8-2017

TWO MULTI-OBJECTIVE STOCHASTIC MODELS FOR PROJECT TEAM FORMATION UNDER UNCERTAINTY IN TIME REQUIREMENTS

Fahimeh Rahmanniyay
University of Tennessee, Knoxville, frahmann@vols.utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by Fahimeh Rahmanniyay entitled "TWO MULTI-OBJECTIVE STOCHASTIC MODELS FOR PROJECT TEAM FORMATION UNDER UNCERTAINTY IN TIME REQUIREMENTS." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Industrial Engineering.

Andrew J. Yu, Major Professor

We have read this thesis and recommend its acceptance:

James L. Simonton, Janice Tolk

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

TWO MULTI-OBJECTIVE STOCHASTIC MODELS FOR PROJECT TEAM FORMATION UNDER UNCERTAINTY IN TIME REQUIREMENTS

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Fahimeh Rahmanniyay

August 2017

ACKNOWLEDGEMENTS

Thanks to Dr. Yu for his support

And

To my lovely husband, Mahdi,

For his endless love, support and encouragement

Thanks to Javad Seif for his contribution in coding of Model 2

And

Thanks to UTSI staff for their kind support and help

ABSTRACT

Team formation is one of the key stages in project management. The cost associated with the individuals who form a team and the quality of the tasks completed by the team are two of the main concerns in team formation problems. In this study, two mathematical models to optimize simultaneously cost and quality in a team formation problem are developed. Because team formation problem arises in uncertain environment, different scenarios are defined for the time requirement of the project. Two-stage stochastic programming and multi-stage stochastic programming are applied to solve the first and the second model respectively. The presented models and their solution methodology can be applied in different types of projects. In this study, a project that involves an overhaul of an aircraft is presented as a case study in which the goals are to minimize staffing costs and maximize the reliability of the aircraft by staffing workforce with high competency.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter One

INTRODUCTION AND GENERAL INFORMATION

Projects typically have a wide variety of goals and involve many internal and external actors in different activity sectors. Project management is an approach for operation with a single predetermined final product (Beaudry& Dionne, 1989). The project management approach should be flexible and effective for achieving a specific output while respecting the limitations on budget, schedule and quality. Project managers must always take into account the trade-off among the project completion time, the project cost and the project performance (Gagnon et al., 2012). A project team is typically formed with members from different fields and groups with different skills that work together to accomplish a project (Wang and Zhang, 2015; Tavana et al., 2013; Shipley and Johnson, 2009; Corgnet, 2010; Tseng et al., 2004). Personnel selection for a project team is a challenging problem for most organizations because it involves the evaluation of different criteria. Therefore, it can be considered as a multi-criteria decision making (MCDM) problem. Selecting appropriate persons from a pool of candidates has a significant impact on the success of a project (Pitchai et al., 2016). The selection process must satisfy the assignment of candidates to the appropriate roles. It also needs to assure that the qualities of each candidate are optimally matched with the team positions that have different functional requirements (Wi et al., 2009; Agustín-Blas et al., 2011; Zhang and Zhang, 2013; Dorn et al., 2011; Boon and Sierksma, 2003).

If the team members of a project do not have required competencies, the performance and the quality of the project can be jeopardized. (Snyder 2014). Competency is the knowledge, skills, ability and attitude (KSAA) that an individual needs to complete specific job-related tasks successfully (Liu, Ruan et al. 2005). Individual's KSAA is obtained and developed through education, training, and on-the- job experience. In the personnel selection process, organizations and projects have a set of competencies associated with the tasks that are directly related to the job. An individual possesses the KSAAs enable

them to perform the desired task in an acceptable level of competency. In this study, the quality of the project is measured in terms of the competency of team members. Competency based on NASA 2009 competency model generally includes personal effectiveness competencies, academic competencies, workplace competencies, industry-wide technical competencies, industry-sector technical competencies and occupational-specific requirement competencies. NASA 2009 competency model is shown in Figure 1-1. The arrangement of the tiers in a pyramidal shape is not meant to be hierarchical, or to imply that competencies at the top are at a higher level of skill. The model's shape represents the increasing specialization and specificity in the application of skills as you move up the tiers. Tiers 1 through 3 contain Foundation Competencies, which form the foundation needed to be ready to enter the workplace. Tiers 4 and 5 contain Industry Competencies, which are specific to an industry or industry sector. Tires 6 to 9 include occupation competencies.

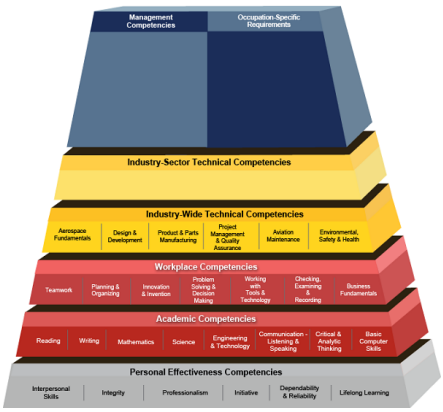

Figure 1-1: NASA Competency based model

In competency-based selection procedure, the critical competencies associated with performing a job are defined and the competency of job candidates is evaluated and determined if they match for the job, they are applying [Competency-Based Employee Selection].

Also, one of the most related operational expenses in the projects is personal hiring, wage and training cost(Maurer 2010); therefore, companies try to use these resources appropriately. As a result, team formation problem has attracted many researchers. However, formation of an effective and successful team with conflicting objectives is still an open problem (Cheatham and Cleereman, 2006; Tavana et al., 2013; Pitchai et al., 2016).

Through reviewing the related research on the team formation problem, it can be observed that this problem has been studied in different aspects such as the competency of the team members (Baykasoglu et al., 2007; Strnad and Guid, 2010; Zhang and Zhang, 2013), the communication among the team members (Zhang and Zhang, 2013; Wi et al, 2009) and the psychological factors (Corgnet, 2010; Tseng et al., 2004).

In this research, I develop two stochastic mathematical modeling for a team formation problem in multi-disciplinary projects to optimize cost and competency simultaneously.

Chapter Two

LITERATURE REVIEW

Developing the project team problem has been intensely studied in various domains. A review of the relevant literature is presented.

Zzkarian and Kusiak (1999) were the first to present an analytical model to build multi-functional teams in the domain of concurrent engineering. The methodology was based on the Analytical Hierarchy Process (AHP) approach and the Quality Function Deployment (QFD) method. A QFD planning matrix was used to organize the factors considered in the team selection. The importance measure for each team member was determined with the AHP approach. A mathematical programming model was developed to determine the composition of a team. The methodology developed in that paper was tested by the selection of teams in concurrent engineering. Then, Chen and Lin (2004) applied a similar approach and developed an AHP for a team formation problem. They considered three fundamental description for the team members. The first was multi-functional knowledge of the team members, the second was teamwork capability of the team member by considering their experience and communication skills, and the third was flexibility in job assignments. Karsak (2000) developed a fuzzy multi-objective linear mathematical modeling to select individuals from a pool of candidates for a certain job. Fuzzy numbers were used to describe candidate skills. The proposed method integrated the decision-maker's linguistic assessments about subjective factors such as excellence in oral communication skills, personality, leadership, and quantitative factors such as aptitude test score within the multiple objective programming framework. The importance degree of each objective was considered by applying the composition operator to the objective's membership function and the membership function corresponding to its fuzzy priority defined by linguistic variables.

Tsai, Moskowitz et al. (2003) presented a model in which candidates and tasks were

respectively set as controllable and uncontrollable factors. Taguchi's parameter design was used to identify selected candidates, which produced robust performance and reduced total project cost and time. In this model, three levels (optimistic, rational and pessimistic) for the candidate ability for each task type were defined before applying Taguchi's parameter design. Tseng, Huang et al. (2004) suggested a group formation model using fuzzy set theory and gray theory to develop a methodology for team formation in multi-functional projects. Fuzzy sets theory was applied to deal with problems involving ambiguities, which are normally confronted in multi-functional teams formation practice and form groups, when there is no clear boundary for relationship between customers' requirements and project characteristics. Grey decision theory was also used to select desired team members through abstractural information. Specifically, the team member was required to be competent in his/her work and also able to share other's responsibility.

Baykasoglu, Dereli et al. (2007) developed an analytical model for the project team selection problem by considering several human and nonhuman factors. Because of the imprecise nature of the problem, fuzzy concepts like triangular fuzzy numbers and linguistic variables were used. The proposed model was a fuzzy multiple objective optimization model with fuzzy objectives and crisp constraints. The skill suitability of each team candidate was reflected to the model by suitability values. These values were obtained by using the fuzzy ratings method. The suitability values of the candidates and the size of the each project team were modeled as fuzzy objectives. The proposed algorithm considered the time and the budget limitations of each project and interpersonal relations between the team candidates. These issues were modeled as hard-crisp constraints. The proposed model used fuzzy objectives and crisp constraints to select the most suitable team members to form the best possible team for a given project. Simulated annealing procedure was used to obtain optimal solution and the Zimmerman's max–min method was used to select the preferred solution among other several local optima solution.

Wi, Oh et al. (2009) presented a framework for analyzing the knowledge of the candidates in a project team and their collaboration ability for managers and team

members. They proposed a nonlinear model and used a weighted method to combine two objectives; then, a genetic algorithm and social network measures were applied to choose the team manager and team members. However, the cost and number of people who are needed for the project is not considered. Feng, Jiang et al. (2010) suggested a multi-objective integer programming model for member selection of cross functional team with respected to the individual capabilities of candidates and the collaborative performance on a pair of candidates. In order to select the desired members, firstly, a multi-objective 0–1 programming model was built using the individual and collaborative performances, which was an NP-hard problem. To solve the model, an improved non-dominated sorting genetic algorithm II (INSGA-II) was developed.

Strnad and Guid (2010) presented a new fuzzy-genetic analytical model for the project team formation. They used fuzzy description to express the required team capabilities. It built on previous quantitive approaches, but added several modeling enhancements like derivation of personnel attributes from dynamic quantitive data, complex attribute modeling, and handling of necessary over competency. A single compound objective function, which incorporated multiple opposing criteria was defined and in order to select multiple project teams with possibly conflicting requirements a special adaptation of island genetic algorithm with mixed crossover was proposed.

Zhang and Zhang (2013) proposed a multi-objective nonlinear model for new product development projects that considered capabilities of all members and relationship of each pair of candidates. A Fuzzy Analytic Hierarchy Process based on fuzzy linguistic preference relations was applied and the Multi- Objective Particle Swarm Optimization (MOPSO) algorithm was implemented to search for Pareto solutions. The cost and capability of each skill is not considered; only the overall capability required for the project is considered. Tavana, Azizi et al. (2013) presented a two-phase frame work for player selection in multi-player sports. The first phase assessed the players with fuzzy ranking method and selected the top performers; then, at the second phase evaluated the combination of the selected players with a Fuzzy Interface System and selected the best

combination for the team formation. Gutiérrez, Astudillo et al. (2016) proposed a mathematical model for the Multiple Team Formation Problem (MTFP) to maximize the efficiency understood as the number of positive interpersonal relationships among people who share a multidisciplinary work cell.

According to the literature review, most of the existing research presents analytical models and focuses on the individual performance of candidates for the project, while it seldom considers the individual performance for each role/task in the member selection process. Also, the human resource cost and the required time that is needed to accomplish a task are not considered. To the best of our knowledge, there is no study in the literature that applied stochastic programming into the team formation problem. But, in the real-world problems, the exact modeling of many situations may not be possible due to the different types of perturbation in the business environment. Therefore, adopting a stochastic optimization approach in the decision-making process is inevitable to obtain a robust decision.

The innovation of this research is developing two mathematical modeling for the team formation problems. The models maximize the competency of each candidate for each work discipline, and minimize the accommodation cost, the wage cost as well as the idle cost of selected candidates simultaneously with the augmented epsilon-constraint method (Mavrotas, 2009). To make the model more practical, different scenarios are considered for the required amount of time that is needed to accomplish a task. The first mathematical modeling is for the project with different work unit that work independently from each other each work unit has different work disciplines. The second mathematical modeling is for the project with different work units that are worked independently from each other and the work assigned to the work units is mutually exclusive in terms of time period and the work of each work unit should be finished in a sequence of time and each work unit has some work disciplines.

Chapter Three

MATERIALS AND METHODS

The innovation of this research is developing two mathematical modeling for the team formation problems. The models maximize the competency of each candidate for each work discipline, and minimize the accommodation cost, the wage cost as well as the idle cost of selected candidates simultaneously with the augmented epsilon-constraint method (Mavrotas, 2009). To make the model more practical, different scenarios are considered for the required amount of time that is needed to accomplish a task. The first mathematical modeling is for the project with different work unit that work independently from each other each work unit has different work disciplines. The second mathematical modeling is for the project with different work units that are worked independently from each other and the work assigned to the work units is mutually exclusive in terms of time period and the work of each work unit should be finished in a sequence of time and each work unit has some work disciplines.

3.1. Two-stage stochastic modeling for a team formation problem to optimize cost and competency

3.1.1. Problem definition

This paper investigates a multi-objective stochastic team formation problem for a project- oriented organization based on the competency of candidates and cost. To make this research applicable to commercial projects, the following assumptions are made.

- There are *U* work units. Work unit refers to a place where work is preformed (Birge 1982) such as shop floors where manufacturing work is performed or offices where office work is performed. Work units are independent from each other.
- Each work unit has *W* work disciplines (WDs).

- There are *N* candidates, and some of them should be assigned to the project.
- Each individual can be assigned to more than one WDs.
- Each selected candidate can work in a particular WD in a unit with only one competency level.
- The decision maker (DM) evaluates the competency of each candidate for each WD.
- Different scenarios are defined to estimate the required amount of time for accomplishing each WD in each unit.
- The Wage of each candidate for each WD is directly related to his or her competency in that WD.
- The time that selected candidate needs to accomplish a task is indirectly related to his or her competency.
- The amount of time that each individual spends in each WD in each scenario should be more than the minimum required for that WD.
- Total time that selected candidate spends during the project should be less than his or her available time.

3.1.1.1. Linguistic Parameter Definition

In practice, natural language is more often applied in the decision process that crisp values cause imprecision and cannot reflect the expert judgments. Therefore, linguistic terms are used for assessment of competency. Five-levels scale (Greatly exceeds expectations, Exceeds Expectations, Meets Expectations, Occasionally Meets Expectations and Unsatisfactory) is used to evaluate competencies. The DM scores personal effectiveness competencies, academic competencies, workplace competencies, industry-wide technical competencies, industry-sector technical competencies and occupational-specific requirement competencies for each candidate for each WD from 5 to 1 based on their competency levels on that WD as shown in Table 3.1. Then, the total score for each candidate for each WD is obtained by the average of competency scores in all mentioned nine fields.

Table 3.1: Competency level

Competency Linguistic Parameter	Score
Greatly exceeds expectations	5
Exceeds Expectations	4
Meets Expectations	3
Occasionally Meets Expectations	2
Unsatisfactory	0

3.1.1.2. Model Formulation

The following notations are used in the model formulation.

Sets:

I	Set of all available workforces for the project
U	Sets of all units
W	Sets of all work disciplines
S	Sets of all possible scenarios

Parameters:

T_{uw}^{s}	Required time for work discipline *w* in unit *u* in scenario *s*
b_{uw}	Minimum percentage time that each candidate who is selected for work discipline *w* in unit *u* in scenario *s* should spend
H_i^0	Available time of candidate *i* at the beginning of planning horizon
E_{iuw}	competency score of candidate *i* in work unit *u* in WD *w*
C	Accommodation cost of working for the project
r_{iuw}	Cost of working of candidate *i* in work unit *u* in WD *w*
V_i	The unit cost for idle time of selected candidate *i*
P^s	Probability of scenario *s*
α_{iuw}	The rate of completing a task in work unit *u* in WD *w* that assigned to candidate *i*

Variables:

X_{iuw}	1 if candidate *i* is selected for work discipline *w* in unit *u*, 0 otherwise
Y_i	1 if candidate *i* is selected for the project
Id_i^s	The idle time of selected candidate at the end of time horizon in scenario *s*
t_{iuw}^s	The time that candidate *i* work in work discipline *w* in unit *u* in scenario *s*

In terms of the above-mentioned notations, the multi-objective stochastic team formation problem can be formulated as follows.

Model 1-1

$$MinZ_1 = \sum_{i=1}^{I} CY_i + (\sum_{s=1}^{S}\sum_{u=1}^{U}\sum_{w=1}^{W}\sum_{i=1}^{I} P^s r_{iuw} t_{iuw}^s) + \sum_{s=1}^{S}\sum_{i=1}^{I}(P^s Id_i^s V_i) \quad (1-1)$$

$$Max\ Z2 = \sum_{s=1}^{S}\sum_{u=1}^{U}\sum_{w=1}^{W}\sum_{i=1}^{I} P^s (E_{iuw} t_{iuw}^s \alpha_{iuw} / T_{uw}^s) \quad (1-2)$$

$$Y_i \geq X_{iuw} \quad \forall i,u,w \quad (1-3)$$

$$\sum_{u=1}^{U}\sum_{w=1}^{W} X_{iuw} \geq Y_i \quad \forall i \quad (1-4)$$

$$\sum_{i=1}^{I} t_{iuwk}^s \alpha_k = T_{uw}^s \quad \forall u,w,s \quad (1-5)$$

$$t_{iuwk}^s \geq b_{uw}\, T_{uw}^s X_{iuw} \quad \forall s,i,u,w \quad (1-6)$$

$$t_{iuwk}^s \leq M X_{iuw} \quad \forall s,i,u,w,k \quad (1-7)$$

$$\sum_{u=1}^{U}\sum_{w=1}^{W} t_{iuwk}^s \leq h_i Y_i \quad \forall i \quad (1-8)$$

$$h_i Y_i - \sum_{u=1}^{U}\sum_{w=1}^{W} t_{iuwk}^s = Id_i^s \quad \forall i,s \quad (1-9)$$

$$Y_i, X_{iuw} \in \{0,1\} \quad \forall i,u,w \qquad t_{iuw}^s, Id_i^s \geq 0 \quad \forall i,u,w,s \quad (1-10)$$

Equation 1 and 2 imply the elements of objective functions. Equation 1 is to minimize the total cost including accommodation cost (training, hiring etc.) of selected candidates, the wage of the selected candidates based on their competency and the cost of idle time of

the selected candidates during the project. Equation 2 is to maximize the total competency of selected candidates. Equations 3 restricts WD assignment to the candidate who is selected for the project. Equation 4 ensures that if a candidate is selected, he or she should be assigned to at least one WD. Equation 5 ensures the fulfillment of required time in scenario s based on competency level. Equation 6 implies that the allocated time for each selected candidate for a specific WD should be more than the minimum required for that WD in scenario s. Equation7 ensures the time assignment to each WD in each unit to the selected candidate for that specific WD in that unit in scenario s. Equation8 implies that the total time that each selected candidate works in all WD in all units should be less than or equal to his or her availability. Equation 9 calculates the idle time of each selected candidate at the end of the project under scenario s. Equation 10 declares variables.

3.1.2. Solution methodology

We have two challenges to solve the model. First, the model is stochastic and the second the model has more than one objective; so, we cannot obtain a single solution. For the first challenge, two-stage stochastic programming is used and for the second challenge the augmented epsilon constraint method is used.

3.1.2.1. Stochastic optimization methodology

Stochastic optimization is currently one of the most robust tools for decision-making that is used to handle uncertainty. In real life, the exact value of many parameters are unknown and the exact modeling of many situations may not be possible. Therefore, stochastic programming is extensively used in in a wide range of problems such as supply chain, finance, production planning, energy, etc. (Körpeoğlu, Yaman et al. 2011, Ramezani, Bashiri et al. 2013, Valladão, Veiga et al. 2014, Cobuloglu and Büyüktahtakın 2017) to make a robust decision. Stochastic programming methods is used whenever the probability of the distribution of the input data are known or can be estimated. Therefore, the results of the decisions taken at present time are not known until the unknown data is revealed (Housh, Ostfeld et al. 2013).

Two-stage stochastic programs are the most extensively used version of stochastic programs. In a two-stage approach, the DM takes some decision in the first stage, before the realization of the uncertainty. After uncertainty is realized a recourse decision can be made in the second stage to compensate any possible negative effect of the decision that have been made in the first-stage.

A standard formulation of two-stage stochastic linear program is as follows:

$Min_x C^T x + E[Q(x, \varepsilon(w))]$

St:

$Ax=b \quad x \geq 0$

where $Q(x,\varepsilon(\omega))$ is the optimal value of the second-stage problem

$Min_x q^T y$

St:

$Tx+Wy=h \quad y \geq 0$

where *x* and *y* are the first and second stage variables, respectively. The second stage problem depends on data (*q, h, T, W*) where any or all elements can be random.

In the presented model, the decision variables specifying the selected candidates for the project and the assignment of candidates to each WD, namely those binary variables are considered as the first- stage variables. The second stage variables are the continues variables related to the amount of time that is assigned to each selected candidate and the idle time of each selected candidate, which can be made after the realization of uncertain parameters.

3.1.2.2. Multi-objective Methodology

There are three methods for solving multi-objective problems. They are prior methods, interactive methods, and posterior methods. These methods are classified based on the phase in which a DM is involved in the decision-making process (Hwang and Masud 2012). In prior methods, the DM expresses his or her preferences prior to the solution process such

as setting goals or weights to the objective function. The criticism of this method is its difficulty for the DM to quantify his or her preference accurately before knowing the values of the objectives. In the interactive methods, the DM progressively conveys the search into the most preferred solution. The disadvantage of this method is that the DM never sees the whole picture (Pareto set) of the solution. In the posterior methods, efficient solutions of the problems are generated and then the DM selects the most preferred solution among all of the solutions. The drawback of this method is its difficulty in computation because the calculation of the efficient solution is usually a time-consuming process (Mavrotas 2009). In this paper, the Augmented Epsilon-Constraint method (Marvotas, 2009), which is the improved version of the conventional epsilon-constraint method is applied. The efficiency of the obtained solutions is guaranteed by the improved version as it only produces the efficient solutions, but not the conventional one (Marvotas, 2009).

In the augmented epsilon constraint method, the most important objective function (the first objective in this paper) is optimized while the other objectives (here the second) are added to the constraints as follows:

Model G (General Augmented Epsilon Constraint method):

$$\text{Max } \left(-f_1(x,t) + eps^{\,s_2}/_{r_2}\right) \qquad \text{(G-1)}$$

Subject to:

$$f_2(x,t) - s_2 = e_2 \qquad \text{(G-2)}$$

$$x, t \ \in S \qquad \text{(G-3)}$$

where r_2 is the range of the second objective function, *eps* is a very small number (10^{-3} to 10^{-9}) and s_2 is the slack variable.

The range of e_2 can be calculated by the payoff table that is obtained from the lexicographic table subjected to the feasible set *S*.

Then, by dividing the range of constrained objectives ($f_2(x,t)$) to q equal intervals, different values for e_2 can be calculated as follows:

$$r_2 = f_2^{max}(x,t) - f_2^{min}(x,t) \qquad (\mathrm{G}-4)$$

$$e_2^k = f_2^{min}(x,t) + {r_2}/{q} \quad for\ k = 0, \dots, q-1 \qquad (G-5)$$

After solving Model 1 for each grid of (e_2^i), a Pareto set is constructed.

After finding the grid points (e_2^k) from the pay-off table, the single objective stochastic model is presented for each grid point (Model 2).

Model 2:

$$MinZ^k = \sum_{i=1}^{I} CY_i + (\sum_{s=1}^{S}\sum_{u=1}^{U}\sum_{w=1}^{W}\sum_{i=1}^{I} P^s r_{iuw} t_{iuw}^s + \sum_{s=1}^{S}\sum_{i=1}^{I}(P^s Id_i^s V_i) + eps\, {s_2^k}/{r_2} \qquad (1-16)$$

$$\sum_{u=1}^{U}\sum_{w=1}^{W} X_{iuw} \geq Y_i \qquad \forall i \qquad (1-17)$$

$$\sum_{i=1}^{I} t_{iuwk}^s \alpha_k = T_{uw}^s \qquad \forall u, w, s \qquad (1-18)$$

$$t_{iuwk}^s \geq b_{uw}\, T_{uw}^s X_{iuw} \quad \forall s, i, u, w \qquad (1-19)$$

$$t_{iuwk}^s \leq M X_{iuw} \qquad \forall s, i, u, w, k \qquad (1-20)$$

$$\sum_{u=1}^{U}\sum_{w=1}^{W} t_{iuwk}^s \leq h_i Y_i \qquad \forall i \qquad (1-21)$$

$$h_i Y_i - \sum_{u=1}^{U}\sum_{w=1}^{W} t_{iuwk}^s = Id_i^s \qquad \forall i, s \qquad (1-22)$$

$$\sum_{s=1}^{S}\sum_{u=1}^{U}\sum_{w=1}^{W}\sum_{i=1}^{I} P^s({E_{iuw} t_{iuw}^s \alpha_{iuw}}/{T_{uw}^s}) - s_2^k = e_2^k \quad \forall k \in K \qquad (1-23)$$

$$Y_i, X_{iuw} \in \{0,1\}\ \forall i, u, w \qquad t_{iuw}^s, Id_i^s, s_2^k, e_2^k \geq 0 \quad \forall i, u, w, s, k \qquad (1-24)$$

The presented model is implemented using Gurobi and Python. It takes less than 1 minutes to solve a realistic problem instance with as many candidates as 120. The project team formation problem is not a daily operational problem that needs to be run multiple times a day, so the solution time is not overly concerned.

3.2. Multi-stage stochastic modeling for a team formation problem to optimize cost and competency

When the industry is faced with sequential decisions over time such as team formation during the project period, multi-stage stochastic programming is one of the most appropriate methods to achieve robust decisions in the presence of future uncertainty. Therefore, in the second model, we assume that work units work independently from each other in a sequence of time; for example, Work Unit 2 starts working after finishing the work in Work Unit 1. The innovation of this model is that the competency of each candidate for each work discipline, and the accommodation cost, wage cost based on competency and the idle cost of selected candidates are considered and optimized simultaneously with the Augmented Epsilon-Constraint method (Mavrotas, 2009). To make the model more practical, different scenarios in each stage are considered for the required amount of time that is needed to accomplish a task. Having sufficient scenarios to realistically represent real-world environments often leads to higher computational complexity for the presented problem. Thus, I have adopted Scenario Cluster Decomposition (SCD) methods in my modeling and developed a heuristic algorithm. It is shown that the model can be solved for problems with a practical size.

3.2.1. Problem Definition

This model investigates a multi-objective stochastic team formation problem for a multi-disciplinary project considering competency of candidate and cost. To make this research applicable for commercial projects, the following assumptions are made.

- There are *U* work units. Work unit refers to a place where work is performed such as shop floors where manufacturing work is performed or offices where office work is performed (Wil, Desel et al. 2003). Work units are independent from each other, i.e., each work units can be set up independently from other work units.
- Work assigned to the work units is mutually exclusive in terms of time period.

- Each work unit has *W* work disciplines (WDs).
- There are *N* candidates and some of them should be assigned to the project.
- Each individual can be assigned to more than one WD.
- The competency of each candidate for each WD is identified.
- Different scenarios are defined to estimate the required time for accomplishing each WD in each unit.
- For each WD, a competency profile is defined which describes job-relevant behavior, motivation, and its required skills. Each decision maker (DM) evaluates the competency of each candidate for each WD.
- Three levels of competency are defined. Level 1 is the worst, level 2 is moderate, and level 3 is excellent.
- The time that a selected candidate needs to accomplish a task is related to his or her competency.
- The wage of each candidate for WD is directly proportional to his or her competency in that WD.
- The amount of time that each individual spends in each WD in each scenario should be more than the minimum required for that WD.
- Total time that selected candidate spends during the project should be less than his or her available time

3.2.1.1. Model Formulation

The following notations are used in the model formulation.

Sets:

I	Set of all available workforces for the project
U	Sets of all units
W	Sets of all work disciplines
L	Sets of all competency levels

S	Sets of all possible scenarios
Parameters:	
N	Number of candidates
T_{uw}^{s}	Required time for WD *w* in unit *u* in scenario *s*
b_{uw}	Minimum percentage time that each candidate who is selected for WD *w* in unit *u* in scenario *s* should spend on it
H_i^0	Available time of candidate *i* at the beginning of the project
E_{iwl}	1 if candidate *i* has level *l* in discipline *w*, 0 otherwise
C_i	Fixed cost of training for candidate *i*
r_{wl}	Wage of working in WD *w* at level *l* in each unit time
K_i	The unit cost for idle time of selected candidate *i*
P^s	Probability of scenario *s*
α_l	The rate of completing a task at competency level *l*
Variables:	
Y_i	1 if candidate *i* is selected for the project
$X^s{}_{iuw}$	1 if candidate *i* is selected for WD *w* in unit *u* in scenario *s*, 0 otherwise
h_i^{us}	The availability of candidate *i* at the end of working in unit *u* in scenario *s*
t_{iuwl}^{s}	The time that candidate *i* work in level *l* in WD *w* in unit *u* in scenario *s* (during the time period *u*)
$IDLT_i^s$	The idle time of selected candidate *i* at the end of the last unit in scenario *s*

With the consideration of all the above assumptions, the multi-objective stochastic model can be developed as follows:

***Model* 2.1**:

$$\text{Min } Z_1 = \sum_{i=1}^{I} C_i Y_i + \sum_{s=1}^{S}\sum_{u=1}^{U}\sum_{w=1}^{W}\sum_{i=1}^{I}\sum_{l=1}^{L} r_{wl} t_{iuwl}^{s} P^s + P^s\left(\sum_{i=1}^{I} K_i h_i^U Y_i\right) \quad (2.1\text{-}1)$$

$$\text{Max } Z_2 = \sum_{s=1}^{S}\sum_{u=1}^{U}\sum_{w=1}^{W}\sum_{i=1}^{I}\sum_{l=1}^{L} E_{iwl}\, t_{iuwl}^{s}\, l P^s \quad (2.1\text{-}2)$$

Subject to:

$$Y_i \le \sum_{w=1}^{W} X_{iuw}^{s} \quad \forall i, u, s \quad (2.1\text{-}3)$$

$$Y_i \le X^s{}_{iuw} \quad \forall i, w, u, s \quad (2.1\text{-}4)$$

$$\sum_{l=1}^{L}\sum_{i=1}^{I} t_{iuwl}^{s}\, \alpha_l \ge T_{wu}^{s} \quad \forall u\, s, w \quad (2.1\text{-}5)$$

$$\sum_{l=1}^{L} t_{iuwl}^{s} \geq b_{uw} T_{uw}^{s} X_{iuw}^{s} \quad \forall u, i, s, w \tag{2.1-6}$$

$$\sum_{l=1}^{L} t_{iuwl}^{s} \leq M X_{iuw}^{s} \quad \forall u, i, s, w \tag{2.1-7}$$

$$\sum_{w=1}^{W} \sum_{l=1}^{L} t_{iuwl}^{s} \leq H_i^0 \, Y_i \quad \forall i, s, u \tag{2.1-8}$$

$$h_i^{1s} = H_i^0 - \sum_{w=1}^{W} \sum_{l=1}^{L} t_{i1wl}^{s} \quad \forall i, s \tag{2.1-9}$$

$$h_i^{us} = h_i^{u-1s} - \sum_{w=1}^{W} \sum_{l=1}^{L} t_{iuwl}^{s} \quad \forall i, s, u \geq 2 \tag{2.1-10}$$

$$\sum_{w=1}^{W} \sum_{l=1}^{L} t_{iuwl}^{s} \leq h_i^{u-1s} \quad \forall i, s, u \tag{2.1-11}$$

$$Y_i, X^s{}_{iuw} \in \{0,1\}, t_{iuwl}^{s}, h_i^{us} \geq 0 \tag{2.1-12}$$

Equations 1 and 2 are the two objective functions. Equation 1 is to minimize the total cost including the fixed cost of hiring selected candidates, the wages of selected candidates are based on their competency and cost of their idle times. Equation 2 is to maximize the total competency of selected candidates. Equation 3 ensures that if a candidate is selected, he or she should be assigned to at least one WD in each scenario. Equation 4 restricts WD assignment to the candidate who is selected. Equation 5 ensures the fulfillment of required time based on competency in scenario *s*. Equation 6 implies that the allocated time for each selected candidate for a specific WD should be more than the minimum required for that WD under scenario *s*. Equation 7 ensures the time assignment to each WD in each unit to the candidates selected for that specific WD in that unit under scenario *s*. Equation 8 implies that the total time that each selected candidate works in a unit should be less than his or her availability. Equation 9 calculates the availability of each selected candidate right after completing the work in the first unit. Equation 10 calculates the availability of each selected candidate right after finishing the work in each unit under scenario *s*. Equation 11 shows that the time that each selected candidate spends in all WD in all levels at the u^{th} unit should be less than his or her availability at the end of the previous unit under scenario *s*. Equation 12 declares variables.

The third term in Equation 1 is non-linear. To linearize it, the third term is changed to the following and Equations 1-14 – 1-16 are added to the model.

$P^s \sum_{i=1}^{I} K_i IDLT_i^s$

$$h_i^{Us} \geq IDLT_i^s \quad \forall\, i, s \tag{2.1-14}$$

$$IDLT_i^s \leq MY_i \quad \forall\, i, s \tag{2.1-15}$$

$$IDLT_i^s \geq M(1 - Y_i) + h_i^{Us} \quad \forall\, i, s \tag{2.1-16}$$

3.2.2. Solution Methodology

There exist multiple challenges in solving the presented problem. In this section, these challenges are addressed and appropriate solution methods are presented. The first challenge is dealing with more than one objective. A Multi-Objective method is applied when there is more than one objective function and, in general, when there is no single optimal solution that simultaneously optimizes all of the objective functions.

The next challenge is remodeling the presented problem as a stochastic programming model in which different scenarios are considered. Because the number of scenarios is large in real-world applications, the final challenge is developing a method that can efficiently handle a problem instance with a large number of scenarios.

3.2.2.1. Multi-Objective Methodology

For solving multi-objective, the Augmented epsilon constraint method is used and it was explained in Section 3.2.2.2.

3.2.2.2. Stochastic Optimization Methodology

Stochastic optimization is one of the best tools to provide a robust solution in decision making. It is used to handle uncertainty. In real life, the exact values of many parameters are unknown. Therefore, stochastic programming is extensively applied in real-world applications in a broad range of problems such as supply chain, finance, production planning, energy, etc.(Körpeoğlu, Yaman et al. 2011, Ramezani, Bashiri et al. 2013, Valladão, Veiga et al. 2014, Cobuloglu and Büyüktahtakın 2017). In stochastic programming, the main source of uncertainty is randomness and uncertain parameters are

considered as random variables with a known probability distribution(Housh, Ostfeld et al. 2013).

Two-stage stochastic programs are the most extensively used version of stochastic programs. In a two-stage approach, the plan for the entire multi-period planning horizon is decided before uncertainty is recognized, and only a limited recourse decision can be made afterward. A detailed explanation about stochastic programming, its applications, and solution techniques can be found in (Birge and Louveaux 2011)

Multi-stage Stochastic Programming (MSP) models generally appear in multi-period planning problems under uncertain parameters with dynamic and non-stationary behavior over the planning horizon (Zanjani, Bajgiran et al. 2016). In MSP, the planning decisions are made in several stages instead of two stages. Hence, this approach allows us to correct the decisions when more information regarding the uncertainty is recognized. Therefore, in comparison to two–stage, MSP models provide better results since they incorporate data as they become available. As a result, the MSP model is more appropriate for the dynamic planning process.

MSP places an emphasis on the decisions that must be made here-and-now, given present information, future uncertainties and possible recourse action in the future. The decisions at each stage are made while considering that modification and correction will be possible at later stages (recourse decisions) (Kazemi Zanjani, Nourelfath et al. 2010, Housh, Ostfeld et al. 2013). The MSP solves for an optimal policy that contains the first-stage decision (constant) and the recourse decision (updated based on past realization). The first-stage decision does not depend on observations and can always be implemented on any new scenario. However, the recourse decision at each successive stage relies on information revealed up to this stage. In this problem, each work unit is considered as a stage. For instance, Work Unit 1 is considered as Stage 1. The selected candidates are considered as the first-stage decision and the assignment of selected candidates to each WD in each unit and the time that each selected candidate should spend on each WD in

each unit are considered as recourse decision. A detailed explanation of MSP can be found in Birge and Louveaux (2010).

The size of MSP problems is typically large. Hence, it is hard to solve it by a direct solution technique. The difficulty in solving is because the problem dimension increases exponentially as the number of scenarios and stages increases (Rosa and Ruszczyński 1996). To handle this computational difficulty, decomposition techniques are applied in solution approaches by decomposing the main problem into smaller and easier-to-solve sub-problems. These techniques can be found in (Ruszczynski 1989),(Rockafellar and Wets 1991), (Escudero, Kamesam et al. 1993), (Escudero, Garín et al. 2012) and (Zanjani, Bajgiran et al. 2016).

The method used by Kazemi et al. (2016) for Multi-Stage Stochastic Mixed-Integer problems interests us most. I extended the method and applied it to the proposed Multi-Objective MSP.

Defining scenarios is a common tool to represent the stochastic process (Dupačová, Consigli et al. 2000). The uncertainty is represented through a scenario tree. Such a scenario tree is shown in Figure 1. A scenario tree consists of nodes and stages. Each stage indicates the stage of time when new information is revealed. Each planning horizon might have a number of periods (Kazemi Zanjani, Nourelfath et al. 2010). Each stage encompasses a number of nodes and arcs. Each node presents a possible state of the stochastic process, and the root node represents the current time. Each node has a unique ancestor. The arcs represent the links between the nodes and are associated with the conditional probability. The probability of each node in the scenario tree is calculated by multiplying the probability of that arc from the root node to that node, and the sum of the probabilities of nodes at each stage should be equal to one. A path from the root node to the leaf node (end of the path) represents a scenario (Kazemi et al., 2010). Figure 3-1 represents a four-stage scenario tree. Each node has two branches to the subsequent stage

that demonstrate two possible scenarios for the subsequent stage. As a result, there are eight scenarios at the end of Stage 4.

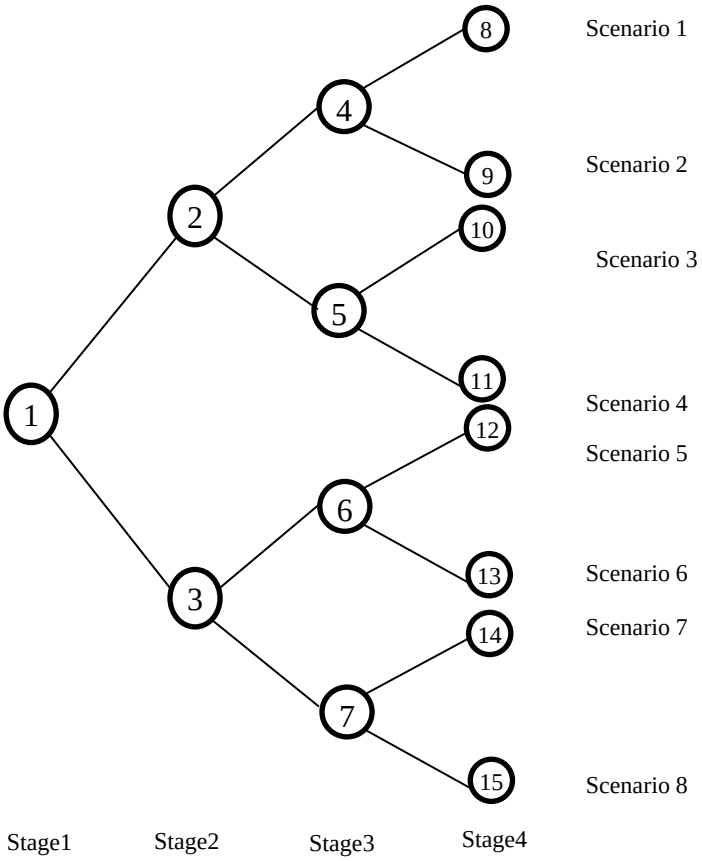

Figure 3-1: A four-stage scenario tree

In this paper, a multi-stage stochastic program for a team formation problem with uncertainty in required work time (work estimation) for each WD in each work unit is proposed. Required time uncertainty originates from the condition of the work unit. For instance, the required time for completing the tasks of each unit based on its condition can be low, moderate or high. We assume that the uncertain required time for each WD in each work unit evolves as a discrete stochastic process during the planning horizon, which forms a scenario tree. Each stage in a scenario tree corresponds to a work unit. Therefore, each

work unit corresponds to a time period. In addition, each stage has a limited number of work time estimation scenarios (low, moderate, high). To maintain the MSP model at a manageable size, we assume that each node contains a number of WDs. This means that uncertain work estimation is stationary for all the WDs included in a node in a particular time period. For example, if a work estimation is low for the first WD at node *n* in its corresponding time period, it should be low for the rest of the WDs at node *n*. To obtain the multi-stage stochastic formulation, each decision variable (x_t) in Model 1 should be considered for each scenario. However, the flow of available information should be conformed by the decision process. This means that the decisions must be non-anticipative or implementable (Kazemi et al., 2010). The non-anticipativity condition (NAC) demonstrates that the decision variables for each node in a scenario tree at Stage *t* take the same value for any pair of indistinguishable scenarios at that stage. For example as shown in Figure 1, Scenarios 1, 2, 3 and 4 at Stage 2 in Node 2 are indistinguishable scenarios.

There are two approaches to impose NAC in MSP leading to split variable and compact variable formulation. In split variable formulation, non-anticipativity is enforced by adding extra constraints explicitly. Even though this method increases the size of the problem, the decomposition approach can be used for splitting variables formulation. The explicit NAC for every pair of scenarios (s and s') that are indistinguishable up to Stage *t* can be expressed as follows:

$$(x_1(s), \dots, x_t(s)) = (x_1(s'), \dots, x_t(s'))$$

In the compact formulation approach, the non-anticipativity is considered in an implicit way and decision variables are associated with the nodes in a scenario tree. In this problem, the split variable approach is used to enforce NAC. There are several decomposition strategies in the literature for solving large-scale multi-stage stochastic programs (Ruszczyński 1997). I found that the Hybrid Scenario Cluster Decomposition (HSCD) algorithm for the mixed integer programming model presented by Kazemi et al.

(2016) is very interesting to us. I adopted the algorithm and modified it for the proposed multi-objective linear programming model.

The procedures that are required to solve the multi-objective MSP team formation problem are summarized in Figure 3-2.

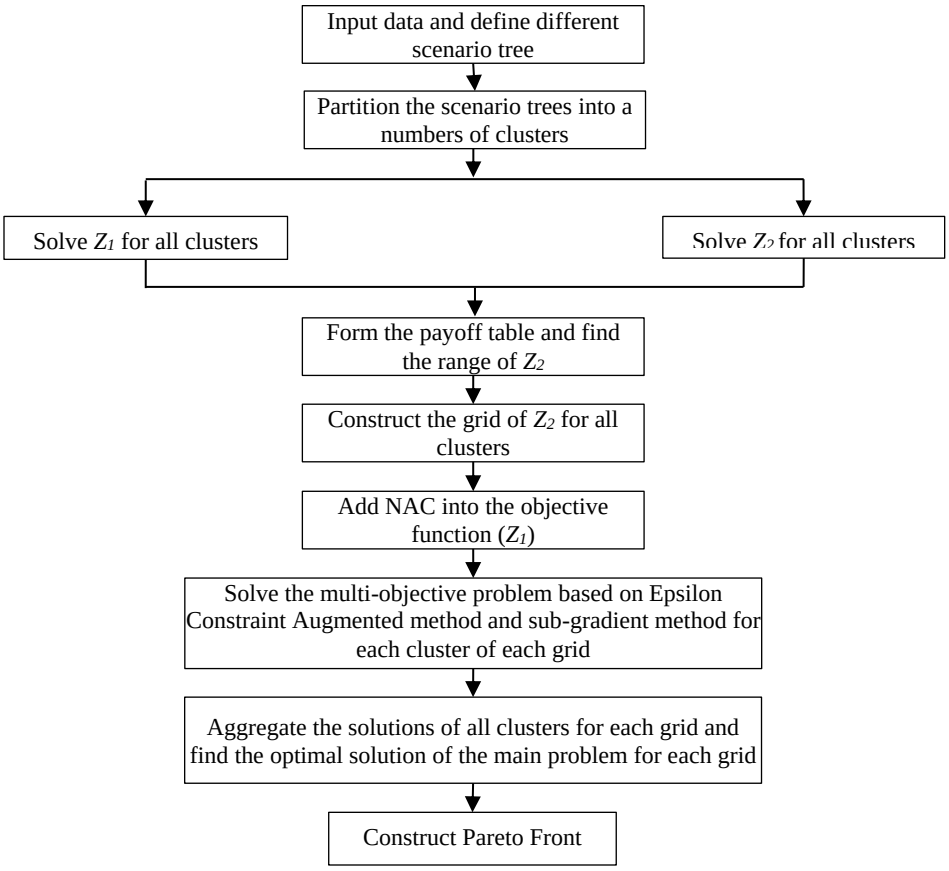

Figure 3-2: The framework of the proposed model

3.2.2.3. Scenario Cluster Decomposition Algorithm

In the SCD algorithm, the original scenario tree is broken down into smaller sub-trees that share a number of predecessor nodes. Then, the multi-stage stochastic model is decomposed into scenario cluster sub models with an addition of a Lagrangian penalty to

its objective function to compensate the lack of NAC (Kazemi et al 2016). In this algorithm, instead of solving all of the scenarios together that causes computational difficulty, each cluster is solved separately, and the solutions of all clusters are aggregated for the final solution.

3.2.2.3.1. Partitioning the scenario tree into scenario cluster sub-trees

The first step in an SCD algorithm is to choose the break stage (Rockfellar and Wets, 1991; Escudero et al., 2012) to breakdown the scenario tree into a set of scenario cluster sub-trees. After choosing the break stage, the scenario tree is decomposed into P scenario cluster sub-trees.

The definitions that are needed in partitioning the scenario and formulation of scenario clusters are described in Table 3.2.

Table 3.2: The Notations for the SCD Algorithm

Notation	Definition
Ω	Set of Scenarios
ω	Specific Scenario
τ^*	Break-stage $\tau^* \in T$
W^{ω}	The probability assigned to ω
P	Set of scenario cluster sub-tree $p \in P$
N^p	Set of nodes that belong to sub-tree $p \in P$
N_1	Set of nodes that belong to stage not after τ^*
N_2	Set of nodes that belong to stage after τ^*
$\overline{N}_{\tau}^{\ p}$	Set of nodes that belong to stage $\tau \in T$ of sub-tree $p \in P$
$\overline{N}_{\tau}$	Set of nodes that belong to stage $\tau \in T$
N_1^p	$N_1 \cap N^p$
N_2^p	$N_2 \cap N^p$
η^n	Set of sub-trees that have node n in common
Ω^{p}	Set of scenarios belonging to sub-tree $p \in P$
$\underline{p}_{\eta^n}$	The first ordered sub-trees belonging to η^n
$\overline{p}_{\eta^n}$	The last ordered sub-trees belonging to η^n

For example, in Figure 3.1, the breaking stage is at $\tau^* = 2$ and there are four scenario cluster sub-trees. Additionally,$N_1 = \{1,2,3\}$, $N_2 = \{4,5, \dots ,15\}$,$N_1^1 = \{1,2\}$ and $N_2^1 = \{4,8,9\}$.

3.2.2.3.2. Scenario cluster sub-model formulation

After breaking down the scenario tree into *P* clusters, converting the multi-objective problem into its equivalent single objective, and finding the pay-off table and e_i (i=1) for each cluster, the model is decomposed into *P* sub-models accordingly. In this problem, selecting candidates is considered as the first-stage decision. The assignment of selected candidates to each WD in each unit and their time allocation are considered as a recourse decision. Therefore, the single objective multi-stage stochastic sub-model for each cluster is presented by the compact formulation. This model is equivalent to the multi-objective model.

Model 2.2:

$$\text{Min } Z_1{}^p = \sum_{i=1}^{n} C_i Y_i + \sum_{n\in N_1^p} \sum_{s=1}^{S} \sum_{w=1}^{W} \sum_{i=1}^{I} \sum_{l=1}^{L} w^p(n) r_{wl} (t_{iwl}^s(n))^p +$$

$$\sum_{n\in N_2^p} \sum_{s=1}^{S} \sum_{w=1}^{W} \sum_{i=1}^{I} \sum_{l=1}^{L} pr(n) r_{wl} (t_{iwl}^s(n))^p + \sum_{n\in \overline{N}_4{}^p} \sum_{i=1}^{n} pr(n) K_i(IDLT_i(n)) +$$

$$\sum_{n\in N^p} eps \times s_2^p(n)/r_2 \qquad (2.2\text{-}1)$$

$$Y_i \le \sum_{w=1}^{W} \left(X^s{}_{iw}(n)\right)^p \quad \forall i,p,n \in N^p, s \qquad (2.2\text{-}2)$$

$$Y_i \le \left(X^s{}_{iw}(n)\right)^p \quad \forall i,w,n \in N^p, s \qquad (2.2\text{-}3)$$

$$\sum_{l=1}^{L} \sum_{i=1}^{I} \left(t_{iwl}^s(n)\right)^p * \alpha_l \ge T_{wu}^s \quad \forall u \in U_n, n \in N^p, s, w \qquad (2.2\text{-}4)$$

$$\sum_{l=1}^{L} \left(t_{iwl}^s(n)\right)^p \ge b_{uw} T_{uw}^s \left(X^s{}_{iw}(n)\right)^p \quad \forall n \in N^p, u \in U_n, i, w, s \qquad (2.2\text{-}5)$$

$$\sum_{l=1}^{L} \left(t_{iwl}^s(n)\right)^p \le M \left(X^s{}_{iw}(n)\right)^p \quad \forall\, i,w,n \in N^p, s \qquad (2.2\text{-}6)$$

$$\left(h_i^s(n)\right)^p = H_i^0 - \sum_{w=1}^{W} \sum_{l=1}^{L} t_{iwl}^s(n) \quad \forall n \in \left(N^p \cap \overline{N}_1\right), i, s \qquad (2.2\text{-}7)$$

$$(h_i^s(n))^p \ge (h_i^s(a(n)))^p - \sum_{w=1}^{W} \sum_{l=1}^{L} (t_{iwl}^s(n))^p \quad \forall n \in N^p, i, s \qquad (2.2\text{-}8)$$

$$\sum_{w=1}^{W} \sum_{l=1}^{L} (t_{iwl}^s(n))^p \le H_i^0 Y_i \qquad \forall\, n \in N^p, i, s \qquad (2.2\text{-}9)$$

$$\sum_{w=1}^{W} \sum_{l=1}^{L} (t_{iwl}^s(n))^p \le h_i^s\left(a(n)\right) \quad n \in N^p, i, s \qquad (2.2\text{-}10)$$

$$(h_i^s(n))^p \ge (Z_i(n))^p \quad \forall n \in \overline{N}_4{}^p, s, i \qquad (2.2\text{-}11)$$

$$(Z_i(n))^p \le MY_i \quad \forall n \in \overline{N}_4{}^p, i, s \tag{2.2-12}$$

$$(Z_i(n))^p \ge M(1-Y_i) + (h_i^s(n))^p \quad \forall n \in \overline{N}_4{}^p, i, s \tag{2.2-13}$$

$$\sum_{s=1}^{S}\sum_{u=1}^{U}\sum_{w=1}^{W}\sum_{i=1}^{I}\sum_{l=1}^{L} E_{iwl}\left(t_{iwl}^s(n)\right)^p l - s_2^p(n) = e_2^{kp}(n) \forall n \in N^p, k \in q \tag{2.2-14}$$

$$(X_{iw}^s(n))^p, Y_i \in \{0,1\}, \left(h_i(n)\right)^p \ge 0, (t_{iwl}^s(n))^p \ge 0, s_2^p(n) \ge 0, IDLT_i(n) \ge 0 \tag{2.2-15}$$

where *n* represents nodes in a scenario tree, *a(n)* is the ancestor of node *n*, *s* is the scenario of work estimation, U_n is a set of work units related to node *n*. The node probability before break stage for each sub-tree p can be expressed as
$w^p(n) = \sum_{\omega \epsilon \Omega^p} w^\omega(n)$, where w^ω is the likelihood of scenario (Escudero et al., 2012).

The sub-problems in Model 2 should be connected to each other by enforcing NAC:

$$\left(h_i(n)\right)^p - \left(h_i(n)\right)^{p'} = 0 \quad \forall i, p, p' \in \eta^n : p \ne p', n \in N_1 \tag{2.2-16}$$

$$(t_{iwl}^s(n))^p - (t_{iwl}^s(n))^{p'} = 0 \quad \forall i, w, l, s, p, p' \in \eta^n : p \ne p', n \in N_1 \tag{2. 2-17}$$

$$(X_{iwl}^s(n))^p - (X_{iwl}^s(n))^{p'} = 0 \quad \forall i, w, l, s, p, p' \in \eta^n : p \ne p', n \in N_1 \tag{2.2-18}$$

Each of the equality constraints in Equations (2.2-16, 2.2-17, 2.2-18) can be converted into two inequality constraints as follows (Kazemi et al., 2016).

$$\left(h_i(n)\right)^p - \left(h_i(n)\right)^{p+1} \le 0 \qquad \forall p = \underline{p}_{\eta^n}, \dots, \overline{p}_{\eta^n}, n \in N_1 \tag{2.2-19}$$

$$(h_i(n))^{\overline{p}_{\eta^n}} - (h_i(n))^{\underline{p}_{\eta^n}} \le 0 \qquad \forall n \in N_1 \tag{2.2-20}$$

$$(t_{iwl}^s(n))^p - (t_{iwl}^s(n))^{p+1} \le 0 \quad \forall p = \underline{p}_{\eta^n}, \dots, \overline{p}_{\eta^n}, n \in N_1 \tag{2.2-21}$$

$$(t_{iwl}^s(n))^{\overline{p}_{\eta^n}} - (t_{iwl}^s(n))^{\underline{p}_{\eta^n}} \le 0 \quad \forall n \in N_1 \tag{2.2-22}$$

$$(X_{iwl}^s(n))^p - (X_{iwl}^s(n))^{p+1} \le 0 \quad \forall p = \underline{p}_{\eta^n}, \dots, \overline{p}_{\eta^n}, n \in N_1 \tag{2.2-23}$$

$$(X_{iwl}^s(n))^{\overline{p}_{\eta^n}} - (X_{iwl}^s(n))^{\underline{p}_{\eta^n}} \le 0 \quad \forall n \in N_1 \tag{2.2-24}$$

Then, the multi-stage stochastic (MSS) model can be formulated by using the splitting variable approach for all the sub-trees (Escudero et al., 2012) of each grid, as shown in Model 3.

Model 3:

$$Min\ Z_{MSSP} = \sum_{i=1}^{I} CY_i + \sum_{p=1}^{P}\sum_{n\in N_1^p}\sum_{s=1}^{S}\sum_{w=1}^{W}\sum_{i=1}^{I}\sum_{l=1}^{L} w^p(n) r_{wl}(t_{iwl}^s(n))^p +$$

$$\sum_{p=1}^{P}\sum_{n\in N_2^p}\sum_{s=1}^{S}\sum_{w=1}^{W}\sum_{i=1}^{I}\sum_{l=1}^{L} pr(n) r_{wl}(t_{iwl}^s(n))^p +$$

$$\sum_{p=1}^{P}\sum_{n\in \overline{N}_4{}^p}\sum_{i=1}^{n} pr(n) K_i (IDLT(n))^p + \sum_{p=1}^{P}\sum_{n\in N^p} eps \times s_2^p(n)/r_2 \qquad (2.3\text{-}1)$$

Subject to:

$$Y_i \le \sum_{w=1}^{W}\left(X^s{}_{iw}(n)\right)^p \quad \forall p\in\eta^n, i, n\in\{N_1, N_2\}, s \qquad (2.3\text{-}2)$$

$$Y_i \le \left(X^s{}_{iw}(n)\right)^p \quad \forall p\in\eta^n, w, n\in\{N_1, N_2\}, s \qquad (2.3\text{-}3)$$

$$\left(h_i(n)\right)^p = H_i^0 - \sum_{w=1}^{W}\sum_{l=1}^{L} t_{iwl}^s(n) \quad \forall p\in\eta^n, n\in\overline{N}_1, i, s \qquad (2.3\text{-}4)$$

$$\left(h_i(n)\right)^p = \left(h_i(a(n))\right)^p - \sum_{w=1}^{W}\sum_{l=1}^{L}(t_{iwl}^s(n))^p \quad \forall p\in\eta^n, n\in\{N_1, N_2\}, i, s \qquad (2.3\text{-}5)$$

$$\sum_{w=1}^{W}\sum_{l=1}^{L}(t_{iwl}^s(n))^p \le H_i^0 Y_i \qquad \forall p\in\eta^n, n\in\{N_1, N_2\}, i, s \qquad (2.3\text{-}6)$$

$$\sum_{w=1}^{W}\sum_{l=1}^{L}(t_{iwl}^s(n))^p \le h_i\left(a(n)\right) \quad \forall p\in\eta^n, n\in\{N_1, N_2\}, i, s \qquad (2.3\text{-}7)$$

$$\sum_{l=1}^{L}\sum_{i=1}^{I}(t_{iwl}^s(n))^p \alpha_l) \ge T_{uw}^s(n) \ \forall\ p\in\eta^n, n\in\{N_1, N_2\}, u\in U_n, w, s \qquad (2.3\text{-}8)$$

$$\sum_{l=1}^{L}(t_{iwl}^s(n))^p \ge b_{uw} T_{uw}^s \left(X^s{}_{iw}(n)\right)^p \ \forall\ p\in\eta^n, n\in\{N_1, N_2\}, u\in U_n, i, w, s \qquad (2.3\text{-}9)$$

$$\sum_{l=1}^{L}(t_{iwl}^s(n))^p) \le M(X^s{}_{iw}(n))^p \ \ \forall\ p\in\eta^n, n\in\{N_1, N_2\}, u\in U_n, i, w, s \qquad (2.3\text{-}10)$$

$$(h_i^s(n))^p \ge (IDLT_i(n))^p \quad \forall\ p\in\eta^n, n\in \overline{N}_4{}^p, s, i \qquad (2.3\text{-}11)$$

$$(IDLT_i(n))^p \le MY_i \quad \forall\ p\in\eta^n, n\in \overline{N}_4{}^p, s, i \qquad (2.3\text{-}12)$$

$$(IDLT_i(n))^p \ge M(1-Y_i) + (h_i^s(n))^p \ \ \forall\ p\in\eta^n, n\in \overline{N}_4{}^p, s, i \qquad (2.3\text{-}13)$$

$$\sum_{s=1}^{S}\sum_{u=1}^{U}\sum_{w=1}^{W}\sum_{i=1}^{I}\sum_{l=1}^{L} E_{iwl}\left(t_{iwl}^s(n)\right)^p l - s_2^p(n) = e_2^{kp}(n) \ \forall\ p\in\eta^n, n\in \{N_1, N_2\}, k\in q \qquad (2.3\text{-}14)$$

$$\left(h_i(n)\right)^p - \left(h_i(n)\right)^{p+1} \le 0 \qquad \forall p = \underline{p}_{\eta^n}, \dots, \overline{p}_{\eta^n}, n\in N_1 \qquad (2.3\text{-}15)$$

$$(h_i(n))^{\overline{p}_{\eta^n}} - (h_i(n))^{\underline{p}_{\eta^n}} \le 0 \qquad \forall p = \underline{p}_{\eta^n}, \dots, \overline{p}_{\eta^n}, n\in N_1 \qquad (2.3\text{-}16)$$

$$(t_{iwl}^s(n))^p - (t_{iwl}^s(n))^{p+1} \le 0 \quad \forall p = \underline{p}_{\eta^n}, \dots, \overline{p}_{\eta^n}, n\in N_1 \qquad (2.3\text{-}17)$$

$$(t_{iwl}^s(n))^{\overline{p}_{\eta^n}} - (t_{iwl}^s(n))^{\underline{p}_{\eta^n}} \le 0 \quad \forall p = \underline{p}_{\eta^n}, \dots, \overline{p}_{\eta^n}, n\in N_1 \qquad (2.3\text{-}18)$$

$$(X_{iw}^s(n))^p - (X_{iw}^s(n))^{p+1} \le 0 \quad \forall p = \underline{p}_{\eta^n}, \dots, \overline{p}_{\eta^n}, n\in N_1 \qquad (2.3\text{-}19)$$

$$(X_{iw}^s(n))^{\overline{p}_{\eta^n}} - (X_{iw}^s(n))^{\underline{p}_{\eta^n}} \le 0 \quad \forall p = \underline{p}_{\eta^n}, \dots, \overline{p}_{\eta^n}, n\in N_1 \qquad (2.3\text{-}20)$$

$Y_i, X_{iw}^p(n) \in \{0,1\}, \ \left(h_i(n)\right)^p \geq 0, (t_{iwl}^s(n))^p \geq 0, s_2^p(n) \geq 0, IDLT_i^p(n) \geq 0 \ \forall\, p \in \eta^n,$
$n \in \{N_1, N_2\}$ (2.3-21)

The NACs constraints in Equations (2.3-15 – 2.3-20) in Model 3 can be relaxed by adding Lagrangian penalty $\mu_1^p(n)$ for Constraints (2.3-15 and 2.3-16), $\mu_2^p(n)$ for Constraints (2.3-17 and 2.3-18), and $\mu_3^p(n)$ for Constraints (2.3-19 and 2.3-20) to the objective function to compensate for the lack of non-anticipativity. The sub-gradient algorithm can be used to solve the Lagrangian model (Boyds, 2008). The Lagrangian relaxation of Model 3 after relaxing the NACs can be presented as follows:

Model 4 (μ_1, μ_2, μ_3, P):

$$Min\ Z_{SCD} = \sum_{i=1}^{I} CY_i + \sum_{p=1}^{P} \sum_{n \in N_1^p} \sum_{s=1}^{S} \sum_{u=1}^{U} \sum_{w=1}^{W} \sum_{i=1}^{I} \sum_{l=1}^{L} w^p(n) r_{wl} (t_{iuwl}^s(n))^p +$$
$$\sum_{p=1}^{P} \sum_{n \in N_2^p} \sum_{s=1}^{S} \sum_{u=1}^{U} \sum_{w=1}^{W} \sum_{i=1}^{I} \sum_{l=1}^{L} pr(n) r_{wl} (t_{iuwl}^s(n))^p +$$
$$\sum_{p=1}^{P} \sum_{n \in \overline{N}_4{}^p} \sum_{i=1}^{n} pr(n) K_i (IDLT_i(n))^p + \sum_{p=1}^{P} \sum_{n \in N^p} eps(s_2^p(n)/r_2 +$$
$$\sum_{p=\underline{p}_{\eta^n}}^{\overline{p}_{\eta^n - 1}} \sum_{n \in N_1} \sum_{i=1}^{I} \mu_1^p(n) \left(\left(h_i(n)\right)^p - \left(h_i(n)\right)^{p+1}\right) + \sum_{n \in N_1} \sum_{i=1}^{I} \mu_1^{\overline{p}_{\eta^n}}(n) \left(\left(h_i(n)\right)^{\overline{p}_{\eta^n}} - \right.$$
$$\left.\left(h_i(n)\right)^{\underline{p}_{\eta^n}}\right) + \sum_{p=\underline{p}_{\eta^n}}^{\overline{p}_{\eta^n - 1}} \sum_{n \in N_1} \sum_{s=1}^{S} \sum_{u=1}^{U} \sum_{w=1}^{W} \sum_{i=1}^{I} \sum_{l=1}^{L} \mu_2^p(n) \left((t_{iuwl}^s(n))^p - \right.$$
$$\left.(t_{iuwl}^s(n))^{p+1}\right) + \sum_{n \in N_1} \sum_{s=1}^{S} \sum_{u=1}^{U} \sum_{w=1}^{W} \sum_{i=1}^{I} \sum_{l=1}^{L} \mu_2^{\overline{p}_{\eta^n}}(n) \left((t_{iuwl}^s(n))^{\overline{p}_{\eta^n}} - \right.$$
$$\left.(t_{iuwl}^s(n))^{\underline{p}_{\eta^n}}\right) + \sum_{p=\underline{p}_{\eta^n}}^{\overline{p}_{\eta^n - 1}} \sum_{n \in N_1} \sum_{w=1}^{W} \sum_{i=1}^{I} \mu_3^p(n) \left((X_{iw}^s(n))^p - (X_{iw}^s(n))^{p+1}\right) +$$
$$\sum_{n \in N_1} \sum_{i=1}^{I} \mu_3^{\overline{p}_{\eta^n}}(n) \left((X_{iw}^s(n))^{\overline{p}_{\eta^n}} - \right.$$
$$\left.(X_{iw}^s(n))^{\underline{p}_{\eta^n}}\right) \qquad (2.4\text{-}1)$$

Subject To:

Equations (2.3-2 – 2.3-14) and (2.3-21).

According to the SCD algorithm, Model 4 can be decomposed into *P* smaller sub-problems. Its objective function for each grid can be obtained by summing the objective function values of all sub-trees for that grid.

For each cluster p, that $p=\underline{p}_{\eta^{n^\circ}}+1,\ldots,\overline{p}_{\eta^{n^0}}$ where $n^0 \in N_1$, the scenario cluster sub-model can be presented (in compact formulation) as follows:

Model 5:

$$Min\ Z_{SCD}^{p} = \sum_{i=1}^{I} CY_i +$$
$$\sum_{n\in lastnodes}\sum_{i=1}^{n} pr(n)K_i(IDLT_i(n))\sum_{n\in N_1^p}\sum_{s=1}^{S}\sum_{u=1}^{U}\sum_{w=1}^{W}\sum_{i=1}^{I}\sum_{l=1}^{L} w^p(n)r_{wl}(t_{iuwl}^{s}(n))^p +$$
$$\sum_{n\in N_2^p}\sum_{s=1}^{S}\sum_{u=1}^{U}\sum_{w=1}^{W}\sum_{i=1}^{I}\sum_{l=1}^{L} pr(n)r_{wl}(t_{iuwl}^{s}(n))^p + \sum_{n\in N_1^p}\sum_{i=1}^{I}(\mu_1^p(n) -$$
$$\mu_1^{p-1}(n))(h_i(n))^p + \sum_{n\in N_1^p}\sum_{s=1}^{S}\sum_{u=1}^{U}\sum_{w=1}^{W}\sum_{i=1}^{I}\sum_{l=1}^{L}(\mu_2^p(n) - \mu_2^{p-1}(n))\ (t_{iuwl}^{s}(n))^p\ +$$
$$\sum_{n\in N_1^p}\sum_{w=1}^{W}\sum_{i=1}^{I}(\mu_3^p(n) - \mu_3^{p-1}(n))\left(X^s{}_{iw}(n)\right)^p + \sum_{n\in N^p} eps \times s_2^p(n)/r_2 \qquad (2.5\text{-}1)$$

Subject to:

Equations (2.2-2 – 2.2-15).

For $p = \underline{p}_{\eta^{n^\circ}}$, the term $(\mu_2^p(n) - \mu_2^{p-1}(n))$ should be changed to $(\mu_2^{\underline{p}_{\eta^n}}(n) - \mu_2^{\overline{p}_{\eta^n}}(n))$ in the objective function. The scenario cluster sub-models in Model 5 are solved for each grid (e_2^k) by implementing the sub-gradient algorithm (Kazemi et al., 2016).

The SCD algorithm solves each cluster independently. Therefore, the first stage decision cannot be fixed for all the clusters. To fix the first stage decision for all the clusters, a First Stage Variable Fixing Heuristic (FVFH) algorithm is proposed. Model 5 is solved for all clusters by the sub-gradient algorithm after implementing the FVFH algorithm. After fixing the first stage decision by FVFH, the constraint in Equation (2.5-2) is added to Model 5 and the model is solved by fixing the values for Y_i.

Algorithm 1: FVFH Algorithm

Step 1: Solve the MSMIP (Model 5) for all clusters for each grid and obtain the vector of binary decision variable (Y_i) for all clusters

For $(\forall\ k \in q, i \in I)$ do

Step 2: Over all clusters, (p) count the number of clusters where the given binary variable takes 1.

Step 3: Update Counter = $\sum_{p=1}^{P} Y_i$

Step 4: Fix the value of (Y_i) as follows and solve the MSMIP by the updated value of (Y_i) (first stage decision variable).

$$Y_i^e = \begin{cases} 1 & Counter > \frac{P}{2} \\ 0 & Counter \leq \frac{P}{2} \end{cases}$$

End for

If Y_i^e equals one, constraint in Equation (5-2) needs to be added to Model 5.

$$Y_i = Y_i^e \tag{2.5-2}$$

The presented solution methodology is implemented using Gurobi and Python. It takes less than six hours to solve a realistic problem instance with as many as 120 candidates. The project team formation problem is not a daily operational problem that needs to be run multiple times a day, so the solution time is not overly concerned.

Chapter Four

RESULTS AND DISCUSSION

The presented models are implemented using Gurobi and Python. It takes less than 40 minutes for the first model and less than 6 hours for the second model to solve a realistic problem instance with as many candidates as 120 . The project team formation problem is not a daily operational problem that needs to be run multiple times a day, so the solution time is not overly concerned.

4.1. Case Study

The global cost of Maintenance, Repair, and Overhaul (MRO) activities in 2015 was nearly $67.1B. Therefore, MRO plays a significant role in the economic activities of a business (Harrison.M). The main concerns of MRO are downtime, quality, and cost. An airline only earns revenue when its aircraft fly. As a result, downtime is a critical factor that should be considered in MRO. The quality of the MRO activities, which is expressed by reliability, is another priority. The quality of MRO depends on the quality of labors used in an MRO project, and it also affects the downtime. The price of MRO activities is an important issue in the industry(Borkowski 2007). Therefore, it is essential to develop a model that can help plan the MRO activities with objectives such as maximizing the quality of the tasks and minimizing the cost of performing the activities.

The MRO market is divided into four distinct areas: aircraft heavy maintenance, engine overhaul, component MRO, and line maintenance. The global MRO market shared by each area in 2015 is shown in Figure 4.1 (Harrison.M). Labor cost for heavy maintenance, line maintenance, engine and component MRO account for 70%, 85%, 20% and 40% of the total cost, respectively (Aero Strategy Management Consulting, 2009). As a result, labor cost is the dominant cost component of heavy maintenance and line maintenance. Therefore, developing a model that can minimize the labor cost and simultaneously maximize the quality of labors in these two types of MRO is necessary

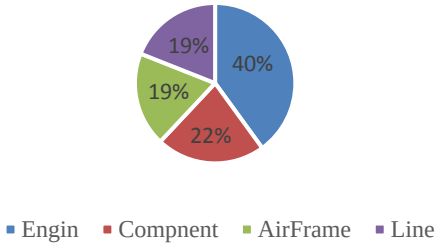

Figure 4-1: MRO market by activity type

4.1.1. Case Study for the First Model

In this case study, a contract for Level D check of aircraft heavy maintenance for one aircraft is considered. Level D check is the inspection and repair of the aircraft airframe, performed at specified time intervals. In Level D check, the aircraft is taken apart completely and each part checked thoroughly in order to return the aircraft to its original condition. Also, it requires 10000 to 30000 man hours (Aero Strategy Management Consulting, 2009). Level D check includes 4-unit works, namely mechanical, airframe and structure, avionic-electrical equipment and avionic- nonelectrical equipment. Each unit has five WDs.

The amount of required time for an aircraft is stochastic and will not be known until after inspections. Three discrete scenarios are considered for the total amount of time needed for an aircraft. It is a random variable drawn from a range as shown in Table 4-1. (Aero Strategy Management Consulting, 2009).

Table 4.1: The data for total required man hours in different scenarios

Scenario	Total Required man hours	Probability
1	Uniform(8000,10000)	1/3
2	Uniform(15000,20000)	1/3
3	Uniform(25000,30000)	1/3

The first stage decision is to employ a subset of 120 candidates and assign them to different WDs in different units. Then in the second stage, allocate the time to each selected

candidate for working in WDs in different units. What distinguishes individuals from each other is their competency to work in each WD and their availability. We assume that each individual works for 12 hours and 7 days of a week (Borkowski 2007). And, the overhaul for each aircraft should be accomplished in 6 weeks. Therefore, the time period of the project is six month. After each individual is employed in the first stage, there will be a fixed cost equal to $2000 for training and accommodation. We consider five competency levels for each WD and obtain the score of each candidate for each WD based on the average of their score in all of the competencies for that WD.

The employees are paid hourly with a rate that is dependent upon their competency in each WD that ranges from $10/hr to $50/hr. The rate of completing a task at competency levels 2 to 5 are from 0.1 to 1 task/hr and for level 0 is 0.01 task/hr. We assume a penalty cost of $20/hr for idle (unutilized) time of the selected candidates at the end of the project. Once an induvial is assigned to a WD in a unit he or she should work at least 8% of total time of that task. Figure 4.2 illustrates the results in a Pareto optimal curve. As shown in the plot the competency increases as the cost increases and it confirms that the two objective functions in this problem are conflicting. In Grid Points 1 to 5, 102, 98, 90, 86, and 84 candidates are selected respectively. The Pareto optimal curves help DMs to make a decision by considering the whole picture and based on DMs' preference one grid point can be selected as the solution. As an example if DM prefers cost to competency the Grid Point 1 or 2 can be selected.

The expected value of perfect information (EVPI) and the value of stochastic solution (VSS) have been used to determine the importance of using stochastic programming in mathematical modeling (Birge 1982).

The EVPI measures the maximum amount of money a DM would be worthwhile to spend in order to know the complete and accurate information about the future.

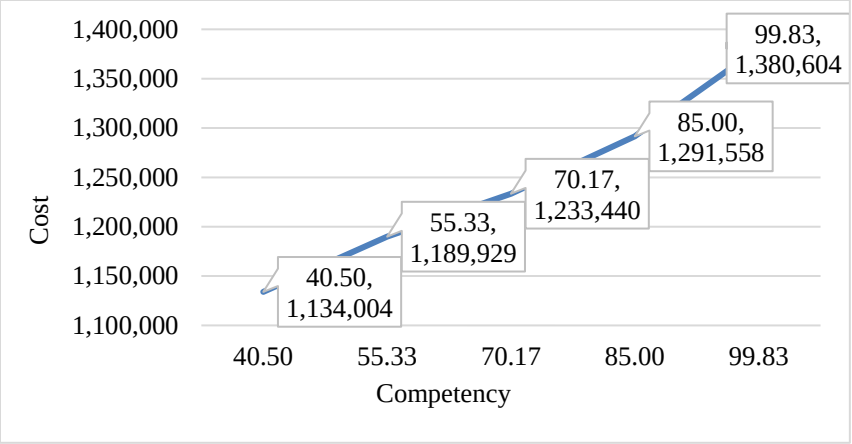

Figure 4-2: The Pareto optimal curve

The EVPI is the differences between Wait-and-See (WS) solution and the recourse problem solution (RP) approach and is defined as EVPI=RP-WS. In the WS approach, perfect information is available. And the model is solved for each scenario separately, and the mean of objective functions is known as WS solution. Table 4.2 shows the solution of RP, WS and EVPI for Z_1 of each grid. The positive value of EVPI for each grid shows the maximum amount of money DM would be ready to spend to obtain the perfect information about the future. These results are consistent with the following expressions (4-1 and 4-2) those are proven the necessity of using two-stage stochastic programming in our specific problem (Madansky 1960).

$RP \geq WS$ (4.1)

$EVPI \geq 0$ (4.2)

Table 4.2: The results of EVPI

Grid	RP	WS	EVPI
1	1,134,004	826,710	307,294
2	1,189,929	896,888	293,041
3	1,233,440	956,027	277,413
4	1,291,558	1,031,695	259,863
5	1,380,604	1,105,910	274,694

The value of stochastic solution (VSS) is the concept that precisely measures the expected gain form solving the stochastic model instead of solving the deterministic model. The VSS is the differences between the expected value solution (EEV) and recourse problem solution (RP) and is defined as VSS=EEV-RP. In EEV approach, first all random variables are replaced by their mean value and the model is solved (EV). Then the expected cost is obtained when the optimal variables of EV approach are considered as an input. The computation shows that there is no feasible solution for the second-stage decision in our problem. Therefore, according to Bridge (1981)(Birge 1982) Bridge (1981)$EEV = +\infty$. And these results are consistent with the following inequalities (Madansky 1960):

$$EEV \geq RP \geq WS \geq EV$$

These reports confirm the accurateness of two-stage stochastic program for our specific problem.

4.1.2. Case Study for the Second Model

In this case study, a team formation problem is considered for a level D of aircraft heavy maintenance project. Aircraft heavy maintenance is the inspection and repair of the aircraft airframe, performed at specified time intervals. Scheduled inspections are typically based on a fixed number of flight hours and the number of take-offs and landings. Level D includes a comprehensive structural inspection and overhaul of the aircraft, intending to return it to its original condition. The frequency of a level D check is approximately every 20,000 to 24,000 flight hours, and it takes 30 to 45 days. Additionally, it requires 10,000 to 30,000 man hours (Aero Strategy Management Consulting, 2009).

A contract for overhaul of four commercial aircraft is considered. Due to the limitation of resources such as the capacity of the maintenance station (hangar) and repair shops, only one aircraft at a time can undergo the overhaul. The amount of required man hours for each aircraft is stochastic and will not be known until after inspections. Three discrete scenarios

are considered for the total amount of work needed for each aircraft, which is a random variable drawn from a range, as shown in Table 4.3.

Table 4.3: The data for total required man hours in different scenarios

Scenario	Total Required man hours	Probability
1	Uniform (8000,10000)	1/3
2	Uniform (15000,20000)	1/3
3	Uniform (25000,30000)	1/3

There are five WDs for each aircraft, and each aircraft is considered as a work unit. The first stage decision is to employ a subset of 115 candidates. Then, in each stage, we assign the individuals from the subset to the different WD in the work unit corresponding to that stage. What distinguishes individuals from each other is their competency to work in each WD and their availability. We assume that each individual works for 12 hours and 7 days a week (Borkowski, 2007). After each individual is employed in each stage, there will be a fixed cost equal to $2,000 for training and accommodation. We consider three competency levels for each WD. The employees are paid hourly with a rate that is dependent on their competency in each WD that ranges from $10/hr to $40/hr. The rate of completing a task at competency levels 1, 2, and 3 are 0.2, 0.8, and 1 task/hr, respectively. We assume a penalty cost of $20/hr for the idle (unutilized) time of the individuals at the end of the last stage. Once an individual is assigned to a WD in a unit, he or she should work at least 8% of total time of that task. The solution for the presented problem instance is as follows.

The value of the total cost and total competency for each grid is shown in Table 4.4. Additionally, the Pareto optimal set is depicted in Figure 4. As shown in Figure 4.3 and Table 4.4, competency increases as the cost increases and it confirms that two objective functions in this problem are conflicting. As it is shown in Table 4.4, the maximum competency is obtained at Grid 4, and its corresponding cost is 8,932,187. Furthermore, minimum cost is achieved at Grid 1 and its corresponding competency is 4,999,374. Therefore, the Pareto optimal set helps DMs to make the decision by considering the whole

picture, and, based on DMs' preference, one grid point is selected as the solution. In Grid 1, 90 individuals are selected as the first-stage decision, in Grid 2, 64 candidates are selected, and in Grids 3 and 4, 87 and 110 candidates are selected, respectively, as the first-stage decision. Additionally, the solution of each cluster for each grid will be provided to DMs. The selected candidates in Grid 1 do not have the maximum competency and because of this, more candidates are selected to accomplish the project.

Table 4.4: The value of each objective function for each grid

Grid	Z_1	Z_2
1	$4,999,374	451,651
2	$5,978,060	657,733
3	$7,220,329	863,814
4	$8,932,187	1,069,895

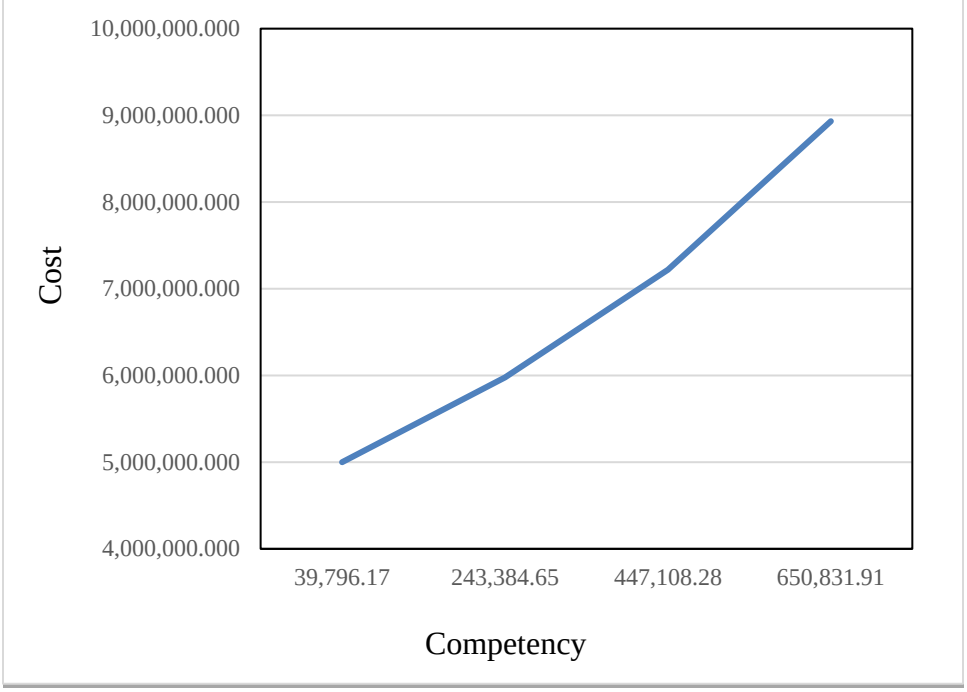

Figure 4-3: The Pareto optimal set

Chapter Five

CONCLUSIONS AND RECOMMENDATIONS

In this study, I proposed two new and generic multi-objective stochastic model for a team formation problem for simultaneous optimization of cost and competency (the two important factors in team formation problems). I applied the Augmented Epsilon-Constraint method to convert the multi-objective model to its equivalent single objective model. The computational challenge in the multi-stage stochastic model was resolved by partitioning the relatively large size of scenarios into small sub-scenarios; subsequently, SCD and a sub-gradient algorithm were adopted for the multi-objective model. The set of Pareto-optimal solutions has been generated by the Augmented Epsilon Constraint method, which showed the tradeoff between the two objectives and gave important insights into the problem. A case study from the airline industry was presented to demonstrate an application of the model and its solution methodology, in which a contractor for overhauling a set of aircraft was to make a decision for employing a subset of individuals from a pool of candidates such that labor cost is minimized and quality of maintenance is maximized by choosing candidates with high competency. The model can help DMs to form a team in an effective and efficient way under uncertain environments where the amount of required man hours is unknown in different stages of a project.

Several promising new research directions could be explored beyond the current research. Additional objectives such as maximization of compatibility in collaboration between individuals who work in the same WD can be considered. Additional employment scenarios can also be considered in modeling. Application and sensitivity analysis of the presented work in various industries are also encouraged.

LIST OF REFERENCES

Aero Strategy Management Consulting. Global MRO Market and Assessment. Prepared for: Aeronautical Repair Station Association 2009.

Baykasoglu, A., T. Dereli and S. Das (2007). "Project team selection using fuzzy optimization approach." Cybernetics and Systems: An International Journal **38**(2): 155-185.

Birge, J. R. (1982). "The value of the stochastic solution in stochastic linear programs with fixed recourse." Mathematical programming **24**(1): 314-325.

Birge, J. R. and F. Louveaux (2011). Introduction to stochastic programming, Springer Science & Business Media.

Borkowski, J. (2007). Aircraft maintenance repair and overhaul market study: Glasgow international airport. Airport regions conference (ARC).

Chen, S.-J. and L. Lin (2004). "Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering." IEEE Transactions on Engineering Management **51**(2): 111-124.

Cobuloglu, H. I. and İ. E. Büyüktahtakın (2017). "A Two-Stage Stochastic Mixed-Integer Programming Approach to the Competition of Biofuel and Food Production." Computers & Industrial Engineering.

Dupačová, J., G. Consigli and S. W. Wallace (2000). "Scenarios for multistage stochastic programs." Annals of operations research **100**(1-4): 25-53.

Escudero, L. F., M. A. Garín, M. Merino and G. Pérez (2012). "An algorithmic framework for solving large-scale multistage stochastic mixed 0–1 problems with nonsymmetric scenario trees." Computers & Operations Research **39**(5): 1133-1144.

Escudero, L. F., P. V. Kamesam, A. J. King and R. J. Wets (1993). "Production planning via scenario modelling." Annals of Operations research **43**(6): 309-335.

Feng, B., Z.-Z. Jiang, Z.-P. Fan and N. Fu (2010). "A method for member selection of cross-functional teams using the individual and collaborative performances." European Journal of Operational Research **203**(3): 652-661.

Gutiérrez, J. H., C. A. Astudillo, P. Ballesteros-Pérez, D. Mora-Melià and A. Candia-Véjar (2016). "The multiple team formation problem using sociometry." Computers & Operations Research **75**: 150-162.

Harrison.M. (2016). "https://www.iata.org/whatwedo/workgroups/Documents/MCC-2016-BKK/D1-1000-1030-mro-forecast-icf.pdf." IATA 12th Maintenance Cost Conference, Bankok, Thailand.

Housh, M., A. Ostfeld and U. Shamir (2013). "Limited multi-stage stochastic programming for managing water supply systems." Environmental modelling & software **41**: 53-64.

Hwang, C.-L. and A. S. M. Masud (2012). Multiple objective decision making—methods and applications: a state-of-the-art survey, Springer Science & Business Media.

Karsak, E. E. (2000). A fuzzy multiple objective programming approach for personnel selection. Systems, Man, and Cybernetics, 2000 IEEE International Conference on, IEEE.

Kazemi Zanjani, M., M. Nourelfath and D. Ait-Kadi (2010). "A multi-stage stochastic programming approach for production planning with uncertainty in the quality of raw materials and demand." International Journal of Production Research **48**(16): 4701-4723.

Körpeoğlu, E., H. Yaman and M. S. Aktürk (2011). "A multi-stage stochastic programming approach in master production scheduling." European Journal of Operational Research **213**(1): 166-179.

Liu, X., D. Ruan and Y. Xu (2005). "A study of enterprise human resource competence appraisement." Journal of enterprise information management **18**(3): 289-315.

Madansky, A. (1960). "Inequalities for stochastic linear programming problems." Management science **6**(2): 197-204.

Maurer, I. (2010). "How to build trust in inter-organizational projects: The impact of project staffing and project rewards on the formation of trust, knowledge acquisition and product innovation." International Journal of Project Management **28**(7): 629-637.

Mavrotas, G. (2009). "Effective implementation of the ε-constraint method in multi-objective mathematical programming problems." Applied mathematics and computation **213**(2): 455-465.

Ramezani, M., M. Bashiri and R. Tavakkoli-Moghaddam (2013). "A new multi-objective stochastic model for a forward/reverse logistic network design with responsiveness and quality level." Applied Mathematical Modelling **37**(1): 328-344.

Rockafellar, R. T. and R. J.-B. Wets (1991). "Scenarios and policy aggregation in optimization under uncertainty." Mathematics of operations research **16**(1): 119-147.

Rosa, C. H. and A. Ruszczyński (1996). "On augmented Lagrangian decomposition methods for multistage stochastic programs." Annals of Operations Research **64**(1): 289-309.

Ruszczynski, A. (1989). "Regularized decomposition and augmented Lagrangian decomposition for angular linear programming problems." Aspiration Based Decision Support Systems: 80-91.

Ruszczyński, A. (1997). "Decomposition methods in stochastic programming." Mathematical programming **79**(1): 333-353.

Snyder, C. S. (2014). A Guide to the Project Management Body of Knowledge: PMBOK (®) Guide, Project Management Institute.

Strnad, D. and N. Guid (2010). "A fuzzy-genetic decision support system for project team formation." Applied soft computing **10**(4): 1178-1187.

Tavana, M., F. Azizi, F. Azizi and M. Behzadian (2013). "A fuzzy inference system with application to player selection and team formation in multi-player sports." Sport Management Review **16**(1): 97-110.

Tsai, H.-T., H. Moskowitz and L.-H. Lee (2003). "Human resource selection for software development projects using Taguchi's parameter design." European Journal of Operational Research **151**(1): 167-180.

Tseng, T.-L. B., C.-C. Huang, H.-W. Chu and R. R. Gung (2004). "Novel approach to multi-functional project team formation." International Journal of Project Management **22**(2): 147-159.

Valladão, D. M., Á. Veiga and G. Veiga (2014). "A multistage linear stochastic programming model for optimal corporate debt management." European Journal of Operational Research **237**(1): 303-311.

Wi, H., S. Oh, J. Mun and M. Jung (2009). "A team formation model based on knowledge and collaboration." Expert Systems with Applications **36**(5): 9121-9134.

Wil, v. d. A., J. Desel and A. Oberweis (2003). Business process management: models, techniques, and empirical studies, Springer.

Zanjani, M. K., O. S. Bajgiran and M. Nourelfath (2016). "A hybrid scenario cluster decomposition algorithm for supply chain tactical planning under uncertainty." European Journal of Operational Research **252**(2): 466-476.

Zhang, L. and X. Zhang (2013). "Multi-objective team formation optimization for new product development." Computers & Industrial Engineering **64**(3): 804-811.

Zzkarian, A. and A. Kusiak (1999). "Forming teams: an analytical approach." IIE transactions **31**(1): 85-97.

APPENDIX

Model 1

```
from math import exp, exp
import copy
import random
import math
import numpy as np
from numpy import linalg as LA
from numpy import matrix
import xlsxwriter
workbook = xlsxwriter.Workbook('17a.xlsx')
worksheet = workbook.add_worksheet('results')
obj_format = workbook.add_format({'num_format': '0.000000'})
worksheet.write('A1', 'grid')
worksheet.write('B1', 'candidate')
worksheet.write('C1', 'W')
worksheet.write('D1', 'U')
worksheet.write('E1', 'value')
worksheet.write('F1', 'grid')
worksheet.write('G1', 'Z1')
worksheet.write('H1', 'Z2')
worksheet.write('I1', 'gride')
worksheet.write('J1', 'L=1')
worksheet.write('K1', 'L=2')
worksheet.write('L1', 'L=3')
worksheet.write('M1', 'L=4')
worksheet.write('N1', 'L=5')
worksheet.set_column(5, 47, 2)
import numpy as np
from gurobipy import *
I={}#workforcel
I=range(120)
U={}#units to be overhauled
U=range(1,5)
W={}#work discipline
W=range(5)
L={}#Competency Levels
L=range(1,6)
Lprime={}#Competency Levels
Lprime=range(2,3)
S={}#Scenarios
S=range(1,4)
M=1000000 #

E={}#competency of individuals in work disciplines
for i in I:
    for w in W:
        for l in L:

            E[i,w,l]=0 #np.random.random_integers(0,1) #2bechanged
```

```
            E[i,w,1]=0 #np.random.random_integers(0,1) #2bechanged
for i in I:
    for w in W:
        beta=random.randint(1,6)
        E[i,w,beta]=1

T={}

for u in U:
    for w in W:
        for s in range (1,2):
            T[u,w,s]=random.randint(400,500)
        for s in range (2,3):
            T[u,w,s]=random.randint(750,1000)
        for s in range (3,4):
            T[u,w,s]=random.randint(1250,1500)

bb={}#minimum work time
for w in W:
    for u in U:
        bb[w,u]=0.1 #2bechanged
h={}#available time of individual workforce members
for i in I:

    h[i]=336

C={}#fixed hiring cost
for i in I:
    C[i]=2000

r={}#rate (hourly cost) for working at a certain level of a work discipline
for w in W:
    for l in (1,2):
        r[w,l]=10
    for l in (2,3):
        r[w,l]=20
    for l in (3,4):
        r[w,l]=25
    for l in (4,5):
        r[w,l]=40
    for l in (5,6):
        r[w,l]=50
alpha={} #The rate of working in level l
for l in (1,2):
        alpha[l]=0.1
for l in (2,3):
        alpha[l]=0.3
for l in (3,4):
        alpha[l]=0.5
for l in (4,5):
        alpha[l]=0.8
```

```
for l in (5,6):
        alpha[l]=1

v={}
for i in I:
    v[i]=20
prob={}
for ss in S:
    prob[ss]=float (1.0/len(S))
    print (prob[ss])
```

```
def getOV1(I,U,W,T,h,bb,C,r,S,fmode,z2ub,finterval1,iii):



    model = Model("17a")
    tout={}
    xout={}
    yout={}

    IO1=copy.deepcopy(I)
    UO1=copy.deepcopy(U)
    WO1=copy.deepcopy(W)
    LO1=copy.deepcopy(L)
    EO1=copy.deepcopy(E)
    TO1=copy.deepcopy(T)
    hO1=copy.deepcopy(h)
    bO1=copy.deepcopy(bb)



    x={}#binary: assignment of worker to a work discipline in a unit
    for i in IO1:
        for u in UO1:
            for w in WO1:
                x[i,u,w]=model.addVar(vtype='B',lb=0,ub=1)
    y={}#binary: whether s/o is selected
    for i in IO1:
        y[i]=model.addVar(vtype='B',lb=0,ub=1)
    t1={}#actual time spent
    #    for s in ss:
    for i in IO1:
        for u in UO1:
            for w in WO1:
                for l in LO1:
                    for ss in S:
                        t1[i,u,w,l,ss]=model.addVar(lb=0,ub=GRB.INFINITY)
    Id={}
    for i in IO1:
        for ss in S:
            Id[i,ss]=model.addVar(lb=0,ub=GRB.INFINITY)
```

```
    WT={}
    for i in IO1:
        for ss in S:
            WT[i,ss]=model.addVar(lb=0,ub=GRB.INFINITY)
    f1={}#Actual time spent together
    #    for s in ss:
    for i in IO1:
        for j in IO1:
            for u in UO1:
                for w in WO1:
                    for ss in S:
                        f1[i,j,u,w,ss]=model.addVar(lb=0,ub=GRB.INFINITY)

    model.update()
    C2={}
    #    for s in ss:
    for u in UO1:
        for w in WO1:
            for l in LO1:
                for ss in S:
                    C2[u,w,l] = model.addConstr(quicksum(t1[i,u,w,l,ss]*alpha[l] for i in IO1 for l in LO1)>=TO1[u,w,ss]
    C3={}
    C4={}
    for i in IO1:
        for u in UO1:
            for w in WO1:
                for ss in S:
                    C3[i,u,w,ss] = model.addConstr(quicksum(t1[i,u,w,l,ss] for l in LO1)>=bO1[w,u]*TO1[u,w,ss]*x[i,u,w])
                    C4[i,u,w,ss] = model.addConstr(quicksum(t1[i,u,w,l,ss] for l in LO1)<=M*x[i,u,w])
#                     C10[i,u,w,ss] = model.addConstr(quicksum(t1[i,u,w,l,ss] for l in LO1)>x[i,u,w])
    C5={}
    for i in IO1:
        for ss in S:
            C5[i,ss] = model.addConstr(quicksum(t1[i,u,w,l,ss] for u in UO1 for w in WO1 for l in LO1)<=hO1[i]*y[i])
    C6={}
    for i in IO1:
        for w in WO1:
            for u in UO1:
                C6[i,w,u] = model.addConstr(y[i]>=x[i,u,w])
```

```
C7={}
for i in IO1:
    C7[i] = model.addConstr(quicksum(x[i,u,w] for u in UO1 for w in WO1)>=y[i])
C8={}
for i in IO1:
    for ss in S:
        C8[i,ss] = model.addConstr(quicksum(t1[i,u,w,l,ss] for u in UO1 for w in WO1 for l in LO1)==WT[i,ss])
C9={}
for i in IO1:
    for ss in S:
        C9[i,ss] = model.addConstr(hO1[i]*y[i]-WT[i,ss]==Id[i,ss])

model.update()
o={}
o[0]=0
#    o[0]=-quicksum(C[u,w]*x[i,u,w] for u in UO1 for w in WO1 for i in IO1)
for i in IO1:
    for U in UO1:
        for w in WO1:
            o[0]=o[0]-C[i]*y[i]
for ss in S:
    for i in IO1:
        o[0]=o[0]-float(1.0/len(S))*(v[i]*Id[i,ss])

for i in IO1:
    for w in WO1:

        for u in UO1:
            for l in LO1:
                for ss in S:

                    o[0]=o[0]-float(1.0/len(S))*r[w,l]*t1[i,u,w,l,ss]
if fmode>=1:
        o[1]=0
        for i in IO1:
            for u in UO1:
                for w in WO1:
                    for l in LO1:
                        for s in S:
                                o[1]+=float(prob[s])*(EO1[i,w,l]*t1[i,u,w,l,s] )
```

```
        s2 = model.addVar(lb=0,ub=GRB.INFINITY)
#         s3 = model.addVar(lb=0,ub=GRB.INFINITY)
        model.update()

        cz2 = model.addConstr(o[1]-s2==z2ub)
#         cz3 = model.addConstr(o[2]-s3==z3ub)
        model.update()
        o[0]+=0.01*(s2/float(finterval1*3))


model.modelSense = GRB.MAXIMIZE
#optimizing the first objective without considering the other two objectives
model.setObjective(o[0])
model.setParam("MIPGap",0.05)
model.setParam("OutputFlag",0)
model.update()
model.optimize()


Z1=0
if model.status==GRB.OPTIMAL:
    Z1=model.objVal
```

```
    for i in IO1:

        for u in UO1:
            for w in WO1:

                xout[i,u,w]=copy.deepcopy(x[i,u,w].x)
                for l in LO1:
                    for ss in S:
                        |
                            tout[i,u,w,l,ss]=copy.deepcopy(t1[i,u,w,l,ss].x)

    for i in IO1:
        yout[i]=copy.deepcopy(y[i].x)


    Z12=0
    for i in IO1:
        for u in UO1:
            for w in WO1:
                for l in LO1:
                    for s in S:
                            Z12+=float(prob[s])*(EO1[i,w,l]*t1[i,u,w,l,s].x )



    return (-Z1,Z12,0,tout,xout,yout)
else:
    return (-1,-1,-1)
```

```
def getOV2(I,U,W,L,E,T,h,bb,C,r,S):



    model = Model("17a")
    tout={}
    xout={}
    yout={}

    IO1=copy.deepcopy(I)
    UO1=copy.deepcopy(U)
    WO1=copy.deepcopy(W)
    LO1=copy.deepcopy(L)
    EO1=copy.deepcopy(E)
    TO1=copy.deepcopy(T)
    hO1=copy.deepcopy(h)
    bO1=copy.deepcopy(bb)

    x={}#binary: assignment of worker to a work discipline in a unit
    for i in IO1:
        for u in UO1:
            for w in WO1:
                x[i,u,w]=model.addVar(vtype='B',lb=0,ub=1)
    y={}#binary: whether s/o is selected
    for i in IO1:
        y[i]=model.addVar(vtype='B',lb=0,ub=1)
    t1={}#actual time spent
    #    for s in ss:
    for i in IO1:
        for u in UO1:
            for w in WO1:
                for l in LO1:
                    for ss in S:
                        t1[i,u,w,l,ss]=model.addVar(lb=0,ub=GRB.INFINITY)
    f1={}#Actual time spent together
    #    for s in ss:
    for i in IO1:
        for j in IO1:
            for u in UO1:
                for w in WO1:
                    for ss in S:
                        f1[i,j,u,w,ss]=model.addVar(lb=0,ub=GRB.INFINITY)


    Id={}
    for i in IO1:
        for ss in S:
            Id[i,ss]=model.addVar(lb=0,ub=GRB.INFINITY)
```

```
            Id[i,ss]=model.addVar(lb=0,ub=GRB.INFINITY)
    WT={}
    for i in IO1:
        for ss in S:
            WT[i,ss]=model.addVar(lb=0,ub=GRB.INFINITY)

    model.update()
    C2={}
    #    for s in ss:
    for u in UO1:
        for w in WO1:
            for l in LO1:
                for ss in S:
                    C2[u,w,l] = model.addConstr(quicksum(t1[i,u,w,l,ss]*alpha[l] for i in IO1 for l in LO1)>=TO1[u,w,ss]
    C3={}
    C4={}
    for i in IO1:
        for u in UO1:
            for w in WO1:
                for ss in S:
                    C3[i,u,w,ss] = model.addConstr(quicksum(t1[i,u,w,l,ss] for l in LO1)>=bO1[w,u]*TO1[u,w,ss]*x[i,u,w])
                    C4[i,u,w,ss] = model.addConstr(quicksum(t1[i,u,w,l,ss] for l in LO1)<=M*x[i,u,w])
#                     C10[i,u,w,ss] = model.addConstr(quicksum(t1[i,u,w,l,ss] for l in LO1)>=x[i,u,w])
    C5={}
    for i in IO1:
        for ss in S:
            C5[i,ss] = model.addConstr(quicksum(t1[i,u,w,l,ss] for u in UO1 for w in WO1 for l in LO1)<=hO1[i]*y[i])
    C6={}
    for i in IO1:
        for w in WO1:
            for u in UO1:
                C6[i,w,u] = model.addConstr(y[i]>=x[i,u,w])
    C7={}
    for i in IO1:
        C7[i] = model.addConstr(quicksum(x[i,u,w] for u in UO1 for w in WO1)>=y[i])

    C8={}
    for i in IO1:
        for ss in S:
            C8[i,ss] = model.addConstr(quicksum(t1[i,u,w,l,ss] for u in UO1 for w in WO1 for l in LO1)==WT[i,ss])
    C9={}
    for i in IO1:
        for ss in S:
            C9[i,ss] = model.addConstr(hO1[i]*y[i]-WT[i,ss]==Id[i,ss])

    model.update()
    o={}
```

```
o[0]=0
for i in IO1:
    for u in UO1:
        for w in WO1:
            for l in LO1:
                for s in S:
                    o[0]+=float(prob[s])*(EO1[i,w,l]*t1[i,u,w,l,s] )



model.modelSense = GRB.MAXIMIZE
#optimizing the first objective without considering the other two objectives
model.setObjective(o[0])
model.setParam("MIPGap",0.05)
model.setParam("OutputFlag",0)
model.update()
model.optimize()


Z2=0
if model.status==GRB.OPTIMAL:
    Z2=model.objVal
#         print ("Optimal z*1:", model.objVal)

    for i in IO1:
        for u in UO1:
            for w in WO1:
                for l in LO1:
                    for ss in S:
                        tout[i,u,w,l,ss]=copy.deepcopy(t1[i,u,w,l,ss].x)

    for i in IO1:
          for u in UO1:
                for w in WO1:
                    xout[i,u,w]=copy.deepcopy(x[i,u,w].x)
    for i in IO1:
        yout[i]=copy.deepcopy(y[i].x)

    Z21=0
    for i in IO1:

        Z21=Z21+C[i]*y[i].x
    for ss in S:
        for i in IO1:
             Z21=Z21+float(1.0/len(S))*Id[i,ss].x
             for w in WO1:

                for u in UO1:
                    for l in LO1:
```

```
                        Z21=Z21+float(1.0/len(S))*r[w,l]*t1[i,u,w,l,ss].x


    return (Z21,Z2,0,tout,xout,yout)
else:
    return (-1,-1,-1)
```

```
ElementofParetoSet={}
payoffClust={}
worstZ1={}
myTableau=[[0,0],[0,0]]



jay0=getOV1(I,U,W,T,h,bb,C,r,S,0,0,0,0)
print("jay0", jay0[0],jay0[1])
myTableau[0][0]+=jay0[0]
myTableau[0][1]+=jay0[1]
jay1=getOV2(I,U,W,L,E,T,h,bb,C,r,S)


myTableau[1][0]+=(jay1[0])
myTableau[1][1]+=(jay1[1])

print("jay1", myTableau[1][0],myTableau[1][1])
f1min=min(myTableau[0][0],myTableau[1][0])
f2min=min(myTableau[0][1],myTableau[1][1])

f1max=max(myTableau[0][0],myTableau[1][0])
f2max=max(myTableau[0][1],myTableau[1][1])

f1interval=float(f1max-f1min)/4
f2interval=float(f2max-f2min)/4
print("f2interval",f2interval)

f1points=[0,0,0,0,0]
f2points=[0,0,0,0,0]

for ii in range(4):
    f1points[ii]=f1min+ii*f1interval
    f2points[ii]=f2min+ii*f2interval

f1points[4]=f1max
worstZ1=f1max #upper bound of Z1 to be used in the next cell
f2points[4]=f2max

for ii in range(5):

    payoffClust[ii]=[f2points[ii]]
    print([f2points[ii]] )
```

```
import time
finalGrid={}



totalTic=time.time()

row=1
TT={}
for i in range(5):
    for l in L:
        for ss in S:
            TT[i,l,ss]=0
    worksheet.write_number(row, 0, i )
    jay={}#z2
    jay[i]=getOV1(I,U,W,T,h,bb,C,r,S,1,payoffClust[i][0],f2interval,i)



    ZZ1=([jay[i][0]] )
    ZZ2=([jay[i][1] ])

    finalGrid[i+1]=[ZZ1,ZZ2]
    print("gird",i,finalGrid[i+1])
    for ii in I:
         worksheet.write_number(row, 1, ii )
         if   jay[i][5][ii]!=0:
             worksheet.write_number(row, 2,jay[i][5][ii] )

             for u in U:
                  worksheet.write_number(row, 3, u )

                  for w in W:
                      worksheet.write_number(row, 4, w )
                      if jay[i][4][ii,u,w]!=0:

                        # print("x for",ii,u,w,"=",jay[i][4][ii,u,w])
                         for l in L:
                                worksheet.write_number(row, 5, l )
                                for ss in S:
                                    worksheet.write_number(row, 6, ss )
                                    if jay[i][3][ii,u,w,l,ss]!=0:
                                      #  print("t for",l,ss,jay[i][3][ii,u,w,l,ss])
                                        worksheet.write_number(row,7, jay[i][3][ii,u,w,l,ss] )
                                        row=row+1
```

```
print("Final grid: ",finalGrid)

import operator
MM={}
NN={}

for i in range(5): #z2

    MM[i]=(finalGrid[i+1][0])

    print(MM[i])



sorted_MM = sorted(MM.items(), key=operator.itemgetter(1))
print(sorted_MM )
print("==================")
NN={}


for i in range(5): #z2

    NN[i]=(finalGrid[i+1][1])

import operator

sorted_NN = sorted(NN.items(), key=operator.itemgetter(1))
print(sorted_NN )

print("####################")



workbook.close()
```

Model 2:

```
# In[1]:

from math import exp, exp
import copy
import random
import math
import numpy as np
from numpy import linalg as LA
from numpy import matrix
import xlsxwriter

workbook = xlsxwriter.Workbook('16b3.xlsx')
worksheet = workbook.add_worksheet('results')

obj_format = workbook.add_format({'num_format': '0.000000'})


worksheet.write('A1', 'grid')
worksheet.write('B1', 'cluster.')
worksheet.write('C1', 'candidate')
worksheet.write('D1', 'value1')
worksheet.write('E1', 'value2')
worksheet.write('F1', 'grid')
worksheet.write('G1', 'Z1')
worksheet.write('H1', 'Z2')

worksheet.set_column(5, 47, 2)



# In[2]:

import numpy as np
from gurobipy import *
I={}#workforcel
I=range(125)

U={}#units to be overhauled
```

```
U=range(1,5)
W={}#work discipline
W=range(5)
L={}#Competency Levels
L=range(3)
Lprime={}#Competency Levels
Lprime=range(1,3)
S={}#Scenarios
S=range(1,4)
M=1000000 #a sufficiently big number


F={}#collaboration b/w people

for i in I:
    for j in I:
        F[i,j]= random.sample([0.85, 0.65, 0.45, 0.25, 0.15], 1) #2bechanged
        F[i,j]=F[i,j][0]
for i in I:
    F[i,i]=0


E={}#competency of individuals in work disciplines
for i in I:
    for w in W:
        for l in L:
            E[i,w,l]=0 #np.random.random_integers(0,1) #2bechanged
for i in I:
    for w in W:
        beta=random.randint(0,2)
        E[i,w,beta]=1
```

```
Tprim={}#Min time req of each individual

for u in U:
    for w in W:
        for s in range (1,2):
            Tprim[u,w,s]=random.randint(400,700)
        for s in range (2,3):
            Tprim[u,w,s]=random.randint(2000,3000)
        for s in range (3,4):
            Tprim[u,w,s]=random.randint(4000,6000)


bb={}#minimum work time
for w in W:
    for u in U:
        bb[w,u]=0.3 #2bechanged
h={}#available time of individual workforce members
for i in I:

#    h[i]=4000#2bechanged
    h[i]=1300

C={}#fixed hiring cost
C=2000 #2bechanged

r={}#rate (hourly cost) for working at a certain level of a work discipline
for w in W:
    for l in (0,1):
        r[w,l]=10
    for l in (1,2):
        r[w,l]=20
    for l in (2,3):
        r[w,l]=40
alpha={} #The rate of working in level l
for l in (0,1):
        alpha[l]=0.4
for l in (1,2):
        alpha[l]=0.8
```

```
for l in (2,3):
        alpha[l]=1
v={}   ###cost (hourly cost) of idle time for each person
for i in  I:
    v[i]=20
prob={}
for ss in range(1,pow(3,len(U))+1):
    prob[ss]=float(1.0/pow(3.0,len(U)))


# In[3]:

class Node:
    eyed = 0
    scenario = 1
    stage = 0
    fatherid = 0
    setscenario = set([])
    setclusters = set([])

    def __init__(self, eyed, scenario, stage, fatherid, setscenario, setcluster):
        self.eyed = eyed
        self.scenario = scenario
        self.stage = stage
        self.fatherid = fatherid
        self.setscenario = setscenario
        self.setcluster = setcluster

def make_node(eyed, scenario, stage, fatherid, setscenario, setcluster):
    node = Node(eyed, scenario, stage, fatherid, setscenario, setcluster)
    return node
# ================================================


# In[4]:

# populate the tree
numNode=1;
for u in U:
    numNode=numNode+pow(3, u)
```

```
allnodes=set([i for i in range(1,numNode+1)])
N1=set([1,2,3,4])
N1prime=set([1])
N3=set([2])
N2prim=N1.difference(N1prime)
print("N2prim",N2prim)
N2=allnodes.difference(N1)

allscenarios=set([i for i in range(1,pow(3,len(U)))])

# l=3
numClust=pow(3,len(U)-1)#/3 #will be integer bc the tree is symmetric

allclusters=set([i for i in range(1,numClust+1)])

nodes=[make_node(1,1,0,0,allscenarios,allclusters)]

def make_children(father):
    numnode=father.eyed*3-2
    currentstage=father.stage+1
    numChildren=pow(3,len(U)-currentstage)

    xx=0
    for i in range(currentstage):
        xx=xx+pow(3,i)

    scen1start=(numnode+1-xx-1)*numChildren+1 #formula of this
    setscen1=set([scen1start+i for i in range(numChildren)])
    scen2start=(numnode+2-xx-1)*numChildren+1 #formula of this
    setscen2=set([scen2start+i for i in range(numChildren)])
    scen3start=(numnode+3-xx-1)*numChildren+1 #formula of this
    setscen3=set([scen3start+i for i in range(numChildren)])

    child1=make_node(numnode+1,1,currentstage,father.eyed,setscen1,set([]))
    child2=make_node(numnode+2,2,currentstage,father.eyed,setscen2,set([]))
    child3=make_node(numnode+3,3,currentstage,father.eyed,setscen3,set([]))

    nodes.append(child1)
    nodes.append(child2)
```

```
clusters={}
IP={}
IP2={}
IPprim={}
num=1+sum([pow(3,u) for u in range(1,len(U))])

for i in range(1,numClust+1):
    clusters[i]=set([])
    #add fathers
    x=1
    y=num+2
    y=y/3
    clusters[i].add(y)
    z=sum([ii for ii in range(numNode) if nodes[ii].eyed==y])
    nodes[z].setcluster.add(i)
    while x<=len(U)-1:
        s=nodes[int(y-1)].scenario
        if s==1:
            y=(y+1)/3
        elif s==2:
            y=y/3
        elif s==3:
            y=(y-1)/3
        clusters[i].add(y)
        z=sum([ii for ii in range(numNode) if nodes[ii].eyed==y])
        nodes[z].setcluster.add(i)
        x=x+1
    #add leaves
    clusters[i].add(num+1)
    z=sum([ii for ii in range(numNode) if nodes[ii].eyed==num+1])
    nodes[z].setcluster.add(i)

    clusters[i].add(num+2)
    z=sum([ii for ii in range(numNode) if nodes[ii].eyed==num+2])
    nodes[z].setcluster.add(i)

    clusters[i].add(num+3)
    z=sum([ii for ii in range(numNode) if nodes[ii].eyed==num+3])
    nodes[z].setcluster.add(i)
```

```
    IP[i]=set([])
    IP[i]=N1.intersection(clusters[i])
    IP2[i]=set([])
    IP2[i]=N2.intersection(clusters[i])
    num=num+3
    IPprim[i]=set([])                          ############
    IPprim[i]=N2prim.intersection(clusters[i])
#     print ("IP",i,IP[i])

#     print(clusters[i])
#     print("IPprim",IPprim[i])

#     print(len(IP[i]))

# for n in range(numNode):
#         print("======node #",n,nodes[n].eyed)
#         print("set of scenarios:",nodes[n].setscenario)
#         print("set of clusters:",nodes[n].setcluster)
# for i in range (1, numClust+1):
#          print("==============Cluster#",i,clusters[i])
pr={}
for i in (1, numClust+1):
    for j in (1,3):#generalization: (1,len(IP[i])+1):
        pr[i,j]=(float(3)/pow(3,len(U)))
# #         print ("Pr", '%.15f' % pr[i,j])
# reprez={}
# for i in range(len(nodes)):
#     print(nodes[i].eyed)
# print(nodes[1].stage)
# print(nodes[i].eyed)


# In[7]:

# lastnodes=[]
# for n in allnodes:
#     myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
#     if nodes[myn].stage==4:
#         lastnodes.append(nodes[myn].eyed)
```

```
nodeset2 = clusters[1]
lastnode2=[]
for n in nodeset2:
        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
        if nodes[myn].stage==4:
            lastnode2.append(nodes[myn].eyed)


print(lastnode2)


# In[8]:

def getOV1(I,U,W,L,E,SS,T,h,bb,C,r,alpha,fmode,z2ub,finterval1,mu,mu2,mu3,clustNum,k,fix,finalXX,iii):

    nodeset2 = clusters[clustNum]
    nodeset=nodeset2.difference(N1prime)
    nodeset3=nodeset2.difference(N1)
#    print("clustNum,nodeset3",clustNum,nodeset3)
    clustScen=set([(clustNum-1)*3+1,(clustNum-1)*3+2,(clustNum-1)*3+3])
    nodeindex=[]
    for i in (nodeset):
        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==i])
        nodeindex.append(myn)

    lastnodes=[]
    for n in nodeset:
        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
        if nodes[myn].stage==4:
            lastnodes.append(nodes[myn].eyed)
    model = Model("16b")

    tout={}
    xout={}
    Hout={}
    yout={}

    IO1=copy.deepcopy(I)

    UO1=copy.deepcopy(U)
    WO1=copy.deepcopy(W)
```

```
    LO1=copy.deepcopy(L)
    EO1=copy.deepcopy(E)
    TO1=copy.deepcopy(T)
    hO1=copy.deepcopy(h)
    bO1=copy.deepcopy(bb)

    vO1=copy.deepcopy(v)

    x={} #binary: assignment of worker to a work discipline in a unit

    for n in nodeset:
        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
        for i in IO1:
            for w in WO1:
                mySS = nodes[myn].setscenario.intersection(clustScen)
                for s in mySS:
                        x[n,i,w,s]=model.addVar(vtype='B',lb=0,ub=1)
    y={}#binary: whether s/o is selected
    for i in IO1:
        y[i]=model.addVar(vtype='B',lb=0,ub=1)
    z={}#binary: whether s/o is selected
    for i in IO1:
        z[i]=model.addVar(vtype='B',lb=0,ub=GRB.INFINITY)

    t1={}#actual time spent
#    for s in ss:
    for i in IO1:
        for n in nodeset:
            myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
#         for u in UO1:
            for w in WO1:
                for l in LO1:
                    mySS = nodes[myn].setscenario.intersection(clustScen)
                    for s in mySS:

                        t1[n,i,w,l,s] = model.addVar(lb=0,ub=GRB.INFINITY)
```

```
    HH={}

    for n in nodeset:
        for i in IO1:
            HH[n,i]=model.addVar(lb=0,ub=GRB.INFINITY)

    model.update()
    C2={}
#    for s in ss:
#     for u in UO1:
    for w in WO1:
        for l in LO1:
            for n in nodeset:
                myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
                mySS = nodes[myn].setscenario.intersection(clustScen)
                u=nodes[myn].stage
                for s in mySS:
                    C2[n,w,l,s] = model.addConstr((quicksum(t1[n,i,w,l,s]*alpha[l] for i in IO1 for l in LO1))>=TO1[u,w,s])
#                     C2[n,w,l,s] = model.addConstr(quicksum(t1[n,i,w,l,s]*alpha[l] for i in IO1 for l in LO1))>=TO1[u,w,s]

#                for ss in SS:
#                    C2[u,w,l] = model.addConstr(quicksum(t1[i,u,w,l,ss]*alpha[l] for i in IO1 for l in LO1)>=TO1[u,w,ss])
    C3={}
    C4={}
    for i in IO1:
        for w in WO1:
            for n in nodeset:
                myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
                mySS = nodes[myn].setscenario.intersection(clustScen)
                u=nodes[myn].stage
                for s in mySS:
                    C3[n,i,w,s] = model.addConstr(quicksum(t1[n,i,w,l,s] for l in LO1)>=bO1[w,u]*TO1[u,w,s]*x[n,i,w,s])
                    C4[n,i,w,s] = model.addConstr(quicksum(t1[n,i,w,l,s] for l in LO1)<=M*x[n,i,w,s])
    C5={}
    for i in IO1:
        for ss in clustScen:
            newnodeset = [nodes[n].eyed for n in nodeindex if ss in nodes[n].setscenario]
            C5[i,ss] = model.addConstr(quicksum(t1[n,i,w,l,ss] or w in WO1 for l in LO1 for n in newnodeset)<=hO1[i]*y[i])
    C6={}
    for i in IO1:
```

```
     for n in nodeset:
        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
        for w in WO1:
            mySS = nodes[myn].setscenario.intersection(clustScen)
            for s in mySS:
                C6[n,i,w,s] = model.addConstr(y[i]>=x[n,i,w,s])

C8={}
for i in IO1:
     for n in nodeset:
        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])

        mySS = nodes[myn].setscenario.intersection(clustScen)
        for s in mySS:
            C8[n,i,w,s] = model.addConstr(quicksum(x[n,i,w,s]  for w in WO1)>=y[i])


C9={}

for i in IO1:
    for n in nodeset:
        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
        mySS = nodes[myn].setscenario.intersection(clustScen)

        for s in mySS:
            if nodes[myn].stage==1:
                C9[n,i,s]=model.addConstr(HH[n,i]==hO1[i]-quicksum(t1[n,i,w,l,s] for w in WO1 for l in LO1))
            else:
                C9[n,i,s]=model.addConstr(HH[n,i]==HH[nodes[myn].fatherid,i]-quicksum(t1[n,i,w,l,s] for w in WO1 for l in LO1))


C10={}
for i in IO1:
    for n in nodeset3:
        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
        mySS = nodes[myn].setscenario.intersection(clustScen)

        for s in mySS:
            C10[n,i,s]=model.addConstr(quicksum(t1[n,i,w,l,s] for w in WO1 for l in LO1)<=HH[nodes[myn].fatherid,i])
```

```
    C11={}
    for i in IO1:
        for n in lastnodes:
            myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])


            C11[n,i]=model.addConstr(HH[nodes[myn].eyed,i]>=z[i])
    C12={}
    for i in IO1:
        C12[i]=model.addConstr(z[i]<=y[i]*M)
    C13={}
    for i in IO1:
        for n in lastnodes:
            myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
            C13[n,i]=model.addConstr(z[i]>=HH[nodes[myn].eyed,i]+(y[i]-1)*M)


    if fix==1:
       C14={}
       for i in IO1:

           if finalXX[iii,i]==1:
              C14[i]=model.addConstr(y[i]==finalXX[iii,i])

    model.update()
    o={}
    o[0]=0
#    o[0]=-quicksum(C[u,w]*x[i,u,w] for u in UO1 for w in WO1 for i in IO1)
    for i in IO1:

        o[0]=o[0]-C*y[i]

    for i in IO1:
        for w in WO1:
            for n in nodeset:
                myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
                mySS = nodes[myn].setscenario.intersection(clustScen)
                for l in LO1:

                    for s in mySS:
                        o[0]=o[0]-float(1.0/len(mySS))*r[w,l]*t1[n,i,w,l,s]
```

```
    for i in IO1:

           o[0]=o[0]-float(1.0/len(mySS))*vO1[i]*z[i]

    if fmode>=1:
        o[1]=0
        for i in IO1:
#             for u in UO1:

            for w in WO1:
                for l in LO1:
                    for n in nodeset:
                        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
                        mySS = nodes[myn].setscenario.intersection(clustScen)
                        for s in mySS:
                            o[1]+=float(1.0/len(mySS))*(EO1[i,w,l]*t1[n,i,w,l,s]*l )


        s2={}
#         s3={}
        for n in nodeset:
            s2[n] = model.addVar(lb=0,ub=GRB.INFINITY)
#             s3[n] = model.addVar(lb=0,ub=GRB.INFINITY)
        model.update()
        cz2={}
##         cz3={}
        for n in nodeset:
            cz2[n] = model.addConstr(o[1]-s2[n]==z2ub)
#             cz3[n] = model.addConstr(o[2]-s3[n]==z3ub)
        model.update()
        for n in nodeset:
            o[0]+=0.001*(1.0/len(nodeset))*(s2[n]/float(finterval1*3))

        if fmode>=2:
            #>clustNum
            o[0]=0
#             clustScen=set([(clustNum-1)*3+1,(clustNum-1)*3+2,(clustNum-1)*3+3])

            for n in IPprim[clustNum]:     ####################stage=0 should be removed##########
```

```
            myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
            myrange=clustScen.intersection(nodes[myn].setscenario)
            ww=float(sum([prob[scen] for scen in myrange])) #range should be the intersection of the scenarios of this node and the scenarios of this cluster
            mystage=nodes[myn].stage
            for i in IO1:

                o[0]-=float(ww)*C*y[i]
                o[0]=o[0]-float(ww)*vO1[i]*z[i]

                for w in WO1:
                    for l in LO1:
                        mySS = nodes[myn].setscenario.intersection(clustScen)
                        for s in mySS:
                            o[0]=o[0]-float(ww)*float(1.0/len(mySS))*r[w,l]*t1[n,i,w,l,s]

        for n in IP2[clustNum]:
            myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
            mystage=nodes[myn].stage
            prr=float(len(nodes[myn].setscenario)/len(SS))

            o[0]-=float(prr)*C*y[i]
            for i in IO1:
                o[0]=o[0]-float(prr)*vO1[i]*z[i]
                for w in WO1:
                    for l in LO1:
                        mySS = nodes[myn].setscenario.intersection(clustScen)
                        for s in mySS:
                             o[0]=o[0]-float(prr)*float(1.0/len(mySS))*r[w,l]*t1[n,i,w,l,s]

        for n in nodeset:
            o[0]+=0.001*(1.0/len(nodeset))*(s2[n]/float(finterval1*3))
    if fmode==3:

        for n in IPprim[clustNum]:
            myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
            mystage=nodes[myn].stage
             for myclust in nodes[myn].setcluster:
            if (clustNum>min(nodes[myn].setcluster)):
                  for i in IO1:


                    o[0]-=float(mu2[k,n,clustNum][i]-mu2[k,n,clustNum-1][i])*HH[n,i]
                    for w in WO1:


                        for ss in  range(len(SS)):
                            o[0]-=float(mu3[k,n,clustNum][i][w][ss]-mu3[k,n,clustNum-1][i][w][ss])*x[n,i,w,SS[ss]]

                            for l in LO1:
                                  print("clustnum>min,i,w,l,ss,SS[ss],k,mystage,clustNum,n",i,w,l,ss,SS[ss],k,mystage,myclust,n)
                                  o[0]-=float(mu[k,n,clustNum][i][w][l][ss]-mu[k,n,clustNum-1][i][w][l][ss])*t1[n,i,w,l,SS[ss]]

      #                    for eta in odes[myn].setcluster.intersection(range(min(nodes[myn].setcluster)+1,max(nodes[myn].setcluster)+1)):
        elif clustNum==min(nodes[myn].setcluster):
                mm=max(nodes[myn].setcluster)      ##########################It can't be as an indices####
                mmi=min(nodes[myn].setcluster)
                      print("mm,n",mm,n)
                for i in IO1:

                    o[0]-=float(mu2[k,n,mmi][i]-mu2[k,n,mm][i])*HH[n,i]
                    for w in WO1:


                        for ss  in range(len(SS)):
                            o[0]-=float(mu3[k,n,mmi][i][w][ss]-mu3[k,n,mm][i][w][ss])*x[n,i,w,SS[ss]]
                            for l in LO1:
                                        print"!!!!!!!!! i,w,l,ss,SS[ss],k,mystage,n,mm,mmi: ",i,w,l,ss,SS[ss],k,mystage,n,mm,mmi

                                 o[0]-=float(mu[k,n,mmi][i][w][l][ss]-mu[k,n,mm][i][w][l][ss])*t1[n,i,w,l,SS[ss]]


    #            o[0]+=0.001*(s2/(finterval1*4)+s3/(finterval2*4))   #######Should be addesd



model.modelSense = GRB.MAXIMIZE
#optimizing the first objective without considering the other two objectives
model.setObjective(o[0])
model.setParam("MIPGap",0.05)
model.setParam("OutputFlag",0)
```

```
    model.update()
    model.optimize()


    Z1=0
    if model.status==GRB.OPTIMAL:
        Z1=model.objVal
#         print ("Optimal z*1:", model.objVal)

        for i in IO1:
            for n in nodeset:
                myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
                mySS = nodes[myn].setscenario.intersection(clustScen)

                for w in WO1:
                    for l in LO1:
                         for ss in mySS:
                            tout[n,i,w,l,ss]=copy.deepcopy(t1[n,i,w,l,ss].x)

        for i in IO1:
            yout[i]=copy.deepcopy(y[i].x)




        for i in IO1:
            for n in nodeset:
                myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
                mySS = nodes[myn].setscenario.intersection(clustScen)

                for w in WO1:

                    for ss in mySS:
                        xout[n,i,w,ss]=copy.deepcopy(x[n,i,w,ss].x)

#         for i in IO2:
```

```
        for n in nodeset:
            for i in IO1:
                Hout[n,i]=copy.deepcopy(HH[n,i].x)
        Z21=0
        for i in IO1:
             for n in nodeset:
                myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
                mySS = nodes[myn].setscenario.intersection(clustScen)
                for w in WO1:
                    for l in LO1:
                        for ss in mySS:
                             ############Should be added#########
                            Z21=Z21+float(1.0/len(mySS))*EO1[i,w,l]*float(t1[n,i,w,l,ss].x)*l



        return (-Z1,Z21,0,tout,yout,Hout,xout)
    else:
        return (-1,-1,-1)


# In[9]:

# Finding the upper bound of Z2

def getUB2(L,W,U,I,T,ss,E,h,bb,F,C):

    myL=copy.deepcopy(L)
    myW=copy.deepcopy(W)
    myU=copy.deepcopy(U)
    myI=copy.deepcopy(I)
    myT={}
    myT=copy.deepcopy(T)

#    ss=copy.deepcopy(s)

    myh={}
    myh=copy.deepcopy(h)
```

```
    vO1=copy.deepcopy(v)

    x={} #binary: assignment of worker to a work discipline in a unit

    for n in nodeset:
        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
        for i in IO1:
            for w in WO1:
                mySS = nodes[myn].setscenario.intersection(clustScen)
                for s in mySS:
                        x[n,i,w,s]=model.addVar(vtype='B',lb=0,ub=1)
    y={}#binary: whether s/o is selected
    for i in IO1:
        y[i]=model.addVar(vtype='B',lb=0,ub=1)
    z={}#binary: whether s/o is selected
    for i in IO1:
        z[i]=model.addVar(lb=0,ub=GRB.INFINITY)

    t1={}#actual time spent
#    for s in ss:
    for i in IO1:
        for n in nodeset:
            myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
#         for u in UO1:
            for w in WO1:
                for l in LO1:
                    mySS = nodes[myn].setscenario.intersection(clustScen)
                    for s in mySS:

                        t1[n,i,w,l,s] = model.addVar(lb=0,ub=GRB.INFINITY)



    HH={}

    for n in nodeset:
        for i in IO1:
            HH[n,i]=model.addVar(lb=0,ub=GRB.INFINITY)

    model.update()
```

```
    C2={}
#    for s in ss:
#     for u in UO1:
    for w in WO1:
        for l in LO1:
            for n in nodeset:
                myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
                mySS = nodes[myn].setscenario.intersection(clustScen)
                u=nodes[myn].stage
                for s in mySS:
                    C2[n,w,l,s] = model.addConstr((quicksum(t1[n,i,w,l,s]*alpha[l] for i in IO1 for l in LO1))>=TO1[u,w,s])
#                     C2[n,w,l,s] = model.addConstr(quicksum(t1[n,i,w,l,s]*alpha[l] for i in IO1 for l in LO1))>=TO1[u,w,s]

#                for ss in SS:
#                     C2[u,w,l] = model.addConstr(quicksum(t1[i,u,w,l,ss]*alpha[l] for i in IO1 for l in LO1)>=TO1[u,w,ss])
    C3={}
    C4={}
    for i in IO1:
        for w in WO1:
            for n in nodeset:
                myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
                mySS = nodes[myn].setscenario.intersection(clustScen)
                u=nodes[myn].stage
                for s in mySS:
                    C3[n,i,w,s] = model.addConstr(quicksum(t1[n,i,w,l,s] for l in LO1)>=bO1[w,u]*TO1[u,w,s]*x[n,i,w,s])
                    C4[n,i,w,s] = model.addConstr(quicksum(t1[n,i,w,l,s] for l in LO1)<=M*x[n,i,w,s])
    C5={}
    for i in IO1:
        for ss in clustScen:
            newnodeset = [nodes[n].eyed for n in nodeindex if ss in nodes[n].setscenario]
            C5[i,ss] = model.addConstr(quicksum(t1[n,i,w,l,ss] or w in WO1 for l in LO1 for n in newnodeset)<=hO1[i]*y[i])
    C6={}
    for i in IO1:
         for n in nodeset:
            myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
            for w in WO1:
                mySS = nodes[myn].setscenario.intersection(clustScen)
                for s in mySS:
                    C6[n,i,w,s] = model.addConstr(y[i]>=x[n,i,w,s])
```

```
C8={}
for i in IO1:
     for n in nodeset:
        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])

        mySS = nodes[myn].setscenario.intersection(clustScen)
        for s in mySS:
            C8[n,i,w,s] = model.addConstr(quicksum(x[n,i,w,s]  for w in WO1)>=y[i])


C9={}

for i in IO1:
    for n in nodeset:
        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
        mySS = nodes[myn].setscenario.intersection(clustScen)

        for s in mySS:
            if nodes[myn].stage==1:
                C9[n,i,s]=model.addConstr(HH[n,i]==hO1[i]-quicksum(t1[n,i,w,l,s] for w in WO1 for l in LO1))
            else:
                C9[n,i,s]=model.addConstr(HH[n,i]==HH[nodes[myn].fatherid,i]-quicksum(t1[n,i,w,l,s] for w in WO1 for l in LO1))


C10={}
for i in IO1:
    for n in nodeset3:
        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
        mySS = nodes[myn].setscenario.intersection(clustScen)

        for s in mySS:
            C10[n,i,s]=model.addConstr(quicksum(t1[n,i,w,l,s] for w in WO1 for l in LO1)<=HH[nodes[myn].fatherid,i])


C11={}
for i in IO1:
    for n in lastnodes:
        myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])


        C11[n,i]=model.addConstr(HH[nodes[myn].eyed,i]>=z[i])
```

```
    C12={}
    for i in IO1:
        C12[i]=model.addConstr(z[i]<=y[i]*M)
    C13={}
    for i in IO1:
        for n in lastnodes:
            myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
            C13[n,i]=model.addConstr(z[i]>=HH[nodes[myn].eyed,i]+(y[i]-1)*M)
    model.update()
    o2={}
    o2[0]=0
    for i in IO1:
        for n in nodeset:
                myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
                mySS = nodes[myn].setscenario.intersection(clustScen)
                for w in WO1:
                    for l in LO1:
                        for ss in mySS:
                             ############Should be added#########
                            o2[0]=o2[0]+float(1.0/len(mySS))*EO1[i,w,l]*(t1[n,i,w,l,ss])*l

    model.modelSense = GRB.MAXIMIZE
    #optimizing the first objective without considering the other two objectives
    model.setObjective(o2[0])
    model.setParam("MIPGap",0.05)
    model.setParam("OutputFlag",0)
    model.update()
    model.optimize()


    Z2=0
    if model.status==GRB.OPTIMAL:
        Z2=model.objVal
        print ("=======Optimal z*2==========:", model.objVal)

        Z21=0
#    o[0]=-quicksum(C[u,w]*x[i,u,w] for u in UO1 for w in WO1 for i in IO1)
        for i in IO1:

            Z21=Z21+C*y[i].x
```

```
        for i in IO1:
            for w in WO1:
                for n in nodeset:
                    myn=sum([ii for ii in range(numNode) if nodes[ii].eyed==n])
                    mySS = nodes[myn].setscenario.intersection(clustScen)
                    for l in LO1:

                        for s in mySS:
                            Z21=Z21+float(1.0/len(mySS))*r[w,l]*t1[n,i,w,l,s].x
        for i in IO1:

               Z21=Z21+float(1.0/len(mySS))*vO1[i]*z[i].x


        return (Z2,Z21,0)
    else:
        print("inf")
        return (-1)

# In[10]:

clustScenario={}
for i in range (1,numClust+1):

    #get the scenarios of cluster i
    x=(i-1)*3
    clustScenario[i]=[x+1,x+2,x+3]
#    print("clustScenario[i]",i,clustScenario[i])


# In[ ]:

ElementofParetoSet={}
payoffClust={}
worstZ1={}
clustScenario={}
for i in range(1,numClust+1):
    myTableau=[[0,0,0],[0,0,0]]
    #get the scenarios of cluster i
    x=(i-1)*3
    clustScenario[i]=[x+1,x+2,x+3]
```

```
    jay0=getOV1(I,U,W,L,E,clustScenario[i],T,h,bb,C,r,alpha,0,0,0,0,0,0,i,0,0,0,0)
    print("jay0", jay0[0],jay0[1])

    myTableau[0][0]+=jay0[0]     ##################################myTableau[0][0]=jay0[0] ###############
    myTableau[0][1]+=jay0[1]
    jay1=getOV2(I,U,W,L,E,clustScenario[i],T,h,bb,C,r,v,alpha,i)

#         jay2=getUB3(I,U,W,L,Lprime,E,ss,T,h,bb)
#        print("jay2", jay2[0],jay2[1],jay2[2])
        #note: devided by 3 for expected value
    myTableau[1][0]+=(jay1[1])
    myTableau[1][1]+=(jay1[0])
#        myTableau[1][2]+=float(jay1[2])/3
#        myTableau[2][0]+=float(jay2[0])/3
#        myTableau[2][1]+=float(jay2[1])/3
#        myTableau[2][2]+=float(jay2[2])/3

    f1min=min(myTableau[0][0],myTableau[1][0])
    f2min=min(myTableau[0][1],myTableau[1][1])
#     f3min=min(myTableau[0][2],myTableau[1][2],myTableau[2][2])
    f1max=max(myTableau[0][0],myTableau[1][0])
    f2max=max(myTableau[0][1],myTableau[1][1])
#     f3max=max(myTableau[0][2],myTableau[1][2],myTableau[2][2])
    f1interval=float(f1max-f1min)/5
    f2interval=float(f2max-f2min)/5
    print("f2interval",f2interval)
#     f3interval=float(f3max-f3min)/3
#     print("f3interval",f3interval)
    f1points=[0,0,0,0,0,0]
    f2points=[0,0,0,0,0,0]
#     f3points=[0,0,0,0]
    for ii in range(5):
        f1points[ii]=f1min+ii*f1interval
        f2points[ii]=f2min+ii*f2interval
#         f3points[ii]=f3min+ii*f3interval
    f1points[5]=f1max
    worstZ1[i]=f1max #upper bound of Z1 to be used in the next cell
    f2points[5]=f2max
```

```
        payoffClust[i,ii]=[f2points[ii]]  ##Should be removed##
###            jay0=getOV1(I,U,W,L,E,clustScenario,T,h,bb,C,r,alpha,2,f2points[ii],f3points[jj],f2interval,f3interval,0,i)  ##
#             ElementofParetoSet[i,ii,jj]=[jay0[0],jay0[1],jay0[2]]
#    print("gridClust#",i,payoffClust[i])
#    print("upper bound for Z1: ",worstZ1[i])


# In[79]:

# Final grid using lagrangian
import time
finalGrid={}
finalX={}
finalXX={}

diff={}
totalTic=time.time()
listN2prim=list(N2prim)
# for iii in range(4): #z2
#     for jjj in range(4):
#         for ii in I:
#             for u in U:
#                 for w in W:
#                     fianlXX[iii,jjj,ii,u,w]=0

for i in range(6): #z2


    isConverged=[[[[[0 for ss in range(3)]for l in L]for w in W]for ii in I]for n in range(len(N2prim))]   #[n][ii][w][l][ss]
    Smatrix={}
    Smatrix2={}
    Smatrix3={}
    #STEP0 (Initialization)
    step0Tic=time.time()
    print("**********STEP 0 INITIATED FOR i:",i)
    k=0
    mu={}
    mu2={}
    mu3={}
```

```
for n in N2prim:
    myn=min([ii for ii in range(numNode) if nodes[ii].eyed==n])
    numcl=len(nodes[myn].setcluster)
         u=nodes[myn].stage
    for c in nodes[myn].setcluster:
             print(clustScenario[c])
        mu[k,n,c]=[[[[0 for ss in clustScenario[c]]for l in L]for w in W]for ii in I]
        mu2[k,n,c]=[0 for ii in I]
        mu3[k,n,c]=[[[0 for ss in clustScenario[c]]for w in W]for ii in I]

for c in range(1,numClust+1):
    jay[i,c]=getOV1(I,U,W,L,E,clustScenario[c],T,h,bb,C,r,alpha,2,payoffClust[c,i][0],f2interval,mu,mu2,mu3,c,k,2,0,i)
    print("cluster=",c)

ZZ1=sum([jay[i,c][0] for c in range(1,numClust+1)])
ZZ2=sum([jay[i,c][1] for c in range(1,numClust+1)])

finalGrid[i+1]=[ZZ1,float(ZZ2/numClust)]
step0Toc=time.time()
print("=====END OF STEP 0. Total elapsed time is:",step0Toc-step0Tic)
flag=0

alfa={}
while flag<1:
    print ("iteration #",k,":")
    # STEP 1
    print("**********STEP 1 INITIATED FOR i,k:",i,k)
    step1Tic=time.time()
    for n in N2prim:
        myn=min([ii for ii in range(numNode) if nodes[ii].eyed==n])
             u=nodes[myn].stage
        state=nodes[myn].scenario
        clu=sorted(nodes[myn].setcluster)
        numcl=len(clu)

        Smatrix[k,n]=[[[[[0 for ss in clustScenario[clu[p]]] for p in range(len(clu))] for l in L] for w in W] for ii in I]
        Smatrix2[k,n]=[[0 for p in range(len(clu))] for ii in I]
        Smatrix3[k,n]=[[[[0 for ss in clustScenario[clu[p]]] for p in range(len(clu))] for w in W] for ii in I]
             print(Smatrix)
          for ii in I:
```

```
            for a in range (numcl-1):
                for ss in range(len(clustScenario[clu[a]])):
                    myc=clu[a]
                    mycplus=clu[a+1]
                    mys=clustScenario[myc][ss]
                    mysplus=clustScenario[mycplus][ss]
                    for ii in I:
                        Smatrix2[k,n][ii][a]= float (jay[i,myc][5][n,ii]-jay[i,mycplus][5][n,ii])
                        for w in W:

                            Smatrix3[k,n][ii][w][a][ss]=                            float (jay[i,myc][6][n,ii,w,mys]-jay[i,mycplus][6][n,ii,w,mysplus])
                            for l in L:
                                Smatrix[k,n][ii][w][l][a][ss]=                                  float (jay[i,myc][3][n,ii,w,l,mys]-jay[i,mycplus][3][n,ii,w,l,mysplus])
                for  ss in range(len(clustScenario[clu[numcl-1]])):
                    mys=clustScenario[clu[numcl-1]][ss]
                    mysplus=clustScenario[clu[0]][ss]
                    for ii in I:
                        Smatrix2[k,n][ii][numcl-1]=float(jay[i,clu[numcl-1]][5][n,ii]-jay[i,clu[0]][5][n,ii])
                        for w in W:
                            Smatrix3[k,n][ii][w][numcl-1][ss]=float(jay[i,clu[numcl-1]][6][n,ii,w,mys]-jay[i,clu[0]][6][n,ii,w,mysplus])
                            for l in L:
                                Smatrix[k,n][ii][w][l][numcl-1][ss]=float(jay[i,clu[numcl-1]][3][n,ii,w,l,mys]-jay[i,clu[0]][3][n,ii,w,l,mysplus])



        step1Toc=time.time()
        print("===END OF STEP 1 for k=",k,"total time is:",step1Toc-step1Tic)
#             print(Smatrix)

        #STEP 2
        print("**********STEP 2 INITIATED FOR i,k:",i,k)
        step2Tic=time.time()
        alfa[k]=0.0003
        for n in N2prim:
            myn=min([ii for ii in range(numNode) if nodes[ii].eyed==n])
#               u=nodes[myn].stage
            state=nodes[myn].scenario
            clu=sorted(nodes[myn].setcluster)
            numcl=len(clu)
```

```
            for c in nodes[myn].setcluster:
#
                mu[k+1,n,c]=[[[[0 for ss in clustScenario[c]]for l in L]for w in W]for ii in I]
                mu2[k+1,n,c]=[0 for ii in I]
                mu3[k+1,n,c]=[[[0 for ss in clustScenario[c]]for w in W]for ii in I]

            for a in range(len(clu)):
                myp=clu[a]
#
                for ii in I:
                    for w in W:
                        for l in L:
                            for ss in range(len(clustScenario[clu[a]])):
                                mys=clustScenario[clu[a]][ss]
                                #calc norm manually instead of LA.norm()
                                mynorm=0
                                mynorm2=0
                                mynorm3=0
                                myel=0
                                myel2=0
                                myel3=0
                                if isConverged[listN2prim.index(n)][ii][w][l][ss]==0:
                                    for pp in range(numcl):
#                                           print("myel",k,u,n,ii,w,l,pp,ss)
                                        myel=float(Smatrix[k,n][ii][w][l][pp][ss])
                                        myel2=float(Smatrix2[k,n][ii][pp])
                                        myel3=float(Smatrix3[k,n][ii][w][pp][ss])
#                                             print(myel)
                                        mynorm+=float(myel*myel)
                                        mynorm2+=float(myel2*myel2)
                                        mynorm3+=float(myel3*myel3)
                                    if mynorm<0.0001:
                                        mynorm=0
                                    if mynorm2<0.0001:
                                        mynorm2=0
                                    if mynorm3<0.0001:
                                        mynorm3=0
#                                     if mynorm<0:
#                                         print("=============mynorm*************",mynorm)
#                                     if mynorm2<0:
```

```
      print("====================mynorm2**********************",mynorm2)
 if float(mynorm)==0  and float(mynorm2)==0 and float(mynorm3)==0:
     isConverged[listN2prim.index(n)][ii][w][l][ss]==1
     break

       else:
           mu[k+1,n,myp][ii][w][l][ss]=mu[k,n,myp][ii][w][l][ss]+float(alfa[k]*((worstZ1[myp]-jay[myp][0])/mynorm)*Smatrix[k,n][ii][w][l][a][ss])
 if float(mynorm)==0 and float(mynorm2)!=0 and float(mynorm3)==0:

         print("mynorm==0")
         mu[k+1,n,myp][ii][w][l][ss]=mu[k,n,myp][ii][w][l][ss]+float(alfa[k]*((worstZ1[myp]-jay[myp][0])/mynorm)*Smatrix[k,n][ii][w][l][a][ss])
     mu2[k+1,n,myp][ii]=float (mu2[k,n,myp][ii])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm2)*Smatrix2[k,n][ii][a])

 if float(mynorm)!=0 and float(mynorm2)==0 and float(mynorm3)==0:
       mu[k+1,n,myp][ii][w][l][ss]=float (mu[k,n,myp][ii][w][l][ss])+float((alfa[k]*(-jay[i,myp][0]+worstZ1[myp])/mynorm)*Smatrix[k,n][ii][w][l][a][ss])


           print("mynorm2==0")

         mu2[k+1,n,myp][ii]=mu2[k,n,myp][ii]+float(alfa[k]*((worstZ1[myp]-jay[myp][0])/mynorm2)*Smatrix2[k,n][ii][a])

 if float(mynorm)!=0 and float(mynorm2)!=0 and float(mynorm3)==0:
       mu[k+1,n,myp][ii][w][l][ss]=float (mu[k,n,myp][ii][w][l][ss])+float((alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm)*Smatrix[k,n][ii][w][l][a][ss]))
       mu2[k+1,n,myp][ii]=float(mu2[k,n,myp][ii])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm2)*Smatrix2[k,n][ii][a])

 if float (mynorm)==0 and float(mynorm2)==0 and float(mynorm3)!=0:
      mu3[k+1,n,myp][ii][w][ss]=float (mu3[k,n,myp][ii][w][ss])+float((alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm3)*Smatrix3[k,n][ii][w][a][ss]))

 if float (mynorm)!=0 and float(mynorm2)==0 and float(mynorm3)!=0:
      mu3[k+1,n,myp][ii][w][ss]=float (mu3[k,n,myp][ii][w][ss])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm3)*Smatrix3[k,n][ii][w][a][ss])

      mu[k+1,n,myp][ii][w][l][ss]=float (mu[k,n,myp][ii][w][l][ss])+float((alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm)*Smatrix[k,n][ii][w][l][a][ss]))
 if float(mynorm)==0 and float(mynorm2)!=0 and float(mynorm3)!=0:
      mu2[k+1,n,myp][ii]=float(mu2[k,n,myp][ii])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm2)*Smatrix2[k,n][ii][a])
      mu3[k+1,n,myp][ii][w][ss]=float(mu3[k,n,myp][ii][w][ss])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm3)*Smatrix3[k,n][ii][w][a][ss])


 if float(mynorm)!=0 and float(mynorm2)!=0 and float(mynorm3)!=0:




                                 mu[k+1,n,myp][ii][w][l][ss]=float (mu[k,n,myp][ii][w][l][ss])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm)*Smatrix[k,n][ii][w][l][a][ss])
                                 mu2[k+1,n,myp][ii]=float(mu2[k,n,myp][ii])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm2)*Smatrix2[k,n][ii][a])
                                 mu3[k+1,n,myp][ii][w][ss]=float(mu3[k,n,myp][ii][w][ss])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm3)*Smatrix3[k,n][ii][w][a][ss])


step2Toc=time.time()
print("===END OF STEP 2 for k=",k,"total time is:",step2Toc-step2Tic)

#STEP 3
print("**********STEP 3 INITIATED FOR i,k:",i,k)
step3Tic=time.time()

k+=1
for c in range(1,numClust+1):

    jay[i,c]=getOV1(I,U,W,L,E,clustScenario[c],T,h,bb,C,r,alpha,3,payoffClust[c,i][0],f2interval,mu,mu2,mu3,c,k,2,0,i)
    print("cluster",c)
    lastZZ1=finalGrid[i+1][0]
    ZZ1=sum([jay[i,cc][0] for cc in range(1,numClust+1)])
    ZZ2=sum([jay[i,cc][1] for cc in range(1,numClust+1)])

    finalGrid[i+1]=[ZZ1,float(ZZ2/numClust)]
    print("i=",i+1, "finalGrid[i+1",finalGrid[i+1])

    for ii in I:


        finalX[i,c,ii]=jay[i,c][4][ii]

                    print("============finalX","i=",i,"j=",j,"c",c,"finalX",finalX[i,j,c,ii,u,w])
step3Toc=time.time()
print("===END OF STEP 3 for k=",k-1,"total time is:",step3Toc-step3Tic)


#STEP 4
#check if the value of Z1 has improved

if k==1:
    for ii in I:
```

```
                if sum([finalX[i,c,ii]for c in range(1, numClust+1)])>float(numClust/2):
                            finalXX[i,ii]=1
                else:
                            finalXX[i,ii]=0
            for ii in I:
                print("i=",i,"ii=",ii,"value=",finalXX[i,ii])
        if lastZZ1-finalGrid[i+1][0]<=0.05*lastZZ1 or k==2:
            flag=1



totalToc=time.time()
print("Total Elapsed Time is:",totalToc-totalTic)
print("Final grid: ",finalGrid)


# In[82]:

import time
finalGrid={}

final2X={}
diff={}
totalTic=time.time()
listN2prim=list(N2prim)
for i in range(4): #z2
   #z3
    isConverged=[[[[[0 for ss in range(3)]for l in L]for w in W]for ii in I]for n in range(len(N2prim))]   #[n][ii][w][l][ss]
    Smatrix={}
    Smatrix2={}
    #STEP0 (Initialization)
    step0Tic=time.time()
    print("**********STEP 0 INITIATED FOR i:",i)
    k=0
    mu={}
    mu2={}
    mu3={}
    jay={}
```

```
    for n in N2prim:
        myn=min([ii for ii in range(numNode) if nodes[ii].eyed==n])
        numcl=len(nodes[myn].setcluster)
#           u=nodes[myn].stage
        for c in nodes[myn].setcluster:
#               print(clustScenario[c])
            mu[k,n,c]=[[[[0 for ss in clustScenario[c]]for l in L]for w in W]for ii in I]
            mu2[k,n,c]=[0 for ii in I]
            mu3[k,n,c]=[[[0 for ss in clustScenario[c]]for w in W]for ii in I]

    for c in range(1,numClust+1):
        jay[i,c]=getOV1(I,U,W,L,E,clustScenario[c],T,h,bb,C,r,alpha,2,payoffClust[c,i][0],f2interval,mu,mu2,mu3,c,k,1,finalXX,i)
        print("cluster",c)
    ZZ1=sum([jay[i,c][0] for c in range(1,numClust+1)])
    ZZ2=sum([jay[i,c][1] for c in range(1,numClust+1)])

    finalGrid[i+1]=[ZZ1,float(ZZ2/numClust)]
    step0Toc=time.time()
    print("=====END OF STEP 0. Total elapsed time is:",step0Toc-step0Tic)
    flag=0

    alfa={}
    while flag<1:
        print ("iteration #",k,":")
        # STEP 1
        print("**********STEP 1 INITIATED FOR i,k:",i,k)
        step1Tic=time.time()
        for n in N2prim:
            myn=min([ii for ii in range(numNode) if nodes[ii].eyed==n])
#               u=nodes[myn].stage
            state=nodes[myn].scenario
            clu=sorted(nodes[myn].setcluster)
            numcl=len(clu)

            Smatrix[k,n]=[[[[0 for ss in clustScenario[clu[p]]] for p in range(len(clu))] for l in L] for w in W] for ii in I]
            Smatrix2[k,n]=[[0 for p in range(len(clu))] for ii in I]
            Smatrix3[k,n]=[[[0 for ss in clustScenario[clu[p]]] for p in range(len(clu))]  for w in W] for ii in I]
#               print(Smatrix)
#             for ii in I:
#                 for w in W:
#                     for l in L:
```

```
    for a in range (numcl-1):
        for ss in range(len(clustScenario[clu[a]])):
            myc=clu[a]
            mycplus=clu[a+1]
            mys=clustScenario[myc][ss]
            mysplus=clustScenario[mycplus][ss]
            for ii in I:
                Smatrix2[k,n][ii][a]= float (jay[i,myc][5][n,ii]-jay[i,mycplus][5][n,ii])
                for w in W:

                    Smatrix3[k,n][ii][w][a][ss]=                        float (jay[i,myc][6][n,ii,w,mys]-jay[i,mycplus][6][n,ii,w,mysplus])
                    for l in L:
                        Smatrix[k,n][ii][w][l][a][ss]=                          float (jay[i,myc][3][n,ii,w,l,mys]-jay[i,mycplus][3][n,ii,w,l,mysplus])
    for  ss in range(len(clustScenario[clu[numcl-1]])):
        mys=clustScenario[clu[numcl-1]][ss]
        mysplus=clustScenario[clu[0]][ss]
        for ii in I:
            Smatrix2[k,n][ii][numcl-1]=float(jay[i,clu[numcl-1]][5][n,ii]-jay[i,clu[0]][5][n,ii])
            for w in W:
                Smatrix3[k,n][ii][w][numcl-1][ss]=float(jay[i,clu[numcl-1]][6][n,ii,w,mys]-jay[i,clu[0]][6][n,ii,w,mysplus])
                for l in L:
                    Smatrix[k,n][ii][w][l][numcl-1][ss]=float(jay[i,clu[numcl-1]][3][n,ii,w,l,mys]-jay[i,clu[0]][3][n,ii,w,l,mysplus])


step1Toc=time.time()
print("===END OF STEP 1 for k=",k,"total time is:",step1Toc-step1Tic)
      print(Smatrix)

#STEP 2
print("**********STEP 2 INITIATED FOR i,k:",i,k)
step2Tic=time.time()
alfa[k]=0.0003
for n in N2prim:
    myn=min([ii for ii in range(numNode) if nodes[ii].eyed==n])
         u=nodes[myn].stage
    state=nodes[myn].scenario
    clu=sorted(nodes[myn].setcluster)
    numcl=len(clu)
    for c in nodes[myn].setcluster:

            mu[k+1,n,c]=[[[[0 for ss in clustScenario[c]]for l in L]for w in W]for ii in I]
            mu2[k+1,n,c]=[0 for ii in I]
            mu3[k+1,n,c]=[[[0 for ss in clustScenario[c]]for w in W]for ii in I]
        for a in range(len(clu)):
            myp=clu[a]

            for ii in I:
                for w in W:
                    for l in L:
                        for ss in range(len(clustScenario[clu[a]])):
                            mys=clustScenario[clu[a]][ss]
                            #calc norm manually instead of LA.norm()
                            mynorm=0
                            mynorm2=0
                            mynorm3=0
                            myel=0
                            myel2=0
                            myel3=0
                            if isConverged[listN2prim.index(n)][ii][w][l][ss]==0:
                                for pp in range(numcl):
                                        print("myel",k,u,n,ii,w,l,pp,ss)
                                    myel=float(Smatrix[k,n][ii][w][l][pp][ss])
                                    myel2=float(Smatrix2[k,n][ii][pp])
                                    myel3=float(Smatrix3[k,n][ii][w][pp][ss])
                                        print(myel)
                                    mynorm+=float(myel*myel)
                                    mynorm2+=float(myel2*myel2)
                                    mynorm3+=float(myel3*myel3)
                                if mynorm<0.0001:
                                    mynorm=0
                                if mynorm2<0.0001:
                                    mynorm2=0
                                if mynorm3<0.0001:
                                    mynorm3=0
                                  if mynorm<0:
                                      print("=============mynorm*************",mynorm)
                                  if mynorm2<0:
                                      print("====================mynorm2**********************",mynorm2)
                                if float(mynorm)==0  and float(mynorm2)==0 and float(mynorm3)==0:
                                    isConverged[listN2prim.index(n)][ii][w][l][ss]==1
                                    break
```

```
                                else:
                                    mu[k+1,n,myp][ii][w][l][ss]=mu[k,n,myp][ii][w][l][ss]+float(alfa[k]*((worstZ1[myp]-jay[myp][0])/mynorm)*Smatrix[k,n][ii][w][l][a][ss])
                            if float(mynorm)==0 and float(mynorm2)!=0 and float(mynorm3)==0:

                                    print("mynorm==0")
                                    mu[k+1,n,myp][ii][w][l][ss]=mu[k,n,myp][ii][w][l][ss]+float(alfa[k]*((worstZ1[myp]-jay[myp][0])/mynorm)*Smatrix[k,n][ii][w][l][a][ss])
                                mu2[k+1,n,myp][ii]=float (mu2[k,n,myp][ii])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm2)*Smatrix2[k,n][ii][a])

                            if float(mynorm)!=0 and float(mynorm2)==0 and float(mynorm3)==0:
                                mu[k+1,n,myp][ii][w][l][ss]=float (mu[k,n,myp][ii][w][l][ss])+float((alfa[k]*(-jay[i,myp][0]+worstZ1[myp])/mynorm)*Smatrix[k,n][ii][w][l][a][ss])


                                    print("mynorm2==0")

                                mu2[k+1,n,myp][ii]=mu2[k,n,myp][ii]+float(alfa[k]*((worstZ1[myp]-jay[myp][0])/mynorm2)*Smatrix2[k,n][ii][a])

                            if float(mynorm)!=0 and float(mynorm2)!=0 and float(mynorm3)==0:
                                mu[k+1,n,myp][ii][w][l][ss]=float (mu[k,n,myp][ii][w][l][ss])+float((alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm)*Smatrix[k,n][ii][w][l][a][ss]))
                                mu2[k+1,n,myp][ii]=float(mu2[k,n,myp][ii])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm2)*Smatrix2[k,n][ii][a])

                            if float (mynorm)==0 and float(mynorm2)==0 and float(mynorm3)!=0:
                                mu3[k+1,n,myp][ii][w][ss]=float (mu3[k,n,myp][ii][w][ss])+float((alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm3)*Smatrix3[k,n][ii][w][a][ss]))

                            if float (mynorm)!=0 and float(mynorm2)==0 and float(mynorm3)!=0:
                                mu3[k+1,n,myp][ii][w][ss]=float (mu3[k,n,myp][ii][w][ss])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm3)*Smatrix3[k,n][ii][w][a][ss])

                                mu[k+1,n,myp][ii][w][l][ss]=float (mu[k,n,myp][ii][w][l][ss])+float((alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm)*Smatrix[k,n][ii][w][l][a][ss]))
                            if float(mynorm)==0 and float(mynorm2)!=0 and float(mynorm3)!=0:
                                mu2[k+1,n,myp][ii]=float(mu2[k,n,myp][ii])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm2)*Smatrix2[k,n][ii][a])
                                mu3[k+1,n,myp][ii][w][ss]=float(mu3[k,n,myp][ii][w][ss])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm3)*Smatrix3[k,n][ii][w][a][ss])


                            if float(mynorm)!=0 and float(mynorm2)!=0 and float(mynorm3)!=0:

                                mu[k+1,n,myp][ii][w][l][ss]=float (mu[k,n,myp][ii][w][l][ss])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm)*Smatrix[k,n][ii][w][l][a][ss])
                                mu2[k+1,n,myp][ii]=float(mu2[k,n,myp][ii])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm2)*Smatrix2[k,n][ii][a])
                                mu3[k+1,n,myp][ii][w][ss]=float(mu3[k,n,myp][ii][w][ss])+float(alfa[k]*((-jay[i,myp][0]+worstZ1[myp])/mynorm3)*Smatrix3[k,n][ii][w][a][ss])
    step2Toc=time.time()
    print("===END OF STEP 2 for k=",k,"total time is:",step2Toc-step2Tic)

    #STEP 3

        print("**********STEP 3 INITIATED FOR i,k:",i,k)
        step3Tic=time.time()

        k+=1

        for c in range(1,numClust+1):
            jay[i,c]=getOV1(I,U,W,L,E,clustScenario[c],T,h,bb,C,r,alpha,3,payoffClust[c,i][0],f2interval,mu,mu2,mu3,c,k,1,finalXX,i)
            print("cluster",c)
            lastZZ1=finalGrid[i+1][0]

            ZZ1=sum([jay[i,c][0] for c in range(1,numClust+1)])
            ZZ2=sum([jay[i,c][1] for c in range(1,numClust+1)])

            finalGrid[i+1]=[ZZ1,float(ZZ2/numClust)]
            print("i=",i+1, "finalGrid[i+1",finalGrid[i+1])
        for c in range(1,numClust+1):
            for ii in I:

#                           print("jay[4]",jay[i,c][4][ii,u,w], "i=",i,"c=",c,"ii=",ii,"u=",u,"w=",w)
                final2X[i,c,ii]=jay[i,c][4][ii]

#                            print("============finalX","i=",i,"j=",j,"c",c,"finalX",finalX[i,j,c,ii,u,w])
        step3Toc=time.time()
        print("===END OF STEP 3 for k=",k-1,"total time is:",step3Toc-step3Tic)


        #STEP 4
        #check if the value of Z1 has improved



        if lastZZ1-finalGrid[i+1][0]<=0.05*lastZZ1 or k==5:
            flag=1

totalToc=time.time()
print("Total Elapsed Time is:",totalToc-totalTic)
print("Final grid: ",finalGrid)
```

VITA

Fahimeh Rahmanniyay was born in Iran, on March 25, 1987. She got the diploma in mathematics at Farzanegan high school which is, in fact, affiliated with the National Organization for Development of Exceptional Talents (NODET). After finishing high-school, she succeeded to pass the Nationwide Entrance Exam for undergraduate level and ranked 320 among 400000 participants. She completed undergraduate studies in the industrial engineering program at Sharif University of Technology, the best university in the fields of engineering and fundamental sciences in Iran, in 2009. By then, she continued to study in social and economic systems -- operation research program in University of Tehran on Master's level. University of Tehran is one of top three universities of Iran. She received her master degree in September 2012. Fahimeh accepted a graduate research assistantship position at The University of Tennessee, Knoxville, in the industrial and systems engineering program in August 2015. She is going to continue her education for PhD level in Business analytics and statistics at Haslam Collage of Business at UTK.