

# A Branch-and-Bound Algorithm for Team Formation on Social Networks

Nihal Berktaş

Department of Industrial Engineering, Bilkent University, Ankara, Turkey, [nihal.berktas@bilkent.edu.tr](mailto:nihal.berktas@bilkent.edu.tr)

Hande Yaman

Present address: ORSTAT, Faculty of Economics and Business, KU Leuven, Belgium, [handeyaman@kuleuven.be](mailto:handeyaman@kuleuven.be)

## Abstract

Team Formation Problem (TFP) aims to construct a capable team that can communicate and collaborate effectively. The cost of communication is quantified using the proximity of the potential members in a social network. We study TFP with two measures for communication effectiveness; namely, we minimize the sum of communication costs and we impose an upper bound on the largest communication cost. This problem can be formulated as a constrained quadratic set covering problem. Our experiments show that a general purpose solver is capable of solving small and medium size instances to optimality. We propose a branch and bound algorithm to solve larger sizes: We reformulate the problem and relax it in such a way that it decomposes into a series of linear set covering problems and we impose the relaxed constraints through branching. Our computational experiments show that the algorithm is capable of solving large size instances, which are intractable for the solver. *keywords*: team formation problem, quadratic set covering, branch and bound, reformulation

## 1 Introduction

The complexity of products and services in today's world requires various skills, knowledge and experience from different fields while the pace of consumption demands agility in the production and development phases. To be able to meet these requirements, people are working in teams both physically and virtually in various organizations such as governments, NGOs, universities, hospitals and business firms. The quality of the work done depends on the technical capabilities of the team members as well as the effectiveness of communication among them. In the studies investigating the factors affecting the success of teams, communication has been considered as one of the key factors if not the most important one (Hoegl and Gemuenden, 2001), especially in virtual teams (Jones, 2005).

In addition to regular organizations that build physical and virtual teams for projects, there is a new concept of outsourcing called "Team as a Service". The companies that use this model build a team according to the needs of a given project and provide managerial service throughout. The concept is claimed to provide the agility that companies need in today's fast-moving market as it reduces the burden on the core permanent employees by offering a self-sufficient team (Centric Digital, 2016).

Motivated by this new concept of Team as a Service, we are interested in the team formation problem, which is the problem of selecting a group of people from a candidate set so that they work together on a given task

that requires some technical skills. Our aim is to build a team whose members can collaborate effectively and we do this by minimizing their communication cost.

In the operations research literature, the team formation problem has been studied in different contexts. The studies of Zakarian and Kusiak (1999) on product design, Boon and Sierksma (2003) on sports teams and Agustín-Blas et al. (2011) on teaching groups are some examples in which the objective is to maximize the technical capability or the knowledge of the team. In the studies of Chen and Lin (2004), Fitzpatrick and Askin (2005) and Zhang and Zhang (2013), communication is taken into consideration using the personal characteristics of the team members. Well-known personality tests such as Myers-Briggs and Kolbe Conative are used to measure the effectiveness of communication. Baykasoglu et al. (2007) incorporate communication by specifying people who do not prefer to be in the same project. Gutiérrez et al. (2016) model interpersonal relations via the sociometric matrix, which consists of -1, 0 and 1's representing the negative, neutral and positive relations, respectively. Another method to incorporate communication into the problem, the one chosen in this study, is via a social network of individuals. To the best of our knowledge in the operations research literature, the study by Wi et al. (2009) is the first one to use social networks for team formation. The authors form a network using fuzzy familiarity scores among candidates via collaboration data and formulate a nonlinear program whose objective is a weighted sum of performance, familiarity and size of the team. More recently, Farasat and Nikolaev (2016) use edge, 2-star, 3-star and triangle network structures to measure the collaborative strength of the team. The objective is to maximize the weighted sum of structures in multiple teams and the skills of people are not considered. The solution techniques suggested in the above studies are either not designed for real size data or they are heuristic approaches.

The team formation problems where a social network is considered are mainly studied in the knowledge discovery and data mining field, initiated by the work of Lappas et al. (2009) and followed by many others. This line of work is motivated by the existence of numerous online social networks and the advances in social network analysis. It utilizes a social network in which the edge weights are considered as measures of the effort required for candidates to communicate as team members. Clearly, a lower weight for edge  $\{i, j\}$  implies that candidates  $i$  and  $j$  can collaborate more effectively. Lappas et al. (2009) study two variants of the problem with different communication cost functions. The first is the diameter of the team, which is the largest distance between any pair of team members where the distance between two people is taken as the shortest path weight in the network. The second function is the cost of a minimum-cost Steiner tree that spans the team members. Following this study other functions are defined and used for the problem. The studies of Kargar and An (2011), Kargar et al. (2012) and Bhowmik et al. (2014) are among the ones that define the communication cost of the team as the sum of distances, which is the sum of the shortest path lengths between all pairs of team members. Kargar and An (2011) define leader distance as the sum of shortest path lengths between the leader and the person chosen for each required skill. Given a team, the bottleneck cost is defined by Majumder et al. (2012) as the maximum edge weight in a tree that minimizes this and that spans the team members. Dorn and Dustdar (2010) and Gajewar and Sarma (2012), on the other hand, use communication cost functions that are related to the density of the team's subgraph.

We adopt the problem definition of Lappas et al. (2009) and use a social network to quantify and minimize the communication cost. The technical capability of the team is ensured using a binary skill matrix built by considering minimum expertise levels. We propose to minimize the sum of distances and to impose an upper bound on the diameter. We derive a mixed integer programming formulation for this new problem and test it

using a large set of instances. We observe that small and medium size instances can be solved using a general purpose solver but memory problems occur for large instances. We present a novel branch and bound algorithm which is very effective in solving these instances.

The remaining part of the paper is organized as follows: In the next section we formally define the team formation problem and provide quadratic and linear mathematical models. We present our branch and bound algorithm in Section 3. In Section 4, we first introduce our datasets and explain our instance generation method. Then we present the results of an extensive computational study. We conclude in Section 5.

## 2 Problem Definition and Mathematical Models

In this section we formally define the team formation problem, explain how the communication costs are computed and provide mathematical models.

Let  $K$  be the set of required skills for a given task and let  $N$  be the set of candidates. We assume that the skills of the candidates are known. We need to select team members such that for each skill there is at least one person in the team having that skill. Such teams are called “capable teams”. An undirected collaboration network of the candidates,  $G = (N, E)$ , is given. In a collaboration network, two people (nodes) are connected by an edge if they have collaborated before. Edge  $\{i, j\}$  has weight  $c_{ij}$ . These weights are commonly calculated in the following way: Let  $i$  and  $j$  be two people and  $P_i$  and  $P_j$  be the sets of projects they have taken part in, respectively. Then  $|P_i \cap P_j|$  is the number of their collaborations and the weight of edge  $\{i, j\}$  is taken as  $1 - (|P_i \cap P_j| / |P_i \cup P_j|)$  which is the Jaccard metric, a well-known dissimilarity measure introduced by Jaccard (1912). The Jaccard distance between any two people with no collaboration equals to one. Instead of taking the distance between all such unconnected pairs as one, Lappas et al. (2009) and the others use the shortest path distances among these pairs. This method differentiates the unconnected pairs who have neighbours that collaborated often from the ones who have distant connections. We follow the same approach and define the cost of communication between  $i$  and  $j$ , denoted by  $p_{ij}$ , to be equal to  $c_{ij}$  if  $P_i \cap P_j \neq \emptyset$ , to be equal to the weight of the shortest path between  $i$  and  $j$  if  $P_i \cap P_j = \emptyset$  and to be equal to a sufficiently large number if there is no path between them. By construction, all communication costs  $p_{ij}$ 's are nonnegative.

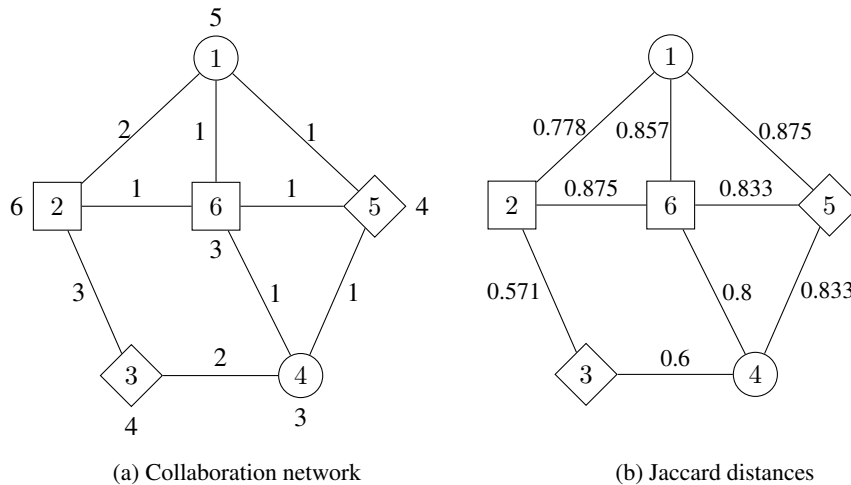


Figure 1: Collaboration network and corresponding Jaccard distances

	1	2	3	4	5	6
1	0	0.778	1.349	1.657	0.875	0.857
2	-	0	0.571	1.171	1.653	0.875
3	-	-	0	0.6	1.433	1.4
4	-	-	-	0	0.833	0.8
5	-	-	-	-	0	0.833
6	-	-	-	-	-	0

Table 1: Communication cost matrix for the people in the collaboration network

Before moving on to the problem definition we demonstrate the cost calculation procedure on a small example. In Figure 1, on the left, we have a collaboration network where the nodes represent people and the shapes indicate the skill they have. The number next to each node is the total number of projects that the person has worked on. The number on each edge shows the number of collaborations of the people corresponding to the end nodes of the edge. The numbers on edges of the network on the right are the Jaccard distances calculated from the collaboration data. Then calculating the shortest paths we get the distance (communication cost) matrix in Table 1.

Under the setting given above, team formation problem (TFP) is defined as finding a capable team with minimum communication cost. With communication costs computed as described, minimizing the sum of the distances amounts to maximizing the average familiarity of the team. There are empirical studies in the literature indicating positive effects of team familiarity on the performance of teams. The results of Huckman et al. (2009)’s study on a software service company indicate a positive and significant relation between team familiarity and operational performance. Analyzing software development teams of a telecommunications firm, Espinosa et al. (2007) find that team familiarity is more beneficial when coordination is more challenging due to team size or dispersion. The study of Avgerinos and Gokpinar (2016) on productivity of surgical teams also shows that the benefit of familiarity increases as the task gets more complex. Moreover, the performance analysis in the study suggests that the bottleneck pair, i.e., the pair with the lowest familiarity, significantly reduces team productivity. In terms of the communication cost measures, the least familiar pair in a team amounts to the nodes whose distance equals to the diameter of the team.

Motivated by the results of these studies, we choose to study the problem where we minimize the sum of distances and bound the diameter. We call this problem “the diameter constrained TFP with sum of distances objective (DC-TFP-SD)”.

In the remaining part of this section, we provide mathematical models for DC-TFP-SD. For each person  $i \in N$ , we define a binary variable  $y_i$  to be one if this person is in the team and zero otherwise. We define parameter  $a_{ik}$  to be one if person  $i \in N$  possesses skill  $k \in K$  and to be zero otherwise. We let set  $C$  to be the set of pairs of people in conflict, i.e., the set of pairs whose communication cost exceeds the allowed diameter,

and we eliminate teams that include such pairs. DC-TFP-SD can be modeled as follows:

$$\min \sum_{i \in N} \sum_{j \in N: i < j} p_{ij} y_i y_j \quad (1)$$

$$\text{s.t. } \sum_{i \in N} a_{ik} y_i \geq 1 \quad \forall k \in K, \quad (2)$$

$$y_i + y_j \leq 1 \quad \forall \{i, j\} \in C, \quad (3)$$

$$y_i \in \{0, 1\} \quad \forall i \in N. \quad (4)$$

The covering constraints (2) ensure that each required skill is covered, i.e., there is at least one person in the team who has that skill. The family of packing (conflict) constraints (3) forbids conflicting pairs in the team. The objective function is the sum of communication costs of team members.

We can use variables  $z_{ij} = y_i y_j$  for all  $i, j \in N$  with  $i < j$  to linearize the objective function:

$$\min \sum_{i \in N} \sum_{j \in N: i < j} p_{ij} z_{ij} \quad (5)$$

s.t. (2) - (4)

$$z_{ij} \geq y_i + y_j - 1 \quad \forall i, j \in N : i < j, \quad (6)$$

$$z_{ij} \leq y_i \quad \forall i, j \in N : i < j, \quad (7)$$

$$z_{ij} \leq y_j \quad \forall i, j \in N : i < j, \quad (8)$$

$$z_{ij} \geq 0 \quad \forall i, j \in N : i < j. \quad (9)$$

Constraints (6)-(9) are to linearize  $z_{ij} = y_i y_j$  and force  $z_{ij}$  to be one when both  $y_i$  and  $y_j$  are equal to one, and to be zero otherwise (Fortet, 1960). Since the objective function coefficients are nonnegative, constraints (7) and (8) can be dropped without changing the optimal value. Note that one can use constraints  $z_{ij} = 0$  for all  $\{i, j\} \in C$  instead of constraints (3), which gives similar results in terms of computation time. Using both constraints together proved to be less effective.

If  $C = \emptyset$ , then we obtain “the team formation problem with sum of distances objective (TFP-SD)”. The optimal solution of TFP-SD on the network in Figure 1, with  $p_{ij}$ ’s taken as in Table 1, is the team  $\{2,3,4\}$  with cost 2.342. The optimal solution of DC-TFP-SD with a diameter limit of 0.9 is the team  $\{4,5,6\}$  with cost 2.466.

### 3 Branch and bound algorithms

DC-TFP-SD is a quadratic set covering problem with side constraints (packing constraints (3)). One of the earliest studies on the quadratic set covering problem is by Bazaraa and Goode (1975) where the authors propose a cutting plane algorithm. Besides this study, the literature on the quadratic set covering is limited to a study of polynomial approximations by Escoffier and Hammer (2007), a linearization technique by Saxena and Arora (1997), which does not guarantee optimality, as shown by Pandey and Punnen (2017) and a study by Punnen et al. (2019) on comparing different representations of the problem.

As listed in the surveys of Loiola et al. (2007) on the quadratic assignment problem and Pisinger et al. (2007) on the quadratic knapsack problem, the formulations of 0-1 quadratic problems can be based on mixed

integer, convex quadratic or semidefinite programming and mostly they are too large to be solved in their current forms. Therefore they are relaxed and embedded into an algorithm such as a branch and bound, cutting plane, dual ascent algorithm or a combination of those. Most recent studies with semidefinite relaxations include works of Povh and Rendl (2009), Mittelman and Peng (2010), and de Klerk et al. (2015) on the quadratic assignment problem and the work of Guimarães et al. (2020) on the quadratic minimum spanning tree. And among the studies based on mixed integer programming, see for instance, a constraint generation algorithm for the quadratic knapsack by Rodrigues et al. (2012), a branch and cut algorithm for the capacitated vehicle routing problem with quadratic objective by Martinelli and Contardo (2015) and a branch and price algorithm for the quadratic multiple knapsack by Bergman (2019).

As it can be seen from this brief review, the quadratic set covering problem has attracted very little attention as opposed to other quadratic 0-1 problems. In this section, we first present a branch and bound algorithm for TFP-SD, which is a quadratic set covering problem and then extend it to DC-TFP-SD, which is a quadratic set covering problem with side constraints.

### 3.1 Reformulation, relaxation and decomposition

For ease of decomposition, we define variable  $z_{ij}$  for all  $i, j \in N$  such that  $i \neq j$  instead of  $i < j$ . We apply the idea of the well-known reformulation-linearization technique (RLT) of Adams and Sherali (1986) to derive the following inequalities from the original covering constraints by multiplying each one with variable  $y_j$ :

$$\sum_{i \in N \setminus \{j\}} a_{ik} z_{ij} \geq (1 - a_{jk}) y_j \quad \forall k \in K, j \in N.$$

The right hand side of this constraint is equal to 1 when person  $j$  is in the team but does not have skill  $k$ . Hence the constraint implies that, in this case, at least one person having skill  $k$  must be in the team. We can rewrite these constraints as follows:

$$\sum_{i \in N \setminus \{j\}} a_{ik} z_{ij} \geq y_j \quad \forall k \in K, j \in N : a_{jk} = 0. \quad (10)$$

We call these new constraints RLT constraints. By adding these into our previous model and making slight changes we obtain the following reformulation of TFP-SD:

$$\min \frac{1}{2} \sum_{i \in N} \sum_{j \in N \setminus \{i\}} p_{ij} z_{ij}$$

s.t. (2), (4), (10)

$$z_{ij} \leq y_j \quad \forall i, j \in N : i \neq j, \quad (11)$$

$$z_{ij} = z_{ji} \quad \forall i, j \in N : i < j, \quad (12)$$

$$z_{ij} \geq y_i + y_j - 1 \quad \forall i, j \in N : i < j, \quad (13)$$

$$z_{ij} \in \{0, 1\} \quad \forall i, j \in N : i \neq j. \quad (14)$$

In the reformulation, we use constraints  $z_{ij} \in \{0, 1\}$  rather than  $z_{ij} \geq 0$  for all  $i, j \in N$  with  $i \neq j$  even though the latter constraints are also sufficient to have a correct formulation. However, in the sequel, we will

relax some constraints and the integrality of  $z$  variables will not be implied in the relaxed problem.

There are many studies on using RLT to solve quadratic problems. In the works of Adams et al. (2007) and Hahn et al. (2012) different levels of RLT are used for the quadratic assignment problem. In these studies Lagrangian relaxation is applied to the reformulations and embedded into a branch and bound algorithm. The technique is also used for the quadratic knapsack problem by Billionnet and Calmels (1996), Caprara et al. (1999), Pisinger et al. (2007), Fomeni et al. (2014). The main distinction between these reformulations and ours is that constraints of type (13) are redundant in these reformulations due to problem and cost structure, whereas in our case, they are necessary.

We are interested in the relaxation of the reformulation obtained by removing constraints (12) and (13). Let  $(y^*, z^*)$  be an optimal solution of the relaxation. Since constraints (12) are relaxed,  $z_{ij}^*$  may not be equal to  $z_{ji}^*$ . Furthermore, we might get a solution where  $z_{ij}^* \neq y_i^* y_j^*$  or  $z_{ji}^* \neq y_i^* y_j^*$  or both, since we relaxed constraints (13). To remove such infeasibilities, we branch by creating two nodes: at one node we allow at most one of  $i$  and  $j$  to be in the team and at the other node we force both to be in the team. Suppose now that we are at node  $\ell$  of the branch and bound tree and so far while branching we have added the constraints that at most one of  $i$  and  $j$  can be in the team for all  $\{i, j\} \in C_\ell^1$  (by adding the constraints  $y_i + y_j \leq 1$ ,  $z_{in} + z_{jn} \leq y_n$  for all  $n \in N \setminus \{i, j\}$  and  $z_{ij} = z_{ji} = 0$ ) and that  $i$  and  $j$  are both in the team for all  $\{i, j\} \in C_\ell^2$  (by adding the constraints  $y_i = y_j = 1$ ,  $z_{in} = z_{jn} = y_n$  for all  $n \in N \setminus \{i, j\}$  and  $z_{ij} = z_{ji} = 1$ ). Then the relaxation at node  $\ell$ , called  $R_\ell$ , is as follows.

$$\begin{aligned} \min \quad & \frac{1}{2} \sum_{i \in N} \sum_{j \in N \setminus \{i\}} p_{ij} z_{ij} \\ \text{s.t.} \quad & (2), (4), (10), (11), (14) \\ & y_i + y_j \leq 1 \quad \forall \{i, j\} \in C_\ell^1, \tag{15} \\ & y_i = y_j = 1 \quad \forall \{i, j\} \in C_\ell^2, \tag{16} \\ & z_{in} + z_{jn} \leq y_n \quad \forall \{i, j\} \in C_\ell^1, n \in N \setminus \{i, j\}, \tag{17} \\ & z_{in} = z_{jn} = y_n \quad \forall \{i, j\} \in C_\ell^2, n \in N \setminus \{i, j\}, \tag{18} \\ & z_{ij} = z_{ji} = 0 \quad \forall \{i, j\} \in C_\ell^1, \tag{19} \\ & z_{ij} = z_{ji} = 1 \quad \forall \{i, j\} \in C_\ell^2. \tag{20} \end{aligned}$$

Next we show that  $R_\ell$  can be solved by solving  $|N|+1$  linear set covering problems with side constraints. See, for instance, Caprara et al. (1999) for a similar result for the quadratic knapsack problem.

**Proposition 1.** *The relaxation  $R_\ell$  can be solved by solving  $|N|+1$  linear set covering problems with side constraints as follows. For each  $n \in N$ , we solve the linear set covering problem  $(Pr_n)$ , that will be referred*

to as “subproblem  $n$ ”:

$$v_n = \min \sum_{i \in N \setminus \{n\}} p_{in} \zeta_i^n \quad (21)$$

$$\text{s.t. } \sum_{i \in N \setminus \{n\}} a_{ik} \zeta_i^n \geq 1 \quad \forall k \in K : a_{nk} = 0,$$

$$\zeta_i^n + \zeta_j^n \leq 1 \quad \forall \{i, j\} \in C_\ell^1 : i, j \neq n, \quad (22)$$

$$\zeta_i^n = \zeta_j^n = 1 \quad \forall \{i, j\} \in C_\ell^2 : i, j \neq n, \quad (23)$$

$$\zeta_i^n = 0 \quad \forall \{i, n\} \in C_\ell^1, \quad (24)$$

$$\zeta_i^n = 1 \quad \forall \{i, n\} \in C_\ell^2, \quad (25)$$

$$\zeta_i^n \in \{0, 1\} \quad \forall i \in N \setminus \{n\}. \quad (26)$$

with optimal solution  $\bar{\zeta}^n$  and optimal value  $v_n$ . Then the optimal value of  $R_\ell$  can be computed by solving the following “master problem”:

$$\nu = \min \frac{1}{2} \sum_{j \in N} v_j y_j$$

$$\text{s.t. } \sum_{j \in N} a_{jk} y_j \geq 1 \quad \forall k \in K,$$

$$y_i + y_j \leq 1 \quad \forall \{i, j\} \in C_\ell^1,$$

$$y_i = y_j = 1 \quad \forall \{i, j\} \in C_\ell^2,$$

$$y_j \in \{0, 1\} \quad \forall j \in N.$$

Moreover the solution  $(y^*, z^*)$ , where  $y^*$  is an optimal solution of the master problem and  $z_{ij}^* = y_j^* \bar{\zeta}_i^j$  for all  $i, j \in N : i \neq j$ , is an optimal solution for  $R_\ell$ .

**Proof.** It is sufficient to observe that in  $R_\ell$ , for a given vector  $y$ , the problem of computing the best  $z$  decomposes into subproblems, one for each  $n \in N$  with  $y_n = 1$ . When  $y_n = 1$ , the best values of  $z_{in}$ ’s are  $z_{in} = \bar{\zeta}_i^n$  for all  $i \in N \setminus \{n\}$ . Then the best  $y$  can be computed by solving the above master problem.  $\square$

We note that we can also multiply constraints (2) with  $(1 - y_j)$  for  $j \in N$  and obtain valid inequalities  $\sum_{i \in N \setminus \{j\}} a_{ik} (y_i - z_{ij}) \geq 1 - y_j$  for  $k \in K$  after substituting  $z_{ij} = y_i y_j$  for  $i \in N \setminus \{j\}$  and  $y_j(1 - y_j) = 0$ . However, if we add these constraints to our reformulation, then the relaxed problem does not decompose any more.

In our branch and bound algorithm, we propose to work with a weaker relaxation  $R'_\ell$  which is obtained by dropping constraints (17) and (18) in  $R_\ell$ . The relaxation  $R'_\ell$  can be solved by solving for each  $n \in N$  the “relaxed subproblem”  $Pr'_n$ , which is obtained from subproblem  $Pr_n$  by dropping constraints (22) and (23), with optimal solution  $\bar{\zeta}'^n$  and optimal value  $v'_n$  and then by solving the “relaxed master problem”, whose optimal value is  $\nu'$  and in which  $v_j$ ’s are replaced by  $v'_j$ ’s in the objective function.

At the root node  $\ell = 0$ ,  $R'_0$  is the same as  $R_0$  and is solved by solving  $|N| + 1$  linear set covering problems. We need less computation at the other nodes as we explain next in Proposition 2.

**Proposition 2.** *At node  $\ell$  of the branch and bound tree where  $\ell$  is not the root node, the relaxation  $R'_\ell$  can be*



solved by solving at most three linear set covering problems with side constraints if the optimal solutions and optimal values of the subproblems at the parent node are available.

**Proof.** Let  $\ell'$  be the parent node of node  $\ell$ . Suppose that we obtained the current node by adding  $\{i', j'\}$  to  $C_\ell^1$ , i.e.,  $C_\ell^1 = C_{\ell'}^1 \cup \{i', j'\}$  and  $C_\ell^2 = C_{\ell'}^2$ . Then we add the constraint  $y_{i'} + y_{j'} \leq 1$  to the master problem,  $\zeta_{i'}^{j'} = 0$  to the relaxed subproblem  $Pr'_{j'}$ ,  $\zeta_{j'}^{i'} = 0$  to the relaxed subproblem  $Pr'_{i'}$  and the other subproblems remain unchanged. If the optimal solution of  $Pr'_{i'}$  (resp.  $Pr'_{j'}$ ) at node  $\ell'$  satisfies  $\zeta_{j'}^{i'} = 0$  (resp.  $\zeta_{i'}^{j'} = 0$ ), then it is also optimal for subproblem  $Pr'_{i'}$  (resp.  $Pr'_{j'}$ ) at node  $\ell$ . Otherwise we solve these subproblems and then we solve the master problem with the additional constraint  $y_{i'} + y_{j'} \leq 1$ . If the current node is obtained by adding  $\{i', j'\}$  to  $C_\ell^2$  then again we may need to solve the relaxed subproblems  $Pr'_{i'}$  and  $Pr'_{j'}$  with the additional constraints  $\zeta_{j'}^{i'} = 1$  and  $\zeta_{i'}^{j'} = 1$ , respectively, and then the master with  $y_{i'} = 1$  and  $y_{j'} = 1$ .  $\square$

As in  $R_\ell$ , the solution  $(y^*, z^*)$ , where  $y^*$  is an optimal solution of the relaxed master problem and  $z_{ij}^* = y_j^* \bar{\zeta}_i^j$  for all  $i, j \in N : i \neq j$ , where  $\bar{\zeta}^j$  is an optimal solution of the relaxed subproblem  $Pr'_{j'}$ , is an optimal solution for  $R'_\ell$ .

The lower bound we get from  $R'_\ell$  may not be as good as the lower bound of  $R_\ell$  and consequently the branch and bound tree may be larger. However, our preliminary analysis has shown that this approach is faster since the time spent at each node is significantly smaller.

### 3.2 Branching strategy

We should be able to eliminate a solution of the relaxation if it is not feasible for the original problem. We do this by branching. In Observation 1 we present different cases of infeasibility.

**Observation 1.** *If the optimal solution  $(y^*, z^*)$  to the relaxation  $R'_\ell$  at node  $\ell$  is not feasible for the original problem at node  $\ell$ , then there exists at least one pair  $\{i, j\}$  satisfying one of the following conditions:*

- $y_i^* = y_j^* = 1$  and  $z_{ij}^* = z_{ji}^* = 0$  (type 1 pair),
- or  $y_i^* = y_j^* = 1$ ,  $z_{ij}^* = 1$  and  $z_{ji}^* = 0$  (type 2 pair),
- or  $y_i^* = 1$ ,  $y_j^* = 0$ ,  $z_{ij}^* = 0$  and  $z_{ji}^* = 1$ .

We only branch on type 1 or type 2 pairs, by prioritizing the former. If the current solution is not feasible, we branch on the first type 1 pair we find. If none exists, we branch on the first type 2 pair (see Algorithm 1). Next in Proposition 3 we show that branching on only type 1 and type 2 pairs is sufficient.

**Proposition 3.** *If the optimal solution  $(y^*, z^*)$  to the relaxation  $R'_\ell$  at node  $\ell$  is not feasible for the original problem at node  $\ell$ , then there exists either a type 1 pair or a type 2 pair or  $(y^*, \bar{z})$  where  $\bar{z}_{ij} = y_i^* y_j^*$  for all  $i, j \in N$  such that  $i \neq j$  is an alternate optimal solution to the relaxation  $R'_\ell$ .*

*Proof.* Suppose that there is no type 1 or type 2 pair in  $(y^*, z^*)$  and the solution  $(y^*, \bar{z})$  is not an alternate optimal solution to the relaxation  $R'_\ell$ . Then by Observation 1 there exists at least one pair  $\{i, j\}$  such that  $y_i^* = 1$ ,  $y_j^* = 0$ ,  $z_{ij}^* = 0$  and  $z_{ji}^* = 1$ . As  $(y^*, \bar{z})$  is not an alternate optimal solution, for one of such pairs, setting  $z_{ji}$  to zero violates a constraint. Then there exists a skill  $k$  that is covered uniquely by  $j$  in the relaxed subproblem  $Pr'_i$  because otherwise setting  $z_{ji}$  to zero would be feasible. Since  $y_j^* = 0$ , skill  $k$  is covered by

another candidate, say candidate  $t$ , in the relaxed master problem. So  $y_t^* = 1$ . However  $\bar{\zeta}_t^i$  and consequently  $z_{ti}^*$  must be zero since  $k$  is covered uniquely by  $j$  in the subproblem  $Pr'_i$ . Then  $\{i, t\}$  is a pair with  $y_i^* = y_t^* = 1$  and  $z_{ii}^* = 0$  and is either a type 1 or type 2 pair. This contradicts our assumption.  $\square$

---

**Algorithm 1** BranchPair( $y^*, z^*$ )

---

```

1: for  $i \in N : y_i^* = 1$  do
2:   for  $j \in N : j > i, y_j^* = 1$  do
3:     if  $z_{ij}^* = z_{ji}^* = 0$  then
4:       pair  $\leftarrow \{i, j\}$ ;
5:       break
6: if pair=null then
7:   for  $i \in N : y_i^* = 1$  do
8:     for  $j \in N : j > i, y_j^* = 1$  do
9:       if  $z_{ij}^* \neq z_{ji}^*$  then
10:        pair  $\leftarrow \{i, j\}$ ;
11:        break
12: Return pair

```

---

### 3.3 Upper bounds

There are two ways to update the upper bound in our algorithm: via the subproblems and the master problem.

**Proposition 4.** Let  $N_j = \{i \in N : \bar{\zeta}_i^j = 1\} \cup \{j\}$  where  $\bar{\zeta}^j$  is an optimal solution to the relaxed subproblem  $Pr'_j$  for  $j \in N$  and  $N' = \{i \in N : y_i^* = 1\}$  where  $y^*$  is an optimal solution of the relaxed master problem solved at any node of the branch and bound tree. Then  $w^j = 1/2 \sum_{i' \in N_j} \sum_{j' \in N_j \setminus \{i'\}} p_{i'j'}$  for  $j \in N$  and  $w^0 = 1/2 \sum_{i' \in N'} \sum_{j' \in N' \setminus \{i'\}} p_{i'j'}$  are upper bounds for the optimal value.

**Proof.** For each  $j \in N$ , due to constraints (10) in the relaxed subproblem,  $N_j$  is a capable team. Similarly, due to constraints (2) in the master problem,  $N'$  is also a capable team. Their sum of distances values give upper bounds.  $\square$

At each node, after solving the relaxed subproblems and the master problem we update the upper bound and the incumbent solution if we find a better solution.

### 3.4 The algorithm

The branch and bound algorithm is presented in Algorithm 2. The current lower and upper bounds are denoted as  $LB$  and  $UB$ . At each node  $\ell$ , we keep the optimal solution of the subproblem  $\ell.\bar{\zeta}^n$  of  $Pr'_n$ , its optimal value  $\ell.v'_n$  for all  $n \in N$ , the optimal value of the relaxed master problem  $\ell.v'$  and its optimal solution  $(\ell.y^*, \ell.z^*)$ .

The initial step is to create the root node, 0, at which, we solve the relaxed subproblems  $Pr'_n$  for all  $n \in N$  and then the relaxed master problem whose optimal value becomes the first lower bound. As we preprocess our instances we do not need check for feasibility at the root node. As explained in Proposition 4, each time a relaxed subproblem or a relaxed master problem is solved, we check if we can update the upper bound and the incumbent solution, team  $T$ . If  $LB < UB$ , then we initialize the queue,  $Q$ , by adding the root node.

The algorithm runs until the lower bound is equal to the upper bound. We follow the best-first search rule for choosing the next node to process, breaking ties arbitrarily. Let  $\ell$  be a node in  $Q$  with the lowest lower bound. We remove  $\ell$  from the queue and find its branch pair, say  $\{i, j\}$ . We create child nodes  $\ell_1$  and  $\ell_2$  and solve relaxations  $R'_{\ell_1}$  and  $R'_{\ell_2}$  as explained in Proposition 2. Node  $\ell_1$  (resp.  $\ell_2$ ) is added to the queue only if  $\ell_2.\nu'$  (resp.  $\ell_1.\nu'$ ) is less than the current upper bound.

Throughout the algorithm when a relaxed subproblem or a relaxed master problem is infeasible, its objective value is set to infinity. Therefore if  $R'_\ell$  is infeasible then  $\ell.\nu' = \infty$ . In this case we discard node  $\ell$  since it does not satisfy  $\ell.\nu' < UB$ . This amounts to pruning by infeasibility. Furthermore if the solution  $(y^*, z^*)$  of relaxation  $R'_\ell$  is feasible for the original problem or it is not feasible but  $(y^*, \bar{z})$  where  $\bar{z}_{ij} = y_i^* y_j^*$  for all  $i, j \in N$  such that  $i \neq j$  is an alternate optimal solution to  $R'_\ell$ , then  $\ell.\nu' \geq UB$  since these solutions are used to update the upper bound. This corresponds to pruning by optimality. If the node is not pruned by infeasibility or optimality and  $\ell.\nu' \geq UB$  then the node is pruned by bound. Hence, if a node is added to the queue then it satisfies  $\ell.\nu' < UB$  and has at least one type 1 or type 2 branch pair.

---

**Algorithm 2** Branch and Bound

---

```

1:  $UB := \infty, T = \emptyset$ 
2: Create root node 0 with  $0.\nu' := \infty, C_0^1 := \emptyset, C_0^2 := \emptyset$ 
3: for  $n \in N$  do
4:   Solve  $Pr'_n$ 
5:    $0.\bar{\zeta}^n := \zeta^n$  and  $0.v'_n := v'_n$  ▷ update  $UB$  and  $T$  if possible
6: Solve the relaxed master problem
7:  $0.y^* := y^*, 0.z^* := z^*, 0.\nu' := \nu', LB := \nu'$  ▷ update  $UB$  and  $T$  if possible
8: if  $LB < UB$  then  $Q := \{0\}$ 
9: while  $LB < UB$  do
10:   $\ell = \arg \min_{\ell' \in Q} \{\ell'.\nu'\}, Q := Q \setminus \{\ell\}$ 
11:   $\{i, j\} := \text{BranchPair}(\ell.y^*, \ell.z^*)$ 
12:  Create node  $\ell_1 : \ell_1.v'_n = \ell.v'_n, \ell_1.\bar{\zeta}^n = \ell.\bar{\zeta}^n \forall n \in N, \ell_1.\nu' := \infty, C_{\ell_1}^1 := C_\ell^1 \cup \{i, j\}, C_{\ell_1}^2 := C_\ell^2$ 
13:  if  $\ell.\bar{\zeta}^i_j = 1$  then
14:    Solve  $Pr'_i$ 
15:    if feasible then  $\ell_1.v'_i := v'_i, \ell_1.\bar{\zeta}^i := \bar{\zeta}^i$  else  $\ell_1.v'_i := \infty$  ▷ update  $UB$  and  $T$  if possible
16:  if  $\ell.\bar{\zeta}^j_i = 1$  then
17:    Solve  $Pr'_j$ 
18:    if feasible then  $\ell_1.v'_j := v'_j, \ell_1.\bar{\zeta}^j := \bar{\zeta}^j$  else  $\ell_1.v'_j := \infty$  ▷ update  $UB$  and  $T$  if possible
19:  Solve relaxed master problem
20:  if feasible then  $\ell_1.y^* := y^*, \ell_1.z^* := z^*, \ell_1.\nu' = \nu'$  ▷ update  $UB$  and  $T$  if possible
21:  if  $\ell_1.\nu' < UB$  then  $Q := Q \cup \{\ell_1\}$ 
22:  Create node  $\ell_2 : \ell_2.v'_n = \ell.v'_n, \ell_2.\bar{\zeta}^n = \ell.\bar{\zeta}^n \forall n \in N, \ell_2.\nu' = \infty, C_{\ell_2}^1 := C_\ell^1, C_{\ell_2}^2 := C_\ell^2 \cup \{i, j\}$ 
23:  if  $\ell.\bar{\zeta}^i_j = 0$  then
24:    Solve  $Pr'_i$ 
25:    if feasible then  $\ell_2.v'_i := v'_i, \ell_2.\bar{\zeta}^i := \bar{\zeta}^i$  else  $\ell_2.v'_i := \infty$  ▷ update  $UB$  and  $T$  if possible
26:  if  $\ell.\bar{\zeta}^j_i = 0$  then
27:    Solve  $Pr'_j$ 
28:    if feasible then  $\ell_2.v'_j := v'_j, \ell_2.\bar{\zeta}^j := \bar{\zeta}^j$  else  $\ell_2.v'_j := \infty$  ▷ update  $UB$  and  $T$  if possible
29:  Solve relaxed master problem
30:  if feasible then  $\ell_2.y^* := y^*, \ell_2.z^* := z^*, \ell_2.\nu' := \nu'$  ▷ update  $UB$  and  $T$  if possible
31:  if  $\ell_2.\nu' < UB$  then  $Q := Q \cup \{\ell_2\}$ 
32:   $LB := \min_{\ell' \in Q} \{\ell'.\nu'\}$ 
33: Return  $UB$  and  $T$ 

```

---

### 3.5 Example

We illustrate the branch and bound algorithm on a small example. We would like to solve TFP-SD on the social network given in Figure 2. There are five candidates and the shortest path lengths are as shown on the edges. The project requires three skills and the skills of people are indicated by the shape of nodes.

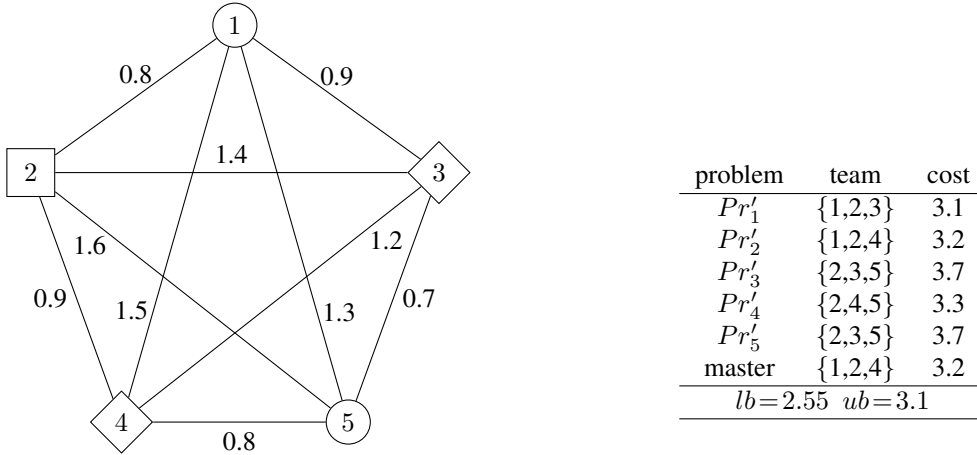


Figure 2: Example network, optimal solutions of the subproblems and the master and the bounds at the root node

At the root node of the branch and bound tree, we solve relaxation  $R_0 = R'_0$ , which requires solving five subproblems and then a master problem. In Figure 2 we summarize the information we get from these problems in the table next to the network. For example, the first row shows that the optimal solution of subproblem 1 is  $\bar{z}_2^1 = \bar{z}_3^1 = 1$ . The team consisting of people 1, 2 and 3 has a cost 3.1. This is the upper bound we get from this subproblem and actually it is best bound among all subproblems so the corresponding solution becomes the incumbent. The solution of the master problem is  $y_1^* = y_2^* = y_4^* = 1$  and  $y_3^* = y_5^* = 0$  with objective value of 2.55. This becomes the lower bound. We check whether we can use the solution of the master problem to update the upper bound. The team {1,2,4} costs 3.2, which is greater than the upper bound we get from subproblem 1 so the incumbent stays as {1,2,3}.

The entire branch and bound tree is illustrated in Figure 3. Next to each node, we summarize the solution and bound information in a table, similar to the one in Figure 2.

The solution at the root node is optimal unless we find a branch pair. Among  $i$  and  $j$  with  $y_i^* = y_j^* = 1$  we first look for a pair with  $z_{ij}^* = z_{ji}^* = 0$ . And {1,4} becomes our first branch pair. At the odd numbered nodes we ensure that the people in the branch pair are not teammates and at the even numbered ones they are forced to be in the team together. So in node (1) the problem  $R'_1$  has the sets  $C_1^1 = \{\{1, 4\}\}$  and  $C_1^2 = \emptyset$ . And in node (2), problem  $R'_2$  has  $C_2^1 = \emptyset$  and  $C_2^2 = \{\{1, 4\}\}$ .

At node (1), we only solve the relaxed master problem since the solution of the relaxed subproblem 1 (resp. 4) already satisfies  $\bar{z}'_4{}^1 = 0$  (resp.  $\bar{z}'_1{}^4 = 0$ ). The optimal solution of the relaxed master problem is team {1,2,3} and the lower bound we get at this node is 2.75. We do not update the upper bound as no better solution has been found. At node (2), we solve both relaxed subproblems, update  $v'_1$  and  $v'_4$ , and solve the relaxed master problem. Since the lower bound we get at this node is greater than the current incumbent we prune the node by bound. The algorithm continues with node (1) and the next branch pair becomes {1,3} which is a type 2 pair.

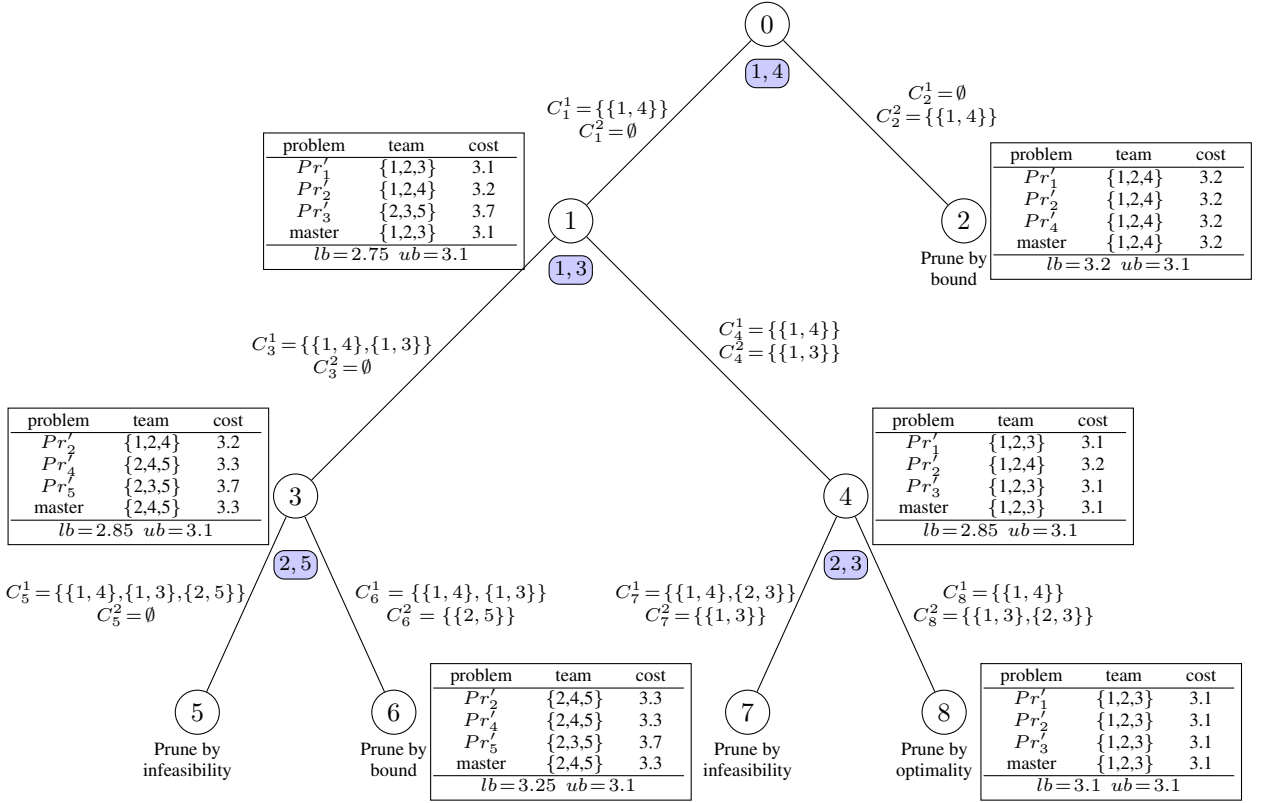


Figure 3: The branch and bound tree

We create node (3) and problem  $R'_3$  with  $C_3^1 = \{\{1, 4\}, \{1, 3\}\}$  and  $C_3^2 = \emptyset$ . We solve the relaxed subproblem 1 at this node, update  $v'_1$  and solve the relaxed master problem. The lower bound at this node becomes 2.85. At node (4) we create problem  $R'_4$  with  $C_4^1 = \{\{1, 4\}\}$  and  $C_4^2 = \{\{1, 3\}\}$ . We solve the relaxed subproblem 3, update  $v'_3$  and then solve the relaxed master which gives the same lower bound as node 3. We can continue with either of them so we choose node 3 and the branch pair is  $\{2, 5\}$ . At node (5) we create problem  $R'_5$  with  $C_5^1 = \{\{1, 4\}, \{1, 3\}, \{2, 5\}\}$  and  $C_5^2 = \emptyset$ . We solve relaxed subproblem 5 and update  $v'_5$  but the relaxed master problem becomes infeasible and we prune the node. Continuing in this manner the algorithm terminates at node (8) proving that the upper bound 3.1 found at the root node is actually the optimal value.

### 3.6 Branch and bound algorithm for DC-TFP-SD

We can use a similar branch and bound algorithm to solve DC-TFP-SD by making two adjustments. The first adjustment is in the relaxation that we solve to compute a lower bound and the second adjustment is in the way we update upper bounds.

Recall that  $C$  is the set of pairs in conflict and we forbid them by constraints (3) in the formulation of DC-TFP-SD. Also recall that  $R'_\ell$  is the weaker relaxation of the reformulation of TFP-SD at node  $\ell$  of the branch and bound tree.

For DC-TFP-SD, we can treat the conflict constraints (3) like the constraints we use in branching and add them to the master and the related subproblems. However, our preliminary analysis has shown that it is better

to work with a further relaxation. We define  $R_\ell''$  to be the relaxation obtained by adding constraints (19) for all  $\{i, j\} \in C$  to  $R_\ell'$ . In other words, we add the conflict constraint for pair  $\{i, j\} \in C$  to the subproblems  $i$  and  $j$  and not to the other subproblems nor the master. As a result, we have weaker lower bounds but we work with a smaller master problem.

The second adjustment is in the upper bounding procedure. In Proposition 4 we define the set  $N_j$  for  $j \in N$  and  $N'$  by the solutions of subproblem  $j$  and master problem, respectively. For TFP-SD, the teams defined by these sets were capable teams so their cost values,  $w^j$  for  $j \in N$  and  $u^0$  gave upper bounds. In DC-TFP-SD, these are still capable teams but they might have a pair in conflict. Thus the second adjustment in the algorithm is to check the feasibility of these teams. If these teams have no pairs in conflict their cost values are upper bounds for the optimal value of DC-TFP-SD.

Using the relaxation  $R_\ell''$  and this upper bounding procedure, we obtain valid lower and upper bounds. Next, we prove that if the optimal solution  $(y^*, z^*)$  that we obtain by solving  $R_\ell''$  does not satisfy the conflict constraints (3), then there exists a type 1 pair that we can branch on.

**Proposition 5.** *Let  $(y^*, z^*)$  be the optimal solution of  $R_\ell''$ . If there exists a pair  $\{i, j\} \in C$  for which  $(y^*, z^*)$  violates the conflict constraint (3), i.e.,  $y_i^* = y_j^* = 1$ , then  $\{i, j\}$  is a type 1 branch pair.*

**Proof.** Suppose that  $(y^*, z^*)$  violates the conflict constraint (3) for pair  $\{i, j\} \in C$ . Then  $y_i^* = y_j^* = 1$ . Since the subproblems for  $i$  and  $j$  contain constraints (19), we have  $z_{ij}^* = z_{ji}^* = 0$ . Then  $\{i, j\}$  is a type 1 pair.  $\square$

## 4 Experiments

In this section we first introduce the social networks used in our computational study and explain how we generate our instances. Then we present the performance results of our branch and bound algorithm and its comparison with the mathematical models.

### 4.1 Datasets and instance generation

Wi et al. (2009) use collaborative data from a R&D institute and form a social network of 45 researchers to test their genetic algorithm. Farasat and Nikolaev (2016) use existing social network datasets to test their heuristics and the number of nodes in these networks varies from 15 to 500. On the other hand, larger social networks are preferred in knowledge discovery and data mining literature. We follow the latter course and use the IMDb and DBLP datasets in our computations.

IMDb is used by Anagnostopoulos et al. (2012) and Kargar and An (2011). We create our instances using the same part of the database used in the comparative study by Wang et al. (2015). The collaboration and skill information is provided by one of the authors on his website<sup>1</sup>. The nodes of the network are the actors who appeared in the movies from year 2000 to 2002. There are 1021 actors, i.e.,  $|N| = 1021$ . The skills are the genres of the movies and there are 27 skills. The social network contains an edge between actors  $i$  and  $j$  if they have worked together in a movie and the weight of the edge equals to the Jaccard distance as explained in Section 2.

<sup>1</sup><http://home.cse.ust.hk/faculty/wilfred/wangxinyu/>

DBLP is the most common database used to generate instances for TFP. It provides bibliographic information of papers published in major computer science journals and proceedings. We generate a social network from this database searching the papers published between years 2010-2016. We narrow the search space by specifying journals and conferences. Since there is no keyword information for the papers in the database, we search the titles of the papers for some keywords and treat these keywords as the skills of the authors. There is an edge between two authors if they have at least two common papers in whole history. With this setting, we end up with 58 skills and a collaboration network which has 12855 nodes and 53890 edges whose weights equal to the Jaccard distances. In both networks, we compute the shortest path lengths between all pairs and if there is no path between  $i$  and  $j$ , we make the communication cost between  $i$  and  $j$ ,  $p_{ij}$ , equal to a sufficiently large number. In Figure 4, to give an idea about the magnitudes and distribution of the communication costs, we plot the percentage of pairs whose distance is at most  $d$  for each network.

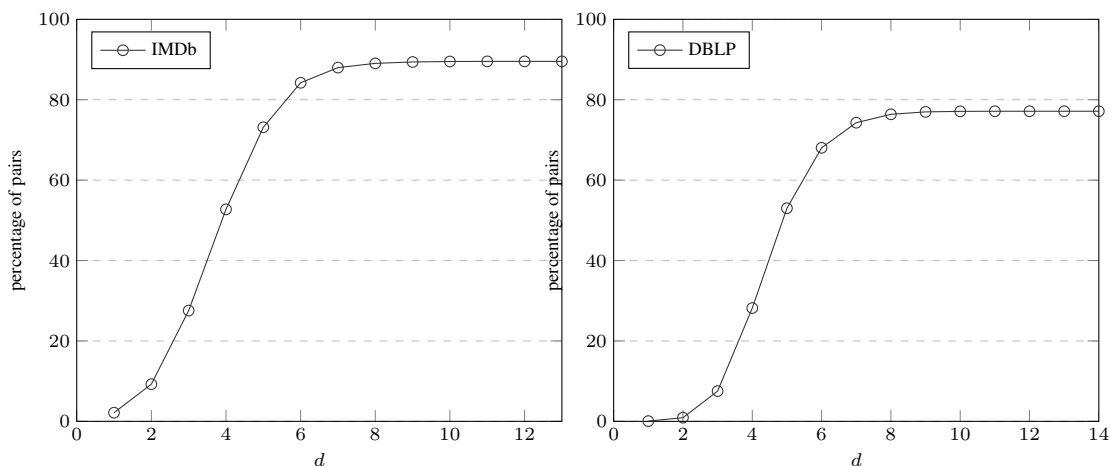


Figure 4: The percentage of pairs whose shortest distance is at most  $d$  in the IMDb (left) and DBLP (right) networks

For both social networks, we have created instances in the following way. The number of required skills,  $m$ , comes from the set  $\{4, 6, 8, 10, 12, 14, 16, 18, 20\}$  and 100 random instances are generated for each  $m$ . The data sets and the instances used in the computational experiments are available in our Github repository<sup>2</sup>.

## 4.2 Computational results

The mathematical models and the branch and bound algorithms are implemented in Java using CPLEX 12.7 and run on a personal computer with an Intel(R) Core(TM) i7-6700HQ 2.6 GHz and 16 GB of RAM. All computational times reported in the tables are wall-clock times in seconds.

For each instance it is sufficient to consider people who have one of the required skills. Therefore, we preprocess the input data and shrink the social network by removing people who do not possess any of the required skills. We call the remaining nodes in the network as the qualified ones and their number is denoted by  $qno$  in the sequel. For the diameter constrained version of the problem, we are able to reduce the network further by eliminating a person if he/she cannot cover all the skills together with the people who are at most

<sup>2</sup><https://github.com/nihalberktas/TFP-data>

Table 2: Results for TFP-SD on the IMDb instances.

$m$	$qno$	QP		MIP		B&C		B&B	
		$time$	$time$	$LP - gap$	$time$	$time$	$nodes$	$lb - gap$	$ub - gap$
4	422.51	6.66	7.14	63.60	0.81	1.13	2.08	2.12	0.00
6	541.81	22.63	23.21	77.75	1.74	2.66	14.11	4.12	0.05
8	653.41	28.5	29.54	77.07	3.16	4.19	24.6	5.95	0.06
10	731.82	30.41	31.12	75.47	5.63	5.92	41.97	10.27	0.30
12	791.51	32.6	33.47	75.90	7.59	7.28	52.36	12.31	0.22
14	838.48	43.13	44.7	74.00	10.62	9.83	111.34	13.31	0.50
16	879.02	51.81	53.04	72.76	15.57	12.27	157.72	13.58	0.18
18	917.68	83.76	81.04	71.92	18.77	14.31	164.98	15.13	0.77
20	947.69	77.98	78.54	71.23	24.93	13.93	167.69	16.24	0.70

allowed diameter away from him/her. We do this elimination iteratively until there is no one to remove from the network. After this preprocessing, the network only involves people who are capable of forming a feasible team respecting the bound on the diameter. The number of candidates after preprocessing is denoted by  $fno$ .

In addition to the quadratic formulation (1), (2), (4), denoted by QP, the mixed integer formulation (2), (4)-(9), denoted by MIP, and the B&B algorithm, we implemented a branch and cut algorithm for TFP-SD to overcome the memory problems for larger instances. In the MIP formulation, the constraints (6), (7) and (8) grow quadratically in the size of the problem. Since the objective coefficients are nonnegative in our instances, it is sufficient to use only constraints (6) but even in this case we have memory run-outs in the model generation phase for large instances. When we use the original MIP formulation without constraints (7) and (8) and add constraints (6) using the lazy cut pool (the constraints in this pool are only checked when an integer feasible solution is found and violated constraints are added to the formulation), a large number of lazy constraints are added and consequently this approach takes more time than solving the MIP formulation directly. However when we add the RLT constraints (10), only a small number of lazy constraints are generated and this improves the solution times. The cuts can also be applied at the fractional solutions by putting constraints (6) to the user cut pool besides the lazy cut pool but the computation times are longer in this case. Therefore in our branch and cut implementation (B&C), we solve the mixed integer programming formulation (2), (4), (5), (9), (10) by putting constraints (6) to the lazy cut pool.

We report the average solution times of all solution procedures for TFP-SD on the IMDb instances in Table 2. The averages are taken over 100 instances for each  $m$ . We present more detailed results for our branch and bound algorithm:  $nodes$  is the number of nodes evaluated,  $lb - gap = 100(opt - lb)/opt$  and  $ub - gap = 100(ub - opt)/opt$  where  $lb$  and  $ub$  are the lower and upper bounds at the root node, respectively, and  $opt$  is the optimal value. To see the strength of the linear programming relaxation of the mixed integer formulation (2), (4)-(9), we also report  $LP - gap = 100(opt - LP)/opt$ , where  $LP$  is the optimal value of the linear programming relaxation. As it can be seen in Table 2, the continuous relaxation is very weak.

The performances of the QP and MIP formulations for TFP-SD turn out to be very similar for the IMDb instances. On the average the optimal solution is reported within a minute or two by the solver with both mathematical models. When we compare these with the B&B algorithm we clearly see the efficiency of the algorithm as it reaches the optimal solution six times faster than the models on the average. The instance with the longest solution time requires more than 1300 seconds for both formulations and it is solved in 19 seconds



by the B&B algorithm. The longest time the B&B algorithm spends for an IMDb instance is actually 48.19 seconds. With the B&C, we are able to solve 98.6% of the instances within a minute while this percentage is 78% both for the QP and the MIP. When the number of required skills,  $m$ , is low, this method is as efficient as the B&B algorithm; but as  $m$  grows, the B&B algorithm outperforms the B&C as well. Analyzing the detailed results, we observe that for all instances with  $m = 4$  the first incumbent found by the B&B algorithm is optimal. Although the quality degrades as the instances get larger, the initial upper bound is at most 1% away from the optimal in 93.55% of the instances.

Table 3: Results for TFP-SD on the DBLP instances.

$m$	$qno$	QP		MIP		B&C		B&B	
		<i>solved</i>	<i>time</i>	<i>solved</i>	<i>time</i>	<i>solved</i>	<i>time</i>	<i>solved</i>	<i>time</i>
4	1650.5	10	343.04	10	359.75	10	53.95	10	9.84
6	2239.80	4	336.79	4	352.96	10	142.59	10	20.65
8	2896.50	2	386.29	2	2508.42	8	279.39	10	36.56

In Table 3, we present the results for TFP-SD on the DBLP instances. As the DBLP network is a larger one, we could not obtain a solution from the mathematical models for most of the instances. Therefore we only include the results for  $m = 4, 6$  and  $8$  in this table to compare the performances. In general, we observe memory problems when the number of qualified people,  $qno$ , exceeds 2100 and  $m$  is greater than 4. The column *solved* indicates the number of instances that can be solved to optimality out of 10. The average solution times are given for the instances solved. We see that MIP and QP can only solve four instances with  $m = 6$  and two instances with  $m = 8$  whereas strengthening the model with RLT constraints and putting constraints (6) to the lazy cut pool in the B&C framework enables us to solve more instances within less time. But eventually this method also fails with memory problems as the size of instances increases. Furthermore the performances of MIP and the QP which were very similar on the IMDb instances start to differ as the problem size gets larger. The instances solved without memory problems are the same for both formulations, however, the solution times of QP are lower than those of MIP. Having average solution times under a minute, the efficiency of the B&B algorithm is clearly seen in this table. Its longest solution time among these instances is actually 62.2 seconds.

Table 4: Detailed results of the B&B algorithm for TFP-SD on the DBLP instances.

$m$	$qno$	<i>solved</i>	<i>time</i>				<i>nodes</i>	<i>lb - gap</i>	<i>ub - gap</i>
			<i>min</i>	<i>avg</i>	<i>max</i>	<i>std</i>			
4	1540.22	100	0.48	8.59	42.43	9.02	20.08	4.97	0.05
6	2255.9	100	1.53	20.68	67.55	13.59	30.54	6.47	0.27
8	2963.26	100	1.88	37.69	107.27	21.57	110.52	8.16	0.48
10	3604.4	100	6.75	59.86	191.79	17.10	239.10	7.99	0.69
12	4189.49	99	20.65	89.41	275.92	49.05	480.52	8.56	0.89
14	4789.13	99	35.60	249.25	4921.88	633.18	3374.63	8.79	0.89
16	5298.52	99	47.47	274.76	3571.15	482.30	3099.22	8.62	0.66
18	5857.6	97	66.48	482.83	4743.17	707.73	5637.57	9.25	0.76
20	6412.48	91	114.81	680.89	4998.51	1030.91	6439.47	9.32	0.82

We present detailed results of the B&B algorithm on the DBLP instances in Table 4. We also consider

Table 5: Results of B&B algorithm for TFP-SD on IMDb<sup>r</sup> and DBLP<sup>r</sup>: the IMDb and DBLP instances with randomly generated skill matrices.

$m$	IMDb <sup>r</sup>					DBLP <sup>r</sup>				
	$qno$	$min$	$avg$	$max$	$std$	$qno$	$min$	$avg$	$max$	$std$
4	531.91	0.50	1.95	3.23	0.59	1602.76	1.30	11.14	26.41	5.32
6	693.47	1.43	3.96	10.89	1.13	2247.75	5.25	21.67	41.12	7.23
8	794.86	2.79	5.44	15.22	1.30	2878.79	9.23	34.54	74.43	10.18
10	863.80	4.44	6.91	9.75	0.85	3485.32	16.41	55.12	152.94	19.98
12	912.65	6.55	9.40	26.72	3.08	4065.39	30.42	80.16	161.53	25.24
14	949.29	8.99	12.86	62.74	6.10	4587.13	55.27	142.26	866.07	95.97
16	971.96	11.87	18.30	88.25	9.19	5112.02	71.10	248.54	1239.47	214.24
18	986.62	15.23	30.20	215.19	28.92	5591.90	89.30	396.69	1735.91	362.95
20	997.61	20.20	48.58	421.12	62.72	6018.28	115.99	580.47	3718.81	704.72

larger  $m$  values here. The column titled *solved* indicates the number of instances solved to optimality within a two hour time limit over 100 instances for each  $m$ . The computational details presented in the table is for the instances that are solved within the time limit. We present the minimum, average and maximum solution times for each  $m$  and also the standard deviation of these times under the columns titled *min*, *avg*, *max*, *std*, respectively. The algorithm is able to solve all DBLP instances with  $m = 4, 6, 8, 10$  within the limit and actually the highest solution time among these instances is around 3 minutes. When  $m = 12$  there is only one instance that cannot be solved within two hours and as  $m$  increases we have few more. Among all, the algorithm is able to solve 43% of the instances in a minute and 97.8% of them in an hour. Similar to the results with the IMDb instances the upper bound at the root node is very close to the optimal solution. Approximately at 69% of the instances, this upper bound is at most 1% away from the optimal value.

In the IMDb and DBLP instances we use, due to the way skills are defined and assigned, it might be possible that closer nodes in the network have more skills in common. To investigate whether the performance of the algorithm is affected by such possible correlation between distances and skills, we generated purely random skill matrices that do not have any connection to the distances. As before, we generate 100 instances for each  $m$  value. In Table 5 we present the results on these instances for TFP-SD where the new sets are denoted by IMDb<sup>r</sup> and DBLP<sup>r</sup>.

Each one of these instances is solved within the two hour time limit by the B&B algorithm. The solution times of the random IMDb<sup>r</sup> instances are higher than those of the original ones presented in Table 2 when we compare the corresponding rows for each  $m$ . However we must observe that the average number of qualified people, *qno*, for each  $m$  is also higher in the new set of instances.

In the rest of this section we present the results of the computational experiments for the diameter constrained version of the problem, DC-TFP-SD, using QP ((1)-(4)), MIP ((2)-(9)) and the B&B algorithm. For the IMDb instances, we first found the optimal diameters. In Table 6, we present the solution times for DC-TFP-SD where we use the optimal diameter values as the diameter bounds. Due to preprocessing, the network is reduced significantly (we remind that *fno* is the number of candidates after preprocessing) and therefore the solutions times of all methods are very small. For example when  $m = 20$  the network consists of more than 900 qualified people on the average but the number of candidates reduces to approximately 200 people when we exclude the people who cannot build a team respecting the bound on the diameter. Therefore the solution times are only 1

Table 6: Results for DC-TFP-SD on the IMDb instances where the bound on the diameter is taken as the optimal diameter

$m$	$qno$	$fno$	QP	MIP	B&B			
			$time$	$time$	$time$	$nodes$	$lb - gap$	$ub - gap$
4	422.51	8.69	0.01	0.01	0.02	0.67	0.45	0.00
6	541.81	20	0.02	0.02	0.09	4.57	0.64	0.05
8	653.41	41.52	0.06	0.09	0.30	6.54	1.36	0.00
10	731.82	69.77	0.13	0.18	0.28	13.77	2.01	0.01
12	791.51	91.99	0.24	0.31	0.44	22.31	2.61	0.12
14	838.48	119.16	0.49	0.56	0.65	22.04	2.70	0.04
16	879.02	152.62	0.89	0.98	1.10	45.02	3.67	0.06
18	917.68	178.94	1.18	1.35	1.49	69.81	3.93	0.08
20	947.69	216.11	1.80	2.07	2.20	104.34	4.71	0.09

or 2 seconds for all methods.

Table 7: Results for DC-TFP-SD on the IMDb instances.

$m$	D=2					D=3				
	$feas$	$fno$	MIP	QP	B&B	$feas$	$fno$	MIP	QP	B&B
4	94	250.01	8.26	5.38	0.47	97	317.16	15.24	7.86	0.72
6	88	266.10	10.94	7.80	0.76	92	375.60	25.60	16.67	1.47
8	79	265.19	11.75	8.48	0.95	87	402.48	28.89	19.74	1.89
10	66	279.82	13.10	9.47	1.28	79	427.18	36.49	20.97	2.43
12	60	255.93	13.11	8.97	1.31	74	424.51	34.69	20.47	2.71
14	48	208.94	11.65	7.59	1.28	68	389.79	30.79	17.85	2.85
16	36	208.03	10.97	7.84	1.30	60	382.97	29.96	17.21	2.76
18	24	165.79	10.72	6.73	1.49	54	353.80	23.37	15.22	3.22
20	14	192.79	16.30	8.42	1.47	45	349.62	21.74	16.03	3.81

We continued the experiments of DC-TFP-SD with the IMDb instances with varying bounds on the diameter. In Table 7 we present the results with  $D$  equal to 2 and 3. Under the column  $feas$ , the number feasible instances is given over 100 instances for each  $m$ . In the IMDb instances, the optimal diameter is usually less than 2 and therefore the number of candidates that remain after preprocessing is greater with  $D = 2, 3$  than with  $D$  taken as the optimal diameter. Thus, as the bound on the diameter increases, so does the size of the network and we start to observe differences in the performances of the solution methods. When the bound on diameter is taken as its optimal value, all solution procedures are able to find optimal solutions within 1 or 2 seconds. When we take the bound as 2 and 3, the solution times of MIP and QP become 15 seconds on the average while it is still a couple of seconds for the branch and bound. To be more specific, the maximum solution times of MIP, QP and B&B with  $D = 2$  are 60, 43, 12 and with  $D = 3$  they are 197, 189, 23 seconds, respectively.

For the DBLP instances, we used 1, 2, 3 and 4 as the bound on the diameter. To be able to compare the solution procedures we first present the results of the first 10 instances with  $m = 4, 6, 8$  and 10 for  $D = 2$  and  $D = 3$  in Table 8. For these instances average solution time of MIP is few minutes and usually less than that of QP. Nevertheless QP is able to solve all these instances while we encounter memory errors with MIP when

Table 8: Results for DC-TFP-SD on the DBLP instances

<i>m</i>	D=2						D=3							
	MIP		QP		B&B		MIP		QP		B&B			
	<i>feas</i>	<i>solved</i>	<i>time</i>	<i>solved</i>	<i>time</i>	<i>solved</i>	<i>time</i>	<i>feas</i>	<i>solved</i>	<i>time</i>	<i>solved</i>	<i>time</i>		
4	9	9	64.48	9	98.61	9	1.47	10	10	303.60	10	340.22	10	5.15
6	5	5	37.95	5	48.09	5	1.43	10	10	318.63	10	285.92	10	6.87
8	3	3	81.08	3	113.66	3	3.36	10	9	328.54	10	535.28	10	10.04
10	1	1	244.01	1	415.34	1	22.41	8	7	161.45	8	661.48	8	9.78

$D = 3$  and  $m$  exceeds 6. The B&B algorithm, on the other hand, is able to solve each of these instances under 30 seconds.

Table 9: Results of the B&B algorithm for DC-TFP-SD on the DBLP instances

<i>m</i>	D=1			D=2			D=3			D=4		
	<i>feas</i>	<i>solved</i>	<i>time</i>	<i>feas</i>	<i>solved</i>	<i>time</i>	<i>feas</i>	<i>solved</i>	<i>time</i>	<i>feas</i>	<i>solved</i>	<i>time</i>
4	35	35	0.08	83	83	1.87	99	99	4.06	100	100	6.60
6	7	7	0.02	64	64	1.48	97	97	7.51	100	100	15.11
8	1	1	0.03	36	36	6.60	92	92	12.26	100	100	27.02
10	0	0	0	21	21	15.24	82	82	22.42	98	98	35.14
12	0	0	0	14	14	10.64	78	78	124.95	96	96	51.77
14	0	0	0	9	9	9.76	68	67	56.31	94	94	131.83
16	0	0	0	2	2	5.51	56	54	139.12	93	91	176.08
18	0	0	0	2	2	3.57	49	48	267.20	90	87	297.41
20	0	0	0	0	0	0.00	38	35	187.83	87	83	366.99

In Table 9, we present the results for all DBLP instances using the B&B algorithm for DC-TFP-SD with  $D = 1, 2, 3, 4$ . The averages of solution times are taken over the instances that are solved within two hours of time limit and there are only 16 unsolved instances among 1791 feasible ones. The algorithm is able to solve 79% and 96% of the feasible instances within one and ten minutes, respectively.

## 5 Conclusion

In this study, we formulated the team formation problem as a quadratic set covering problem with packing constraints and developed a novel branch and bound algorithm to solve it. The algorithm uses a relaxation that can be solved by solving a series of linear set covering problems and a different branching rule compared to existing branch and bound methods for quadratic 0-1 optimization problems. Our computational experiments on team formation problem instances show that the algorithm is capable of solving large sizes that are intractable for the solver. The same approach can be used to solve other binary quadratic problems, however the success depends, among other things, on how quickly the relaxation (the corresponding 0-1 linear problems) can be solved.

In terms of application the present work can be extended in several ways. First, the communication cost may be quantified with respect to tasks in which case the problem also requires assigning people to tasks. Second,

the uncertainty in the communication costs can be incorporated into the decision making process using robust optimization and stochastic programming. This can be done in a single stage setting where the worst case or expected communication cost can be minimized or it can be done in a multi stage setting where decisions can be updated over time to improve the performance of the team.

## Acknowledgements

Part of the research of the second author was carried out at the Department of Industrial Engineering, Bilkent University.

## References

- W. P. Adams and H. D. Sherali. A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science*, 32(10):1274–1290, 1986.
- W. P. Adams, M. Guignard, P. M. Hahn, and W. L. Hightower. A level-2 reformulation–linearization technique bound for the quadratic assignment problem. *European Journal of Operational Research*, 180(3):983–996, 2007.
- L. E. Agustín-Blas, S. Salcedo-Sanz, E. G. Ortiz-García, A. Portilla-Figueras, Á. M. Pérez-Bellido, and S. Jiménez-Fernández. Team formation based on group technology: A hybrid grouping genetic algorithm approach. *Computers & Operations Research*, 38(2):484–495, 2011.
- 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*, pages 839–848. ACM, 2012.
- E. Avgerinos and B. Gokpinar. Team familiarity and productivity in cardiac surgery operations: The effect of dispersion, bottlenecks, and task complexity. *Manufacturing & Service Operations Management*, 2016.
- A. Baykasoglu, T. Dereli, and S. Das. Project team selection using fuzzy optimization approach. *Cybernetics and Systems: An International Journal*, 38(2):155–185, 2007.
- M. S. Bazaraa and J. J. Goode. A cutting-plane algorithm for the quadratic set-covering problem. *Operations Research*, 23(1):150–158, 1975.
- D. Bergman. An exact algorithm for the quadratic multiknapsack problem with an application to event seating. *INFORMS Journal on Computing*, 2019.
- A. Bhowmik, V. S. Borkar, D. Garg, and M. Pallan. Submodularity in team formation problem. In *SDM*, pages 893–901. SIAM, 2014.
- A. Billionnet and F. Calmels. Linear programming for the 0–1 quadratic knapsack problem. *European Journal of Operational Research*, 92(2):310–325, 1996.
- B. H. Boon and G. Sierksma. Team formation: Matching quality supply and quality demand. *European Journal of Operational Research*, 148(2):277–292, 2003.

- A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11(2):125–137, 1999.
- Centric Digital. What is taas (team as a service) and why is it becoming so popular? Retrieved April 6, 2017, <https://centricdigital.com/blog/digital-trends/what-is-team-as-a-service/>, 2016.
- S.-J. Chen and L. Lin. Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering. *IEEE Transactions on Engineering Management*, 51(2):111–124, 2004.
- E. de Klerk, R. Sotirov, and U. Truetsch. A new semidefinite programming relaxation for the quadratic assignment problem and its computational perspectives. *INFORMS Journal on Computing*, 27(2):378–391, 2015.
- C. Dorn and S. Dustdar. Composing near-optimal expert teams: a trade-off between skills and connectivity. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 472–489. Springer, 2010.
- B. Escoffier and P. L. Hammer. Approximation of the quadratic set covering problem. *Discrete Optimization*, 4(3-4):378–386, 2007.
- J. A. Espinosa, S. A. Slaughter, R. E. Kraut, and J. D. Herbsleb. Familiarity, complexity, and team performance in geographically distributed software development. *Organization Science*, 18(4):613–630, 2007.
- A. Farasat and A. G. Nikolaev. Social structure optimization in team formation. *Computers & Operations Research*, 74:127–142, 2016.
- E. L. Fitzpatrick and R. G. Askin. Forming effective worker teams with multi-functional skill requirements. *Computers & Industrial Engineering*, 48(3):593–608, 2005.
- F. D. Fomeni, K. Kaparis, and A. N. Letchford. A cut-and-branch algorithm for the quadratic knapsack problem. Technical report, Lancaster University Management School, UK, 2014.
- R. Fortet. Applications de l'algebre de boole en recherche opérationelle. *Revue Française de Recherche Opérationelle*, 4(14):17–26, 1960.
- A. Gajewar and A. D. Sarma. Multi-skill collaborative teams based on densest subgraphs. In *Proceedings of the 2012 SIAM International Conference on Data Mining*. SIAM, 2012.
- D. A. Guimarães, A. S. da Cunha, and D. L. Pereira. Semidefinite programming lower bounds and branch-and-bound algorithms for the quadratic minimum spanning tree problem. *European Journal of Operational Research*, 280(1):46–58, 2020.
- J. H. Gutiérrez, C. A. Astudillo, P. Ballesteros-Pérez, D. Mora-Melià, and A. Candia-Véjar. The multiple team formation problem using sociometry. *Computers & Operations Research*, 75:150–162, 2016.
- P. M. Hahn, Y.-R. Zhu, M. Guignard, W. L. Hightower, and M. J. Saltzman. A level-3 reformulation-linearization technique-based bound for the quadratic assignment problem. *INFORMS Journal on Computing*, 24(2):202–209, 2012.

- M. Hoegl and H. G. Gemuenden. Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence. *Organization Science*, 12(4):435–449, 2001.
- R. S. Huckman, B. R. Staats, and D. M. Upton. Team familiarity, role experience, and performance: Evidence from indian software services. *Management Science*, 55(1):85–100, 2009.
- P. Jaccard. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50, 1912.
- R. Jones. *Working Virtually: Challenges of Virtual Teams: Challenges of Virtual Teams*. IGI Global, 2005.
- M. Kargar and A. An. Discovering top-k teams of experts with/without a leader in social networks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 985–994. ACM, 2011.
- M. Kargar, A. An, and M. Zihayat. Efficient bi-objective team formation in social networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 483–498. Springer, 2012.
- 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*, pages 467–476. ACM, 2009.
- E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657–690, 2007.
- A. Majumder, S. Datta, and K. Naidu. Capacitated team formation problem on social networks. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1005–1013. ACM, 2012.
- R. Martinelli and C. Contardo. Exact and heuristic algorithms for capacitated vehicle routing problems with quadratic costs structure. *INFORMS Journal on Computing*, 27(4):658–676, 2015.
- H. Mittelman and J. Peng. Estimating bounds for quadratic assignment problems associated with hamming and manhattan distance matrices based on semidefinite programming. *SIAM Journal on Optimization*, 20(6):3408–3426, 2010.
- P. Pandey and A. P. Punnen. On a linearization technique for solving the quadratic set covering problem and variations. *Optimization Letters*, 11(7):1357–1370, 2017.
- W. D. Pisinger, A. B. Rasmussen, and R. Sandvik. Solution of large quadratic knapsack problems through aggressive reduction. *INFORMS Journal on Computing*, 19(2):280–290, 2007.
- J. Povh and F. Rendl. Copositive and semidefinite relaxations of the quadratic assignment problem. *Discrete Optimization*, 6(3):231–241, 2009.
- A. P. Punnen, P. Pandey, and M. Friesen. Representations of quadratic combinatorial optimization problems: A case study using quadratic set covering and quadratic knapsack problems. *Computers & Operations Research*, 112:104769, 2019.
- C. Rodrigues, D. Quadri, P. Michelon, and S. Gueye. 0-1 quadratic knapsack problems: an exact approach based on a t-linearization. *SIAM Journal on Optimization*, 22(4):1449–1468, 2012.

- R. Saxena and S. Arora. A linearization technique for solving the quadratic set covering problem. *Optimization*, 39(1):33–42, 1997.
- X. Wang, Z. Zhao, and W. Ng. A comparative study of team formation in social networks. In *International conference on database systems for advanced applications*, pages 389–404. Springer, 2015.
- H. Wi, S. Oh, J. Mun, and M. Jung. A team formation model based on knowledge and collaboration. *Expert Systems with Applications*, 36(5):9121–9134, 2009.
- A. Zakarian and A. Kusiak. Forming teams an analytical approach. *IIE transactions*, 31(1):85–97, 1999.
- L. Zhang and X. Zhang. Multi-objective team formation optimization for new product development. *Computers & Industrial Engineering*, 64(3):804–811, 2013.