A MIXED INTEGER LINEAR PROGRAMMING APPROACH FOR DEVELOPING SALARY ADMINISTRATION SYSTEMS

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Taner Cokyasar

December 2016

Copyright © 2016 by Taner Cokyasar

All rights reserved. This work or any part of the work may not be copied or used without the written permission of the author.

DEDICATIONS

I dedicate this work to Turkish Ministry of National Education who has given me the golden opportunity of studying abroad by financially supporting me and to my lovely wife, Pakize, who has sacrificed her youth years for my academic life by leaving all her family behind and put up with their longings. I also want to thank my father-in-law, Osman Durak and my wife's relative, Emin Erdurcan for supporting me during my education. I cannot bring my dedications to an end without expressing my sincere appreciations to Dr. Alberto Garcia, who has been the great face of the University of Tennessee, Knoxville to me during my Master's program.

ACKNOWLEDGEMENTS

I don’t know how much appreciation is really enough for everything Dr. Alberto Garcia taught to me. He has been the greatest mentor I have ever had in my academic life. Without Dr. Garcia, I would not even get any close to accomplish my study in this field. In addition to Dr. Garcia, I also owe a great debt of gratitude for:

Dr. Rifat Kurban	Erciyes University
Ahmet Nusret Toprak	Erciyes University
Masoud Barah	University of Tennessee, Knoxville

ABSTRACT

Determining salary increases of executive personnel is a challenging decision process for many companies. Salary administration policies that aid in the determination of salary increases and other compensation benefits have a wide variety of advantages for both a company and its employees. This thesis develops a mathematical programming approach to create a salary administration system that recognizes the importance of performance and potential of employees for future promotions as major components of a salary increase policy for executive personnel. A number of companies all over the world use salary administration systems that integrate work performance and potential for advancement to develop compensation packages that include benefits in addition to the base salary of their executive personnel. Some of these systems aid decision making concerning salary increase percentages, frequency of salary increases, and guidelines for promotion. The specific policy considered in this thesis assigns salary increase categories and intervals between successive salary increases based on performance and potential assessments. Two mixed integer linear programming models are formulated to assign personnel to salary increase categories and to determine intervals. Solution procedures are clearly illustrated based on a hypothetical application. The optimization toolbox of the commercial software, MATLAB, is used as the problem solver.

TABLE OF CONTENTS

CHAPTER I

BACKGROUND AND GOALS 1

1.1. Introduction 1

1.2. Background 2

1.3. Problem Statement 4

1.4. Study Goals 5

1.5. Approaches 5

1.5.1. Mixed Integer Linear Programming (MILP) 6

1.5.2. MATLAB's 'intlinprog' Algorithm 7

1.5.3. The Branch-and-Bound (B&B) Method 8

1.5.4. The Cutting Plane Method 9

1.5.5. MATLAB's Heuristics Method 10

1.5.6. Thesis Organization 11

CHAPTER II

LITERATURE REVIEW 12

2.1. OR Perspective of Salary Administration 12

2.2. PFP Based Salary Increase Systems 13

2.3. Performance Measurement Guide 15

CHAPTER III

MODEL DEVELOPMENT 18

3.1. Conceptual Approach 18

3.2. Phase One: Data Collection & Preparation 20

3.2.1. Assumptions for Data Preparation 21

3.3. Phase Two: Model Development & Optimization Process 22

3.3.1. Model I 22

3.3.2. Model II 32
CHAPTER IV
SAMPLE APPLICATION 39
4.1. Assumptions for Parameters 39
4.2. Sample Solution via Branch & Bound Method 41
4.3. Sample Solution via Cutting-Plane Method 45
4.4. Sample Solution via MATLAB's Heuristics Method 46
4.5. Overall Commentary on Solution Methods 48
CHAPTER V
SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS 51
5.1. Summary 51
5.2. Conclusions 51
5.3. Recommendations 52
BIBLIOGRAPHY 53
APPENDICES 57
A. Performance Measurement Guide 58
B. Salary Administration Problem Solution Code 62
VITA 71

LIST OF TABLES

Table 1: Sample Data 21
Table 2: Parameter Value Assumptions 40
Table 3: Comparison Table via B&B Method 45
Table 4: Comparison Table via Addition of Cutting-Plane Method 46
Table 5: Comparison Table via Addition of Heuristics Method 48

LIST OF FIGURES

Figure 1: Salary Optimization Guideline 19
Figure 2: Sample Data Solution via B&B Method 43
Figure 3: B&B Tree Nodes 44
Figure 4: Sample Data Solution via Addition of Cutting-Plane Method 47
Figure 5: Sample Data Solution via Addition of Heuristics Method 50

ACRONYMS

B&B	Branch-and-Bound
CEO	Chief Executive Personnel
HLM	Hierarchical Linear Modeling
HR	Human Resources
LINDO	Linear Interactive and Discrete Optimizer
LP	Linear Programming
MATLAB	Matrix Laboratory
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Programming
OLS	Ordinary Least Square
OR	Operations Research
PFP	Pay-for-Performance

CHAPTER I

BACKGROUND AND GOALS

1.1. Introduction

A typical compensation package consists of a *base salary* and several *benefits* including bonus payments, health plans, retirement plans, and other perquisites. The term *salary administration* is often used, although not exclusively, to refer to the management of the base salary. A fundamental goal of salary administration is to compensate employees in such a way that the salary structure of a firm will exhibit both *external* and *internal consistency*. This means equity within the company and competitiveness within the geographic region where the company operates. More specifically,

- Internal consistency exists when salary differentials reflect the differences in levels of importance of the positions held by the employees of an organization.
- External consistency exists when the salaries of an organization are similar to those of another organization with similar positions in the same geographic region.

Milkovich & Newman [21] outline the importance of developing sound compensation policies for executive personnel. Even though traditional ways of handling personnel salary increases are designed without using mathematical models, a handful of publications have proposed mathematical models to facilitate decision-making in this area. For example, Garcia-Diaz and Hogg [11], Garcia-Diaz et al. [10], Kwak et al. [15], Lal and Srinivasan [16], Fabozzi and Bachner [9], Bruno [5], Charnes et al. [7] developed OR techniques address important aspects of a salary administration. Additionally, Ronen et al. [23], Howard and Miller [14], and Loeb [19] used some other techniques to bring solutions to the same problem.

The main goal of this thesis is to develop a sound and realistic mathematical approach to develop a salary increase guide that recommends salary raises and intervals between successive raises as a function of both performance and potential assessments. Several assumptions are made to enhance the flexibility of the guide. For a given set of

data, MATLAB is used to solve the models. Specific data required for running the MATLAB solver includes employee performance ratings, employee potential assessment, current salaries, and some parameters needed to enhance the flexibility of the salary guide being developed. Salary increases are determined considering external consistency as the measure of effectiveness and internal consistency as a main component of the constraint set of the model.

This study is an extension of the work by Garcia-Diaz and Hogg [11]. In their article, they introduced two separate models. The first model determines salary increase percentages based on performance and potential. Once the salary increase percentages are obtained, they are considered as input in the second model to determine a schedule of salary increase actions along a given period (say one year). They used mixed integer linear programming (MILP) to optimize both models. Thirteen years later, in 1996, Garcia-Diaz et al. [10] published a second paper on the same topic and proposed heuristic procedures to solve the models, to avoid the use of MILP, which requires rather large models for small problems. However, nowadays, microcomputers are capable of solving large-scale problems. Thus, a versatile review and an adaptation of those mathematical models into new computerized solvers can bring out a new compact and improved model to solve the problem more efficiently. To sum up with, a MATLAB (Matrix Laboratory) code to solve the problem in a more effective manner and in large dimensions is generated with improved mathematical models given in the article of Garcia-Diaz et al. [10].

In addition to this introduction, this thesis has eight sections devoted to background material to facilitate discussions and formulations, goals, literature review, conceptual approach, formulations, computerization, applications, and a summary with conclusions and recommendations.

1.2. Background

Operations Research techniques have solved many problems of industries, since 1950s. Every year, many new ways of solving different types of problems through OR techniques are proposed. However, linear programming, which was introduced in 1827 by Joseph Fourier [25], developed in 1939 (during World War II) by Leonid Kantorovich [24], and took its place in the literature by the invention of the Simplex Method in 1947 by George Dantzig, has never lost its popularity among all those invented techniques. Mixed

Integer Linear Programming (MILP), which is the method used in this thesis, was constructed on the basis of Linear Programming (LP).

There are many different interpretations on terminologies used in this study. For this reason, in order for a better understanding of the study, following terms are defined:

- Salary: a fixed amount of (usually monthly) payment made to employees by their employers in exchange for their service provided to employers.
- Compensation: a more general term often used interchangeably in the HR literature covering all benefits (salary, vacation, insurance, etc.) employees receive.
- Merit Increase: a growth in employees' salary/wage based on the past performance and/or efficiency of employees. Also known as pay-for-performance in the literature.
- Potential: likelihood of being promoted to an upper status in an organization.
- Performance: a measurement result of employees' exhibited quality and/or quantity of service.
- Interval: elapsed time between two merit increases.
- Internal incentive: a type of motivation ('monetary' in this context) given to employees so as to encourage them to increase their performance and potential.
- Employee: in this study, the word 'employee' often refers to an executive employee, unless otherwise stated.

The importance of salary administration for executive personnel and PFP method to evaluate the both quantity and quality of work done by employees are mentioned in the previous part. Potential evaluation, which refers to the measure of behavioral reactions exhibited by employees in this study, is another method for appraisal plans. While PFP often measures both the quantity and quality of work done by employees at the end of evaluation period, potential evaluations measures behaviors of employees during the evaluation period.

In most manufacturing organizations, the numerical output of an employee's contributions is a very crucial factor that determines the future of the employee in that company. However, in service industries, behaviors exhibited by employees may even take precedence over the quantitative results of employees. Nowadays, many large organizations run their businesses on both manufacturing and service industries. Even if they do not include both types of operations, they may still need to evaluate some of their

employees' behaviors. For example, the behavior of a supervisor in a manufacturing company affects all sub-workers. Thus, in addition to performance evaluation, the potential evaluation may also improve the quality of employee performance evaluation systems. In this study, both evaluations are considered in the proposed salary administration guideline.

1.3. Problem Statement

Baker et al. [3] emphasized the significant key role of internal incentives in firms on the grounds that they affect behaviors of personnel and pointed out the salary administration as a major growth research area in management and its related sciences.

Salary of executive personnel is one of the most obtrusive cost item in any typical organization's balance-sheet. Arye Bebchuk and Fried [2] indicated the impact of executive personnel on their own salary increase determination process. According to authors, executive managers may tend to secure their positions in their companies and might have an effect on human resource departments, thus influence them on salary increase decisions. Basically, managers may create such an organization culture that can enable them to have authority on all decisions of their companies even their own compensation packages.

While managers may create a danger for their companies by having a possible effect on decisions of their salary increases, they also have a gear role for many businesses. Accordingly and apparently, creating a well-balanced incentive plan for executive employees is an essential duty for the future of many organizations. The main problem falls into place in this phase: in which way a well-balanced incentive plan should be organized so that both managers and company owners are satisfied? As explained in the previous sections, PFP method and potential appraisals play great roles in this problem's solution process.

More specifically, many firms want to correlate their employees' salaries with other employees who work for competitive companies and have similar work statuses. Another aspect is that some companies may want to increase their employees' salaries earlier than the fiscal year because of their high performance. In this case, the question: "when is the best time to increase my employees' salaries?" comes to the ground. All these problems can be solved by using OR techniques as proposed in this study.

1.4. Study Goals

Garcia-Diaz et al. [10] stated that the package optimization software Linear Interactive and Discrete Optimizer, Classic LINDO, were used to solve the problem in their study. This software requires optimization problems to be typed manually on the computer. So, having a large scale problem requires an extensive amount of effort for adapting the problem to the use of Classic LINDO software. Additionally, this software has some restrictions on number of variables and constraints it can solve. According to LINDO's website (www.lindo.com), the classic free version of LINDO software is capable of dealing with maximum 150 constraints, 300 variables, and 30 integer variables in an ordinary linear optimization problem. All these restraints unable users to solve the problem for high number of employees, thus, preclude possible application attempts of the proposed mathematical models in business life.

As mentioned in the introduction part, the main goal of this study is to adapt these mathematical formulations into a MATLAB software code format and use MATLAB software to solve the problem for a large number of executive employees. Another purpose of the study is to develop the formulation of these existing mathematical models. After doing so, companies will be able to use this MATLAB code so as to determine salary increase percentages of their executive personnel and adjust their salaries based upon results provided by the software.

1.5. Approaches

In this section, solution approaches for the problem, mixed integer linear programming, MATLAB's 'intlinprog' algorithm, branch and bound (B&B) method, cutting plane method (also known as cut generations and cutting plane algorithm), and MATLAB's heuristics method are introduced. Mixed integer linear programming is used to formulate mathematical models given in the study. Branch and bound method is applied to solve mathematically modeled problems. Cutting-plane method and MATLAB's heuristics method are also alternative methods that could be used for solving the MILP models. MATLAB's intlinprog algorithm is the main solution package that covers B&B method, cutting-plane method, and heuristics. There are some options for applications of

these algorithms. For more detailed information regarding with options given in the subsections, please refer to MATLAB Optimization Toolbox User's Guide (r2016a) [26].

1.5.1. Mixed Integer Linear Programming (MILP)

Mixed integer linear programming is constructed on the base of Dantzig's Simplex Method. Bixby [4] claims that the first commercially used mixed integer linear programming code dates back to 1960s. Briefly, there are two types of variables in mathematical models, continuous and discrete. Continuous variables can take any real number value, while discrete variables only take integer values. MILP (also known as MIP) basically adds variables that only take a value of either 0 or 1. These variables are also known as binary variables. The following model is illustrated to specify the slight difference between LP and MILP languages.

$$min \quad c^T x$$

$$s.t. \quad Ax \leq b$$

$$and\ x \geq 0$$

In this classical demonstration of LP model, *min* (or *max*) depicts the objective of creating the model. For example, the objective function of a cost problem usually aims to minimize the cost, while a profit problem often tends to maximize the objective function. '*c*' is a row vector that represents coefficients of unknowns in the objective function. '*b*' is a column vector that demonstrates right hand side coefficients of constraints. '*T*' is the transpose symbol that converts the row vector to a column vector. The abbreviation '*s.t.*' stands for 'subject to' in optimization context, but, it is also known as 'such that' in mathematics literature. '*A*' is a matrix that contains coefficients of unknowns in constraints. '*x*' is a column vector that symbolizes unknown variables which can take continuous or discrete values. At this point, MILP comes to the ground; if the problem assumes *x* as a continuous variable, then the problem is a LP problem, otherwise (*x* is discrete or integer), it is an MILP problem. Therefore, adding one more constraint, such as $x \in Z^n$, would make the LP problem an MILP problem.

MILP is a broadly used optimization method for solving many types of business problems, such as travelling salesman problem (a very well-known problem as TSP), transportation problems etc. Many optimization package programs exist, such as

MATLAB's Optimization Toolbox, LINDO, Gurobi, Mathcad, MOSEK, GAMS, OptimJ, AMPL There are also many other packages that uses C, C++, CPLEX, Java, FORTRAN, Visual Basic, .net, MATLAB interfaces to solve optimization problems. Moreover, the Microsoft Excel has an Excel Solver add-on that is capable to deal with optimization problems. Branch and bound and cut generations are most commonly applied techniques to solve MILP problems. This study uses MATLAB's intlinprog algorithm that enables its users to select the desired solving method among these existing techniques. It also has a heuristics procedure that may simplify the solution procedure and decrease solution time in some cases. This algorithm and its corresponding options will be explained with more details in upcoming parts.

1.5.2. MATLAB's 'intlinprog' Algorithm

MATLAB basically has many types of algorithms for a wide variety of optimization problems, such as linear, nonlinear, and quadratic programming problems in its optimization toolbox. MATLAB's intlinprog algorithm is one of the tools involved in this toolbox for solving MILP problems. The word 'intlinprog' is the abbreviation of integer linear programming in the context. This algorithm was added into the optimization toolbox in 2014. Considering that MILP has become a known mathematical language in the literature since 1960s, the addition date of this algorithm might be labeled as a late release. However, there has already existed other codes for other programming languages to solve MILP problems among optimization software. The difference that attracts users to prefer this algorithm is its user-friendliness. This algorithm serves a considerably large variety of options that can be customized by its users to approach the solution ways of problems from different aspects. There are basically three methods that this algorithm uses so as to find optimal solutions for MILP problems. These are B&B method, cutting plane method, and MATLAB's heuristics method.

The algorithm is designed as a minimizer and requires objective function to aim to minimize Thus, a maximization problem needs to be converted to a minimization problem by multiplying its objective function by '-1' in order to be able fit this solver's solution approach. Additionally, all inequality constraints of any problem (minimization or maximization) type which has greater than or equal to ($\geq$) signs should be also converted

into a less than or equal to (≤) by multiplying both sides of constraints by '-1'. To sum up with, the intlinprog algorithm solves problems in the following form:

$$min\ f^T\ x$$
$$s.t.\ \ A \cdot x \leq b$$
$$Aeq \cdot x = beq$$
$$lb \leq x$$
$$x \leq ub$$
$$x(intcon)\ \text{values are integers.}$$

The intlinprog algorithm basically requires some matrices and vectors, such as '*A*' matrix for coefficients of inequality constraints, '*b*' column vector for inequality constraints' corresponding right hand sides, '*Aeq*' matrix for coefficients of equality constraints, '*beq*' column vector for equality constraints' corresponding right hand sides, '*f*' vector for coefficients of variables in the objective function, '*ub*' and '*lb*' column vectors to set upper bounds and lower bounds for values of variables given in the objective function, and '*intcon*' row vector to differentiate binary variables from continuous variables.

After setting these matrices and vectors for a particular MILP problem, the intlinprog algorithm applies the mentioned three MILP solving procedures and detects the optimal solution. The algorithm intuitively selects the method among these options by targeting to solve the problem in the least time. In some cases, the algorithm can apply two or three of these options in a sequence.

Since this study is based on the application of B&B method, cut generations and heuristics options are set to 'none' implying that they are not being used in the solution procedure. Thus, sample data given in the study is basically solved by the application of B&B method. However, all methods are also set free to use in another application attempt for the same sample to investigate the running time difference between these implications.

1.5.3. The Branch-and-Bound (B&B) Method

Branch-and-bound method first appeared in the literature for discrete programming in 1960 by contributions of Land and Doig [17]. Winston and Goldberg [27] asserts that most MILP problems are solved by using B&B method. They also describe this method by

stating: "Branch-and-bound methods find the optimal solution to an IP by efficiently enumerating the points in a sub-problem's feasible region" in their book [27]. The method creates a tree involving subsets of the solution set and identifies branches of the tree. These branches are also named as nodes. Then, the method checks every node's LP relaxation by using an estimated upper and lower bound on the variables. If the recent sub-problem's solution is worse than preceding solutions, the method automatically discards that branch and explores the following branch so as to find the optimal solution. This iterative process continues until the method finds the integer optimal solution, which minimizes (or maximizes) the objective function.

In MATLAB's intlinprog algorithm code, users are free to set options for branching rules. MATLAB's intlinprog algorithm has following options for branching rules [26] (These options are quoted from MATLAB's Optimization Toolbox User's Guide):

- 'maxpscost' – Choose the fractional variable with maximal pseudocost.
- 'mostfractional' – Choose the variable with most fractional part.
- 'maxfun' – Choose the variable with maximal corresponding absolute value in the objective vector.

After the branching process is done, two different nodes exist to select and proceed with one. The following options determine the method for node selection criterion [26] (These options are quoted from MATLAB's Optimization Toolbox User's Guide):

- 'minobj' – Choose the node that has the lowest objective value.
- 'mininfeas' – Choose the node with the minimal sum of integer infeasibilities.
- 'simplebestproj' – Choose the node with the best projection.

1.5.4. The Cutting Plane Method

The cutting plane method/algorithm is an alternative approach for branch-and-bound method to solve MILP problems. The cutting plane method was first introduced to solve MILP problems by Ralph E. Gomory in the late 1950s [13]. By imagining the feasible region as a plane, a cut means to narrow this plane with additional constraints which cuts and reduces the size of the plane so as to draw near and detect the integer optimal solution. So, every cut gets closer to the optimal solution for a typical MILP problem.

There are several cutting methods applied in the literature. MATLAB's intlinprog algorithm code provides several cut generation options to its users and collect these options under three heading: basic, intermediate, and advanced. These options involve following techniques to generate cuts [26] (These options are quoted from MATLAB's Optimization Toolbox User's Guide):

'basic' cuts include:

- ❖ Mixed-integer rounding cuts
- ❖ Gomory cuts
- ❖ Cliques cuts
- ❖ Cover cuts
- ❖ Flow cover cuts

'intermediate' cuts include:

- ❖ Simple lift-and-project cuts
- ❖ Simple pivot-and-reduce cuts
- ❖ Reduce-and-split cuts

'advanced' cuts include:

- ❖ Strong Chvatal-Gomory cuts
- ❖ Zero-half cuts

1.5.5. MATLAB's Heuristics Method

The B&B method investigates feasible points in order to find an upper bound on the objective function; there are such techniques called heuristics that has a possibility of finding these upper bounds faster (these techniques may also fail) [26]. The intlinprog algorithm has the following heuristics methods options [26] (These options are quoted from MATLAB's Optimization Toolbox User's Guide):

- 'rins' – intlinprog searches the neighborhood of the current best integer feasible solution point (if available) to find a new and better solution.
- 'rss' – intlinprog applies a hybrid procedure combining ideas from 'rins' and local branching to search for integer feasible solutions.

- ‘round’ – intlinprog takes the LP solution to the relaxed problem at a node. It rounds the integer components in a way that attempts to maintain feasibility.

1.5.6. Thesis Organization

In the Chapter 2, literature regarding with PFP based salary increases, performance determination processes, and OR approaches on creating salary increase guidelines are mentioned. Chapter 3 illustrates mathematical model development procedure which includes the conceptual approach, variables, and parameters to create these guidelines with an OR based approach. The consistently mentioned MATLAB computerization process is depicted in the Chapter 4. Afterwards, in Chapter 5, an application with a sample data is elaborately demonstrated. The final chapter infers the work done by the study and basically summarizes the whole study.

CHAPTER II

LITERATURE REVIEW

In this chapter, an OR point of view for salary increase systems, how the performance of an employee is determined, and the role of PFP based salary increase systems in the business world are thoroughly explained.

2.1. OR Perspective of Salary Administration

No more than a handful studies mentioned the use of OR techniques in the determination of executive personnel's salary increase percentages and intervals. Garcia-Diaz and Hogg [11], Garcia-Diaz et al. [10], Kwak et al. [15], Lal and Srinivasan [16], Fabozzi and Bachner [9], Bruno [5], Charnes et al. [7] used different OR methods so as to solve compensation problems.

Garcia-Diaz and Hogg [11] developed two separate mixed integer linear programming models to first assign executive personnel to salary increase categories and then determine number of months that they have to wait until they may obtain another salary increase. 13 years later, Garcia-Diaz et al. [10] published another article to suggest using a heuristic methodology to be able to solve the problem with less computation power.

Kwak et al. [15] contributed the topic by using goal programming model for analyzing annual merit salary adjustments in large organizations. Lal and Srinivasan [16] used dynamic and liner programming techniques by applying Holmstrom-Milgrom Method to construct compensation plans for salesforces. Fabozzi and Bachner [9] proposed a different perspective to civil service salary determination by using goal programming and linear programming. Bruno [5] explained how a linear programming model can be used for teachers' salary evaluation. Charnes and et al. [7] also used linear programming for executive compensation.

Additionally, Ronen et al. [23] gave a salary compensation model example while mentioning the broad use of spreadsheet analysis. Howard and Miller [14] used data envelopment analysis for estimation of pay equity in professional baseball. Loeb [19]

offered use of some statistical methods, such as ordinary least square (OLS) and hierarchical linear modeling (HLM) to justify salaries of faculty members.

All these publications draw a remarkable attention on the applicability of OR techniques and mathematical approaches in the personnel salary administration subject of HR field. Even if HR personnel may have an exiguity of OR knowledge, a very well-known and broadly used software, MATLAB, would be able to eliminate these problems and allow them to easily apply the proposed method so as to administrate executive personnel salaries.

Since the salary increase percentage problem has a similar structure with an ordinary assignment problem, application of OR techniques in this study area would be appropriate to efficiently and quickly solve the problem in larger dimensions. Briefly stated, in this salary administration system, employees are assigned to salary increase categories and salary increase time intervals based on some constraints depending on the sector that the business runs. As an example of a constraint, some companies may not desire to make salary increases for all employees in one month, instead, they may want to distribute these salary increases equally among the planning horizon. Another example can be that companies may want to have a fair pay system which ensures that they pay to their workers similar salaries with other companies in the same region. In this manner, constraints such these would aggravate the salary administration problem for human resources departments. Despite the fact that OR techniques could easily deal with all these constraints and assign employees to optimal categories.

Nowadays, the availability of high potential microcomputers enable operations researchers to solve extremely large scale problems in incredibly short times. Thus, taking advantage of technologic developments and creating a single compact solution method for the salary administration problem has been undertaken as a duty in this study.

2.2. PFP Based Salary Increase Systems

Pay-for-performance (PFP hereafter) was proposed as a solution for agency theory problem, which implies that a loss for companies may occur when company owners and managers have unbalanced targets regarding with their companies [6]. More clearly, some managers may pretend that their companies make so much profit by tuning balance sheet

a little bit in order to show themselves as successful leaders. However, this tuning increases the tax amount those companies pay to the government. In this situation, company owners become extremely unsatisfied, since their money goes to the government unnecessarily. Basically, managers sometimes may tend to deceive their companies' stakeholders for their own profit. So as to pretend this conflict, pay-for-performance is suggested by business theorists. Pay for performance prescribes that employees would get salary increase percentages based on their performances.

In the HR literature, many techniques, such as self-evaluation method, weighted checklist method, and structured 360° feedback, to implement PFP based compensations are proposed. While some of the traditional techniques had some issues with regard to measurement quality, new innovative techniques, such as 360° feedback gained a considerable amount of attention from companies. Even though many employees do not favor PFP based measurement systems owing to the belief of unfairness or ruthlessness, these systems almost underlie the basic operations of HR departments. On the other hand, many firms consider these systems necessary to move their businesses to higher ranks in their industries.

PFP system has proven its applicability in many business fields. Indeed, many articles are published about the topic (e.g. Cadsby et al. [6], Abowd [1]). In recent years, many companies use PFP systems to reward their employees. For example, according to Millman[22], the technology leader Apple uses PFP systems successfully in their organization. Also, their criteria on the performance evaluation shows that Apple evaluates potential of employees too. Similar to many other companies, Apple wants to make sure that it pays its CEO a closer amount to its peers. Another company, the automotive giant General Motors which also uses PFP systems. The telecommunication company AT&T also used PFP and potential methods for appraising their employees.

Kuwait-Turkish Participation Bank (at where the author was a former sales assistant) in Turkey uses PFP to allocate their employees to increase categories. According to their system, the bank first evaluates their employees quarterly in every fiscal year. The bank measures all of their employees based on their expectations from their employees. These expectations vary for every other position. For example, they expect a sales assistant to market a certain number of credit cards, a certain amount of mortgage loans, and car loans. Actually, there are more than thirty different types of services that a sales assistant is required to market. So, the bank determines the relevant importance of each service,

weights all these services, and gives credit to each sales assistant for meeting these sales targets. In addition to these sales target evaluations, the bank also takes some other behaviors that their employees perform into consideration. For example, tardiness, loyalty, and passion of employees are some of these other criteria that the bank pays attention to. Even smiling to a customer matters. The bank has mystery shoppers that detects every branch in arbitrary times and these shoppers give credits to employees. At the end of the fiscal year, the bank's human resources and related departments gather all these information and determine bonuses and salary increases that they are going to apply for employees. So, this company evaluates their employees based on their performance and potentials.

The mentioned example types of companies have become very common in the business life. Even small businesses take into consideration these types of applications in their management processes. In recent years, not only private sector companies, but also some public health care companies are also very interested in PFP and potential based incentive systems. This inclination indicates that most companies will be using PFP and potential appraisal systems for their employee rewarding policies, if not using right now.

2.3. Performance Measurement Guide

Performance evaluation results often shape future careers of employees. For this reason, fairness of these evaluations is vitally essential. In the human resources literature, many papers were published about offering ways for establishing elegant personnel performance measurement systems. Gerhart et al. [12] examine performance measures in two sets of categories. First, results-oriented versus behavior-based performance criteria, which compares objective results (e.g., number of produced units or personal sales records) with subjective measures (e.g., traditional merit ratings). Second, individual and group performance measures.

The first measurement puts emphasis on evaluating employees not only with their numerical outputs, but also by their behaviors. Some work positions may not even have any numerical outputs. In this case, including behavioral measures may be effective in the determination of employee's performance. These measurements are also covered in

potential appraisal, which predicts an employee's capability to deal with the job requirements of a higher status.

Some positions may have more interest on number of outputs. In such a situation, objective criteria may be more useful on the performance evaluation. Indeed, these two evaluation aspects are complementary for each other. Therefore, including both objective and subjective measurement criteria can create a well-balanced evaluation form.

On the other hand, the second category implies the vital importance of evaluating employees with both their individual contributions and their group benefit for the work. Deming [8] explains the negative effects of considering individual performance evaluation solely by stating: "Everyone propels himself forward, or tries to, for his own good… The organization is the loser". From this statement, the exiguity of merely individual evaluations is unfolded. Conversely, adding group performance into the measurement can obviously make the evaluation form more multifaceted. Adding potential evaluations as a third dimension into this measurement process, the quality of evaluation can be increased. Thus, employees can be evaluated based on their individual benefits, quality of work, and behaviors.

Among these options and more options from other studies, performance data for employees can be obtained. In this study, a mix of these measurement categories, which is believed to be widely comprehensive, is considered to determine performances of executive employees in the data set provided in the following chapters. Furthermore, this performance evaluation form, which was retrieved from Lead-Deadwood School's website [18], is presented in Appendix A. This sample form does not involve the potential evaluations. Potential evaluations usually measure communication skills, flexibility, leadership, decision making abilities, and performance stability of employees. So, another similar form can also be prepared to measure the potential of employees, which plays a key role in the proposed models.

So as to have a single score for evaluation result (e.g., Performance Score: 5), unweighted average technique is used to calculate the final performance score of employees. Of course, some organizations may prefer using weighted or even additive weighting techniques to degrade all these score into one single score. For instance, some companies may value quantity more, while others might adopt behaviors as an essential

trait. For this reason, choice of using either evaluation techniques is up to a company's objectives and desires.

In the application section of this study, the scale includes five levels except for the "Performance 1", which is considered as an unsatisfactory performance result. This scale can be enlarged or narrowed down depending on how precise decisions companies want to make. However, it should be noted that enlarging the scale would increase number of performance categories, hence, increase number of variables in the model and overcomplicate the problem. For instance, using a scale which is out of 10 or 20 can create an extremely large size model. This model may exceed capabilities of linear programming software to optimize the problem. For this reason, setting a fair enough scale has a key role in the size of model creation.

CHAPTER III

MODEL DEVELOPMENT

3.1. Conceptual Approach

In this chapter, the mathematical models which were mentioned in the previous sections are illustrated with their main sketches, linearized forms, and simplifications. Figure 1 shows two phases of general procedure to apply the solution approach.

Phase one: data collection & preparation

Companies willing to use this system should follow the steps given in Figure 3.1. In order for determining salary increase percentages and time intervals for salary increases, the first step is to collect performance evaluation results mentioned in the previous chapter and discard employees who have unsatisfactory performance results. Thus, these discarded executive employees will not be considered in both models and their increase percentages and salary increase intervals are not computed. The company may fire these employees or apply the minimum increase percentage and the longest increase interval, if there is a strong reason that avert their dismissal, such as a vast amount of gratuity or a key connection that may push the company to higher in the future.

After eliminating unsuccessful executive employees, salary data of evaluees should be gathered. Then, prevailing salary of each employee for his/her corresponding status at nearby companies should be investigated; averages for each salary should be calculated and taken its place in the source data. Based on the budget that was devoted for human resource department, upper and lower bounds for each employee's new salary should be intuitively specified.

Every executive employee would often have the right to promote to a higher status based on position availability sooner or later in many cases as soon as he/she meets required proficiencies. In this solution procedure, only two numerical options are available for potential evaluations, 1 or 0. If an executive employee has enough capabilities to promote, his/her potential score can be noted as 1. In the averse situation, his/her score can be noted as 0. This evaluations should be reported for each evaluee and listed in the data source.

For the use of second mathematical model, which computes time intervals of salary increases, the elapsed time for each employee since the last salary increase should be listed in the source data. The purpose of this action is to be able to compute remaining months to the next salary increase for each employee.

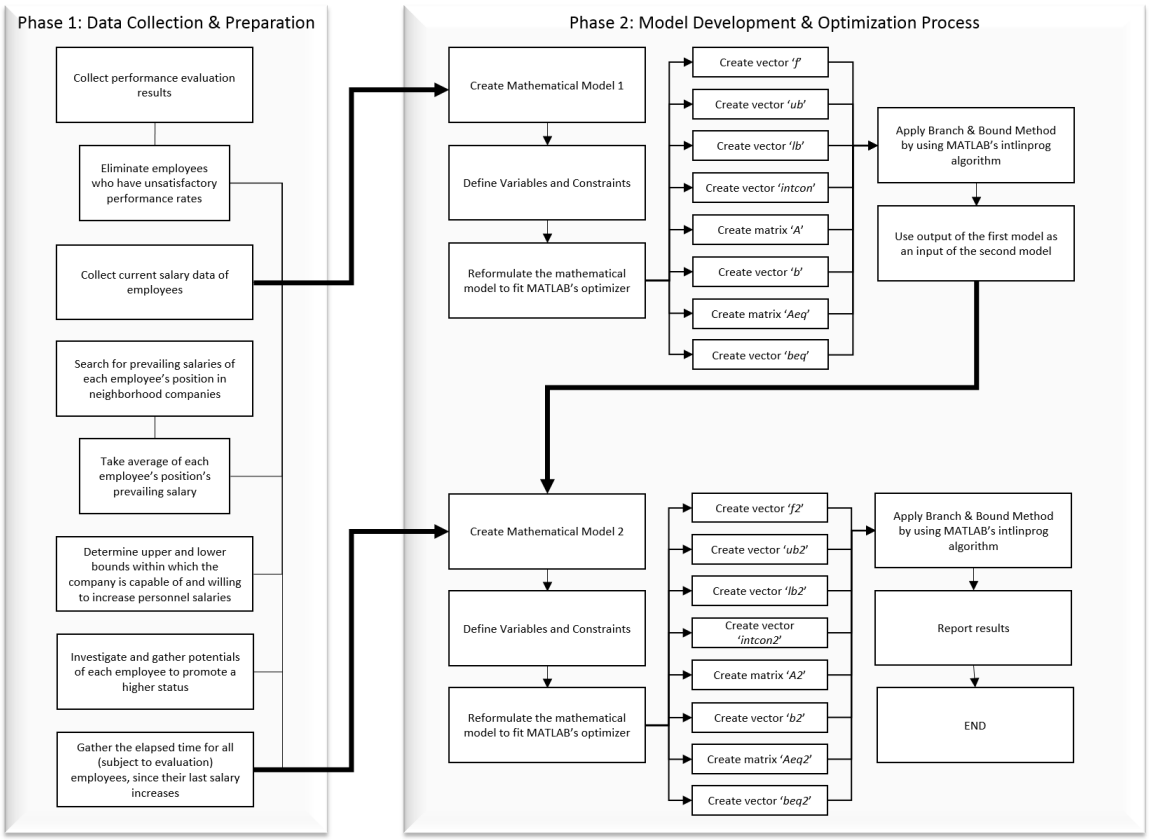


Figure 1: Salary Optimization Guideline

Once the input data is formed for the use of first mathematical model, the model can be shaped and prepared for the use of MATLAB.

Phase two: model development & optimization process

After completing the first phase, the input data sheet which includes all parameters for the use of mathematical models should be ready. This data forms the first model which aims to assign employees to optimal salary increase categories.

The first step of the second phase is to create an MILP model so as to optimize salary increase assignments. Even though this process is illustrated in the salary optimization guideline figure as one single step, it is a whole difficult duty to create a concrete MILP model. This process will be comprehensively explained in the following sections. After the first model is created by defining variables and constraints, the MILP model should be oriented to intlinprog algorithm of MATLAB. This algorithm requires the creation of a bunch of matrices and vectors based on the input data obtained by the first phase.

Once the reformulation step is completed, the intlinprog algorithm should be run, thus, solution for the first model is computed by MATLAB. The output of the first mathematical model is considered to be an input for the second model because the second model aims to sort employees according to their assigned salary increase percentages. By having the result of the first model, a very similar process should be followed with the exception of names of vectors and matrices for the second model. The MATLAB code proposed in this study does not require a second effort to compute the second model. Therefore, by only running the code for the input data which includes desired employee increases and intervals to be optimized, MATLAB optimizer will directly show results for both models. Finally, results obtained from MATLAB can be reported to the respective department.

3.2. Phase One: Data Collection & Preparation

As mentioned in the previous part, required data for the use of the mathematical models should be prepared in this phase. This process can be done via the collaboration of HR department of aforesaid company. Table 1 demonstrates an example of collected data. In order to use MATLAB code provided in this study, the input data should be prepared in the format illustrated in Table 1.

Table 1: Sample Data

N	C	R	U	L	E	P	K
1	5900	7600	8000	6000	3	0	6
2	9000	11000	12000	8500	2	1	3
3	10500	13600	14000	10000	2	1	4
4	7500	8700	10000	7500	5	0	2
5	9100	11800	12000	8500	4	0	4
6	6100	7600	8000	6000	5	1	2
7	6700	6500	7900	5000	4	1	5
8	5800	5900	7800	6200	3	1	1
9	7900	6100	9300	8000	2	0	4
10	9600	12000	14000	10000	5	1	2

Descriptions of headings of the sample data is given below:

N Number of employee

C Current salaries of employees

R Prevailing salaries for employees

U Upper bounds of salaries for employees

L Lower bounds of salaries for employees

E Performance evolution results gathered by HR department

P Potential of employees to promote

K Number of months elapsed since the last given salary increase

3.2.1. Assumptions for Data Preparation

Following assumptions should be considered before formulating mathematical models:

- Upper bound of salary for each employee should be higher than the prevailing salary of corresponding employee.
- Lower bound of salary for each employee should be lower than the prevailing salary of corresponding employee.
- Employees who have unsatisfactory performance results (sample data assumes '1' is an unsatisfactory performance result) should not be included in the input data.

3.3. Phase Two: Model Development & Optimization Process

In this section, notations, variables, and constraints used in the models are provided with detailed information about what type of roles they play in the solution process of the problem. Then, adaptation of these models into MATLAB and their corresponding application procedures are explained. The consistently mentioned two models were explained under separate headings.

3.3.1. Model I

The main purpose of this model is to reassign executive employees to optimal salary increase categories so that the total difference between their new salaries and prevailing salaries corresponding to their statuses is minimized. In this manner, the executive employees are already assigned to their salary categories based on their performance evaluation results by the related department (commonly human resources department). So, by using the word 'reassign', it is meant to distribute employees into salary increase categories based on their performance evaluation results with some additional constraints. For example, a company may want to assign one or several of their executive personnel into a lower salary increase category than the personnel's own salary increase category because the overall sum of salaries given to personnel might be higher than the total amount of prevailing salaries. Thus, the company may want to make sure not to pay to their employees so much more than their rivals. Considering this concept, model I is borrowed from the article of Garcia-Diaz and Hogg [11] to design the optimal way of determining salary increase percentages for every categories.

Notations

Variables

X_m increase percentage corresponding to category m

Z_{im} = 1, if employee *i* is given salary treatment m; Z_{im} = 0, otherwise

W_{it} multiplication of X_m and Z_{im}

λ_i additional variable for linearization purposes

δ_i additional variable for linearization purposes

Subscripts (indices)

i = 1, 2, …, N

m = 1, 2, …, M

Parameters

N number of executive personnel qualified for salary increase

M number of salary increase categories

E_i current performance evaluation result of employee *i*

C_i current salary of employee *i*

R_i prevailing salary for the position of employee *i*

U_i upper bound on the salary of position of employee *i*

L_i lower bound on the salary of position of employee *i*

P_i potential of employee *i* to promote. It is assumed that potential is either 0 or 1 for each employee.

a_1 lower bound on salary increase percentages ($0 \leq a_1 \leq 1$)

a_2 upper bound on salary increase percentages ($a_1 \leq a_2 \leq 1$)

a_3 minimal difference between increase percentages of neighboring categories ($0 \leq a_3 \leq a_2$)

d maximal deviation, in increase categories, allowed between job performances and salary treatments

q maximal position in salary range allowed to low-potential personnel ($0 \leq q \leq 1$)

Mathematical Formulation

Minimize

$$\sum_i \left| C_i \left(\sum_m Z_{im} X_m + 1 \right) - R_i \right| \quad (1)$$

Subject to

$$C_i \left(\sum_m Z_{im} X_m + 1 \right) \leq P_i U_i + (1 - P_i)[L_i + q(U_i - L_i], \text{for all i} \quad (2)$$

$$C_i \left(\sum_m Z_{im} X_m + 1 \right) \geq L_i, \text{for all i} \quad (3)$$

$$X_1 \geq a_1, X_M \leq a_2, X_M - X_{M-1} \geq a_3, \text{for all m} \geq 2 \quad (4)$$

$$\sum_m m Z_{im} \leq e_i, \text{for all i} \quad (5)$$

$$\sum_m m Z_{im} \leq e_i - d, \text{for all i} \quad (6)$$

$$\sum_m Z_{im} = 1, \text{for all i} \quad (7)$$

$$Z_{im} = 0, 1, \text{for all i and all m} \quad (8)$$

As mentioned in the introduction of this section, the objective function (1) aims to minimize the total deviation (including both positive and negative difference) between all new salaries of executive personnel and all prevailing salaries corresponding to these personnel. Thus, the objective function (1) ensures that the external consistency condition mentioned in the previous sections is met. Since the objective function (1) includes a multiplication of two variables, the elucidated model is a non-linear mathematical model. The linearization procedure of this type of non-linear objective function is explained in the following parts of the section.

Constraints (2) and (3) set upper bounds for all employees based on their potentials. Thus, these two constraints target to meet internal consistency condition, the importance of which was mentioned in previous sections. Companies using this mathematical model

can basically set upper and lower bounds based on their budget. Potential in constraint (2) affects the upper bound of salary for each employee. For instance, if an employee has a potential value of 1, then the employe gets a full upper bound. In averse situation, if an employee has a low (0) potential, then the employee's upper bound of new salary is multiplied with a value between 0 and 1 in order to decrease the upper bound and allow the employee to get a salary increase within a lesser boundary. Hence, awarding employees is also assured by affecting their new salaries based on their high potentials, which can be a fair application in competitor working environments.

Set of constraints (4) create an interval for salary increase categories. The first constraint in the set assures that the lowest salary increase category should be at least a certain percetange determined by the model user company. The second constraint indicates the upper bound of the best salary increase category. Therefore, the best salary increase category cannot exceed the highest salary increase percentage that the model user company is willing to set. The last set of constraints define the minimal percentage between neighboring salary increase categories. Hence, the difference between salary increase categories would motivate employees to be more beneficial to the company.

Constraint (5) ensures that employees subject to consideration for assignment should not be assigned to a performance category higher than their predetermined perfomance results. Constraint (6) determines the range of new salary assignments of employees. So, this constraint limits the guideline to assign an employee to an extremely lower salary increase category. In other words, assuming d=1, a company confirms that all employees considered in the model will be assigned to either their predetermined or one category lower than their predetermined salary increase category. Increasing d can enlarge the range, thus, might create more possibilities for the solution set.

Constraints (7) and (8) ensures that every employee included in the guideline is assigned to only one salary increase category.

As a return of the model, X and Z values are obtained. X values are salary increase categories which determine the salary increase percentage of each employee. The parameter a_1 and a_2 define the bounds between the lowest and highest X percentages. Thus, the model determines X percentages based on the internal consistency of user companies. On the other hand, Z variables demonstrate matchings of employees with their corresponding optimal salary increase categories. In other words, the indices of Z variables illustrate the optimal salary increase category for each employee. For example, Z_{13} means

that the first employee in the evaluation is assigned to salary increase category 3. By assuming X_3 to have an optimal value of 11%, it can be inferred that the first employee in the consideration would get 11% salary increase according to findings of Model I. Finally, in this section, the first mathmetical model is illustrated in a non-linear format. Besides, all constraints and their roles in the salary increase category assignment model are explained.

Linearization Procedure

In mathematics and operations research literature, several methods for linearizing non-linear objective functions and constraints exist. One of these techniques is the linearization of minimization of the sum of absolute values, as McCarl et al. [20] explained in their book. According to their method, the new form of the objective function can be illustrated as following:

Minimize

$$\sum_i \lambda_i + \delta_i \text{, for all i} \tag{9}$$

Thus, the linearized objective function becomes a constraint as following:

Subject to

$$C_i \left(\sum_m Z_{im} X_m + 1 \right) - \lambda_i + \delta_i = R_i \text{, for all i} \tag{10}$$

where

$$\lambda_i \geq 0 \text{ and } \delta_i \geq 0 \text{, for all i} \tag{11}$$

Another expression that causes the problem to be non-linear is the multiplication of two variables, *Z* and *X*. So, this non-linear expression can be linearized by switching the multiplication with a new variable which covers the multiplication of both variables. As a result of this process, the following transformation is shown below:

$$W_{im} = Z_{im} X_m \text{, for all i and m} \tag{12}$$

After taking the above transformation into consideration, following constraints are added to hold the transformation:

$$W_{im} \leq Z_{im} \text{, for all i and m} \tag{13}$$

$W_{im} \geq Z_{im} + X_m - 1$, for all i and m (14)

$W_{im} \leq X_m$, for all i and m (15)

After applying all these linearization process, the problem became an MILP problem. Therefore, the final linearized form of the MILP problem is illustrated below:

Minimize

$$\sum_i \lambda_i + \delta_i \text{ , for all i} \quad (16)$$

Subject to

$$C_i\left(\sum_m Z_{im}X_m + 1\right) - \lambda_i + \delta_i = R_i, \text{for all i} \quad (17)$$

$$C_i\left(\sum_m Z_{im}X_m + 1\right) \leq P_iU_i + (1 - P_i)[L_i + q(U_i - L_i], \text{for all i} \quad (18)$$

$$C_i\left(\sum_m Z_{im}X_m + 1\right) \geq L_i, \text{for all i} \quad (19)$$

$X_1 \geq a_1, X_M \leq a_2, X_M - X_{M-1} \geq a_3$, for all m ≥ 2 (20)

$$\sum_m mZ_{im} \leq e_i, \text{for all i} \quad (21)$$

$$\sum_m mZ_{im} \leq e_i - d, \text{for all i} \quad (22)$$

$$\sum_m Z_{im} = 1, \text{for all i} \quad (23)$$

$W_{im} \leq Z_{im}$, for all i and m (24)

$W_{im} \geq Z_{im} + X_m - 1$, for all i and m (25)

$W_{im} \leq X_m$, for all i and m (26)

$Z_{im} = 0, 1$, for all i and all m (27)

$X_m \geq 0, W_{im} \geq 0,\ Z_{im} \geq 0, \lambda_i \geq 0$, and $\delta_i \geq 0$, for all i and m (28)

Reformulation procedure

As mentioned earlier in the MATLAB's 'intlinprog' algorithm section of Chapter I, the intlinprog algorithm only solves minimization problems. Since the objective function of the Model I is a minimization function, no further operation is needed. On the other

hand, both sides of some constraints, such as (19), (20), and (25) should be multiplied by '-1' in order to change greater than or equal to signs into less than or equal to signs. Moreover, simplifying inequalities as much as possible would also help for creating matrices and vectors. After doing these simple operations, the whole model would become as following (All variables are assumed to be positive):

Minimize

$$\sum_i \lambda_i + \delta_i \text{, for all i} \tag{29}$$

Subject to

$$C_i \sum_m W_{im} - \lambda_i + \delta_i = R_i - C_i \text{, for all i} \tag{30}$$

$$\sum_m Z_{im} = 1 \text{, for all i} \tag{31}$$

$$C_i \sum_m W_{im} \leq P_i U_i + (1 - P_i)[L_i + q(U_i - L_i] - C_i \text{, for all i} \tag{32}$$

$$C_i \sum_m -W_{im} \leq C_i - L_i \text{, for all i} \tag{33}$$

$$-X_1 \leq -a_1, X_M \leq a_2, X_{M-1} - X_M \leq -a_3 \text{, for all m} \geq 2 \tag{34}$$

$$\sum_m mZ_{im} \leq e_i \text{, for all i} \tag{35}$$

$$\sum_m -mZ_{im} \leq d - e_i \text{, for all i} \tag{36}$$

$$W_{im} - Z_{im} \leq 0 \text{, for all i and m} \tag{37}$$

$$X_m + Z_{im} - W_{im} \leq 1 \text{, for all i and m} \tag{38}$$

$$W_{im} - X_m \leq 0 \text{, for all i and m} \tag{39}$$

$$Z_{im} = 0, 1 \text{, for all i and all m} \tag{40}$$

Computerization procedure

This procedure contains creation of vectors and matrices illustrated in the salary optimization guideline figure. Even though an obligatory sequence does not exist in the creation of these vectors and matrices, starting with the creation of column vector *f* may ease the whole process. So, a column vector including coefficients of all variables in the objective function should be created first. The following illustration depicts in what

sequence variables can be listed in the vector and how an f vector is created for the mathematical model I.

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_i \\ \delta_1 \\ \delta_2 \\ \vdots \\ \delta_i \\ W_{11} \\ W_{12} \\ \vdots \\ W_{im} \\ Z_{11} \\ Z_{12} \\ \vdots \\ Z_{im} \\ X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix} \xrightarrow{yields} f = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

The sequence of variables in the vector is just an example; this sequence may also be changed. However, changing the sequence would also affect following steps of matrix and vector creation procedure. Thus, keeping the order of variables consistently in all other vector creations is necessary for many cases. Since the objective function only includes lambda and delta variables, zeros are placed for all other variables' coefficients in the f vector. After having an order for all variables given in the model, the next step can be determination of lower and upper bounds of variables and creating their corresponding column vectors. The following vectors elucidate how upper and lower bounds can be vectorized.

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_i \\ \delta_1 \\ \delta_2 \\ \vdots \\ \delta_i \\ W_{11} \\ W_{12} \\ \vdots \\ W_{im} \\ Z_{11} \\ Z_{12} \\ \vdots \\ Z_{im} \\ X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix} \xrightarrow{yields} ub = \begin{bmatrix} inf \\ inf \\ \vdots \\ inf \\ inf \\ inf \\ \vdots \\ inf \\ inf \\ inf \\ \vdots \\ inf \\ 1 \\ 1 \\ \vdots \\ 1 \\ inf \\ inf \\ \vdots \\ inf \end{bmatrix} \text{ and } lb = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

The above illustration means that all variables except for Z variables are continuous. Since Z variables are binary, their upper bounds are 1 meaning that maximum

value they could get is 1. On the other hand, all variables have a lower bound of 0, which also means that all variables are positive.

The next step is to create an *intcon* row vector which contains the sequence number of integer variables. The importance of starting with the creation of vector *f* arises here because *intcon* should be created accordingly to vector *f*. The following vectors theorically demonstrates how an *intcon* vector for this problem should be.

$$[\lambda_1 \quad \lambda_2 \quad \cdots \quad \lambda_i \quad \delta_1 \quad \delta_2 \quad \cdots \quad \delta_i \quad W_{11} \quad W_{12} \quad \cdots \quad W_{im} \quad Z_{11} \quad Z_{12} \quad \cdots \quad Z_{im} \quad X_1 \quad X_2 \quad \cdots \quad X_m]$$

$$\downarrow \ yields$$

$$[2i + im + 1 \quad 2i + im + 2 \quad \cdots \quad 2i + 2im]$$

Consider $i = m = 3$ as an example, then *intcon* vector becomes:

$$[16 \quad 17 \quad 18 \quad 19 \quad 20 \quad 21 \quad 22 \quad 23 \quad 24]$$

The *intcon* vector is a very important component of the problem solution procedure owing to the fact that it determines which variables are considered as integers and conveys this information to the program.

After creating all these vectors, coefficients of constraints should be matricized and their corresponding right hand sides should be vectorized so as to fit MATLAB's 'intlinprog' algorithm. So as to be able to do this, inequality constraints and equality constraints can be separately categorized because matrix *A* should be created for coefficients of inequality constraints, while matrix *Aeq* needs to be created for coefficients of equality constraints. Briefly, dividing constraints into two categories as equality and inequality constraints, the application can be facilitated. Since matrix *A* is a number of constraints by number variables sized matrix, it can have extremely large dimensions with high number of employees and salary increase categories. As a simplistic visual example, the following matrix *A* can be set for the Model I by assuming *N*=*M*=1.

$$A = \overbrace{\begin{matrix} \lambda_1 & \delta_1 & W_{11} & Z_{11} & X_1 \end{matrix}}^{} \begin{bmatrix} 0 & 0 & C_1 & 0 & 0 \\ 0 & 0 & -C_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 1 & 0 & -1 \end{bmatrix} \begin{matrix} (32) \\ (33) \\ (34) \\ (35) \\ (36) \\ (37) \\ (38) \\ (39) \end{matrix}$$

In the above matrix, while parenthesized numbers on the right hand side of the matrix represent constraints, the variables are also indicated on the upper side of the matrix. It should be noted that constraint set (34) contains several constraints in it depending on number of *m*. All other constraints depend on number of *i*, which is the total number of employees considered in the first model. The size of matrix *A* can grow exponentially depending on numbers of *i* and *m*. For instance, while 2 employees and 2 salary increase categories create a 23x14 size matrix *A*, 3 employees and 3 salary increase categories yield a 43x27 size matrix *A*. Basically, the size of matrix *A* can be computed as $4N+M+3NM+1$ by $2N+2NM+M$.

The corresponding *b* vector for matrix *A*, which includes right hand sides of inequality constraints is also needed to be created. Since right hand sides only involve parameters which can be obtained from input data, the creation of this vector is very simple. An example *b* vector by 1 employee and 1 salary increase category is demonstrated below:

$$b = \begin{bmatrix} P_1U_1 + (1-P_1)[L_1 + q(U_1 - L_1] - C_1 \\ C_1 - L_1 \\ -a_1 \\ e_1 \\ d - e_1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} (32) \\ (33) \\ (34) \\ (35) \\ (36) \\ (37) \\ (38) \\ (39) \end{matrix}$$

As it is seen, the dimension of vector *b* is the same as the row number of matrix *A*, since *b* includes only right hand sides of inequality constraints.

A similar process also holds for creating matrix *Aeq* and vector *beq*. Fortunately, this matrix and vector contain lower dimensions than matrix *A* and vector *b*, since the model only involves two equality constraints. Therefore, matrix Aeq and vector beq can be shown as the following for a very simplistic case of $N=M=1$.

$$\overset{\lambda_1 \;\; \delta_1 \;\; W_{11} \;\; Z_{11} \;\; X_1}{Aeq = \overbrace{\begin{bmatrix} -1 & 1 & C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}}} \begin{matrix} (30) \\ (31) \end{matrix}$$

and

$$beq = \begin{bmatrix} R_1 - C_1 \\ 1 \end{bmatrix} \begin{matrix} (30) \\ (31) \end{matrix}$$

Size of the matrix *Aeq* can be calculated as 2*N* by 2*N*+2*NM*+*M*. So, this matrix is a smaller dimension matrix compared to matrix *A*. However, it should be noted that both of the matrices can get a very large scale dimension by considering high number of employees and salary increase categories. Depending on these two factors, sizes change dynamically.

After doing all these operations, the only required task is to call MATLAB's optimizer with a simple function and find the solution. By applying these operations, the company would be able to assign their employees to more optimal salary increase categories and compute new salaries of their employees.

Since creating all these matrices and vectors for every single problem takes so much time and effort, a concrete MATLAB code is a need to automatically meet all requirements of intlinprog algorithm. The challenge in the MATLAB coding is to dynamically create these matrices depending on number of employees, number of increase categories, and varieties of parameters, and then automatically sending it into optimizer with one single touch in a single MATLAB script. Moreover, combining the second model with the first one and reducing all these steps is the centerpiece of this thesis study.

The above mentioned one piece MATLAB script for solving both models is shared in Appendix B. If desired, the second model given in the script can be removed and solutions only for the first model can also be computed. So, removing the second model does not affect the solution of the first model. However, without solving the first model, second model cannot be computed. This is because the second model gets the input data from the solution of the first model. This situation is explained in the second model with more details.

3.3.2. Model II

In recent years, some companies may not desire to apply salary increases for all employees in the same month owing to the fact that accumulation of a high expense in one month can negatively influence the cash flow of companies. Besides, the increasing competiveness in working environments may require companies to reward their successful employees earlier than their low performance employees. Thus, creating another assignment tool might be necessary for these type of applications.

This model aims to distribute employees into salary increase intervals as evenly as possible. In other words, the model targets to assign employees into different salary increase time intervals depending on their salary increase category results obtained by the first model. The Model II is developed by using the model proposed in the article of Garcia-Diaz and Hogg [11].

Notations

Variables

Y_{it} = 1, if employee i gets a salary increase in period t; Y_{it} = 0, otherwise, where i = 1, 2,…, N and t = 1, 2,…, T

σ_t additional variable for linearization purposes

ω_i additional variable for linearization purposes

Subscripts (indices)

i = 1, 2, …, N

t = 1, 2, …, T

Parameters

A lower bound on intervals

B upper bound on intervals ($B>A$)

K_i number of periods passed since the last given salary treatment to employee i ($B>K_i>A$)

T number of periods in the planning horizon

g_i salary treatment given to employee i by Model I, $g_i = \sum mZ_{im}$

n_t number of personnel desired to assign in period t

α_i = min{T; $A - K_i$}

β_i = min{T; $B - K_i$}

Mathematical Formulation

Minimize

$$\sum_{t=1}^{T}\left|\sum_{i=1}^{N} Y_{it} - n_t\right| \tag{41}$$

Subject to

$$\sum_{t=1}^{T} Y_{it} = 1\text{, for all } i \tag{42}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} + K_i = \sum_{\alpha_j=A-f_j}^{\beta_j=B-f_j} tY_{jt} + K_j\text{, for } i \text{ and } j \text{ such that } g_i = g_j \tag{43}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} + K_i \leq \sum_{\alpha_j=A-f_j}^{\beta_j=B-f_j} tY_{jt} + K_j\text{, for } i \text{ and } j \text{ such that } g_i > g_j \tag{44}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} + K_i \geq \sum_{\alpha_j=A-f_j}^{\beta_j=B-f_j} tY_{jt} + K_j\text{, for } i \text{ and } j \text{ such that } g_i < g_j \tag{45}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} + K_i \leq B\text{, for corresponding to minimal } g_i \tag{46}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} + K_i \geq A\text{, for corresponding to maximal } g_i \tag{47}$$

$$Y_{it} = 0, 1\text{, for all } i \text{ and } t \tag{48}$$

The main goal of the objective function (41) is to assign employees into salary increase intervals with a minimal deviation. Thus, employees are assigned to their salary increase time intervals in such a way that does not jeopardize company by causing a cash flow problem regarding with salary increases.

Constraint (42) ensures each employee to be assigned to only one salary increase interval. Constraints (43), (44), and (45) assures that employees with same salary increase category assignments would get their salary increases in the same interval. If one of the employees has a better salary increase category than the other one, then this better

employee gets a salary increase earlier than or at least at the same time with the other employee. It should be noted that $j = i + 1$, where $i = 1, 2, \ldots, N\text{-}1$. This notation satisfies that all employees subject to consideration are compared one by one. Besides, these three (43), (44), and (45) constraints are conditional constraints depending on the result of the Model I. Thus, for every one comparison, only one constraint will be active. For example, if two employees have the same salary increase category result (obtained by the Model I), constraint (43) will be activated/used, while other two constraints are not used.

Constraints (46) and (47) defines the upper and lower bounds of salary increase intervals. Thus, exceeding these limits is assured to be restricted by these two constraints. Finally, constraint (48) satisfies both non-negativity and integrity conditions.

Linearization Procedure

As it can be distinctly seen in the model, linearization of the Model II requires very similar steps applied in Model I. So, applying McCarl et al.'s [20] linearization procedure one more time would linearize the Model II as shown below:

Minimize

$$\sum_t \sigma_t + \omega_t \tag{49}$$

Subject to

$$\sum_t Y_{it} - \sigma_t + \omega_t = n_t\,, t = 1, 2, \ldots, T \tag{50}$$

$$\sigma_t \geq 0, \omega_t \geq 0, t = 1, 2, \ldots, T \tag{51}$$

After the linearization procedure, the complete Model II takes a shape as demonstrated below:

Minimize

$$\sum_t \sigma_t + \omega_t \tag{52}$$

Subject to

$$\sum_{t} Y_{it} - \sigma_t + \omega_t = n_t \text{ , for all } i \text{ and } t = 1, 2, \dots, T \tag{53}$$

$$\sum_{t=1}^{T} Y_{it} = 1 \text{ , for all } i \tag{54}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} + K_i = \sum_{\alpha_j=A-f_j}^{\beta_j=B-f_j} tY_{jt} + K_j, \text{ for } i \text{ and } j \text{ such that } g_i = g_j \tag{55}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} + K_i \leq \sum_{\alpha_j=A-f_j}^{\beta_j=B-f_j} tY_{jt} + K_j, \text{ for } i \text{ and } j \text{ such that } g_i > g_j \tag{56}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} + K_i \geq \sum_{\alpha_j=A-f_j}^{\beta_j=B-f_j} tY_{jt} + K_j, \text{ for } i \text{ and } j \text{ such that } g_i < g_j \tag{57}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} + K_i \leq B, \text{ for corresponding to minimal } g_i \tag{58}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} + K_i \geq A, \text{ for corresponding to maximal } g_i \tag{59}$$

$$Y_{it} = 0, 1, \text{ for all } i \text{ and } t \tag{60}$$

$$Y_{it} \geq 0, \sigma_t \geq 0, \omega_t \geq 0, t = 1, 2, \dots, T \text{ and } i = 1, 2, \dots, N \tag{61}$$

Reformulation procedure

In this procedure, a very similar approach used for Model I is applied to the Model II in order to prepare the problem for adapting MATLAB's solver. Hence, the Model II is rewritten below by applying all required operations.

Minimize

$$\sum_{t} \sigma_t + \omega_t \tag{62}$$

Subject to

$$\sum_{t} Y_{it} - \sigma_t + \omega_t = n_t \text{ , } t = 1, 2, \dots, T \tag{63}$$

$$\sum_{t=1}^{T} Y_{it} = 1 \text{ , for all } i \tag{64}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} - \sum_{\alpha_j=A-f_j}^{\beta_j=B-f_j} tY_{jt} = K_j - K_i, \text{for } i \text{ and } j \text{ such that } g_i = g_j \tag{65}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} - \sum_{\alpha_j=A-f_j}^{\beta_j=B-f_j} tY_{jt} \leq +K_j - K_i, \text{for } i \text{ and } j \text{ such that } g_i > g_j \tag{66}$$

$$\sum_{\alpha_j=A-f_j}^{\beta_j=B-f_j} tY_{jt} - \sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} \leq K_i - K_j, \text{for } i \text{ and } j \text{ such that } g_i < g_j \tag{67}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} tY_{it} \leq B - K_i, \text{for corresponding to minimal } g_i \tag{68}$$

$$\sum_{\alpha_i=A-f_i}^{\beta_i=B-f_i} -tY_{it} \leq K_i - A, \text{for corresponding to maximal } g_i \tag{69}$$

$$Y_{it} = 0, 1, \text{for all } i \text{ and } t \tag{70}$$

Computerization procedure

Here, a very similar computerization procedure with Model I applies for this model. The only difference and challenge for Model II is to construct matrices and vectors dynamically changing with their correspondingly chosen constraints. So, the MATLAB code to solve this model is required to manage conditional constraints with their 'if' conditions. Since g_i values bring an 'if' condition for constraints (65), (66), and (67), these selections should be managed in a manner that either creates a row in matrix A or *Aeq*. For example, if two employees have the same result from the Model I, then constraint (65) should be activated. Activating constraint (65) means to add a row containing coefficients of variables into matrix Aeq, since constraint (65) is an equality constraint. In a reverse situation, where one employee's salary category is better or worse than another employee, their corresponding row including coefficients of variables should be added into matrix A.

As Model I and Model II are combined in one script on MATLAB, titles of matrices should be different to get results for both of the models. For this reason, matrices are named with an additional '2' on their subsequences, such as A2, Aeq2, b2 etc.

Owing to the fact that matrix creation procedure is explained in the previous section, repeating the same procedure is not considered necessary. In Appendix B, the whole coding can be found. Also, some additional explanations regarding what means what in the coding is provided between code lines. As computer coding has a wide variety ways of handling operations, the given code is just a working example of solution for these two models. Of course, a more effective code on MATLAB or another package program which requires less time to solve the problem may always be produced.

By using the produced MATLAB code, no further effort is required to compute solutions except for preparing the input data illustrated as in Table 1. After preparing input data and running the code, results should be evaluate for their accuracy. If there is any conflict regarding the computation results, the input data should be reviewed based on the assumptions provided in data collection and preparation section and any conflict causing data should be removed or changed with an appropriate one that allows the code work.

CHAPTER IV

SAMPLE APPLICATION

In this chapter, a sample application is performed within using the aforementioned MATLAB code, which carries out two mathematical models respectively. The sample data given in section 3.2 is used in the application. In addition to application procedure, a comprehensive qualitative analysis of solution procedure and findings, comparison among other solution options and their corresponding solution speed are provided.

The first step that needs to be followed is to transfer the sample data into a MATLAB .mat file. To do this, the sample data excluding the headings and counters of employees is copied and pasted into a random sized matrix, and then, saved with a name of 'input'.

After creating the input data for MATLAB, the code can be simply run with default settings of intlinprog algorithm. Selecting among three different solution methods can also be an optional step that might be done before running the provided code (provided code is in Appendix B). In this chapter, assumptions for parameters and findings with their analysis are provided for each method (B&B, cutting-plane, and heuristics).

4.1. Assumptions for Parameters

Some parameters i.e. *q*, *d*, *A*, and *B* should be determined in addition to the information provided with sample data. These parameters tremendously affect the size of the problem. For instance, *A* and *B* set the upper and lower bound on salary increase intervals. The complexity and the size of the problem highly depends on the difference between these upper and lower bounds. Also, the parameter *d* creates another complexity by allowing one employee to be a candidate for many salary increase categories. If *d* is assumed to be 1, the new salary increase category assigned by the salary administration package should be either same as the predetermined salary increase category or one lower of the predetermined salary increase category. Thus, assuming *d* as a higher number would

enlarge the bound on this selection process and increases the duty of the program which is in charge of investigating all nodes.

From another perspective, user companies may feel free to use any numbers for these parameters due to the fact that these parameters are very attractive tools for them to manage their employees' salaries and their salary increase frequencies. For example, the parameter *q* determines the upper bound of low potential employees' salaries which means that low potential employees can maximum have a salary with their upper bounds multiplied with *q*. While assuming a very high percentage *q* ($0<q<1$) would punish low potential employees deficiently, a very low percentage *q* would over-punish them. So, setting a fairly balanced *q* percentage is an essential part of an ethic evaluation.

Table 2 shows the values used for parameters of both Model I and Model II in the sample problem.

Table 2: Parameter Value Assumptions

PARAMETERS	VALUES
N	10
M	5
q	0.90
d	1
a_1	0.05
a_2	0.20
a_3	0.03
A	8
B	12
T	12
n	14

Due to the fact that 10 employees with 5 different salary increase categories are evaluated in these models, *N* and *M* are indicated as 10 and 5, respectively. The parameter *q* is assumed to be 0.90 in the sample application, which is thought to be a fair enough

penalizing for low potential employees. The parameter d is determined as 1, which is perceived to be both acceptable for employees and a fair enough deal for companies. The a_1 is the lowest percentage that will be given to worst performing employees. In this sample, it is assumed to be 5%, which is believed to be the minimum amount of increase percentage to avoid leave of employment for low performance employees. It should be noted that executive personnel often are not ordinary type of employees that would connect to their companies with candid relationships. Aversely, executive employees are usually talented and well-educated people that would easily find another job sooner or later. For this reason, even the worst executive employee in any company should be valued by giving a reasonable enough salary increase percentage which is assumed to be 5% in this example. At the same time, while the highest salary increase percentage, a_2, is set to be 20%, the difference between salary increase categories, a_3, are determined as 3% in this sample application. These percentages may surely differ from an industry to another. Fortunately, the code is designed to be capable of handling the problem solution with different desired parameters determined by users.

As the lower bound of desired salary increase interval is designated as 8 months, while its upper bound is identified as 12 months. Additionally, the whole time interval for the salary increase period is also stated as 12 months. The desired number of assignments for each month in the time interval is determined to be 14.

4.2. Sample Solution via Branch & Bound Method

The intlinprog algorithm's default options are already set to use B&B method for solving MILP problems. The method has three different ways to handle the procedure of application. The best option for solving a problem as quickly as possible varies depending on the type of the problem. In the salary administratin problem, the fastest option is 'most fractional' for branching. Hence, the 'most fractional' option to carry out the B&B method is activated in the given code.

As mentioned earlier, B&B method involves investigation of nodes of the B&B tree. So as to practise these investigations for each node, the sub-problems are solved by using LP. Linear programming problems can be solved with either the simplex method or the dual-simplex method via intlinprog algorithm. The dual-simplex method option (RootLPAlgorithm) in the intlinprog algorithm is activated in the provided code because

of its effectiveness on the sample data. This effectiveness may vary for different data set. So, it should also be kept in mind that chnaging this option might also affect the solution speed for other types of problems.

Since the main purpose of this section is to test the efficiency of B&B method, the other two methods (cutting-plane and heuristics) are set to 'none', which disables the algorithm to use those methods, before running the code. Figure 2 illustrates the result of B&B method application via the screenshot of MATLAB software.

The optimal percentages of salary increases for their corresponding categories, the optimal assignments of employees into salary increase categories, and the optimal time intervals for salary increase applications of each employee are computed by using the B&B method in the provided code.

According to the results of both models, while the lowest salary increase percentage corresponds to 8%, the highest percentage is found 20%. This means that the total deviation between current salaries and prevailing salaries of 10 employees is a positive number. In other words, the sum of current salaries is lower than the sum of prevailing salaries. For this reason, the algorithm tends to give the highest possible percentage determined by the range of percentages, a_1 and a_2. After determining X_5, which is the best salary increase category, the algorithm gives the most appropriate percentage for the following salary increase category X_4 based on the percentage determined by a_3. In an averse case, the algorithm would determine the worst salary increase category, X_1, as the lowest possible percentage and would incrementally specify upper increase categories by using the a_3 percentage.

During the adjustments of percentages for salary increase categories, the algorithm also takes the internal consistency into consideration and assign employees to most appropriate salary increase categories based on their predetermined performance results. As shown in Figure 2, the first executive employee in the sample data is assigned to the salary increase category 3; the second personnel is assigned to category 2, and so forth. These can be understood by checking the indices of variables. According to the results of Model II shown in Figure 2, the first employee gets a salary increase 6 months later and the second employee gets a salary increase 9 months later. It can be distinctly seen in the results that any employees are not exposed to any unfair assignment. While the higher performance employee gets the salary increase in a shorter time, the lower performance employee gets the salary increase at most the same increase with the higher performance

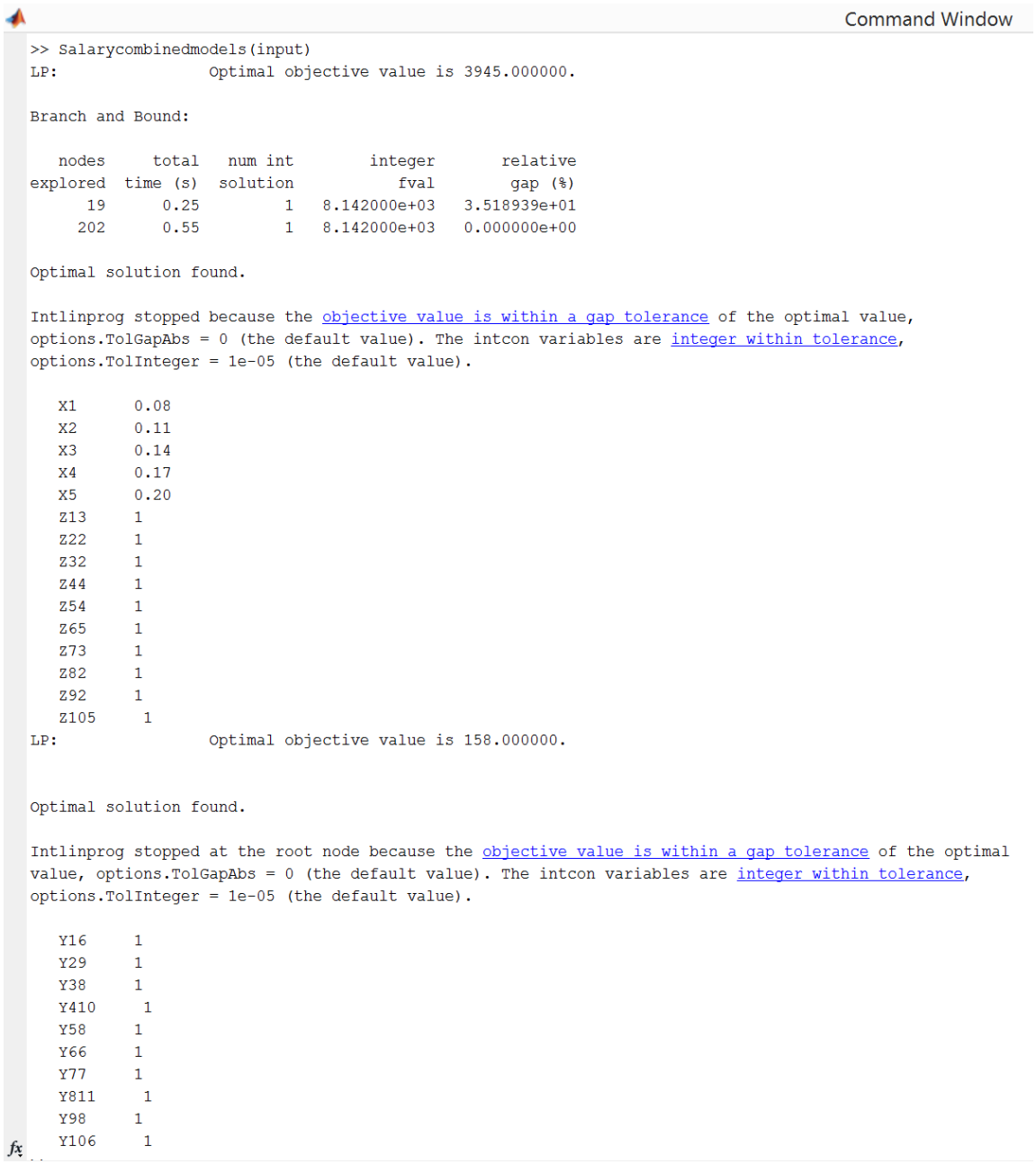


Figure 2: Sample Data Solution via B&B Method

employee depending on the internal and external consistency policies and the desired number of salary increases allowed per specified months.

Figure 3 illustrates the B&B tree for the solution of two models. According to 'Run and Time' function of MATLAB, the total time spent for the evalution of 10 employees is 3.754 seconds. Besides, the optimal solution is found in 202th node of the B&B tree for the Model I, as it can be seen in Figure 3 Model 2 did not even need the B&B method to be proceeded; the solution was automatically found by one single LP relaxation The below Figure 3 explicitly illustrates the increase of optimal objective value by investigating the nodes.

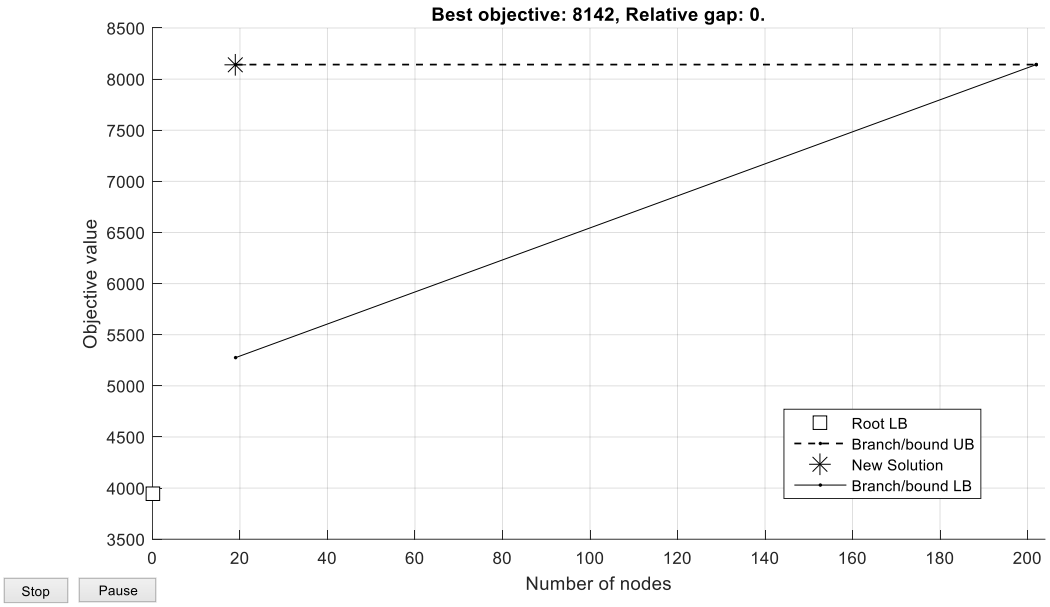


Figure 3: B&B Tree Nodes

According to 'Run and Time' function of MATLAB, the total time spent for the evalution of 10 employees is 3.754 seconds. Besides, the optimal solution is found in 202th node of the B&B tree for the Model I, as it can be seen in Figure 3.

In addition to sample application with 10 employees, computation times of more applications with different numbers of employees (ceteris paribus) are demonstrated in Table 3 to highlight the capabilities of the prepared code.

Table 3: Comparison Table via B&B Method

Number of Employees	Computation Time (seconds)	Number of Branching Node
10	3.754	202
15	4.442	1,524
20	6.013	10,186
25	175.534	923,649
30	937.598	4,259,723

As it can be seen in Table 3, the computation time tremendously boosts with the increase in number of employees. In addition, other elements of models would also affect the computation time. The application with higher number of employees are not carried out owing to the ineffectiveness of the method used by itself.

4.3. Sample Solution via Cutting-Plane Method

As applied in the B&B method, the same procedure is carried out to justify the efficiency of cutting-plane method for solving the problem using sample data. The intlinprog algorithm does not allow users to disable the use of B&B method, since the method is considered as the essential method of solving MILP problems. Thus, two algorithms can be applied in the solution procedure based on the solution speed that the software foresees beforehand. If the algorithm estimates that the use of branching would shorten the computation time, the algorithm can use it; otherwise, the algorithm directly applies cuts to the problem and yields the solution. As it can be seen in the below Figure 4, the software applied cut generations (cutting-plane method) to solve the problem owing to its rapidness.

The below Figure 4 clearly exhibits that the solution provided by the cutting-plane method is identical with the solution given by the B&B method. This situation does not have to hold for all applications due to the fact that different methods may give different solutions in the case of existence of the multiple optimal solutions. For instance, multiple optimal solution may occur when no employees are assigned to the highest or lowest salary

increase category. Some applications, such as the one mentioned in the instance may have multiple optimal solutions in which, for example, while one of the solutions produce X_1 as 8%, the other reveals 0% for the same variable. So, different results of different solution methods should also be accepted and the validity of both results should be investigated before the application of salary increases.

It should also be noted that enabling cutting-plane algorithm in the code expedited the computation speed by decreasing it to 2.834 seconds from 3.754 seconds. The computation times for different cases is also provided in the below Table 4.

Table 4: Comparison Table via Addition of Cutting-Plane Method

Number of Employees	Computation Time (seconds)	Number of Branching Node
10	2.834	-
15	3.272	5
20	3.324	2
25	3.261	3
30	3.011	1
50	4.362	4
100	37.310	8

Table 4 proves that enabling cutting-plane method in the solution procedure extremely decrease the computation time and provide an opportunity to users for evaluation of high number of employees.

4.4. Sample Solution via MATLAB's Heuristics Method

Designed as an assistive additional method by MATLAB, heuristics method is activated in addition to B&B and cutting-plane methods so as to measure its effectiveness on the sample problem. It should be noted that this method cannot solve an MILP by itself; on the contrary, heuristics method facilitates the solution procedure in some cases.

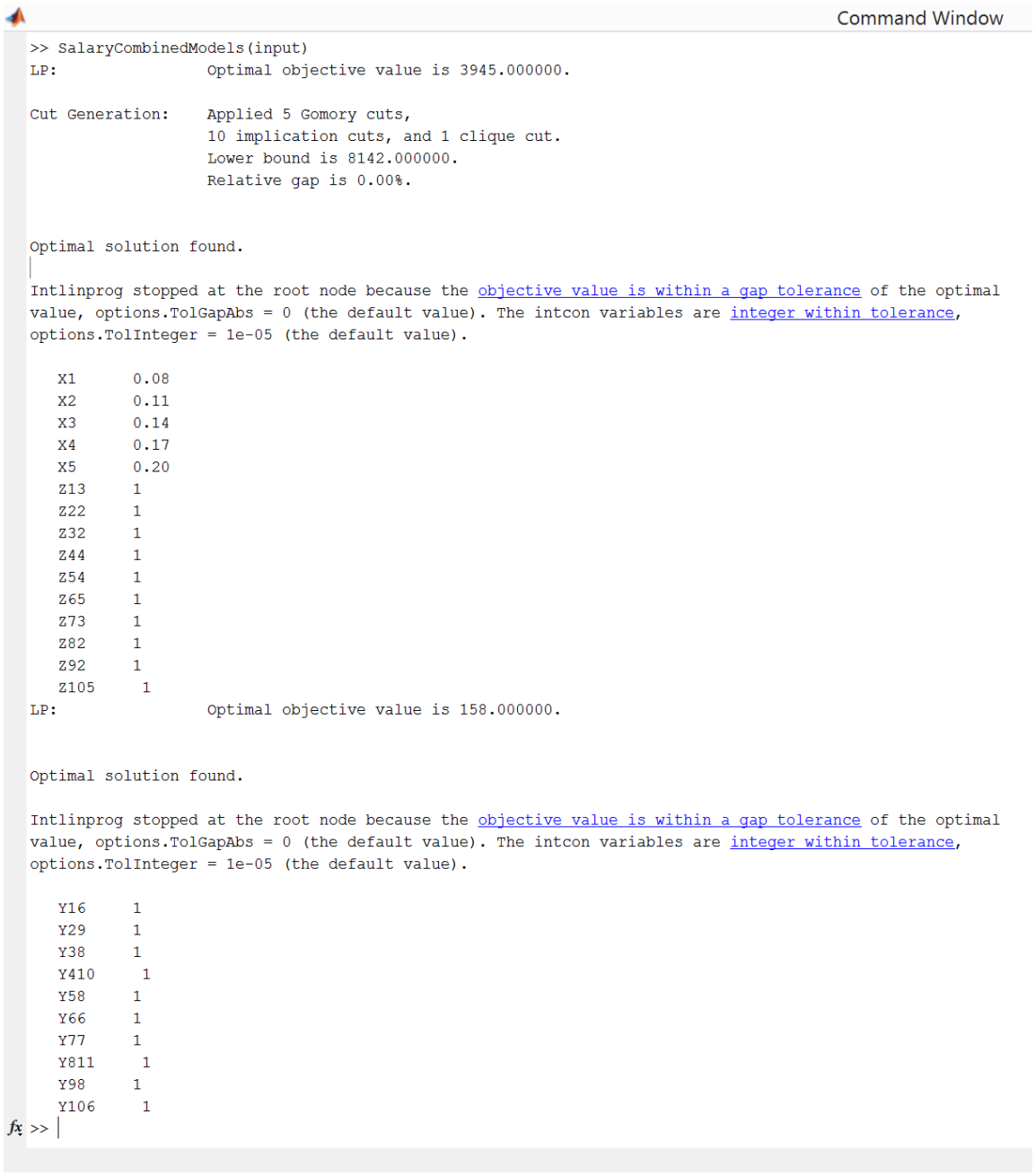


Figure 4: Sample Data Solution via Addition of Cutting-Plane Method

After activating the method and running the code for the sample data, the following results depicted in Figure 5 are obtained. It is apparent from the findings that this method also produced an identically same solution as other options.

The findings reveal that the same solutions with other two methods are found. The comparisons table of computation times with higher number of employees are given below in Table 5. From this table, it is apparent that heuristics option can ease solution process in some cases.

Table 5: Comparison Table via Addition of Heuristics Method

Number of Employees	Computation Time (seconds)	Number of Branching Nodes	Number of Heuristics Operations
10	2.742	-	-
15	2.816	-	1
20	2.920	2	-
25	3.357	2	-
30	3.021	-	1
50	3.959	-	1
100	36.794	-	1

4.5. Overall Commentary on Solution Methods

After carrying out many trial and error operations on a considerably high number of sample applications with sample data, it is understood that the default settings of the intlinprog algorithm works best in many cases. According to the default settings, B&B, cutting-plane, and heuristics are applied when necessities occur. However, it should not also be ignored that tuning these options can also help the computation time in some instances. By comparing computations times of three methods, it is aimed to show the most convenient settings for users.

In the MATLAB Optimization Toolbox User's Guide [26], much more options that can decrease the computation time for intlinprog are explained with details. So, it is crucial

to realize the fact that not only the solution options influence the computation time, but also there are other options that can affect the solution procedure of the algorithm.

As the main purpose of the thesis study was to focus on proposing a one-piece mathematical based computer programming tool to facilitate the decision process of human resources departments in business world, the computation time is not really considered as a substantial obstacle that needs to be overcame. The attention on the study should be paid to the presentation of a working code that conceives an easy computerizing approach to a complicated mathematics based salary administration guideline.

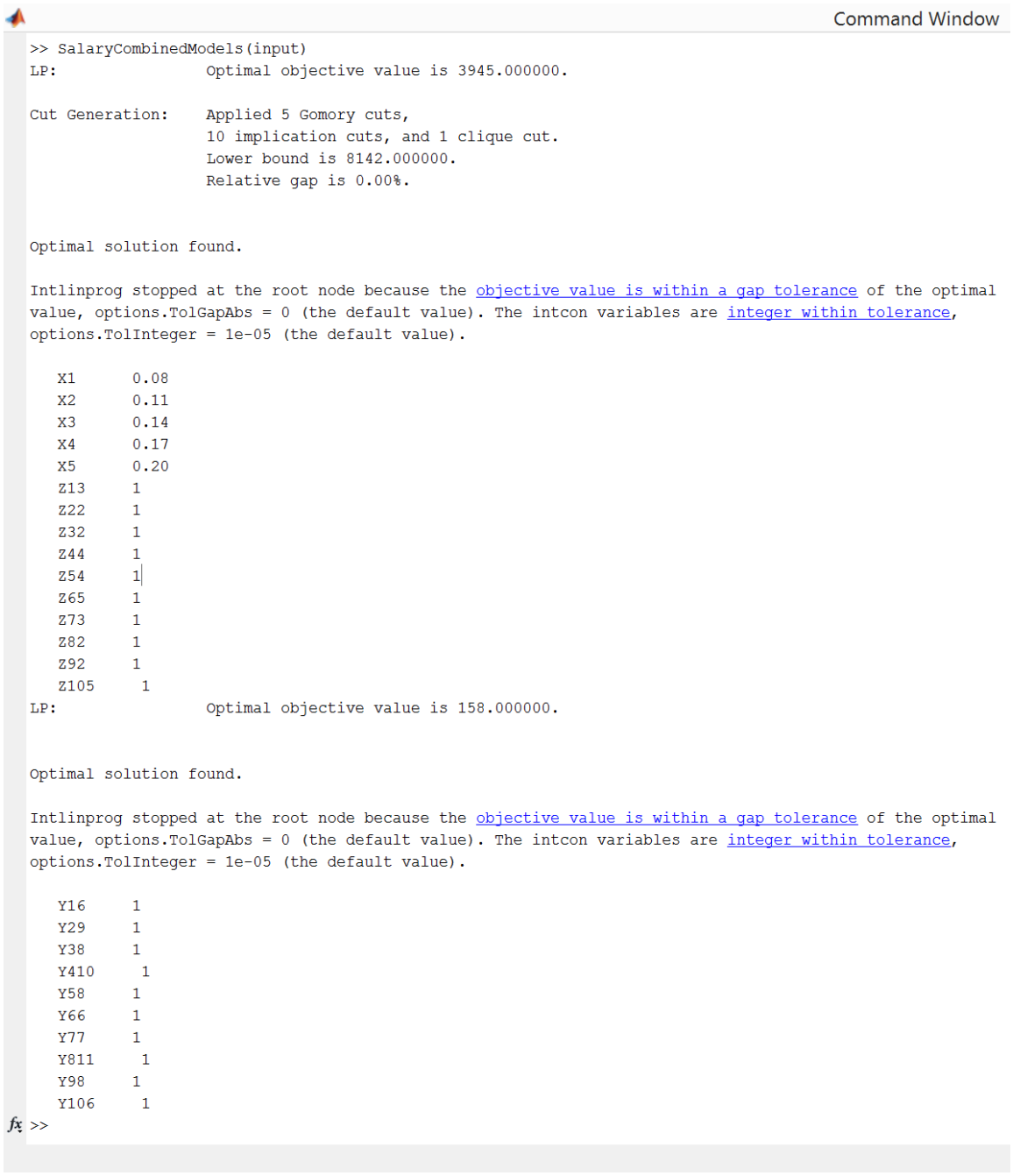


Figure 5: Sample Data Solution via Addition of Heuristics Method

CHAPTER V

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

5.1. Summary

This thesis study focused on a proposal of a computer based mathematical model that carried out requirements of a salary administration guideline to facilitate the executive personnel compensation decisions of HR departments. Based on the assumptions for salary increasing salaries of executive personnel, two existing mathematical models were both developed and coded in a MATLAB script for easier application in industries.

Starting with data collection procedure, following with computerization process, and concluding with a hypothetical application with several different solution methods, the salary administration guide is explained throughout the study. From the findings, the default (except for changing the RootLPAlgorithm into 'dual-simplex) options of MATLAB's intlinprog algorithm is advised as the most suitable way of application in real cases.

As this study involves assumptions that shape two mathematical models, the coding of MATLAB script is a special (also common) case. If a company uses some other criteria for evaluating their employees and these criteria are not reflected to both the models and the code, the proposed code becomes useless. So, additional criteria should be situated into mathematical formulations and then these formulations should be converted into coding in order to have a successful computer-based evaluation tool.

5.2. Conclusions

In this study, the importance of PFP based compensation systems and the role of potential evaluations in promotions are explained. Two mathematical models with their simplifications and linearization procedures are explained in detail. A sample code that can be applied in many firms based on its own assumption is provided. The adaptation procedure of mathematical models into the MATLAB software's solver is examined. A sample application is carried out by using the prepared MATLAB code. Furthermore, different types of solving methods and their corresponding responses were illustrated to mention the importance of different settings for computation effectiveness.

5.3. Recommendations

A future work on this topic can be creating an OR based mathematical model in order to create a better evaluation form based on specifications of companies that HR departments can use while evaluating employees. The OR perspective on human resources department has a capability of changing future of employee performance evaluations. Since one of the main goals of OR studies is to minimize costs and maximize profits, HR departments can have significant appeal in achieving this goal.

Another future work on improving the effectiveness of the proposed code is necessary. The computation time is highly related with the quality of coding. Limits of better quality in coding is undefined. A better code that decreases the computation time and yields more accurate solutions always exists in some minds. The further step of this study will be focused on new and innovative ways of computer programming in order to achieve the goal.

BIBLIOGRAPHY

1. Abowd, J.M., *Does performance-based managerial compensation affect corporate performance?* Industrial & Labor Relations Review, 1990. 43(3): p. 52S-73S.
2. Arye Bebchuk, L. and J.M. Fried, *Executive compensation as an agency problem.* The Journal of Economic Perspectives, 2003. 17(3): p. 71-92.
3. Baker, G.P., M.C. Jensen, and K.J. Murphy, *Compensation and incentives: Practice vs. theory.* The journal of Finance, 1988. 43(3): p. 593-616.
4. Bixby, R.E., *A brief history of linear and mixed-integer programming computation.*
5. Bruno, J.E., *Using linear programming salary evaluation models in collective bargaining negotiations with teacher unions.* Socio-Economic Planning Sciences, 1969. 3(2): p. 103-117.
6. Cadsby, C.B., F. Song, and F. Tapon, *Sorting and incentive effects of pay for performance: An experimental investigation.* Academy of management journal, 2007. 50(2): p. 387-405.
7. Charnes, A., W.W. Cooper, and R.O. Ferguson, *Optimal estimation of executive compensation by linear programming.* Management science, 1955. 1(2): p. 138-151.
8. Deming, W.E., *Out of Crisis, Centre for Advanced Engineering Study.* Massachusetts Institute of Technology, Cambridge, MA, 1986: p. 110.
9. Fabozzi, F.J. and A.W. Bachner, *Mathematical programming models to determine civil service salaries.* European Journal of Operational Research, 1979. 3(3): p. 190-198.
10. Garcia-Diaz, A., B. Flores, and R. Noce, *A computer based heuristic methodology for the development of salary administration guidelines.* Omega, 1996. 24(5): p. 583-595.
11. Garcia-Diaz, A. and G.L. Hogg, *A mathematical programming approach to salary administration.* Computers & Industrial Engineering, 1983. 7(1): p. 7-13.
12. Gerhart, B., S.L. Rynes, and I.S. Fulmer, *6 pay and performance: individuals, groups, and executives.* The Academy of Management Annals, 2009. 3(1): p. 251-315.
13. Gomory, R., *An algorithm for the mixed integer problem*. 1960, DTIC Document.

14. Howard, L.W. and J.L. Miller, *Fair pay for fair play: Estimating pay equity in professional baseball with data envelopment analysis.* Academy of Management Journal, 1993. 36(4): p. 882-894.
15. Kwak, N., T. Allen, and M. Schniederjans, *A multilevel salary compensation model using goal programming.* Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle, 1982. 16(1): p. 21-31.
16. Lal, R. and V. Srinivasan, *Compensation plans for single-and multi-product salesforces: An application of the Holmstrom-Milgrom model.* Management Science, 1993. 39(7): p. 777-793.
17. Land, A.H. and A.G. Doig, *An automatic method of solving discrete programming problems.* Econometrica: Journal of the Econometric Society, 1960: p. 497-520.
18. Lead-Deadwood, S. *Employee Performance Evaluation*. Retrieved August 28, 2016 from https://lead-deadwood.k12.sd.us/forms/classifiedeval.pdf:[
19. Loeb, J.W., *Hierarchical linear modeling in salary equity studies.* New Directions for Institutional Research, 2003. 2003(117): p. 69-96.
20. McCarl, B.A. and T.H. Spreen, *Applied mathematical programming using algebraic systems.* Cambridge, MA, 1997.
21. Milkovich, G., *Compensation*, J. Newman, Editor. 2008, Irwin McGraw Hill: Boston.
22. Millman, G.J. *Apple's Pay-For-Performance Plan Works There, Not Elsewhere*. Feb 27, 2015. The Wall Street Journal. Retrieved September 8, 2016 from http://blogs.wsj.com/riskandcompliance/2015/02/27/apples-pay-for-performance-formula-works-there-not-elsewhere/.
23. Ronen, B., M.A. Palley, and H.C. Lucas Jr, *Spreadsheet analysis and design.* Communications of the ACM, 1989. 32(1): p. 84-93.
24. Schrijver, A., *Theory of linear and integer programming*. 1998: John Wiley & Sons.
25. Sierksma, G., *Linear and integer programming: theory and practice*. 2001: CRC Press.

26. The MathWorks, I., *MATLAB Optimization Toolbox™ User's Guide (r2016a)*. Retrieved July 24, 2016 from http://uk.mathworks.com/help/pdf_doc/optim/optim_tb.pdf.
27. Winston, W.L. and J.B. Goldberg, *Operations research: applications and algorithms*. Vol. 3. 2004: Duxbury press Belmont, CA.

APPENDICES

A. Performance Measurement Guide

Retrieved from Lead-Deadwood School District's Website [25].

EMPLOYEE PERFORMANCE EVALUATION

DATE: _____

NAME: _____ JOB LOCATION: _____

JOB TITLE: _____ DATE OF LAST EVALUATION: _____

Please complete this form carefully and thoroughly. Remember its purpose is to:

Provide objective criteria for personnel performance evaluations on a standard basis within your organization.

Compel you to examine *all* of the individual traits affecting employee performance.

Help you to support your conclusion and recommendation for job classification and compensation improvements.

Produce fairer evaluations of employees.

PROCEDURE:

Pages 59 and 60 describe Fifteen personal traits identified with job success or failure.
Decide for each, the level at which the employee performed for this rating period.
Write the corresponding value number in the rating column. Add the numbers to obtain a total score.

Transfer this total to the rating scale on page 61. This will indicate, and support, your overall opinion of the employee's performance.

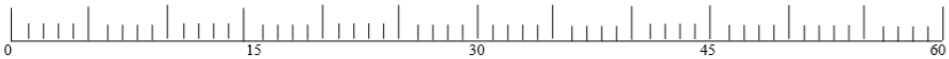
Refer back to pages 59 and 60 to comment on the employee's principal strengths and weaknesses. Your comments should be consistent with your rating of individual traits.

Finally, you should describe the employee's reaction to this evaluation, if you discuss it, and make your recommendation for any changes in the employee's job classification or rate of pay.

PERSONAL TRAITS		UNSATISFACTORY	SOME DEFICIENCIES EVIDENT
		0	1
KNOWLEDGE	The blending of job-related education, skills and experience	Severely lacking in knowledge.	Noticeable deficiencies in job knowledge.
QUANTITY	Level of satisfactory output generated per unit of time.	Usually below acceptable standard.	Barely acceptable level of output. A slow worker.
ACCURACY	Absence of errors	Constantly commits errors	Error level too high. Needs improvement.
JUDGMENT	Capacity to make reasonable decisions	Frequently makes irrational decisions. Poor judgment.	Too often selects wrong alternative.
INNOVATION	Imagination and creativity used to lower costs and improve profits.	Never offers a new procedure or new idea.	Rarely suggests new ideas.
APPEARANCE & HABITS	Personal habits, clothing and grooming (evaluation should consider the nature of the job).	Frequently offensive.	Occasionally sloppy appearance or display of offensive habits.
ORDERLINESS	Organization of the individual's work and work area.	Usually disorderly and chaotic.	Frequently unorganized or work area in disarray
COURTESY	Respect for feelings of others. Politeness on the job.	Frequently rude. Causes noticeable discomfort to others.	Occasionally impolite to coworkers or others.
COOPERATION	Willingness to help others accomplish their objectives	Usually uncooperative. A "roadblock" to coworkers, customers or suppliers.	Too often uncooperative when faced with reasonable requests for assistance.
INITIATIVE	Voluntary starting projects. Attempting non-routine jobs and tasks.	Shows little initiative. Never volunteers. Sticks closely to job routine.	Shows some initiative. Should do more without having to be told.
RELIABILITY	Dependability and trustworthiness.	Not reliable. Often fails to deliver a complete job.	Occasionally impolite to coworkers or others.
PERSEVERANCE	Steadfast pursuit of job objectives when faced with unexpected obstacles.	Frequently quits when faced with unexpected obstacles.	Is sometimes deterred by obstacles which should be overcome.
STABILITY	Even temperament. Acceptance of unavoidable tension and pressure.	Volatile, inconsistent personality. Disrupts work environment.	Occasional display of temper or emotion sufficient to disrupt other and hinder own performance.
ATTENDANCE		Frequent unexcused lateness or absence from work. Very poor attendance record.	Absences or lateness Below standards.
ALERTNESS	Ability to quickly understand new information and situations	Very slow to grasp ideas and events.	Usually needs extra instruction.

SATISFACTORY	EXCEPTIONAL	CLEARLY OUTSTANDING	INSERT NUMERICAL RATING (0 THROUGH 4)
2	3	4	
Understanding job routine. Some knowledge still to be aquired.	Completely understands all aspects of the job.	Understands why all job functions are performed and inter-relationship with other jobs. An expert.	
Satisfactory. Meets expectations of average output.	Usually exceeds the norm. A fast worker.	Exceptional producer. Generates maximal output.	
Makes average number of mistakes	Very accurate. Commits few errors.	Extremely accurate. Rarely commits an error.	
Usually exercises sound judgment.	Above average reasoning ability. Seldom errs in judgment.	Sustains high level of sound judgment. Decisions usually best under circumstances.	
Average number of suggestions for improving methods and procedures.	Often suggests beneficial changes and profit/cost improvements.	Very innovative. Constantly offers imaginative suggestions for improving operations.	
Usually properly dressed and groomed. Few poor personal habits	Rarely exhibits poor appearance or offensive habit.	Always properly dressed for the job. Personal habits are never offensive or in poor taste.	
Work sufficiently organized to efficiently perform the job.	Highly organized and efficient worker. Few instances of poor performance from lack of order.	Exceptionally precise in organizing work. Has immediate access to anything needed. Extremely efficient.	
Observes common courtesies, does not offend.	Very conscientious of other's feelings and rights. Always polite.	Extremely courteous, well mannered and polite. Always considers the comfort and ease of others.	
Generally a cooperative person on the job.	Very cooperative. Often offers assistance. Can usually be counted on to help.	Extremely cooperative. Constantly offers aid and always available to help others.	
Does not shirk. Voluntarily attempts to solve non-routine job problems as they occur.	Above average. A self starter. Will generally volunteer.	Places highest priority on getting things done. Constantly accepts difficult or unpleasant jobs to achieve goals.	
Can be relied on to complete all aspects of the job.	Completes work with little supervision. Will complete occasional special projects.	Extremely dependable and trustworthy. Accepts all assignments. Always performs as expected.	
Is not stopped by most obstacles, works through them.	Displays sufficient drive to overcome unusually difficult obstacles.	Always displays extreme determination. Will rarely quit until objective is reached.	
Even tempered. Absorbs routine pressures of job.	Can tolerate unusual pressure and tension without hindering performance.	Performs consistently and effectively under extreme pressure. Never visbly falters.	
Satisfactory attendance record.	Rarely late or absent.	Almost never late or absent. Always accepts overtime work, if offered.	
Understans most new ideas and developments without excessive explanation.	Fast learner. Grasps new information quickly.	Extremely bright. analyzes and understands with a minimum of instruction.	
		TOTAL **To Top of Page 4**	

Summary Score TOTAL (MARK TOTAL NUMERICAL RATING ON SCALE BELOW)



0 UNSATISFACTORY | 15 SOME DEFICIENCIES EVIDENT | 30 SATISFACTORY | 45 EXCEPTIONAL | 60 CLEARLY OUTSTANDING

Comment on principle strengths: _____

Comment on principle weaknesses and suggestions for improvement: _____

Has this evaluation been discussed with the employee? ☐ **Yes** ☐ **No**

Comments: _____

Your recommendation for present and future job classification: _____

RATED BY (Name and Title): _____

APPROVED BY: _____

Completion of this section by employee, is optional, and subject to the policy of your organization.	**I have reviewed this evaluation and I completely understand its contents.** **Date** _____ **Employee's signature**_____

Retrieved from Lead-Deadwood School District's Website [25].

B. Salary Administration Problem Solution Code

```
function [ ] = SalaryCombinedModels(input)
%SALARYOPT Summary of this function goes here
%Detailed explanation goes here

%%%PART 1: EMPLOYEE-SALARY PERCENTAGE ASSIGNMENTS BASED ON PERFORMANCE



%First, define number of employees and number of performance categories, i and
m, respectively

i = size(input,1);
m = max(input(:,5));

%Compute total number of variables
ps = 2*i + m + 2*i*m;

%Define parameters
C = input(:,1);
R = input(:,2);
U = input(:,3);
L = input(:,4);
E = input(:,5);
P = input(:,6);
a = zeros(m+1,1);
a(1) = -0.05;
a(2) = 0.20;
a(3:end) = -0.03;
q = 0.90;
d = 1;

%Create Column Vector 'b' which denotes right hand side of inequality
constraints
b = [(C - L) ; (-C  + P.*U + (1-P).*(L + q.*(U - L))) ; a; E; d-E;
zeros(i*m,1); zeros(i*m,1); ones(i*m,1) ];

%Create Column Vector 'beq' which denotes right hand side of equality
constraints
beq = [R-C ; ones(i,1)];

%Create Column Vector 'lb' which denotes lower bounds for all variables
```

```
lb = zeros(ps,1);

%Create Column Vector 'ub' which denotes upper bounds for all variables.
%Here, 'h' and 't' represent lambda and delta, respectively.
ubh = inf(i,1);
ubt = inf(i,1);
ubw = inf(i*m,1);
ubz = ones(i*m,1);
ubx = inf(m,1);
ub = [ubh; ubt; ubw; ubz; ubx];

%Create Column Vector 'f' which denotes coefficients of variables in
%objective function. Here, 'h' and 't' represent lambda and delta,
respectively.
fht = ones(2*i,1);
fwzx = zeros(2*i*m+m,1);
f = [fht; fwzx];

%Create Row Vector 'intcon' which denotes integer variables
intcon = linspace(2*i+i*m+1,2*i+i*m+i*m,i*m);

%Create Matrix 'Aeq' which denotes coefficients of variables in equality
constraints
%Here, 'h' and 't' represent lambda and delta, respectively.
Aeq = zeros(i*2,ps);

    %Constraint #30
for j = 1:i
    h = zeros(1,i);
    r = zeros(1,i);
    h(1,j) =  -1;
    r(1,j) = 1;
    W = zeros(1, i*m);
    W(1,(j-1)*m+1:(j-1)*m+m) = C(j);
    Z = zeros(1,i*m);
    X =  zeros(1,m);
    temp = [h, r, W, Z, X];
    Aeq(j,:)= temp;

    %Constraint #31
    h = zeros(1,i);
    r = zeros(1,i);
    W = zeros(1, i*m);
    Z = zeros(1, i*m);
```

```
    Z(1,(j-1)*m+1:(j-1)*m+m) = 1;
    X =  zeros(1,m);
    temp = [h, r, W, Z, X];
    Aeq(j+i,:)= temp;
end

%Create Matrix 'A' which denotes coefficients of variables in inequality
constraints
%Here, 'h' and 't' represent lambda and delta, respectively.
ss = i + i + m + 1 + i + i + (3*(i*m));
A =  zeros(ss,ps);

    %Constraint #32
for j = 1:i
    h = zeros(1,i);
    alpha = zeros(1,i);
    W = zeros(1, i*m);
    W(1,(j-1)*m+1:(j-1)*m+m) = -C(j);
    Z = zeros(1, i*m);
    X =  zeros(1,m);
    temp = [h, alpha, W, Z, X];
    A(j,:)= temp;

    %Constraint #33
    W = zeros(1, i*m);
    W(1,(j-1)*m+1:(j-1)*m+m) = C(j);
    temp = [h, alpha, W, Z, X];
    A(j+i,:)= temp;
end
ind=2*j;
row = zeros(1,ps);
row(1,end-m+1) = -1;
ind=ind+1;
A(ind,:)=row;

    %Constraint #34
row = zeros(1,ps);
row(1,end) = 1;
ind=ind+1;
A(ind,:)=row;
for j = 1:m-1
    ind = ind + 1;
    row = zeros(1,ps);
    row(1,end-m+j) = 1;
```

```
    row(1,end-m+j+1) = -1;
    A(ind,:) = row;
end

    %Constraint #35
for j = 1: i
    ind = ind + 1;
    h = zeros(1,i);
    r = zeros(1,i);
    W = zeros(1, i*m);
    Z = zeros(1, i*m);
    Z(1,(j-1)*m+1:(j-1)*m+m) = 1:m;
    X =  zeros(1,m);
    temp = [h, r, W, Z, X];
    A(ind,:)= temp;

end
    %Constraint #36
ind = ind+1;
A(ind:ind+i-1,:)=-1.*A(ind-i:ind-1,:);
    %Constraint #37
ind = ind + i;
A(ind:ind+(i*m)-1,:)=[zeros(i*m, i), zeros(i*m,i), eye(i*m), -1*eye(i*m),
zeros(i*m,m)];
    %Constraint #38
ind = ind + (i*m);
A(ind:ind+(i*m)-1,:)=[zeros(i*m, i), zeros(i*m,i), eye(i*m), zeros(i*m),
repmat(-1.*eye(m),i,1)];
    %Constraint #39
ind = ind + (i*m);
A(ind:ind+(i*m)-1,:)=[zeros(i*m, i), zeros(i*m,i), -1.*eye(i*m), eye(i*m),
repmat(1.*eye(m),i,1)];

    %Branch & Bound Solution Algorithm
options=optimoptions('intlinprog');
options =
optimoptions(@intlinprog,'OutputFcn',@savemilpsolutions,'PlotFcn',@optimplotmil
p);
options.BranchingRule='mostfractional';
options.CutGeneration='none';
options.Heuristics='none';
options.RootLPAlgorithm='dual-simplex';
[X,FVAL,EXITFLAG,OUTPUT] = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub,options);
Zvalues = round(X(2*i+i*m+1:2*i+i*m+i*m));
```

```
Xpercentages = (X((2*i+i*m+1+i*m):ps));
[Xtnr] = transpose(1:m);
Xtitle = [Xtnr(:), Xpercentages(:)];
fprintf('   X%d      %.2f\n',Xtitle.')
[Z1,Z2] = meshgrid(1:i,1:m);
Ztitle = [Z1(:),Z2(:),Zvalues(:)];
fprintf('   Z%d%d     %d\n',Ztitle(Ztitle(:,3)~=0,:).')

%%%PART 2: EMPLOYEE-INTERVAL ASSIGNMENTS BASED ON PERFORMANCE



%Define parameters
%Lower Bound for Time Intervals
Alb = 8;

%Upper Bound for Time Intervals
Bub = 12;

%Number of personnels allowed for assigning to any month
n = 14;

%Number of months since employee i has gotten the last increase
K = input(:,7);

%Define time periods, s, t, and beta, respectively.
s = 1:12;
alpha = Alb-K;
beta = Bub-K;

%Define Total Number of Variables
tnv = i*max(s)+2*max(s);

%Create Column Vector 'f2' which denotes coefficients of variables in
%objective function.
Ys = zeros(i*max(s), 1);
sigma = ones(max(s), 1);
omega = ones(max(s), 1);

f2 = [Ys; sigma; omega];
%Create Column Vector 'lb2' which denotes lower bounds for all variables
lb2 = zeros(tnv, 1);

%Create Column Vector 'ub2' which denotes upper bounds for all variables
```

```
Yss = ones(i*max(s), 1);
sigmaandomega = inf(2*max(s), 1);
ub2 = [Yss; sigmaandomega];

%Create Row Vector 'intcon2' which denotes integer variables
intcon2 = [1:i*max(s)];

%Create Matrix 'Aeq2' which denotes coefficients of variables in equality
constraints
Aeq2 = zeros(i*2,tnv);

    %Constraint #63
Cons2 = [repmat(eye(max(s)),1,i), -1*eye(max(s)), eye(max(s))];

    %Constraint #64
Cons3sub = zeros(i,i*max(s));
for nnn = 1:i
    Cons3sub(nnn,(nnn-1)*max(s)+1:nnn*max(s))=1;
end
Cons3 = [Cons3sub, zeros(i,max(s)),zeros(i,max(s))];
    %COnstraint #65
si=1;
for ii=1:m:length(Zvalues)
    scalar(si)=[1:m]*Zvalues(ii:ii+m-1);
    si=si+1;
end

beq2new=[];
Cons4=[];
for ii=1:i-1
    for jj=ii+1:i
        if scalar(ii)==scalar(jj)
            aeq2newrow=zeros(1,i*max(s));
            beq2new=[beq2new; K(jj)-K(ii)];
            months=alpha(ii):beta(ii);
            iis=ones(1,length(months))*ii;
            jjs=ones(1,length(months))*jj;
            yind=sub2ind([max(s) i],months,iis);
            aeq2newrow(yind)=months;
            months2=alpha(jj):beta(jj);
            yind2=sub2ind([max(s) i],months2,jjs);
            aeq2newrow(yind2)=-1*(months2);
            aeq2newrow=[aeq2newrow zeros(1,max(s)) zeros(1,max(s))];
            Cons4=[Cons4;aeq2newrow];
```

```
        end
    end
end

    %Constraint #5
b2cons5=[];
Cons5=[];
for ii=1:i-1
    for jj=ii+1:i
        if scalar(ii)>scalar(jj)
            a2newrow=zeros(1,i*max(s));
            b2cons5=[b2cons5; K(jj)-K(ii)];
            months5=alpha(ii):beta(ii);
            iis=ones(1,length(months5))*ii;
            jjs=ones(1,length(months5))*jj;
            yind3=sub2ind([max(s) i],months5,iis);
            a2newrow(yind3)=months5;
            months6=alpha(jj):beta(jj);
            yind4=sub2ind([max(s) i],months6,jjs);
            a2newrow(yind4)=-1*(months6);
            a2newrow=[a2newrow zeros(1,max(s)) zeros(1,max(s))];
            Cons5=[Cons5;a2newrow];
        end
    end
end
    %Constraint #6
b2cons6 = [];
Cons6 = [];
for ii=1:i-1
    for jj=ii++1:i
        if scalar(ii)<scalar(jj)
            a2cons6=zeros(1,i*max(s));
            b2cons6=[b2cons6; K(ii)-K(jj)];
            months01=alpha(ii):beta(ii);
            iiss=ones(1,length(months01))*ii;
            jjss=ones(1,length(months01))*jj;
            yind5=sub2ind([max(s) i],months01,iiss);
            a2cons6(yind5)=-1*months01;
            months02=alpha(jj):beta(jj);
            yind6=sub2ind([max(s) i],months02,jjss);
            a2cons6(yind6)=months02;
            a2cons6=[a2cons6 zeros(1,max(s)) zeros(1,max(s))];
            Cons6 = [Cons6;a2cons6];
        end
```

```
    end
end

si2=1;
for iii=1:m:length(Zvalues)
    scalar2(si2)=[1:m]*Zvalues(iii:iii+m-1);
    si2=si2+1;
end

    %Constraint #7
b2cons7 = [];
a2new = [];
const6=find(scalar2==min(scalar2));
for iii=1:length(const6)
    a2newrow5=zeros(1,i*12);
    monthss=alpha(iii):beta(iii);
    iisss=ones(1,length(monthss))*const6(iii);
    yind7=sub2ind([max(s) i],monthss,iisss);
    a2newrow5(yind7)=monthss;
    a2newrow5=[a2newrow5 zeros(1,max(s)) zeros(1,max(s))];
    a2new=[a2new; a2newrow5];
    b2cons7=[Bub-K(const6)];
end

    %Constraint #8
b2cons8 = [];
a2new2 = [];
const7=find(scalar==max(scalar));
for iiii=1:length(const7);
    a2newrow9=zeros(1,i*12);
    months=alpha(iiii):beta(iiii);
    iiis=ones(1,length(months))*const7(iiii);
    yind8=sub2ind([max(s) i],months,iiis);
    a2newrow9(yind8)=months;
    a2newrow9=[a2newrow9 zeros(1,max(s)) zeros(1,max(s))];
    a2new2=[a2new2; -1*a2newrow9];
    b2cons8=[K(const7)-Alb];
end

%Create Column Vector 'beq2' which denotes right hand side of equality
constraints
nbeq = ones(max(s), 1)*n;
beq2 = [nbeq; ones(i,1); beq2new];
```

```
%Combine all matrices to create the whole 'Aeq2' matrix
Aeq2 = [Cons2; Cons3; Cons4];

%Create 'A2' matrix
A2 = [Cons5; Cons6; a2new; a2new2];

%Create 'b2' matrix
b2 = [b2cons5; b2cons6; b2cons7; b2cons8];
options2=optimoptions('intlinprog');
options2.BranchingRule='mostfractional';
options2.CutGeneration='none';
options2.Heuristics='none';
options2.RootLPAlgorithm='primal-simplex';
[Y,FVAL2,EXITFLAG2,OUTPUT2] =
intlinprog(f2,intcon2,A2,b2,Aeq2,beq2,lb2,ub2,options2);
Yvalues = round(Y(1:i*max(s)));
[Y1,Y2] = meshgrid(1:i,1:max(s));
Ytitle = [Y1(:),Y2(:),Yvalues(:)];
fprintf('   Y%d%d     %d\n',Ytitle(Ytitle(:,3)~=0,:).')
```

Published with MATLAB® R2015b

VITA

Taner Cokyasar was born on October 16th 1990 in Adana, Turkey. He obtained his Bachelor of Science degree in Business Administration from Nevsehir University, Nevsehir, Turkey in 2012. After his graduation, Taner Cokyasar worked for Kuwait-Turkish Participation Bank Inc. as a Retail & Small Business Sales Associate for 2 years. During his banking experience, Taner was awarded with a scholarship that allows him pursuing his graduate career abroad by the Turkish Ministry of National Education. Taner Cokyasar attended the University of Tennessee in 2015 to pursue his Master of Science degree in Industrial Engineering.