this way, our work generalizes a lot of existing work on team formation, relates to work on balancing submodular objectives with other objective functions, and extends ideas from QUBO combinatorial optimization and deep learning.

Algorithmic Team Formation. Early work in team formation focused on algorithmic methods to select experts to collectively cover *all* the skills required by a single task, while collaborating effectively within a social network [16,18]. Related work considered forming multiple teams of experts to cover the skills of multiple tasks while bounding the workload or coordination cost across experts [1,2]. Follow-up works then considered more flexible problem variations that aim to balance partial task coverage with expert cost, maximum workload, and coordination cost. These works primarily employ established algorithmic methods, such as integer programming and greedy heuristics [24,25,34,35].

More recent literature has expanded beyond such methods to leverage deep learning for various team-formation variants. For instance, deep neural networks have been used to recommend new teammates to optimally compose high-performance teams [8,28]. In another relevant example, a variational bayesian neural architecture was used to learn representations for teams whose members have collaborated in the past, enabling the selection of top-k teams of experts that collectively cover a set of skills [13].

Our TeamFormation formulation generalizes several prior formulations by incorporating task coverage and a flexible cost definition into a single objective. Furthermore, our GNN-based method is distinct from the deep learning methods used in prior work.

Submodular Maximization. The coverage function is monotone and submodular, which is useful within discrete objective functions, as it encodes a natural diminishing returns property and also comes with an extensive literature on optimization techniques [9,10,17]. The greedy algorithm achieves a $1-1/e$ approximation for maximizing a nonnegative monotone submodular function subject to a cardinality constraint [23]. There is also work involving maximizing submodular minus modular or linear functions, where no multiplicative approximation guarantees are possible in polynomial time due to potential negativity [14,15].

The $Cov()$ function in our TeamFormation objective is nonnegative monotone submodular, and depending on the definition of $Cost()$ used, variants of our general problem relate to balancing submodular and other functions. However, our solution framework is general and it does not rely on the fact that our functions have these properties.

Combinatorial Optimization and QUBO. Many NP-hard combinatorial optimization problems have been formulated as QUBO problems [11,19]. More recently, QUBO has been used as a framework for mapping discrete optimization problems to quantum and classical solvers. Methods for encoding problem constraints, such as unbalanced penalization and slack variable techniques, en-

able the transformation of constrained combinatorial optimization problems into QUBO [3,22,27,33].

We borrow ideas from prior work to formulate TeamFormation problems as combinatorial optimization using QUBO, and use the unbalanced penalization technique [22] to make our formulation more efficient.

Deep Learning for Combinatorial Optimization. Neural combinatorial optimization has gained traction as an alternative to traditional optimization methods, and recent work in reinforcement learning has explored policy-gradient methods and graph-based architectures [4,7]. Neural networks have also been used to learn representations of discrete sets effectively, enhancing the performance of models in tasks involving set-structured data. [32,37].

GNNs have been used to augment existing solvers by identifying smaller sub-problems to reduce the search space for NP-hard problems such as Set Cover [31]. More closely related to our work, GNNs have been used to solve QUBO-formulated combinatorial optimization problems such as Maximum Independent Set and Maximum Cut, by leveraging their ability to encode graph structures and learn meaningful representations [30].

We extend ideas from the deep learning combinatorial optimization literature to design our GNN architecture to solve the TeamFormation problem.

3 Technical Preliminaries

3.1 Team Formation

Experts, tasks and skills. Consider a set of n experts $\mathcal{X} = \{X_1, \ldots, X_n\}$, and a single task J. We assume a set of m skills S such that the task J *requires* a set of skills (i.e., $J \subseteq S$) and every expert X_i *masters* a set of skills (i.e., $X_i \subseteq S$).

Assignments. We represent an *assignment* of experts to a task J using $\mathbf{x} \in \{0,1\}^n$; $\mathbf{x}(i) = 1$ (resp. $\mathbf{x}(i) = 0$) if expert X_i is (resp. not) assigned to J.

Task Coverage. Given an assignment $\mathbf{x}$, we define the *coverage* of task J, denoted by $Cov(J \mid \mathbf{x})$, as the number of skills required by J that are covered by the experts assigned to J. That is, $Cov(J \mid \mathbf{x}) = |(\cup_{i \in \mathbf{x}} X_i) \cap J|$, with $0 \leq Cov(J \mid \mathbf{x}) \leq |J|$. We denote the *size* of $\mathbf{x}_i$, i.e., the assignment for task J_i, by $z_i = ||\mathbf{x}_i||_1$. This corresponds to the sum of 1-entries in $\mathbf{x}_i$.

Expert Costs. The cost of an assignment $\mathbf{x}$, denoted by $Cost(\mathbf{x})$, encodes the cost of hiring the experts chosen in $\mathbf{x}$. Inspired by prior related research, we consider the following established definitions of cost:

Cardinality cost: It is often necessary to constrain the *size* of the team, such that the total number of assigned experts is less than or equal to a specified size constraint k. This can be encoded as:

$$Cost_k(\mathbf{x}) = \begin{cases} 0 & \text{if } |(\cup_{i \in \mathbf{x}} X_i)| \leq k \\ \infty & \text{otherwise.} \end{cases}$$

Linear cost: The linear cost is based on ideas first introduced by Nikolakaki et al. [25]. In this case, each expert X_i is associated with a cost κ_i, representing the cost of hiring that expert. The total cost of an assignment $\mathbf{x}$ is the sum of costs of the individual experts in the assignment:

$$Cost_L(\mathbf{x}) = \sum_{i \in \mathbf{x}} \kappa_i.$$

Network coordination cost: When a set of experts is hired, then there is coordination cost among the experts. We model this by assuming that there is a graph $G = (\mathcal{X}, E)$ between the experts (nodes) and that their pairwise coordination costs are encoded in the weights of the edges between them. We thus assume that $d(X_i, X_j) : E \to \mathbb{R}_{\geq 0}$ encodes the coordination cost between two experts. The relevant literature has suggested multiple definitions of coordination cost based on such underlying graphs [2,18,34]. Inspired by prior work, we define the total coordination cost of an assignment $\mathbf{x}$ as the sum of pairwise costs of experts in the assignment:

$$Cost_G(\mathbf{x}) = \sum_{(i \in \mathbf{x}, j \in \mathbf{x})} d(X_i, X_j).$$

3.2 Quadratic Unconstrained Binary Optimization

Quadratic unconstrained binary optimization (QUBO) is a mathematical optimization framework used to model combinatorial problems where variables take binary values. For a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ of binary decision variables ($x_i \in \{0, 1\}$), the objective function is represented as a quadratic expression of these binary variables:

$$\min_{\mathbf{x} \in \{0,1\}^n} \mathbf{x}^T Q \mathbf{x} \; = \min_{\mathbf{x} \in \{0,1\}^n} \sum_{i,j} x_i \, Q_{ij} \, x_j, \tag{1}$$

where Q (i.e. the Q-matrix) is an $n \times n$ symmetric matrix, with entries Q_{ij}. The Q-matrix encodes problem-specific interactions between variables. QUBO is an NP-hard optimization problem [21].

Solvers. Classical solvers, such as Gurobi's QUBO optimizer and CPLEX, use mixed-integer programming (MIP), branch-and-bound, and specialized heuristic methods to find optimal or near-optimal solutions to QUBO problems [12].

Linear Programs as QUBO. A linear program (LP) with binary variables $\mathbf{x}$ can be represented as QUBO by reformulating equality constraints using quadratic penalty terms [11,27]. Consider an LP of the form $\min \mathbf{c}^T \mathbf{x}$ subject to equality constraints $A\mathbf{x} = \mathbf{b}$, where $\mathbf{x}$ is any length-n binary vector, A is a $(m \times n)$ matrix and $\mathbf{b}$ is a length-m vector. Denoting $C = \text{diag}(\mathbf{c})$, and for an appropriate scalar penalty p we have the following equivalence:

$$\begin{aligned} \min_{\mathbf{x}} \mathbf{c}^T \mathbf{x} \text{ (s.t. } A\mathbf{x} = \mathbf{b}) &= \min_{\mathbf{x}} \mathbf{x}^T C \mathbf{x} + p(A\mathbf{x} - \mathbf{b})^T (A\mathbf{x} - \mathbf{b}) \\ &= \min_{\mathbf{x}} \mathbf{x}^T Q \mathbf{x} + p \mathbf{b}^T \mathbf{b}. \end{aligned}$$

The optimal solution to the LP $\min_{\mathbf{x}} \mathbf{c}^T\mathbf{x}$ subject to $A\mathbf{x} = \mathbf{b}$ corresponds to the optimal solution to $\min_{\mathbf{x}} \mathbf{x}^T Q\mathbf{x}$, where $Q = C + p(A^TA) - 2p\,\mathrm{diag}(A^T\mathbf{b})$ is the Q-matrix of the QUBO encoding, and we dropped the additive constant $p\mathbf{b}^T\mathbf{b}$.

Unbalanced Penalization. To transform an LP with inequality constraints, typically slack variables are introduced as follows: given a constraint $\sum_i a_{ij}x_i \leq b_j$, where $a_i, b_j \in \mathbb{Z}$ for every $j = \{1, \ldots, m\}$, a non-negative slack variable encoded as a sum of binary variables $\hat{s} = \sum_k 2^k s_k$ (where $s_k \in \{0, 1\}$), is added so the constraint becomes $\sum_i a_i x_i + \hat{s} = b_j$. The reformulated equality is then enforced in the objective function using a quadratic penalty term $p(\sum_i a_i x_i + \sum_k 2^k s_k - b_j)^2$, where p is a sufficiently large penalty coefficient.

The primary drawback of slack variables is the increase in dimensionality of the LP – and the size of the Q-matrix – by $\log\lceil b_j - \sum_i a_i x_i \rceil$ for each inequality constraint. Consequently, for the problems in this paper, we eliminate the need for slack variables by incorporating unbalanced penalization [22]. This technique encodes an asymmetric penalty function (directly into the QUBO objective) which is small when a constraint is satisfied and increases significantly when violated, without increasing the problem's dimensionality.

We provide all mathematical details to use unbalanced penalization to formulate team formation LPs into QUBO in Section 4.

4 QUBO Framework for Team Formation

In this section we introduce the general TeamFormation problem, and detail three variants, which we then formulate using QUBO.

4.1 The TeamFormation Problem

Given a set of experts $\mathcal{X}$, and a task J, we define the general TeamFormation problem as follows: find an assignment $\mathbf{x}$ that *maximizes* the objective

$$F(\mathbf{x}) = \lambda Cov(J \mid \mathbf{x}) - Cost(\mathbf{x}). \tag{2}$$

The above function balances the coverage of task J achieved by a specific team with the cost of the team. Parameter λ is application dependent and can be used to tune the importance of the two components of the objective.

We now define three instantiations of the TeamFormation problem, which have different cost functions. We express each of these problems using constrained linear programming and apply the unbalanced penalization technique (see Sec. 3.2) to construct the corresponding Q-matrix.

Throughout this section we use the vector $\mathbf{y} = \mathbf{s} \,||\, \mathbf{x}$ which represents the solution to our problems. We call $\mathbf{y}$ the *solution vector*. This vector is of size $(m + n)$ and is the concatenation of $\mathbf{s}$ and $\mathbf{x}$, where $\mathbf{s}$ is a binary vector that encodes whether a skill i is covered (resp. not covered) by $\mathbf{s}$ when $s_i = 1$ (resp. $s_i = 0$). We also use the $(n \times m)$ skill-membership matrix E such that $E(i, j) = 1$ (resp. 0) if expert i has (resp. not) skill j.

Due to space constraints, we omit several mathematical details and refer the reader to the supplementary material for derivations of the QUBO formulations.

4.2 Max-k-Cover

Problem 1 (Max-k-Cover). Given a set of n experts $\mathcal{X} = \{X_1, \ldots, X_n\}$, a task J, and a cardinality constraint k, find an assignment $\mathbf{x}$ of experts such that the following is maximized:

$$F(\mathbf{x}) = \lambda Cov(J \mid \mathbf{x}) - Cost_k(\mathbf{x}). \tag{3}$$

QUBO Formulation Sketch. Let $\mathbf{y} = \mathbf{s} \mid\mid \mathbf{x}$, be the $(m+n)$-size solution vector we described above. Now let $\mathbf{c}$ be another $(m+n)$ vector such that $c_i = \lambda$ if $i \leq m$ *and* skill $i \in J$, and $c_i = 0$ otherwise. Then, the Max-k-Cover problem can be expressed by the following linear program:

$$\begin{aligned} \text{maximize} \quad & \mathbf{c}^T\mathbf{y}, \\ \text{such that} \quad & \sum_{i=1}^{n} x_i \leq k \\ & s_j - \sum_{i=1}^{n} E(i,j) \cdot x_i \leq 0 \quad \text{for all } 1 \leq j \leq m, \text{ and} \\ & s_i, x_i \in \{0,1\}. \end{aligned}$$

We derive penalty matrices P_k and P_C corresponding to the LP constraints. Then the $(m+n) \times (m+n)$ square matrix $Q = -\text{diag}(\mathbf{c}) - P_k + P_C$ provides a QUBO formulation of Max-k-Cover, where minimizing $\mathbf{y}^T Q \mathbf{y}$ corresponds to maximizing $F(\mathbf{x}) = \lambda Cov(J \mid \mathbf{x}) - Cost_k(\mathbf{x})$.

4.3 Coverage-Linear-Cost

Problem 2 (Coverage-Linear-Cost). Given a set of n experts $\mathcal{X} = \{X_1, \ldots, X_n\}$ with their corresponding individual costs $\{\kappa_1, \ldots, \kappa_n\}$, and a task J, find an assignment $\mathbf{x}$ of experts such that the following is maximized:

$$F(\mathbf{x}) = \lambda Cov(J \mid \mathbf{x}) - Cost_L(\mathbf{x}). \tag{4}$$

QUBO Formulation Sketch. Let $\mathbf{y} = \mathbf{s} \mid\mid \mathbf{x}$, be the $(m+n)$-size solution vector we described above. Now let $\mathbf{c}$ be another $(m+n)$ vector such that $c_i = \lambda$ if $i \leq m$ *and* skill $i \in J$, $c_i = -\kappa_{i-m}$ if $i > m$; recall that κ_i is the cost of hiring expert i (see Sec. 3). Then Coverage-Linear-Cost can be expressed as:

$$\begin{aligned} \text{maximize} \quad & \mathbf{c}^T\mathbf{y}, \\ \text{such that} \quad & s_j - \sum_{i=1}^{n} E(i,j) \cdot x_i \leq 0 \quad \text{for all } 1 \leq j \leq m, \text{ and} \\ & s_i, x_i \in \{0,1\}. \end{aligned}$$

We create penalty matrices P_1 and P_2 to capture the constraints in the LP. Then, the $(m+n) \times (m+n)$ square matrix $Q = -\text{diag}(\mathbf{c}) - P_1 + P_2$ has the property that minimizing $\mathbf{y}^T Q \mathbf{y}$ corresponds to maximizing $F(\mathbf{x}) = \lambda Cov(J \mid \mathbf{x}) - Cost_L(\mathbf{x})$.

4.4 Coverage-Graph-Cost

Problem 3 (Coverage-Graph-Cost). Given a set of n experts $\mathcal{X} = \{X_1, \ldots, X_n\}$ with a corresponding distance function $d(\cdot, \cdot)$ between any pair of experts, and a task J, find an assignment $\mathbf{x}$ of experts such that we maximize:

$$F(\mathbf{x}) = \lambda Cov(J \mid \mathbf{x}) - Cost_G(\mathbf{x}). \tag{5}$$

QUBO Formulation Sketch. We consider the following constrained linear program that encodes the Coverage-Graph-Cost problem:

$$\begin{aligned} \text{maximize} \quad & \lambda \cdot \sum_{i=1}^{n} s_i - \sum_{(i,j)} d(i,j) \cdot (x_i x_j) \\ \text{such that} \quad & s_j - \sum_{i=1}^{n} E(i,j) \cdot x_i \leq 0 \quad \text{for all } 1 \leq j \leq m, \text{ and} \\ & s_i, x_i \in \{0,1\}. \end{aligned}$$

For the QUBO formulation we need the solution vector $\mathbf{y}$, we defined above. We also need the $(m+n)$ vector $\mathbf{c} = (c_1, \ldots, c_{(m+n)})$, such that $c_i = \lambda$ if $i \leq m$ *and* skill $i \in J$, and $c_i = 0$ otherwise. Then we compute the $(n \times n)$ matrix D of pairwise distances such that $D(i,j) = d(X_i, X_j)$ and add it to the lower-right $(n \times n)$ submatrix of $\text{diag}(\mathbf{c})$ to obtain $\hat{D} = \text{diag}(\mathbf{c}) + \begin{bmatrix} \mathbf{0}_{m\times m} & \mathbf{0}_{m\times n} \\ \mathbf{0}_{n\times m} & D_{n\times n} \end{bmatrix}$ Now, $F(\mathbf{x}) = \mathbf{y}^T \hat{D} \mathbf{y}$ encodes the Coverage-Graph-Cost objective. We create penalty matrices P_1, P_2 to capture the LP constraints; the $(m+n) \times (m+n)$ square matrix $Q = -\hat{D} - P_1 + P_2$ provides a complete QUBO formulation of Coverage-Graph-Cost; that is, minimizing $\mathbf{y}^T Q \mathbf{y}$ corresponds to maximizing $F(\mathbf{x}) = \lambda Cov(J \mid \mathbf{x}) - Cost_G(\mathbf{x})$.

All three TeamFormation problem variants are hard to solve and approximation and heuristic algorithms exist in the literature [14,17,25].

5 Solving TeamFormation Problems

In this section, we describe two different general-purpose methods that leverage the QUBO formulation to solve TeamFormation problems.

5.1 QUBO Solver

We use a QUBO solver implemented by Gurobi [12]. The solver takes the Q-matrix corresponding to a QUBO problem as input, and applies mixed-integer programming methods with specialized heuristics to solve the QUBO instance. We use Gurobi's QUBO solver with the Q-matrix corresponding to the TeamFormation problems, and refer to this method as `Qsolver`.

5.2 Graph Neural Networks

Combinatorial optimization problems are formulated as QUBO [30] and represented as a graph $G = (V, E)$, where each vertex $i \in V$ corresponds to a binary decision variable $y_i \in \{0, 1\}$. The objective function is defined by a Hamiltonian $\mathbb{H}(\mathbf{y})$, which represents the system's energy. The binary state y_i is relaxed into a continuous representation $\pi_i \in [0, 1]$, allowing gradient-based optimization to be applied. The architecture employs multiple layers of message-passing neural networks to iteratively update node representations. At each layer l, the hidden state $\pi_i^{(l)}$ of node i is updated based on its current state and information aggregated from its neighboring nodes $\mathcal{N}(i)$: $\pi_i^{(l+1)} = \sigma\left(W^{(l)}\pi_i^{(l)} + \sum_{j\in\mathcal{N}(i)} W^{(l)}\pi_j^{(l)} + \mathbf{w_0}^{(l)}\right)$ where $W^{(l)}$ and $\mathbf{w_0}^{(l)}$ are the weight matrix and bias vector for layer l, and σ is a nonlinear activation function. The loss function is based on the relaxed Hamiltonian $\mathbb{H}(\pi)$, such that the network is trained to minimize the energy. After training, the continuous node states π_i are projected back to binary y_i, yielding a feasible solution to the original combinatorial optimization problem.

GNNs for TeamFormation. We perform unsupervised node classification using a GNN to solve the QUBO formulation corresponding to TeamFormation. Given the Q matrix that encodes a problem, the goal is to find the $(m+n)$-size solution vector $\mathbf{y} = \mathbf{s}||\mathbf{x}$ that minimizes $\mathbf{y}^T Q\mathbf{y}$, with $\mathbf{x} = (y_{m+1}, \ldots, y_{m+n})$ being the desired solution assignment to the TeamFormation problem.

Graph Creation. We create a graph $G = (V, E)$, where each vertex $i \in V$ corresponds to a binary decision variable $y_i \in \{0, 1\}$; vertices $(1, \ldots, m)$ correspond to the set of all skills, and vertices $(m+1, \ldots, m+n)$ correspond to the experts in the TeamFormation problem instance. For every skill each expert has, we create an unweighted edge in G between the corresponding expert and skill vertices, i.e. $E = \{(i, j) : s_i \in X_j\}$. For Coverage-Graph-Cost, we add weighted edges between expert vertices to encode the pairwise network coordination costs.

Loss Function and Regularization. Since $\mathbf{y}^T Q\mathbf{y}$ is not differentiable and cannot be used as such within the GNN training process, we follow the approach of Schuetz et al. [30] to relax each binary variable $y_i \in \{0, 1\}$ such that $y_i \to \pi_i \in [0, 1]$, where these π_i can be viewed as selection probabilities, i.e. small π_i implies y_i is not selected, and large π_i implies y_i is selected. We then generate the following differentiable loss function used for backpropagation:

$$\mathcal{L}(\pi) = \sum_{i,j} \pi_i \, Q_{ij} \, \pi_j + \alpha \cdot \sum_i \pi_i \, (1 - \pi_i).$$

We include the regularization term $\alpha \cdot \sum_i \pi_i \, (1 - \pi_i)$ to encourage the GNN to converge to binary solutions, where α is a tunable hyperparameter.

We randomly initialize node embeddings for each of the expert and skill nodes, where the dimension of the embeddings is given by the hyperparameter d_0. We denote the set of $(m + n)$ embeddings by $H^{(0)} = H_S^{(0)} \,||\, H_{\mathcal{X}}^{(0)}$, where $||$ represents concatenation of the m skill embeddings and n expert embeddings.

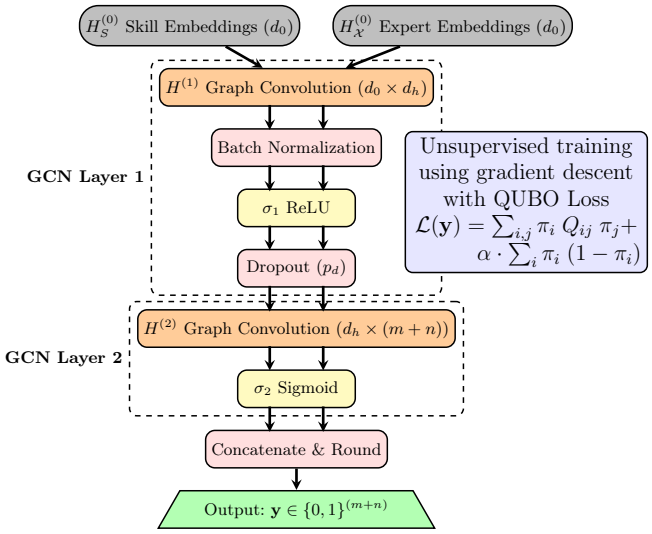

Fig. 2: `QUBO-GNN` model architecture for solving TeamFormation problems.

Graph Convolution. Vertices in G represent skills *and* experts, and thus we have two different types of edges: between experts and skills, and between two experts. To ensure message-passing during GNN training occurs over valid edge types, we adopt a two-layer (heterogeneous) graph convolution network (GCN) architecture, with forward propagation given by $H^{(1)} = \sigma_1\left(\sum_{r\in\mathcal{R}} \Theta_r^0 H^{(0)}\right)$ and $H^{(2)} = \sigma_2\left(\sum_{r\in\mathcal{R}} \Theta_r^1 H^{(1)}\right)$, where $\mathcal{R}$ is the set of different edge types. $H^{(0)}$ represents the input node embeddings of size d_0, and $H^{(1)}$ and $H^{(2)}$ are the hidden and output layer representations of sizes d_h and $(m+n)$, respectively. Θ_r^0 and Θ_r^1 are trainable weight matrices specific to r, allowing the GNN to learn different transformations per edge type; σ_1, σ_2 are non-linear activation functions, applied element-wise; we use ReLU for σ_1 and a sigmoid for σ_2.

We add batch normalization after the first graph convolutional layer to normalize activations and stabilize training. We also introduce dropout after the ReLU activation by randomly setting p_d fraction of neurons in the GNN to zero.

We call our method `QUBO-GNN` and visualize the model architecture in Figure 2. `QUBO-GNN` is parametrized by several hyperparameters; Table 1 provides a summary of the hyperparameters of the `QUBO-GNN` model, and heuristic ranges of values to grid-search. The model hyperparameters d_0, d_h, p_d, α and β can be set heuristically or optimized in an outer-loop using grid-search.

Capturing problem constraints effectively in a QUBO formulation requires the selection of suitable scalar penalties p_1, p_2. In practice, we observed for our problems that the unbalanced penalization scheme yields good solutions for a wide range of values of p_1, p_2. However, to enable convergence to better near-optimal solutions we implement a grid search for p_1, p_2 over the range of heuristic values shown in Table 1.

Table 1: Description of `QUBO-GNN` model parameters.

Parameter	Description	Heuristic range
p_1	QUBO penalty 1	$[10^{-1}, 10^2]$
p_2	QUBO penalty 2	$[10^{-1}, 10^2]$
λ	Normalizing coefficient	$[1, 10^2]$
d_0	Size of node embeddings	$[(m+n)^{1/2}, (m+n)/2]$
d_h	Size of hidden layer	$[(m+n)^{1/2}, (m+n)/2]$
p_d	Dropout probability	$[0.1, 0.3]$
α	Binary regularization weight	$[1, 10]$
β	Learning rate	$[10^{-4}, 10^{-2}]$

Projection Rounding and Output. At the end of unsupervised training, the σ_2 sigmoid activation layer outputs probabilities π_i associated with each node which we can view as soft assignments. We apply a simple rounding scheme: $y_i = \text{int}(\pi_i)$ to project these probabilities π_i back to binary assignments $y_i \in \{0, 1\}$.

6 Experimental Analysis

6.1 Experimental Setup

Datasets. We evaluate our methods on several real-world datasets also used in past team formation papers: *Freelancer*, *IMDB*, *Bbsm* [1,24,25,35]. We follow the method of [2] and create social graphs with expert coordination costs for our datasets. We provide summary statistics of the datasets in Table 2. Detailed descriptions and pre-processing steps of each dataset are available in the supplementary material.

Table 2: Summary statistics of our datasets.

Dataset	Experts	Tasks	Skills	Skills/ expert	Skills/ task	Average path length	Average degree
Freelancer-1	50	250	50	2.2	4.3	2.6	4.5
Freelancer-2	150	250	50	2.2	4.4	2.4	10.4
IMDB-1	200	300	23	3.3	5.0	3.0	0.4
IMDB-2	400	300	23	3.8	5.3	7.1	0.9
IMDB-3	1000	300	25	4.5	5.2	6.2	2.3
Bbsm-1	250	300	75	12.5	5.5	5.9	1.9
Bbsm-2	500	300	75	13.0	5.5	2.6	9.4
Bbsm-3	1000	300	75	13.1	5.5	2.6	13.3

Baselines. For each of the TeamFormation variants, we evaluate the performance of `QUBO-GNN` and `Qsolver` against some problem-specific baselines, which have the same principles across problem variants. We describe those below.

`Greedy`**:** For Max-k-Cover the `Greedy` baseline iteratively picks the expert with the maximum marginal skill coverage. For Coverage-Linear-Cost, `Greedy` implements the `Cost-Scaled Greedy` algorithm introduced by Nikolakaki et al. [25]. For the Coverage-Graph-Cost problem, `Greedy` picks the expert that maximizes the ratio of coverage over coordination cost at each iteration.

`Topk`**:** This is an objective-agnostic algorithm that ranks the experts based on their Jaccard similarity with the input task and then picks the top-k most similar experts, where k is determined by the size of the `Greedy` (or `Qsolver`) solution.

Implementation Details. We used single-process implementations on a 14-core 2.4 GHz Intel Xeon E5-2680 processor for all our experiments. We implement our `QUBO-GNN` architecture in Python using PyTorch [26] and Deep Graph Library[36], and fine-tune model hyperparameters using grid search. For each dataset, we train separate `QUBO-GNN` models for up to 100 different tasks. For the normalizing coefficient we set $\lambda = 50$, which yields a reasonable balance between weighting coverage and cost for our TeamFormation variants. To aid reproducibility, we report the full set of model parameters used in the supplement, and make our code [3] available online.

6.2 Quantitative Comparison

We evaluate our algorithms against the baselines with respect to our overall objective (and the corresponding coverage, size and cost). Due to space constraints, we only show detailed results for Coverage-Linear-Cost, and provide experimental results for Max-k-Cover and Coverage-Graph-Cost in the supplement. Note that the general experimental patterns observed were similar across all three TeamFormation variants.

We observe that `Qsolver` has the best performance for all datasets, and consequently analyze the objective of our methods by first normalizing by the corresponding `Qsolver` objective and then taking the mean across all training tasks. We denote the normalized objective by $\hat{F}(\mathbf{x})$.

Aggregate Performance Evaluation. Figure 3 presents the mean $\hat{F}(\mathbf{x})$ across all training tasks returned by `QUBO-GNN`, `Qsolver`, `Greedy`, and `Topk` across our datasets. We observe that `Qsolver` consistently achieves the highest normalized mean objective values (i.e., values equal to 1): it outperforms the other methods across all datasets for all three TeamFormation variants. We observe that `Greedy` performs slightly worse than `Qsolver`, and `Topk` consistently has the lowest $\hat{F}(\mathbf{x})$. For most datasets, `QUBO-GNN` achieves solutions with objective values that are comparable (but slightly worse) than `Qsolver`. Overall, this is expected as `Qsolver` finds the optimal solution for the same problem that `QUBO-GNN`tries to solve. Moreover, the success of both QUBO-based algorithmic

[3] https://github.com/kvombatkere/Team-Formation-QUBO

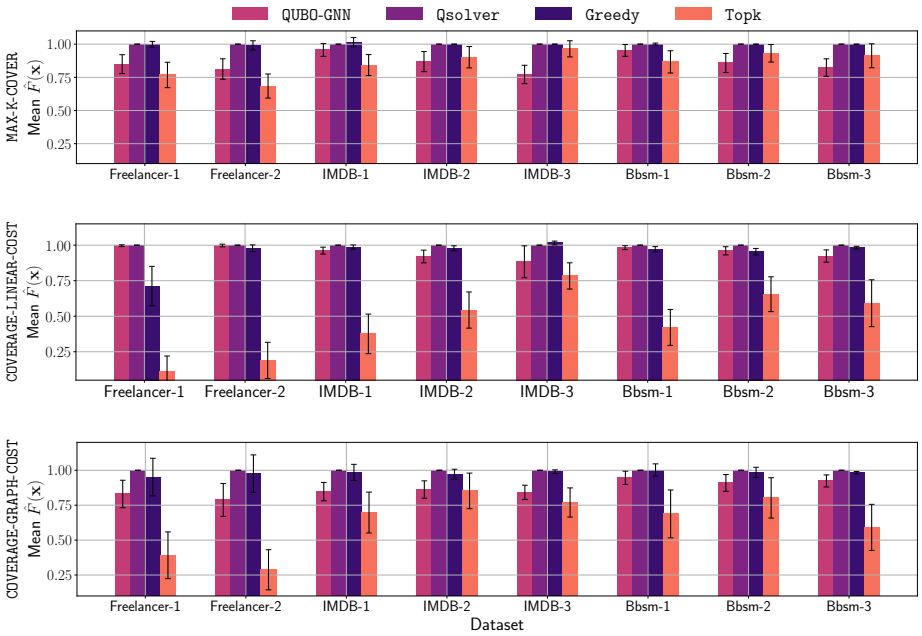

Fig. 3: Bar plots showing the mean `Qsolver`-normalized objective, $\hat{F}(\mathbf{x})$ of `QUBO-GNN`, `Qsolver`, `Greedy` and `Topk`, across all training task instances for all three TeamFormation variants.

solutions demonstrate that our QUBO formulation is appropriate for solving the original team formation problem.

We use $\overline{Cov} = \frac{1}{t}\sum_{i=1}^{t} Cov(J_i|\mathbf{x})$ to denote the mean coverage, and $\overline{z} = \frac{1}{t}\sum_{i=1}^{t} z_i$ to denote the mean solution size, across training tasks $J_1, \ldots, J_t$. We observe from Table 3 that all three methods find solutions yielding high coverages for *IMDB* and *Bbsm*. However, `QUBO-GNN` and `Qsolver` often find assignments with a larger solution size (and larger cost) than `Greedy`. These assignments – particularly for *Freelancer* – lead to higher coverages resulting in superior objective values. This tradeoff highlights the ability of the QUBO formulation to balance cost and team effectiveness better than greedy approaches. Finally, even though `Greedy` was the fastest algorithm in terms of running time, `QUBO-GNN` and `Qsolver` converged to good solutions within a few seconds, even for the largest datasets (i.e. *IMDB-3* and *Bbsm-3*).

Individual Task Evaluation. Figure 4 presents a scatter plot of the objectives F for each training task instance (for each dataset) for Coverage-Linear-Cost; the tasks are sorted in decreasing order of F. We conclude that `QUBO-GNN` is competitive with `Greedy` and even outperforms it in multiple cases, demonstrating that GNN-based approaches can achieve strong performance even with-

Table 3: Mean task coverage, $\overline{Cov}$ and solution size, $\overline{z}$ of `QUBO-GNN`, `Qsolver` and `Greedy` across all training task instances for COVERAGE-LINEAR-COST.

Dataset	Mean Task Coverage, $\overline{Cov}$			Mean Solution Size, $\overline{z}$		
	`QUBO-GNN`	`Qsolver`	`Greedy`	`QUBO-GNN`	`Qsolver`	`Greedy`
Freelancer-1	0.88	0.89	0.48	2.8	2.9	1.4
Freelancer-2	0.98	0.98	0.92	3.2	3.2	2.9
IMDB-1	0.99	1.00	0.99	2.3	2.4	2.1
IMDB-2	0.98	1.00	1.00	2.5	2.3	1.8
IMDB-3	0.88	1.00	1.00	2.5	3.2	1.2
Bbsm-1	1.00	1.00	1.00	3.1	2.7	2.0
Bbsm-2	0.97	1.00	1.00	2.8	2.6	1.6
Bbsm-3	0.98	1.00	1.00	1.8	4.2	1.7

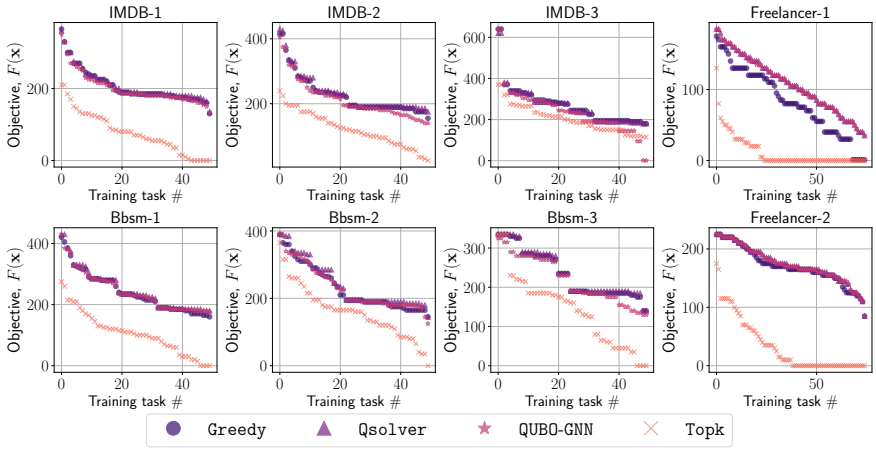

Fig. 4: Comparative performance of `QUBO-GNN`, `Qsolver`, `Greedy` and `Topk`, across individual training tasks, in terms of the sorted objective $F()$.

out explicit heuristic tuning. Furthermore, for the *Freelancer-1* dataset, both `QUBO-GNN` and `Qsolver` outperform `Greedy` by over 30%. In our experiments, `QUBO-GNN` consistently selects experts based on their skill relevance and almost never violates the constraints of the underlying LPs; thus `QUBO-GNN` can identify well-balanced teams without the need for additional filtering mechanisms.

Investigating Node Embeddings. Figure 5 shows two scatter plots of skill and expert node embeddings projected to 2D using t-SNE [20]. Each set of embeddings was generated by a `QUBO-GNN` model for COVERAGE-LINEAR-COST after training on a task from *Freelancer-1*. This figure is representative of the patterns observed in node embeddings for all instances of TEAMFORMATION. We observe that the embeddings corresponding to task skills and relevant experts (i.e. experts who have at least one required skill) differentiate themselves

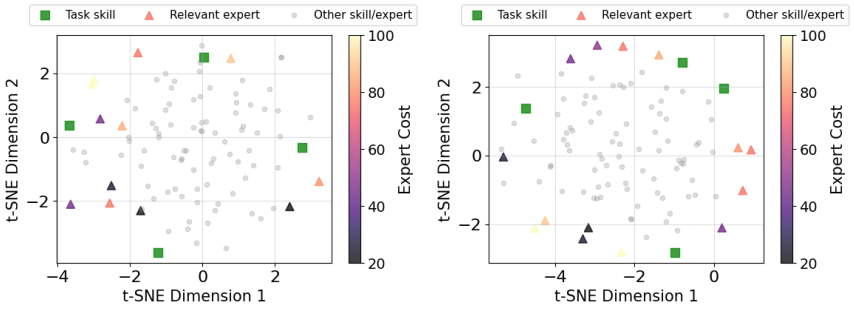

Fig. 5: Scatter plots of skill and expert node embeddings projected to 2D using t-SNE. Each set of embeddings was generated by a Coverage-Linear-Cost `QUBO-GNN` model after training on a task from *Freelancer-1*.

from other skills/experts by forming an outer perimeter and occupying distinct regions of the plot. Experts with similar skills often cluster together, and their embeddings are often similar to those of their common skill(s). This indicates that `QUBO-GNN` successfully learns representations between skills and experts and is able to correctly identify sets of experts that are important for covering a task.

6.3 Transfer Learning

Intuitively, we expect a `QUBO-GNN` model $\mathcal{M}$ to learn node embeddings that result in good assignments for new tasks that are similar to the tasks $\mathcal{M}$ was trained on. Consider t `QUBO-GNN` models that have been trained on their corresponding tasks $J_1, \ldots, J_t$. Given an unseen task J', we first compute the Jaccard similarity of J' with each of $J_1, \ldots, J_t$, and select the `QUBO-GNN` model $\mathcal{M}'$ corresponding to the task that is most similar to J'. Next, we initialize the new TeamFormation instance for J' with the pre-trained node embeddings corresponding to $\mathcal{M}'$, and use model $\mathcal{M}'$ to perform a single forward pass to obtain an assignment $\mathbf{x}$ for J'. We refer to this method `QUBO-GNN-Sim`. For each dataset, we evaluate it against the following two baselines on 100 new tasks.

`QUBO-GNN-Rand`: We use a random sample of 3 pre-trained `QUBO-GNN` models. We perform a single forward pass using each model and select the assignment $\mathbf{x}$ that yields the best objective.

`Qsolver-Sim`: Given a new task J', we use the solution of `Qsolver` corresponding to the task (from $J_1, \ldots, J_t$) that has the highest Jaccard similarity to J' to compute the objective for J'.

Figure 6 shows a scatter plot of sorted objectives F of `QUBO-GNN-Sim` and the two baselines for 100 new tasks across each dataset for Coverage-Linear-Cost. The results for the other two problems are shown in the supplement.

We note that `QUBO-GNN-Sim` outperforms `Qsolver-Sim` for *Freelancer* and *IMDB-3* and *Bbsm-3*, while the two methods have comparable performance for *IMDB-1*, *IMDB-2*, *Bbsm-1* and *Bbsm-2*. `QUBO-GNN-Rand` has poor overall

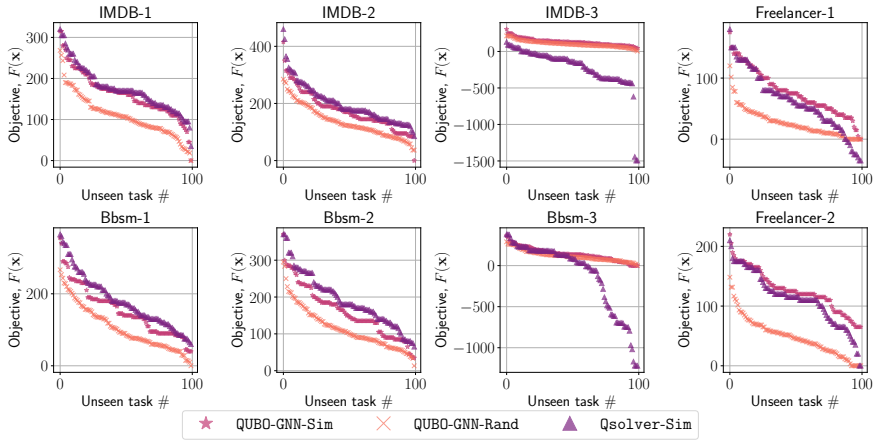

Fig. 6: Evaluation of transfer learning on 100 new tasks across each dataset for Coverage-Linear-Cost in terms of the sorted objective $F()$.

performance. This was expected, since using node embeddings of a random task would not necessarily aid solving of a new problem. `QUBO-GNN-Sim` often finds solutions with high coverages, indicating that the learned node embeddings from the original model can capture useful relationships between skills and experts that can then be leveraged for other tasks. We also find, intuitively, that the efficacy of using node embeddings from a `QUBO-GNN` model (trained on task J_i) for a new task J_j, correlates strongly with the Jaccard similarity of J_i and J_j.

7 Conclusions and Future Work

In this paper, we introduced a unified QUBO-based framework for the general TeamFormation problem, enabling a versatile algorithmic approach across problem variants that balance task coverage with expert costs. We then evaluated our framework using both a QUBO solver, and a GNN method that maximizes the TeamFormation objective by optimizing a QUBO-derived loss function. In our experimental evaluation on real-world datasets from diverse domains, we demonstrated that our methods consistently find expert assignments with high objectives, often outperforming combinatorial baselines designed specifically for certain problem variants. Finally, we highlighted the potential for transfer learning, where learned representations from one problem instance can be effectively used to solve other related instances.

Future Work. Finding optimal penalty parameters for our QUBO formulations is challenging, consequently opening up an avenue for future work on efficient methods to tune these penalties. A natural extension of our work could consider multiple input tasks and explore more (complex) expert cost functions based on workload, team diameter, etc. Finally, there is scope for fine-tuning the `QUBO-GNN` model architecture to improve performance.

Acknowledgment. This work was supported by gifts from Microsoft and Google.

References

1. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., Leonardi, S.: Power in unity: forming teams in large-scale community systems. In: ACM Conference on Information and Knowledge Management, CIKM. pp. 599–608 (2010)
2. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., Leonardi, S.: Online team formation in social networks. In: Proceedings of the 21st international conference on World Wide Web. pp. 839–848 (2012)
3. Ayodele, M.: Penalty weights in qubo formulations: Permutation problems. In: European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoStar). pp. 159–174. Springer (2022)
4. Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940 (2016)
5. Benz, D., Hotho, A., Jäschke, R., Krause, B., Mitzlaff, F., Schmitz, C., Stumme, G.: The social bookmark and publication management system BibSonomy. The VLDB Journal **19**(6), 849–875 (Dec 2010)
6. Cappart, Q., Chételat, D., Khalil, E.B., Lodi, A., Morris, C., Veličković, P.: Combinatorial optimization and reasoning with graph neural networks. Journal of Machine Learning Research **24**(130), 1–61 (2023)
7. Caramanis, C., Fotakis, D., Kalavasis, A., Kontonis, V., Tzamos, C.: Optimizing solution-samplers for combinatorial problems: The landscape of policy-gradient methods. arXiv preprint arXiv:2310.05309 (2023)
8. Dashti, A., Samet, S., Fani, H.: Effective neural team formation via negative samples. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. pp. 3908–3912 (2022)
9. Feige, U., Mirrokni, V.S., Vondrák, J.: Maximizing non-monotone submodular functions. SIAM Journal on Computing **40**(4), 1133–1153 (2011)
10. Feldman, M.: Guess free maximization of submodular and linear sums. Algorithmica **83**(3), 853–878 (2021)
11. Glover, F., Kochenberger, G., Du, Y.: Quantum bridge analytics i: a tutorial on formulating and using qubo models. 4or **17**(4), 335–371 (2019)
12. Gurobi Optimization, LLC: Gurobi OptiMods (2023), `https://github.com/Gurobi/gurobi-optimods`
13. Hamidi Rad, R., Fani, H., Bagheri, E., Kargar, M., Srivastava, D., Szlichta, J.: A variational neural architecture for skill-based team formation. ACM Transactions on Information Systems **42**(1), 1–28 (2023)
14. Harshaw, C., Feldman, M., Ward, J., Karbasi, A.: Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In: International Conference on Machine Learning. pp. 2634–2643. PMLR (2019)
15. Hochbaum, D.S.: Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. Approximation algorithms for NP-hard problems pp. 94–143 (1997)
16. Kargar, M., An, A., Zihayat, M.: Efficient bi-objective team formation in social networks. In: ECML PKDD (2012)
17. Krause, A., Golovin, D.: Submodular function maximization. Tractability **3**(71-104), 3 (2014)

18. Lappas, T., Liu, K., Terzi, E.: Finding a team of experts in social networks. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 467–476 (2009)
19. Lucas, A.: Ising formulations of np problems. Frontiers in physics **2**, 74887 (2014)
20. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)
21. Mehta, V., Jin, F., Michielsen, K., De Raedt, H.: On the hardness of quadratic unconstrained binary optimization problems. Frontiers in Physics **10**, 956882 (2022)
22. Montanez-Barrera, A., Willsch, D., Maldonado-Romo, A., Michielsen, K.: Unbalanced penalization: A new approach to encode inequality constraints of combinatorial problems for quantum optimization algorithms. arXiv preprint arXiv:2211.13914 (2022)
23. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: Analysis of approximations for maximizing submodular set functions. Mathematical programming **14**, 265–294 (1978)
24. Nikolakaki, S.M., Cai, M., Terzi, E.: Finding teams that balance expert load and task coverage. CoRR **abs/2011.04428** (2020)
25. Nikolakaki, S.M., Ene, A., Terzi, E.: An efficient framework for balancing submodularity and cost. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 1256–1266 (2021)
26. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
27. Quintero, R.A., Zuluaga, L.F.: Qubo formulations of combinatorial optimization problems for quantum computing devices. In: Encyclopedia of Optimization, pp. 1–13. Springer (2022)
28. Sapienza, A., Goyal, P., Ferrara, E.: Deep neural networks for optimal team composition. Frontiers in big Data **2**, 14 (2019)
29. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE transactions on neural networks **20**(1), 61–80 (2008)
30. Schuetz, M.J., Brubaker, J.K., Katzgraber, H.G.: Combinatorial optimization with physics-inspired graph neural networks. Nature Machine Intelligence **4**(4), 367–377 (2022)
31. Shafi, Z., Miller, B.A., Eliassi-Rad, T., Caceres, R.S.: Graph-scp: Accelerating set cover problems with graph neural networks. preprint arXiv:2310.07979 (2023)
32. Skianis, K., Nikolentzos, G., Limnios, S., Vazirgiannis, M.: Rep the set: Neural networks for learning set representations. In: International conference on artificial intelligence and statistics. pp. 1410–1420. PMLR (2020)
33. Verma, A., Lewis, M.: Penalty and partitioning techniques to improve performance of qubo solvers. Discrete Optimization **44**, 100594 (2022)
34. Vombatkere, K., Gionis, A., Terzi, E.: Forming coordinated teams that balance task coverage and expert workload. Data Mining and Knowledge Discovery **39**(3), 1–37 (2025)
35. Vombatkere, K., Terzi, E.: Balancing task coverage and expert workload in team formation. In: Proceedings of the 2023 SIAM International Conference on Data Mining (SDM). pp. 640–648. SIAM (2023)
36. Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., et al.: Deep graph library: A graph-centric, highly-performant package for graph neural networks. arXiv preprint arXiv:1909.01315 (2019)
37. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. Advances in neural information processing systems **30** (2017)