# Team as a Service: Team Formation on Social Networks

Nihal Berktaş

Department of Industrial Engineering,Bilkent University, Ankara, Turkey, nihal.berktas@bilkent.edu.tr

Hande Yaman

Present address: ORSTAT, Faculty of Economics and Business, KU Leuven, Belgium, handeyaman@kuleuven.be

Department Industrial Engineering, Bilkent University, Ankara, Turkey

**Abstract**

Team as a Service (TaaS) is a new outsourcing service that enables on-demand creation and management of distributed teams for fast growing companies. The companies that use the TaaS model form a team according to the needs of a given project and provide managerial service throughout. Motivated by this new application, we study the Team Formation Problem (TFP) in which the quality and cost of communication is taken into account using the proximity of the potential members in a social network. Given a set of required skills, TFP aims to construct a capable team that can communicate and collaborate effectively. We study TFP with two measures for communication effectiveness: sum of distances and diameter. We compare the quality of solutions using various performance measures. Our experiments indicate the following: First, considering only the sum of distances or the diameter as the communication cost may yield solutions that perform poorly in other performance measures. Second, the heuristics in the literature find solutions that are significantly far from optimal whereas a general purpose solver is capable of solving small and medium size instances to optimality. To find solutions that perform well on several performance measures, we propose the diameter constrained TFP with sum of distances objective. To solve larger sizes, we propose a novel branch and bound approach: we formulate the problem as a constrained quadratic set covering problem, use the reformulation-linearization technique and relaxation to be able to decompose it into a series of linear set covering problems and impose the relaxed constraints through branching. Our computational experiments show that the algorithm is capable of solving large sizes that are intractable for the solver.

*keywords*: team as a service, 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 dynamics of the team. There are numerous factors effecting the performance of a team such as the size, diversity, personality and familiarity and there is abundant literature on this topic in the fields of management science, organization science and

psychology as summarized in the surveys of Cohen and Bailey (1997) and Stewart (2006). All these factors determine the effectiveness of the collaboration and consequently the quality of teamwork. Hoegl and Gemuenden (2001) regard communication as the most elementary component of their TeamWork Quality concept, which is developed to measure the collaboration in teams. The study of Jones (2005) is among others that emphasize the significance of the communication in teamwork, especially in virtual teams. Considering the fact that the number of people working remotely in the U.S. increased by 115% in the last decade (FlexJobs, 2017), more teams are expected to work remotely and this conversion to virtual teams amplifies the importance of communication.

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 (TaaS)". The companies that use the TaaS model build a team according to the needs of a given project and provide managerial service throughout. Codibly[1] is one of the companies that uses TaaS in software development. According to Centric Digital (2016), a consultancy firm which is on the Inc. 5000' fastest-growing companies list for the last couple of years, TaaS provides 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. This new concept and the importance of communication in teamwork are the main motivations behind this study on the *Team Formation Problem* (TFP), which is defined as finding a group of people who posses a required set of skills and can function as a team by means of effective communication.

The first studies related to TFP mostly focus on the technical performances of the individuals in the assigned tasks and the cost of hiring those individuals. One of the related earliest studies in the operations research (OR) literature is by Zakarian and Kusiak (1999) on constructing multi-functional teams for product design and development. Boon and Sierksma (2003) give a matching model for sports team formation problem where candidate players and positions are matched to maximize sum of player-position weights. Agustín-Blas et al. (2011) study the problem of building teaching groups in a university by rearranging a matrix that represents skill levels of people for the resources. There is a minimum required knowledge level for each team member and also for the whole team. Their objective is to maximize the mean knowledge of the teams. In these studies the communication among the team members is not taken into consideration.

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 considerations 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.

The solution techniques suggested in the above studies are either not designed for real size data or they are heuristic approaches. Furthermore, there is no general method to incorporate communication effectiveness into the problem; some studies focus on personalities only and some represent relations as binary. Motivated by the increasing popularity of virtual teams and TaaS, we integrate communication effectiveness using social networks. To the best of our knowledge, the study by Wi et al. (2009) is the first one using social networks for team formation in the OR literature. The authors form a network using fuzzy familiarity scores among candidates using collaboration data. They formulate a multi-objective nonlinear program whose objective consists of weighted sum of the performance, the familiarity and the size of the team and propose a genetic algorithm. More recently, Farasat and Nikolaev (2016) use edge, 2-star, 3-star and triangle network structures to measure

[1] www.codibly.com

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 authors formulate the problem as an integer program but report memory problems for instances having more than 16 people and 5 teams. An algorithm based on depth neighborhood search is compared with a genetic algorithm and suggested as the solution technique.

Apart from the works of Wi et al. (2009) and Farasat and Nikolaev (2016), social networks are used in TFP mainly in the knowledge discovery and data mining (KDD) 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 on social network analysis. Some examples of these social networks are community-based question answering websites such as StackOverFlow[2] and Quora[3], software development platforms like Github[4], not to mention the online community of Wikipedia. Although these communities usually consist of self-organizing teams, they are actually great platforms for the firms looking for a team of experts to hire. Seizing this opportunity, Stackoverflow offers a service to assist customers in finding and hiring developers.

In a social network, nodes correspond to individuals and edges among them represent the relations. The weight of an edge can be interpreted as the effort required for those people to communicate effectively as team members. In other words, it is the cost of communication between the people connected by that edge. The existence of such a network is an acceptable assumption since either it really exists as in the above online examples or people are part of an actual organization and their relations can be modeled via a social network. In the first case the previous collaborations of people can be used to estimate the communication cost. In the second case, besides previous work statistics, results of questionnaires and personality tests mentioned above can also be incorporated in the calculation of the communication costs. Hence the social network of candidates enables us to devise measures for the communication cost of a team. Our aim is to find a capable team with minimum cost. Contrary to the early studies of TFP which completely ignore communication, we do not maximize the expertise level of the team. The technical capability of the team is ensured in the constraints using a binary skill matrix built by considering minimum expertise levels.

In this study we give integer programming models for various team formation problems and compare the performance of their optimal solutions using different communication cost measures. Based on this comparison, we propose to use the sum of distances as the communication cost in the objective and to bound the diameter to eliminate teams with bottleneck pairs. This variant can be modeled as a quadratic set covering problem with packing constraints. We observe that small and medium size instances can be solved using a general purpose solver, however memory problems occur for large instances. We present a novel branch and bound algorithm to solve these instances. This algorithm can be used to solve 0-1 integer problems with a quadratic objective function without any condition on the sign of objective function coefficients. It applies the reformulation-linearization technique partially to obtain a formulation with redundant constraints. Then it relaxes a set of constraints so that the model can be decomposed into a series of linear set covering problems. Finally, the relaxed constraints are imposed through branching. The algorithm is tested on a large set of instances created using IMDB and DBLP data. All information about the instances are available in our Github repository[5]. The results show that the algorithm is very effective in solving large size instances. In addition, as it decomposes, it is capable of solving instances for which the solver runs out of memory during model generation. We also test

[2] www.stackoverflow.com
[3] www.quora.com
[4] github.com
[5] https://github.com/nihalberktas/TFP-data

a branch and cut algorithm where the constraints that grow quadratically in the size of the problem are added using a lazy constraint callback. Our branch and bound algorithm also outperfoms this approach.

The remaining part of the paper is organized as follows: In the next section we introduce the communication cost measures used in the literature, discuss how they are related to team familiarity and provide a list of algorithms developed to solve TFP with different cost measures. In Section 3 we formally define TFP and provide mathematical models for three variants. We present our branch and bound algorithm in Section 4. In Section 5, we first introduce our datasets and explain our instance generation method. Then we present the results of an extensive computational study. We summarize our results and conclude in Section 6.

## 2 Communication cost measures

In this section, we investigate different communication cost measures used in TFP. In the presence of a social network, the communication cost between two potential team members is given by the weight of the edge connecting them. In the literature, the most common way to determine edge weights is through the collaboration information on the social network. 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 $1 - (|P_i \cap P_j|/|P_i \cup P_j|)$. More collaboration leads to lower edge weights meaning that the communication is less costly.

As opposed to the case of pairwise communication cost, defining overall communication cost of a team is much less straightforward and therefore different functions are used in the literature. Lappas et al. (2009) is the first one to quantify communication cost in TFP using a social network. The authors give two definitions for the communication cost of a team: first as the diameter of the subgraph induced by the team members, where the diameter is the maximum of the shortest path lengths between any two members, and second as the cost of the minimum spanning tree on this subgraph. 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. This is a skill-based definition because if a team member is responsible for $k$ different skills then his distance to the leader is counted $k$ times in the cost calculation. Bottleneck cost is defined by Majumder et al. (2012) as the weight of the maximum weighted edge on the spanning tree of the subgraph formed by the team. 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.

In these TFP studies approximation algorithms, greedy heuristics and metaheuristics are developed and tested. In Table 1 we list the algorithms designed to minimize the most commonly used communication cost functions which are diameter (CC-D), steiner or spanning tree (CC-S), sum of distances (CC-SD) and leader distance (CC-L). As the algorithms are based on different communication cost functions, comparing their performances is not straightforward. In their comparative study, Wang et al. (2015) implement these TFP algorithms and compare them in terms of each communication cost function. The authors discuss the advantages of each algorithm and results of using a specific communication cost function.

The comparative study of Wang et al. (2015) shows that there is no single algorithm that performs best in terms of all communication cost functions. This is expected since each algorithm is designed to minimize a

Table 1: Team formation algorithms for different communication cost functions

| | Communication Cost | | | |
|---|---|---|---|---|
| Algorithm | CC-D | CC-S | CC-SD | CC-L |
| *RarestFirst* (Lappas et al., 2009) | ✓ | | | |
| *EnSteiner* (Lappas et al., 2009) | | ✓ | | |
| *MinSD* (Kargar and An, 2011) | | | ✓ | |
| *MinLD* (Kargar and An, 2011) | | | | ✓ |
| *MinDiaSol* (Majumder et al., 2012) | ✓ | | | |
| *MinAggrSol* (Majumder et al., 2012) | | ✓ | | |
| *MCC, ItReplace* (Kargar et al., 2012) | | | ✓ | |
| *LBRadius* (Anagnostopoulos et al., 2012) | ✓ | | | |
| *LBSteiner* (Anagnostopoulos et al., 2012) | | ✓ | | |

specific type of communication cost. However the results indicated that solutions obtained by sum of distances based algorithms are usually have good performance on other cost functions as well. Therefore sum of distances is claimed to be the best measurement for the communication cost. The authors also point out that communication cost based on diameter, spanning tree and leader distance are very sensitive to changes. For instance, adding one more person to the team may increase the diameter cost substantially or it may decrease the leader distance significantly. In their follow-up study, Wang et al. (2016) again state that sum of distances cost is more robust to changes in the team's network. When one-to-one communication has utmost importance for the task, however, the authors suggest that an algorithm based on diameter cost should be employed.

As the weight of an edge is often calculated using the number of collaborations, minimizing the sum of the distances amounts to maximizing the average familiarity of the team. There are empirical studies in the literature indicating the positive effects of team familiarity on the performance of the team. Huckman et al. (2009) take the quality of output and the adherence to the project schedule as performance measures in their analysis of the teams working in a software service company. The results of their statistical model indicate a positive and significant relation between team familiarity and operational performance. Acknowledging the positive effect of team and task familiarity on team performance, Espinosa et al. (2007) investigate whether the significance of their effect differ in complex circumstances. Analyzing data on software development teams of a telecommunications firm, the authors find that team familiarity is more beneficial for the teams whose coordination is more challenging due to their size or dispersion. Avgerinos and Gokpinar (2016) study the effects of team familiarity on productivity of surgical teams. Their findings show that the benefit of being familiar with each other enhances as the task gets more complex. Moreover, the performance analysis in the study suggests that the bottleneck pair, i.e., the pair with lowest familiarity, significantly reduces the team productivity.

In the light of the results of the studies in the literature, we choose to use sum of distances as the measure for communication cost of the team due to its robustness and its direct relation to the average familiarity. However, as there is also significant evidence that the diameter is critical for the success of the team, we also investigate the problem with diameter objective. Finally, we introduce the diameter constrained variant where both measures are considered simultaneously.

# 3 Mathematical models

In this section we first formally define the team formation problem and provide mathematical models for three variants.

Let $G = (N, E)$ be the weighted undirected social graph of the candidates and $K$ be the set of skills required for a given task. Let $c_{ij}$ denote the weight of edge $\{i, j\} \in E$ and $p_{ij}$ denote the length of the shortest path between $i$ and $j$. By using the shortest path distances we are able to quantify the communication cost between individuals who have not collaborated before but can be connected via other individuals in the network. If the team members are required to be connected directly by an edge, we can enforce this by letting $p_{ij} = c_{ij}$ for each edge $\{i, j\} \in E$ and $p_{ij} = \infty$ if edge $\{i, j\}$ does not exist. Capabilities of individuals are represented using binary parameters: $a_{ik}$ is equal to one if person $i$ has skill $k$ and zero otherwise. We assume that the task type is basic, i.e., having one person $i$ with $a_{ik} = 1$ is sufficient for a skill $k$ to be covered. The team formation problem is finding a group of people who posses the required set of skills and have minimum communication cost.

In the sequel, we provide models for TFP with different communication cost measures. We use the following variables in these models. 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 also define $z_{ij} = y_i y_j$ for all $i, j \in N$ with $i < j$.

## 3.1 TFP with sum of distances objective (TFP-SD)

TFP with sum of distance objective can be formulated as the following quadratic set covering problem:

$$\min \sum_{i,j \in N: i<j} p_{ij} y_i y_j \tag{1}$$

s.t.

$$\sum_{i \in N} a_{ik} y_i \geq 1 \quad \forall k \in K, \tag{2}$$

$$y_i \in \{0, 1\} \quad \forall i \in N. \tag{3}$$

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 objective function is the sum of communication costs of team members.

One of the earliest studies on the quadratic set covering problem is by Bazaraa and Goode (1975). This problem has attracted very little attention as opposed to the quadratic assignment problem (QAP), which has a vast literature (Loiola et al., 2007).

We can linearize this quadratic objective to obtain the following integer model:

$$\min \sum_{i,j \in N: i<j} p_{ij} z_{ij} \tag{4}$$

s.t. (2) and (3)

$$z_{ij} \geq y_i + y_j - 1 \quad \forall i, j \in N : i < j \tag{5}$$

$$z_{ij} \leq y_i \quad \forall i, j \in N : i < j \tag{6}$$

$$z_{ij} \leq y_j \quad \forall i, j \in N : i < j \tag{7}$$

$$z_{ij} \geq 0 \quad \forall i, j \in N : i < j \tag{8}$$

In this formulation, in addition to the covering constraints, we have constraints (5)-(8) to force $z_{ij}$ to be one when both $y_i$ and $y_j$ are equal to one, and to be zero otherwise.

### 3.2 TFP with diameter objective (TFP-D)

To minimize the diameter of the team we define and minimize variable $D$ in the following model:

$$\min D \tag{9}$$

s.t. (2) and (3),

$$D \geq p_{ij}(y_i + y_j - 1) \quad \forall i, j \in N : i < j. \tag{10}$$

Here constraints (10) together with the objective, guarantee that $D$ is equal to the largest one of the pairwise shortest paths.

### 3.3 Diameter constrained TFP with sum of distances objective (DC-TFP-SD)

In this version of TFP we minimize the sum of distances and impose constraints to bound the diameter. To this end, we define 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:

$$\min \sum_{i,j \in N: i<j} p_{ij} y_i y_j$$

s.t. (2) and (3),

$$y_i + y_j \leq 1 \quad \forall \{i, j\} \in C. \tag{11}$$

The only difference of this model with TFP-SD is the family of packing constraints (11), which forbid conflicting pairs in the team. Similarly we can use variables $z_{ij}$ for all $i, j \in N$ with $i < j$ to linearize the objective

and write the following integer model for DC-TFP-SD:

$$\min \sum_{i,j \in N: i<j} p_{ij} z_{ij}$$

s.t. (2), (3), (5)-(8) and (11).

# 4 Branch and bound algorithms

Despite the advances in general purpose solvers for mixed integer and quadratic programs, their abilities are limited especially when the problem size is large. Therefore we propose branch and bound algorithms for both TFP-SD and DC-TFP-SD that are capable of solving larger instances to optimality. The relaxation we use in these algorithms is an integer program that enables decomposition. First we explain how we obtain a reformulation for TFP-SD. We mention the related work in the literature on quadratic problems and emphasize the differences of our problem and solution approach. We explain how the reformulation is relaxed so that it lends itself to decomposition. Then we present our branching rule and the way we calculate the bounds. We illustrate the use of the algorithm for TFP-SD on a small example. Finally we explain how the algorithm is adjusted for DC-TFP-SD and factors related to its efficiency for any 0-1 quadratic program.

## 4.1 Branch and bound algorithm for TFP-SD

For ease of decomposition, we define variable $z_{ij}$ for all $i, j \in N : i \neq j$ 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 the skill $k$. Hence the constraint implies that if $j$ is in the team then at least one person having skill $k$ must be in the team. We can rewrite these as follows:

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

We add the new constraints to our previous model and make slight changes to obtain the following model:

$$\min \sum_{i \in N} \sum_{j \in N \setminus \{i\}} p_{ij} z_{ij} \tag{12}$$

s.t.

$$\sum_{i \in N} a_{ik} y_i \geq 1 \quad \forall k \in K, \tag{13}$$

$$\sum_{i \in N \setminus \{j\}} a_{ik} z_{ij} \geq y_j \quad \forall k \in K, j \in N : a_{jk} = 0, \tag{14}$$

$$z_{ij} \leq y_j \quad \forall i, j \in N : i \neq j, \tag{15}$$

$$z_{ij} = z_{ji} \quad \forall i, j \in N : i < j, \tag{16}$$

$$z_{ij} \geq y_i + y_j - 1 \quad \forall i, j \in N : i < j, \tag{17}$$

$$y_i \in \{0, 1\} \quad \forall i \in N, \tag{18}$$

$$z_{ij} \geq 0 \quad \forall i, j \in N : i \neq j. \tag{19}$$

Notice that due to the change of use of $z_{ij}$, the objective function value will be twice the one of the previous model.

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 applied for QAP. 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 (QKP) by Billionnet and Calmels (1996), Caprara et al. (1999), Pisinger et al. (2007), Fomeni et al. (2014). Billionnet and Calmels (1996) use continuous relaxation of the reformulation to find upper bounds. Caprara et al. (1999) propose a branch and bound algorithm in which a continuous Lagrangian relaxation of the reformulation is solved. Later Pisinger et al. (2007) improve the Lagrangian phase of this work to be able to solve larger instances. Fomeni et al. (2014) use inequalities obtained by RLT in their cut and branch algorithm. The main distinction between the reformulation of these problems and ours is that constraints of type (17) are redundant in the reformulation of QAP due to problem structure and QKP due to the assumption of non-negative objective coefficients while they are necessary in our reformulation. Therefore in all the studies that apply Lagrangian relaxation to the reformulation, only constraints of type (16) are dualized and the resulting relaxation can be solved efficiently. Moreover, in the case of QAP the relaxed problem can be solved without binary restrictions due to problem structure and in the case of QKP Caprara et al. (1999) choose to impose the binary restrictions in the branch and bound so as to exploit polynomial time solvability of continuous knapsack.

In his later study Caprara (2008) calls a quadratic program monotone if constraints of type (17) are redundant in the reformulation. Thus, QAP and QKP with non-negative objective coefficients are monotone problems whereas ours is non-monotone. To the best of our knowledge, this is the first study that uses RLT constraints for the reformulation of a non-monotone 0-1 quadratic problem. As we explain in detail below, our problem decomposes in the absence of constraints (16) and (17). So for the monotone quadratic problems as constraints (17) are redundant only constraints (16) are required to be relaxed for decomposition. And all related studies in the literature prefer using Lagrangian relaxation and dualize constraints (16). Then the linear relaxation of the remaining model is solved and the binary requirements are imposed by branching if necessary. So our second

major difference with the literature is that we are not using any linear relaxation but solving integer models.

We obtain a relaxation of the reformulation by removing constraints (16) and (17). In this relaxation, for a given vector $y \in \{0,1\}^{|N|}$ such that $\sum_{i \in N} a_{ik} y_i \geq 1$ for all $k \in K$, the problem of computing the best $z$ decomposes. In particular, when $y_j$ equals to 0 so are the variables $z_{ij}$ for $i \neq j$ due to constraints (15). When $y_j$ equals to 1, we can find the best values of $z_{ij}$'s by solving the following subproblem:

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

s.t.

$$\sum_{i \in N \setminus \{j\}} a_{ik} z_{ij} \geq 1 \quad \forall k \in K : a_{jk} = 0,$$

$$z_{ij} \in \{0,1\} \quad \forall i \in N \setminus \{j\}.$$

Let $\bar{z}_{ij}$ be an optimal solution of the above problem and $\bar{p}_j$ be the optimal value. The $\bar{p}_j$ values are used in the following set covering problem, which is referred to as the master:

$$\min \sum_{j \in N} \bar{p}_j y_j$$

s.t.

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

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

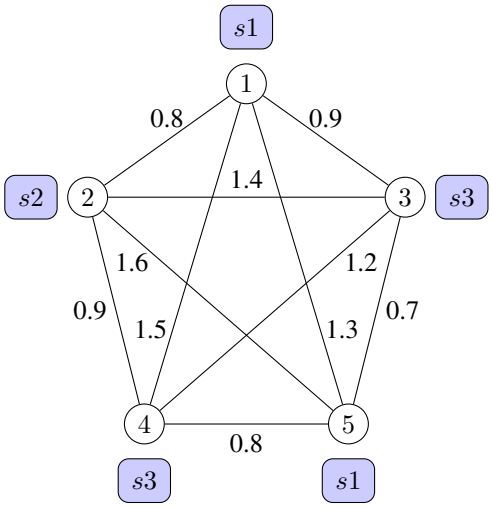
An optimal solution $y^*$ of the master is also optimal for the relaxation and the optimal values of $z_{ij}$'s are equal to $z^*_{ij} = y^*_j \bar{z}_{ij}$. Hence we can solve the relaxation by solving $|N| + 1$ set covering problems. Notice that the aim of the subproblem $j$ is actually finding the closest people to $j$ for the skills that $j$ does not have. Then $\bar{p}_j$ gives a lower bound on the cost induced by person $j$. Using these lower bounds we get a lower bound on the cost of an optimal team.

Since constraints (16) are relaxed, $z^*_{ij}$ may not be equal to $z^*_{ji}$. Furthermore, we might get a solution where $y^*_i = y^*_j = 1$ although $z^*_{ij}$ or $z^*_{ji}$ or both are equal to zero, since we relaxed constraints (17). Therefore we branch on such $z_{ij}$ variables by forcing them to zero at one node and to one at the other. We choose the $\{i,j\}$ pair to branch on using the following rule: for all $\{i,j\}$ pairs such that $y^*_i = y^*_j = 1$ first we check whether there exists a pair with $\bar{z}_{ij} = \bar{z}_{ji} = 0$. We call these pairs type one and if one exists, we branch on the first such pair found. Else we look for a type two pair, i.e., a pair $\{i,j\}$ with $y^*_i = y^*_j = 1$, $\bar{z}_{ij} = 1$ and $\bar{z}_{ji} = 0$.

In a regular branch and bound scheme branching on these $\{i,j\}$ pairs amounts to the following: 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. To do that at the first node we need to solve each subproblem $n \in N \setminus \{i,j\}$ with the additional constraint $z_{in} + z_{jn} \leq 1$, subproblem $i$ with $z_{ji} = 0$, subproblem $j$ with $z_{ij} = 0$ and then the master problem with $y_i + y_j \leq 1$. At the second node we should add constraints $z_{in} = 1$ and $z_{jn} = 1$ to the subproblem $n \in N \setminus \{i,j\}$, $z_{ji} = 1$ to subproblem $i$, $z_{ij} = 1$ to subproblem $j$ and then we need to solve the master with $y_i = 1$ and $y_j = 1$. This implementation requires solving $|N| + 1$ integer programs at each node.

To decrease the number of problems solved at the nodes, we adopt a different approach and solve only the subproblems related to the branch pairs. More precisely at the first node, we solve subproblem $i$ by enforcing $z_{ji} = 0$ and subproblem $j$ by enforcing $z_{ij} = 0$. We update $\bar{p}_i$ and $\bar{p}_j$ and then the master problem is solved with the additional constraint $y_i + y_j \leq 1$. At the second node, again we only solve the subproblems $i$ and $j$ with the additional constraints $z_{ji} = 1$ and $z_{ij} = 1$, respectively and then the master with $y_i = 1$ and $y_j = 1$. In this approach we solve at most three integer programs at each node since if the current $\bar{z}$ already satisfies the new constraint we do not need to solve the subproblem again. The lower bounds we get at each node will not be as good as the lower bounds of the regular approach and consequently the branch and bound tree may be larger. However, our preliminary analysis shows that this approach is faster since the time spent at each node is significantly less.

There are two ways to update the upper bound in our algorithm; via the subproblems and the master problem. For each subproblem $j$ let $N_j = \{i \in N : \bar{z}_{ij} = 1\}$. Due to constraint (14), $N_j \cup \{j\}$ is a feasible solution to the problem and consequently the sum of distances of this team is an upper bound. Similarly the solution of the master problem, $N' = \{i \in N : y^*_i = 1\}$, is also feasible and gives another candidate for a new incumbent. At each node, after solving the subproblems and the master problem we update the upper bound and the incumbent solution. We follow the best-first search rule for choosing the node to solve.




| $j$ | $i : \bar{z}_{ij} = 1$ | $\bar{p}_j$ | $ub$ |
|---|---|---|---|
| 1 | 2,3 | 1.7 | 3.1 |
| 2 | 1,4 | 1.7 | 3.2 |
| 3 | 2,5 | 2.1 | 3.7 |
| 4 | 2,5 | 1.7 | 3.3 |
| 5 | 2,3 | 2.3 | 3.7 |

Figure 1: Example network and optimal solution of subproblem $j$ for each node $j$ in the network

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 1. 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 specified next to the nodes. The initial step of the algorithm is to solve the subproblems. The results obtained by solving five subproblems are given in a table also in Figure 1. For example, the first row shows that the optimal solution of subproblem 1 is $\bar{z}_{21} = \bar{z}_{31} = 1$ with optimal value $\bar{p}_1 = 1.7$. The team consisting of people 1, 2 and 3 has a cost 3.1, therefore it is the incumbent.

The branch and bound tree is illustrated in Figure 2 and the solution obtained by solving the master at each node is presented in a table next to the node. The optimal $y$ vector at the root (0) node is $y^* = (1, 1, 0, 1, 0)$ which means people 1, 2 and 4 are selected to be in the team. Then the first lower bound becomes half of the

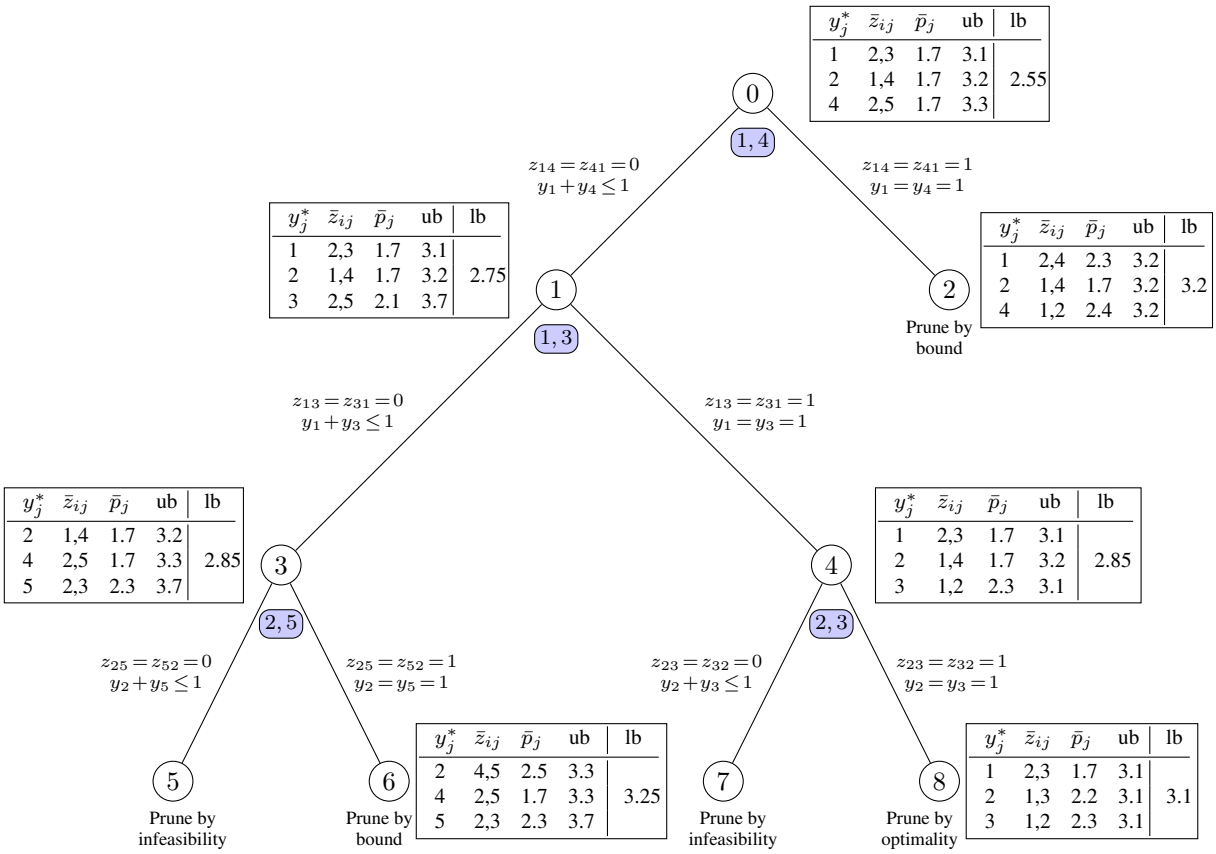



Figure 2: The branch and bound tree

sum $(\bar{p}_1 + \bar{p}_2 + \bar{p}_4)$ which is 2.55. When we look at the $\bar{z}$ values, we see that $\bar{z}_{12} = \bar{z}_{21} = 1$ and $\bar{z}_{24} = \bar{z}_{42} = 1$ however $\bar{z}_{14} = \bar{z}_{41} = 0$ although $y_1^* = 1$ and $y_4^* = 1$. In other words the master problem puts people 1 and 4 in the team together but according to the corresponding subproblems they are not teammates. Therefore $\{1, 4\}$ is chosen to be the branch pair. Note that it is a type one pair.

At the odd numbered nodes we ensure that the people in the branch pair are not teammates and at the even ones they are forced to be in the team together. At node (1), since $\{1, 4\}$ is a type one pair and it is an odd node, only the master problem is solved adding the constraint $y_1 + y_4 \leq 1$ and person 4 is replaced by person 3 in the solution of the master. At node (2) subproblems 1 and 4 are solved by forcing $\bar{z}_{41} = 1$ and $\bar{z}_{14} = 1$, respectively. After updating the $\bar{p}_1$ and $\bar{p}_4$ values, we solve the master by adding constraints $y_1 = 1$ and $y_4 = 1$. Node (2) is pruned by bound since its lower bound which is 3.2 is larger than the current upper bound. The algorithm continues with node (1) and in this case the branch pair is a type two pair since $\bar{z}_{31} = 1$ and $\bar{z}_{13} = 0$. At node (3) we solve subproblem 1 by enforcing $\bar{z}_{31} = 0$ and $\bar{z}_{41} = 0$, then the master is solved with additional constraints $y_1 + y_3 \leq 1$ and $y_1 + y_4 \leq 1$. At node (4) we need to solve subproblem 3 by adding constraint $z_{13} = 1$ and then the master problem using the new $\bar{p}_3$ and constraints $y_1 = 1$, $y_3 = 1$ and $y_1 + y_4 \leq 1$. Nodes (3) and (4) have the same lower bound, we can continue with either of them. In this example node (3) is branched with pair $\{2, 5\}$. At node (5) subproblem 5 is solved by forcing $\bar{z}_{25} = 0$. Then the master turns out to be infeasible with the constraints $y_2 + y_5 \leq 1$, $y_1 + y_3 \leq 1$ and $y_1 + y_4 \leq 1$ and the node is pruned. 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.

The efficiency of the algorithm stems from two points. First is the quality of the upper bound. Our computational analysis shows that the upper bound at the root node is very close to the optimal value in general. Second is the low solution times for sub and master problems despite the fact that they are integer programs. Since we solve at most three set covering problems at each node, the nodes are processed very quickly. Here we would like the mention that it is possible to solve all subproblems at each node instead of solving only the ones in the branch pair. For example, at node (1) we could solve subproblems 2, 3 and 5 by adding the constraint $z_{1j} + z_{4j} \leq 1$ for $j = 2, 3, 5$, respectively. This approach gives stronger lower bounds but it increases the time spent at nodes to a great extent.

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

The branch and bound algorithm can be easily extended to solve DC-TFP-SD. Recall that we defined $C$ to be the set of pairs whose communication cost exceeds the allowed diameter and add constraints (11) to eliminate teams with those pairs. Now similarly we define $C_j$ as the set of people who are in conflict with $j$. We add $\sum_{i \in C_j} z_{ij} = 0$ to subproblem $j \in N$. The algorithm works in the same way as described above with the following exception: due to the diameter constraint, $N_j \cup \{j\}$ for all $j \in N$ and $N'$ found at each node might not be feasible, where $N_j = \{i \in N : \bar{z}_{ij} = 1\}$ and $N' = \{i \in N : y_i^* = 1\}$. Thus for DC-TFP-SD, we need to check whether they include a pair in conflict. If not then they can be used to update the incumbent.

Recall that we have the packing constraints (11) in the MIP formulation of DC-TFP-SD. When we use these in the master problem, we may get better bounds and $N'$ would be feasible. However, these packing constraints enlarge the master problem and increase the solution times. Therefore we exclude them. This does not harm the exactness of the algorithm since the additional constraint in the subproblem and our branching scheme ensure that the diameter constraints are satisfied when the algorithm terminates.

Our branch and bound scheme can be used as an exact solution method for any 0-1 quadratic program with linear constraints without any sign restriction in the objective function coefficients. The efficiency of our algorithm mainly depends on the structure of the sub and master problems. These are integer programs that are solved at each node of the tree; so for the algorithm to terminate quickly these problems have to be solved quickly, which turned out to be the case with set covering problems.

## 5 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 results of our preliminary experiments conducted with various solution procedures. The performance analysis of our branch and bound algorithm and its comparison with the mathematical models are given at the end of the section.

### 5.1 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 IMDB and DBLP datasets in our computations.

IMDB database is used by Anagnostopoulos et al. (2012) and Kargar and An (2011). We create our instances using the same part of the IMDB 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[6]. 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 worked together in a movie and the weight of the edge $\{i, j\}$ is equal to $1 - (|P_i \cap P_j|/|P_i \cup P_j|)$ where $P_i$ is the set of movies of actor $i$.

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. To assign weights to the edges, we use the same approach as in IMBD dataset: we let $P_i$ be the set of papers $i$ has authored and the weight of the edge $\{i, j\}$ be $1 - (|P_i \cap P_j|/|P_i \cup P_j|)$. Then we compute the shortest path lengths between all pairs.

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 $m$. All information about the instances are available in our Github repository[7].

## 5.2 Quality of solutions with different communication cost measures

As a first experiment, we compared the solutions obtained using the mathematical models given in Section 3 in terms of different communication cost functions. Our analysis is similar to that of Wang et al. (2015), but we use optimal solutions rather than solutions found by heuristics. For each instance, we have found the optimal teams in terms of two communication cost: sum of distances (CC-SD) and diameter (CC-D). With these computational experiments our aim is to see the quality of these optimal teams in terms of other communication cost definitions. According to Wang et al.'s (2015) results, among the ones tested, MinSD and MinLD algorithms (Kargar and An, 2011) have the best performances in terms of CC-SD and RarestFirst algorithm (Lappas et al., 2009) in terms of CC-D. We have also tested these algorithms in our preliminary analysis to assess the quality of their solutions.

Notice that given an CC-D optimal team, we can obtain a larger but still optimal team by adding people whose distance to every team member is less than or equal to the optimal diameter value. Looking closely to CC-D-optimal solutions of TFP-D, we observed that the solutions reported by the solver are usually unnecessarily large teams. Although the additional people do not increase the diameter of the team, they may increase the other type of communication costs. If a CC-D-optimal team becomes infeasible when any of the team member

[6] http://home.cse.ust.hk/faculty/wilfred/wangxinyu/

[7] https://github.com/nihalberktas/TFP-data

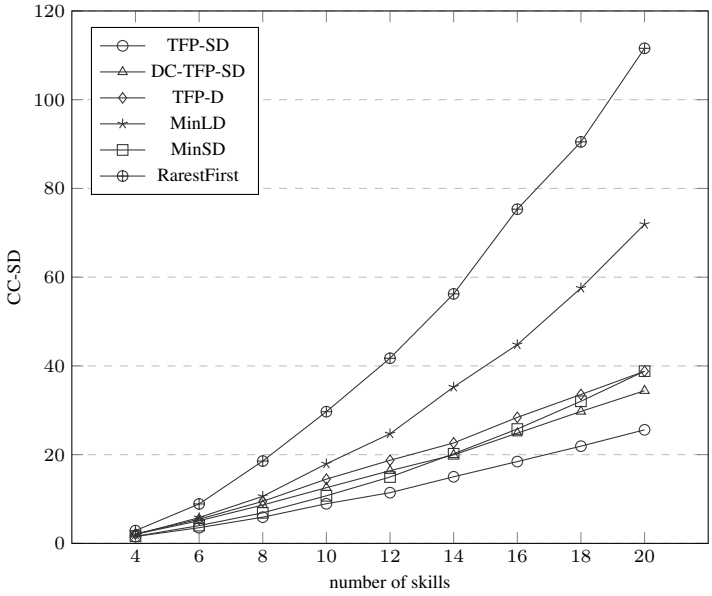



Figure 3: Average CC-SD values

is removed, we call this team minimally optimal. In our analysis, to get minimally optimal solutions we add the term $\sum_{i \in N} y_i$ to the objective of TFP-D with a very small coefficient. Moreover, to investigate whether there are CC-D-optimal teams with lower CC-SD values, we solved DC-TFP-SD by taking the allowed diameter value as the optimal one.

For the preliminary analysis we use the IMDB dataset. Since we have 100 instances for each $m$, we report the average results. Figure 3 depicts the average sum of distances values of teams found by different solution procedures. The optimal CC-SD values are obtained by solving TFP-SD. It is clearly seen that MinLD and RarestFirst algorithms perform very poorly in terms of CC-SD. MinSD algorithm performs much better compared to the previous two, but its performance degrades as the number of required skills increases. In particular, when the number of required skills, $m$, is 4, MinSD finds the optimal team for 90 out of 100 instances but it reports only 2 optimal solutions when $m = 20$. Overall, MinSD algorithm gives the optimal solutions in 28% of the instances. On the average, the teams obtained by the algorithm have 1.3 times of the optimal CC-SD values with a worst performance of 2.78 times the optimal.

The average CC-SD values of the teams obtained by solving TFP-D and DC-TFP-SD are also plotted in Figure 3. On the average, the CC-SD values of the teams found by TFP-D are 1.68 times the optimal. When we solve DC-TFP-SD with the optimal diameter value, the gap decreases a little and we get teams whose CC-SD values are 1.57 times the optimal. Thus minimizing CC-D results in teams with high CC-SD, in other words, focusing on the cost of the worst pair increases the average pairwise cost.

Figure 4 illustrates the average performances of the algorithms and the mathematical models in terms of CC-D. The optimal diameter values are found by solving TFP-D. According to the figure, MinLD and RarestFirst algorithms display similar performances while MinSD algorithm is slightly worse. Over all instances, the average optimality gap of RarestFirst, MinLD and MinSD algorithms are 16.45%, 18.65% and 22.95% respectively. Thus, RarestFirst, a 2-opt approximation algorithm for TFP-D, is the best one among these three. However its performance degrades as the number of required skills increases. Algorithm MinLD beats RarestFirst when $m$

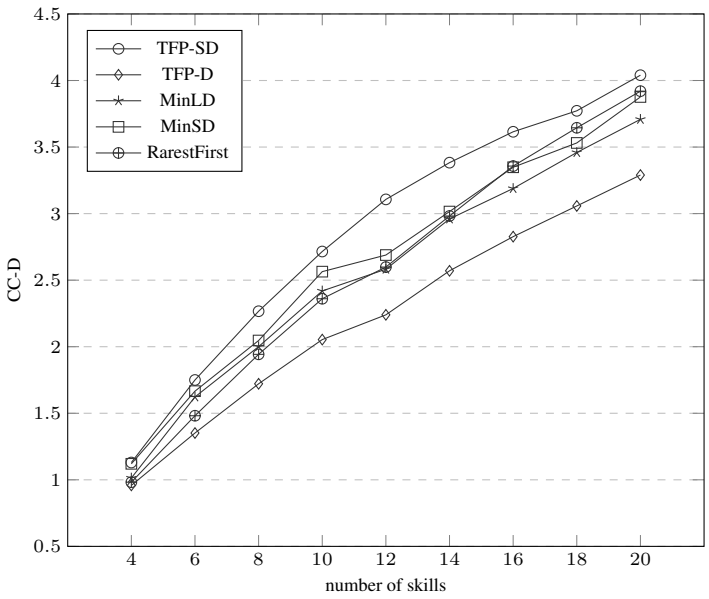



Figure 4: Average CC-D values

is higher.

Figure 4 also shows that the solution quality of TFP-SD is worse than those of the algorithms with respect to CC-D. When we examine the results in detail, we see that on the average the CC-D values of the TFP-SD solutions are around 1.38 times of the optimal. The gap is highest when $m = 12$, in that case the cost is 1.55 times of the optimal. Similar to the analyses of Wang et al. (2015), our experiment indicates that focusing on one type of communication cost leads to high values in another and consequently neither CC-SD-optimal nor CC-D-optimal teams can dominate the other. Nevertheless the results show that the CC-D values of CC-SD-optimal teams are closer to the optimal in comparison to the CC-SD values of CC-D-optimal teams. This is expected since CC-SD is a more general measure. As CC-SD includes all the shortest distances among the team members, it includes the largest one as well.

In the experiments of Wang et al. (2015), MinSD and MinLD algorithms perform best with respect to the cost of the minimum spanning tree on team's subgraph, CC-S. In our experiments MinLD algorithm performed poorly in terms of CC-S as seen in Figure 5. The solutions of MinSD algorithm and TFP-SD are the best ones while the worst performance belongs to RarestFirst algorithm. Moreover, the difference between the results of TFP-D and DC-TFP-SD in Figure 5 shows that minimizing sum of distances while forcing the diameter to be optimal yields lower CC-S values, as opposed to minimizing the diameter alone.

We also compare the solutions of the mathematical models and the algorithms with respect to the leader distance, CC-L. Kargar and An (2011) define CC-L in terms of skills, i.e., if a person is responsible for $k$ different skills then his distance to the leader is counted $k$ times. We give another and more straightforward leader distance definition, CC-L2, as the sum of distances from team members to the leader. We calculate the minimum leader distances of the teams with respect to both definitions and the average results are depicted in Figure 6. The MinLD algorithm is an exact method for CC-L and from Figure 6a we see that CC-L values of the teams given by all other methods are very far from the optimal. CC-L is defined in a way that it gives more importance to the communication of the leader to the people with more responsibilities. Since the mathematical models

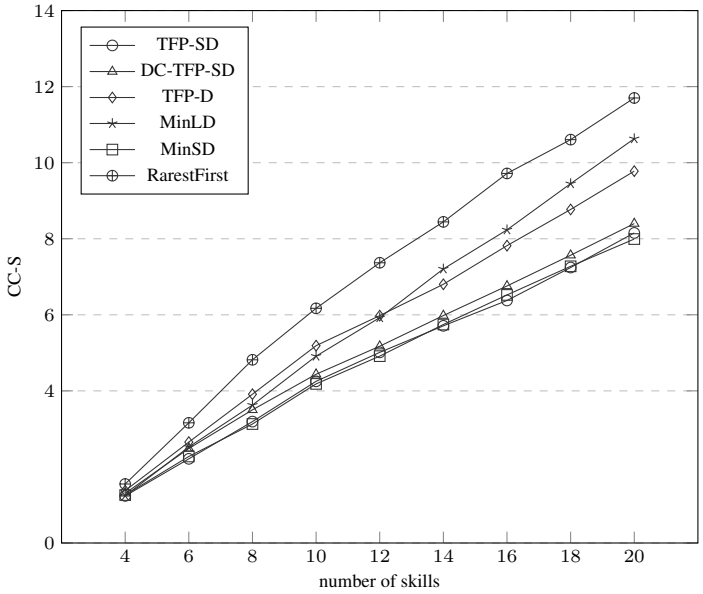
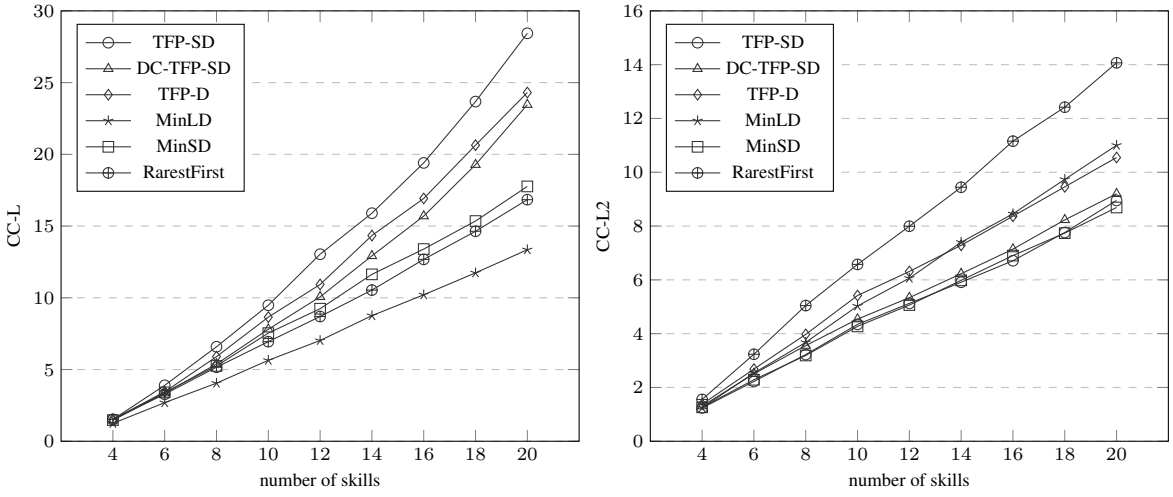



Figure 5: Average CC-S values




(a) (b)

Figure 6: Average CC-L and CC-L2 values

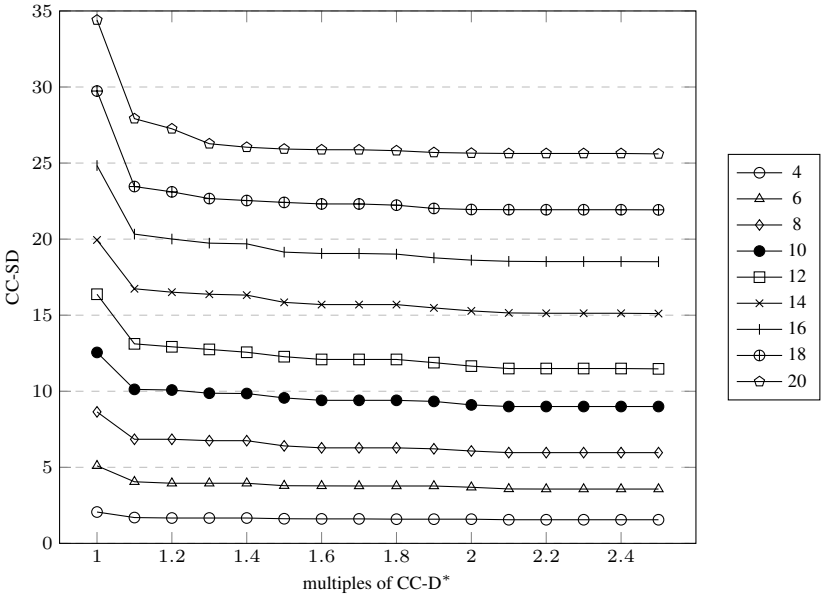


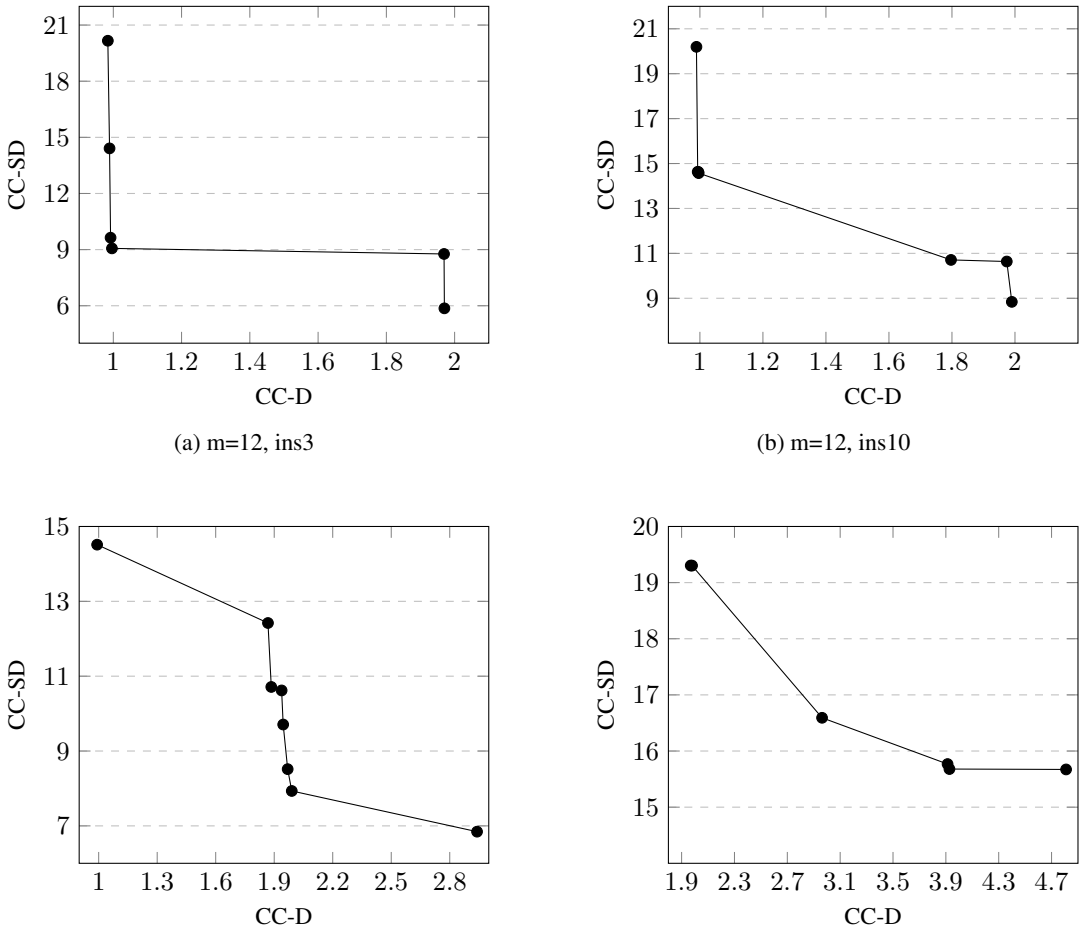

Figure 7: Average CC-SD values of teams with bounded CC-D

consider skills only in terms of coverage, they treat each pairwise communication equally, and consequently their performance is weak with respect to CC-L. As we consider the team as a part of a social network, we regard CC-L2 as a more appropriate definition for leader distance in this comparison. With respect to CC-L2, the solutions of TFP-SD and RarestFirst algorithm have the best quality.

From their comparison of TFP algorithms, Wang et al. (2015) conclude that no single algorithm outperforms in terms of all communication cost functions but algorithms based on CC-SD are able to find teams with low costs with respect to other definition as well. The analysis we have conducted with the IMDB dataset leads to similar findings. We see that optimal solution in terms of one communication cost definition may result in a value very far from the optimal with respect to some other definition. Among the ones we consider, no solution method is able to dominate the other in terms of all cost definitions. Nevertheless, CC-SD-optimal teams found by solving TFP-SD have the lowest costs with respect to CC-S and CC-L2. The problem with those teams is that they are likely to have a bottleneck pair, i.e., CC-D may be high. Solving TFP-D we found CC-D-optimal teams but they had high CC-SD, CC-S and CCL2 values. Then we see that these values are lower for the CC-D-optimal teams obtained by solving DC-TFP-SD. At this point we would like to see how the bound on the diameter affects the quality of the solutions in terms of CC-SD. Therefore we conducted further analysis with DC-TFP-SD. For each instance starting from the optimal CC-D value, we increase the bound on CC-D by increments of 10% and observe the changes in CC-SD. In the rest of this section we present the results of this analysis and pareto optimal solutions of some instances with respect to CC-SD and CC-D.

In Figure 7 for each $m$, we present the average CC-SD values of teams obtained by solving DC-TFP-SD with different bounds on the diameter. For each instance first we use its optimal CC-D value as the allowed diameter in defining set $C$. Then we increase the bound by 10% and solve the problem again. In 95% of the instances the optimal CC-SD is reached when the bound on the diameter is 2.5 times of the optimal therefore we present the results up to this bound. As we expect, the average CC-SD value decreases as we increase the bound on the diameter. We observe the greatest drop in average CC-SD values when the bound on the diameter

is raised from its optimal value by 10%. The decrease is most remarkable when the number of required skills, $m$, is high.




(a) m=12, ins3

(b) m=12, ins10

(c) m=12, ins56

(d) m=12, ins99

Figure 8: Pareto frontier of some IMDB instances

When we examine the solutions in detail we see that although there is an explicit pattern in the average results, instances can actually display very different behaviors. To illustrate those distinct behaviors we found the pareto optimal solutions of some instances with objectives CC-SD and CC-D. Figure 8 shows the pareto frontier of four instances with $m = 12$. For the first instance, illustrated in Figure 8a, the optimal diameter value is less than 1 and the team with the optimal diameter results in a CC-SD value greater than 3 times of the optimal. As plotted in the figure, very little increase in the diameter allows a great amount of decrease in CC-SD for this instance. There are three pareto optimal solutions in which the diameter is still below 1 and CC-SD is much lower and we can actually form a team with CC-SD equal to 1.55 times its optimal value while CC-D is still less than 1. Moreover, relaxing the bound on the diameter does not give us a team with lower CC-SD until the diameter is twice of its optimal value. In other words, there are no pareto optimal points until CC-D is around 2. We observe a similar pattern in the instance illustrated in Figure 8b.

In Figure 8c we are able to form a team with diameter less than 1 and again this team has very high CC-SD.

For this instance CC-SD does not decrease until the diameter is about twice of its optimal value so the pattern is different than what we observe in the average results. There are many pareto efficient solutions when the diameter is around 2 but none when it is between 2 and 3. We reach the CC-SD-optimal point when the CC-D is around 3. Finally, in the instance depicted in Figure 8d we again see a different pattern than the average. The pareto optimal points are more spread for this instance, therefore we see a gradual decrease in CC-SD .

This analysis shows that when we minimize the CC-D we lose the focus on general communication by taking only the distance of the worst pair into account and consequently the resulting team usually has high CC-SD. On the other hand, despite the generality of CC-SD as a communication cost measure, minimizing only CC-SD usually results in a team with high diameter cost. This is an indication of an bottleneck pair and it is undesirable especially when one-to-one communication is significant for the given job. For such jobs it would be reasonable to put a limit on the diameter of the team. These results suggest that the diameter constrained version of the problem, DC-TFP-SD, is the most preferable since it provides the decision maker the flexibility in evaluating the cost of the bottleneck pair while minimizing the average communication cost of the team.

In the sequel, we present all the computational experiments with the mathematical models and the branch and bound algorithms suggested for different versions of TFP. We show the efficiency of our branch and bound algorithm especially in larger networks.

## 5.3 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. As explained earlier we have generated instances with different number of required skills, i.e., $m$ takes values from the set $\{4, 6, 8, 10, 12, 14, 16, 18, 20\}$. For each $m$, 100 instances are randomly generated in a way that the set of required skills is different in each instance. 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 posses 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 even more by eliminating a person if he cannot cover all the skills together with the people who are at most allowed diameter away from him. We do this elimination iteratively until there is no one to remove from the network. After this preprocess, the network only involves people who are capable of forming a feasible team respecting the bound on the diameter and the size of new network is denoted by $fno$.

In Section 3, we have given mathematical models for different versions of the problem. We have provided QP and MIP formulations for TFP-SD, the problem with sum of distances objective; MIP formulation for TFP-D, the one with the diameter objective and another QP and MIP formulation for DC-TFP-SD, the diameter constrained problem with sum of distances objective. In Section 4, we have described the branch and bound algorithms developed for TFP-SD and DC-TFP-SD.

In addition to the QP and MIP formulations 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 (5), (6) and (7) grow quadratically in the size of the problem. Since the objective coefficients are nonnegative in our instances, it is sufficient to use only constraints (5) 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 (6) and (7)) and treat constraints (5) as lazy constraints using the callback feature of Cplex, the

Table 2: Performances for TFP-SD on IMDB instances

| | | QP | MIP | MIP-RLT-lazy | B&B | | | |
|---|---|---|---|---|---|---|---|---|
| m | qno | time | time | time | time | node | lb gap (%) | ub gap (%) |
| 4 | 422.51 | 6.66 | 7.14 | 0.81 | 1.13 | 2.08 | 2.12 | 0.00 |
| 6 | 541.81 | 22.63 | 23.21 | 1.74 | 2.66 | 14.11 | 4.12 | 0.05 |
| 8 | 653.41 | 28.50 | 29.54 | 3.16 | 4.19 | 24.6 | 5.95 | 0.06 |
| 10 | 731.82 | 30.41 | 31.12 | 5.63 | 5.92 | 41.97 | 10.27 | 0.30 |
| 12 | 791.51 | 32.60 | 33.47 | 7.59 | 7.28 | 52.36 | 12.31 | 0.22 |
| 14 | 838.48 | 43.13 | 44.70 | 10.62 | 9.83 | 111.34 | 13.31 | 0.50 |
| 16 | 879.02 | 51.81 | 53.04 | 15.57 | 12.27 | 157.72 | 13.58 | 0.18 |
| 18 | 917.68 | 83.76 | 81.04 | 18.77 | 14.31 | 164.98 | 15.13 | 0.77 |
| 20 | 947.69 | 77.98 | 78.54 | 24.93 | 13.93 | 167.69 | 16.24 | 0.70 |

solver needs to add too many lazy constraints and it takes more time than giving the model to the solver directly. However when we add the inequalities obtained by the partial implementation of RLT, only a small number of cuts are generated and the solution times are improved. We report the average solution times of all solution procedures for TFP-SD using IMDB instances in Table 2. The averages are taken over 100 instances for each $m$ and reported in seconds.

The performances of the QP and the MIP formulation for TFP-SD turn out to be very similar for 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. Among all instances, the longest solution times of the MIP and the QP are 1313 and 1322 seconds, respectively and this instance is solved in 19 seconds by the B&B algorithm. The longest solution time by the algorithm is 48.19 seconds for IMDB instances so it solves all instances within a minute. Moreover, the average solution times of branch and cut algorithm which we use inequalities obtained by RLT and lazy constraint callback are presented under the column *MIP-RLT-lazy*. With this method we are able to solve 98.6% of the instance within a minute while this percentage is 78% for the QP and the MIP. When number of required skills, $m$, is low, the method is as efficient as the B&B algorithm but as $m$ exceed 8, our B&B outperforms this method as well.

In Table 2 we also present the details of the B&B algorithm. The column named $nodes$ indicate the average number of nodes evaluated. To show the quality of the first lower bound and first upper bound, we report the average percentage gap between these bounds and the optimal value in the last two columns. For example 0.00 in the first row of last column indicates that for all instances with $m = 4$ the first incumbent found by the B&B algorithm is optimal. Although it degrades a bit as the problem gets larger, the initial upper bound is at most 1% away from the optimal in 93.55% of the instances.

In Table 3, we present the average solution times of the solution procedures for TFP-SD using DBLP instances. As DBLP is a larger network, we could not obtain a solution from the mathematical models for most of the instances. Therefore we only include the first 10 instances with $m = 4, 6$ 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 $opt$ indicates the number of instances that can be solved out of 10. Hence, we see that the MIP and the QP can only solve 4 instances with $m = 6$ and 2 instances with $m = 8$ whereas strengthening the model with RLT constraints and using lazy callback enables us to solve more within

Table 3: Performances for TFP-SD on DBLP instances

| | | QP | | MIP | | MIP-RLT-lazy | | B&B | |
|---|---|---|---|---|---|---|---|---|---|
| m | qno | opt | time | opt | time | opt | time | opt | time |
| 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 |

less time. But eventually this method also fails with memory problems as the size of instances increases. Furthermore the performances of the MIP and the QP which were very similar on IMDB instances start to differ as the problem size gets larger. The instances solved without memory problems are the same for both formulations and for those, the solution times of the QP are lower than those of the 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 DBLP instances

| m | qno | opt | time | nodes | lb gap | ub gap |
|---|---|---|---|---|---|---|
| 4 | 1540.22 | 100 | 8.59 | 20.08 | 4.97 | 0.05 |
| 6 | 2255.90 | 100 | 20.68 | 30.54 | 6.47 | 0.27 |
| 8 | 2963.26 | 100 | 37.69 | 110.52 | 8.16 | 0.48 |
| 10 | 3604.40 | 100 | 59.86 | 239.10 | 7.99 | 0.69 |
| 12 | 4189.49 | 99 | 89.41 | 480.52 | 8.56 | 0.89 |
| 14 | 4789.13 | 99 | 249.25 | 3374.63 | 8.79 | 0.89 |
| 16 | 5298.52 | 99 | 274.76 | 3099.22 | 8.62 | 0.66 |
| 18 | 5857.60 | 97 | 482.83 | 5637.57 | 9.25 | 0.76 |
| 20 | 6412.48 | 91 | 680.89 | 6439.47 | 9.32 | 0.82 |

We present detailed results of the B&B algorithm with DBLP instances in Table 4. We also consider larger $m$ values here. The column titled $opt$ 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. 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. Furthermore, the average number of nodes evaluated in the algorithm can also be seen in Table 4 under the column named $nodes$. We also give the average gap of the first lower bound and first upper bound found. Similar to the results with IMDB instances the upper bound at the root node is very close to the optimal solution. Approximately at 69% of the instances, the first upper bound is at most 1% away from the optimal value.

In the rest of this section we present the computational experiments done for the diameter constrained version of the problem, DC-TFP-SD, using the QP and MIP formulations and the B&B algorithm. For IMDB instances, we first found the optimal diameter costs using the MIP formulation of TFP-D which solves each instance in 10 seconds on the average. In Table 5, we present the solution times of DC-TFP-SD where we use the optimal diameter values as the bound. Due to the preprocess the network is reduced significantly and

Table 5: Performances for DC-TFP-SD on IMDB instances using optimal CC-D

| | | | QP | MIP | B&B | | | |
|---|---|---|---|---|---|---|---|---|
| m | qno | fno | time | time | time | node | lb gap | up 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 |

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 it 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 or 2 seconds for all methods.

Table 6: Average solution times for DC-TFP-SD on IMDB instances

| | D=2 | | | | | D=3 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| m | # fea | fno | MIP | QP | B&B | # fea | 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 IMDB instances trying different bounds on the diameter. We took the bound on the diameter as 1, 2, 3 and 4 and in Table 6 we present the results with bounds 2 and 3 as indicated by $D$. Under the column named $\#fea$, the number feasible instances is given over 100 instances for each $m$. In IMDB instances, the optimal diameter is usually less than 2 and therefore $fno$'s in this table are greater than the ones in Table 5. Thus, as we increase the bound on the diameter, we are solving the problem on a larger 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 the MIP and the QP become 15 seconds on the average while it is still a couple of seconds for the branch and bound.

For DBLP instances, it was not possible to find the optimal diameter values using the MIP formulation of TFP-D due to memory errors, therefore 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

Table 7: Average solution times for DC-TFP-SD on DBLP instances

| | D=2 | | | | | | | D=3 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MIP | | QP | | B&B | | | MIP | | QP | | B&B | |
| m | #fea | opt | time | opt | time | opt | time | #fea | opt | time | opt | time | opt | time |
| 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 |

for $D = 2$ and $D = 3$ in Table 7. $\#fea$ shows the number of feasible instances out of 10 and *opt* shows the number of instances can be solved without memory error for each solution method. For these instances average solution time of the MIP is few minutes and usually less than that of the QP. Nevertheless the QP is able to solve all these instances while we encounter memory errors with the MIP when $D = 3$ and $m$ exceeds 6. The B&B algorithm, on the other hand, is able to solve each of these instances under 30 seconds.

Table 8: Average solution times of the B&B algorithm for DC-TFP-SD on DBLP instances

| | D=1 | | | D=2 | | | D=3 | | | D=4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| m | #fea | opt | time | fea | opt | time | #fea | opt | time | #fea | opt | time |
| 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 | 0 | 0 | 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 | 68 | 214.34 | 94 | 94 | 131.83 |
| 16 | 0 | 0 | 0 | 2 | 2 | 5.51 | 56 | 56 | 519.96 | 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 8, we present the results for all DBLP instances using the B&B algorithm for DC-TFP-SD with $D = 1, 2, 3, 4$. The algorithm is able to solve 99% of the feasible instances within two hours of time limit.

# 6 Conclusion

The problem of forming optimal teams has attracted attention from different fields. While the OR community has focused on skills and capabilities of individuals, often ignoring the quality of communication, in knowledge, discovery and data science field, the problem has been studied on social networks with different objective functions to quantify the communication effort of the team and to minimize it. However, in this line of work, exact algorithms are not considered; instead, approximation algorithms and greedy heuristics are used.

This study was motivated by the new outsourcing concept Team as a Service and also by the studies underlying the significance of communication in teamwork, especially for virtual teams, which are becoming more popular each day. Furthermore, we were encouraged by the advances in social network analysis, the availability of such networks and their functionality in team formation. The lack of an exact method to solve TFP on large

social networks is another reason of this work.

We studied the problem using two different measures of communication: sum of distances and diameter. After our preliminary analysis in which we compared the quality of the optimal solutions obtained with these two objectives we concluded that although sum of distances is a general and robust measure it may lead to solutions with high diameters. This means that minimizing sum of distances may still yield a team with a bottleneck pair who can degrade the performance of the team substantially. Therefore, we proposed a diameter constrained TFP with sum of distance objective which we formulated as a quadratic set covering problem with packing constraints. As explained in our extensive computational study, small and medium size instances can be solved with a general purpose solver but the solver encounters memory problems for larger instances. To solve these, we proposed a branch and bound algorithm which can be used for any 0-1 program with a quadratic objective. Our computation experiments showed that the algorithm is able to solve instances which are intractable for the solver.

The present work can be extended in several ways. First, the communication cost may be quantified in more detail by paying attention to the communication needs for specific tasks of a project. This requires not only choosing team members but also assigning them 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.

# 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.

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. Constrained 0–1 quadratic programming: Basic approaches and extensions. *European Journal of Operational Research*, 187(3):1494–1503, 2008.

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.

S. G. Cohen and D. E. Bailey. What makes teams work: Group effectiveness research from the shop floor to the executive suite. *Journal of Management*, 23(3):239–290, 1997.

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.

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.

FlexJobs. 2017 State of telecommuting in the u.s. employee workforce. Retrieved December 15, 2017, `https://www.flexjobs.com/2017-State-of-Telecommuting-US/`, 2017.

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.

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.

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.

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.

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.

G. L. Stewart. A meta-analytic review of relationships between team design features and team performance. *Journal of management*, 32(1):29–55, 2006.

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.

X. Wang, Z. Zhao, and W. Ng. Ustf: A unified system of team formation. *IEEE Transactions on Big Data*, 2 (1):70–84, 2016.

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.