DETERMINISTIC AND STOCHASTIC TEAM FORMATION PROBLEMS

A DISSERTATION SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
INDUSTRIAL ENGINEERING

By
Nihal Berktaş
January 2021

Deterministic and Stochastic Team Formation Problems
By Nihal Berktaş
January 2021

We certify that we have read this dissertation and that in our opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Oya Karaşan(Advisor)

Hande Yaman Paternotte(Co-Advisor)

Özlem Çavuş İyigün

Sakine Batun

Mehmet Selim Aktürk

Ignacio E. Grossmann

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

DETERMINISTIC AND STOCHASTIC TEAM FORMATION PROBLEMS

Nihal Berktaş
Ph.D. in Industrial Engineering
Advisor: Oya Karaşan
Co-Advisor: Hande Yaman Paternotte
January 2021

In various organizations, physical or virtual teams are formed to perform jobs that require different skills. The success of a team depends on the technical capabilities of the team members as well as the quality of communication among the team members. We study different variants of the team formation problem where the goal is to build the best team with respect to given criteria. First, we study a deterministic team formation problem which aims to construct a capable team that can communicate and collaborate effectively. To measure the quality of communication, we assume the candidates constitute a social network and we define a cost of communication using the proximity of people in the social network. We minimize the sum of all pairwise communication costs, and we impose an upper bound on the largest communication cost. This problem is formulated as a constrained quadratic set covering problem. Our experiments show that a general-purpose solver is capable of solving small and medium-sized 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-sized instances, which are intractable for the solver.

Second, we consider a two-stage stochastic team formation problem where the objective is to minimize the expected communication cost of the team. We assume that for a subset of pairs the communication costs are uncertain but they have a known discrete distribution. The first stage is a trial stage where the decision-maker chooses a limited number of pairs from this subset. The actual cost values of the chosen pairs are realized before the second stage. Hence, the uncertainty in this problem is decision-dependent, also called endogenous, because the first stage decisions determine for which parameters the uncertainty will resolve. For this problem, we give two formulations, the first one contains

a set of non-anticipativity constraints similar to the models in the related literature. In the second, we are able to eliminate these constraints by changing the objective function into a quadratic one, which is linearized by a set of extra binary variables. We show that the size of instances we can solve with these formulations using a commercial solver is limited. Therefore, we develop a Benders' decomposition-based branch-and-cut algorithm that exploits decision-dependent nature to partition scenarios and use tight linear relaxations to obtain strong cuts. We show the efficiency of the algorithm presenting results of experiments conducted with randomly generated instances.

Finally, we study a multi-stage team formation problem where the objective is to minimize the monetary cost including hiring and outsourcing costs. In this problem, stages correspond to projects which are carried out consecutively. Each project consists of several tasks each of which requires a human resource. We assume that due to incomplete information there is uncertainty in people's performances and consequently the time a person needs to complete a task is random for some person-task pairs. When a person is assigned to a task, we learn how long it takes for this person to finish the task. Hence, the uncertainty is again decision-dependent. If the duration of a task exceeds the allowable time for a project then the manager must hire an external resource to speed up the process. We present an integer programming formulation to this problem and explain that the size of the formulation strongly depends on the number of random parameters and scenarios. While this deterministic equivalent formulation can be solved with a commercial solver for small-sized instances, it easily becomes intractable when the number of random parameters increases by one. For such cases where exact methods are not promising, we investigate heuristic methods to obtain tight bounds and near-optimal solutions. In the related literature, different Lagrangian decomposition methods are developed for such stochastic problems. In this study, we show that the convergence of existing methods is very slow, and we propose an alternative method where a relaxation of the formulation is solved by a decomposition-based branch-and-bound algorithm.

Keywords: team formation problem, quadratic set covering, branch-and-bound, reformulation, decision-dependent uncertainty, decomposition.

ÖZET

DETERMİNİSTİK VE RASSAL EKİP KURMA PROBLEMLERİ

Nihal Berktaş
Endüstri Mühendisliği, Doktora
Tez Danışmanı: Oya Karaşan
İkinci Tez Danışmanı: Hande Yaman Paternotte
Ocak 2021

Günümüz ürünlerinin ve servislerinin karmaşıklığı çok farklı alanlarda bilgi, beceri ve deneyim gerektirmektedir. Bu nedenle şirketler, üniversiteler, hastaneler, belediyeler gibi çok çeşitli kurumlarda ekipler halinde çalışılır. Ekip tarafından yapılan işin kalitesi üyelerin teknik bilgi ve becerilerine bağlı olduğu kadar aralarındaki iletişim kalitesine de bağlıdır. Bu tezde amacın en iyi ekibi oluşturmak olduğu çeşitli ekip kurma problemleri inceliyoruz. İlk olarak, etkili bir şekilde iletişim kurabilen ve işbirliği yapabilen gerekli yeteneklere sahip bir ekip oluşturmayı amaçlayan bir ekip oluşturma problemi üzerinde çalışıyoruz. Kişiler arası iletişim kalitesini ölçmek için, kişilerin bir sosyal ağın parçası olduğu varsayıyor, bu ağdaki yakınlıklarını kullanarak bir iletişim maliyeti tanımlıyoruz. Problemimizde iletişim maliyetlerinin toplamını en aza indirgerken ve en büyük iletişim maliyetine de bir üst sınır koyuyoruz. Bu problemi, kısıtlı karesel bir küme kapsama problemi olarak formüle ediyoruz. Sayısal analizlerimiz, genel amaçlı bir tamsayılı programlama çözücüsünün küçük ve orta ölçekli örnekleri çözebildiğini gösteriyor. Daha büyük boyutları çözmek için bir dal-sınır yöntemi geliştirildi. Bu yöntemde önce problem yeniden formüle edildi, ardından bir dizi doğrusal küme kapsama problemine ayrışacak şekilde gevşetildi. Dallanma yoluyla gevşetilmiş kısıtlamalar dayatıldı. Analizlerimiz, dal-sınır yönteminin çözücü için zor olan büyük boyutlu örnekleri çözebildiğini gösteriyor.

İkinci olarak, amacın takımın beklenen iletişim maliyetini en aza indirmek olduğu iki aşamalı bir rassal takım oluşturma problemini ele alıyoruz. Bazı bireyler arasında iletişim maliyetlerinin belirsiz olduğunu, ancak bu maliyetlerin bilinen bir ayrık dağılıma sahip olduklarını varsayıyoruz. Problemin ilk aşaması, karar vericinin iletişim maliyeti rassal olan çiftler arasından sınırlı sayıda seçtiği bir deneme aşamasıdır. Seçilen çiftlerin gerçek iletişim maliyet değerleri ikinci aşamadan önce öğrenilir. İlk aşama kararlar değişkenleri, belirsizliğin hangi

parametreler için ortadan kalkacağını belirlediği için bu problemdeki belirsizlik karara bağlıdır. Bu problem için iki formülasyon veriyoruz; ilki ilgili literatürdeki modeller ile benzer bir dizi beklentisizlik kısıtları içermektedir. İkincisinde, tanımladığımız karesel amaç fonksiyonu bu beklentisizlik kısıtlarına ihtiyacı ortadan kaldırıyor. Ekstra ikili karar değişkeni tanımlayarak bu karesel fonksiyonu doğrusallaştırıyoruz. Çözücü kullanarak bu formülasyonlarla çözebileceğimiz örneklerin boyutunun sınırlı olduğunu gösteriyoruz. Bu nedenle daha büyük örnekleri çözebilmek için, Benders ayrıştırma yöntemi tabanlı bir dal-kesi algoritması geliştirildi. Algoritmada güçlü kesiler elde etmek için ikinci aşama probleminin güçlendirilmiş bir doğrusal gevşetmesi kullanıldı. Ayrıca karara bağlı yapıdan yararlanılarak algoritmanın her yinelemesinde daha küçük bir senaryo seti yaratılarak çözüm zamanı azaltıldı. Rastgele oluşturulmuş örneklerle yapılan analizlerin sonuçları ile algoritmanın etkinliğini gösterildi.

Son olarak bu tezde, amacın işe alım ve dış kaynak masrafları ile personel ücretlerinin en aza indirmek olduğu çok aşamalı bir ekip oluşturma problemini inceliyoruz. Bu problemde aşamalar, ardışık olarak yürütülen projelere karşılık gelir. Her proje, her biri bir insan kaynağı gerektiren birkaç görevden oluşur. Eksik bilgi nedeniyle insanların performanslarında belirsizlik olduğunu ve dolayısıyla bir kişinin bir görevi tamamlaması için ihtiyaç duyduğu sürenin bazı kişi-görev çiftleri için rassal olduğunu varsayıyoruz. Bir kişi bir göreve atandığında, bu kişinin görevi bitirmesinin ne kadar sürdüğünü öğrendiğimizi kabul ediyoruz. Dolayısıyla, belirsizlik burada yine karara bağlıdır. Bir görevin süresi bir proje için izin verilen süreyi aşarsa, yöneticinin süreci hızlandırmak için harici bir kaynak kiralaması gerekir. Bu problem için bir tamsayılı programlama formülasyonu sunuyoruz ve formülasyonun boyutunun büyük ölçüde rassal parametrelerin ve senaryoların sayısına bağlı olduğunu açıklıyoruz. Bu deterministik eşdeğer formülasyon, küçük örnekler için ticari bir çözücü ile çözülebilirken, rassal parametrelerin sayısındaki bir birim artışla çözülemez hale gelmektedir. Kesin yöntemlerin umut verici olmadığı bu tür durumlarda, sıkı sınırlar ve iyi çözümler elde etmek için sezgisel yöntemler ararız. İlgili literatürde, bu tür rassal problemler için farklı Lagrangian ayrıştırma yöntemleri geliştirilmiştir. Bu çalışmada, mevcut yöntemlerin yakınsamasının çok yavaş olduğunu gösteriyoruz ve formülasyonun gevşetmesinin ayrıştırma tabanlı bir dal-sınır algoritması ile çözüldüğü alternatif bir yöntem öneriyoruz.

Anahtar sözcükler: ekip kurma problemi, karesel küme kapsama, dal-sınır, karara bağlı belirsizlik, ayrıştırma.

Acknowledgement

First and foremost, I would like to express my gratitude to my advisors Prof. Hande Yaman and Prof. Oya Karaşan for their guidance and advice during my Ph.D. studies. Hande Yaman has been my academic role model since I was a sophomore in Bilkent University and I have learned so much from her as her student and advisee. I am very grateful to Oya Karaşan who always supported me during my graduate studies. I consider myself lucky to have such mentors.

I would like to thank the members of my thesis committee Assist. Prof. Sakine Batun and Assist. Prof. Özlem Çavuş İyigün for their valuable comments during the progress of this dissertation in the last four years. I am grateful to Prof. Selim Aktürk for accepting to be in my dissertation examination committee and his insightful comments. I would like to thank Prof. Nilay Noyan for her guidance in Chapter 4 of this thesis.

I am indebted to Prof. Ignacio Grossmann who gave me the opportunity to visit Carnegie Mellon and devoted his valuable time as if I am one of his Ph.D students. He is the kindest person I have met, always ready to help me with his wisdom and academic knowledge. I am extremely grateful to him for his guidance in Chapter 5 of this thesis and also for accepting to be in the examination committee.

I gratefully acknowledge the financial support provided by The Scientific and Technological Research Council of Turkey (TÜBİTAK) with grant number BİDEB-2214A for funding this research.

I would like to thank my "old gang" Burcu Tekin, Merve Meraklı, Nil Karacaoğlu, and Huseyin Gürkan who supported me throughout this journey although we are usually miles away from each other. I am grateful to my friends Ece Demirci, Gizem Özbaygın, Esra Koca, Burak Pac, Ramez Kian, Milad Maleki, and Parinaz Toufani for making the hours in the office enjoyable. I would like to thank Kübra Şahin and Beyza Çelik for the fun they brought to my life. Many thanks to my dear friend Yeliz Dingler who helped me to keep my body and soul healthy during the hard times.

Halenur Şahin, Irfan Mahmutoğulları, Halil İbrahim Bayrak, Cemal İlhan, Cansu Gülcan, Haşim Özlü, and Başak Yazar deserve special thanks for their

friendship and support. Most of the laughs I had in the last few years have been with them.

I have had the chance to meet many wonderful people while at Carnegie Mellon so I would like to thank David Bernal, Can Li, Özgün Elçi, Paulina Ortiz, Akang Wang and Zedong Peng who make my visit enjoyable despite the pandemic.

I am grateful to my parents Hatice and İzzet for believing in me and supporting me even when I doubt myself. Many thanks to my brother İhsan, my sisters Seda and Aylar, and of course my dear niece Ela for their love and encouragement.

Last but not least, I would like to thank my husband who has been my colleague, reviewer, editor, therapist and motivator besides being my best friend. Thank you for your love, patience and everlasting support.

Contents

List of Figures

List of Tables

Chapter 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, non-governmental organizations, 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 such as its diversity, people's personality, and familiarity.

The teamwork can be done physically together as in the cases of surgical teams and construction teams, or the team members can work virtually, which is mostly seen in the software development business. In addition to the classical 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 [2]. Furthermore, both companies and individuals use platforms such as *freelancer*[1], *upwork*[1] and also *github*[1] and

stackoverflow[1] to find appropriate team members for their projects. As the establishment of these online platforms indicates, the way that people are recruited and teams are built change over time and so as the means of communication and the way they collaborate. A significant amount of people are working remotely nowadays. Moreover, teams are built and dispersed more often compared to the past because of the increase in project-based work and increase in pace of work in general.

There are numerous factors affecting 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 surveys [3] and [4]. All these factors determine the effectiveness of the collaboration and consequently the quality of the teamwork. Hoegl and Geumenden [5] 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 [6] is among others that emphasize the significance of the communication in teamwork, especially in virtual teams.

Ineffective communication is one of the major factors behind unsuccessful projects and teamwork in general. Approximately half of the errors and failures are directly related to communication in medical decisions as shown in Joint Commission's 2014 report [7] and in business projects, as revealed in Project Management Institute's 2013 report [8]. While with empirical studies the scientific community tries to detect the key features of successful teams and understand the importance of communication, business firms have started to devote resources to improve teams' performance through online applications that ease project team communication, through new team and working concepts such as *agile* and *scrum teams* [9], and through team building games that strengthen bonds such as *The Go Game*[2]. Such investments and trainings are strategical long term plans that aim to sustain effective communication throughout the organization. Sometimes organizations may require more direct and fast methods to build teams with strong communication. Examples could be a surgical team for a complex surgery

[1] freelancer.com, upwork.com, github.com, stackoverflow.com/talent

[2] thegogame.com/team-building-games

or a project team in a consultancy firm that will work for an important client.

Motivated by the importance of communication in the quality of teamwork, in Chapter 3 and Chapter 4, we study team formation problems focusing on communication. Our work in Chapter 3 is also motivated by the abundance of online platforms and it assumes that team member candidates constitute a social network. The decision-maker here could be an individual who requires a team for a project or it could be a company that uses a Team as a Service model and wants to create a team for a client. In this work, we adopt the problem definition of Lappas et al. [10]. The project consists of several tasks so it requires team members with the necessary skills to perform these tasks. The skills of the candidates are assumed to be known and represented by a binary skill matrix built by considering minimum expertise levels. The aim is to select team members and form a capable team that can communicate effectively.

To build a team that is good at communicating, we require a measure for the communication. In the literature, different methods are utilized to quantify communication using people's personalities, peer evaluations and/or work history. There are empirical studies indicating positive effects of team members' familiarity on the performance of the team. In general, familiarity is one's knowledge about the other members of the team . Huckman et al. [11] define team familiarity as the average number of times that each team member has worked with every other team member. For the teams working in a software service company, the authors show the existence of a positive and significant relation between team familiarity and operational performance. Analyzing software development teams of a telecommunications firm, Espinosa et al. [12] find that team familiarity is more beneficial when coordination is more challenging due to team size or dispersion. The study of Avgerinos and Gokpinar [13] on productivity of surgical teams also shows that the benefit of familiarity increases as the task gets more complex.

Motivated by these studies, in Chapter 3, to quantify the communication cost between two people in the social network we use a metric that is inversely proportional to their familiarity, which depends on the number of times they worked

before. Hence, pairs with higher familiarity have lower communication costs. This definition of communication cost between two people is also used by Lappas et al. [10], and by many others who study team formation in social networks. Clearly, the familiarity of team members is not the only factor that affects the team performance. For example, diversity is considered as a positive factor since it boosts creativity [14]. These concepts are not mutually exclusive and can be considered simultaneously if desired, either by taking them into consideration while assigning a value to the communication costs or by additional constraints.

After assigning a value to the communication cost between two people, we need to define the communication cost of whole team. Different cost functions are defined and optimized in the related literature. We propose to minimize the sum of all pairwise communication costs and to impose an upper bound on the highest one. We show that the problem can be formulated as a quadratic set covering problem with packing constraints. Using the existing real datasets, it is shown that small and medium-sized 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 algorithm is based on a reformulation of the problem, which we relax in a way that it decomposes into a series of linear set covering problems and can be solved efficiently. The relaxed constraints are imposed through branching.

In Chapter 4, we study a two-stage stochastic team formation problem where for some pairs, the cost of communication is not known with certainty but the possible values it can take and their respective probabilities are known. The first stage is a trial stage, which gives an opportunity to observe the communication of such pairs. A capable team with minimum expected communication cost is built in the second stage in the light of the observations. There is a limit on the number of pairs that can be observed during the first stage. This can be regarded as allocating a budget for learning and the decision-maker can decide on this limit/budget according to the available resources. This type of problem is more likely to occur in a project-based company, which creates a team for each job and has the opportunity to observe the communication among its employees by assigning small tasks, which corresponds to the trial stage in the problem.

Hence, the number of candidates in consideration in this problem is much smaller than the ones in the problem in Chapter 3, where we consider social networks of thousands of people. In Chapter 4, we have a smaller setting since we consider building a team in a department of a company, where capable people are limited to tens, not thousands. For this reason, we generate and use random instances in this problem, rather than using real social networks. Furthermore, we do not make any assumptions about how communications costs are quantified.

The uncertainty in the problem studied in Chapter 4 is decision-dependent or endogenous because we assume the resolution of uncertainty for the pairs that are selected in the first stage. For this problem, the value of the stochastic solution concept does not apply, and therefore we define a concept called *value of learning*, which is a measure of improvement we get by the information obtained in the first stage. We present two mathematical formulations for the two-stage stochastic team formation problem and show their equivalence. In the first formulation, we use the same modeling approach in the related literature and it contains a higher number of non-anticipativity constraints. The second formulation does not have these constraints but has a quadratic objective function, which is linearized by defining an extra set of binary variables. By generating instances with different sizes, we show that for small-sized instances these formulations can be solved by a commercial solver in reasonable time. To be able to solve larger sizes, we propose a Benders' decomposition-based branch-and-cut algorithm where the duality-based optimality cuts are obtained by a stronger linear relaxation of the second stage problems. This stronger relaxation does not only generate stronger cuts but also decreases the computational burden of solving the integer problems by providing integral solutions often. The algorithm is capable of solving problems with thousands of scenarios because at each iteration it works on a smaller scenario set, which we are able to create thanks to the decision-dependent structure.

In Chapter 5, we study a multi-stage team formation problem where each stage corresponds to a project. While in the first two problems the focus is on the quality of communication among the members, in the last one the concern is monetary. The aim is to minimize the expected hiring and outsourcing costs for

the whole horizon while having qualified team members to complete the required tasks. We assume randomness in the time required by a person to finish a task. Similar to the problem in the previous chapter, the uncertainty here is endogenous because we assume that the true value of a random parameter is learned once the related decision is made in the previous stages. This might be an interesting problem for an individual who manages several projects and mostly recruits people online using the platforms mentioned before. With each project, the manager evaluates the performance of the team members and decides whether to hire the same person for the following projects.

Unfortunately, it is not possible to develop an alternative formulation to this problem in the way it is done for the two-stage problem in Chapter 4. Hence the formulation of the problem consists of a large number of non-anticipativity constraints. The performance of commercial solvers is very sensitive to the number of random parameters present in an instance. We show that instances of very limited size can be solved to optimality directly with a general-purpose solver. For larger sizes, we investigate efficient methods to obtain near-optimal solutions. On randomly generated instances, we test the existing decomposition methods and show that they fail to give tight bounds in reasonable time. We also show that with a different relaxation and decomposition approach the bound can be improved but it requires more computational time. As an alternative, we propose a decomposition-based branch-and-bound algorithm, which exploits the combinatorial structure of the problem and uses scenario groups.

Chapter 2

Literature Review

In this chapter, we provide an overview of the related literature for the problems studied in the thesis. We start with a summary of team formation problems, which are studied in different fields such as knowledge discovery and data mining, concurrent engineering and project management. Then we present the literature on stochastic programming with decision-dependent uncertainty explaining existing solution methods.

2.1 Team Formation Problems

In general, the team formation problem (TFP) concerns an optimal selection of team members for a single or multiple projects with respect to a set of criteria. In operations research (OR) literature, the earliest related study belongs to Zakarian and Kusiak [15], which is on constructing multi-functional teams for product design and development. They first propose a methodology to prioritize types of team members with respect to engineering characteristics and then provide an integer programming formulation where the objective is to maximize the total priority weights of the teams. The total number of teams and the number of teams a person can join are limited in this study. Boon and Sierksma [16] give

a matching model for sports team formation problem where candidate players and positions are matched to maximize sum of player-position weights. The weights indicate the performance of the players for the positions so the aim is to form a team having maximum level of performance. Agustín-Blas et al. [17] 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. Although these studies deal with team formation problems in different areas, all three focus on the technical performance of the team, which is defined as the sum of members' performances. In these studies the communication among the team members or their personalities are not taken into consideration.

In the studies of Chen and Lin [18], Fitzpatrick and Askin [19], and Zhang and Zhang [20], in addition to the technical skills of team members, their personal characteristics are taken into consideration. Well-known personality tests such as Myers-Briggs and Kolbe Conative are used to determine personality types of candidates, which serves as a tool to measure their ability to work with each other. In their project team selection problem, Baykaşoğlu et al. [21] incorporate the concern of ability to work together by having a constraint that prevents two people, who do not want to work with each other, from being teammates. Gutiérrez et al. [22] study a multiple team formation problem and model interpersonal relations via the sociometric matrix, which consists of -1, 0 and 1's representing the negative, neutral and positive relations, respectively. They define an efficiency function for a project that inputs people's skills and relations, and the goal is to create teams with the maximum weighted sum of efficiencies. The authors present computational experiments where a constraint programming approach and two heuristic methods are compared.

To the best of our knowledge in the operations research literature, the study by Wi et al. [23] is the first one to use social networks in team formation to quantify the quality of communication among people. The authors form a network by generating fuzzy familiarity scores among candidates using collaboration data. They formulate a nonlinear program whose objective is a weighted sum of the

performance, the familiarity and the size of the team, and a genetic algorithm is proposed in the study. In the multi-objective member selection problem by Feng et al. [24], one objective is related to the individual performances of the team members while two others are related to the collaborative performances, which can be identified by cooperation, communication, knowledge sharing, mutual trust etc. Farasat and Nikolaev [25] 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 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 proposed and compared with a genetic algorithm.

Apart from the studies [23] [24] [25] mentioned above, the TFPs where a social network is considered are mainly studied in the knowledge discovery and data mining (KDD) field, initiated by the work of Lappas et al. [10] 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. 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. [10] 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 [26], Kargar et al. [27] and Bhowmik et al. [28] 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. In [26] leader distance is defined 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 and Datta [29] as the maximum edge weight in a tree that

minimizes this and that spans the team members. Dorn and Dustdar [30] and Gajewar and Sarma [31], on the other hand, use communication cost functions that are related to the density of the team's subgraph. In all of these studies in KDD field, approximation algorithms, greedy heuristics and metaheuristics are developed and tested.

The work we present in Chapter 3 is closer to the ones in KDD field in terms of the problem definition, but in terms of modeling and solution methodology it is quite different because we present an integer programming formulation for the problem and develop an exact branch-and-bound algorithm. Although team formation problems are modeled as integer programs in OR field, in those studies the models are either solved to optimality for very small examples or heuristic methods are applied. In our work, we show that our algorithm is able to solve large instances to optimality.

In Chapters 4 and 5 we study stochastic team formations problems. To the best of our knowledge there are no similar studies in the literature in terms of the problem setting. Therefore, we will mention the closest studies in the literature. In Chapter 4, we study a two-stage stochastic team formation problem where the stochasticity stems from the uncertainty in communication cost among people. There are a couple of studies that address uncertainty in the TFPs and the probabilistic aspect chosen in those studies is related to the availability or reliability of a team member. Therefore, the terms robust and recoverable are used in these studies. The aim of the study by Crawford et al. [32] is to find a minimum cost team who still covers the required skills after k agents are removed. The cost of the team is defined as a linear function of agents' individual costs. Demirović et al. [33] study a similar problem where they define a cost of recovery if the team becomes incapable after the removal of k members. Fathian et al. [34] categorize candidates as reliable and unreliable where the latter can leave the team with a known probability. The problem is to decide which main and back-up agents to assign to each position in order to maximize the quality of collaboration.

In Chapter 5, we study a multi-stage stochastic project team formation problem where each stage corresponds to an independent but similar project. Each project consists of tasks that require resources to be completed. This problem can be considered as a variant of human resource allocation or personnel selection problem where, in simple terms, the aim is to minimize cost or maximize profit by assigning resources to tasks. It has many fields of applications such as production management, project management, healthcare and education. Most of the studies, especially in project management, are conceptual and focused on determining inputs and performance measures. In modeling and solution-oriented studies, various types of assignment models are suggested for the problem [35]. Among these studies that consider multiple projects, the work by Certa et al. [36] on human resource optimization for R&D project assumes that projects are performed simultaneously. In the study of Gutjhar et al. [37] the projects are done consecutively but they are selected from a portfolio. Chen et al. [38] study a problem where an IT product development job is divided into projects consisting of tasks and both tasks and projects have precedence relations. Furthermore, the majority of the research in this field defines multiple objectives, which are mostly related to project quality, cost, time and team member relations.

The problem investigated in Chapter 5 considers randomness in task durations due to incomplete information on people's competencies. Rahmanniyay et al. [39] study a multi-objective multi-stage project team formation problem with uncertainty in time requirements. In their problem, a stage corresponds to a work unit, which is part of a single project. Once hired, people can work on several tasks in different stages but they have limited available time throughout the project. This type of uncertainty in activity duration is also considered in resource-constrained scheduling problems where a single job requires several activities with precedence relations. The works of Bruni et al. [40] and [41] are examples of such problems. To model the uncertainty and solve the problem, each of these studies follows a different method, namely stochastic programming, chance constraints and robust optimization, but in all of them the uncertainty is assumed to be exogenous, that is, the decisions do not have any effect on the values of parameters or their time of resolution. In contrast in Chapter 5,

we assume that the uncertainty in durations is due to lack of information, and consequently once a resource is allocated to a task, the true value of the task duration for that resource reveals.

2.2 Decision-dependent Uncertainty

In this section we review the studies on endogenous or decision-dependent uncertainty in stochastic programming literature. But we note that the decision dependence is studied in robust optimization as well, in the context of adjustable robust optimization where the decision is a function of observed data and also by defining decision dependent uncertainty sets.

Uncertainty is decision-dependent or endogenous when the decision can directly change the probability distribution of the random variables or it affects whether the uncertainty is relevant to the problem and the time it is resolved [42]. The study of Ahmed [43] on network design, server selection, and facility location problems and the work of Peeta et al. [44], where the failure probabilities of roads depend on the investments made, are examples of the first type endogenous uncertainty where the decision changes the structure of the distribution. The study of Goel and Grossmann [45] on gas field development planning, clinical trial planning by Colvin and Maravelias [46], project portfolio optimization by Solak et al. [47] are examples of the second type where the decision controls the resolution of the uncertainty. In these studies the uncertain parameter has a discrete distribution and a vector of realizations constitutes a scenario.

As the resolution of uncertainty directly depends on the first stage decision in the stochastic problems with endogenous uncertainty, the modeling requires more effort compared to the exogenous case. The most common type of modeling used for these problems is disjunctive programming as in [45], [48], [49], [50]. In these studies, they linearize the disjunctions. On the other hand, in [51] and [47] the problem is formulated as a linear program directly.

Goel and Grossmann [45] study gas field development planning where the size of reserve is resolved immediately if the site is chosen to be drilled. It is an example of multi-stage stochastic programming problem with endogenous uncertainty and full resolution. They devise a decomposition-based algorithm where they use a restricted model which forces the platform installation decisions to be the same under all scenarios. This model is relaxed so that it decomposes by scenario and it gives an upper bound since the problem is maximization. Lower bounds are obtained by generating feasible solutions from the restricted model. The solution of the expected value problem is used to generate different platform installation decisions. Goel et al. [52] propose a branch-and-bound algorithm based on Lagrangian relaxation for the same problem. At each node of the tree, they solve a Lagrangian dual problem which is obtained by dualizing some of the non-anticipativity constraints and completely relaxing others. The violated constraints are imposed by branching. Also at each node, feasible solutions are generated from the relaxation heuristically and lower bounds are obtained.

In the disjunctive models in these studies, the authors define a boolean variable for scenario pairs and each stage. The variable becomes true if the scenarios are not distinguishable at the stage with respect to the previous decisions. If not, then the decisions under these scenarios must be the same. So they use two sets of disjunctions: one to relate the Boolean variable to previous decisions and another to force decisions to be the same under indistinguishable scenarios when the boolean variable is true. The relaxation is obtained by relaxing the disjunctions and dualizing the non-anticipativity constraints of the first stage. A similar solution methodology is developed for the multi-stage process network optimization problem by Tarhan and Grossmann [49] where the uncertainty in the process yields resolves gradually. Tarhan et al. [53] used gradual resolution framework for the oil/gas field development problem considering nonlinear reservoir behavior as well.

Solak et al. [47] study a multi-stage project portfolio management problem where the investment requirements of the projects reveal gradually. They use a sample average approximation method where Lagrangian relaxation is used to

solve the sample problems. Boland et al. [54] study the open pit mine production scheduling problem with endogenous uncertainty. The authors present a mixed-integer linear programming model and ways to reduce number of non-anticipativity constraints. They suggest that non-anticipativity constraints can be regarded as lazy constraints when they are large in number. It means the solver starts with a model that does not have any non-anticipativity constraints. Whenever a feasible solution is found, it checks whether a non-anticipativity constraint is violated and adds it to the model if it is. Similarly, Colvin and Maravelias [51] consider endogenous uncertainty in the result of clinical trials and propose a branch-and cut-algorithm where non-anticipativity constraints are added only when violated. In all of these studies, problem specific and/or general reduction strategies are developed to decrease the number of non-anticipativity constraints in the model. Later Boland et al. [55] show how a minimum sufficient set for these constraints can be generated.

Gupta and Grossmann [56] develop a new Lagrangian decomposition algorithm to solve large-scale multistage stochastic programs with endogenous uncertainties using scenario grouping. The idea is to keep a subset of non-anticipativity constraints and dualize or relax the rest of them. Then the model decomposes into scenario groups instead of scenarios. Christian and Cremaschi [57] present two heuristic approaches for multi-stage stochastic problems with endogenous uncertainty. First one is based on a shrinking horizon approach where the problem is solved using two-stage approximations. These approximations are obtained by removing all non-anticipativity constraints except for the current time period. The second heuristic is a knapsack decomposition algorithm.

Apap and Grossmann [58] consider both endogenous and exogenous uncertainty in a multi-stage setting and present two solution methods. The first is a sequential scenario decomposition heuristic in which endogenous subproblems are solved to determine and fix binary investment decisions, and then the model is solved to find feasible solutions. The second method is based on Lagrangian decomposition.

In Chapter 4 and Chapter 5, we study stochastic problems with endogenous

uncertainty and develop different modeling and algorithmic techniques to solve these problems.

Chapter 3

A Branch-and-Bound Algorithm for Team Formation Problem

In this chapter, we study a deterministic team formation problem where we adopt the problem definition of Lappas et a. [10] and use a social network to quantify and minimize the communication cost among team members.

In Section 3.1, we formally define the team formation problem and provide quadratic and linear mathematical models. In Section 3.2, we present a branch-and-bound algorithm that uses a relaxation that can be solved by solving a series of linear set covering problems and utilizes a novel branching rule compared to existing branch-and-bound methods for quadratic 0-1 optimization problems. This section also includes an application of the algorithm on a toy problem. In Section 3.3, we first introduce our datasets and explain our instance generation method. Then we present the results of an extensive computational study. We conclude this chapter with a brief summary and final remarks in Section 3.4.

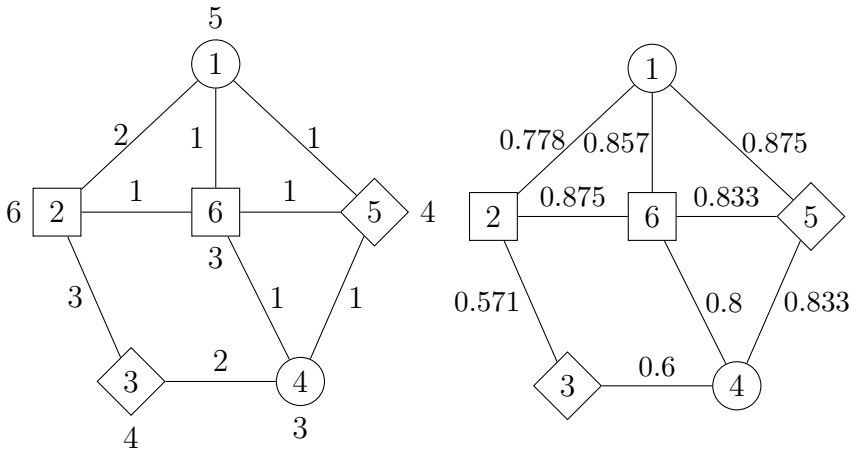
The results of this chapter are published in INFORMS Journal on Computing [59].

3.1 Problem Definition and Formulation

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 [60]. Thus for a pair of nodes, this metric assigns a distance between zero and one, such that the pairs whose common work over total work ratio is higher has a smaller distance. According to this definition 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. [10] 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 are nonnegative.

Before moving on to the problem definition, we demonstrate the cost calculation procedure on a small example. In Figure 3.1, 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



(a) Collaboration network (b) Jaccard distances

Figure 3.1: 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 3.1: Communication cost matrix for the people in the collaboration network

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 the edges of the network on the right are the Jaccard distances calculated from the collaboration data for the pairs who have common work. Calculating the shortest paths distances, we write the distance (communication cost) matrix of the whole network in Table 3.1.

In the presence of such a social network, the *team formation problem* (TFP) is defined as finding a capable team with minimum communication cost. With communication costs computed as described above by Jaccard distances, minimizing the sum of the distances amounts to maximizing the average familiarity of the team. In general familiarity is defined as the knowledge about the other members of the team. Team familiarity can be expressed in numbers as the average number of times that each team member has worked with every other member

of the team. There are empirical studies in the literature indicating the positive effects of team familiarity on the performance of teams. The results of the study by Huckman et al. [11] 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. [12] find that team familiarity is more beneficial when coordination is more challenging because of team size or dispersion.

The study by Avgerinos and Gökpınar [13] 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, that is, the pair with the lowest familiarity, significantly reduces team productivity. In terms of the communication cost measures, the least familiar pair on a team amounts to the nodes whose distance equals 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 the 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 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. The DC-TFP-SD can be modeled as follows:

$$\min \sum_{i \in N} \sum_{j \in N: i<j} p_{ij} y_i y_j \tag{3.1}$$

$$\text{s.t.} \sum_{i \in N} a_{ik} y_i \geq 1 \quad \forall k \in K, \tag{3.2}$$

$$y_i + y_j \leq 1 \qquad \forall \{i, j\} \in C, \tag{3.3}$$

$$y_i \in \{0, 1\} \qquad \forall i \in N. \tag{3.4}$$

The covering constraints (3.2) ensure that each required skill is covered; that is, there is at least one person in the team who has that skill. The family of packing

(conflict) constraints (3.3) forbids conflicting pairs in the team. The objective function is the sum of communication costs of team members. We can write the objective function in quadratic form as $y^T \mathbf{P} y$ where $\mathbf{P}$ is the communication cost matrix. $\mathbf{P}$ is a matrix with nonnegative elements and all entries in the diagonal are zero. We do not make any assumption about positive semi-definiteness of this matrix so the continuous relaxation of this quadratic problem could be convex or not.

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

$$\min \sum_{i \in N} \sum_{j \in N: i<j} p_{ij} z_{ij} \tag{3.5}$$

s.t. (3.2) - (3.4)

$$z_{ij} \geq y_i + y_j - 1 \quad \forall i, j \in N : i < j, \tag{3.6}$$

$$z_{ij} \leq y_i \quad \forall i, j \in N : i < j, \tag{3.7}$$

$$z_{ij} \leq y_j \quad \forall i, j \in N : i < j, \tag{3.8}$$

$$z_{ij} \geq 0 \quad \forall i, j \in N : i < j. \tag{3.9}$$

Constraints (3.6)-(3.9) 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 [61]. Because the objective function coefficients are nonnegative, constraints (3.7) and (3.8) can be dropped without changing the optimal value. One can use constraints $z_{ij} = 0$ for all $\{i, j\} \in C$ instead of constraints (3.3), 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 the TFP-SD on the network in Figure 3.1, with p_{ij}'s taken as in Table 3.1, is the team {2,3,4} with cost 2.342. The optimal solution of the DC-TFP-SD with a diameter limit of 0.9 is the team {4,5,6} with cost 2.466.

3.2 Branch-and-Bound Algorithms

The DC-TFP-SD is a quadratic set covering problem with side constraints (packing constraints (3.3)). One of the earliest studies on the quadratic set covering problem is by Bazaraa and Goode [62] 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 [63]; a linearization technique by Saxena and Arora [64], which does not guarantee optimality, as shown by Pandey and Punnen [65] and a study by Punnen et al. [66] on comparing different representations of the problem.

As listed in the surveys of Loiola et al. [67] on the quadratic assignment problem and Pisinger et al. [68] 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 [69], [70], and [71] on the quadratic assignment problem and [72] on the quadratic minimum spanning tree. Among the studies based on mixed-integer programming, see, for instance, a constraint-generation algorithm for the quadratic knapsack [73], a branch-and-cut algorithm for the capacitated vehicle routing problem with quadratic objective [74], and a branch-and-price algorithm for the quadratic multiple knapsack [75].

As 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 the TFP-SD, which is a quadratic set covering problem, and then extend it to the DC-TFP-SD, which is a quadratic set covering problem with side constraints.

3.2.1 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 [76] 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 one 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:

$$\sum_{i \in N \setminus \{j\}} a_{ik} z_{ij} \geq y_j \quad \forall k \in K, j \in N : a_{jk} = 0. \tag{3.10}$$

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

$$\begin{aligned}
\min\ & \frac{1}{2} \sum_{i \in N} \sum_{j \in N \setminus \{i\}} p_{ij} z_{ij} \\
& \text{s.t. (3.2), (3.4), (3.10)} \\
& z_{ij} \leq y_j \quad \forall i, j \in N : i \neq j, && (3.11) \\
& z_{ij} = z_{ji} \quad \forall i, j \in N : i < j, && (3.12) \\
& z_{ij} \geq y_i + y_j - 1 \quad \forall i, j \in N : i < j, && (3.13) \\
& z_{ij} \in \{0, 1\} \quad \forall i, j \in N : i \neq j. && (3.14)
\end{aligned}$$

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 what follows, 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. [77] and Hahn et al. [78], 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 [79], Caprara et al. [80], Pisinger et al. [68], and Fomeni et al. [81]. The main distinction between these reformulations and ours is that constraints of type 3.13 are redundant in these reformulations because of problem and cost structure, whereas in our case they are necessary.

We are interested in the relaxation of the reformulation obtained by removing constraints (3.12) and (3.13). Let $(\mathbf{y}^*, \mathbf{z}^*)$ be an optimal solution of the relaxation. Because constraints (3.12) 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 (3.13). 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 by adding a new set of constraints. Suppose now that we are at node ℓ of the branch-and-bound tree. Let C^1_ℓ be the set of pairs who are not allowed to be in the team together, and C^2_ℓ be the set of pairs who are forced to be in the team at node ℓ. Then the relaxation we solve at node ℓ, called R_ℓ, is as follows.

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

s.t. (3.2), (3.4), (3.10), (3.11), (3.14)

$$y_i + y_j \leq 1 \quad \forall \{i,j\} \in C^1_\ell, \tag{3.15}$$

$$y_i = y_j = 1 \quad \forall \{i,j\} \in C^2_\ell, \tag{3.16}$$

$$z_{in} + z_{jn} \leq y_n \quad \forall \{i,j\} \in C^1_\ell, n \in N \setminus \{i,j\}, \tag{3.17}$$

$$z_{in} = z_{jn} = y_n \quad \forall \{i,j\} \in C^2_\ell, n \in N \setminus \{i,j\}, \tag{3.18}$$

$$z_{ij} = z_{ji} = 0 \quad \forall \{i,j\} \in C^1_\ell, \tag{3.19}$$

$$z_{ij} = z_{ji} = 1 \quad \forall \{i,j\} \in C^2_\ell. \tag{3.20}$$

Constraints (3.15) and (3.19) ensure that pairs in C^1_ℓ are not in the team

together. Constraints (3.17) guarantee that a person cannot be in the team with i and j at the same time for $\{i,j\} \in C_\ell^1$. Constraints (3.16) and (3.20) ensure that i and j are both in the team for $\{i,j\} \in C_\ell^2$. Constraints (3.18) guarantee that if person n is in the team then he/she is in the team together with both i and j for $\{i,j\} \in C_\ell^2$. In short, at node ℓ constraints (3.15)-(3.20) fix the infeasibilities, which occur due to lack of constraints (3.12) and (3.13), for the pairs of nodes in sets C_ℓ^1 and C_ℓ^2.

Next we show that R_ℓ can be solved by solving $|N|+1$ linear set covering problems with side constraints. A similar result for the quadratic knapsack problem can be seen in [80].

Proposition 1 *The relaxation R_ℓ 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), which will be referred to as subproblem n:*

$$v_n = min \sum_{i \in N \setminus \{n\}} p_{in}\zeta_i^n \tag{3.21}$$

$$s.t. \sum_{i \in N \setminus \{n\}} a_{ik}\zeta_i^n \geq 1 \quad \forall k \in K : a_{nk} = 0, \tag{3.22}$$

$$\zeta_i^n + \zeta_j^n \leq 1 \quad \forall \{i,j\} \in C_\ell^1 : i,j \neq n, \tag{3.23}$$

$$\zeta_i^n = \zeta_j^n = 1 \quad \forall \{i,j\} \in C_\ell^2 : i,j \neq n, \tag{3.24}$$

$$\zeta_i^n = 0 \quad \forall \{i,n\} \in C_\ell^1, \tag{3.25}$$

$$\zeta_i^n = 1 \quad \forall \{i,n\} \in C_\ell^2, \tag{3.26}$$

$$\zeta_i^n \in \{0,1\} \quad \forall i \in N \setminus \{n\}. \tag{3.27}$$

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

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

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

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

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

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

Moreover the solution $(\mathbf{y}^*, \mathbf{z}^*)$*, where* $\mathbf{y}^*$ *is an optimal solution of the master problem and* $z^*_{ij} = y^*_j \bar{\zeta}^j_i$ *for all* $i, j \in N : i \neq j$*, is an optimal solution for* R_ℓ*.*

Proof. It is sufficient to observe that in R_ℓ, for a given vector $\mathbf{y}$, the problem of computing the best $\mathbf{z}$ 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 $z_{in} = \bar{\zeta}^n_i$ for all $i \in N \setminus \{n\}$. Then the best $\mathbf{y}$ can be computed by solving the above master problem. □

We note that we can also multiply constraints (3.2) 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.

We also would like comment on the meaning of the subproblem and master problem defined in Proposition 1. Subproblem n forms a team with respect to person n by finding a teammate for the skills n does not have so we can say that variable ζ^n_i in subproblem n indicates whether i is in n's team or not. Subproblem n builds a team around n such that the sum of communication costs between n and his/her teammates is minimum. Hence the objective value of subproblem n, v_n, is a lower bound on n's contribution to a capable team's overall communication cost. The master problem use this lower bound as the cost of each person and forms a capable team with minimum cost. And the teams in the sub and master problems respect the constraints about the pairs who are not allowed to be in the team (C^1_ℓ), and who are forced to be in the team (C^2_ℓ).

In our branch-and-bound algorithm, we propose to work with a weaker relaxation R'_ℓ which is obtained by dropping constraints (3.17) and (3.18) in R_ℓ. The relaxation R'_ℓ can be solved by solving for each $n \in N$ the *relaxed subproblem* Pr'_n, which is obtained by subproblem Pr_n by dropping constraints (3.23) and (3.24), with optimal solution $\bar{\zeta'}^n$ and optimal value v'_n, and then by solving the *relaxed master problem*, whose optimal value is ν' and in which v_j is replaced by v'_j 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 ℓ of the branch-and-bound tree where ℓ is not the root node, the relaxation R'_ℓ 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 ℓ' be the parent node of node ℓ. Suppose that the we obtained the current node by adding $\{i', j'\}$ to C^1_ℓ, i.e., $C^1_\ell = C^1_{\ell'} \cup \{i', j'\}$ and $C^2_\ell = C^2_{\ell'}$. Then we add the constraint $y_{i'} + y_{j'} \leq 1$ to the master problem, $\zeta^{j'}_{i'} = 0$ to the relaxed subproblem $Pr'_{j'}$, $\zeta^{i'}_{j'} = 0$ to the relaxed subproblem $Pr'_{i'}$, and the other subproblems remain unchanged. If the optimal solution of $Pr'_{i'}$ (respectively, $Pr'_{j'}$) at node ℓ' satisfies $\zeta^{i'}_{j'} = 0$ (respectively, $\zeta^{j'}_{i'} = 0$), then it is also optimal for subproblem $Pr'_{i'}$ (respectively, $Pr'_{j'}$) at node ℓ. 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^2_ℓ, then again we may need to solve the relaxed subproblems $Pr'_{i'}$ and $Pr'_{j'}$ with the additional constraints $\zeta^{i'}_{j'} = 1$ and $\zeta^{j'}_{i'} = 1$, respectively, and then the master with $y_{i'} = 1$ and $y_{j'} = 1$. □

As in R_ℓ, the solution $(\mathbf{y}^*, \mathbf{z}^*)$, where $\mathbf{y}^*$ is an optimal solution of the relaxed master problem and $z^*_{ij} = y^*_j \bar{\zeta'}^j_i$ 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'_ℓ.

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

3.2.2 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* $(\mathbf{y}^*, \mathbf{z}^*)$ *to the relaxation* R'_ℓ *at node* ℓ *is not feasible for the original problem at node* ℓ*, 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* $(\mathbf{y}^*, \mathbf{z}^*)$ *to the relaxation* R'_ℓ *at node* ℓ *is not feasible for the original problem at node* ℓ*, then there exists either a type 1 pair or a type 2 pair or* $(\mathbf{y}^*, \bar{\mathbf{z}})$ *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'_ℓ*.*

Proof. Suppose that there is no type 1 or type 2 pair in $(\mathbf{y}^*, \mathbf{z}^*)$ and the solution $(\mathbf{y}^*, \bar{\mathbf{z}})$ is not an alternate optimal solution to the relaxation R'_ℓ. Then by Observation 1 there exists at least one pair $\{i, j\}$ such that $y_i^* = 1$, $y_j^* = 0$, $z_{ij}^* = 0$ and $z_{ji}^* = 1$. Because $(\mathbf{y}^*, \bar{\mathbf{z}})$ 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. Therefore, $y_t^* = 1$. However, $\bar{\zeta'}_t^i$ and consequently z_{ti}^* must be zero because k is covered uniquely by j in the subproblem Pr'_i. Then $\{i, t\}$ is a pair with $y_i^* = y_t^* = 1$ and $z_{ti}^* = 0$ and is either a type 1 or type 2 pair. This contradicts our assumption. □

Algorithm 1 BranchPair($\mathbf{y}^*, \mathbf{z}^*$)

```
for i ∈ N : y*_i = 1 do
    for j ∈ N : j > i, y*_j = 1 do
        if z*_ij = z*_ji = 0 then
            pair ← {i, j};
            break
if pair=null then
    for i ∈ N : y*_i = 1 do
        for j ∈ N : j > i, y*_j = 1 do
            if z*_ij ≠ z*_ji then
                pair ← {i, j};
                break
Return pair
```

3.2.3 Upper Bounds

There are two ways to update the upper bound in our algorithm: via the subproblems and via 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 $\mathbf{y}^*$ is an optimal solution of the relaxed master problem solved at any node of the branch-and-bound tree. Then $u^j = 1/2 \sum_{i' \in N_j} \sum_{j' \in N_j \setminus \{i'\}} p_{i'j'}$ for $j \in N$ and $u^0 = 1/2 \sum_{i' \in N'} \sum_{j' \in N' \setminus \{i'\}} p_{i'j'}$ are upper bounds for the optimal value.*

Proof. For each $j \in N$, because of constraints (3.10) in the relaxed subproblem, N_j is a capable team. Similarly, because of constraints (3.2) 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.

3.2.4 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 ℓ, we keep the optimal solution of the subproblem $\ell.\bar{\zeta}'^{\mathbf{n}}$ of Pr'_n, its optimal value $\ell.v'_n$ for all $n \in N$, the optimal value of the relaxed master problem $\ell.\nu'$ and its optimal solution $(\ell.\mathbf{y}^*, \ell.\mathbf{z}^*)$.

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. Because we preprocess our instances, we do not need to 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 whether 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 ℓ be a node in Q with the lowest lower bound. We remove ℓ from the queue and find its branch pair, say $\{i, j\}$. We create child nodes ℓ_1 and ℓ_2 and solve relaxations R'_{ℓ_1} and R'_{ℓ_2} as explained in Proposition 2. Node ℓ_1 (respectively, ℓ_2) is added to the queue only if $\ell_1.\nu'$ (respectively, $\ell_2.\nu'$) 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'_ℓ is

infeasible, then $\ell.\nu' = \infty$. In this case, we discard node ℓ because it does not satisfy $\ell.\nu' < UB$. This amounts to pruning by infeasibility. Furthermore, if the solution $(\mathbf{y}^*, \mathbf{z}^*)$ of relaxation R'_ℓ is feasible for the original problem or it is not feasible but $(\mathbf{y}^*, \bar{\mathbf{z}})$ 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'_ℓ, then $\ell.\nu' \geq UB$ because 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

$UB := \infty$, $T = \emptyset$
Create root node 0 with $0.\nu' := \infty, C^1_0 := \emptyset, C^2_0 := \emptyset$
for $n \in N$ **do**
    Solve Pr'_n
    $0.\bar{\zeta}'^{\mathbf{n}} := \bar{\zeta}'^{\mathbf{n}}$ and $0.v'_n := v'_n$ ▷ update UB and T if possible
Solve the relaxed master problem
$0.\mathbf{y}^* := \mathbf{y}^*$, $0.\mathbf{z}^* := \mathbf{z}^*$, $0.\nu' := \nu'$, $LB := \nu'$ ▷ update UB and T if possible
if $LB < UB$ **then** $Q := \{0\}$
while $LB < UB$ **do**
    $\ell = \arg\min_{\ell' \in Q} \{\ell'.\nu'\}$, $Q := Q \setminus \{\ell\}$
    $\{i, j\} := BranchPair(\ell.\mathbf{y}^*, \ell.\mathbf{z}^*)$
    Create node ℓ_1 : $\ell_1.v'_n = \ell.v'_n$, $\ell_1.\bar{\zeta}'^{\mathbf{n}} = \ell.\bar{\zeta}'^{\mathbf{n}}$ $\forall n \in N$,
     $\ell_1.\nu' := \infty, C^1_{\ell_1} := C^1_\ell \cup \{i, j\}$, $C^2_{\ell_1} := C^2_\ell$
    if $\ell.\bar{\zeta}'^i_j = 1$ **then**
        Solve Pr'_i
        if feasible **then** $\ell_1.v'_i := v'_i$, $\ell_1.\bar{\zeta}'^{\mathbf{i}} := \bar{\zeta}'^{\mathbf{i}}$
        else $\ell_1.v'_i := \infty$ ▷ update UB and T if possible
    if $\ell.\bar{\zeta}'^j_i = 1$ **then**
        Solve Pr'_j
        if feasible **then** $\ell_1.v'_j := v'_j$, $\ell_1.\bar{\zeta}'^{\mathbf{j}} := \bar{\zeta}'^{\mathbf{j}}$
        else $\ell_1.v'_j := \infty$ ▷ update UB and T if possible
    Solve relaxed master problem

if feasible **then**

$\ell_1.\mathbf{y}^* := \mathbf{y}^*$, $\ell_1.\mathbf{z}^* := \mathbf{z}^*$, $\ell_1.\nu' = \nu'$ ▷ update UB and T if possible

if $\ell_1.\nu' < UB$ **then** $Q := Q \cup \{\ell_1\}$

Create node ℓ_2 : $\ell_2.v'_n = \ell.v'_n$, $\ell_2.\bar{\zeta'}^{\mathbf{n}} = \ell.\bar{\zeta'}^{\mathbf{n}}$ $\forall n \in N$,

$\ell_2.\nu' = \infty, C^1_{\ell_2} := C^1_\ell$, $C^2_{\ell_2} := C^2_\ell \cup \{i, j\}$

if $\ell.\bar{\zeta'}^i_j = 0$ **then**

Solve Pr'_i

if feasible **then** $\ell_2.v'_i := v'_i$, $\ell_2.\bar{\zeta'}^{\mathbf{i}} := \bar{\zeta'}^{\mathbf{i}}$

else $\ell_2.v'_i := \infty$ ▷ update UB and T if possible

if $\ell.\bar{\zeta'}^j_i = 0$ **then**

Solve Pr'_j

if feasible **then** $\ell_2.v'_j := v'_i$, $\ell_2.\bar{\zeta'}^{\mathbf{j}} := \bar{\zeta'}^{\mathbf{j}}$

else $\ell_2.v'_j := \infty$ ▷ update UB and T if possible

Solve relaxed master problem

if feasible **then**

$\ell_2.\mathbf{y}^* := \mathbf{y}^*$, $\ell_2.\mathbf{z}^* := \mathbf{z}^*$, $\ell_2.\nu' := \nu'$ ▷ update UB and T if possible

if $\ell_2.\nu' < UB$ **then** $Q := Q \cup \{\ell_2\}$

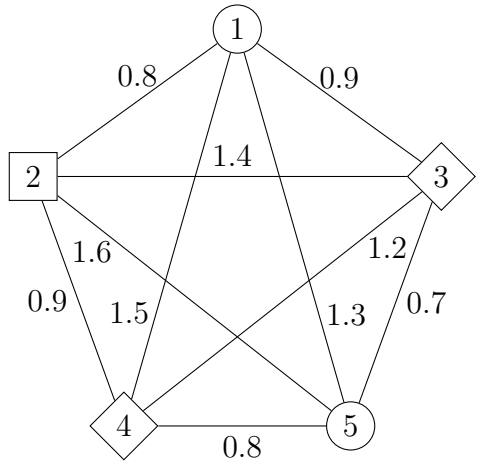
$LB := \min_{\ell' \in Q}\{\ell'.\nu'\}$

Return UB and T

3.2.5 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 3.2. 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.

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 3.2 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 $\bar{\zeta}^1_2 = \bar{\zeta}^1_3 = 1$. The team consisting of people 1, 2 and 3 has a



problem	team	cost
Pr'_1	{1,2,3}	3.1
Pr'_2	{1,2,4}	3.2
Pr'_3	{2,3,5}	3.7
Pr'_4	{2,4,5}	3.3
Pr'_5	{2,3,5}	3.7
master	{1,2,4}	3.2
lb=2.55	ub=3.1	

Figure 3.2: Example network, optimal solutions of the subproblems and the master and the bounds at the root node

cost 3.1. This is the upper bound we get from this subproblem and actually it is the 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 3.3. Next to each node, we summarize the solution and bound information in a table, similar to the one in Figure 3.2. 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}.

At node 1, we only solve the relaxed master problem since the solution of the relaxed subproblem 1 (respectively, 4) already satisfies $\bar{\zeta}'^1_4 = 0$ (respectively,

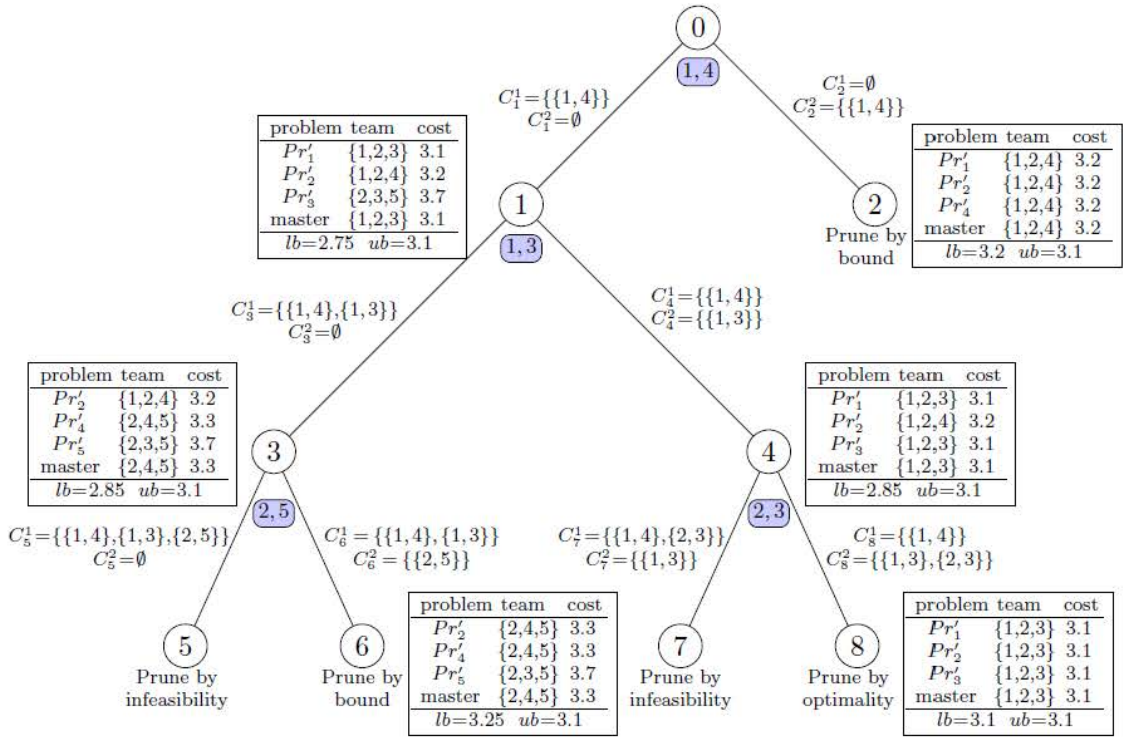


Figure 3.3: The branch-and-bound tree

$\bar{\zeta}'^4_1 = 0$). 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 because 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. Because 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. 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.

3.2.6 Branch-and-bound Algorithm for DC-TFP-SD

We can use a similar branch-and-bound algorithm to solve the 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 (3.3) in the formulation of the DC-TFP-SD. Also, recall that R'_ℓ is the weaker relaxation of the reformulation of the TFP-SD at node ℓ of the branch-and-bound tree.

For the DC-TFP-SD, we can treat the conflict constraints (3.3) 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 further relaxation. We define R''_ℓ to be the relaxation obtained by adding constraints (3.19) for all $\{i, j\} \in C$ to R'_ℓ. 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 the TFP-SD, the teams defined by these sets were capable so their cost values, u^j for $j \in N$ and u^0 gave upper bounds. In the 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 the DC-TFP-SD.

Using the relaxation R''_ℓ and this upper bounding procedure, we obtain valid lower and upper bounds. Next, we prove that if the optimal solution $(\mathbf{y}^*, \mathbf{z}^*)$ that we obtain by solving R''_ℓ does not satisfy the conflict constraints (3.3), then there exists a type 1 pair that we can branch on.

Proposition 5 *Let $(\mathbf{y}^*, \mathbf{z}^*)$ be the optimal solution of R''_ℓ. If there exists a pair $\{i, j\} \in C$ for which $(\mathbf{y}^*, \mathbf{z}^*)$ violates the conflict constraint* (3.3)*, i.e., $y^*_i = y^*_j = 1$, then $\{i, j\}$ is a type 1 branch pair.*

Proof. Suppose that $(\mathbf{y}^*, \mathbf{z}^*)$ violates the conflict constraint (3.3) for pair $\{i, j\} \in C$. Then $y^*_i = y^*_j = 1$. Since the subproblems for i and j contain constraints (3.19), we have $z^*_{ij} = z^*_{ji} = 0$. Then $\{i, j\}$ is a type 1 pair. □

3.3 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.

3.3.1 Datasets and Instance Generation

Wi et al. [23] use collaborative data from an R&D institute and form a social network of 45 researchers to test their genetic algorithm. Farasat and Nikolaev [25] use existing social network datasets to test their heuristics, and the number of nodes in these networks varies from 15 to 500. By contrast, 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. [82] and Kargar and An [26]. We

create our instances using the same part of the database used in the comparative study by Wang et al. [83]. The collaboration and skill information is provided by one of the authors on his website[1]. The nodes of the network are the actors who appeared in the movies from the year 2000 to 2002. There are 1021 actors; that is, $|N| = 1021$. 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 is equal to the Jaccard distance, as explained in Section 3.1.

DBLP is the most common database used to generate instances for the 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 the years 2010-2016. We narrow the search space by specifying journals and conferences. Because 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 3.4, 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.

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[2].

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

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

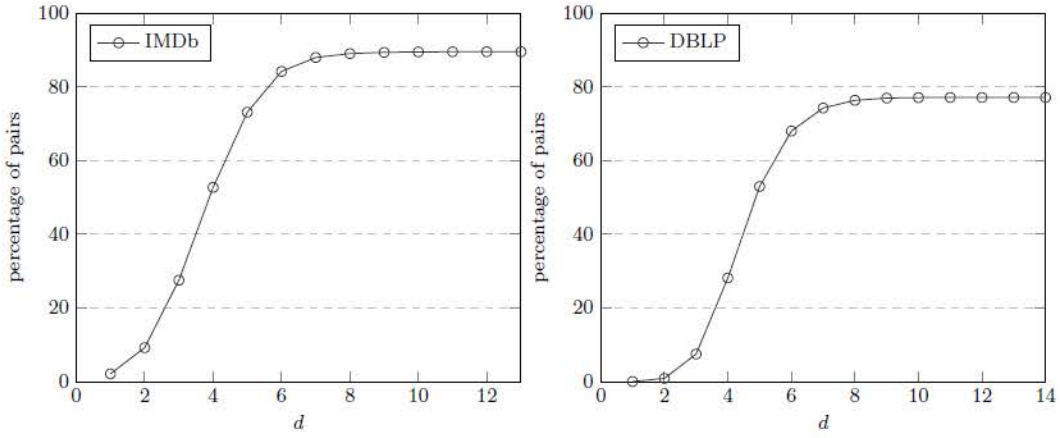


Figure 3.4: The percentage of pairs whose shortest distance is at most d in the IMDb (left) and DBLP (right) networks

3.3.2 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 what follows. 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 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 (3.1), (3.2), (3.4) (denoted by QP); the

mixed-integer formulation (3.2), (3.4)-(3.9) (denoted by MIP); and the branch-and-bound algorithm, we implemented a branch-and-cut algorithm for the TFP-SD to overcome the memory problems for larger instances. In the mixed-integer formulation, the constraints (3.6), (3.7) and (3.8) grow quadratically in the size of the problem. Because the objective coefficients are nonnegative in our instances, it is sufficient to use only constraints (3.6) but even in this case, we have memory run-outs in the model generation phase for large instances. When we use the original mixed-integer formulation without constraints (3.7) and (3.8) and add constraints (3.6) 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 mixed-integer formulation directly. However, when we add the RLT constraints (3.10), 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 (3.6) 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 (denoted by B&C), we solve the mixed-integer programming formulation (3.2), (3.4), (3.5), (3.9), (3.10) by putting constraints (3.6) to the lazy cut pool.

We report the average solution times of all solution procedures for the TFP-SD on the IMDb instances in Table 3.2. 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, $lb\text{-}gap = 100(opt - lb)/opt$ and $ub\text{-}gap = 100(ub - opt)/opt$, where lb and ub are the lower and upper bounds at the root node, respectively, and opt is the optimal value. To show the strength of the linear programming relaxation of the mixed-integer formulation (3.2), (3.4)-(3.9), we also report $LP\text{-}gap = 100(opt - LP)/opt$, where LP is the optimal value of the linear programming relaxation. As it can be seen in Table 3.2, the continuous relaxation is very weak. When we add the RLT constraints (3.10) to the continuous relaxation, the optimality gap improves tremendously. In 98% of IMBDb instances, this relaxation gives an integral, and consequently, an optimal solution. It is denoted by LP-RLT in Table 3.2 where we present the average

Table 3.2: Results for the TFP-SD on the IMDb instances.

		QP	MIP		LP-RLT		B&C	B&B			
m	*qno*	*time*	*time*	*LP-gap*	*time*	*gap*	*time*	*time*	*nodes*	*lb-gap*	*ub-gap*
4	422.51	6.66	7.14	63.6	2.43	0.22	0.81	1.13	2.08	2.12	0
6	541.81	22.63	23.21	77.75	4.68	0.49	1.74	2.66	14.11	4.12	0.05
8	653.41	28.5	29.54	77.07	7.74	0.36	3.16	4.19	24.6	5.95	0.06
10	731.82	30.41	31.12	75.47	15.66	0.86	5.63	5.92	41.97	10.27	0.3
12	791.51	32.6	33.47	75.9	22.42	0.86	7.59	7.28	52.36	12.31	0.22
14	838.48	43.13	44.7	74	37.66	1.02	10.62	9.83	111.34	13.31	0.5
16	879.02	51.81	53.04	72.76	54.03	1.17	15.57	12.27	157.72	13.58	0.18
18	917.68	83.76	81.04	71.92	84.32	1.43	18.77	14.31	164.98	15.13	0.77
20	947.69	77.98	78.54	71.23	134.32	1.65	24.93	13.93	167.69	16.24	0.7

solution time and the gap, which is calculated as $100(opt - LP\text{-}RLT)/opt$.

The performances of the quadratic and mixed-integer formulations for the TFP-SD turn out to be very similar for the IMDb instances. On average, the optimal solution is reported within a minute or two by the solver with both mathematical models. This is expected because in its default setting, the solver linearizes the quadratic formulation and solves it as an mixed-integer problem. Changing the solver settings for the quadratic formulation does not improve the solution times. When we compare these solution times with the branch-and-bound algorithm, we clearly see the efficiency of the algorithm as it reaches the optimal solution six times faster than the models, on 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 branch-and-bound algorithm. The longest time the branch-and-bound algorithm spends for an IMDb instance is actually 48.19 seconds. With the branch-and-cut, we are able to solve 98.6% of the instances within a minute while this percentage is 78% for both quadratic and mixed-integer formulations. When the number of required skills, m, is low, this method is as efficient as the branch-and-bound algorithm; but as m grows, the branch-and-bound algorithm outperforms the branch-and-cut as well. Analyzing the detailed results, we observe that for all instances with $m = 4$ the first incumbent found by the branch-and-bound 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 3.3: Results for the TFP-SD on the DBLP instances.

		QP		MIP		LP-RLT		B&C		B&B	
m	qno	*solved*	*time*	*solved*	*time*	*solved*	*time*	*solved*	*time*	*solved*	*time*
4	1650.5	10	343.04	10	359.75	10	185.87	10	53.95	10	9.84
6	2239.8	4	336.79	4	352.96	5	81.67	10	142.59	10	20.65
8	2896.5	2	386.29	2	2508.42	4	453.93	8	279.39	10	36.56

In Table 3.3, we present the results for the TFP-SD on the DBLP instances. Because 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 with the mixed-integer and quadratic formulations we can only solve four instances with $m = 6$ and two instances with $m = 8$. The instances solved without memory problems are the same for the mixed-integer and the quadratic formulations, however, the solution times with the quadratic formulation are lower than those of with the mixed-integer formulation. This difference in the solution times is unexpected since the solver linearizes the objective function and solves the quadratic formulation as a mixed-integer problem as well. Examining the log file of the solver, we realize that given the quadratic formulation, the solver generates an additional cut, which seems to speed up the branch-and-cut process.

LP-RLT, the continuous relaxation that is strengthened by the RLT constraints, gives integral solutions to the DBLP instances for which the memory is sufficient. By strengthening the mixed-integer formulation with RLT constraints and putting constraints (3.6) to the lazy cut pool in the branch-and-cut framework enables us to solve more instances within less time. However, eventually, this method also fails with memory problems as the size of instances increases. Having average solution times under a minute, the efficiency of the branch-and-bound algorithm is clearly seen in this table. Its longest solution time among these instances is actually 62.2 seconds.

Table 3.4: Detailed results of the branch-and-bound algorithm for the TFP-SD on the DBLP instances.

			time						
m	*qno*	*solved*	*min*	*avg*	*max*	*std*	*nodes*	*lb-gap*	*ub-gap*
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 branch-and-bound algorithm on the DBLP instances in Table 3.4. We also consider 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 are 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 a 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, because of 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 a 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 3.5, we present

the results on these instances for the TFP-SD, where the new sets are denoted by IMDb^r and DBLP^r.

Table 3.5: Results of the branch-and-bound algorithm for the TFP-SD on IMDb^r and DBLP^r: the IMDb and DBLP instances with randomly generated skill matrices.

	IMDb^r					DBLP^r				
m	*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

Each one of these instances is solved within the two-hour time limit by the branch-and-bound algorithm. The solution times of the random IMDb^r instances are higher than those of the original ones presented in Table 3.2 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, the DC-TFP-SD. In what follows, QP represents the quadratic formulation ((3.1)-(3.4)), MIP represents the mixed-integer formulation ((3.2)-(3.9)), and B&B represents the branch-and-bound algorithm developed for the DC-TFP-SD.

In Table 3.6, we present the solution times for the DC-TFP-SD, where we use the optimal diameter values as the diameter bounds. We remind that *qno* is the number of qualified people given the required skills and fno is the number of people after preprocessing given the bound on the diameter. Because of preprocessing, the network is reduced significantly, and therefore, the solution times of all methods are very small. For example when $m = 20$, the network consists of more than 900 qualified people on average but the number of candidates reduces

Table 3.6: Results for the DC-TFP-SD on the IMDb instances where the bound on the diameter is taken as the optimal diameter

			QP	MIP	B&B			
m	qno	fno	$time$	$time$	$time$	$nodes$	$lb\text{-}gap$	$ub\text{-}gap$
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

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 3.7: Results for the DC-TFP-SD on the IMDb instances

	D=2					D=3				
m	$feas$	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 the DC-TFP-SD with the IMDb instances with varying bounds on the diameter. In Table 3.7 we present the results with D equal to 2 and 3. Under the column $feas$, the number of 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 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.

Table 3.8: Results for the DC-TFP-SD on the DBLP instances

	D=2							D=3						
		MIP		QP		B&B			MIP		QP		B&B	
m	*feas*	*solved*	*time*	*solved*	*time*	*solved*	*time*	*feas*	*solved*	*time*	*solved*	*time*	*solved*	*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 the DBLP instances, 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 for $D = 2$ and $D = 3$ in Table 3.8. For these instances, the average solution time of MIP is a 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 $D = 3$ and m exceeds 6. The B&B algorithm, by contrast, is able to solve each of these instances under 30 seconds.

In Table 3.9, we present the results for all DBLP instances using the branch-and-bound algorithm for the 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.

Table 3.9: Results of the branch-and-bound algorithm for the DC-TFP-SD on the DBLP instances

	D=1			D=2			D=3			D=4		
m	$feas$	$solved$	$time$	$feas$	$solved$	$time$	$feas$	$solved$	$time$	$feas$	$solved$	$time$
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	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

3.4 Conclusion

In this study, we formulated the team formation problem as a quadratic set covering problem with packing constraints. In terms of the problem definition, our study is close to the ones in data science literature but none of those studies gave a mathematical formulation to the problem. In operations research literature, the closest studies, ([23], [24], [25]), had a multi-objective structure and although the problems were formulated as integer programs, heuristics were suggested as solution methods. We showed that the quadratic and mixed-integer programming formulations of the problem can be solved by a commercial solver for instances up to 2000 candidates. To solve larger sizes to optimality, we developed a novel branch-and-bound algorithm. The algorithm uses a relaxation that can be solved by solving a series of linear set covering problems and utilizes a different branching rule compared to existing branch-and-bound methods for quadratic 0-1 optimization problems. With computational experiments we showed 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. In the next chapter, we tackle a two-stage team formation problem in which uncertainty is present in the communication costs.

Chapter 4

Stochastic Team Formation Problem

In this chapter, we study a two-stage stochastic team formation problem where the quality of communication is uncertain for some pairs in the candidate pool but the first stage allows observing the communication of a limited number of such pairs. Then according to their observations, the decision-maker builds a team, which is capable of the required tasks and can communicate effectively.

In classical two-stage stochastic programming, the actual realization of the random variables are assumed to be known after the first stage, hence the uncertainty resolves completely in the second stage. The uncertainty in this problem is endogenous or decision-dependent because the resolution of uncertainty depends on the decisions. Among the random communication costs, we only learn the true value of the ones that are chosen to be observed in the first stage. This means that we assume an immediate resolution in this problem.

In Section 4.1, we formally define the two-stage stochastic team formation problem. In Section 4.2 we give two mathematical formulations for the problem and prove their equivalence. In Section 4.3, we present our Benders'

decomposition-based branch-and-cut algorithm. First, we explain the decomposition structure, the strengthening process and cuts, then we describe how the number of problems solved at each iteration is decreased by the creation of new scenario sets. We conclude this section by the summary of the proposed algorithm. In Section 4.4, we first explain the instance generation process and pre-process applied to the instances, then we provide the results of our computational experiments where we compare performances of both mathematical formulations and different versions of the algorithm. The chapter is concluded with a brief summary and possible extensions in Section 4.5. The work in this chapter is based on [84].

4.1 Problem Definition and Value of Learning

As we mention in Section 2.1, different methods are chosen to quantify the communication cost among team members such as personality tests, familiarity scores, and direct evaluations of candidates. It is difficult to single out the best method to quantify communication, each having its own advantage, and we leave this up to the decision-maker. With respect to the method the decision-maker sees fit, they will require some type of data such as answers to questionnaires or collaboration information. Unfortunately, it is not always possible to obtain such data completely, especially when there are new people in the respective community, whether it is a company or an online platform. Therefore, the communication costs of the pairs, for which sufficient data is missing, are subject to uncertainty.

When there is uncertainty in the communication costs, an opportunity to observe and learn the true cost value may lead to a better team selection. As explained in Chapter 1, the quality of communication has a significant impact on the performance of a team. When there is uncertainty in the communication costs, if we build a team considering the expected or estimated communication costs, we may end up with a team with poor communication. In this case, we can either continue with this team or after identifying the pairs who cannot collaborate effectively, we can replace one or more team members accordingly. In

the first option, we accept the performance of the team will be limited so we will encounter some drawbacks such as low work quality or lengthened project duration. In the second option, by changing the team members we might have a better team in terms of communication. However, these changes disrupt the project and it may decrease the work pace as the new members will require time to be familiar with the tasks.

Instead of building the team and making a change in the team structure during the project, it might be more preferable to obtain more information about the communication costs and make a more educated decision. In other words, with an opportunity to observe some of the uncertain communication costs we can make a better team selection. In a company, this can be done by assigning small tasks to people in pairs so that their communication can be observed before the construction of the actual team. This amounts to putting more effort in the beginning of team formation process rather than dealing with changes in team members during the project.

With this motivation, we study a two-stage problem; in the first stage a limited number of communication costs are observed and a capable team with minimum expected communication cost is built in the second stage. In this study, the pairs, whose communication cost is uncertain, are represented by a set and the possible cost values are represented by scenarios. In other words, the communication costs are taken as discrete random variables. In what follows, we explain the two-stage stochastic team formation problem in more detail, introducing the notation first and describing the type and resolution of uncertainty next, over a small example.

There is a project that requires a team with a set of required skills. Let K be the set of those skills and let N be the set of candidates among which team members are selected. We assume that the skills of the candidates are known and represented by parameter a_{ik}, which equals to one if person $i \in N$ possesses skill $k \in K$ and to zero otherwise. A team is *capable* if for each required skill, there is at least one team member possessing that skill. Let $E = \{\{i, j\} : i, j \in N, i < j\}$ and $M \subset E$ be the set of pairs whose communication cost is random. We assume that possible values these costs can take and respective probabilities

are known, and we create the scenario set S as the cross product of all possible realizations. We define parameter c_{ij}^s as the communication cost between i and j under scenario $s \in S$. We use $\bar{c}_{ij}$ to represent the mean communication cost between i and j. For the pairs whose communication cost is known with certainty, the value of the corresponding parameter is the same under all scenario, that is, $\bar{c}_{ij} = c_{ij}^s$ for all $\{i, j\} \in E \setminus M$. The communication costs are assumed to be independent, hence the probability of scenario s, p_s, is equal to the product of individual communication cost probabilities. Clearly (N, E) constitutes an undirected complete graph, where c_{ij}^s is the cost or weight of edge $\{i, j\}$ under scenario s. Therefore, we will sometimes use the terms *random* or *uncertain edges* to refer to the pairs whose communication cost is random.

In the first stage, the decision is to choose the uncertain pairs to be observed and the budget allows us to select u pairs. After observing the true cost values of these pairs, we form a capable team with the minimum expected cost in the second stage. We define the communication cost of a team as the sum of all pairwise communication costs. Next, we illustrate the problem on a small example and introduce the concept of *value of learning*.

In Figure 4.1 we have a network where four nodes represent four candidates. We want to form a capable team for a project that requires three skills and the shapes of nodes indicate the skill each person has. In this example, we have two capable teams: {1, 2, 3} and {1, 3, 4}. The values on the solid edges are the true communication costs of the corresponding pair. The values on the dashed edges

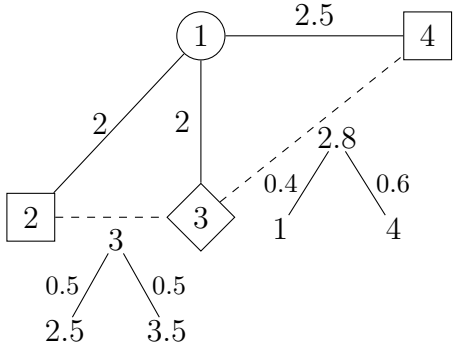


Figure 4.1: A social network with uncertain edges {2,3} and {3,4}

s	p_s	c^s_{23}	c^s_{34}
1	0.2	2.5	1
2	0.3	2.5	4
3	0.2	3.5	1
4	0.3	3.5	4

Table 4.1: Scenarios of the small example

are the mean values of the random costs. The possible values these costs can take are written with the respective probabilities. The cost of edge $\{2, 3\}$ is 2.5 or 3.5 with equal probabilities and the cost of edge $\{3, 4\}$ is 1 with probability 0.4 and 4 with probability 0.6. So we have $M = \{\{2, 3\}, \{3, 4\}\}$ and we have 4 scenarios summarized in Table 4.1.

Taking $u = 1$ we will solve the problem by enumeration. First, we note that the expected cost of the team {1, 2, 3} is 7 and that of {1, 3, 4} is 7.3. Because $u = 1$, we can either observe $\{2, 3\}$ or $\{3, 4\}$ in the first stage. If we select $\{2, 3\}$ and its true cost reveals to be 2.5 then we know that the cost of the team {1, 2, 3} is 6.5. Since this is less than 7.3 the optimal solution is {1, 2, 3}. If the true cost of 3.5 then the cost of {1, 2, 3} is 7.5, which is greater than 7.3. In this case, the optimal solution is to select the team {1, 3, 4}. Both of these possibilities being equally likely, the expected cost of this solution becomes $0.5 \times 6.5 + 0.5 \times 7.3 = 6.9$.

If $\{3, 4\}$ is selected in the first stage, with probability 0.4 its true cost is 1, in which case the cost of the team {1, 3, 4} is 5.5. This is better than 7 so we choose this team. With probability 0.6, the cost of edge $\{3, 4\}$ is 4, which makes the cost 8.5 for the team {1, 3, 4}. Then in this case the best team becomes {1, 2, 3}. The minimum expected cost of this first stage solution is $0.4 \times 5.5 + 0.6 \times 7 = 6.4$. Therefore, the optimal first stage decision is to select $\{3, 4\}$.

Recall that the expected costs of the possibles teams were 7 and 7.3 so if we chose without any observation we would choose the team {1, 2, 3} with expected cost 7. With this learning opportunity, we decreased the cost to 6.4. We call this difference between the expected value problem and two-stage stochastic problem as *the value of learning* (VE). This definition can be regarded as the adaptation

of the value of stochastic solution (VSS) concept to our problem. In classical two-stage stochastic programming with complete recourse, VSS is defined by the difference between the result of using an expected value solution (EEV) and the recourse problem solution [85]. And EEV is calculated by solving the mean value problem, fixing the first stage solutions and solving for each scenario in the second stage independently. EVV, and consequently VSS, do not apply to our problem since the first stage does not have a meaning alone. The expected value of perfect information (EVPI) concept, which is the difference between the result of recourse problem and wait-and-see, however, does apply to our problem. In our example, EVPI is zero because the result of wait-and-see solution is 6.4, same as the result of the stochastic problem.

4.2 Formulations

In our problem, the resolution of uncertainty depends on the first stage decisions. In the example presented in the previous section, since $u = 1$, we are allowed to observe only one of the random edges. When we choose to observe edge {2,3}, we have no information on edge {3,4}. Consequently, we are not capable of distinguishing scenarios 1 and 2, and also 3 and 4, because we have only learned the true cost of edge {2,3}. Therefore, the second stage decisions under scenarios 1 and 2, and 3 and 4, must be identical. In the literature of the problems with type 2 endogenous uncertainty, this is enforced by a set of constraints called *conditional non-anticipativity constraints*. The name comes from their similarity to non-anticipativity constraints used in multi-stage stochastic programming. Regular non-anticipativity constraints ensure that the decisions under the scenarios, which have the same history up to the current stage, are identical. In problems with endogenous uncertainty, these constraints are enforced on pairs of scenarios, which are indistinguishable after the observations in the previous stages. Because of this condition on indistinguishability, they are called *conditional* non-anticipativity constraints and abbreviated as CNAC. As we will explain in detail, these constraints are very large in nature and lead to computationally hard integer programming formulations. Therefore, in this study, we develop an alternative

integer programming model which do not require these constraints.

We define binary variable w_{ij} to be one if edge $\{i,j\} \in M$ is selected to be observed in the first stage and zero otherwise. In the second stage, we have binary variable y_i^s which is one if person $i \in N$ is in the team under scenario $s \in S$, and zero otherwise. We also define $z_{ij}^s = y_i^s y_j^s$ for all $\{i,j\} \in E$, and for all $s \in S$. Next, we define the sets which are used to write the CNACs. Let $D_{ss'}$ to be the set of edges that distinguishes scenarios s and s', i.e., $D_{ss'} = \{\{i,j\} : c_{ij}^s \neq c_{ij}^{s'}\}$. We could write the CNACs for all scenario pairs but since some constraints imply the others it is sufficient to write the constraints for a subset of the scenario pairs. We denote this subset by $\mathcal{L}$. As proved in the study by Gupta and Grossmann [56], and later generalized by Boland et al. [55], when the scenario set is defined by the cross product of all possible realizations, to have the minimum number of pairs, we can define $\mathcal{L}$ by the pairs which have only one different parameter, that is, $\mathcal{L} = \{\{s,s'\} : s,s' \in S, s < s', |D_{ss'}| = 1\}$. Then in line with the models in the literature, we can formulate our problem with the CNACs as follows:

$$\min \sum_{s \in S} \sum_{\{i,j\} \in E} p_s c_{ij}^s z_{ij}^s \tag{4.1}$$

$$\text{st.} \sum_{\{i,j\} \in M} w_{ij} = u, \tag{4.2}$$

$$\sum_{i \in N} a_{ik} y_i^s \geq 1 \qquad \forall k \in K, s \in S, \tag{4.3}$$

$$z_{ij}^s \leq y_i^s \qquad \forall i,j \in N : i < j, s \in S, \tag{4.4}$$

$$z_{ij}^s \leq y_j^s \qquad \forall i,j \in N : i < j, s \in S, \tag{4.5}$$

$$z_{ij}^s \geq y_i^s + y_j^s - 1 \qquad \forall i,j \in N : i < j, s \in S, \tag{4.6}$$

$$z_{ij}^s \geq 0 \qquad \forall \{i,j\} \in E, s \in S, \tag{4.7}$$

$$y_i^s \in \{0,1\} \qquad \forall i \in N, s \in S, \tag{4.8}$$

$$w_{ij} \in \{0,1\} \qquad \{i,j\} \in M, \tag{4.9}$$

$$y_i^s - y_i^{s'} \leq w_{mn} \qquad \forall i \in N, \{s,s'\} \in \mathcal{L}, \{m,n\} = D_{ss'}, \tag{4.10}$$

$$y_i^{s'} - y_i^s \leq w_{mn} \qquad \forall i \in N, \{s,s'\} \in \mathcal{L}, \{m,n\} = D_{ss'}. \tag{4.11}$$

Constraints (4.2) and (4.9) ensure that u pairs are selected to be observed

in the first stage. Constraints (4.3) are the covering constraints that guarantee existence of at least one capable person in the team for each required skill under each scenario. Constraints (4.4)-(4.7) imply $z_{ij}^s = y_i^s y_j^s$ for all $\{i,j\} \in E, s \in S$. (4.10) and (4.11) are the conditional non-anticipativity constraints. In these constraints, the right-hand side is equal to zero when the edge distinguishing scenarios s and s' is not selected in the first stage, and consequently, it enforces the decisions under those scenarios to be equal to each other. The objective function is the expected cost of the team built in the second stage. Having $|M|$ random parameters and each having two possible values, the number of CNACs in this formulation is $|N||M|2^{|M|}$ which makes the problem computationally difficult.

The following formulation, which does not have CNACs (4.10) and (4.11), is an alternative formulation to the same problem:

$$\min \sum_{s\in S} p_s \Bigg(\sum_{\{i,j\}\in E} \bar{c}_{ij} z_{ij}^s + \sum_{\{i,j\}\in M} (c_{ij}^s - \bar{c}_{ij}) w_{ij} z_{ij}^s \Bigg) \tag{4.12}$$

$$\text{st. } (4.2) - (4.9)$$

Except for the absence of the conditional non-anticipativity constraints, the only difference of this formulation with the previous one is the objective function (4.12). When $\{i,j\} \in M$ is not chosen in the first stage, we do not know its true cost value. In this case, we can directly use the mean cost value since the aim is to minimize the expected cost. Therefore, when $w_{ij} = 0$, the cost term related to this pair is $\bar{c}_{ij} z_{ij}^s$ in the objective (4.12) for any scenario s. When the pair is chosen, that is, $w_{ij} = 1$, we learn its true cost value, which is c_{ij}^s with probability p_s. In this case, $\bar{c}_{ij} z_{ij}^s + (c_{ij}^s - \bar{c}_{ij}) w_{ij} z_{ij}^s$ in (4.12) becomes $c_{ij}^s z_{ij}^s$ so we use the communication cost c_{ij}^s under scenario s.

We linearize the quadratic objective function by defining variable $v_{ij}^s = w_{ij} z_{ij}^s$ for all $\{i,j\} \in M$ and $s \in S$:

$$\min \sum_{s\in S} p_s \Bigg(\sum_{\{i,j\}\in E} \bar{c}_{ij} z_{ij}^s + \sum_{\{i,j\}\in M} (c_{ij}^s - \bar{c}_{ij}) v_{ij}^s \Bigg) \tag{4.13}$$

$$\text{st. } (4.2) - (4.9),$$

$$v_{ij}^s \le z_{ij}^s \qquad \{i,j\} \in M, s \in S, \tag{4.14}$$

$$v_{ij}^s \leq w_{ij} \qquad \{i,j\} \in M, s \in S, \ (4.15)$$

$$v_{ij}^s \geq z_{ij}^s + w_{ij} - 1 \qquad \{i,j\} \in M, s \in S, \ (4.16)$$

$$v_{ij}^s \geq 0 \qquad \{i,j\} \in M, s \in S. \ (4.17)$$

Constraints (4.14)-(4.17) are to ensure $v_{ij}^s = w_{ij} z_{ij}^s$ for all $\{i,j\} \in M$ and $s \in S$. We denote the above formulation (4.2)-(4.9), (4.13)-(4.17) by *IF*, and the formulation with the CNACs (4.1)-(4.11) by *CF*.

Observation 2 *Let $(\tilde{\mathbf{w}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}})$ be a feasible solution of CF. Let $\tilde{H}$ be the set of edges whose cost will be realized after the first stage, that is, $\tilde{H} := \big\{\{i,j\} \in M : \tilde{w}_{ij} = 1\big\}$. If the number of different weights $\{i,j\} \in \tilde{H}$ can have is r_{ij}, then $\tilde{q} = \prod_{\{i,j\} \in \tilde{H}} r_{ij}$ is the number of different realizations we can see given $\tilde{\mathbf{w}}$. We can partition the scenario set S to the sets $\tilde{S}_n$ for $n = 1, \ldots, \tilde{q}$ such that the cost values of the selected edges are the same for all scenarios in $\tilde{S}_n$, that is, $c_{ij}^s = c_{ij}^{s'}$ for $\{i,j\} \in \tilde{H}$ for any $s, s' \in \tilde{S}_n$. For any n, the scenarios in $\tilde{S}_n$ are actually the ones that are indistinguishable. Therefore, because of (4.10)-(4.11) in CF, we have $\tilde{\mathbf{y}}^s = \tilde{\mathbf{y}}^{s'}$, and consequently $\tilde{\mathbf{z}}^s = \tilde{\mathbf{z}}^{s'}$, for any $s, s' \in \tilde{S}_n$.*

Proposition 6 *IF and CF are equivalent formulations in the following sense. For any feasible solution of CF there is a corresponding feasible solution of IF with the same objective value. For any feasible solution of IF, either there is a corresponding feasible solution of CF with the same objective value or this feasible solution is not optimal for IF.*

Proof. Let $(\tilde{\mathbf{w}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}})$ be a feasible solution of CF. Then $(\tilde{\mathbf{w}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}}, \tilde{\mathbf{v}})$, where $\tilde{\mathbf{v}} = \tilde{\mathbf{z}} \circ \tilde{\mathbf{w}}$, is clearly a feasible solution of IF, where $\circ$ denotes the Hadamard product (element-wise multiplication). We prove the equivalence of the objective values of these solutions term by term with respect to edges.

- For $\{i,j\} \in E \setminus M$, the terms in the objective functions (4.1) and (4.13) are identical since $c_{ij}^s = \bar{c}_{ij}$ for all $s \in S$.

- For $\{i,j\} \in M$ with $\tilde{w}_{ij} = 1$, we have $\tilde{v}^s_{ij} = \tilde{z}^s_{ij}$ and consequently in (4.13) we have $\bar{c}_{ij}\tilde{z}^s_{ij} + (c^s_{ij} - \bar{c}_{ij})\tilde{v}^s_{ij} = c^s_{ij}\tilde{z}^s_{ij}$, which is identical to the term in (4.1).

- For $\{i,j\} \in M$ with $\tilde{w}_{ij} = 0$, we have $\tilde{v}^s_{ij} = 0$ so it is sufficient to show that $\sum_{s\in S} p_s c^s_{ij}\tilde{z}^s_{ij} = \sum_{s\in S} p_s \bar{c}_{ij}\tilde{z}^s_{ij}$. Let $\tilde{S}_1, \ldots, \tilde{S}_{\tilde{q}}$ be the partition of the scenarios with respect to $\tilde{w}$ as described in Observation 2. We define $\tilde{\zeta}^n_{ij} = \tilde{z}^s_{ij}$ for all $s \in \tilde{S}_n$ and for $n \in \{1, \ldots, \tilde{q}\}$ using the fact that the second stage decisions under scenarios $\tilde{S}_n$ are identical. The sum of probabilities of scenarios in set $\tilde{S}_n$ is $\tilde{p}_n = \sum_{s\in\tilde{S}_n} p_s$ for $n \in \{1, \ldots, \tilde{q}\}$. Then for the term in (4.13) we can write:

$$\sum_{s\in S} p_s \bar{c}_{ij}\tilde{z}^s_{ij} = \sum_{n=1}^{\tilde{q}} \sum_{s\in\tilde{S}_n} p_s \bar{c}_{ij}\tilde{z}^s_{ij} = \sum_{n=1}^{\tilde{q}} \tilde{\zeta}^n_{ij}\bar{c}_{ij} \sum_{s\in\tilde{S}_n} p_s = \sum_{n=1}^{\tilde{q}} \tilde{\zeta}^n_{ij}\bar{c}_{ij}\tilde{p}_n \qquad (*)$$

  Let ${}^1c_{ij}, \ldots, {}^rc_{ij}$ be the possible cost values of $\{i,j\}$ with probabilities $\rho^1_{ij}, \ldots, \rho^r_{ij}$ so $\sum_{l=1}^{r} \rho^l_{ij}\,{}^lc_{ij} = \bar{c}_{ij}$. We can partition each $\tilde{S}_n$ to sets $\tilde{S}^1_n, \ldots, \tilde{S}^r_n$ such that for $s \in \tilde{S}^l_n$, $c^s_{ij} = {}^lc_{ij}$ for $l \in \{1, \ldots, r\}$. The sum of probabilities of the scenarios in $\tilde{S}^l_n$ equals to $\rho^l_{ij}\tilde{p}_n$. Then for the term in (4.1) we can write:

$$\begin{aligned}\sum_{s\in S} p_s c^s_{ij}\tilde{z}^s_{ij} &= \sum_{n=1}^{\tilde{q}} \sum_{s\in\tilde{S}_n} p_s c^s_{ij}\tilde{z}^s_{ij} = \sum_{n=1}^{\tilde{q}} \tilde{\zeta}^n_{ij} \sum_{s\in\tilde{S}_n} p_s c^s_{ij} = \sum_{n=1}^{\tilde{q}} \tilde{\zeta}^n_{ij} \sum_{l=1}^{r} \sum_{s\in\tilde{S}^l_n} p_s\,{}^lc^s_{ij} \\ &= \sum_{n=1}^{\tilde{q}} \tilde{\zeta}^n_{ij} \sum_{l=1}^{r} \rho^l_{ij}\tilde{p}_n\,{}^lc_{ij} = \sum_{n=1}^{\tilde{q}} \tilde{\zeta}^n_{ij}\tilde{p}_n \sum_{l=1}^{r} \rho^l_{ij}\,{}^lc_{ij} = \sum_{n=1}^{\tilde{q}} \tilde{\zeta}^n_{ij}\tilde{p}_n\bar{c}_{ij} \quad (**)\end{aligned}$$

  From (*) and (**), we conclude that the terms in (4.1) and (4.13) are identical.

Now let $(\tilde{\mathbf{w}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}}, \tilde{\mathbf{v}})$ be a feasible solution of IF. If $(\tilde{\mathbf{w}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}})$ is feasible for CF then their objective values are equal as proved above. If $(\tilde{\mathbf{w}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}})$ is not feasible for IF, then there exists at least one scenario pair $\{s, s'\}$ with $s < s'$ such that $\tilde{w}_{mn} = 0$ for $D_{ss'} = \{m, n\}$ but $\tilde{y}^s_i \neq \tilde{y}^{s'}_i$ for some $i \in N$. Let $\tilde{\mathcal{L}}$ be the set of such scenario pairs ordered lexicographically. Let $\tilde{\nu}^s = \sum_{\{i,j\}\in E} \bar{c}_{ij}z^s_{ij} + \sum_{\{i,j\}\in M} (c^s_{ij} - \bar{c}_{ij})v^s_{ij}$.

- If $\tilde{\nu}^s = \tilde{\nu}^{s'}$ for all $\{s, s'\} \in \tilde{\mathcal{L}}$, then we update $\tilde{\mathbf{y}}^{\mathbf{s}'} := \tilde{\mathbf{y}}^{\mathbf{s}}, \tilde{\mathbf{z}}^{\mathbf{s}'} := \tilde{\mathbf{z}}^{\mathbf{s}}$, $\tilde{\mathbf{v}}^{\mathbf{s}'} := \tilde{\mathbf{v}}^{\mathbf{s}}$ for all $\{s, s'\} \in \tilde{\mathcal{L}}$ in order. The updated solution is feasible for CF and has the same objective value.

- If there exist $\{s, s'\} \in \mathcal{L}$ with $\tilde{\nu}^s < \tilde{\nu}^{s'}$ ($\tilde{\nu}^s > \tilde{\nu}^{s'}$) then the solution is not optimal because updating $\tilde{\mathbf{y}}^{\mathbf{s}'} := \tilde{\mathbf{y}}^{\mathbf{s}}, \tilde{\mathbf{z}}^{\mathbf{s}'} := \tilde{\mathbf{z}}^{\mathbf{s}}, \tilde{\mathbf{v}}^{\mathbf{s}'} := \tilde{\mathbf{v}}^{\mathbf{s}}$ ($\tilde{\mathbf{y}}^{\mathbf{s}} := \tilde{\mathbf{y}}^{\mathbf{s}'}, \tilde{\mathbf{z}}^{\mathbf{s}} := \tilde{\mathbf{z}}^{\mathbf{s}'}, \tilde{\mathbf{v}}^{\mathbf{s}} := \tilde{\mathbf{v}}^{\mathbf{s}'}$) we obtain a better solution.

□

4.3 Branch-and-Cut Algorithm

Although IF, the alternative formulation without CNACs, has much less constraints compared to CF, the formulation with CNACs, as the number of scenarios increases, the size of this formulation becomes too large to be solved with a commercial integer programming solver as well. Therefore, we develop an integer L-shaped algorithm, which is a decomposition-based branch-and-cut algorithm, to solve our two-stage stochastic team formation problem. The algorithm is based on the formulation IF. The L-shaped method, which is based on Benders' decomposition [86], was first proposed by Van Slyke and Wets [87] for two-stage stochastic linear programs with continuous variables. Laporte and Louveaux [88] developed the integer L-shaped method for two-stage problems with integer variables in both stages. The algorithm is embedded into a branch-and-bound procedure to ensure optimality. It utilizes cuts that require optimal objective value of the second stage problems and a valid lower bound for those. This means integer programs are required to be solved at each iteration. Therefore, the performance of the algorithm directly depends on the computational difficulty of those integer programs and obtaining a good lower bound.

We propose an algorithm where the duality-based optimality cuts are obtained by a stronger linear relaxation of the second stage problems by applying the well-known reformulation-linearization technique of Adams and Sherali [76]. This

stronger relaxation not only generates stronger cuts but also decreases the computational burden of solving the integer problems by providing integral solutions often. Moreover, by exploiting the problem structure, at each iteration of the algorithm we generate an alternative scenario set, which is much smaller than the original one and the second stage problems are solved over those scenarios. Lastly, using the fact that solving integer second stage problems gives an incumbent, we pass that upper bound information to the solver and add a cut to eliminate the corresponding first stage solution.

In the following subsection, we explain the decomposition of the problem, present the stronger linear relaxation. We describe the generation of the new scenarios and the cuts. Then we explain the algorithm step by step.

4.3.1 The Decomposition and Cuts

We can rewrite IF using the standard two-stage stochastic programming representation as follows:

$$\begin{aligned}
&\min \sum_{s\in S} p_s Q_s(\boldsymbol{w}) \\
&\text{s.t.} \\
&\quad \sum_{\{i,j\}\in M} w_{ij} = u, \\
&\quad w_{ij} \in \{0,1\} \qquad \forall \{i,j\} \in E.
\end{aligned}$$

where, for each $s \in S$ we have

$$\begin{aligned}
Q_s(\hat{\boldsymbol{w}}) = \quad \min \; & \sum_{\{i,j\}\in E} \bar{c}_{ij} z^s_{ij} + \sum_{\{i,j\}\in M} (c^s_{ij} - \bar{c}_{ij}) v^s_{ij} \\
\text{st.} \; & \sum_{i\in N} a_{ik} y^s_i \geq 1 && \forall k \in K, \\
& z^s_{ij} \leq y^s_i && \forall \{i,j\} \in E, \\
& z^s_{ij} \leq y^s_j && \forall \{i,j\} \in E, \\
& z^s_{ij} \geq y^s_i + y^s_j - 1 && \forall \{i,j\} \in E,
\end{aligned}$$

$$
\begin{aligned}
& z_{ij}^s \geq 0 && \forall \{i,j\} \in E, \\
& y_i^s \in \{0,1\} && \forall i \in N, \\
& v_{ij}^s \leq z_{ij}^s && \{i,j\} \in M, \\
& v_{ij}^s \leq \hat{w}_{ij} && \{i,j\} \in M, \\
& v_{ij}^s \geq z_{ij}^s + \hat{w}_{ij} - 1 && \{i,j\} \in M, \\
& v_{ij}^s \geq 0 && \{i,j\} \in M.
\end{aligned}
$$

Let $Q_s^*(\hat{\boldsymbol{w}})$ be the optimal objective function value of problem $Q_s(\hat{\boldsymbol{w}})$. Clearly the feasibility of $Q_s(\hat{\boldsymbol{w}})$ for all $s \in S$ does not depend on $\hat{\boldsymbol{w}}$ but on the existence of a capable team. Hence the feasibility of the second stage amounts to feasibility of the whole problem and we assume that it is checked beforehand. Therefore there are no feasibility cuts in our problem. To generate the optimality cuts we use the following relaxation, $RQ_s(\hat{\boldsymbol{w}})$, of $Q_s(\hat{\boldsymbol{w}})$:

$$
\begin{aligned}
\min \ & \sum_{\{i,j\} \in E} \bar{c}_{ij} z_{ij}^s + \sum_{\{i,j\} \in M} (c_{ij}^s - \bar{c}_{ij}) v_{ij}^s \\
\text{s.t.} \ & \sum_{i \in N} a_{ik} y_i^s \geq 1 && \forall k \in K, && (\alpha_k^s) && (4.18) \\
& y_i^s - z_{ij}^s \geq 0 && \forall \{i,j\} \in E, && (\gamma_{ij}^s) && (4.19) \\
& y_j^s - z_{ij}^s \geq 0 && \forall \{i,j\} \in E, && (\Gamma_{ij}^s) && (4.20) \\
& z_{ij}^s - y_i^s - y_j^s \geq -1 && \forall \{i,j\} \in E, && (\beta_{ij}^s) && (4.21) \\
& z_{ij}^s - v_{ij}^s \geq 0 && \forall \{i,j\} \in M, && (\epsilon_{ij}^s) && (4.22) \\
& -v_{ij}^s \geq -\hat{w}_{ij} && \forall \{i,j\} \in M, && (\lambda_{ij}^s) && (4.23) \\
& v_{ij}^s - z_{ij}^s \geq \hat{w}_{ij} - 1 && \forall \{i,j\} \in M, && (\Pi_{ij}^s) && (4.24) \\
& \sum_{i<j} a_{ik} z_{ij}^s + \sum_{i>j} a_{ik} z_{ji}^s - y_j \geq 0 && \forall j \in N, k \in K : a_{jk} = 0, && (\delta_{jk}^s) && (4.25) \\
& \sum_{i \neq j} a_{ik} y_i^s + y_j^s - \sum_{i<j} a_{ik} z_{ij}^s - \sum_{i>j} a_{ik} z_{ji}^s \geq 1 && \forall j \in N, k \in K, && (\sigma_{jk}^s) && (4.26) \\
& -y_i^s \geq -1 && \forall i \in N, s \in S, && (\mu_i^s) && (4.27) \\
& (\mathbf{y^s}, \mathbf{z^s}, \mathbf{v^s}) \geq 0
\end{aligned}
$$

Without constraints (4.25)-(4.26), above formulation is the continuous relaxation of $Q_s(\hat{w})$ written in canonical form. Constraints (4.25) and (4.26) are obtained by multiplying the covering constraint (4.18) by y_j^s and $1 - y_j^s$ respectively. Hence these are the inequalities obtained by the reformulation-linearization technique

(RLT) [76] and they strengthen the relaxation.

The dual variables associated with the constraints are indicated alongside. For the multi-cut version of the algorithm let η^s be the first stage decision variable that approximates the second stage cost under scenario s and let $\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\Pi}}, \hat{\boldsymbol{\lambda}}$ and $\hat{\boldsymbol{\mu}}$ be the optimal dual vectors. Then we write the optimality cuts as follows:

$$\eta^s \geq \sum_{k\in K} \hat{\alpha}^s_k + \sum_{i\in N}\sum_{k\in K} \hat{\sigma}^s_{ik} - \sum_{\{i,j\}\in E} \hat{\beta}^s_{ij} + \sum_{\{i,j\}\in M} \big((w_{ij}-1)\hat{\Pi}^s_{ij} - w_{ij}\hat{\lambda}^s_{ij}\big) - \sum_{i\in N} \hat{\mu}_i \quad (4.28)$$

For the single-cut version, we define variable η to approximate the expected cost of the second stage problem and aggregate the cuts as follows:

$$\eta \geq \sum_{s\in S} p_s \Bigg(\sum_{k\in K} \hat{\alpha}^s_k + \sum_{i\in N}\sum_{k\in K} \hat{\sigma}^s_{ik} - \sum_{\{i,j\}\in E} \hat{\beta}^s_{ij} + \sum_{\{i,j\}\in M} \big((w_{ij}-1)\hat{\Pi}^s_{ij} - w_{ij}\hat{\lambda}^s_{ij}\big) - \sum_{i\in N} \hat{\mu}_i \Bigg) \quad (4.29)$$

Besides the classical optimality cuts obtained via duality as done above, there are two other types of cuts utilized in the literature when the master problem is pure binary. The first one is introduced by Laporte and Louveaux [88]. The efficiency of this cut depends on whether we can generate strong lower bounds for the second stage problems and it is built on the assumption that given a first stage solution, the second stage problems are easy to solve. For our problem given the first stage solution $\hat{w}$ we write this optimality cut as follows:

$$\eta^s \geq Q^*_s(\hat{\boldsymbol{w}}) + (L_s - Q^*_s(\hat{\boldsymbol{w}}))(u - \sum_{\{i,j\}\in \hat{H}} w_{ij}) \quad (4.30)$$

In this cut, L_s is a lower bound on the optimal value of second stage problem under scenario s and it is independent from the first stage decision. Also $\hat{H} = \{\{i,j\} \in M : \hat{w}_{ij} = 1\}$. Notice that when $\boldsymbol{w} = \hat{\boldsymbol{w}}$ this cut ensures that $\eta^s \geq Q^*_s(\hat{\boldsymbol{w}})$ and for any other solution its quality strictly depends on the lower bound L_s.

The other well-known cut is no-good cut that is used to eliminate a first stage solution for which the second stage becomes infeasible. In our case, the second stage problem is always feasible. Nevertheless, for our problem given a first stage solution $\hat{\boldsymbol{w}}$ and set $\hat{H}$, no-good cut can be written as below:

$$\sum_{\{i,j\}\in \hat{H}} w_{ij} \leq u - 1 \quad (4.31)$$

We are able to write no-good cuts in this form because the first stage constraint is in equality form, and consequently $|\hat{H}| = u$ for any feasible $\hat{\boldsymbol{w}}$. When the first stage constraint is in inequality form, $\leq$, we write the no-good cut as follows:

$$\sum_{\{i,j\}\in\hat{H}} (1 - w_{ij}) + \sum_{\{i,j\}\notin\hat{H}} w_{ij} \geq 1 \tag{4.32}$$

In standard application of L-shaped algorithms, whether it is multi-cut or single-cut, the relaxation of the second stage problems are solved for each scenario to generate the cuts. In the next section, we show that we can calculate the optimal dual vectors by solving a number of problems much less than the number of scenarios.

4.3.2 Scenario Reduction

From Observation 2, we know that given a first stage solution $\hat{\boldsymbol{w}}$, we can partition the scenarios set S to the sets $\hat{S}_n$ for $n = 1, \ldots, \hat{q}$ such that $c^s_{ij} = c^{s'}_{ij}$ for $\{i,j\} \in \hat{H}$ for any s, s' in $\hat{S}_n$ where $\hat{H} = \big\{\{i,j\} \in M : \hat{w}_{ij} = 1\big\}$. This partition simply groups the scenarios that are indistinguishable and we also know that, for each group, the second stage decisions under the scenarios in that group are identical. This means that, given the first stage solution $\hat{\boldsymbol{w}}$, it is sufficient to solve a single second stage problem for each scenario group. To do that, we define a new scenario set $\hat{S}$ with $\hat{q}$ number of scenarios, $\{\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_{\hat{q}}\}$, each of which corresponds to one of the scenario groups in the partition. We define the cost coefficients $\hat{\boldsymbol{c}}^{\hat{\boldsymbol{s}}_{\boldsymbol{n}}}$ for the new scenario $\hat{s}_n \in \hat{S}$ such that $\hat{c}^{\hat{s}_n}_{ij}$ takes the corresponding scenario value if $\{i,j\}$ is observed in the first stage and takes the mean value otherwise. That is, $\hat{c}^{\hat{s}_n}_{ij} = c^s_{ij}$ for $\{i,j\} \in \hat{H}$ and $s \in \hat{S}_n$, and $\hat{c}^{\hat{s}_n}_{ij} = \bar{c}_{ij}$ for $\{i,j\} \notin \hat{H}$. Let $Q_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$ be the second stage problem for scenario $\hat{s}_n$ for $n \in \{1, \ldots, \hat{q}\}$, given first stage solution $\hat{\boldsymbol{w}}$ and with cost coefficients $\hat{\boldsymbol{c}}$.

Proposition 7 *Given a first stage solution $\hat{\boldsymbol{w}}$, partition $(\hat{S}_1, \ldots, \hat{S}_{\hat{q}})$ and new scenario set $\hat{S}$ constructed as above, the optimal solution of $RQ_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$ (respectively, $Q_{\hat{s}_n}(\hat{\boldsymbol{w}})$) is optimal for $RQ_s(\hat{\boldsymbol{w}})$ (respectively, $Q_s(\hat{\boldsymbol{w}})$) for $s \in \hat{S}_n$, $n \in \{1, \ldots, \hat{q}\}$,*

Proof. For $n \in \{1, \ldots, \hat{q}\}$, $RQ_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$ and $RQ_s(\hat{\boldsymbol{w}})$ for $s \in \hat{S}_n$ have the same constraint set and the terms in the objective are identical except for $\{i,j\} \notin \hat{H}$. Given $\hat{\boldsymbol{w}}$, any feasible second stage solution $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \hat{\boldsymbol{v}})$ of $RQ_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$, satisfies $\hat{v}_{ij}^{\hat{s}_n} = 0$ because $\hat{w}_{ij} = 0$ for $\{i,j\} \notin \hat{H}$ so the solution $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \hat{\boldsymbol{v}})$ has the same objective value in $RQ_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$ and $RQ_s(\hat{\boldsymbol{w}})$. Hence an optimal solution to $RQ_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$ is optimal for $RQ_s(\hat{\boldsymbol{w}})$. □

Although the primal optimal solution of the second stage problem under the new scenarios are optimal for the second stage problems under original scenarios, the dual optimal solutions are required to be adjusted for feasibility, which we explain next.

Proposition 8 *Given a first stage solution $\hat{\boldsymbol{w}}$, let $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\Gamma}}, \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\epsilon}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\Pi}}, \hat{\boldsymbol{\delta}}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\mu}})$ be the optimal solution of $RQ_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$ for $n \in \{1, ..., \hat{q}\}$. This is optimal for the dual of $RQ_s(\hat{\boldsymbol{w}})$ for any $s \in \hat{S}_n$ if we update $\hat{\boldsymbol{\lambda}}$ as follows:*

$$\hat{\lambda}_{ij}^s = \begin{cases} (\hat{\Pi}_{ij}^{\hat{s}_n} - \hat{\epsilon}_{ij}^{\hat{s}_n} - c_{ij}^s + \bar{c}_{ij})^+, & \text{for } \{i,j\} \notin H_r \\ \hat{\lambda}_{ij}^{\hat{s}_n}, & \text{for } \{i,j\} \in H_r \end{cases}$$

Proof. For $n \in \{1, \ldots, \hat{q}\}$, s_n and $s \in \hat{S}_n$, let $DRQ_s(\hat{\boldsymbol{w}})$ and $DRQ_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$ denote the duals of $RQ_s(\hat{\boldsymbol{w}})$ and $RQ_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$ respectively. The mathematical formulation of $DRQ_s(\hat{\boldsymbol{w}})$ is as follows:

$$\max \sum_{k \in K} \alpha_k^s + \sum_{i \in N} \sum_{k \in K} \sigma_{ik}^s - \sum_{\{i,j\} \in E} \beta_{ij}^s + \sum_{\{i,j\} \in M} \left((\hat{w}_{ij} - 1)\Pi_{ij}^s - \hat{w}_{ij}\lambda_{ij}^s \right) - \sum_{i \in N} \mu_i$$

s.t.

$$\sum_{k \in K} a_{ik}\alpha_k^s - \sum_{k \in K: a_{ik} = 0} \delta_{ik}^s + \sum_{k \in K} \sigma_{ik} + \sum_{j < i} \sum_{k \in K} \sigma_{jk} + \sum_{j > i} (\gamma_{ij}^s - \beta_{ij}^s) + \sum_{j < i} (\Gamma_{ij}^s - \beta_{ji}^s) - \mu_i \leq 0 \qquad \forall i \in N, \tag{4.33}$$

$$\sum_{k \in K: a_{ik} = 0} a_{jk}\delta_{ik}^s + \sum_{k \in K: a_{jk} = 0} a_{ik}\delta_{jk}^s - a_{jk}\sigma_{ik} - a_{ik}\sigma_{jk} + \beta_{ij}^s - \gamma_{ij}^s - \Gamma_{ij}^s + \epsilon_{ij}^s - \Pi_{ij}^s \leq \bar{c}_{ij} \qquad \forall \{i,j\} \in E, \tag{4.34}$$

$$-\epsilon_{ij}^s - \lambda_{ij}^s + \Pi_{ij}^s \leq c_{ij}^s - \bar{c}_{ij} \qquad \forall \{i,j\} \in M, \tag{4.35}$$

$$(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\Gamma}}, \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\epsilon}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\Pi}}, \hat{\boldsymbol{\delta}}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\mu}}) \geq 0$$

The formulations of $DRQ_s(\hat{\boldsymbol{w}})$ and $DRQ_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$ are identical except for the constraints (4.35) for $\{i,j\} \notin \hat{H}$ because that is when $c^s_{ij} \neq c^{\hat{s}_n}_{ij}$. Hence any solution to $DRQ_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$ satisfies the constraints of $DRQ_s(\hat{\boldsymbol{w}})$ except for these constraints. By updating $\hat{\lambda}^s_{ij} = (\bar{\Pi}^{\hat{s}_n}_{ij} - \bar{\epsilon}^{\hat{s}_n}_{ij} - c^s_{ij} + \bar{c}_{ij})^+$ for $\{i,j\} \notin \hat{H}$ we satisfy (4.35) hence the solution becomes feasible for $DRQ_s(\hat{\boldsymbol{w}})$. The corresponding objective values of these solutions are equal since the cost coefficient of $\hat{\lambda}^s_{ij}$ is zero for $\{i,j\} \notin \hat{H}$, hence the updated solution is optimal for $DRQ_s(\hat{\boldsymbol{w}})$. $\square$

With these two propositions, we have shown that, given a first stage solution $\hat{\boldsymbol{w}}$, it is sufficient to solve $\hat{q}$ number of linear programming problems in the second stage to generate optimality cuts and again it is sufficient to solve $\hat{q}$ number of integer programming problems in the second stage to obtain optimal second stage solution given $\hat{\boldsymbol{w}}$.

4.3.3 The Algorithm

We use the lazy-constraint callback feature of the solver and generate and add optimality cuts whenever an integral first stage solution is found. Let $\hat{\boldsymbol{w}}$ be the integral first stage solution found at the current node of the branch-and-bound tree. We create the new scenarios set $\hat{S} = \{\hat{s}_1, \ldots, \hat{s}_{\hat{q}}\}$ as described before Proposition 7. We solve the dual of the stronger relaxation of second stage problems, $DRQ_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$, for $n = 1, \ldots, \hat{q}$ and update the solution as described in Proposition 8. Here, we record whether the corresponding primal solution, the solution of $RQ_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$, is integral or not. It suffices to check integrality of $\hat{\boldsymbol{y}}^{s_n}$ because integrality of $\hat{\boldsymbol{y}}^{s_n}$ implies the integrality of $\hat{\boldsymbol{z}}^{s_n}$ and $\hat{\boldsymbol{v}}^{s_n}$. We generate and add the optimality cut (4.29) if it cuts the current solution. If it does not, then we solve the integer second stage problems $\hat{Q}_{\hat{s}_n}(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$ for $\hat{s}_n$ such that $\hat{\boldsymbol{y}}^{\boldsymbol{s_n}}$ is not integral. Let $(\hat{\boldsymbol{y}}^*, \hat{\boldsymbol{v}}^*, \hat{\boldsymbol{z}}^*)$ be the optimal integral solution for the second stage and $\hat{Q}^*(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$ be its objective value. $\hat{Q}^*(\hat{\boldsymbol{w}}, \hat{\boldsymbol{c}})$ is an upper bound for the original problem so we pass this information to the solver using heuristic callback and add no-good cut (4.31) to eliminate first stage solution $\hat{\boldsymbol{w}}$.

In our algorithm, we use the aggregated cuts (4.29) instead of scenario-dependent cuts (4.28). Theoretically, with the multi-cut implementation, we could get better approximations of the second stage problems. It requires having a decision variable η^s for each scenario in the master problem and the number of cuts added at each iteration can be as large as the number of scenarios. Consequently, the master problem becomes too large too quickly and the time required to solve it increases substantially. Therefore, we add these cuts as a single aggregated cut of form (4.29). Furthermore, since we do not want to define η^s for each scenario, we do not use the cuts of the form 4.30, which requires the optimal objective value of the integer second stage problem and a lower bound for this. Of course, these cuts can be aggregated and added as a single cut as well but we choose not to because to have strong cuts of this form we need a strong lower bound that requires computational time.

We use a linear relaxation of the second stage problems, which are strengthened by the inequalities derived by the reformulation-linearization technique. In literature, there are studies where the relaxation is strengthened iteratively with additional cuts such as in [89] and [90] with Gomory cuts and in [91] and [92] with disjunctive cuts. In our algorithm, we use a strong relaxation from the beginning. Moreover, the way we solve integer second stage problems is similar to the alternating strategy of Angulo et al. [93] because we only solve the integer problems if the duality-based optimality cuts (4.29) does not cut the current solution. The difference is that because we solve a strong relaxation, it gives integral solutions frequently and we usually do not need to solve the integer problems. In Section 4.4.3, presenting computational results for different variants of the algorithm, we explain why we propose this version.

4.4 Experiments

In this section, we first explain the instance generation and pre-process we apply. Then we compare the computational efficiency of formulations CF and IF,

and show their limitations. We present the computational experiments for various versions of the branch-and-cut algorithm and explain why we decide on the form explained in the previous section. All formulations and 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 are wall-clock times in seconds.

4.4.1 Data Generation and Pre-process

To test our formulations and the algorithms, we have generated random instances. $|N|$ random points are generated in the two-dimensional coordinate system, and the mean value of the communication costs, $\bar{c}_{ij}$'s, are taken as the Euclidean distance between these points plus a random term. Having $|K|$ skills, we randomly assign skills to people ensuring feasibility by checking the existence of a capable person per skill. $|M|$ edges are randomly selected as the uncertain ones. Each of these $|M|$ edges has two possible costs. The low or optimistic cost of $\{i, j\} \in M$ is calculated by $c^l_{ij} = d\bar{c}_{ij}$ with probability ρ_{ij} where d and ρ_{ij} for $\{i, j\} \in M$ take random values from sets $\{0.3, 0.4, \ldots, 0.7\}$ and $\{0.3, 0.4, \ldots, 0.9\}$ respectively. The high or pessimistic cost of edge $\{i, j\}$, c^h_{ij}, is calculated accordingly. Scenario set is the cross product of all possible realization of edge costs and probability of a scenario is the product of edge probabilities.

In Section 4.1, through a small example, we explained that learning the true cost value of a random edge can lead to a team with less communication cost. In that example, by observing either of the random edges we are able to build a better team than the optimal solution of the mean value problem. However, observing a random edge might not always lead to a better decision. In other words, the information we obtain by observing an edge might be insignificant. By detecting such edges and ignoring their randomness we can substantially decrease the number of scenarios because having $|M|$ random edges each having 2 possible cost values we have $2^{|M|}$ scenarios. Moreover, identifying such edges also increases the chance of solving more interesting instances in terms of the value of learning,

which is the expected improvement we get from the observations. For example, if $|M| = 10$ in an instance but most of the random edges are irrelevant, then it is less likely for the value of learning to be high as opposed to an instance where all 10 random edges are relevant.

There can be various ways to identify irrelevant edges. We use the following approach utilizing the fact that the deterministic problem can be solved very efficiently. For each $\{i, j\} \in M$, we find the minimum cost team having edge $\{i, j\}$ with respect to the most optimistic scenario, i.e., by solving a deterministic problem with c^l_{mn} for all $\{m, n\} \in M$. Let T^l_{ij} be the cost of this team. We also solve a deterministic problem with respect to the most pessimistic case and let T^h be its objective value. For $\{i, j\} \in M$ if $T^l_{ij} > T^h$ we remove $\{i, j\}$ from M because learning this edge is not useful since it is never going to be part of a capable team under any scenario. We note that this process does not allow us to detect all irrelevant edges. A naive approach to detect all could be finding the best team under each scenario and conclude that an edge is irrelevant for the stochastic problem if it does not show up in any of the scenarios. Of course, considering the number of scenarios, this approach can take significant time as it amounts to solving $2^{|M|}$ deterministic problems.

4.4.2 Comparision of *CF* and *IF*

We have generated small-sized instances and solved these instances with formulations IF and CF using CPLEX 12.7 as the solver. In Table 4.2 we present the instance information and the corresponding solution times. $relM$ indicates the number of random edges left in set M after the pre-process. For example, in the first instance, this number is reduced to 8 from 14. Each instance is solved twice by taking $u = 4$ and $u = 6$. We present the solution times under the columns IF and CF, and dash means the solver could not find an optimal solution within 1 hour. Recall that, value of learning (VE) is the difference between the objectives of the Expected Value Problem (EVP) and the two-stage stochastic problem (SP). We define VE$^{\%}$= 100*(EVP-SP)/EVP.

Table 4.2: Comparison of IF and CF

					u=4			u=6		
no	N	K	M	relM	IF	CF	$VE^{\%}$	IF	CF	$VE^{\%}$
1	10	8	14	8	23.09	6.64	0.73	12.35	2.22	0.9
2	10	8	14	9	50.36	31.37	6.24	61.7	46.02	6.47
3	10	8	14	10	105.92	96.6	5.08	104.19	31.01	5.35
4	10	8	14	11	30.89	5.46	0	27.54	5.69	0
5	10	8	14	12	603.62	499.21	4.24	696.88	211.76	4.28
6	10	8	14	12	64.068	10.406	0	63.069	8.458	0
7	12	8	14	8	59.81	42.35	0	71.08	19.15	0.06
8	12	8	14	9	1196.96	2704.76	6.17	1407.96	402.71	6.36
9	12	8	14	9	-	-	$\geq$5.95	-	-	$\geq$5.66
10	12	8	14	10	2322.93	2249.29	5.75	2111.43	476.14	5.78
11	12	8	14	10	172.18	169.98	3.71	159.2	62.17	3.71
12	12	8	14	11	421.77	1206.62	5.08	235.935	194.54	5.40
13	12	8	14	11	2293.77	2865.97	13.49	2348.92	1462.37	14.18
14	15	8	14	9	77.56	232.6	7.11	77.69	57.07	7.16
15	15	8	14	9	-	1999.46	2.06	-	336.57	2.37
16	15	8	10	10	51.66	123.59	0.44	74.71	21.13	0.44
17	15	8	14	10	-	-	$\geq$1.51	-	2254.82	2.11
18	15	8	14	10	504.94	1780.93	6.69	466.07	499.15	6.69

When $u = 4$, in half of the instances, the solutions times of IF and CF do not have much difference. For the cases where the difference is large, we point it out by highlighting the smaller times. For $u = 4$, the performance of IF is slightly better, although there are instances that CF solved faster. When $u = 6$, the solution times of CF is almost always better than those of IF. For this case, we highlighted the ones where the difference is substantial.

We also see that, for the majority of the instances, $\text{VE}^{\%}$ is positive, meaning that the cost of the team is improved by observing u edges or relations in the first stage. It is not surprising for $\text{VE}^{\%}$ to be equal to or close to zero for small instances and we have more positive $\text{VE}^{\%}$ values as the instances get larger. Furthermore, when we compare instances with the same size, such as instances 5 and 6 or 12 and 13, we see that the ones with the higher $\text{VE}^{\%}$ took more time to solve. In other words, it seems that the instances where the value of learning is higher is computationally more difficult.

Table 4.3: Comparison of continuous relaxations of IF and CF

		IF			CF		
N	relM	time	lp-time	lp-gap	time	lp-time	lp-gap
10	8	57.77	0.43	38.75	13.98	0.54	36.63
10	9	4.074	1.39	1.52	5.98	0.723	0.97
10	10	41.35	5.09	4.11	31.27	1.99	2.19
10	11	12.63	1.77	0.00	3.856	1.9	0.00
10	12	216.52	42.42	11.46	292.81	5.453	8.70
10	13	391.94	68.39	2.18	189.32	15.59	1.95
12	6	1.595	0.139	23.40	1.151	0.17	18.37
12	7	3.63	0.183	8.12	5.093	0.28	5.91
12	8	92.12	0.69	45.69	56.741	1.01	43.18
12	9	23.27	2.93	7.10	12.138	1.144	4.04
12	10	987.23	4.534	43.59	961.93	2.28	37.80
12	11	221.13	47.19	2.48	433.09	5.74	1.15
12	12	218.91	72.13	0.21	73.85	18.37	0.21
14	8	31.28	0.468	30.74	27.73	1.203	28.48
14	9	19.037	1.29	10.27	34.97	1.34	8.14
14	10	79.02	7.41	3.94-	46.74	4.94	2.02
16	8	52.07	0.638	45.03	88.60	2.35	41.90
16	9	-	2.417	100.00	-	146.535	100.00

We also compare formulations IF and CF with respect to their continuous relaxations. In Table 4.3 we present the solution times of these relaxations and their gaps calculated by 100*(opt-lp)/opt where opt refers to the optimal objective value of the integer problem and lp is the objective value of the continuous relaxation. In each of these instances, the number of skills, $|K|$ is 8 and the limit on the observations, u, is 4. As seen under the column *lp-time*, the continuous relaxation of CF gives a slightly better bound than that of IF in all of the instances. In the last instance, neither CF nor IF reaches an optimal solution within one hour and interestingly for these instances the objective value of the relaxations are both zero.

These results show that the size of the instances we can solve using the formulation IF and CF is limited. Next, we present experiments with the branch-and-cut algorithm.

4.4.3 Experiments on Different Versions of the Branch-and-Cut Algorithm

Algorithms based on Benders' decomposition are open to various modifications. One can have many different versions by changing the type of cut added to the master, when to add which cut, the type of relaxation solved, etc. In this section, we present the results of the experiments we made in order to compare performances of different versions of our branch-and-cut algorithm and to reach the best possible variant of it.

As we explained in Section 4.3.3, we add the optimality cuts that are based on linear programming duality as a single cut. It is more efficient than the multi-cut version for our problem since the number of scenarios is high. We have created different versions of the algorithm in terms of which relaxation of the second stage problem is used and in terms of when no-good cut (4.31) is added. There are three alternatives we tested in terms of the relaxation. These are full-rlt case (F) that we solve the dual of formulation RQ_s, half-rlt case (H) where we solve dual

Table 4.4: Comparison of multi-cut and single-cut versions

		F-o					*F-o-multi*				
N	relM	iter	node	time	dualcut	nogood	iter	node	time	dualcut	nogood
10	9	75	126	4.67	69	6	49	86	3.33	5364	13
10	11	202	403	8.1	201	1	62	157	62.39	39476	11
10	13	250	517	12.5	246	4	89	298	3413.06	252084	10

of formulation RQ_s except constraints (4.26) and lastly no-rlt case (N) where we solve the dual of continuous relaxation of the second stage problem. In terms of the no-good cut we experimented on two options. In the first one, (e), no-good cut is generated and added at each iteration. In the second case, (o), no-good cut is only added if the current solution does not violate the current optimality cut (4.29).

Before presenting computational experiments of the single-cut versions, we present some results of multi-cut and single-cut versions over few instances to show the superiority of the single-cut case in terms of the solution time. In Table 4.4, we have three instances solved by *F-o*, which is the single-cut version of the algorithm where we solve the dual of formulation RQ_s and add the no-good cut only if the optimality cut of the form (4.29) is not violated. We also solve these instances with the multi-cut version, *F-o-multi*, where no-good cuts are added only if optimality cuts of the form (4.28) are not violated. As seen under columns named, *iter* and *nodes*, the multi-cut version terminates with less iterations and nodes. However, the number of cuts added to the master problem is much higher in multi-cut case as seen under the column name *dualcut*. The master problem grows faster with this high number of cuts in the multi-cut version. For instances with small number of random parameters, this does not increase the solution time, as we see in the first row. But, as the number of random parameters and consequently the number of scenarios increase, the solution time becomes much longer in multi-cut case, as we see in the last two rows. Hence, the single-cut version outperforms the multi-cut in terms of the solution time.

In Table 4.5 we present the solution times of few instances using formulations and different versions of the algorithm. We would like to remind how we name

these different versions: the first letter indicates the type of dual formulation used and the second letter indicates when no-good cut is added, e.g. *H-e* refers to the half-rlt version where no-good cut is added at each iteration.

Table 4.5: Comparison of formulations and different versions of the algorithm

no	N	relM	u	*IF*	*CF*	*F-e*	*H-e*	*N-e*	*F-o*	*H-o*	*N-o*	VE%
1	12	9	4	7.52	3.31	1.35	1.15	0.68	1.62	1.39	0.96	3.9
			6	5.6	2.31	3.93	3.19	3.28	3.83	3.51	2.99	3.9
2	12	9	4	60.05	51.18	4.09	3.33	32.68	4.18	3.78	36.23	2.03
			6	61.97	25.21	12.38	10.94	78.93	14.27	12.74	84.53	2.31
3	12	10	4	163.50	301.80	4.10	3.68	44.31	3.56	3.16	53.02	5.36
			6	167.06	111.33	20.08	14.61	185.80	20.99	16.60	199.79	5.36
4	12	10	4	101.42	192.61	4.69	9.87	15.71	4.95	7.09	15.44	0.84
			6	117.21	93.75	24.54	34.26	92.66	22.77	31.64	87.48	0.84
5	12	11	4	2890.11	3182.81	8.78	6.75	78.6	8.5	6.55	86.54	4.62
			6	2746.49	1056.97	51.51	37.82	463.87	52.5	41.68	544.18	4.9
6	12	11	4	871.93	-	2.12	1.40	71.08	1.76	1.47	66.74	5.52
			6	907.12	601.19	11.41	12.25	369.86	12.26	13.84	367.99	5.52

The smallest solution time for each instance is highlighted in Table 4.5. There is not much difference in the performances of the solution methods for "easy" instances such as the first instance where all solution times are few seconds. For the rest, full-rlt (F) and half-rlt (H) versions of the algorithm perform the best. This table clearly indicates the superiority of the algorithm to the formulations. For example, instance 6 with $u = 4$ could not be solved with CF within an hour while some versions of the algorithm solved it in a second. However, this table does not give sufficient information to favor a specific version of the algorithm. According to this data, we could only say that no-rlt versions (N), where we use the continuous relaxation of the second stage problem without strengthening it, perform poorly compared to the other versions.

In Table 4.6, we present more computational details about the algorithms over one instance. The details belong to instance 4 with $u = 4$ in Table 4.5. In this table, IP time refers to the total time spent to solve the integer second stage problems. Other row names are self-explanatory. We can see that the solution times of no-rlt versions are higher due to the time spent in solving the integer second stage problems. This time is zero in full-rlt versions, which means that

Table 4.6: Computational details of algorithms for one instance

	F-e	*H-e*	*N-e*	*F-o*	*H-o*	*N-o*
total time	4.694	9.87	15.71	4.95	7.09	15.44
# iteration	97	176	116	107	176	145
# single cut	97	176	64	102	171	59
# no-good	97	176	116	5	5	86
# incumbent update	4	3	2	2	4	3
IP time	0	2.45	11.81	0	0	10.64
# nodes	173	266	103	165	316	96

the relaxation gave integral solutions. full-rlt versions also have less number of iterations.

In the light of these results, we remove no-rlt versions from the experiments, which leaves us with *F-e*, *H-e*, *F-o*, and *H-o* versions. Recall that with preprocessing we identify some irrelevant edges and ignore their randomness because they are not part of the optimal solution under any scenario. With the same reasoning we can actually remove these edges from the problem completely by adding $z^s_{ij} = 0$ to the integer second stage problem and to its relaxation for any irrelevant edge $\{i, j\}$ under scenario s. Therefore, we created 4 new versions of the algorithm where we eliminate these edges and we add the abbreviation *eli* to denote these versions. In Table 4.7 we compare these versions using larger instances. Again we use a one-hour time limit so dash means the corresponding versions of the algorithm could not solve the problem in one hour. The minimum solution time for each instance is highlighted and in the last row, we share the average solution times over these instances. For the cases where the time limit is exceeded, we use 3600 seconds in the average calculation.

According to these results, the *eli*-versions where we eliminate the irrelevant edges are not always more efficient than their counterparts. Approximately for half of the instances, the extra constraints added to the integer second stage problem and the relaxation increase the solution times. In general, from these results, we see that almost for all versions, there is an instance on which it performed better than the others, therefore, it is difficult to single out one version at first glance. However, according to the average solution times, the version *F-o* gives

Table 4.7: Comparison of different versions of the algorithm with larger instances

N	K	relM	u	*F-e*	*F-e-eli*	*H-e*	*H-e-eli*	*F-o*	*F-o-eli*	*H-o*	*H-o-eli*	VE%
20	6	11	6	258.38	156.60	1048.79	1036.78	162.77	322.39	1047.09	1103.15	13.88
20	6	12	6	41.996	44.746	67.85	59.56	39.27	38.236	75.45	49.35	24.60
20	6	13	6	106.91	133.78	548.20	766.185	110.62	137.62	451.17	490.83	16.45
20	6	14	6	72.94	118.60	1015.08	1012.22	70.81	119.95	986.07	1027.53	6.59
30	8	11	6	372.18	306.27	938.46	555.76	435.24	347.78	885.54	601.26	7.28
30	8	12	6	104.12	111.39	167.24	134.18	114.66	115.60	155.55	126.15	12.82
30	8	13	6	482.31	279.00	219.62	209.96	510.10	308.71	240.22	215.21	15.88
30	8	14	6	1553.88	1358.16	1414.50	1304.40	871.02	777.37	551.79	483.55	4.93
40	8	11	6	70.68	85.33	74.77	75.99	77.68	80.33	77.65	70.33	0.32
40	8	12	6	538.68	586.04	1130.35	847.40	523.70	639.17	869.32	768.01	17.5
40	8	13	6	94.14	133.48	132.33	129.94	99.93	117.20	187.67	135.17	9.16
40	8	14	6	416.83	637.58	804.96	691.59	391.99	608.73	670.20	621.14	4.55
40	10	10	6	365.35	838.35	665.91	516.55	435.19	1108.84	782.37	593.60	5.49
40	10	11	6	-	-	-	-	2909.12	2858.22	-	-	7.4
40	10	12	6	349.18	490.03	537.53	336.75	350.80	465.90	329.50	330.85	3.16
40	10	13	6	-	-	-	-	-	-	-	-	≥1.24
40	10	14	6	-	-	-	-	-	-	-	-	≥7.31
50	8	11	6	210.35	605.58	561.53	454.32	220.52	652.89	665.09	482.09	7.36
avg				951.87	946.70	1199.06	1116.32	835.06	953.88	1140.56	1023.02	

Table 4.8: Results on *F-o* version of the algorithm with two-hour time limit

N	K	M	relM	u	time	iter
40	8	50	11	4	151.26	271
				5	902.39	419
				6	1546.47	180
40	8	50	12	4	135.82	167
				5	1263.42	362
				6	4423.49	309
40	8	50	13	4	99.95	96
				5	805.97	198
				6	3765.62	228
40	8	50	14	4	409.32	428
				5	4529.31	1220
				6	-	477
50	8	50	13	4	225.99	136
				5	2553.82	386
				6	-	272
50	8	50	14	4	262.06	130
				5	2502.12	331
				6	-	218
50	8	50	15	4	176.04	59
				5	3344.14	306
				6	-	164

the best performance, and that is why we described this version in Section 4.3.3 as the solution method for the problem.

In Table 4.8, we present solution times of instances solved by the F-o version with a two-hour time limit. As u gets larger, the number of problems solved in the second stage gets larger exponentially, and consequently, each iteration takes more time. Among these instances, the average time of an iteration is 1.4 seconds for $u = 4$, 5.5 for $u = 5$ and 22.6 for $u = 6$. For the two-stage stochastic team formation problem, it is acceptable to assume that the first stage allows only a couple of observations. If one desires to solve an instance of this problem or a similar-structured two-stage problem where the number of resolutions after the first stage is high, increasing computation capacity and using parallel computing techniques for the second stage problems could decrease the solution time.

4.5 Conclusion

In this study, we introduced a two-stage stochastic team formation problem where the uncertainty in the communication costs are endogenous and their true values reveal in the second stage if they are selected to be observed in the first stage. We gave two mathematical formulations for this problem and show their equivalence in terms of optimality. A Benders' decomposition-based branch-and-cut algorithm was developed based on the second formulation. The algorithm utilizes a strong linear relaxation of the second stage problem, which not only gives strong optimality cuts but also decreases the burden of solving the integer problems by providing integral solution frequently. The algorithm is able to solve the problems with thousands of scenarios because at each iteration it creates and works on a smaller scenario set, which is possible due to the decision-dependent structure. Both in terms of modeling and solution methodology, the work proposes an alternative to the relevant literature where Lagrangian relaxation methods are applied over the formulations with conditional non-anticipativity constraints.

The work can be extended by requiring direct assignment to tasks, which changes the structure of the second stage problems and optimality cuts. In this case, one can also consider uncertainty in people's capability and availability, which can be exogenous or endogenous. Moreover, an interesting extension can be a multi-stage version of the problem where each stage requires a capable team and the team is updated according to the observations from the previous stages. In the next chapter, we study a different type of team formation problem in a multi-stage setting.

Chapter 5

Multi-Stage Stochastic Project Team Formation

In this chapter, we study a multi-stage stochastic team formation problem where each stage corresponds to a different project which requires a team. In this problem, we consider uncertainty in people's performances and minimize the expected cost of hiring and outsourcing resources for the whole horizon. We assume that once a person is assigned to a task of a project, we observe their true performance on this task so the uncertainty resolves. Hence, the type of uncertainty here is again decision-dependent or endogenous.

For the small-sized instances of this problem, where the number of random parameters is less than 7, the deterministic equivalent formulation can be directly solved with a commercial solver. For instances with higher random parameters, exact solution methods can be very time-consuming or even impossible to solve due to the size of the formulation. Therefore, we developed a decomposition-based branch-and-bound algorithm which solves a relaxation of the problem and provides tight bounds for the optimal value.

In Section 5.1 we state the problem and provide a mathematical formulation.

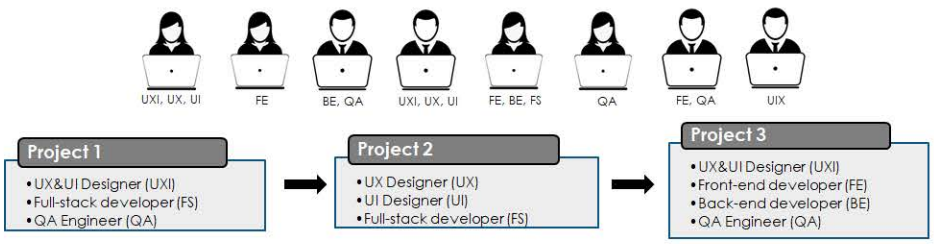


Figure 5.1: An illustrative example with three stages/projects

In Section 5.2 we explain the calculation of value of stochastic solution for multi-stage stochastic problems with endogenous uncertainty. In Section 5.3 we describe our solution methodology pointing out its difference from the existing methods. In Section 5.4 we present computational experiments where we compare the existing methods with ours. We conclude in Section 5.5. The work in this chapter is based on [94].

5.1 Problem Definition and Formulation

To draw a clear picture of the problem, before giving the formal definition, we first explain it through an example. In this example, a project manager wants to recruit freelancers to conduct three software-development projects which will be done consecutively. Job titles required for each project and the available people with their capabilities are depicted in Figure 5.1. The projects are assumed to be independent of, but similar to each other, and consequently, they require common tasks. The manager aims to form a team for each project so that the projects are completed on time with minimum cost. Different than the problems investigated in the previous chapters, here, the cost is defined by the monetary payments made to the team members.

Although all available people are assumed to possess an acceptable level of knowledge in their area of expertise, their capabilities are not identical. Therefore, the time required to complete a task might differ from person to person. We assume that for some person-task pairs the time required for a person to finish a

task is not known precisely, but there are possible values it can take. Hence, in this problem, the source of stochasticity is the uncertainty in the performances of people. We assume that the uncertainty in a person-task pair resolves, and the true performance is observed once the person is assigned to the task. Hence the uncertainty is decision-dependent. In the example, if the performance of the person assigned to Q&A Engineering in the first project is random, the true performance of this person on this job will be observed during this project. The decision whether to select the same person for the job on the third project will be made considering this observation. In this problem, the performances of people are differentiated by the time they require to complete a task. When this time exceeds the planned project duration, an external source is used to speed up the process. Hence, if the assigned Q&A Engineer in the first project cannot complete the job on time, the manager must pay for an external resource to speed up the process and respect the project duration.

Here we would like to note that the differences in performances of people can be measured and represented in the problem in other ways instead of task duration. For instance, if the focus is on the quality of the projects, it might be more appropriate to measure one's performance by the quality of their work, and in this case, an external source is needed when the quality of the completed task is not at an acceptable level.

There are three types of costs in our problem. There is a fixed cost of hiring people, the payment made to the person for the specific job assigned to them, and lastly the outsourcing cost. Naturally, outsourcing is assumed to be more costly than regular hiring and we assume that such a source is always available. The fixed cost is only incurred once, even if the person works on more than one project.

Let T be the set of stages/projects. Let K_t be the set of tasks to be completed for the project t and K be the set of tasks in general. I is the set of available people for regular hiring and $I_k \subset I$ is the set of people who can be assigned to task k. In any project, each task must be assigned to a single person and a person cannot be responsible for more than one task. S is the set of scenarios and d_{ik}^s

is the time required by person i to complete task k under scenario s which has probability p_s. Δ_t is the planned duration of the project t and all tasks should be completed within this time. When the assigned person is not able to complete a task within the planned duration, we outsource another person so that the task is complete on time. o_k is the cost of the outsourcing to decrease the completion time of task k one unit. This actually corresponds to what's called crashing cost in project management literature. f_i is the fixed cost of hiring person i. c_{ik} is the payment person i demands to complete task k for a project which is called assignment cost in short. The objective is to minimize the total expected cost of the projects.

Before giving the mathematical formulation for this multi-stage stochastic problem, we will first formulate its deterministic version and explain its relation to well-known operations research problems. Let binary variable y_i be one if person i is hired for any project, and zero otherwise. x_{ikt} is equal to one if person i is assigned to task k in project t. z_{kt} is the amount of time task k is crashed in project t, that is, amount of time the completion time of the task is reduced to respect the project duration Δ_t. d_{ik} is the time person i requires to finish task k. Then we formulate the deterministic problem as follows:

$$\min \sum_{i\in I} f_i y_i + \sum_{t\in T}\sum_{k\in K_t}\sum_{i\in I} c_{ik}x_{ikt} + \sum_{t\in T}\sum_{k\in K_t} o_k z_{kt} \tag{5.1}$$

$$\text{s.t} \sum_{k\in K_t} x_{ikt} \le y_i \qquad \forall i \in I, t \in T \tag{5.2}$$

$$\sum_{i\in I_k} x_{ik} = 1 \qquad \forall k \in K_t, t \in T \tag{5.3}$$

$$\sum_{i\in I} d_{ik}x_{ikt} \le \Delta_t + z_{kt} \qquad \forall k \in K_t, t \in T \tag{5.4}$$

$$x_{ikt} \in \{0,1\} \qquad \forall k \in K_t, i \in I, t \in T \tag{5.5}$$

$$0 \le z_{kt} \le \Delta_t \qquad \forall k \in K_t, t \in T \tag{5.6}$$

$$y_i \in \{0,1\} \qquad \forall i \in I. \tag{5.7}$$

Constraints (5.2) prevent allocation of a person to multiple tasks for each project and they guarantee that if a person is used in any project their fixed cost is included in the objective function. Constraints (5.3) ensure that in every project a

person is assigned to each required task. Constraints (5.4) guarantee that each project is finished within the planned duration. The objective (5.1) is the summation of fixed cost, assignment cost and outsourcing cost. This formulation is very close to those of several well-studied problems in the literature. Without the outsourcing option, i.e., removing z-variables, the single stage/project version of the deterministic problem has the same formulation as the fixed-charge assigning users to sources problem studied by Neebe and Rao [95] and the capacitated facility location problem with single sourcing studied by Holmberg et al. [96]. The formulation is also similar to the one given for the multi-period single-sourcing problem by Freling et al. [97]. In the multi-period single-sourcing problem there is no fixed cost and the stages are connected due to inventory decisions whereas in our multi-stage deterministic problem the stages are connected due to having fixed cost and y-variables. All these problems are actually extensions of generalized assignment problem (GAP) [98] which has been extensively studied in the literature. The optimal solution is obtained usually via branch-and-bound and branch-and-price algorithms for this NP-hard problem [99]. The single stage deterministic version of our problem is a special case of GAP where the available capacity of each resource is one.

In broad terms, GAP deals with assignment of agents to tasks where limited resources are available to the agents. Many different problems can be formulated as a GAP including facility location, inventory allocation and scheduling problems. In our case we assign people to tasks for the projects which are performed consecutively and there is uncertainty in the performances of the people. As we study this problem under the team formation title, we refer to agents as people. But the agents could be machines, computers or facilities in a problem where there is uncertainty in their performance such as in the time required by the computer or amount of resources used by the machine.

Next we give and explain the formulation of the stochastic multi-stage team formation problem. We assume that the planned projects are similar to each other and it is very likely that some of the tasks they require are common. Because of this similarity assumption, we do not need to distinguish parameters c_{ik} and d_{ik}^s for each project by adding an index t. We define binary variable y_i^s to be one

if person i is hired for any project under scenario s, and zero otherwise. x^s_{ikt} is equal to one if person i is assigned to task k in project t under scenario s. z^s_{kt} is the amount of time task k is crashed in project t, that is, amount of time the completion time of the task is reduced. We define $D_{ss'}$ as the set of person-task pairs that distinguishes scenarios s and s', i.e., $D_{ss'} = \{\{i,k\} : d^s_{ik} \neq d^{s'}_{ik}\}$. So person-task $\{i,k\}$ is in set $D_{ss'}$ if the time required by i to complete task k is different under scenarios s and s'. $\mathcal{L}$ is the set of scenario pairs which have only one different parameter, that is, $\mathcal{L} = \{\{s,s'\} : s,s' \in S, s < s', |D_{ss'}| = 1\}$.

$$\min \sum_{s\in S}\sum_{i\in I} p_s f_i y^s_i + \sum_{s\in S}\sum_{t\in T}\sum_{k\in K_t} p_s\big(o_k z^s_{kt} + \sum_{i\in I_k} c_{ik} x^s_{ikt}\big) \tag{5.8}$$

$$\text{s.t} \sum_{i\in I_k} x^s_{ikt} = 1 \qquad \forall s\in S, t\in T, k\in K_t \tag{5.9}$$

$$\sum_{k\in K_t} x^s_{ikt} \leq y^s_i \qquad \forall s\in S, t\in T, i\in I, \tag{5.10}$$

$$\sum_{i\in I_k} d^s_{ik} x^s_{ikt} \leq \Delta_t + z^s_{kt} \qquad \forall s\in S, t\in T, k\in K_t, \tag{5.11}$$

$$x^{s-1}_{ik1} = x^s_{ik1} \qquad \forall k\in K_1, i\in I_k, s\in S\setminus\{1\}, \tag{5.12}$$

$$x^s_{ikt} - x^{s'}_{ikt} \leq \sum_{\tau=1}^{t-1} x^s_{i'k'\tau} \qquad \forall t\in T, k\in K_t, i\in I_k, (s,s')\in\mathcal{L}, \{i',k'\} = D_{ss'}, \tag{5.13}$$

$$x^{s'}_{ikt} - x^s_{ikt} \leq \sum_{\tau=1}^{t-1} x^s_{i'k'\tau} \qquad \forall t\in T, k\in K_t, i\in I_k, (s,s')\in\mathcal{L}, \{i',k'\} = D_{ss'}, \tag{5.14}$$

$$y^s_i \in \{0,1\} \qquad foralli\in I, s\in S, \tag{5.15}$$

$$x^s_{ikt} \in \{0,1\} \qquad \forall t\in T, k\in K_t, i\in I_k, s\in S, \tag{5.16}$$

$$0 \leq z^s_{kt} \leq \Delta_t \qquad \forall t\in T, k\in K_t, s\in S. \tag{5.17}$$

Constraints (5.9) ensure that in every project a person is assigned to a task. Constraints (5.10) prevents allocation of a person to multiple tasks for each project and they guarantee that if a person is used in any project their fixed cost is included in the objective function. Constraints (5.11) ensure that each project is finished within the planned duration. (5.12) are the first stage non-anticipativity constraints which force the first stage decisions under any scenario to be identical. (5.13)-(5.14) are the conditional non-anticipativity constraints.

They ensure the assignment decisions are the same under scenarios s and s' on stage t, if no person-task pair in $D_{ss'}$ is observed in the earlier stages under scenario s.

Notice that, assuming o_k is positive for any $k \in K$, an optimal solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}})$ satisfies $\tilde{z}^s_{kt} = (\sum_{i \in I_k} d^s_{ik} \tilde{x}^s_{ikt} - \Delta_t)^+$ for all $t \in T, k \in K_t, s \in S$ because $\sum_{i \in I_k} \tilde{x}^s_{ikt} = 1$ for all $t \in T, k \in K_t, s \in S$. Defining ${d^s_{ik}}^+ = (d^s_{ik} - D_t)^+$, we can formulate the problem without variable z^s_{kt} as follows:

$$\min \sum_{s \in S} \sum_{i \in I} p_s f_i y^s_i + \sum_{s \in S} \sum_{t \in T} \sum_{k \in K_t} p_s \big(o_k (\sum_{i \in I_k} {d^s_{ik}}^+ x^s_{ikt}) + \sum_{i \in I_k} c_{ik} x^s_{ikt} \big)$$
$$\text{s.t} \quad (5.9) - (5.10), (5.12) - (5.16)$$

In the preliminary experiments, the formulation with variable $\mathbf{z}$ gave slightly better solution times than the above formulation, therefore we use the original formulation and apply all relaxations on top of that formulation. In Table 5.1 we present some results from these experiments. Except for the first instance, the solution times are lower for the formulation that has variable $\mathbf{z}$. In this table, we also show how the solver (CPLEX) reduces the formulation size with preprocessing. In all of these instances, there are 12 people, 3 projects each requiring 4 tasks among 6 different tasks, and the number of random parameters is 6. When we use the formulation with $\mathbf{z}$, the solver itself eliminates these variables as we can see under the column named *Eliminated columns*. The number of variables and constrains are quite close to each other in the final formulations reduced by the solver. Despite having the same size, the solution times of the formulations are different. This might be due to the fact that the effect and efficiency of the methods applied during the preprocessing depend on the initial formulation given to the solver.

There are more than 10^5 variables and 10^6 constraints in the reduced formulations of the instances in Table 5.1. Next we will use the example in Figure 5.1 to give a better sense of the size of the formulation and how the increase in the number of random parameters affects the size. In this example, we have 16 person-task pairs in total. Assuming that four of them are random, each having

Table 5.1: Comparison of two formulations

		Eliminated rows	Eliminated columns	Reduced rows	Reduced columns	Time (sec.)
1	with **z**	13063	7959	263957	37988	721.34
	without **z**	4370	0	263902	37933	585.53
2	with **z**	122198	8640	139296	22508	229.56
	without **z**	113558	0	139188	22400	277.87
3	with **z**	13118	8188	339718	47412	1059.96
	without **z**	4370	0	339718	47412	1172.22
4	with **z**	93470	8212	190596	29669	426.14
	without **z**	84722	0	190596	29669	555.50
4	with **z**	12169	7582	323171	45447	1129.29
	without **z**	3641	0	322951	45227	1237.19

three possible values, we have $3^4 = 81$ scenarios and $|\mathcal{L}| = 4 \times 3^{(4-1)} = 108$. With this number of scenarios, the problem has around 2,500 binary variables and 8,000 constraints. If there are 8 random person-task pairs, the number of binary variables exceed 200,000 and the constraints exceeds 600,000 while more than half of these are CNACs as $|\mathcal{L}| = 17.496$. This example shows that how an increase in the number of random parameters results in an exponential increase in the size of the formulation.

5.2 Value of Stochastic Solution in Multi-stage Problems with Endogenous Uncertainty

In classical two-stage stochastic programming with recourse, the expected value problem (EVP) is the deterministic problem where the random parameters are replaced by their mean values. Using the solution of EVP we fix the first stage decisions, optimize the second stage and the resulting objective function value is called the expected result of using EVP solution (EEVP). The value of the stochastic solution is the difference between EEVP and the expected cost of the recourse problem (RP). For minimization VSS = EEVP - RS [100].

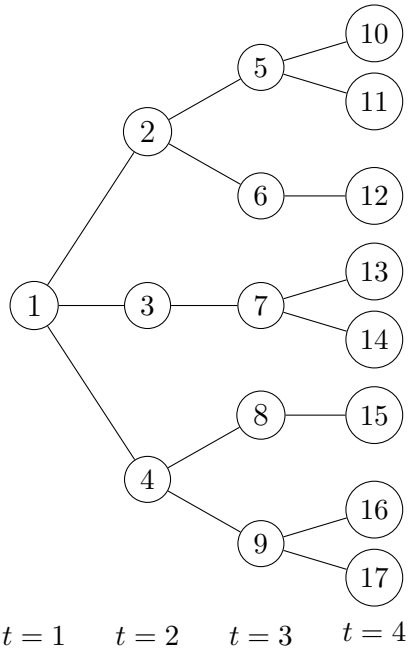


Figure 5.2: A scenario tree [1]

In their study Escudero et al. [1] generalize these definitions to multi-stage stochastic programming with complete recourse and the type of uncertainty is exogenous. The calculation of EEVP for problems with more than two stages is as follows. First, as in the two-stage case, EVP is solved and the first stage decision variables are fixed. Then, the random parameters which are related to the first stage will be observed. For each possible realization of those parameters, a new deterministic problem for the rest of the horizon is created such that the random parameters related to the first stage take their scenario values and all other random parameters take their mean values. Solving these problems the second stage decisions are fixed. For a problem whose scenario tree is shown in Figure 5.2 for example, after fixing the first stage solutions three deterministic problems are created and solved as there are three nodes in this stage. The procedure continues in the same manner until all solutions are fixed and the cost of this solution becomes EEVP. VSS is again the difference between EEVP and the optimal value of the stochastic problem.

In the problems with exogenous uncertainty, the scenario tree is fixed in the sense that we know which random parameters will be observed in which stage.

Consequently, by looking at the scenario tree in Figure 5.2 we know that we need to solve three deterministic problems in the second stage and five in the third stage while calculating EEVP. We also know which random parameter takes its scenario values and which takes its mean value in these problems in advance. In the problems with endogenous uncertainty, the scenario tree is shaped by the decisions made. Therefore, in the calculation of EEVP, at any stage, the values of the random parameters depend on the decisions in the previous stages.

5.3 A Decomposition-based Branch-and-Bound Algorithm

Multi-stage stochastic programming problems are usually computationally difficult mostly due to high number of scenarios. Therefore decomposition methods, which enable separation of the problem under scenarios, are commonly used in the literature. For problems with exogenous uncertainty, nested Benders' decomposition-based approaches, such as stochastic dual dynamic programming [101] and stochastic dual dynamic integer programming [102] methods, are developed. For endogenous case, due to decision-dependent structure, this type of decomposition is not possible, and consequently, Lagrangian decomposition-based methods dominate the literature. The existing methods differ in terms of the type and number of non-anticipativity constraints relaxed and/or dualized, and whether further procedures are applied to impose the relaxed constraints. For example, Goel et al. [52] developed a Lagrangian duality based branch-and-bound algorithm where some of the non-anticipativity constraints are relaxed and others are dualized. Relaxed constraints are imposed by branching similar to the procedure of Caroe and Schultz [103]. Gupta et al. [56] developed a Lagrangian decomposition algorithm where only a subset of non-anticipativity constraints are dualized so that the model decomposes into scenario groups instead of scenarios and give tighter bounds compared to the conventional technique that decomposes by scenarios.

In our problem, for instances with a small number of random parameters the deterministic equivalent formulation introduced in Section 5.1 can be easily solved by a commercial solver. But when we increase the number of random parameters a little further, the solver fails to reach a solution in reasonable time and similarly Lagrangian decomposition algorithms mentioned above require a great amount of time to converge. Therefore, using the combinatorial structure of our problem we propose a branch-and-bound algorithm which improves the lower bound faster than using Lagrange multiplier updating schemes. In our solution methodology we use the idea of scenario groups of Gupta et al. [56] but instead of updating the Lagrange multipliers to approximate the relaxation, we use branching and solve the relaxation to optimality. As it can be understood from the computational experiments in the next chapter, the value of the stochastic solution can be zero or very close to zero in our instances. Therefore, in some cases the solution of the mean value problem is the optimal solution. In such cases, we do not need to solve the stochastic problem or try to find a lower bound. Therefore, if the size of the problem allows, we suggest solving the relaxation where we remove all CNACs first. If the gap between the objective value of the relaxation and EEVP is high enough, then it is meaningful to invest time and computational power to close the gap.

Next, we introduce the relaxation we aim to solve to obtain tight bounds for the problem and describe our branching scheme. Then we explain how different groupings of the scenarios may lead to different performances and describe the heuristic used to obtain upper bounds.

5.3.1 The Relaxation and Branching

We define G to be the index set such that $\bigcup_{g \in G} S_g = S$ and $S_g \cap S_{g'} = \emptyset$ for any g,$g' \in G$. That is, we create a partition of the scenario set S. For S_g, let $\mathcal{L}_g$ be the set of minimal scenarios pairs to write CNACs for the scenarios in S_g only. $\mathcal{L}_g$ can be computed via the greedy algorithm suggested in [55]. Then we aim to

solve the following relaxation of the original problem:

$$\min \sum_{g\in G}\sum_{s\in S_g}\sum_{i\in I} p_s f_i y_i^s + \sum_{g\in G}\sum_{s\in S_g}\sum_{t\in T}\sum_{k\in K_t} p_s\big(o_k z_{kt}^s + \sum_{i\in I_k} c_{ik} x_{ikt}^s\big) \tag{5.18}$$

$$\text{s.t} \sum_{i\in I_k} x_{ikt}^s = 1 \qquad \forall s\in S_g, g\in G, t\in T, k\in K_t \tag{5.19}$$

$$\sum_{k\in K_t} x_{ikt}^s \le y_i^s \qquad \forall s\in S_g,, g\in G, t\in T, i\in I, \tag{5.20}$$

$$\sum_{i\in I_k} d_{ik}^s x_{ikt}^s \le \Delta_t + z_{kt}^s \qquad \forall s\in S_g,, g\in G, t\in T, k\in K_t, \tag{5.21}$$

$$x_{ik1}^{s-1} = x_{ik1}^s \qquad \forall k\in K_1, i\in I_k, s\in S\setminus\{1\}, \tag{5.22}$$

$$x_{ikt}^s - x_{ikt}^{s'} \le \sum_{\tau=1}^{t-1}\sum_{\{i',k'\}\in D_{ss'}} x_{i'k'\tau}^s \qquad \forall t\in T, k\in K_t, i\in I_k, (s,s')\in\mathcal{L}_g, g\in G, \tag{5.23}$$

$$x_{ikt}^{s'} - x_{ikt}^s \le \sum_{\tau=1}^{t-1}\sum_{\{i',k'\}\in D_{ss'}} x_{i'k'\tau}^s \qquad \forall t\in T, k\in K_t, i\in I_k, (s,s')\in\mathcal{L}_g, g\in G, \tag{5.24}$$

$$y_i^s \in \{0,1\} \qquad \forall i\in I, s\in S, \tag{5.25}$$

$$x_{ikt}^s \in \{0,1\} \qquad \forall t\in T, k\in K_t, i\in I_k, s\in S, \tag{5.26}$$

$$0 \le z_{kt}^s \le \Delta_t \qquad \forall t\in T, k\in K_t, s\in S. \tag{5.27}$$

The above formulation is a relaxation because CNACs are no longer written for the scenario pairs in $\mathcal{L}$ but for the $\mathcal{L}_g$ for $g\in G$. So, the CNACs between the scenarios that are in different groups are removed from the formulation. However, the scenarios are still connected by the first stage non-anticipativity constraints (5.22). This relaxation can be decomposed into $|G|$ problems if constraints (5.22) are omitted. To make this decomposition possible, we replace constraints (5.22) with the following:

$$x_{ik1}^s = x_{ik1}^{s'} \qquad \forall k\in K_1, i\in I_k, g\in G, s,s'\in S_g : o_g(s) = o_g(s') - 1 \tag{5.28}$$

where $o(s)$ is the order of scenario $s\in S_g$. This way the first stage non-anticipativity constraints are intact within each scenario set S_g for $g\in G$ and the resulting formulation can be solved by solving $|G|$ independent problems.

By replacing (5.22) with (5.28) we destroy the connection between scenarios which are in different groups, thus the first stage decisions under scenarios $s\in S_g$

and $s' \in S_{g'}$ for $g \neq g'$ might be different. We use branching to remedy this and force the first stage decisions to be identical under all scenarios. First, we explain the notation which will be used to describe the branch-and-bound algorithm. Assume that we are at node r of the tree. We define set Γ_r^+ to be the set of the first stage tasks whose assignments are fixed and let i_k^r be the person who will be assigned to task $k \in \Gamma_r^+$. Set Γ_r^- consists of the first stage tasks which have forbidden assignments and I_k^r is the set of people to which $k \in \Gamma_r^-$ cannot be assigned. Then at node r, given the scenario groups G we solve the following problems for all $g \in G$:

$$\min \sum_{s\in S_g}\sum_{i\in I} p_s f_i y_i^s + \sum_{s\in S_g}\sum_{t\in T}\sum_{k\in K_t} p_s\big(o_k z_{kt}^s + \sum_{i\in I_k} c_{ik} x_{ikt}^s\big) \tag{5.29}$$

$$\text{s.t} \sum_{i\in I_k} x_{ikt}^s = 1 \qquad \forall s \in S_g, t \in T, k \in K_t \tag{5.30}$$

$$\sum_{k\in K_t} x_{ikt}^s \leq y_i^s \qquad \forall s \in S_g, t \in T, i \in I, \tag{5.31}$$

$$\sum_{i\in I_k} d_{ik}^s x_{ikt}^s \leq \Delta_t + z_{kt}^s \qquad \forall s \in S_g, t \in T, k \in K_t, \tag{5.32}$$

$$x_{ik1}^s = x_{ik1}^{s'} \qquad \forall k \in K_1, i \in I_k, s, s' \in S_g : o(s) = o(s') - 1 \tag{5.33}$$

$$x_{ikt}^s - x_{ikt}^{s'} \leq \sum_{\tau=1}^{t-1} \sum_{\{i',k'\}\in D_{ss'}} x_{i'k'\tau}^s \qquad \forall t \in T, k \in K_t, i \in I_k, (s,s') \in \mathcal{L}_g, \tag{5.34}$$

$$x_{ikt}^{s'} - x_{ikt}^s \leq \sum_{\tau=1}^{t-1} \sum_{\{i',k'\}\in D_{ss'}} x_{i'k'\tau}^s \qquad \forall t \in T, k \in K_t, i \in I_k, (s,s') \in \mathcal{L}_g, \tag{5.35}$$

$$y_i^s \in \{0,1\} \qquad \forall i \in I, s \in S_g, \tag{5.36}$$

$$x_{ikt}^s \in \{0,1\} \qquad \forall t \in T, k \in K_t, i \in I_k, s \in S_g, \tag{5.37}$$

$$0 \leq z_{kt}^s \leq \Delta_t \qquad \forall t \in T, k \in K_t, s \in S_g, \tag{5.38}$$

$$x_{i_k^r k1}^s = 1 \qquad s \in S_g, k \in \Gamma_r^+, \tag{5.39}$$

$$x_{ik1}^s = 0 \qquad s \in S_g, k \in \Gamma_r^-, i \in I_k^r. \tag{5.40}$$

Constraints (5.39) ensure assignment of tasks in Γ_r^+ while constrains (5.40) guarantee that the tasks in Γ_r^- are not assigned to any person in their forbidden set. Let P_r denote the relaxation solved at node r by solving these $|G|$ separate problems. We use $r.\mathbf{x}$ to refer to the optimal $\mathbf{x}$ vector of P_r and $r.\nu$ denotes its

Algorithm 3 Branch-and-Bound-MSTFP

UB:=EEVP, LB=0, $\mathbf{x}_{inc}$, UB_{relax} :=*EEVP*
Create root node 0 with $0.\nu := \infty, \Gamma_0^- := \emptyset, \Gamma_0^+ := \emptyset$
Solve P_0 ▷ update UB, UB_{relax} and $\mathbf{x}_{inc}$ if possible
$LB := 0.\nu$
$0.k = \text{TasktoBranch}(0.\mathbf{x})$
if $LB < UB$ **then** $Q := \{0\}$
while $100(UB - LB)/UB > 0.5$ & $LB < UB_{relax}$ **do**
    $r = \arg\min_{\bar{r} \in Q} \{\bar{r}.\nu\}$, $Q := Q \setminus \{r\}$
    for $i \in I^r_{r.k}$ **do**
        Create child node $r.i$ with $\Gamma^+_{r_i} := \Gamma^+_r \cup \{i\}$, $\Gamma^-_{r_i} := \Gamma^-_r$
        Solve P_{r_i} ▷ update UB, UB_{relax} and $\mathbf{x}_{inc}$ if possible
        if $r_i.\nu < UB$ & $r_i.\nu < UB_{relax}$ **then**
            $Q := Q \cup \{r_i\}$
            $r_i.k = \text{TasktoBranch}(r_i.\mathbf{x})$
    Create child node r' with $\Gamma^+_{r'} := \Gamma^+_r$, $\Gamma^-_{r'} := \Gamma^-_r \cup \{i\}$
    Solve $P_{r'}$ ▷ update UB, UB_{relax} and $\mathbf{x}_{inc}$ if possible
    if $r'.\nu < UB$ & $r'.\nu < UB_{relax}$ **then**
        $Q := Q \cup \{r'\}$
        $r'.k = \text{TasktoBranch}(r'.\mathbf{x})$
    $LB := \min_{\bar{r} \in Q} \{\bar{r}.\nu\}$
Return UB, UB_{relax} and $\mathbf{x}_{inc}$

optimal value. The steps of the algorithm are presented in Algorithm 3. The lower and upper bounds for the full problem are denoted as LB and UB. UB_{relax} is the upper bound for the relaxation. The initial step is to create the root node, 0, at which, we solve problem P_0 where the sets Γ_0^- and Γ_0^+ are empty. Using the solution of this problem we check whether we can update the upper bounds and the incumbent solution. This procedure will be explained in the following section. The algorithm runs until the gap between LB and UB is sufficiently small. We follow the best-first search rule for choosing the next node to process, breaking ties arbitrarily. Let r be the current node and $r.k$ be the task used to create child nodes. In child node r_i, $r.k$ is forced to be assigned to $i^r_{r.k}$ and in child node r', $r.k$ is not allowed to be assigned any person in $I^r_{r.k}$.

To choose $r.k$ we use the method called TaskToBranch which is summarized in Algorithm 4. In this method, given the optimal first stage solution of P_r, we

Algorithm 4 TaskToBranch($r.\hat{\mathbf{x}}$)

1: **for** $k \in K_1$ **do**
2: **for** $i \in I_k$ **do**
3: $freq_{ik} = \sum_{s \in S} \hat{x}^s_{ik1}$
4: **for** $k \in K_1$ **do**
5: $I^r_k = \{i \in I_k : freq_{ik} > 0\}$
6: $K_{\text{candidates}} = \left\{ k' \in K_1 : |I^r_{k'}| = \max_{k \in K_1} \{|I^r_k|\} \right\}$
7: **if** $|K_{\text{candidates}}| = 1$ **then**
8: $k' := \{k \in K_{\text{candidates}}\}$
9: **else**
10: **for** $k \in K_{\text{candidates}}$ **do**
11: $diff_k = \max_{i \in I^r_k} \{freq_{ik}\} - \min_{i \in I^r_k} \{freq_{ik}\}$
12: $k' = argmin_{k \in K_{\text{candidates}}} diff_k$
13: **for** $i \in I^r_{k'}$ **do**
14: **if** $freq_{ik}/|S| < \alpha$ **then**
15: $I^r_{k'} := I^r_{k'} \setminus \{i\}$
16: Return k'

record the people who are assigned to a task and in how many of the scenarios they are assigned. We want to choose the task that has the highest assignment variety or diversity. For example, if task k is assigned to three different people under different scenarios and all others are assigned to two different people, then we choose k. If there is a tie, then we favor the one where the assignments are more balanced and we do this in the following way. Let $freq_{ik}$ be the number of scenarios in which k is assigned to i. We choose the task for which the difference between the maximum and minimum of $\{freq_{ik}\}_i$ is smallest. Let k' be the task given by the method. Up to line 12, $I^r_{k'}$ includes all people that is assigned to task k' in at least one of the scenarios. Considering all of these people may result in many child nodes because the number of children of node r will be $|I^r_{k'} + 1|$. Therefore we remove person i from the set I^r_k if the ratio of the scenarios where i is assigned to k' is less than α.

Recall that at node r we solve relaxation P_r by solving $|G|$ independent problems. If all first stage solutions under the scenario groups are identical at node r, then we prune this node and use its objective value, $r.\nu$ to update UB_{relax}.

We prune the node by bound if the lower bound at the node is greater than UB or UB_{relax}. Note that, any feasible solution of the original problem obtained throughout the algorithm can be used as an upper bound as well, which will be explained in the following section.

5.3.2 Scenario Groups and Upper Bounds

Every time a relaxation is solved at a node of the branch-and-bound tree, we use the solution of the relaxation to generate feasible solutions. At node r, in the solution $r.\mathbf{x}$, the first stage assignments can differ among groups since a subset of the first stage non-anticipativity constraints are missing in P_r. Throughout the algorithm, we keep the record of different first stage solutions. Say at node r, we find a new set of first stage decisions $\hat{x}_{ik1}$'s. We create a new scenario set $\hat{S}$ using the scenario durations of the person-task pairs $\{i,k\}$'s that are random and satisfy $\hat{x}_{ik1} = 1$, and using the mean duration values for every other person-task pair. Then we solve a T stage stochastic problem with scenario set $\hat{S}$ and first stage solutions are fixed by $\hat{x}_{ik1}$'s. Then, again we create a new scenario set, considering the second stage solutions of this problem. By fixing the first and second stage decisions under the new scenario set, we again solve a T stage problem. We continue in this manner until all decisions are fixed. Then we calculate the objective value of this feasible solution and update the upper bound and the incumbent solution if necessary. Recall that this is the same procedure that we use to calculate EEVP. The only difference is that in EEVP, the first stage decisions are fixed using the solution of the mean value problem. In short, we apply the same procedure to different first stage solutions to obtain feasible solutions throughout the algorithm.

The required time to solve each problem for $g \in G$ and the tightness of the bound are directly affected by the structure of the partition. When $|G|$ is low, it means the number of scenarios at each group is high, and consequently solving the problems may take longer compared to the case where $|G|$ is high and $|S_g|$ is low. Because in the latter case we have problems with smaller sizes. For instances,

having 27 scenarios, it takes less time to solve a partition where we have 6 groups with $S_1 = \{1, 2, 3\}$, $S_2 = \{4, 5, 6\}, \ldots, S_9 = \{25, 26, 27\}$ compared to the partition with $S_1 = \{1, \ldots, 9\}$, $S_2 = \{10, \ldots, 18\}$, $S_2 = \{9, \ldots, 27\}$. Furthermore, when we have a high number of scenarios in the groups, we expect tighter lower bounds because the number of relaxed constraints is lower. In our 27 scenario case, we expect the partition with 3 groups to give better bounds.

The quality of the bound depends on the content of the scenario groups as well as their cardinality. When we put the scenarios into groups randomly the resulting relaxation might give tighter bounds compared to putting scenarios sequentially. Using the results of the study by Sandıkçı and Özaltın [104] on bounds for multi-stage stochastic programming problems, we suggest the following reasoning for the tightness of the bounds: when we group scenarios randomly we have higher heterogeneity which means each group approximates the scenario sets better than the sequential case where scenarios are much similar to each other. The aforementioned study investigates many different scenario grouping techniques in a more general setting. For example, a scenario can be in multiple groups, or, if exists, a reference scenario can be added to each group. Furthermore, in the study, the group subproblem is solved separately whereas in our study with branching we ensure that the first stage decisions are identical among all groups.

In the following section, we show how different group structures change the performance of both Lagrangian decomposition methods and our branch-and-bound algorithm. For our problem, we suggest using sequential grouping because with random grouping each subproblem takes more time to solve. Also the relation between the group structure and the quality of the bound needs to be investigated in detail, which we consider as future work.

5.4 Experiments

In this section, we first explain our instance generation process and then show how far we can go in terms of the instance size with the mathematical formulation. Then we present some computations to show the limitations of existing solution methods in the multi-stage stochastic programming literature. Lastly, we present experiments on the proposed method and its comparison with the existing methods. In all of the algorithms, to obtain a valid upper bound, we use the heuristic method explained in the previous section. All formulations and 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 are wall-clock times in seconds.

To test our formulations and the algorithm, we have generated random instances. We first randomly decide which tasks are required in the projects, that is, sets K_t are created for $t \in T$. Then we assign expertise levels to people in terms of tasks, which can be equal to 0, 1, 2, or 3. If for task k the level is 0 for a person, then this person is not capable for the task, that is, he or she is not in set I_k. If the expertise level is 3, then the duration this person requires for the task is known with certainty, and it is sufficiently small to complete the project on time. If it is equal to 1 or 2, then it is random, and it can take three values. We refer to these possible durations as low, medium, and high, and we randomly assign values considering the expertise levels. If person i has level 1 expertise on task k and j has level 2, the lowest duration person i requires to finish task k is likely to be higher than the lowest duration person j requires to finish the same task. Similarly, the costs are generated in a way that we usually have $c_{ik} < c_{jk}$.

With this setting, we may end up with many random person-task pairs but the randomness of some of these pairs is most likely to be irrelevant to the problem. Notice that, if task k does not appear in any project or it appears only once, then we do not need to consider any randomness concerning this task. Because, for the former case, the task is completely irrelevant to the problem and for the latter, the information has no value as the task is not repetitive. With this logic, we update

the set of random parameters. Then, we apply a similar pre-process used in the previous chapter. We solve a deterministic problem for each random person-task pair to decide whether their randomness is irrelevant or not. In the deterministic problem solved for $\{i, k\}$, we force this pair to be part of the solution by adding the constraint $\sum_{t \in T} x_{ikt} \geq 1$. Furthermore, in this deterministic problem we use the lowest or the most optimistic duration values for all random parameters. If the objective value of this problem worse than EEVP, we ignore the randomness of this parameter because a solution involving this person-task assignment cannot be optimal.

Table 5.2: Results of full formulation and its relaxation

			Relaxation		Full Formulation	
T	m	EEVP	obj	time	obj	time
3	5	262.22	260.12	0.96	262.00	37.39
3	5	282.56	281.78	3.70	282.56	708.04
3	7	312.40	309.81	11.99	312.27	2183.5
3	7	281.71	275.85	33.68	-	-
3	8	306.95	303.54	115.33	-	-
3	8	263.92	257.00	170.74	-	-
3	8	267.00	258.00	347.01	-	-
4	5	372.01	368.17	0.97	368.17	8.52
4	5	355.49	351.66	0.74	352.00	13.74
4	5	328.93	320.79	0.61	321.32	22.02
4	5	346.21	342.46	0.55	345.00	97.18
4	6	380.95	375.23	4.38	380.95	477.61
4	6	351.04	343.69	5.57	347.00	1404.14
4	6	372.85	367.35	6.30	371.00	2145.75
4	6	365.36	361.63	5.30	364.91	2471.36
4	6	337.30	333.29	5.84	-	-
4	6	338.94	333.00	10.16	-	-
4	7	324.11	318.40	38.66	-	-
4	8	341.62	333.40	556.24	-	-

In Table 5.2 we present results obtained by solving the full deterministic equivalent formulation and its relaxation where CNACs, (5.13)-(5.14), are removed. All instances in this table have 10 people, 6 tasks and each project requires 4 tasks, that is $|I| = 10, |K| = 6, |K_t| = 4$ for $\forall t \in T$. The number of stages/projects are written under the column T and the number of random person-task pairs are

indicated by m. The solver is able to solve the relaxation in a few minutes for all instances but full formulation cannot be solved within an hour when $m = 6$ or higher. This is due to the increase in the number of scenarios and CNACs. The solver requires more than one hour to solve the relaxation when m exceeds 8.

The highlighted rows are the instances that have a clear positive VSS. For the others, VSS is zero or very close to zero or unknown. It is expected to have low VSS for the instances with a small number of random parameters and as we increase this number we encounter more interesting instances. Hence, as it was the case for the problem studied in the previous chapter, for this problem as well, the instances with higher potential of positive VSS are the ones that cannot be solved with the solver directly due to their size. As explained earlier, to solve instances with a higher number of random parameters and scenarios, Lagrangian decomposition methods are proposed in the literature. Next, we compare the performances of several relaxation techniques over some instances.

Lagrangian relaxation (LR) is a widely used method in combinatorial optimization, going back to the seminal work of Held and Karp [105] on the traveling salesman problem. In LR, a set of complicating constraints of the problem is dualized which means they are removed from the constraint set and added to the objective with a penalty term λ also called Lagrange multiplier or dual vector. This "dualized constraint" is chosen in a way that, given λ it is much easier to solve the relaxation LR_λ compared to the original problem. The Lagrangian Dual (LD) is the problem where we find the optimal λ to maximize LR_λ (for a minimization problem). The bound of LD is as tight as the one given by the continuous relaxation of the problem. A drawback of LR is the loss of structure due to dualized constraints. To avoid this loss, Lagrangian decomposition is suggested where copies of the original variables are created and a set of constraints are written with the copy variables while the rest (or all) is written with the original ones [106]. The dualized constraint in this case is the equality of the original and copy variables. This decomposition method is commonly used in stochastic programming. In two-stage stochastic programming problems, by creating a copy of the first stage variables under all scenarios and then dualizing the equality of these variables, we end up with a relaxation that decomposes into

independent problems for each scenario. In multi-stage stochastic problems, the non-anticipativity constraints, which force the equality of decision variables under the scenarios that have the same history, are relaxed or dualized so that the problem decomposes by scenarios.

In Lagrangian relaxation or decomposition, the Lagrangian dual is solved or approximated by multiplier updating schemes among which the subgradient method has been used extensively. It is an adaptation of the gradient method where gradients are replaced by the subgradients. Let $Ax \leq b$ is the dualized constraint in a minimization problem and x^k be the solution at iteration k of the update scheme, then the multiplier is updated by the rule $\lambda_{k+1} = \lambda_k + t_k(Ax^k - b)$ where t_k is the step size. The step size is usually taken as $t_k = \dfrac{\alpha(UB - LD(\lambda_k))}{||Ax^k - b||^2}$, where UB is the best upper bound known, $LD(\lambda_k)$ is the value of Lagrangian dual with the current multiplier and α is the scaling factor. Since this method gives a lower bound for the problem (minimization), heuristics are applied to generate feasible solutions and upper bounds.

In all of the Lagrangian decomposition methods implemented in this study, to update the Lagrange multipliers, we use the subgradient method of Held and Karp [105][107] and its standard application described by Fisher [108]. We start with all multipliers being equal to zero. If the lower bound is not improved in a certain number of consecutive iterations, the scale is halved. There are different updating schemes developed and used in the related literature, such as [109] and [110], which use a combination of cutting planes, subgradient, and trust-region strategies. Although these procedures have the potential to improve the algorithm, they are still likely to require a high number of iterations to reach good lower bounds. Our focus in this study is not the multiplier updating scheme but to suggest a completely different alternative to it.

In Figure 5.3 we show the improvement of the lower bounds in different Lagrangian decomposition algorithms. The instance we use in this comparison has the following set sizes: $|T| = 4, |I| = 10, |K| = 5, |K_t| = 4$ for $t \in T$ and $m = 8$. For this instance, we are able to solve the relaxation, *noCNACs*, where CNACs

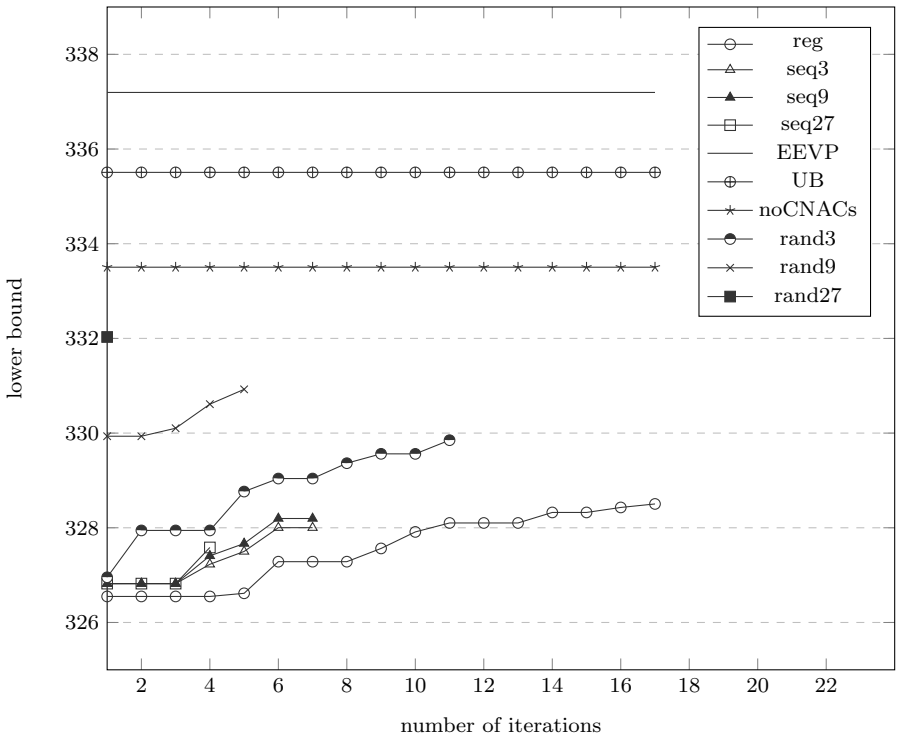


Figure 5.3: Lower bound improvements of various algorithms over an instance with $|T| = 4$, $|I| = 10$, $|K| = 5$, $|K_t| = 4$ for $t \in T$, $m = 8$

are relaxed, and it takes 452.82 seconds. We chose an instance for which the relaxation can be solved so that we can compare this bound with the bounds from the algorithms. The algorithms are stopped at the first iteration that exceeds 1000 seconds of running time.

- The term *reg* refers to the conventional Lagrangian decomposition where first stage non-anticipativity constraints are dualized and all CNACs are relaxed. The multipliers are updated via the subgradient method.
- The term *seq** refers to the decomposition technique proposed in [56] where $*$ corresponds to the number of scenarios in each group and the scenarios are put in groups sequentially. It amounts to dualizing the first stage non-anticipativity constraints (5.33) in the formulation (5.18)-(5.27). And the corresponding Lagrange multipliers are updated via the subgradient method.

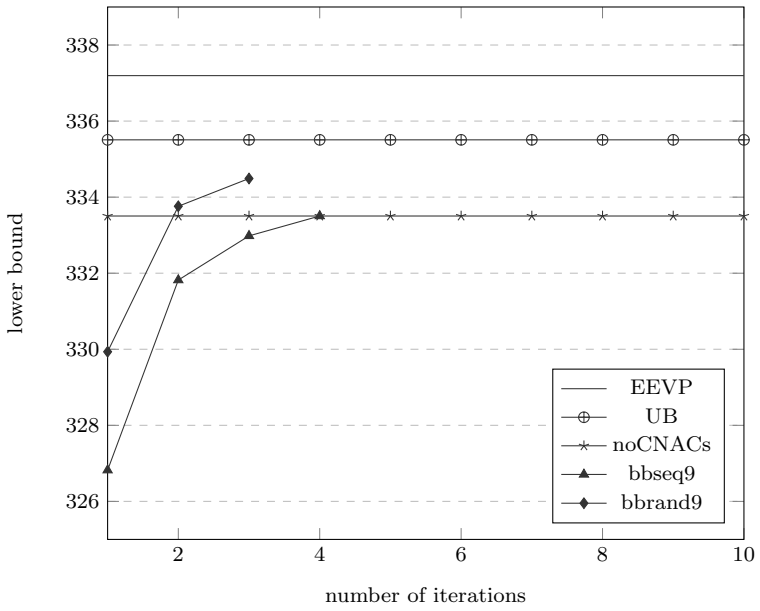


Figure 5.4: Lower bound improvements of the branch-and-bound algorithms over the instance with $|T| = 4$, $|I| = 10$, $|K| = 5$, $|K_t| = 4$ for $t \in T$

- *rand** is the same algorithm as *seq∗* except that the scenarios are put in groups randomly rather than sequentially.

Besides the bound improvements of these algorithms, we present the EEVP and the upper bound (UB) found throughout the algorithms in the plots. We use the same method to update the upper bounds and if the bound is shown in the plot then in all of algorithms presented this bound is reached. When we compare the sequential methods, *seq3*, *seq9*, *seq27*, among themselves we see how the bounds improve when we increase the number of scenarios in a group. We can regard *reg* as *seq1*. The best bound *reg* can reach is exactly the bound we get from the relaxation *noCNACs* and in the figure we see how slow the convergence is. The bounds provided by the random groupings are clearly better than their corresponding sequential cases and again we can see that as the size of a group increases the bound increases as well. We can also see how the time each iteration takes differs as we change the group size. With *rand27* a single iteration takes around 1000 seconds where with *rand9* it takes approximately 200 seconds.

In Figure 5.4, for the same instance, we plot the progress of two branch-and-bound algorithms. *bbseq*9 has sequential scenario groups each having 9 scenarios, and *bbrand*9 has random scenario groups. In this case, the number of iterations refers to the number of levels in the branch-and-bound tree. *bbseq*9 takes 880 seconds while *bbrand*9 takes 1429 seconds. Notice that *bbseq*9 reaches to the same bound as *noCNACs* and while the bound provided by *bbrand*9 is tighter. Clearly, there is a trade-off between computational time and the quality of the bound. The cases where we get tighter bounds are the cases where each iteration takes longer to solve.

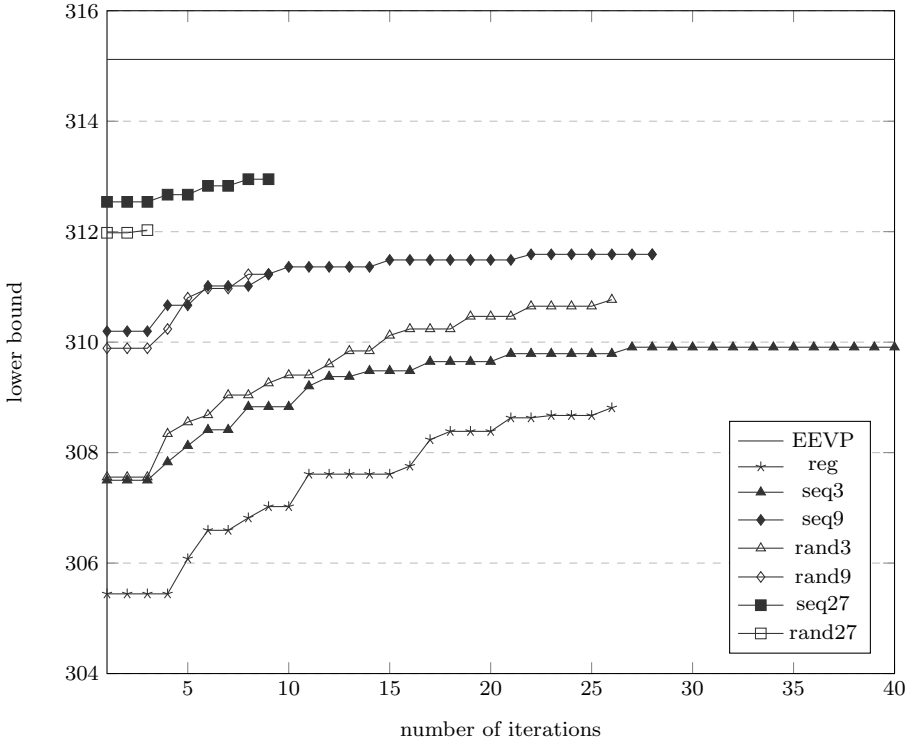


Figure 5.5: Comparison of decomposition algorithms over an instance with $|T| = 4$, $|I| = 12$, $|K| = 6$, $|K_t| = 4$ for $t \in T$, $m = 9$ and at least two-hours of running time

The instance whose results are shared in Figure 5.5 has 4 stages, 12 people, and 9 random parameters, while the total number of tasks and the number of tasks required at each stage is 6 and 4 respectively. Here the algorithms are stopped at the first iteration completed after 2 hours. Again we can see how bounds improve when we increase the sizes of the scenario groups and also how

each iteration takes much longer time. In this example, with groups of size 3, the bound of the random groups is quite better than the sequential groups. For sizes 9 and 27, sequential approaches perform better both in terms of time and bound. We would like to point out that detailed extensive computations are required to compare sequential and random group strategies such as the one in the study of Sandıkçı and Özaltın [104].

For the instance in Figure 5.5, we present the results of the branch-and-bound algorithms with different scenario groups in Table 5.3. If the algorithm is terminated before two hours, it means that the corresponding relaxation is solved to optimality, as it is the case for *bbrand9*. Among these branch-and-bound algorithms, *bbseq27* gives the best bound in less than 2 hours. With this bound, it proves that the solution obtained by the mean value problem has an optimality gap of 0.3%.

The optimal way of grouping the scenarios may be a research problem itself. To provide some insight, we present the initial bounds of *seq*9 and *rand*9 algorithms over two instances in Table 5.4. As mentioned before, we believe that random groupings usually give tighter bounds because each group individually represents the whole scenario set, and consequently the problem, better with respect to a group created sequentially. In other words, the scenarios in a randomly generated group are more likely to be more different than one another, which makes the group more heterogeneous. One can actually devise a metric to measure this heterogeneity of groups. Here we present a very basic idea just to deliver our argument clearly. In our instances, each random parameter can take three values, which are low, medium and high. We represent these values as 0, 1, and 2. Using

Table 5.3: Bound improvements of the branch-and-bound algorithms over the instance with $|T| = 4$, $|I| = 12$, $|K| = 6$, $|K_t| = 4$ for $t \in T$, $m = 9$

	bbseq9		bbrand9		bbseq27		bbrand27	
iter	time	LB	time	LB	time	LB	time	LB
1	258	310.19	4747	312.68	778	312.54	2793	311.98
2	1206	311.05	8636	313.02	4553	314.13	10501	313.468
3	2358	312.57	-	-	5946	314.16	-	-

Table 5.4: Average heterogeneity and bounds of different algorithms

	Instance 1		Instance 2	
	bound	avg heterogeneity	bound	avg heterogeneity
*seq*9	202.92	72	195.41	72
*rand*9-1	204.79	188	198.33	185.06
*rand*9-2	204.71	185.95	198.96	186.29
*rand*9-3	204.67	185.06	198.5	187.28

this representation of the scenarios and regarding them as points in space, we define the distance between two scenarios as the Manhattan distance. Then, the heterogeneity of a group is measured by the sum of all pairwise distances of the scenarios. Having 6 random parameters, when we put scenarios into groups of 9 scenarios sequentially, heterogeneity of each group becomes 72 which makes the average heterogeneity 72 as well. In the following table we show that when we group the scenarios randomly in *rand*9, the average heterogeneity increases and the algorithm gives stronger initial bounds. As the performance of the algorithm with random groups requires more investigation and as it usually has longer solution times, we leave it for future research and continue our experiments with the algorithm that groups the scenarios sequentially.

For another instance with three stages, we plot the results of *seq*9, *seq*27, *bbseq9*, and *bbseq27* in Figure 5.6. In this plot x-axis is the time. The problem we solve at the first node of the branch-and-bound tree in *bbseq9* (*bbseq27*) and the problem we solve at the first iteration of *seq9* (*seq27*) are the same. Within this first iteration, in all algorithms, we find a better solution than that of EEVP so we are able to update the upper bound. Among these four algorithms, *bbseq9* performs the best and it solves the relaxation in 47 minutes while *bbseq27* takes 56 minutes. The convergence of *seq9* and *seq27* takes more than two hours. By the bound provided by *bbseq9*, we learn that the optimality gap of the incumbent is 1.1%. Hence for such instances, for which we are not able to solve the relaxation without CNACs, the branch-and-bound algorithm provides tight bounds faster than the Lagrangian decomposition methods. Furthermore, even for the instances where the relaxation can be solved within the time limit, the branch-and-bound algorithm is still worth the effort because it solves a stronger relaxation.

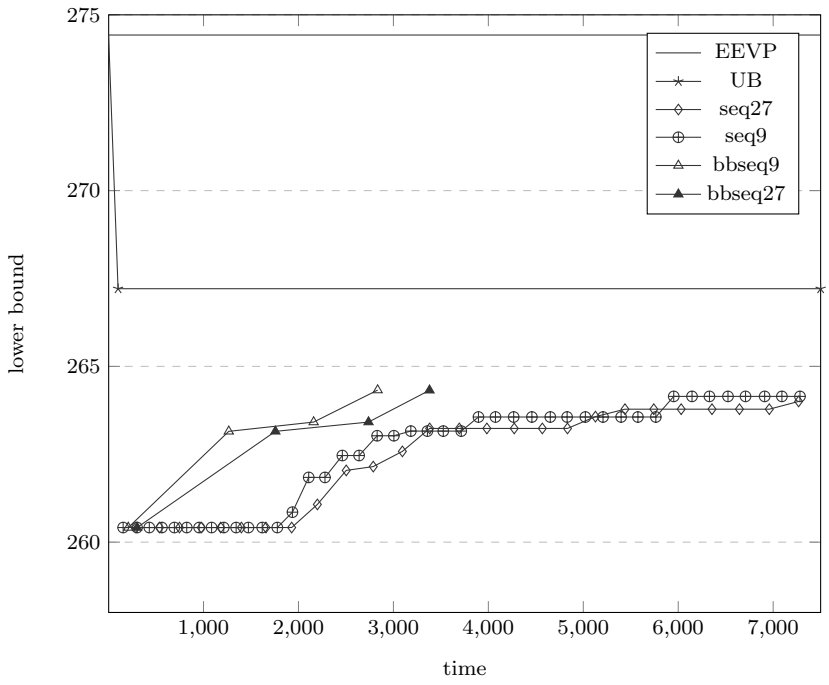


Figure 5.6: Comparison of decomposition algorithms with an instance with $|T| = 3$, $|I| = 15$, $|K| = 6$, $|K_t| = 4$ for $t \in T$,$m = 9$ and at least two-hours of running time

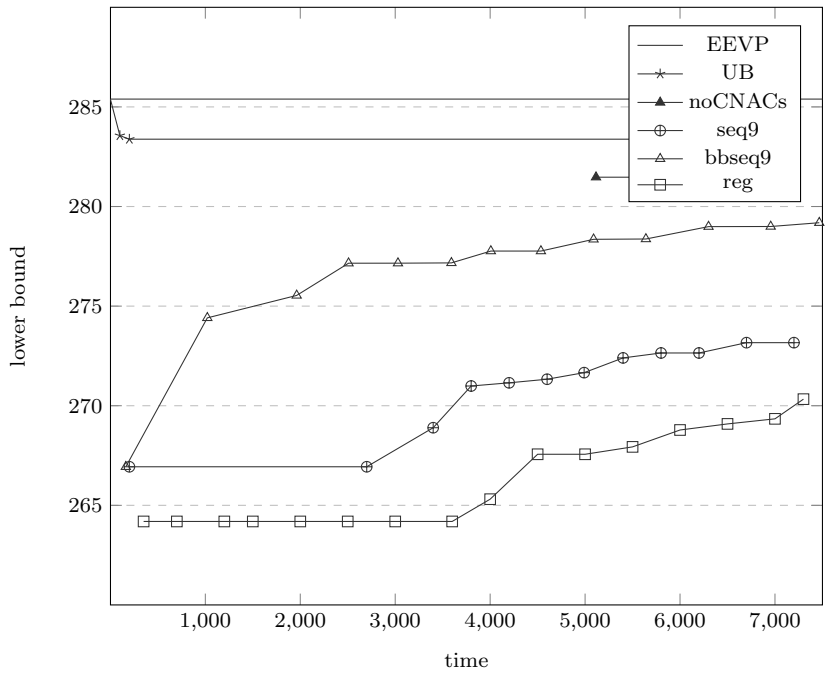


Figure 5.7: Comparison of decomposition algorithms with an instance with $|T| = 3$, $|I| = 15$, $|K| = 6$, $|K_t| = 4$ for $t \in T$, $m = 9$ and at least two-hours of running time

In Figure 5.7, the solver is able to solve the relaxation $noCNACs$ within 2 hours. Compared to $seq9$ and $reg9$, our algorithm $bbseq9$ performs quite well but in this instance it cannot reach the bound of $noCNACs$ within 2 hours. This result does not undermine the importance of our algorithm because memory problems are very likely for the instances of this size and solving $noCNACs$ is not always possible. Furthermore, our algorithm has the potential to exceed the bound of $noCNACs$.

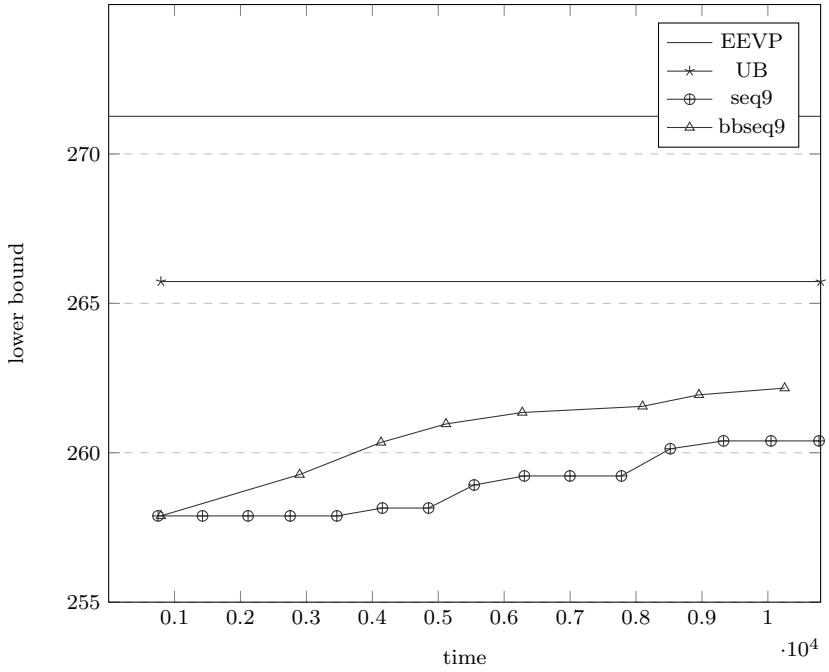


Figure 5.8: Comparison of $bbseq9$ and $seq9$ with an instance with $|T| = 3$, $|I| = 15$, $|K| = 6$, $|K_t| = 4$ for $t \in T$,$m = 10$ and 3-hours of running time

In Figure 5.8, we compare the bound improvements for an instance where the number of random parameters is 10. For this instance the relaxation $noCNACs$ cannot be solved due to insufficient memory. We keep the solution times longer for this instance, up to 3 hours. In 3 hours $bbseq$ is able to decrease the gap to 1.36% while seq decreases it to 2.04%.

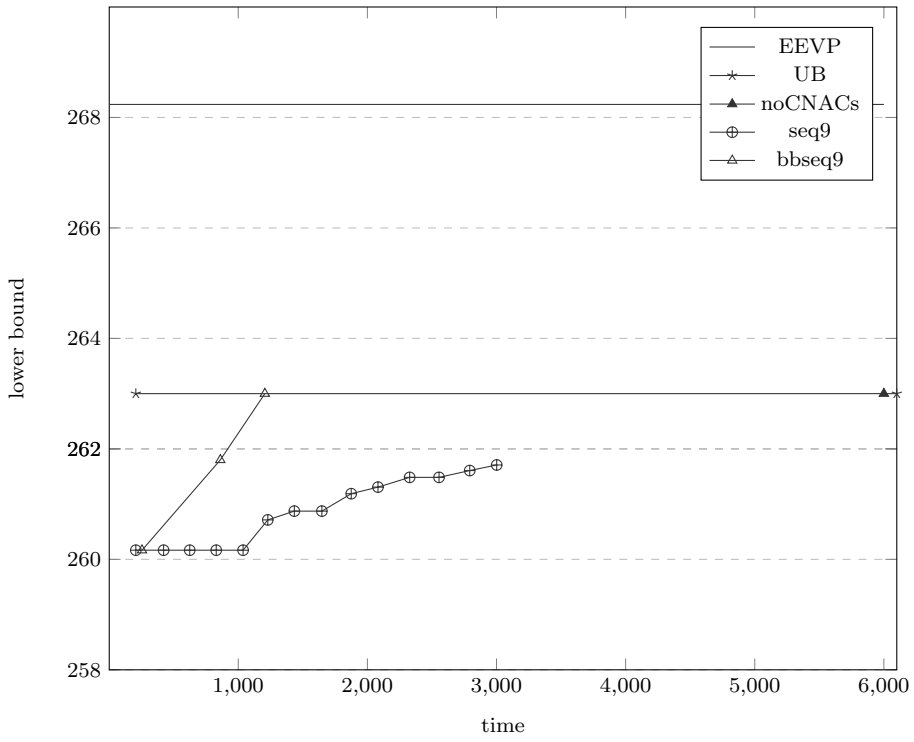


Figure 5.9: Comparison of *bbseq*9 and *seq*9 with an instance with $|T| = 3$, $|I| = 15$, $|K| = 6$, $|K_t| = 4$ for $t \in T$,$m = 9$ where

In Figure 5.9, we compare the bound improvements for an instance where the relaxation $noCNACs$ can be solved within 2 hours and its objective value, 263, turns out be optimal which we know because we have found a feasible solution with objective value 263 throughout the algorithm. For this instance, *bbseq*9 proves that this feasible solution is actually optimal by reaching the lower bound 263 in 1200 seconds.

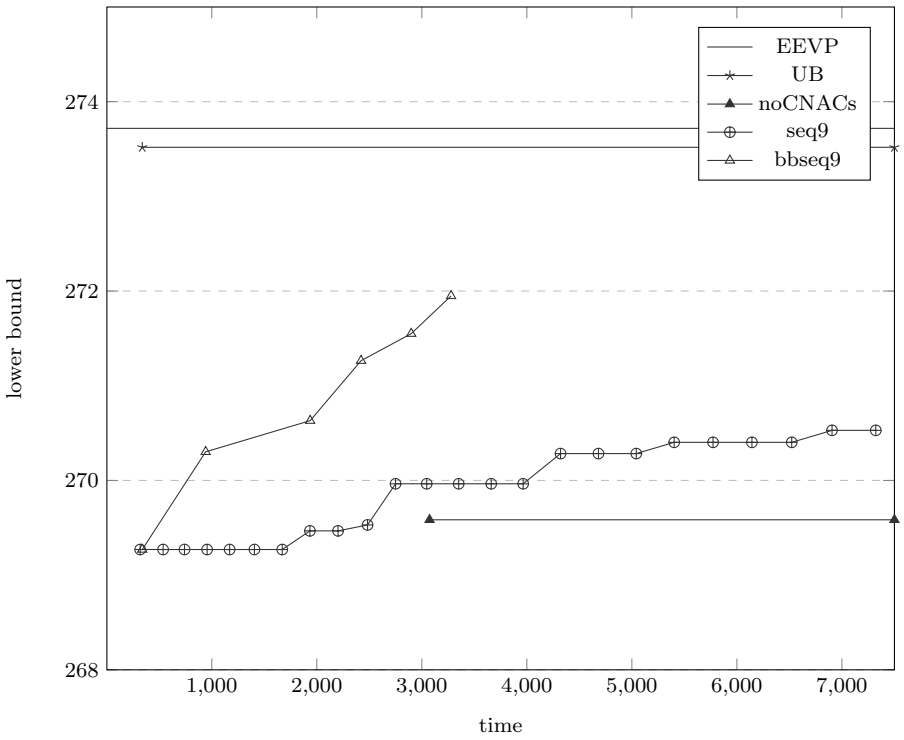


Figure 5.10: Comparison of *bbseq*9 and *seq*9 with an instance with $|T| = 3$, $|I| = 15$, $|K| = 6$, $|K_t| = 4$ for $t \in T$,$m = 9$ where the relaxation $noCNACs$ is solved within 1 hour

In Figure 5.9, we compare the bound improvements for an instance where the relaxation $noCNACs$ can be solved within an hour. Our algorithm *bbseq*9 also terminates within an hour with 0.5% gap and a lower bound that is higher than the one provided by $noCNACs$.

5.5 Conclusion

In this chapter, we introduced a multi-stage stochastic team formation problem where each stage corresponds to a different but similar project. Each project requires several tasks and by deciding which person to assign to which tasks, we aim to complete the projects with minimum expected cost of hiring and outsourcing. We assume for some person-task pairs the time required to complete

the task is random but once it is observed, its true value becomes known, thus the uncertainty is endogenous or decision-dependent. Noticing the gap in the related literature, we first explained how the value of the stochastic solution can be calculated for the problems of this type. Then we presented an integer programming formulation for the problem and showed that the size of the formulation easily exceeds the capabilities of commercial solvers. To be able to provide tight lower bounds for the problem we proposed a decomposition-based branch-and-bound algorithm where some non-anticipativity constraints are left in the problem, some are relaxed and some are forced through branching. Presenting computational experiments with random instances, we showed that our algorithm is an alternative to the existing decomposition algorithms and bounding methods in the literature. It is mostly suitable for the problems where the number of scenarios is high and the number of first stage decisions is relatively limited.

The processing time of the algorithm can be significantly decreased with the use of higher computational power and parallel computing techniques. Moreover, when necessary more sophisticated methods can be incorporated into the algorithm to create feasible solutions and decrease the upper bound.

Chapter 6

Conclusion

In this dissertation, we studied different variants of the team formation problem. First, we considered a deterministic team formation problem where potential team members are assumed to constitute a social network in which an edge weights is a measure for the quality of communication between the people it connects. We refer to the edge weights as communication costs and the aim is to build a team with the minimum communication cost while the team members must be collectively capable of a given set of required tasks. Investigating different cost functions in the related literature, we decided to minimize the sum of all pairwise communication costs while putting a limit on the highest one. The technical capabilities of people are represented by a binary skill matrix. We formulated this problem as a quadratic set covering problem with packing constraints.

We showed that with this formulation, small and medium-sized instances can be solved using a general-purpose solver but memory problems occur for large instances. We developed a novel branch-and-bound algorithm which is very effective in solving these instances. We first presented a reformulation to the problem using a partial application of the reformulation-linearization technique. Then, the reformulation is relaxed in a way that it decomposes into a series of linear set covering problems and can be solved efficiently. The relaxed constraints are imposed by a novel branching strategy.

Next, we studied a two-stage stochastic team formation problem with random communication costs. The first stage is a trial stage where a limited number of communication costs can be observed by selecting the corresponding pairs. In the second stage, we form a capable team with minimum expected communication cost. So this problem can be regarded as an extension of the previous one where a trial stage is added to the beginning. We assumed that for a subset of pairs, the cost of communication is not known with certainty but the possible values it can take and their respective probabilities are known. If a pair with random cost is selected in the first stage, then we learn the true cost value. Hence, the uncertainty in this problem is decision-dependent because we assume a complete resolution of uncertainty for the pairs which are selected in the first stage. For this problem, we define a concept called *value of learning* which measures the improvement we get by the information obtained in the first stage.

We first presented a formulation of the problem where we use the same modeling approach in the related literature and this formulation contains a higher number of non-anticipativity constraints. Then we gave an alternative formulation which does not have these constraints but a quadratic objective function which is linearized by defining an extra set of binary variables. We proved that these formulations are equivalent. By generating instances with different sizes, we show that for small-sized instances these formulations can be solved by a commercial solver. To be able to solve larger sizes, a Benders' decomposition-based branch-and-cut algorithm was developed on top of the second formulation. The algorithm utilizes a strong linear relaxation of the second stage problem which not only gives strong optimality cuts but also decreases the burden of solving the integer problems by providing integral solution frequently. The algorithm solves the problems with thousands of scenarios because at each iteration it creates and works on a smaller scenario set which is possible due to the decision-dependent structure. Both in terms of modeling and solution methodology, our approach contributes to the relevant literature.

As a final problem, we studied a multi-stage team formation problem where each stage corresponds to a project. While in the first two problems the focus was on the quality of communication among the members, in the last one the concern

is monetary. The aim is to minimize the expected hiring and outsourcing costs for the whole horizon while having adequate team members to complete the required tasks. We consider uncertainty in people's performances and represented it by the randomness in the time required by a person to finish a task. We assumed that once a person is assigned to a task of a project, we observe their true performance on this task so here again the type of uncertainty is decision-dependent.

The alternative modeling we gave for the two-stage stochastic problem is not applicable for multi-stage problems. Therefore, the formulation we presented for this problem consists of a large number of non-anticipativity constraints. We showed that instances of very limited size can be solved to optimality directly with the solver using this formulation. For larger sizes, we investigated efficient methods to obtain near-optimal solutions. On randomly generated instances, we tested existing Lagrangian decomposition methods and showed that different decomposition rules can have very different performances and there is a trade-off between quality of the bound and the time spent at each iteration. As an alternative to Lagrange multiplier updating schemes whose convergences are very slow, we proposed a decomposition-based branch-and-bound algorithm where first stage non-anticipativity constraints are imposed by branching on first stage decisions.

Next we summarize the contributions of the thesis and conclude with possible future research directions. This thesis in general contributes to the literature of team formation problems. The work in Chapter 3 can be considered as a bridge between the studies in data science and operations research literatures. Because it adopts the problem definitions used in data science field but provides an integer programming formulation, and unlike the related studies in the operations research, we solve this integer formulation to optimality with a novel branch-and-bound algorithm. Proposing this algorithm, Chapter 3 also contributes to the literature of 0-1 quadratic problems as the algorithm can be utilized to solve any such problem.

The problems in Chapter 4 and 5 contribute the stochastic programming literature with endogenous uncertainty which is not as vast as the exogenous case. In

Chapter 4 we provided an alternative formulation to a two-stage stochastic problem with endogenous uncertainty. This formulation does not have the conditional non-anticipativity constraints, which are a set of complicating constraints that the stochastic problems with this type uncertainty usually have. This alternative formulation is significant because without the conditional non-anticipativity constraints, we are able to apply Benders' decomposition to this problem. Thus, we developed a Benders' decomposition based branch-and-cut algorithm where the strength of the cuts were enhanced with stronger linear relaxations. In Chapter 5, we introduced a multi-stage team formation problem and we proposed a decomposition-based branch-and-bound algorithm as an alternative to Lagrange multiplier update methods. We showed the superiority of the algorithm over existing Lagrangian decomposition techniques. This work contributes to the multi-stage stochastic programming in general because the algorithm can be utilized to generate tight bounds for any type of stochastic problem and it would be promising especially for the problems with combinatorial structure and a high number of scenarios.

6.1 Future Research

In terms of application, the work in Chapter 3 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. In this case, additional constraints such as capacity constraints can be added. In terms of extending the work computationally, one can investigate other 0-1 quadratic problems that the branch-and-bound algorithm can solve efficiently.

In Chapter 4, we consider the uncertainty in the communication costs and assume that a subset of team can be learned in the trial stage. An interesting extension could be a stochastic problem with two or more stages where we must have a capable team at each stage. In this case, according to our observations of communication cost, we can update the team but now there can be extra restrictions such as a bound on the number of team members that can be changed.

Or the changes might incur a cost. In terms of modeling of stochastic problems with endogenous uncertainty, one can also investigate if the problems in the literature can be modeled without conditional non-anticipativity constraints as well.

The work in Chapter 5 can be extended in many ways. Obviously experimenting on the grouping idea and investigating the difference of the bounds with respect to the structure of the scenario groups would be beneficial. Other Lagrange multiplier update methods can be applied to the decomposition. If these methods accelerate the algorithm, they can be utilized within the branch-and-bound as well and we can solve stronger relaxation at the nodes of the branch-and-bound tree. Furthermore, with parallel computing the solution time of the branch-and-bound can be reduced. Parallel computing is a type of computation where the execution of processes are carried out simultaneously using multiple processors. Once the child nodes of the current node are created, they can be solved simultaneously if we have the sufficient computing power. As a future work, the algorithm can be tested on other stochastic programming problems with endogenous uncertainty in literature and can be adjusted for the ones that have exogenous uncertainty as well.

Bibliography

[1] L. F. Escudero, A. Garín, M. Merino, and G. Pérez, "The value of the stochastic solution in multistage problems," *Top*, vol. 15, no. 1, pp. 48–64, 2007.

[2] 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.

[3] 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*, vol. 23, no. 3, pp. 239–290, 1997.

[4] G. L. Stewart, "A meta-analytic review of relationships between team design features and team performance," *Journal of management*, vol. 32, no. 1, pp. 29–55, 2006.

[5] M. Hoegl and H. G. Gemuenden, "Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence," *Organization Science*, vol. 12, no. 4, pp. 435–449, 2001.

[6] R. Jones, *Working Virtually: Challenges of Virtual Teams: Challenges of Virtual Teams*. IGI Global, 2005.

[7] Joint Commission, "Sentinel event statistics released for 2014." Retrieved October 20, 2019, `https://www.jointcommission.org/assets/1/23/JC_Online_March_13.pdf`, 2015.

[8] Project Management Institute, "The high cost of low performance:the essential role of communications." Retrieved October 20, 2019, https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/the-essential-role-of-communications.pdf, 2013.

[9] K. Schwaber, *Agile project management with Scrum*. Microsoft press, 2004.

[10] 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*, pp. 467–476, ACM, 2009.

[11] R. S. Huckman, B. R. Staats, and D. M. Upton, "Team familiarity, role experience, and performance: Evidence from indian software services," *Management science*, vol. 55, no. 1, pp. 85–100, 2009.

[12] 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*, vol. 18, no. 4, pp. 613–630, 2007.

[13] 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.

[14] N. Bassett-Jones, "The paradox of diversity management, creativity and innovation," *Creativity and innovation management*, vol. 14, no. 2, pp. 169–175, 2005.

[15] A. Zakarian and A. Kusiak, "Forming teams an analytical approach," *IIE transactions*, vol. 31, no. 1, pp. 85–97, 1999.

[16] B. H. Boon and G. Sierksma, "Team formation: Matching quality supply and quality demand," *European Journal of Operational Research*, vol. 148, no. 2, pp. 277–292, 2003.

[17] 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*, vol. 38, no. 2, pp. 484–495, 2011.

[18] 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*, vol. 51, no. 2, pp. 111–124, 2004.

[19] E. L. Fitzpatrick and R. G. Askin, "Forming effective worker teams with multi-functional skill requirements," *Computers & Industrial Engineering*, vol. 48, no. 3, pp. 593–608, 2005.

[20] L. Zhang and X. Zhang, "Multi-objective team formation optimization for new product development," *Computers & Industrial Engineering*, vol. 64, no. 3, pp. 804–811, 2013.

[21] A. Baykasoglu, T. Dereli, and S. Das, "Project team selection using fuzzy optimization approach," *Cybernetics and Systems: An International Journal*, vol. 38, no. 2, pp. 155–185, 2007.

[22] 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*, vol. 75, pp. 150–162, 2016.

[23] H. Wi, S. Oh, J. Mun, and M. Jung, "A team formation model based on knowledge and collaboration," *Expert Systems with Applications*, vol. 36, no. 5, pp. 9121–9134, 2009.

[24] B. Feng, Z.-Z. Jiang, Z.-P. Fan, and N. Fu, "A method for member selection of cross-functional teams using the individual and collaborative performances," *European Journal of Operational Research*, vol. 203, no. 3, pp. 652–661, 2010.

[25] A. Farasat and A. G. Nikolaev, "Social structure optimization in team formation," *Computers & Operations Research*, vol. 74, pp. 127–142, 2016.

[26] 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*, pp. 985–994, ACM, 2011.

[27] 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*, pp. 483–498, Springer, 2012.

[28] A. Bhowmik, V. S. Borkar, D. Garg, and M. Pallan, "Submodularity in team formation problem.," in *SDM*, pp. 893–901, SIAM, 2014.

[29] 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*, pp. 1005–1013, ACM, 2012.

[30] 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"*, pp. 472–489, Springer, 2010.

[31] 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.

[32] C. Crawford, Z. Rahaman, and S. Sen, "Evaluating the efficiency of robust team formation algorithms," in *International Conference on Autonomous Agents and Multiagent Systems*, pp. 14–29, Springer, 2016.

[33] E. Demirović, N. Schwind, T. Okimoto, and K. Inoue, "Recoverable team formation: Building teams resilient to change," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1362–1370, International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[34] M. Fathian, M. Saei-Shahi, and A. Makui, "A new optimization model for reliable team formation problem considering experts' collaboration network," *IEEE Transactions on Engineering Management*, vol. 64, no. 4, pp. 586–593, 2017.

[35] S. Bouajaja and N. Dridi, "A survey on human resource allocation problem and its applications," *Operational Research*, vol. 17, no. 2, pp. 339–369, 2017.

[36] A. Certa, M. Enea, G. Galante, and C. Manuela La Fata, "Multi-objective human resources allocation in r&d projects planning," *International Journal of Production Research*, vol. 47, no. 13, pp. 3503–3523, 2009.

[37] W. J. Gutjahr, S. Katzensteiner, P. Reiter, C. Stummer, and M. Denk, "Competence-driven project portfolio selection, scheduling and staff assignment," *Central European Journal of Operations Research*, vol. 16, no. 3, pp. 281–306, 2008.

[38] R. Chen, C. Liang, D. Gu, and J. Y. Leung, "A multi-objective model for multi-project scheduling and multi-skilled staff assignment for it product development considering competency evolution," *International Journal of Production Research*, vol. 55, no. 21, pp. 6207–6234, 2017.

[39] F. Rahmanniyay, A. J. Yu, and J. Seif, "A multi-objective multi-stage stochastic model for project team formation under uncertainty in time requirements," *Computers & Industrial Engineering*, vol. 132, pp. 153–165, 2019.

[40] M. E. Bruni, P. Beraldi, F. Guerriero, and E. Pinto, "A heuristic approach for resource constrained project scheduling with uncertain activity durations," *Computers & Operations Research*, vol. 38, no. 9, pp. 1305–1318, 2011.

[41] M. E. Bruni, L. D. P. Pugliese, P. Beraldi, and F. Guerriero, "An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations," *Omega*, vol. 71, pp. 66–84, 2017.

[42] T. W. Jonsbråten, R. J. Wets, and D. L. Woodruff, "A class of stochastic programs withdecision dependent random elements," *Annals of Operations Research*, vol. 82, pp. 83–106, 1998.

[43] S. Ahmed, *Strategic planning under uncertainty: Stochastic integer programming approaches*. PhD thesis, University of Illinois at Urbana-Champaign, 2000.

[44] S. Peeta, F. S. Salman, D. Gunnec, and K. Viswanath, "Pre-disaster investment decisions for strengthening a highway network," *Computers & Operations Research*, vol. 37, no. 10, pp. 1708–1719, 2010.

[45] V. Goel and I. E. Grossmann, "A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves," *Computers & chemical engineering*, vol. 28, no. 8, pp. 1409–1429, 2004.

[46] M. Colvin and C. T. Maravelias, "A stochastic programming approach for clinical trial planning in new drug development," *Computers & Chemical Engineering*, vol. 32, no. 11, pp. 2626–2642, 2008.

[47] S. Solak, J.-P. B. Clarke, E. L. Johnson, and E. R. Barnes, "Optimization of r&d project portfolios under endogenous uncertainty," *European Journal of Operational Research*, vol. 207, no. 1, pp. 420–433, 2010.

[48] V. Goel and I. E. Grossmann, "A class of stochastic programs with decision dependent uncertainty," *Mathematical programming*, vol. 108, no. 2-3, pp. 355–394, 2006.

[49] B. Tarhan and I. E. Grossmann, "A multistage stochastic programming approach with strategies for uncertainty reduction in the synthesis of process networks with uncertain yields," *Computers & Chemical Engineering*, vol. 32, no. 4-5, pp. 766–788, 2008.

[50] B. Tarhan, I. E. Grossmann, and V. Goel, "Computational strategies for non-convex multistage minlp models with decision-dependent uncertainty and gradual uncertainty resolution," *Annals of Operations Research*, vol. 203, no. 1, pp. 141–166, 2013.

[51] M. Colvin and C. T. Maravelias, "Modeling methods and a branch and cut algorithm for pharmaceutical clinical trial planning using stochastic programming," *European Journal of Operational Research*, vol. 203, no. 1, pp. 205–215, 2010.

[52] V. Goel, I. E. Grossmann, A. S. El-Bakry, and E. L. Mulkay, "A novel branch and bound algorithm for optimal development of gas fields under uncertainty in reserves," *Computers & chemical engineering*, vol. 30, no. 6-7, pp. 1076–1092, 2006.

[53] B. Tarhan, I. E. Grossmann, and V. Goel, "Stochastic programming approach for the planning of offshore oil or gas field infrastructure under decision-dependent uncertainty," *Industrial & Engineering Chemistry Research*, vol. 48, no. 6, pp. 3078–3097, 2009.

[54] N. Boland, I. Dumitrescu, and G. Froyland, "A multistage stochastic programming approach to open pit mine production scheduling with uncertain geology," *Optimization online*, pp. 1–33, 2008.

[55] N. Boland, I. Dumitrescu, G. Froyland, and T. Kalinowski, "Minimum cardinality non-anticipativity constraint sets for multistage stochastic programming," *Mathematical Programming*, vol. 157, no. 1, pp. 69–93, 2016.

[56] V. Gupta and I. E. Grossmann, "A new decomposition algorithm for multistage stochastic programs with endogenous uncertainties," *Computers & Chemical Engineering*, vol. 62, pp. 62–79, 2014.

[57] B. Christian and S. Cremaschi, "Heuristic solution approaches to the pharmaceutical r&d pipeline management problem," *Computers & Chemical Engineering*, vol. 74, pp. 34–47, 2015.

[58] R. M. Apap and I. E. Grossmann, "Models and computational strategies for multistage stochastic programming under endogenous and exogenous uncertainties," *Computers & Chemical Engineering*, vol. 103, pp. 233–274, 2017.

[59] N. Berktaş and H. Yaman, "A branch-and-bound algorithm for team formation on social networks," *INFORMS Journal on Computing*, 2020.

[60] P. Jaccard, "The distribution of the flora in the alpine zone. 1," *New phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[61] R. Fortet and E. Mourier, "Les fonctions aléatoires comme éléments aléatoires dans un espace de banach," *J. Math. Pures Appl*, vol. 38, no. 9, pp. 347–364, 1959.

[62] M. S. Bazaraa and J. J. Goode, "A cutting-plane algorithm for the quadratic set-covering problem," *Operations Research*, vol. 23, no. 1, pp. 150–158, 1975.

[63] B. Escoffier and P. L. Hammer, "Approximation of the quadratic set covering problem," *Discrete Optimization*, vol. 4, no. 3-4, pp. 378–386, 2007.

[64] R. Saxena and S. Arora, "A linearization technique for solving the quadratic set covering problem," *Optimization*, vol. 39, no. 1, pp. 33–42, 1997.

[65] P. Pandey and A. P. Punnen, "On a linearization technique for solving the quadratic set covering problem and variations," *Optimization Letters*, vol. 11, no. 7, pp. 1357–1370, 2017.

[66] 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*, vol. 112, p. 104769, 2019.

[67] 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*, vol. 176, no. 2, pp. 657–690, 2007.

[68] W. D. Pisinger, A. B. Rasmussen, and R. Sandvik, "Solution of large quadratic knapsack problems through aggressive reduction," *INFORMS Journal on Computing*, vol. 19, no. 2, pp. 280–290, 2007.

[69] J. Povh and F. Rendl, "Copositive and semidefinite relaxations of the quadratic assignment problem," *Discrete Optimization*, vol. 6, no. 3, pp. 231–241, 2009.

[70] H. Mittelmann and J. Peng, "Estimating bounds for quadratic assignment problems associated with hamming and manhattan distance matrices based on semidefinite programming," *SIAM Journal on Optimization*, vol. 20, no. 6, pp. 3408–3426, 2010.

[71] 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*, vol. 27, no. 2, pp. 378–391, 2015.

[72] 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*, vol. 280, no. 1, pp. 46–58, 2020.

[73] 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*, vol. 22, no. 4, pp. 1449–1468, 2012.

[74] R. Martinelli and C. Contardo, "Exact and heuristic algorithms for capacitated vehicle routing problems with quadratic costs structure," *INFORMS Journal on Computing*, vol. 27, no. 4, pp. 658–676, 2015.

[75] D. Bergman, "An exact algorithm for the quadratic multiknapsack problem with an application to event seating," *INFORMS Journal on Computing*, vol. 31, no. 3, pp. 477–492, 2019.

[76] W. P. Adams and H. D. Sherali, "A tight linearization and an algorithm for zero-one quadratic programming problems," *Management Science*, vol. 32, no. 10, pp. 1274–1290, 1986.

[77] 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*, vol. 180, no. 3, pp. 983–996, 2007.

[78] 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*, vol. 24, no. 2, pp. 202–209, 2012.

[79] A. Billionnet and F. Calmels, "Linear programming for the 0–1 quadratic knapsack problem," *European Journal of Operational Research*, vol. 92, no. 2, pp. 310–325, 1996.

[80] A. Caprara, D. Pisinger, and P. Toth, "Exact solution of the quadratic knapsack problem," *INFORMS Journal on Computing*, vol. 11, no. 2, pp. 125–137, 1999.

[81] F. D. Fomeni, K. Kaparis, and A. N. Letchford, "A cut-and-branch algorithm for the quadratic knapsack problem," tech. rep., Lancaster University Management School, UK, 2014.

[82] 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*, pp. 839–848, ACM, 2012.

[83] 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*, pp. 389–404, Springer, 2015.

[84] N. Berktaş, N. Noyan, and H. Yaman, "Stochastic team formation problem." In preparation.

[85] J. R. Birge, "The value of the stochastic solution in stochastic linear programs with fixed recourse," *Mathematical programming*, vol. 24, no. 1, pp. 314–325, 1982.

[86] J. Benders, "Partitioning procedures for solving mixed-variable programming problems, numerische matkematic 4," 1962.

[87] R. M. Van Slyke and R. Wets, "L-shaped linear programs with applications to optimal control and stochastic programming," *SIAM Journal on Applied Mathematics*, vol. 17, no. 4, pp. 638–663, 1969.

[88] G. Laporte and F. V. Louveaux, "The integer l-shaped method for stochastic integer programs with complete recourse," *Operations research letters*, vol. 13, no. 3, pp. 133–142, 1993.

[89] D. Gade, S. Küçükyavuz, and S. Sen, "Decomposition algorithms with parametric gomory cuts for two-stage stochastic integer programs," *Mathematical Programming*, vol. 144, no. 1-2, pp. 39–64, 2014.

[90] M. Zhang and S. Kucukyavuz, "Finitely convergent decomposition algorithms for two-stage stochastic pure integer programs," *SIAM Journal on Optimization*, vol. 24, no. 4, pp. 1933–1951, 2014.

[91] S. Sen and J. L. Higle, "The c 3 theorem and a d 2 algorithm for large scale stochastic mixed-integer programming: Set convexification," *Mathematical Programming*, vol. 104, no. 1, pp. 1–20, 2005.

[92] Y. Qi and S. Sen, "The ancestral benders' cutting plane algorithm with multi-term disjunctions for mixed-integer recourse decisions in stochastic programming," *Mathematical Programming*, vol. 161, no. 1-2, pp. 193–235, 2017.

[93] G. Angulo, S. Ahmed, and S. S. Dey, "Improving the integer l-shaped method," *INFORMS Journal on Computing*, vol. 28, no. 3, pp. 483–499, 2016.

[94] N. Berktaş and I. E. Grossmann, "Multi-stage stochastic project team formation." In preparation.

[95] A. Neebe and M. Rao, "An algorithm for the fixed-charge assigning users to sources problem," *Journal of the Operational Research Society*, vol. 34, no. 11, pp. 1107–1113, 1983.

[96] K. Holmberg, M. Rönnqvist, and D. Yuan, "An exact algorithm for the capacitated facility location problems with single sourcing," *European Journal of Operational Research*, vol. 113, no. 3, pp. 544–559, 1999.

[97] R. Freling, H. E. Romeijn, D. R. Morales, and A. P. Wagelmans, "A branch-and-price algorithm for the multiperiod single-sourcing problem," *Operations research*, vol. 51, no. 6, pp. 922–939, 2003.

[98] G. T. Ross and R. M. Soland, "A branch and bound algorithm for the generalized assignment problem," *Mathematical programming*, vol. 8, no. 1, pp. 91–103, 1975.

[99] O. E. Kundakcioglu and S. Alizamir, *Generalized assignment problem*, pp. 1153–1162. Boston, MA: Springer US, 2009.

[100] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[101] M. V. Pereira and L. M. Pinto, "Multi-stage stochastic optimization applied to energy planning," *Mathematical programming*, vol. 52, no. 1-3, pp. 359–375, 1991.

[102] J. Zou, S. Ahmed, and X. A. Sun, "Stochastic dual dynamic integer programming," *Mathematical Programming*, vol. 175, no. 1-2, pp. 461–502, 2019.

[103] C. C. Carøe and R. Schultz, "Dual decomposition in stochastic integer programming," *Operations Research Letters*, vol. 24, no. 1-2, pp. 37–45, 1999.

[104] B. Sandıkçı and O. Y. Ozaltın, "A scalable bounding method for multi-stage stochastic programs," *SIAM Journal on Optimization*, vol. 27, no. 3, pp. 1772–1800, 2017.

[105] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees: Part ii," *Mathematical programming*, vol. 1, no. 1, pp. 6–25, 1971.

[106] M. Guignard and S. Kim, "Lagrangean decomposition: A model yielding stronger lagrangean bounds," *Mathematical programming*, vol. 39, no. 2, pp. 215–228, 1987.

[107] M. Held, P. Wolfe, and H. P. Crowder, "Validation of subgradient optimization," *Mathematical programming*, vol. 6, no. 1, pp. 62–88, 1974.

[108] M. L. Fisher, "An applications oriented guide to lagrangian relaxation," *Interfaces*, vol. 15, no. 2, pp. 10–21, 1985.

[109] S. Mouret, I. E. Grossmann, and P. Pestiaux, "A new lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling," *Computers & Chemical Engineering*, vol. 35, no. 12, pp. 2750–2766, 2011.

[110] F. Oliveira, V. Gupta, S. Hamacher, and I. E. Grossmann, "A lagrangean decomposition approach for oil supply chain investment planning under uncertainty with risk considerations," *Computers & Chemical Engineering*, vol. 50, pp. 184–195, 2013.